Package ‘BayesianFROC’
April 27, 2020

Type Package
Title FROC Analysis by Bayesian Approaches
Version 0.2.3
Maintainer Issei Tsunoda <tsunoda.issei1111@gmail.com>

Description
Execute BayesianFROC::fit_GUI_Shiny() (or fit_GUI_Shiny_MRMC()) for a graphical user interface via Shiny. The free-response receiver operating characteristic (FROC) method is a generalization of receiver operating characteristic (ROC) analysis. However, Chakraborty's classical model is non-generative in the sense that it cannot synthesize data. This package aims to modify his models to be generative using a Bayesian approach, and to verify that our models fit practical datasets. In signal detection theory, the number of true positives never exceeds the number of targets. However, this is not explained by any existing model. Thus, in this package, the author contributes to FROC theory by refining Chakraborty’s model to obtain models that are generative. This modification allows us to use FROC analysis in a general statistical scheme, and as a benefit, our generative model can be applied to calculations of posterior predictive p values that require generation of synthetic datasets from fitted models. Furthermore, this package presents new models for comparison of modalities. Modality comparison is a common problem in radiology, and has been studied extensively. However, in many medical studies, such problems are addressed with non-Bayesian methods such as ANOVA. As a supplementary topic, this work presents a Bayesian model that includes individual differences. With this model, we can account for differences between individual readers when comparing modalities, using Bayesian rather than ML-methods. The author found the existing FROC model in [1] to be non-generative for calculation of posterior predictive p values. Replacing the ML-based method with a Bayesian approach differs from standard practice but provides insight into the problems of existing methods. Please execute the following R scripts from the R (R studio) console, demo(demo_MRMC, package = ”BayesianFROC”); demo(demo_src, package = ”BayesianFROC”); demo(demo_stan, package = ”BayesianFROC”); demo(demo_drawcurves_src, package = ”BayesianFROC”); demo_Bayesian_FROC(); demo_Bayesian_FROC_without_pause(). References: [1] Dev Chakraborty (1989) <doi:10.1118/1.596358> Maximum likelihood analysis of free - response receiver operating characteristic (FROC) data. Pre-print: Issei Tsunoda; Generative Models for free-response receiver operating characteristic analysis. See the vignettes for more details.

License MIT + file LICENSE
Encoding UTF-8
LazyData true

RoxygenNote 7.1.0

Imports knitr, readxl, xlsx, stats, graphics, tcltk, grDevices,
ggplot2, methods, car, crayon, DiagrammeR, bridgesampling,
rhandsontable, shiny, praca, shinydashboard, shinythemes

Suggests openxlsx, hexbin, MASS, ggmcmc, magrittr

Depends rstan (>= 2.18.2), R (>= 3.5.0), Rcpp

NeedsCompilation yes

VignetteBuilder knitr

Collate 'AFROC.R' 'Author_vs_Chakraborty_for_AUC.R' 'BayesianFROC.R'
  'Close_all_graphic_devices.R' 'ConfirmConvergence.R'
  'CoronaVirus_Disease_2019.R' 'DrawCurves.R'
  'Draw_an_area_of_AUC_for_srs.R' 'FROC_via_ggplot.R'
  'Make_TeX_file_for_summary.R' 'Phi_and_Phi_inv.R' 'QQQ.R'
  'Profile.R' 'Simulation_Based_Calibration.R'
  'Stan_model_minimal_incomplete.R' 'StartupMessage.R'
  'StatisticForANOVA.R'
  'Test_Null_Hypothesis_that_all_modalities_are_same.R'
  'apply_foo.R' 'array_easy_example.R'
  'array_of_hit_and_false_alarms_from_vector.R'
  'check_hit_is_less_than_NL.R' 'check_rhat.R'
  'chi_square_goodness_of_fit.R' 'clearWorkspace.R' 'color.R'
  'convertFromJafroc.R' 'create_dataset.R' 'css.shiny.R'
  'dark_theme.R' 'dataset_creator_by_specifying_only_M_Q.R'
  'dataset_creator_for_many_Readers.R'
  'dataset_creator_new_version.R' 'demo_Bayesian_FROC.R'
  'demo_Bayesian_FROC_without_pause.R'
  'development_Tools_and_Memorandum.R' 'document_dataset_MRMC.R'
  'document_dataset_srs.R' 'document_true_param.R'
  'draw_latent_distribution.R' 'empty_cell_shiny.R'
  'error_message.R'
  'error_message_on_imaging_device_rhat_values.R' 'ex.R'
  'explanation_about_package_BayesianFROC.R'
  'explanation_for_what_curves_are_drawn.R'
  'extract_EAP_by_array.R'
  'extract_data.frame_from_dataList_MRMC.R' 'fffaaabbb.R'
  'fit_Bayesian_FROC.R' 'fit_GUI.R' 'fit_GUI_MRMC.R'
  'fit_GUI_MRMC_new.R' 'fit_GUI_Shiny.R' 'fit_GUI_dashboard.R'
  'fit_GUI_simple_from_appppp_file.R' 'fit_MRMC_versionTWO.R'
  'foo_of_a_list_of_arrays.R' 'fut_GUI_MRMC.shiny.R'
  'get_posterior_variance.R' 'give_name_srs_data.R'
  'hit_rate_adjusted_from_the_vector_p.R'
  'initial_values_specification_for_stan_in_case_of_MRMC.R'
  'install_imports.R' 'layout.R' 'm_q_c_vector_from_M_Q_C.R'
  'make_true_parameter_MRMC.R' 'metadata.R' 'method.R'
  'stanfitExtended.R' 'methods.R' 'methods_print.R'
'minimal_model_MRMC.R' 'minimal_model_MRMC2.R'
'minimal_model_MRMC2_to_check_causes.R' 'minimal_model_MRMC3.R'
'minimal_model_MRMC_development.R' 'modelComparison.R'
'multiple_diget_plot.R' 'operator.R'
'p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit.R'
'pairs_plot_if_divergent_transition_occurred.R' 'pause.R'
'plotFROC.R' 'plot_FPF_and_TPF_from_a_dataset.R'
'plot_curve_and_hit_rate_and_false_rate_simultaneously.R'
'pnorm_or_qnorm.R' 'ppp.R' 'print_minimal_reproducible_code.R'
'priorResearch.R' 'prior_predictor.R' 'prior_print.R'
'save_an_R_object.R' 'sbcVer2.R' 'sbc_MRMC.R' 'sbc_new.R'
'showGraphicalModel.R' 'show_codes_in_my_manuscript.R'
'size_of_return_value.R' 'small_margin.R'
'snippet_for_BayesianFROC.R' 'sortAUC.R'
'stability_of_AUC_ranking_in_case_of_MRMC_data.R'
'summarise_MRMC.R' 'summary_AUC_comparation_MRMC.R'
'summary_EAP_CI_srcsr.R' 'the_row_number_of_logical_vector.R'
'validation_MRCM_Create_dataList_MRMC_Hit_from_rate_etc.R'
'validation_MRCM_UNDER_CONSTRUCTION.R'
'validation_error_srcsr.R' 'viewdata.R' 'waic.R'
'without_double_quote.R'

Author  Issei Tsunoda [aut, cre]
Repository  CRAN
Date/Publication  2020-04-27 14:00:07 UTC

R topics documented:

  AFROC ............................................................. 7
  AFROC_curve .................................................... 8
  array_easy_example ............................................. 9
  array_of_hit_and_false_alarms_from_vector ................... 10
  Author_vs_classic_for_AUC .................................... 12
  BayesianFROC .................................................. 13
  check_hit_is_less_than_NL ................................... 35
  check_rhat ..................................................... 38
  chi_square_at_replicated_data_and_MCMC_samples_MRMC ........ 39
  chi_square_goodness_of_fit .................................. 42
  chi_square_goodness_of_fit_from_input_all_param ........... 46
  chi_square_goodness_of_fit_from_input_all_param_MRMC ...... 50
  Chi_square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean ... 54
  clearWorkspace ............................................... 58
  Close_all_graphic_devices ................................... 59
  compare ....................................................... 59
  comparison .................................................... 60
  ConfirmConvergence ........................................... 61
  Confirm_hit_rates_are_correctly_made_in_case_of_MRMC ...... 63
  convertFromJafroc ............................................. 64
CoronaVirus_Disease_2019 .................................................. 71
CoronaVirus_Disease_2019_prevalence .................................. 73
create_dataList_MRMC ..................................................... 75
create_dataset ............................................................. 81
Credible_Interval_for_curve .............................................. 82
d ................................................................. 86
dark_theme ............................................................... 86
data.bad.fit ............................................................. 87
data.hier.ficitious ....................................................... 89
data.MultiReaderMultiModality ........................................ 90
data.nonconverge.srsc .................................................. 90
data.SingleReaderSingleModality ...................................... 92
dataList.Chakra.1 ....................................................... 92
dataList.Chakra.1.with.explanation .................................... 94
dataList.Chakra.2 ....................................................... 96
dataList.Chakra.3 ....................................................... 97
dataList.Chakra.4 ....................................................... 99
dataList.Chakra.Web .................................................... 101
dataList.Chakra.Web.orderd ............................................. 104
dataList.divergent.transition.in.case.of.srsc ......................... 107
dataList.High .......................................................... 109
dataList.high.ability .................................................... 109
dataList.Low ........................................................... 110
dataList.Low.ability ..................................................... 110
dataList.one.modality ................................................... 111
dataset_creator_by_specifying_only_M_Q .............................. 111
dataset_creator_for_many_Readers .................................... 113
dataset_creator_new_version ........................................... 114
data_2modaities_2readers_3confidence ............................... 115
data_of_36_readers_and_a_single_modality .......................... 116
dd ................................................................. 122
dd.orderd .............................................................. 126
ddd ................................................................. 130
dddd ................................................................. 132
ddddd ............................................................... 135
dddddd ............................................................. 137
ddddddd ........................................................... 139
dddddddd .......................................................... 141
demo_Bayesian_FROC .................................................. 142
demo_Bayesian_FROC_without_pause ................................ 142
draw.CFP.CTP.from.dataList ........................................... 142
DrawCurves .............................................................. 147
DrawCurves_MRMC ....................................................... 154
DrawCurves_MRMC_pairwise ........................................... 155
DrawCurves_MRMC_pairwise_BlackWhite ............................. 158
DrawCurves_MRMC_pairwise_col ..................................... 159
DrawCurves_ssrc ....................................................... 160
Draw_an_area_of_AUC_for_srsc ....................................... 162
Draw_AUC ............................................................. 162
R topics documented:

Draw_a_prior_sample .................................................. 164
Draw_a_simulated_data_set .......................................... 165
Draw_a_simulated_data_set_and_Draw_posterior_samples ............ 168
draw_latent_noise_distribution ..................................... 171
draw_latent_signal_distribution .................................... 174
dz ................................................................. 178
Empirical_FROC_via_ggplot .......................................... 178
error_message .......................................................... 182
error_message_on_imaging_device_rhat_values ...................... 183
error_MRMC ............................................................ 185
error_ssrc ............................................................. 188
error_ssrc_error_visualization ............................... 196
error_ssrc_variance_visualization .......................... 199
explanation_about_package_BayesianFROC ......................... 200
explanation_for_what_curves_are_drawn ........................ 200
extractAUC .............................................................. 201
extract_data_frame_from_dataList_MRMC ......................... 202
extract_EAP_by_array ............................................... 203
extract_EAP_CI ......................................................... 206
extract_estimates_MRMC .............................................. 209
extract_parameters_from_replicated_models .................... 210
false_and_its_rate_creator ........................................ 213
false_and_its_rate_creator_MRMC .................................. 217
fffaaabbb ............................................................. 220
fit_a_model_to ........................................................ 220
fit_Bayesian_FROC ................................................... 225
fit_GUI ................................................................. 253
fit_GUI_dashboard ..................................................... 254
fit_GUI_MRMC .......................................................... 256
fit_GUI_MRMC_new ..................................................... 257
fit_GUI_Shiny ........................................................... 258
fit_GUI_Shiny_MRMC ..................................................... 261
fit_GUI_simple_from_appnp_file .................................. 264
fit_MRMC .............................................................. 265
fit_MRMC_versionTWO .................................................. 274
fit_Null_hypothesis_model_to .................................... 279
fit_ssrc ............................................................... 283
flatnames .............................................................. 289
flat_one_par .......................................................... 290
foo ................................................................. 291
foo00 ............................................................... 291
foo_of_a_List_of_Arrays ............................................. 292
FROC_curve ............................................................ 293
from_array_to_vector .............................................. 294
get_posterior_variance .............................................. 295
get_samples_from_Posterior_Predictive_distribution .............. 296
ggplotFROC ........................................................... 299
ggplotFROC.EAP ......................................................... 302
R topics documented:

give_name_srsc_CFP_CTP_vector ........................................... 306

give_name_srsc_data ......................................................... 309

hits_creator_from_rate ...................................................... 313

hits_false_alarms_creator_from_thresholds .............................. 317

hits_from_thresholds .......................................................... 322

hits_rate_creator ............................................................. 323

hit_rate_adjusted_from_the_vector_p .................................... 324

initial_values_specification_for_stan_in_case_of_MRC .................. 326

install_imports ................................................................. 329

inv_Phi ............................................................ 330

make_TeX ............................................................ 331

make_true_parameter_MRC .................................................. 331

metadata_srsc_per_image ................................................... 332

metadata_to_DrawCurve_MRC ................................................ 335

metadata_to_fit_MRC .......................................................... 335

mu ............................................................................. 341

mu_truth ............................................................... 342

mu_truth_creator_for_many_readers_MRC_data ............................ 343

m_q_c_vector_from_M_Q_C ................................................... 345

p ............................................................................. 347

pairs_plot_if_divergent_transition_occurred ............................ 347

pause ............................................................... 348

Phi ............................................................... 349

Phi_inv ............................................................... 350

plot,stanfitExtended,missing-method ...................................... 351

plotFROC ............................................................. 352

plot_curve_and_hit_rate_and_false_rate_simultaneously ............ 353

plot_empirical_FROC_curves .............................................. 355

plot_FPF_and_TPF_from_a_dataset ........................................ 359

plot_FPF_TPF_via_dataframe_with_split_factor ......................... 365

plot_test .............................................................. 371

pnorm_or_qnorm .......................................................... 371

ppp .............................................................. 372

ppp_MRC .............................................................. 375

ppp_srsc ............................................................... 377

print,stanfitExtended-method .............................................. 382

print_minimal_reproducible_code_in_case_of_MRC ...................... 384

print_stanfitExtended ..................................................... 385

priorResearch .............................................................. 385

prior_predictor ............................................................ 386

prior_print_MRC .......................................................... 387

prior_print_srsc .......................................................... 387

p_truth ............................................................... 388

p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit .... 388

rank_statistics_with_two_parameters ................................... 392

replicate_model_MRC ....................................................... 393

replicate_MRC_dataList .................................................... 396

sbcc ............................................................... 399
An AFROC curve is a plane curve characterized by two real numbers denoted by \( a, b \). In the following, \( \Phi() \) denotes the cumulative distribution function on the standard Gaussian distribution.

The so-called AFROC curve is defined by
\[(\xi(t), \eta(t)) = (1 - e^{-t}, \Phi(b\Phi^{-1}(\exp(-t)) - a))\]

for all \(t > 0\).

By specifying two real numbers \(a\) and \(b\), we can plot an AFROC curve.

The area under the AFROC curve, or briefly AUC, is calculated as follows, which are used to evaluate how physicians detect lesions in radiographs.

\[AUC = \frac{a}{\sqrt{1 + b^2}}.\]

Usage

\[\text{AFROC}(x, a = 0.14, b = 0.19, x\text{.coordinate.also} = \text{FALSE})\]

Arguments

- \(x\): A real number which moves in the domain of FROC curve
- \(a, b\): One of the parameter of model which characterize AFROC curve
- \(x\text{.coordinate.also}\): Logical, whether a vector of 1−\(\exp(-x)\) is included in a return value.

Value

if \(x\text{.coordinate.also} = \text{TRUE}\), then A list, contains two vectors of \(x, y\) coordinates for drawing curves. If \(x\text{.coordinate.also} = \text{FALSE}\), then return is a vector, consisting of \(y\) coordinate only, \((x\) coordinates is omitted.)

Examples

\[
x \leftarrow \text{stats}::\text{runif}(1000, 0.001, 100)
\]

\[
a \leftarrow \text{AFROC}(x, x\text{.coordinate.also} = \text{TRUE})
\]

\[
\text{plot}(a\$x, a\$y)
\]
Arguments

x  A real number which moves in the domain of FROC curve
a  a generated parameter of model which characterize AFROC curve
b  a generated parameter of model which characterize AFROC curve

Details

Technique of plotting AFROC is difficult because it has two points in which the gradients are infinity and it causes the following warinings. Revised 2019 Nov. 20

Warning messages: 1: In stats::qnorm(exp(1 - x)) : NaNs produced 2: In stats::qnorm(exp(1 - x)) : NaNs produced 3: Removed 50 rows containing missing values (geom_path).

Value

none

Examples

# This function is under construction.
x <- runif(1000,1,10)
y <- AFROC_curve(x)
plot(x,y)

array_easy_example

Example array

Description

Make a three dim array whose component is its index. For example

a[2,3,4] = 234

Usage

array_easy_example(I = 2, J = 3, K = 4)

Arguments

I  natural number less than 10
J  natural number less than 10
K  natural number less than 10

Value

An array of three dimension.
array_of_hit_and_false_alarms_from_vector

Array of hits and false alarms; 2019 Jun 18

Description

Return value is a three dimensional array of type $[C,M,Q]$ representing the number of confidence levels and modalities and readers, respectively. This array includes the number of hit and the number of false alarms.

Revised 2019 Nov. 20

Usage

array_of_hit_and_false_alarms_from_vector(dataList)

Arguments

dataList A list, consisting of the following R objects: m, q, c, h, f, NL, C, M, Q each of which means from the right
m : A vector, indicating the modality ID = 1, 2, ... which does not include zero.
q : A vector, indicating the reader ID = 1, 2, ... which does not include zero.
c : A vector, indicating the confidence = 1, 2, ... which does not include zero.
h : A vector, indicating the number of hits
f : A vector, indicating the number of false alarm
NL : An positive integer, indicating the number of lesions for all images
C : An positive integer, indicating the highest number of confidence level
M : An positive integer, indicating the number of modalities
Q : An positive integer, indicating the number of readers.
The detail of these dataset, please see the example datasets, e.g. dd.

Details

The author also implemented this in the metadata_to_fit_MRMC which is an old version. However, the old version uses "for" sentences, and it is not so better. On the other hand, this function use the function aperm() and array() and they are better than "for" sentence.

Revised 2019 Nov. 20 Revised 2019 Dec. 12

Examples

a <- array_example(2,3,4)
**Value**

A list, whose components are arrays of the number of hits \( h \) and the number of false alarms \( f \) of dimension \([c,M,Q]\). Do not confuse \([c,Q,M]\) or \([M,Q,C]\), etc. Revised 2019 Nov. 20

**See Also**

[Chi_square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean](#)

**Examples**

```r
# Validation of program

h1 <- array_of_hit_and_false_alarms_from_vector(dd)$harray
h2 <- metadata_to_fit_MRMC(dd)$harray
h1 == h2

f1 <- array_of_hit_and_false_alarms_from_vector(dd)$farray
f2 <- metadata_to_fit_MRMC(dd)$farray
f1 == f2

# subtraction for (hit - hit.expected)

# In the chi square calculation,
# we need to subtract expected value of hit from hit rate,
# thus the author made this function.

## Not run:

# Prepare example data

dd <- BayesianFROC::dd

# Fit a model to data

fit <- fit_Bayesian_FROC(dataList = dd, ite = 1111)

# Extract a collection of expected hits as an array
```
harray.expected <- extract_EAP_by_array(fit, ppp)*dd$NL

# Prepare hit (TP) data as an array

harray <- array_of_hit_and_false_alarms_from_vector(dd)$harray

# Calculate the difference of hits and its expectation..

Difference <- harray - harray.expected

# The above calculation is required in the chi square goodness of fit

#=============================================
# array format hit and false
#=============================================

harray <- array_of_hit_and_false_alarms_from_vector(dataList = ddd)$harray
farray <- array_of_hit_and_false_alarms_from_vector(dataList = ddd)$farray

## End(Not run)

Author_vs_classic_for_AUC

validation of AUC calculation
BayesianFROC

Description
This is for the author.

Usage
Author_vs_classic_for_AUC(StanS4class)

Arguments
StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc

Value
AUCs

Author(s)
Issei Tsunoda

Description
The crucial difference of the author’s model and the classical Chakraborty’s model is that the former is a ‘generative’ model but the later is “not.” In the general theory of Statistic, the fake data synthesized from models are important. Thus, the author made such generative models and implements these models in the package.

The following two Shiny based GUI are available.

  fit_GUI_Shiny()  GUI for a single reader and single modality
  fit_GUI_Shiny_MRMC()  GUI for multiple readers and multiple modalities

The aim of FROC analysis is to compare imaging modalities, which are imaging methods such as MRI, CT, PET, etc. We want to find an imaging method with which we can find more many lesions in radiographs.

To investigate modality comparison, we have to do a trial in order to obtain a dataset consisting of TP and FP.
Details

Here is what this package implements.

Overview of FROC analysis

In general data-analysis such as generalized linear models, the data can be plotted such as scatter plot, and we fit a model to the data such that the model can be visualized as an expected curve of data. And we can check how model fits to data intuitively. This procedure is available in the FROC paradigm. First, FROC data are plotted as scatter plot, each point is a pair of the so-called false positive fraction (FPF) and true positive fraction (TPF). And the fitted curve to this scatter plot is called FROC curve. However, the FROC curve has an infinite area under the curve (AUC), thus we modify the curve so that the AUC of modified curve has finite AUC, more precisely between zero and one. The modified curve is called AFROC curve. Using the AUC of AFROC curve, we evaluate the observer performance. Namely, high AUC means physicians can find more lesions in x-ray films.

To compare imaging modalities such as MRI, CT, PET, etc, we do a trial from which Data arise and we fit a model to the data. Using the resulting model, we can compare modalities or evaluate the observer performance based on AUC.

In the sequel, we give a complete description about the following three terms.

Trial from which data arise.

Data consist of the number of TPs and FPs.

Modeling calculates the probability law in which data (TPs and FPs) arise

To introduce FROC trial, let us consider the following terms.

A reader (in other words, player) who is a physician or radiologist challenges to find lesions (in other words, it is called signals, targets, nodules, ...) from radiographs.

Images (in other words, radiographs, x-ray films such as CT, MRI, PET, etc.) containing shadows (not necessarily caused by lesions). We assume that $N_l$ lesions make shadows as targets. (Note that each image can contain one more lesions and this multiple signals for a single image distinct FROC trial from the ordinal ROC trial). The number of images are denoted by $N_I$.

A researcher (in other words, data-analyst) knows true lesion locations (signal) and she can count reader’s True Positives and False Positives after his lesion finding task.

For the sake of simplicity, we consider a single reader.

Throughout this explanation, we follow the convention that readers are male and the researcher is female. So, "he" means the reader, and "she" means a data-analyst.

FROC trial and data

The following table is a dataset to be fitted a model.

Let us see how it arises.

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms (FP:False Positive)</th>
<th>No. of hits (TP:True Positive)</th>
</tr>
</thead>
</table>
Suppose that Bob is a reader (physician) and Alice is a researcher (Data-analyst).

A. "Hi, Bob."

B. "Hi, Alice"

A. "Now, there are $NI$ radiographs."

B. "What are you gonna do today?"

A. "Ahem, now, I evaluate your observer performance ability, namely ability of finding lesions from radiographs."

B. "Seriously? Duh..."

He was disappointed because he wanted to yada yada yada with her.

A. "Find the tumors from these images and I check your answer, by assigning a true positive or a false positive to your answer."

Alice gave Bob the first image (radiograph).

B. "OK! Let's start. Hmmm ... Hmmm... I think the first image contains two suspicious tumors."

A. "Locarize your two suspicious tumors locations in the first image."

She gave him a pen.

B. "OK! ... Swish, Swish"

He marked two locations in the first image.

A. "In addition, assign your confidence levels to your two suspicious tumors."

B. "How?"

A. "It is a number, 1,2,3,4,5. If you think a shadow is definitely tumor, then you choose 5. Similary, 4 is probably, ...., 2 is subtle, 1 is very subtle.

B. "OK! Now, I doubt two shadows are tumors, I think that one is absolutely tumor, so I rate 5 for this shadow. On the other hand, for the another shadow, I think that it is probably a tumor, so I rate 3 for it."

Swish, Swish, He rated for his two suspicious locations.

A. "Let's check your answer for the first image! The first suspicous tumor with rating 5 is correctly locarized."

B. "I did it! Yay! Hooray!! Woohoo!!! Booyah!!!!"

A. "But the second suspicous shadow locarized with rating 3 is not correct, so,..., it is not a tumor."

B. "Oops, I did it."
A. "In the first image, there are several tumors being not detected and we ignore them in this FROC trial."
B. "Oopsies. Gah!"
A. "So, now, you have one hit with rating 5 and one false alarm with rating 3. Next, we will work for the second images."
Alice gave Bob the second image (radiograph).
B. "In the second image, I think there are three suspicious shadows."
A. "OK, localize your suspicious locations."
B. "Swish Swish Swish"
Bob localized his three suspicious locations.
A. "OK, rate your confidence level for each localized shadow."
B. "The first shadows is 3, the second shadow is 5, the third shadows is 2."
A. "OK, I check your answer. So, the answer is true, true, false."
B. "Uh-huh, .... mm hm"
A. "So, in the second image you have one hits with confidence level 3 and one hits with rating 5 and one false alarm with rating2. Combining the first image and the second image, now, you have two hits with rating 5, and one hit with rating 3, and one false alarm with rating 2 and one false alarm with rating 3. Next, we consider the third image."
Alice and Bob did this trial for all images, and they summarized the number of hits and false alarms in the following table.

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
</table>

A. "Phew, I summarized the evaluation in the following table"
B. "How kind of you!"
A. "Phew, you are finished for the day. Sayonara, Bob!"
B. "Boo!"
He was impatient because, today, he wanted to yada yada yada with her.
B. "Hey, Alice"
A. "!?"
B. "Hey, I am done at work now, so I am free to yada yada yada with you today!!"
A. "Eww, today, I cannot, cuz I have to fit a FROC model to the data and draw a fitted FROC curve and calculate AUC to evaluate your observer performance ability!"

B. "Ugh..., Duh ...."

Unfortunately, Bob's yada yada plan was a complete failure.

1. First trial start  The researcher gives the reader the first image which contains suspicious shadows, each of which is noise or lesion.

2. LESION FINDING TASK for the first image (trial)  The reader marks (localizes) his suspicious locations of shadow (multiple answer is allowed) each of which is also assigned an integer indicating his confidence levels (if he thinks some shadow is obviously a lesion, then he gives a higher integer with respect to the shadow). So, reader marks two things: location and confidence for each suspicious shadow.

3. Second trial and LESION FINDING TASK for the second image (trial)  The researcher gives the reader the second image and reader does the above LESION FINDING TASK for the second image.

4. Repeat this trial for all images.  The reader do the LESION FINDING TASK for all images.

5. Evaluation of TP and FP  The researcher counts the number of their true marking positions (hit) and false making positions (false alarm).

Consequently, we obtain the following table.

**Example data and its Format:**

A single reader and a single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI=63, NL=124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In R console -&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>definitely present</td>
<td>c = 5</td>
<td>f[1] = F_5 = 1</td>
</tr>
</tbody>
</table>

We use two notations for the same number of FPs, e.g., one is f[1] and the other is F_5. We use the former f[1] for programming and the later F_5 is used for descriptions of the theory.

By the R code `BayesianFROC::viewdata(BayesianFROC::dataList.Chakra.1.with.explanation)`, we can see example data named "dataList.Chakra.1.with.explanation".

**Modeling**

Our goal now is to define a model of the random variables H_c, F_c, namely, give a probability law of H_c, F_c.

First, we shall discuss our model rigorously (ignore the confidence). First, to simplify our argument, first we reduce the FP and TP dataset from H_c, F_c to H, F by ignoring the confidence level.
Suppose that there are $N_L$ targets (signal), and radiological context, target is lesion. Suppose that a radiologist try to find these lesions from radiographs. Suppose that now, the reader found $H$ lesions from radiographs which contains $N_L$ lesions, then it is natural to assume that

$$H \sim \text{Binomial}(\theta_H, N_L)$$

where, the Bernoulli success rate $\theta_H$ is one of parameter for our model, which should be estimated.

In addition, suppose that the reader fails $F$ times, namely, the reader marked $F$ locations in radiographs each of which is not a true lesion location. In other words, the reader marked $F$ false positives. Then it is natural to assume that

$$F \sim \text{Poisson}(\theta_F)$$

where, $\theta_F$ is also an another parameter of model, which should be estimated from given data. So, our model has a vector $\theta_H, \theta_F$ as a model parameter.

The above two is very simple, since data is only $H, F$, indicating the number of TP and the number of FP. 

Unfortunately, the FROC data is more complex than above, namely, we have to make a model for data $F_c, H_c, c = 1, \ldots, 5$ instead of the above simplified data $H, F$. That is, reader answers with his confidence level for each suspicious location, which is usually an integer such as 1, 2, 3, 4, 5.

We give a probability law for the random variables $F_c$ and $H_c$ for $c = 1, \ldots, 5$.

Suppose that there are $N_L$ targets, and radiological context, each target is a lesion contained in $N_I$ Radiographs. Suppose that a radiologist try to find lesions. Suppose that now, he found $H_c$ lesions with his $c$-th confidence, then we assume that each random variable $H_c$ is distributed by the following law.

$$H_5 \sim \text{Binomial}(p_5(\theta), N_L)$$

$$H_4 \sim \text{Binomial}(\frac{p_4(\theta)}{1 - p_5(\theta)}, N_L - H_5)$$

$$H_3 \sim \text{Binomial}(\frac{p_3(\theta)}{1 - p_5(\theta) - p_4(\theta)}, N_L - H_5 - H_4)$$

$$H_2 \sim \text{Binomial}(\frac{p_2(\theta)}{1 - p_5(\theta) - p_4(\theta) - p_3(\theta)}, N_L - H_5 - H_4 - H_3)$$

$$H_1 \sim \text{Binomial}(\frac{p_1(\theta)}{1 - p_5(\theta) - p_4(\theta) - p_3(\theta) - p_2(\theta)}, N_L - H_5 - H_4 - H_3 - H_2)$$

where, hit rates $p_1(\theta), p_2(\theta), p_3(\theta), p_4(\theta)$ and $p_5(\theta)$ are some functions of a model parameter $\theta$. We also denote them simply by $p_c$ for $p_c(\theta), c = 1, 2, 3, 4, 5$. In addition, suppose that the reader fails $F_c$ times with his $c$-th confidence, that is, the reader marked $F_c$ false locations in radiographs with his $c$-th confidence. Then it natural to assume that

$$F_5 \sim \text{Poisson}(q_5(\theta)N_X)$$

$$F_4 \sim \text{Poisson}(q_4(\theta)N_X)$$
\[ F_3 \sim \text{Poisson}(q_3(\theta)N_X) \]
\[ F_2 \sim \text{Poisson}(q_2(\theta)N_X) \]
\[ F_1 \sim \text{Poisson}(q_1(\theta)N_X) \]

where, \( N_X = N_I \) or \( N_L \).

The false rates \( q_1(\theta), q_2(\theta), q_3(\theta), q_4(\theta) \) and \( q_5(\theta) \) are functions of a parameter of model.

The above model gives the probability law for the the random variables \( H_c, F_c, c = 1, 2, \ldots, C \), indicating the number of TP and the number of FP for each confidence level \( c = 1, 2, \ldots, C \).

We define \( p_c(\theta) \) and \( q_c(\theta) \) in terms of the model parameter \( \mu, \sigma, \theta_c, c = 1, 2, \ldots, C \).

\[
p_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} \text{Gaussian}(z|\mu, \sigma)dz
\]
\[
q_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} \frac{d}{dz} \log \Phi(z)dz
\]

For any given dataset, we will estimate the model parameter vector \( \theta \);

\[
\theta = (\theta_1, \theta_2, \ldots, \theta_C; \mu, \sigma).
\]

Intuitively, the reason why we choose such functions for \( p_c(\theta) \) is the assumption that each lesion is equipped with i.i.d. latent variable, \( X \) distributed by \( \text{Gaussian}(z|\mu, \sigma) \), and if \( X \) associated to some lesion falls into the interval \( \theta_c < X < \theta_{c+1} \), then we consider that the reader marks this lesion with his \( c \)-th confidence level. In order to emphasize that each \( X \) is associated to some \( l \)-th lesion, \( l = 1, 2, \ldots, N_L \) we denote the latent variable by \( X_l \) for the \( l \)-th lesion instead the latent decision variable \( X \). Here, we uses \textit{latent} to means that the variable \( X \) cannot be observed. Since the latent variable relates decision of reader, and thus, in this context the latent variable is called a \textit{decision} variable.

Similarly, suppose that each image (radiograph) is associated some latent variable \( Y \) distributed by \( N_I \frac{d}{dx} \Phi(z) \) and if the \( Y \) associated to some image falls into interval the interval \( \theta_c < Y < \theta_{c+1} \), then we consider that the reader will false decision with his \( c \)-th confidence level for the image.

**Fundamental equations**

The reason why we use the hit rates such as \( 1 - p_c - p_4 - p_3 \) instead of \( p_c \) is that it ensures the equality \( E[H_c/N_L] = p_c \). This equality is very important to establish Bayesian FROC theory so that it is compatible with the classical FROC theory. It is easy to see that the following two equality holds,

\[
E[H_c/N_L] = p_c,
\]
\[
E[F_c/N_X] = q_c,
\]

where \( E \) denotes the expectation and \( N_X \) is the number of lesion or the number of images and \( q_c \) is a false alarm rate, namely, \( F_c \sim \text{Poisson}(N_X q_c) \).

More precisely or to express the above with model parameter explicitly, we should rewrite it as follows.
\[
E_\theta[H_c/N_L] = p_c(\theta),
\]
\[
E_\theta[F_c/N_X] = q_c(\theta),
\]
where \(E_\theta[X]\) denotes the expectation of a random variable \(X\) with the likelihood \(f(\omega|\theta)\) for data \(\omega\) parameter \(\theta\), namely,
\[
E_\theta[X] := \int X(\omega)f(\omega|\theta)d\omega.
\]
So, the above two equations are rewritten as follows.
\[
E_\theta[H_c/N_L] := \int H_c(\omega)/N_L f(\omega|\theta)d\omega = p_c(\theta),
\]
\[
E_\theta[F_c/N_X] := \int F_c(\omega)/N_X f(\omega|\theta)d\omega = q_c(\theta).
\]

What redundant explanation I am!

These two family of equations are most important one, and the author made this model to satisfy this. Using these equations, we can define the FROC curve such that the curve can be interpreted as the points of expectations.

We call these equations fundamental equations of FROC analysis. Using this, we can calculates the expectations of FPF and TPF in the later.

Some bitch will ask the author what is the original or new? So,..., for such a bitch I remark the following.

The new model by the author is a generative model The classical model can not synthesize dataset so that the total number of hits is bounded from above by the number of lesions.

Love The new model is made with great love of the author and poor condition and poor books (to tell the truth, I did not read any books when I made a prototype) without any support of money.

A details of model The formulation of hit rate differs from the classical theory.

The new model excludes the number of images The formulation of false rate differs from the classical theory and it allows us to exclude the number of images from modeling.

A multiple chemical sensitivity The author diseased the serious, so, the author is a patient of the chemical sensitivity, which make his life of quality much lower.

A multiple chemical sensitivity The author diseased the serious, so, the author is a patient of the chemical sensitivity, which make his life of quality much lower.

# Using the above two equations, we can establish the alternative Bayesian FROC theory preserving classical notions and formulas.

To fit a model to any dataset, we use the code:

```python
fit_Bayesian_FROC()
```
Priors on the Model Parameter.

Recall that our model has the following parameter.

\[ \theta = (\theta_1, \theta_2, \ldots, \theta_C; \mu, \sigma). \]

In this section, we give priors on this parameter. Only one necessarily prior is to ensure the monotonicity on the thresholds parameters.

\[ \theta_1 < \theta_2 < \ldots < \theta_C. \]

To give this monotonicity, we have to assume .... UNDER CONSTRUCTION

Recall that the number of false alarms is distributed by Poisson with rate

\[ q_c(\theta) = \log \frac{\Phi(\theta_{c+1})}{\Phi(\theta_c)}. \]

Visualization of TP, FP by FPF, TPF

How to visualize our data constructed by hit and false alarms, that is, TP and FP? Traditionally, the so-called FPF:False Positive Fraction and TPF:True Positive Fraction are used. Recall that our data format:

A single reader and a single modality case auxiliary: number of images and lesions NI, NL ———

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms (FP:False Positive)</th>
<th>No. of hits (TP:True Positive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>5</td>
<td>( F_5 )</td>
</tr>
<tr>
<td>probably present</td>
<td>4</td>
<td>( F_4 )</td>
</tr>
<tr>
<td>equivocal</td>
<td>3</td>
<td>( F_3 )</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>( F_2 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>( F_1 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( H_5 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( H_4 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( H_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( H_2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( H_1 )</td>
</tr>
</tbody>
</table>

In the above table, we introduce two kinds of random variables \( F_c \) \( H_c \) \( c = 1, 2, 3, 4, 5 \) which are non-negative integers and please keep in mind the notations because, from now on, we use them frequently throughout this paper.

Recall that FPF (False Positive Fraction) is defined as follows;

\[ FPF(5) := \frac{F_5}{N_I}. \]
BayesianFROC

\[ FPF(4) := \frac{F_4 + F_5}{N_I}, \]
\[ FPF(3) := \frac{F_3 + F_4 + F_5}{N_I}, \]
\[ FPF(2) := \frac{F_2 + F_3 + F_4 + F_5}{N_I}, \]
\[ FPF(1) := \frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I}. \]

Similarly, \( TPF \) (True Positive Fraction) is defined as follows;

\[ TPF(5) := \frac{H_5}{N_L}, \]
\[ TPF(4) := \frac{H_4 + H_5}{N_L}, \]
\[ TPF(3) := \frac{H_3 + H_4 + H_5}{N_L}, \]
\[ TPF(2) := \frac{H_2 + H_3 + H_4 + H_5}{N_L}, \]
\[ TPF(1) := \frac{H_1 + H_2 + H_3 + H_4 + H_5}{N_L}. \]

Combining TPF and FPF, we obtain the pairs:

\((FPF(1), TPF(1))\),
\((FPF(2), TPF(2))\),
\((FPF(3), TPF(3))\),
\((FPF(4), TPF(4))\),
\((FPF(5), TPF(5))\).

Plotting these five points in a two-dimensional plain, we can visualize our dataset..

In addition, connecting these points by lines, we obtain the so-called empirical FROC curve.

interpretation of the empirical FROC curve

In fact, if a reader (physician) has a high signal detection ability, namely, he can find more lesions in Radiographs (image), then the number of TPs denoted by \( H_1, H_2, H_3, H_4, H_5 \) will be more and more greater. Thus, the

\( TPF(1), TPF(2), TPF(3), TPF(4), TPF(5) \)

is also greater. Consequently, the points

\((FPF(1), TPF(1))\),
\((FPF(2), TPF(2))\),
BayesianFROC

\[
(FPF(3), TPF(3)), \\
(FPF(4), TPF(4)), \\
(FPF(5), TPF(5)).
\]

are located in upper positions. This indicates that the high observer performance leads the empirical FROC curve to be more upper positions in the plane.

**Visualization of our model by curve**

In this section, we provide the so-called *FROC curve* which is our desired visualization of estimated model. Roughly speaking, an **FROC curve is expected pairs of FPF and TPF**. Namely, the points of FPF and TPF will be on FROC curve if model is well fitting to data. So, comparing the FROC curve and the FPF and TPF, we can evaluate our goodness of fit.

In the above, ha,... I want to die.

Define \( x(c), y(c), c = 1, 2, 3, 4, 5 \) by the expectations of FPF and TPF, respectively, namely,

\[
x(c) := E[FPF(c)],
\]

\[
y(c) := E[TPF(c)].
\]

for \( c = 1, 2, 3, 4, 5 \).

Using the fundamental equations \( E[\theta_c/X] = p_c(\theta), E[\theta_c/N] = q_c(\theta) \), we can rewrite them in terms of the parameters \( \mu, \sigma \) of the latent Gaussian, as follows.

\[
x(c) = E[FPF(c)] = \int_{\theta_c}^\infty \frac{d}{dz} \log \Phi(z) dz = -\log \Phi(\theta_c),
\]

\[
y(c) = E[TPF(c)] = \int_{\theta_c}^\infty \text{Gaussian}(z|\mu, \sigma) dz = \Phi(\frac{\theta_c - \mu}{\sigma}).
\]

From the first equation, we obtain that \( \theta_c = \Phi^{-1}(\exp(-x(c))) \). Substituting this into the second equation, it follows that

\[
y(c) = \Phi(\frac{\Phi^{-1}(\exp(-x(c))) - \mu}{\sigma}).
\]

This implies that the set of points \((x(c), y(c)), c = 1, 2, 3, 4, 5\) consisting of all expectations for the pair of FPF and TPF is contained in the following set:

\[
\{(x, y)|y = \Phi(\frac{\Phi^{-1}(\exp(-x)) - \mu}{\sigma})\}.
\]

We can regard this set as an image of smooth curves. Namely, here we define the so-called FROC curve as a map from 1-dimensional Euclidean space to 2-dimensional Euclidean space, mapping each \( t > 0 \) to
Because \( x(t) = t, t > 0 \) is not bounded, the area under the FROC curve is infinity.

To calculate an alternative notion of AUC in the ordinal ROC theory, we define the so-called AFROC curve:

\[
(x(t), y(t)) = (t, \Phi\left(\Phi^{-1}(\exp(-t)) - \mu\right)/\sigma)
\]

which contains in the rectangular space \([0, 1]^2\). The area Under the (AFROC) curve (briefly, we call it AUC) represents the observer performance. For example, if radiologist detects more lesions with small False Positives (FPs), then AUC would be high.

Using the parameter of the signal distribution, we express AUC as follows,

\[
AUC = \int \eta d\xi = \frac{\mu / \sigma}{\sqrt{1 + 1/\sigma^2}}.
\]

Introducing new parameter \( a := \mu / \sigma \) and \( b := 1 / \sigma \), we can also write

\[
AUC = \frac{a}{\sqrt{1 + b^2}}.
\]

**Generalized Model**

Until now, we use the following two

\[
p_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} Gaussian(z|\mu, \sigma)dz
\]

\[
q_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} \frac{d}{dz} \log \Phi(z)dz
\]

for hit rates and false alarm rates.

However, the explicit representations of these integrands of \( p_c(\theta), q_c(\theta) \) are not determined in a prior manner. So, such explicit representations are redundant for a general theory. So, to simplify our argument in the following, we use general notations \( P(z|\theta_P), Q(z|\theta_Q) \) instead of the above two integrands \( Gaussian(z|\mu, \sigma) \) and \( \frac{d}{dz} \log \Phi(z) \), and rewrite them as follows,

\[
p_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} P(z|\theta_P)dz,
\]

\[
q_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} Q(z|\theta_Q)dz.
\]
In the sequel, we assume that $P(z|\theta_P)$ is a **probability density** function (namely, its total integral is one) and $Q(z|\theta_Q)$ is a **positive** function (not necessarily to be a probability function). Namely,

$$\int P(z|\theta_P)dz = 1,$$

for all $\theta_P$ and

$$Q(z|\theta_Q) > 0,$$

for all $z$ and $\theta_Q$.

*A single reader and a single modality*

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

We give a probability law for the random variables $F_c, H_c, c = 1, \ldots, 5$.

Suppose that there are $N_L$ targets, and radiological context, each target is a lesion contained in some Radiograph as a shadow. Suppose that a radiologist try to find lesions for $N_I$ radiographs. Suppose that now, the radiologist fined $H_c$ lesions with his $c$-th confidence, then we assume that

$$H_5 \sim Binomial(p_5(\theta), N_L)$$

$$H_4 \sim Binomial\left(\frac{p_4(\theta)}{1-p_5(\theta)}, N_L - H_5\right)$$

$$H_3 \sim Binomial\left(\frac{p_3(\theta)}{1-p_5(\theta) - p_4(\theta)}, N_L - H_5 - H_4\right)$$

$$H_2 \sim Binomial\left(\frac{p_2(\theta)}{1-p_5(\theta) - p_4(\theta) - p_3(\theta)}, N_L - H_5 - H_4 - H_3\right)$$

$$H_1 \sim Binomial\left(\frac{p_1(\theta)}{1-p_5(\theta) - p_4(\theta) - p_3(\theta) - p_2(\theta)}, N_L - H_5 - H_4 - H_3 - H_2\right)$$

where, hit rates $p_1(\theta), p_2(\theta), p_3(\theta), p_4(\theta)$ and $p_5(\theta)$ are functions of a model parameter $\theta$. In addition, suppose that the reader fails $F_c$ times with his $c$-th confidence, that is, the reader marked $F_c$ false positives. Then it natural to assume that
\[ F_5 \sim \text{Poisson}(q_5(\theta)N_X) \]
\[ F_4 \sim \text{Poisson}(q_4(\theta)N_X) \]
\[ F_3 \sim \text{Poisson}(q_3(\theta)N_X) \]
\[ F_2 \sim \text{Poisson}(q_2(\theta)N_X) \]
\[ F_1 \sim \text{Poisson}(q_1(\theta)N_X) \]

where, \( N_X = N_I \) or \( N_L \) false rates \( q_1(\theta), q_2(\theta), q_3(\theta), q_4(\theta) \) and \( q_5(\theta) \) are functions of a parameter of model.

The above model calculates the event of the data \( H_c, F_c, c = 1, 2, \ldots, C \) arises, indicating the number of TP and the number of FP.

We use Gaussian distributions for the functions \( p_c(\theta) \) and \( q_c(\theta) \) as follows.

\[ p_c(\theta) = \int_{\theta_c}^{\theta_c+1} P(z|\theta_P)dz \]
\[ q_c(\theta) = \int_{\theta_c}^{\theta_c+1} Q(z|\theta_Q)dz \]

where the model parameter vector is

\[ \theta = (\theta_1, \theta_2, \ldots, \theta_C; \theta_P, \theta_Q). \]

Recall that \( FPF \) is defined as follows;

\[ FPF(5) := \frac{F_5}{N_I}, \]
\[ FPF(4) := \frac{F_4 + F_5}{N_I}, \]
\[ FPF(3) := \frac{F_3 + F_4 + F_5}{N_I}, \]
\[ FPF(2) := \frac{F_2 + F_3 + F_4 + F_5}{N_I}, \]
\[ FPF(1) := \frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I}. \]

Similarly, \( TPF \) is defined as follows;

\[ TPF(5) := \frac{H_5}{N_L}, \]
\[ TPF(4) := \frac{H_4 + H_5}{N_L}, \]
\[TPF(3) := \frac{H_3 + H_4 + H_5}{N_L},\]
\[TPF(2) := \frac{H_2 + H_3 + H_4 + H_5}{N_L},\]
\[TPF(1) := \frac{H_1 + H_2 + H_3 + H_4 + H_5}{N_L}.
\]
Combining TPF and FPF, we obtain the pairs.
\[(FPF(1), TPF(1)),\]
\[(FPF(2), TPF(2)),\]
\[(FPF(3), TPF(3)),\]
\[(FPF(4), TPF(4)),\]
\[(FPF(5), TPF(5)).\]

Plotting these five points in a 2-dimensional plain, we can visualize our dataset.

**Visualization of a generalized model by curve**

In this section, we provide the so-called FROC curve which is our desired visualization of estimated model. Roughly speaking, an FROC curve is expected pairs of FPF and TPF. Namely, the points of FPF and TPF will be on FROC curve if model is well fitting to data. So, comparing the FROC curve and the FPF and TPF, we can evaluate our goodness of fit.

Let \(c = 1, 2, 3, 4, 5\).

Define
\[x(c) := E[FPF(c)],\]
\[y(c) := E[TPF(c)].\]

Using the fundamental equations \(E_\theta[H_c/N_L] = p_c(\theta), E_\theta[F_c/N_X] = q_c(\theta),\)
\[y(c) = E[TPF(c)] = \int_{\theta_c}^{\infty} P(x|\theta_P)dx =: \Psi_P(\theta_c),\]
\[x(c) = E[FPF(c)] = \int_{\theta_c}^{\infty} Q(x|\theta_Q)dx =: \Psi_Q(\theta_c),\]

where \(\Psi_P\) and \(\Psi_Q\) denote the cumulative functions of the functions \(P\) and \(Q\), respectively. (That is, \(\Psi_P(x) := \int_x^{\infty} P(t)dt\) and \(\Psi_Q(x) := \int_x^{\infty} Q(t)dt\).)

Note that we assume that \(P\) is a probability density function but \(Q\) is not. So, \(\Psi_P\) is a cumulative distribution function, but \(\Psi_Q\) is not a cumulative ‘distribution’ function.

This implies that all expectations for the pair of FPF and TPF, namely \((x(c), y(c)) = (E[FPF(c)], E[TPF(c)])\), is on the following set:
\{(x(t), y(t))|x(t) = \Psi_Q(t), y(t) = \Psi_P(t), t > 0\}.

We can regard this set as the image of the smooth curve which is called the generalized FROC curve in this manuscript.

From the first equation, we obtain that \(\theta_c = \Psi_Q^{-1}(x(c))\). Substituting this into the second equation, we obtain that

\[y(c) = \Psi_P(\Psi_Q^{-1}(x(c))).\]

This implies that all exceptions for the pair of FPF and TPF is on the set:

\[\{(x, y)|y = \Psi_P(\Psi_Q^{-1}(x)).\}.

We can regard this set as an image of smooth curves.

\[(x(t), y(t)) = (t, \Psi_P(\Psi_Q^{-1}(t)))\]

Since \(x(t) = t, t > 0\) is not bounded, the area under the FROC curve is infinity.

To calculate the alternative notion of AUC in the ordinal ROC theory, we define the so-called AFROC curve:

\[(\xi(t), \eta(t)) = (1 - e^{-t}, \Psi_P(\Psi_Q^{-1}(x)))\]

### MRMC Model for Multiple Readers and Multiple Modalities (MRMC)

<table>
<thead>
<tr>
<th>modality ID</th>
<th>reader ID</th>
<th>confidence</th>
<th>No. of FPs</th>
<th>No. of TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI=63, NL=124</td>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
</tr>
<tr>
<td>definitely present</td>
<td>1</td>
<td>1</td>
<td>c[1] = 5</td>
<td>f[1] = F_{1,1,5}</td>
</tr>
<tr>
<td>probably present</td>
<td>1</td>
<td>1</td>
<td>c[2] = 4</td>
<td>f[2] = F_{1,1,4}</td>
</tr>
<tr>
<td>equivocal</td>
<td>1</td>
<td>1</td>
<td>c[3] = 3</td>
<td>f[3] = F_{1,1,3}</td>
</tr>
<tr>
<td>subtle</td>
<td>1</td>
<td>1</td>
<td>c[4] = 2</td>
<td>f[4] = F_{1,1,2}</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>1</td>
<td>c[5] = 1</td>
<td>f[5] = F_{1,1,1}</td>
</tr>
<tr>
<td>definitely present</td>
<td>1</td>
<td>2</td>
<td>c[6] = 5</td>
<td>f[6] = F_{1,2,5}</td>
</tr>
<tr>
<td>probably present</td>
<td>1</td>
<td>2</td>
<td>c[7] = 4</td>
<td>f[7] = F_{1,2,4}</td>
</tr>
<tr>
<td>equivocal</td>
<td>1</td>
<td>2</td>
<td>c[8] = 3</td>
<td>f[8] = F_{1,2,3}</td>
</tr>
<tr>
<td>subtle</td>
<td>1</td>
<td>2</td>
<td>c[9] = 2</td>
<td>f[9] = F_{1,2,2}</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>2</td>
<td>c[10] = 1</td>
<td>f[10] = F_{1,2,1}</td>
</tr>
<tr>
<td>probably present</td>
<td>2</td>
<td>1</td>
<td>c[12] = 4</td>
<td>f[12] = F_{2,1,4}</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>1</td>
<td>c[14] = 2</td>
<td>f[14] = F_{2,1,2}</td>
</tr>
<tr>
<td>definitely present</td>
<td>2</td>
<td>2</td>
<td>c[16] = 5</td>
<td>f[16] = F_{2,2,5}</td>
</tr>
</tbody>
</table>
An example data in this package

R codes

R object named dd is an example data, and to show the above table format, execute the following codes

library(BayesianFROC);viewdata(dd)

In this section we use the abbreviation MRMC which means Multiple Readers and Multiple Modalities. In MRMC, Observer performance ability has individualities caused by readers and modalities. Once we includes these individual differences in our Bayesian model, such model will give us an answer for the modality comparison issues.

The author implements several models for MRMC.

1) Non hierarchical MRMC model
2) hierarchical MRMC model
3) A Single reader and multiple modalities model

I am a patient of Multiple Chemical Sensitivity (CS) which cause inflammations in the brain and it makes me hard to write this. I know there are many mistakes. When I read my writing, I always find and fix. Please forgive me, because CS makes me foolish.

MRMC model Without hyper parameter

To include heterogeneity caused by readers and modalities, the author first made a hierarchical model. However, the model has divergent transitions in MCMC iterations. Thus the author also made a non-hierarchical model in which the author removed the hyper parameters to get more stable MCMC simulation and he confirmed that the new model is divergent free with my fake data.

In MRMC models, the model parameter is a vector denoted by

$$\theta = (\theta_1, \theta_2, ..., \theta_C; \mu, \sigma),$$

where each $\theta_i (i = 1, 2, ..., C)$ is a real number and $\mu, \sigma$ are $(M, R)$-matrices whose components are denoted by

$$\mu_{1,1}, \mu_{1,2}, \mu_{1,3}, ..., \mu_{1,r}, ..., \mu_{1,R},$$

$$\mu_{2,1}, \mu_{2,2}, \mu_{2,3}, ..., \mu_{2,r}, ..., \mu_{2,R},$$

$$\mu_{3,1}, \mu_{3,2}, \mu_{3,3}, ..., \mu_{3,r}, ..., \mu_{3,R},$$

$$...,$$

$$\mu_{m,1}, \mu_{m,2}, \mu_{m,3}, ..., \mu_{m,r}, ..., \mu_{m,R},$$
\[ \mu_{M,1}, \mu_{M,2}, \mu_{M,3}, \ldots, \mu_{M,r}, \ldots, \mu_{M,R}, \]

and

\[ \sigma_{1,1}, \sigma_{1,2}, \sigma_{1,3}, \ldots, \sigma_{1,r}, \ldots, \sigma_{1,R}, \]
\[ \sigma_{2,1}, \sigma_{2,2}, \sigma_{2,3}, \ldots, \sigma_{2,r}, \ldots, \sigma_{2,R}, \]
\[ \sigma_{3,1}, \sigma_{3,2}, \sigma_{3,3}, \ldots, \sigma_{3,r}, \ldots, \sigma_{3,R}, \]
\[ \ldots, \]
\[ \sigma_{m,1}, \sigma_{m,2}, \sigma_{m,3}, \ldots, \sigma_{m,r}, \ldots, \sigma_{m,R}, \]
\[ \ldots, \]
\[ \sigma_{M,1}, \sigma_{M,2}, \sigma_{M,3}, \ldots, \sigma_{M,r}, \ldots, \sigma_{M,R}, \]

where the subscripts \( m \) and \( r \) indicate the \( m \)-th modality and the \( r \)-th reader, respectively.

Note that we use the notation \( \theta \) for \[ \theta = (\theta_1, \theta_2, \ldots, \theta_C: \mu, \sigma) \]
and do not confuse it with \[ (\theta_1, \theta_2, \ldots, \theta_C). \]

Using the model parameter \( \theta \), we can define AUC associated with each pair of reader and modality as follows.

\[
AUC_{m,r} = \frac{\mu_{m,r}/\sigma_{m,r}}{\sqrt{1 + 1/\sigma_{m,r}^2}}.
\]

Furthermore, we can extract the efficacy of modality.

\[
AUC_m = \frac{1}{R} \sum_{r=1}^{R} AUC_{m,r},
\]

which is also denoted by \( A[m], m=1, 2, \ldots, M \) in the R console (or R studio console) and retained in the R object of the S4 class (the so-called stanfit or its extended class).

Using \( A[m], m=1, 2, \ldots, M \), we can compare modalities such as MRI, CT, PET, etc. Note that if our trial use x-ray films taken by MRI and CT, then \( M=2 \). If images are taken by MRI, CT, PET, then \( M=3 \). So, \( A[m], m=1, 2, \ldots, M \) is a function of the model parameter. In Bayesian sense, the estimates are posterior samples and thus, \( A[m], m=1, 2, \ldots, M \) are obtained as MCMC samples. Using these, we can calculate posterior probabilities of any events. This is the author’s main scheme. Ha,, I want to

Of course, these AUCs are defined as the area under the AFROC curve for the \( r \)-th reader and the \( m \)-th modality. The so-called FROC curve for the \( r \)-th reader and the \( m \)-th modality is a map from 1-dimensional Euclidean space to 2-dimensional Euclidean space, mapping each \( t > 0 \) to
\[(x_{m,r}(t), y_{m,r}(t)) = (t, \Phi(\frac{\Phi^{-1}(\exp(-t)) - \mu_{m,r}}{\sigma_{m,r}}))\]

Because \(x(t) = t, t > 0\) is not bounded, the area under the FROC curve is infinity.

To calculate an alternative notion of AUC in the ordinal ROC theory, we define the so-called AFROC curve:

\[(\xi_{m,r}(t), \eta_{m,r}(t)) = (1 - e^{-t}, \Phi(\frac{\Phi^{-1}(\exp(-t)) - \mu_{m,r}}{\sigma_{m,r}}))\]

which contains in the rectangular space \([0, 1]^2\).

**Probability law of hits**

In the sequel, the subscripts \(m, r\) mean the \(m\)-th modality and the \(r\)-th reader, respectively.

Random variables of hits are distributed as follows.

\[H_{5,m,r} \sim \text{Binomial}(p_{5,m,r}(\theta), N_L),\]

where the notation \(H_{5,m,r}\) denotes the number of hits (TPs) with confidence level 5 of the \(m\)-th modality for the \(r\)-th reader.

Now, the \(H_{5,m,r}\) targets (signals, lesions) are found by the reader (radiologist), and the number of remaining targets is \(N_L - H_{5,m,r}\).

Thus, the number of hits with the 4-th confidence level \(H_{4,m,r}\) should be drawn from the binomial distribution with remaining targets whose number is \(N_L - H_{5,m,r}\) and thus

\[H_{4,m,r} \sim \text{Binomial}(\frac{p_{4,m,r}(\theta)}{1 - p_{5,m,r}(\theta)}, N_L - H_{5,m,r}).\]

Similarly,

\[H_{3,m,r} \sim \text{Binomial}(\frac{p_{3,m,r}(\theta)}{1 - p_{5,m,r}(\theta) - p_{4,m,r}(\theta)}, N_L - H_{5,m,r} - H_{4,m,r}).\]

\[H_{2,m,r} \sim \text{Binomial}(\frac{p_{2,m,r}(\theta)}{1 - p_{5,m,r}(\theta) - p_{4,m,r}(\theta) - p_{3,m,r}(\theta)}, N_L - H_{5,m,r} - H_{4,m,r} - H_{3,m,r}).\]

\[H_{1,m,r} \sim \text{Binomial}(\frac{p_{1,m,r}(\theta)}{1 - p_{5,m,r}(\theta) - p_{4,m,r}(\theta) - p_{3,m,r}(\theta) - p_{2,m,r}(\theta)}, N_L - H_{5,m,r} - H_{4,m,r} - H_{3,m,r} - H_{2,m,r}).\]

**Probability law of false alarms**

Let \(N_X\) be the one of the followings.

1) \(N_X = N_L\) (The number of lesions), if \(\text{ModifiedPoisson} = \text{TRUE}\).
2) \(N_X = N_I\) (The number of images), if \(\text{ModifiedPoisson} = \text{FALSE}\).
Using $N_X$, we assume the following,

$$ F_{5,m,r} \sim \text{Poisson}(q_5(\theta)N_X), $$

$$ F_{4,m,r} \sim \text{Poisson}(q_4(\theta)N_X), $$

$$ F_{3,m,r} \sim \text{Poisson}(q_3(\theta)N_X), $$

$$ F_{2,m,r} \sim \text{Poisson}(q_2(\theta)N_X), $$

$$ F_{1,m,r} \sim \text{Poisson}(q_1(\theta)N_X), $$

where subscripts $m, r$ mean the $m$-th modality and the $r$-th reader, respectively.

The rate $p_{c,m,r}(\theta)$ and $q_c(\theta)$ are calculated from the model parameter $\theta$.

We use a Gaussian distribution and the cumulative distribution function $\Phi()$ of the standard Gaussian for the functions $p_{c,m,r}(\theta)$ and $q_c(\theta)$ as following manner.

$$ p_{c,m,r}(\theta) = \int_{\theta_c}^{\theta_{c+1}} \text{Normal}(z|\mu_{c,m,r},\nu_{c,m,r})dz $$

$$ q_c(\theta) = \int_{\theta_c}^{\theta_{c+1}} \frac{d}{dz} \log \Phi(z)dz $$

where the model parameter vector is

$$ \theta = (\theta_1, \theta_2, ..., \theta_C; \theta_P, \theta_Q). $$

By specifying a model parameter $\theta = (\theta_1, \theta_2, ..., \theta_C; \theta_P, \theta_Q)$, we can make a fake dataset consisting of hit data $H_{c,m,r}$, false alarm data $F_{c,m,r}$ for each $c, m, r$. So, our model is a generative model and this is a crucial difference between our model and the classical one.

**Without hyper parameter MRMC model**

**A Non-Centered Implementation**

$AA[md,qd] \sim \text{Normal}(A[md],\text{hyper}_v[qd])$

Non centered version is the following:

$AA_{\text{tilde}}[md,qd] \sim \text{Normal}(0,1)$

$AA[md,qd] = A[md]+\text{hyper}_v[qd]*AA_{\text{tilde}}$

But, the $AA[md,qd]$ is already defined as follows.

$AA[md,qd]=\text{Phi}( (\mu[md,qd]/v[md,qd])/\text{sqrt}((1/v[md,qd])^2+1) )$;
Thus usual non centered model cannot be implemented.
The assumption
\[ AA[md,qd] \sim \text{Normal}(A[md],\text{hyper\_v}[qd]) \]
is an approximation. So, this model is not correct. I am not sure whether the approximation worsen
my model.
The hyper parameters have been in use for more than 2 years in this package. However it caused
divergent transitions. Thus the author made a new model without these hyper parameters.
Example dataset is dd and ddd and dddd and ddddd and etc.

Validation of model via SBC
SBC tests the Null hypothesis that the MCMC sampling is correct by using some rank statistic
which synthesizes a histogram. If this hits gram is not uniformly distributed, then we reject the null
hypothesis, and we conclude that our MCMC sampling contains bias.

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., and Gelman, A. (2018). Validating Bayesian In-

Validation of model via Posterior Predictive p value
See ppp().
Let \( \theta_1, \theta_2, \ldots, \theta_n \) be MCMC samples from a posterior distribution \( \pi(.|D) \) for a given dataset \( D \). Let
\( L(y|\theta_i) \) be a likelihood function for a dataset \( y \) and model parameter \( \theta_i \). Let
\[ y_i^j \sim L(|\theta_i). \]

For any real-valued function \( \phi = \phi(y, \theta) \), we can calculates its integral with the posterior predictive
measure as the approximation of two steps Monte Carlo integral as follows.
\[
\int \int \phi(y, \theta)L(y|\theta)\pi(\theta|y)dyd\theta \\
= \int \Sigma_i \phi(y_i, \theta_i)L(y|\theta_i)dy \\
= \Sigma_j \Sigma_i \phi(y_j^i, \theta_i)L(y_j^i|\theta_i).
\]
Using \( \phi = 1(T(y, \theta) > T(y, \theta_{\text{observed}})) \), we obtain the so-called posterior predictive p value. (The
author hates this notion.)

In my opinion, this criteria is not clear whether it is reliable quantities for evaluations.

Validation of model; Comparison between truth and estimates of fake data-sets which are
drawn using the truth.

I think this is the most fundamental and intuitive validation.

Under Construction

Appendix: —— Terminology ——

hit which is also called True Positive: TP, which is denoted with each confidence level, \( c = 1, 2, 3, \ldots, C \) as follows: \( H_1, H_2, \ldots, H_C \) or \( h=c(h[1], h[2], \ldots, h[C]) \), where \( h[1]=H_C \) corresponds a number of hit with most high confidence level
**False alarm** which is also called False Positive: FP, which is denoted with each confidence, $c = 1, 2, 3, \ldots, C$ levels as follows: $F_1, F_2, \ldots, F_C$ or $f = c(f[1], f[2], \ldots, f[C])$, where $f[1] = F_C$ corresponds a number of false alarms with most high confidence level.

**Modality** Imaging methods, such as MRI, CT, PET,...etc. In another context, it means efficacy of treatment.

**Reader** is a radiologist, physician, who try to detect lesions from radiographs. For a single image, reader can answer multiple suspicious shadows and he assigns to each suspicious shadow his or her confidence level. So, the reader localizes and rates for each suspicious shadows. A data analyst evaluates whether each reader’s localization of lesion is true or false. Note that a single image can synthesize multiple-false positives or multiple true positives. Such a multiplicity distinguishes FROC analysis with ordinal ROC analysis.

**Image** is a radiograph taken by MRI, CT, PET, etc.

**Modality comparison** The question that which modality (MRI, CT, PET, ... etc) is best to detect lesions in radiographs? In order to answer this question, the FROC analysis exists.

**hit rate** Each lesion can synthesize a hit of confidence level $c$ according to Bernoulli distribution with probability of $p_c$, which call hit rate (of $c$)

**false alarm rate** Each image synthesize a false alarm (False Positive: FP) of confidence level $c$ according to Poisson distribution with probability of $\lambda_c$, which call false alarm rate (of $c$) or simply false rate.

**Number of images** which is denoted by $N_I$. An image means a radiograph or an X ray film, including shadows, each of which is caused by lesions or noise. Namely, each radiograph does not necessarily includes lesions.

**Number of lesions** Suppose that there are $N_I$ radiographs. Then by summing the number of lesions over all radiographs, we obtain the number of lesion $N_L$.

**FROC curve** alternative notion of ROC curve in FROC context.

**AFROC curve** Alternative-FROC curve, whose area under the curve indicates observer performance. Since area under the FROC curve is infinity, we use this area under the AFROC curve instead of the area under the FROC curve.

**AUC** A real number between 0 and 1, indicating how many lesions radiologist can detect from radiographs. It is the area under the AFROC curve. In ROC context, AUC should be greater than 0.5, but in FROC context, the interpretation of AUC is not same as that in ROC context. For example, AUC $= 0.5$ does not means that it is same as the most bad observer performance.

**Chi square** The difference of expectation minus observation, namely it is estimates minus actual observed data. Smaller is better.

**Posterior Predictive P value (PPP)** This is a posterior predictive probability of the event that a test statistic is greater than its observed value. The author implements the $\chi^2$ goodness of fit as a test statistic and in this context, if the PPP is small then we reject the null hypothesis that the model is well fit to data. The author hates this traditional bitch.

**FPF:False Positive Fraction** Cumulative sum of false alarms (FPs) divided by the number of Images or the number of lesions. Using FPFs as x and TPFs as y, we can visualize FPs and TPs.

**TPF:True Positive Fraction** Cumulative sum of hits (TPs) decided by the number of Lesions (signals, targets). Using FPFs as x and TPFs as y, we can visualize FPs and TPs.
check hit is less than NL

Now, I am in very serious condition both money and employment. I cannot get any job, this package development cannot save my life.
I am a chemical sensitivity patient. I cannot overcome this serious disease.
When I made this package, I hoped this makes my life safe, but it cannot.
I really Despair my life.
I do not study Statistics, but geometry, differential geometry.

check_hit_is_less_than_NL

Check total hit is less than NL for each reader and each modality

Description

This check a give dataset consisting of MRMC data satisfies the condition that the number hits is less than the number of lesions for each reader and each modality.

Usage

check_hit_is_less_than_NL(dataList)

Arguments

dataList A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of mutiple readers or mutiple modalities, then modality ID and reader ID are included also.
The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data.
For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
h = c(41,22,14,8,1), # number of hits for each confidence level
f = c(1,2,5,11,13), # number of false alarms for each confidence level
NL = 124, # number of lesions (signals)
NI = 63, # number of images (trials)
C = 5) # number of confidence, .. the author thinks it can be calculated as the length of h or f ...? ha,why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).
To make this R object dataList representing FROC data, this package provides three functions:

convertFromJafroc() If data is a JAFROC .xlsx formulation.
dataset_creator_new_version() Enter TP and FP data by table .
create_dataset() Enter TP and FP data by interactive manner.
Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (src), dataList is a list consisting of f, h, NL,NI,C where f, h are numeric vectors and NL,NI,C are positive integers.

- f: Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- h: Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- NL: A positive integer, representing Number of Lesions.
- NI: A positive integer, representing Number of Images.
- C: A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

data Format:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
</table>

*false alarms = False Positives = FP
*hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion)
case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector $c$ should not be specified. If specified, will be ignored, since it is created by $c <- c(rep(C:1))$ automatically in the program and do not refer from user input data even if it is specified explicitly, where $C$ is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector $c <- c(rep(C:1))$ via a table which can be displayed by the function `viewdata()`.

### Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an R list object representing FROC data must contain components $m, q, c, h, f, NL, C, M, Q$.

- $C$ A positive integer, representing the highest number of confidence level, this is a scalar.
- $M$ A positive integer vector, representing the number of modalities.
- $Q$ A positive integer, representing the number of readers.
- $m$ A vector of positive integers, representing the modality ID vector.
- $q$ A vector of positive integers, representing the reader ID vector.
- $c$ A vector of positive integers, representing the confidence level. This vector must be made by $rep(rep(C:1), M*Q)$
- $h$ A vector of non-negative integers, representing the number of hits.
- $f$ A vector of non-negative integers, representing the number of false alarms.
- $NL$ A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by $C$) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from $C$. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$q$</td>
<td>$c$</td>
<td>$f$</td>
<td>$h$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
</tbody>
</table>
check_rhat

1 1 2 29 55
1 1 1 21 22
1 2 3 6 100
1 2 2 15 44
1 2 1 22 11
2 1 3 6 66
2 1 2 24 55
2 1 1 23 1
2 2 3 5 66
2 2 2 30 55
2 2 1 40 44

* false alarms = False Positives = FP
* hits = True Positives = TP

Value
Logical, TRUE or FALSE. If TRUE, then the format of dataset is correct. If not, then the dataset is incorrect in the sense that the number of hits is greater than the number of lesions for some reader and some imaging modality.

Examples

```r
logical <- check_hit_is_less_than_NL(BayesianFROC::dd)
```

check_rhat

Diagnosis of MCMC sampling

Description
This function evaluate R hat statistics for any fitted model object of class stanfit.

Usage
check_rhat(StanS4class)

Arguments
StanS4class An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and etc.
Details

It evaluate whether or not r hat statistics are all near 1.

Value

Logical, that is TRUE or FALSE. If model converges then TRUE, and if not FALSE.

Author(s)

betanalpha, so not my function. But I modified it. So, alphabetan is one of the standeveloper, so his function will has consensus, thus I use it.

References


Description

To pass the return value to the calculator of the posterior predictive p value.

Usage

chi_square_at_replicated_data_and_MCMC_samples_MRMC(
  StanS4class,
  summary = TRUE,
  seed = NA,
  serial.number = NA
)

Arguments

StanS4class An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc

summary Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

seed This is used only in programming phase. If seed is passed, then, in procedure indicator the seed is printed. This parameter is only for package development.
serial.number  An positive integer or Character. This is for programming perspective. The author use this to print the serial numbre of validation. This will be used in the validation function.

Details

For a given dataset $D_0$, let $\pi(\cdot | D_0)$ be a posterior of the given data $D_0$, then we can draw posterior samples.

$$
\theta_1 \sim \pi(\cdot | D_0), \\
\theta_2 \sim \pi(\cdot | D_0), \\
\theta_3 \sim \pi(\cdot | D_0), \\
\ldots, \\
\theta_n \sim \pi(\cdot | D_0).
$$

We let $L(\cdot)$ be a likelihood function, which is also denoted by $L(y|\theta)$ for a given data $y$. But, the specification of a data $y$ is somehow coversome, thus, to denote the function sending each $y$ into $L(y|\theta)$, we use the notation $L(\cdot)$.

Then we can synthesize samples of data in only one time drawing from the collection of likelihoods $L(\cdot | \theta_1), L(\cdot | \theta_2), \ldots, L(\cdot | \theta_n)$.

$$
y_1 \sim L(\cdot | \theta_1), \\
y_2 \sim L(\cdot | \theta_2), \\
y_3 \sim L(\cdot | \theta_3), \\
\ldots, \\
y_n \sim L(\cdot | \theta_n).
$$

Altogether, using these pair of samples $(y_i, \theta_i)$, $i = 1, 2, \ldots, n$, we calculates the return value of this function. That is,

$$
\chi(y_1 | \theta_1), \\
\chi(y_2 | \theta_2), \\
\chi(y_3 | \theta_3), \\
\ldots, \\
\chi(y_n | \theta_n).
$$

This is contained in a return value, so the return value is a vector of length is the number of MCMC iterations except the burn-in period.

Application of this return value: calculate the so-called Posterior Predictive P value.

In other functions, the author use this function with many seeds. Namely, chaning seed, we can obtain
\[
y_{1,1}, y_{1,2}, y_{1,3}, \ldots, y_{1,j}, \ldots, y_{1,J} \sim L(.|\theta_1),
\]
\[
y_{2,1}, y_{2,2}, y_{2,3}, \ldots, y_{2,j}, \ldots, y_{2,J} \sim L(.|\theta_2),
\]
\[
y_{3,1}, y_{3,2}, y_{3,3}, \ldots, y_{3,j}, \ldots, y_{3,J} \sim L(.|\theta_3),
\]
\[\ldots,
\]
\[
y_{i,1}, y_{i,2}, y_{i,3}, \ldots, y_{i,j}, \ldots, y_{i,J} \sim L(.|\theta_i),
\]
\[\ldots,
\]
\[
y_{I,1}, y_{I,2}, y_{I,3}, \ldots, y_{I,j}, \ldots, y_{I,J} \sim L(.|\theta_I).
\]

where \(L(.|\theta_i)\) is a likelihood function for a model parameter \(\theta_i\). And thus, we calculates the chi square statistics.

\[
\chi(y_{1,1}|\theta_1), \chi(y_{1,2}|\theta_1), \chi(y_{1,3}|\theta_1), \ldots, \chi(y_{1,j}|\theta_1), \ldots, \chi(y_{1,J}|\theta_1),
\]
\[
\chi(y_{2,1}|\theta_2), \chi(y_{2,2}|\theta_2), \chi(y_{2,3}|\theta_2), \ldots, \chi(y_{2,j}|\theta_2), \ldots, \chi(y_{2,J}|\theta_2),
\]
\[
\chi(y_{3,1}|\theta_3), \chi(y_{3,2}|\theta_3), \chi(y_{3,3}|\theta_3), \ldots, \chi(y_{3,j}|\theta_3), \ldots, \chi(y_{3,J}|\theta_3),
\]
\[\ldots,
\]
\[
\chi(y_{i,1}|\theta_i), \chi(y_{i,2}|\theta_i), \chi(y_{i,3}|\theta_i), \ldots, \chi(y_{i,j}|\theta_i), \ldots, \chi(y_{i,J}|\theta_i),
\]
\[\ldots,
\]
\[
\chi(y_{I,1}|\theta_I), \chi(y_{I,2}|\theta_I), \chi(y_{I,3}|\theta_I), \ldots, \chi(y_{I,j}|\theta_I), \ldots, \chi(y_{I,J}|\theta_I).
\]

which are used when we calculate the so-called Posterior Predictive \(P\) value to test the null hypothesis that our model is fitted a data well.

Revised 2019 Sept. 8 Revised 2019 Dec. 2

Revised 2020 March

Value

From any given posterior MCMC samples \(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n\) (provided by stanfitExtended object), it calculates a return value as a vector of the form \(\chi(y_i|\theta_i), i = 1, 2, \ldots\), where each dataset \(y_i\) is drawn from the corresponding likelihood \(likelihood(.|\theta_i), i = 1, 2, \ldots\), namely,

\[
y_i \sim likelihood(.|\theta_i).
\]

The return value also retains these \(y_i, i = 1, 2, \ldots\)

Revised 2019 Dec. 2
Examples

```r
## Not run:
fit <- fit_Bayesian_FROC( ite = 1111, dataList = ddd )
a <- chi_square_at_replicated_data_and_MCMC_samples_MRMC(fit)
b <- a$List_of_dataList
lapply(b, plot_FPF_and_TPF_from_a_dataset)

## End(Not run)
```

## chi_square_goodness_of_fit

**Chi square goodness of fit statistics at each MCMC sample w.r.t. a given dataset.**

### Description

Calculates a vector, consisting of the Goodness of Fit (Chi Square) for a given dataset $D$ and each posterior MCMC samples $\theta_i = \theta_i(D)$, $i = 1, 2, 3, \ldots$, namely,

$$
\chi^2(D|\theta_i)
$$

for $i = 1, 2, 3, \ldots$ and thus its dimension is the number of MCMC iterations.

Note that in MRMC cases, it is defined as follows.

$$
\chi^2(D|\theta) := \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{c=1}^{C} \left( \frac{[H_{c,m,r} - N_L \times p_{c,m,r}(\theta)]^2}{N_L \times p_{c,m,r}(\theta)} + \frac{[F_{c,m,r} - (\lambda_c - \lambda_{c+1}) \times N_L]^2}{(\lambda_c(\theta) - \lambda_{c+1}(\theta)) \times N_L} \right).
$$

where a dataset $D$ consists of the pairs of the number of False Positives and the number of True Positives $(F_{c,m,r}, H_{c,m,r})$ together with the number of lesions $N_L$ and the number of images $N_I$ and $\theta$ denotes the model parameter.

### Usage

```r
chi_square_goodness_of_fit(
  StanS4class,
  dig = 3,
  h = StanS4class$dataList$h,
  f = StanS4class$dataList$f
)
```
**chi_square_goodness_of_fit**

**Arguments**

StanS4class  
An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and etc.

dig  
To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `...`. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

h  
A vector of positive integers, representing the number of hits. This variable was made in order to substitute the hits data drawn from the posterior predictive distributions. In famous Gelman’s book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.

f  
A vector of positive integers, representing the number of false alarms. This variable was made in order to substitute the false alarms data drawn from the posterior predictive distributions. In famous Gelman’s book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.

**Details**

To calculate the chi square (goodness of fit) $\chi^2(y|\theta)$ test statistics, the two variables are required; one is an observed dataset $y$ and the other is an estimated parameter $\theta$. In the classical chi square values, MLE(maximal likelihood estimator) is used for an estimated parameter $\theta$ in $\chi^2(y|\theta)$. However, in the Bayesian context, the parameter is not deterministic and we consider it is a random variable such as samples from the posterior distribution. And such samples are obtained in the Hamiltonian Monte Carlo Simulation. Thus we can calculate chi square values for each MCMC sample.

**Value**

Chi squares for each MCMC sample.

$$\chi^2 = \chi^2(D|\theta_i), i = 1, 2, ..., N$$

So, the return values is a vector of length $N$ which denotes the number of MCMC iterations except the warming up period. Of course if MCMC is not only one chain, then all samples of chains are used to calculate the chi square.

In the sequel, we use the notations

for a prior $\pi(\theta)$,

posterior $\pi(\theta|D)$,

likelihood $f(D|\theta)$,

parameter $\theta$,

datasets $D$, for example, we can write as follows;

$$\pi(\theta|D) \propto f(D|\theta)\pi(\theta).$$
Let us denote the **posterior MCMC samples** of size $N$ for a given data-set $D$ by

$$\theta_1, \theta_2, \theta_3, \ldots, \theta_N$$

which are drawn from posterior $\pi(\theta|D)$ of given data $D$.

Recall that the chi square goodness of fit statistics $\chi$ depends on the model parameter $\theta$ and data $D$, namely,

$$\chi^2 = \chi^2(D|\theta)$$

The function calculates a vector of length $N$ whose components is given by:

$$\chi^2(D|\theta_1), \chi^2(D|\theta_2), \chi^2(D|\theta_3), \ldots, \chi^2(D|\theta_N),$$

So, the return value is a vector of size $N$.

As an application of this return value $$(\chi^2(D|\theta_i); \ i = 1, \ldots, N),$$ we can calculate the posterior mean of $\chi = \chi(D|\theta)$, namely, we get

$$\chi^2(D) = \int \chi^2(D|\theta) \pi(\theta|D) d\theta,$$

as its Monte Carlo integral

$$\frac{1}{N} \sum_{i=1}^{N} \chi^2(D|\theta_i),$$

In my model, almost all example, result of calculation shows that

$$\int \chi^2(D|\theta) \pi(\theta|D) d\theta > \chi^2(D) \int \theta \pi(\theta|D) d\theta$$

The above inequality is true for all $D$?? I conjecture it.

Revised 2019 August 18 Revised 2019 Sept. 1 Revised 2019 Nov 28

Our data is **2C categories**, that is,

the number of hits : $h[1], h[2], h[3], \ldots, h[C]$ and

the number of false alarms: $f[1], f[2], f[3], \ldots, f[C]$.

Our model has **C+2 parameters**, that is,

the thresholds of the bi normal assumption $z[1], z[2], z[3], \ldots, z[C]$ and

the mean and standard deviation of the signal distribution.

So, the degree of freedom of this statistics is calculated by

No. of categories - No. of parameters - 1 = 2C-(C+2)-1 =C -3.

This differ from Chakraborty’s result C-2. Why ?
Examples

```r
## Not run:
#========================================================================================
# Synthesize the MCMC samples from a dataset.
#========================================================================================
fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1,
ite = 1111,
summary =FALSE,
cha = 2)
#========================================================================================
# The chi square discrepancies are calculated by the following code
#========================================================================================
Chi.Square.for.each.MCMC.samples <- chi_square_goodness_of_fit(fit)
#========================================================================================
# With Warning
#========================================================================================
chi_square_goodness_of_fit(fit)
#========================================================================================
# Without warning
#========================================================================================
chi_square_goodness_of_fit(fit,
h=fit@dataList$h,
f=fit@dataList$f)
#========================================================================================
# Get posterior mean of the chi square discrepancy.
#========================================================================================
m<- mean(Chi.Square.for.each.MCMC.samples)
#========================================================================================
# The author read at 2019 Sept. 1, it helps him. Thanks me!!
```
# Calculate the p-value for the posterior mean of the chi square discrepancy.
#========================================================================================

stats::pchisq(m,df=1)
#========================================================================================

# Difference between chi sq. at EAP and EAP of chi sq.
#========================================================================================

mean( fit@chisquare - chi_square_goodness_of_fit(fit))

## End(Not run)# dottest

---

**chi_square_goodness_of_fit_from_input_all_param**

*Not vetor: The Goodness of Fit (Chi Square) Calculator*

### Description

Chi square goodness of fit statistics for each MCMC sample with a fixed dataset. Our data is 2C categories, that is, the number of hits :h[1], h[2], h[3].....h[C] and the number of false alarms: f[1],f[2], f[3].....f[C].

Our model has C+2 parameters, that is, the thresholds of the bi normal assumption z[1],z[2],z[3].....z[C] and the mean and standard deviation of the signal distribution.

So, the degree of freedom of this statistics is calculated by 2C-(C+2)-1 =C -3.

This differ from Chakraborty’s result C-2. Why ?

### Usage

```r
chi_square_goodness_of_fit_from_input_all_param(
  h,
  f,
  p,
  lambda,
  NL,
  NI,
  ModifiedPoisson = FALSE,
  dig = 3
)
```
Arguments

h  A vector of non-negative integers, indicating the number of hits. The reason why the author includes this variable is to substitute the false alarms from the posterior predictive distribution. In famous Gelman’s book, he explains how to make test statistics in the Bayesian context, and it requires the samples from posterior predictive distribution. So, in this variable author substitute the replication data from the posterior predictive distributions.

f  A vector of non-negative integers, indicating the number of false alarms. The reason why the author includes this variable is to substitute the false alarms from the posterior predictive distribution. In famous Gelman’s book, he explains how to make test statistics in the Bayesian context, and it requires the samples from posterior predictive distribution. So, in this variable author substitute the replication data from the posterior predictive distributions.

p  A vector of non-negative integers, indicating hit rate. A vector whose length is number of confidence levels.

lambda  A vector of non-negative integers, indicating False alarm rate. A vector whose length is number of confidence levels.

NL  An integer, representing Number of Lesions

NI  An integer, representing Number of Images

ModifiedPoisson  Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly,

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows (\(F_c\) denotes the number of false alarms with confidence level \(c\))

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]
where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the \textit{False Positive Fraction (FPF) per lesion.}

On the other hand, if ModifiedPoisson = FALSE (Default), then \textit{False Positive Fraction (FPF)} is given by

$$\frac{F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_4 + F_5}{N_I},$$

$$\frac{F_5}{N_I},$$

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the \textit{False Positive Fraction (FPF) per image.}

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = FALSE) or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE).

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is a redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

dig

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named . . . ??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5,
Details

To calculate the chi square test statistics, the two quantities are needed, that is, data and parameter. In the classical (frequentists) chi square values, as the estimates of parameter, for example, MLE (maximal likelihood estimator) is chosen. In Bayesian sense, the parameter can be taken for all MCMC iterations, that is, parameter is not deterministic and we consider it is a random variable or samples from the posterior distribution. And such samples are obtained in the Hamiltonian Monte Carlo Simulation with the author’s Bayesian Model. Thus we can calculate chi square values with MCMC samples.

Value

A number!! Not list nor dataframe nor vector!! Only A number represent the chi square for your input data.

Examples

```r
## Not run:
# Make a stanfit object (more precisely its inherited S4 class object)
fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1,
ite = 1111,
summary =FALSE,
cha = 2)

# The chi square discrepancies (Goodness of Fit) are calculated
# by the following code with the posterior mean as a parameter.#

NI <- fit@dataList$NI
NL <- fit@dataList$NL
f.observed <- fit@dataList$f
h.observed <- fit@dataList$h
C <- fit@dataList$C

# p <- rstan::get_posterior_mean(fit, par=c("p"))
#lambda <- rstan::get_posterior_mean(fit, par=c("l"))
# Note that get_posterior_mean is not a number but a matrix when
# Chains is not 1.
# So, instead of it, we use
#
# e <- extract_EAP_CI(fit,"l",fit@dataList$C)
lambda <- e$l.EAP

e <- extract_EAP_CI(fit,"p",fit@dataList$C)
p <- e$p.EAP

Chi.Square <- chi_square_goodness_of_fit_from_input_all_param( h = h.observed,
```
Chi square in the case of MRMC at a given dataset and a given parameter.

Description
Given parameter and data, the chi square is calculated.

Usage

```r
chi_square_goodness_of_fit_from_input_all_param_MRMC(
    ppp,
    dll,
    dataList,
    summary = TRUE
)
```

Arguments

- `ppp` An array of `[C,M,Q]`, representing hit rate, where C, M, Q denotes the number of confidences, modalities, readers, respectively.
- `dll` An vector of length C M Q representing false alarm rate, where C, M, Q denotes the number of confidences, modalities, readers, respectively.
**dataList**  
A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of mutiple readers or mutiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. This is a variable in the function \texttt{rstan::sampling()} in which it is named data. For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(  
  h = c(41,22,14,8,1),# number of hits for each confidence level  
  f = c(1,2,5,11,13),# number of false alarms for each confidence level  
  NL = 124,# number of lesions (signals)  
  NI = 63,# number of images (trials)  
  C = 5) # number of confidence,.. the author thinks it can be calculated  
  as the length of h or f ...? ha,why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply \texttt{fit_Bayesian_FROC()}  
such as \texttt{fit_Bayesian_FROC(dataList.Example)}.
To make this \texttt{R} object dataList representing FROC data, this package provides  
three functions:  
\texttt{convertFromJafroc()}  
\texttt{dataset_creator_new_version()}  
\texttt{create_dataset()}  
Enter TP and FP data by \texttt{table}.
Before fitting a model, we can confirm our dataset is correctly formulated by  
using the function \texttt{viewdata()}.

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), dataList is a list consisting  
of \texttt{f,h,NL,NI,C} where \texttt{f,h} are numeric vectors and \texttt{NL,NI,C} are positive  
integers.
\texttt{f} Non-negative integer vector specifying number of false alarms associated  
with each confidence level. The first component corresponding to the highest  
confidence level.
\texttt{h} Non-negative integer vector specifying number of Hits associated with each  
confidence level. The first component corresponding to the highest confidence  
level.
\texttt{NL} A positive integer, representing Number of Lesions.
\texttt{NI} A positive integer, representing Number of Images.
\texttt{C} A positive integer, representing Number of Confidence level.
The detail of these dataset, see the datasets endowed with this package. 'Note  
that the maximal number of confidence level, denoted by \texttt{C}, are included, however,  
Note that confidence level vector \texttt{c} should not be specified. If specified,  
will be ignored, since it is created by \texttt{c <- c(rep(C:1))} in the program and
**chi_square_goodness_of_fit_from_input_all_param_MRMC**

do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

**data Format:**
A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP
*hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <- c(rep(C:1)) automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector c <- c(rep(C:1)) via a table which can be displayed by the function viewdata().

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function fit_Bayesian_FROC(), dataset represented by an R list object representing FROC data must contain components m, q, c, h, f, NL, C, M, Q.

- **C** A positive integer, representing the highest number of confidence level, this is a scalar.
- **M** A positive integer vector, representing the number of modalities.
- **Q** A positive integer, representing the number of readers.
- **m** A vector of positive integers, representing the modality ID vector.
q A vector of positive integers, representing the reader ID vector.
c A vector of positive integers, representing the confidence level. This vector must be made by \texttt{rep(rep(C:1),M*Q)}
h A vector of non-negative integers, representing the number of hits.
f A vector of non-negative integers, representing the number of false alarms.
NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above \texttt{R} object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector,

the function \texttt{viewdata()} shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

\textit{Example data.}  
Multiple readers and multiple modalities (i.e., MRMC)

\begin{tabular}{cccccc}
Modality ID & Reader ID & Confidence levels & No. of false alarms & No. of hits. \\
\textit{m} & \textit{q} & \textit{c} & \textit{f} & \textit{h} \\
1 & 1 & 3 & 20 & 111 \\
1 & 1 & 2 & 29 & 55 \\
1 & 1 & 1 & 21 & 22 \\
1 & 2 & 3 & 6 & 100 \\
1 & 2 & 2 & 15 & 44 \\
1 & 2 & 1 & 22 & 11 \\
2 & 1 & 3 & 6 & 66 \\
2 & 1 & 2 & 24 & 55 \\
2 & 1 & 1 & 23 & 1 \\
2 & 2 & 3 & 5 & 66 \\
2 & 2 & 2 & 30 & 55 \\
2 & 2 & 1 & 40 & 44 \\
\end{tabular}

\* \text{false alarms} = \text{False Positives} = \text{FP}
\* \text{hits} = \text{True Positives} = \text{TP}

\textit{summary} Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the \texttt{R} console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

\textit{Value}  
A list, contains $\chi^2(\text{Data}|\theta)$, where \textit{Data} and $\theta$ are specified by user.
Examples

```r
## Not run:
#========================================================================================
# 0)
#========================================================================================
# Chi square depends on data and model parameter, thus what we have to do is:
# prepare data and parameter
# In the following, we use data named ddd as a dataset to be fitted a model,
# and use posterior mean estimates as model parameter
# To do so, we first prepare MCMC samples under the given data named ddd

fit <- fit_Bayesian_FROC(dataList = ddd, ite = 1111)

#========================================================================================
# 1) hit rate and false alarm rate
#========================================================================================

e <- extract_estimates_MRMC(fit);
dl <- e$dl.EAP;
ppp <- e$ppp.EAP;

#========================================================================================
# 2) Calculates chi square using above hit rate and false alarm rate and data named ddd
#========================================================================================

chi_square_goodness_of_fit_from_input_all_param_MRMC(ppp, dl, ddd)

## End(Not run)# dontrun
```

---

**Chi square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean**

Chi square statistic (goodness of fit) in the case of MRMC at the pair of given data and each MCMC sample

---

**Description**

In the following, we explain what this function calculates.
Let $\chi^2(y|\theta)$ be a chi square goodness of fit statistic which is defined by $(\text{Observed data} - \text{Expectation})^2/\text{Expectation}$.

In MRMC cases, it is defined as follows.

$$
\chi^2(D|\theta) := \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{c=1}^{C} \left( \frac{[H_{c,m,r} - N_L \times p_{c,m,r}(\theta)]^2}{N_L \times p_{c,m,r}(\theta)} + \frac{[F_{c,m,r} - (\lambda_c - \lambda_{c+1}) \times N_L]^2}{(\lambda_c(\theta) - \lambda_{c+1}(\theta)) \times N_L} \right).
$$

where a dataset $D$ consists of the pairs of the number of False Positives and the number of True Positives $(F_{c,m,r}, H_{c,m,r})$ together with the number of lesions $N_L$ and the number of images $N_I$ and $\theta$ denotes the model parameter.

Note that we can rewrite the chi square as follows.

$$
\chi^2(D|\theta) := \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{c=1}^{C} \left( \frac{[H_{c,m,r} - E_{\theta}[H_{c,m,r}]]^2}{E_{\theta}[H_{c,m,r}]} + \frac{[F_{c,m,r} - E_{\theta}[F_{c,m,r}]]^2}{E_{\theta}[F_{c,m,r}]} \right).
$$

So, the chi square has two terms.

1) The first term is the difference of hit and its expectation.
2) The second term is the differences of observed false alarms and its expectation.

In this function, we calculate each terms, separately. So, return values retain these two terms, separately.

In this function, we calculate the following (I) and (II):

(I) A vector ————
Let us denote a collection of posterior MCMC samples for a given dataset $D$ by

$$\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N,$$

namely, each $\theta_i$ is synthesized from posterior $\pi(\theta|D)$, $\theta_i \sim \pi(\theta|D)$.

Substituting these MCMC samples into the above definition of the chi square, we obtain the following vector as a return value of this function.

$$
\chi^2(D|\theta_1), \\
\chi^2(D|\theta_2), \\
\chi^2(D|\theta_3), \\
\vdots \\
\chi^2(D|\theta_N).
$$

(II) A mean of the above vector, namely, the posterior mean of the chi square over all MCMC samples ————
Using the above vector $\chi^2(D|\theta_i); i = 1, ..., N$, the function also calculates the posterior mean of the chi square statistic, namely,

$$\frac{1}{N} \sum_{i=1}^{N} \chi^2(D|\theta_i),$$

which is an approximation of the following integral;

$$\int \chi^2(D|\theta)\pi(\theta|D)d\theta,$$

where $\pi(\theta|D)$ denotes the posterior probability density under the given data $D$. Do not confuse it with the following

$$\chi^2(D|\theta^*).$$

where $\theta^*$ denotes the posterior estimates $\int \theta\pi(\theta|D)d\theta$.

Usage

Chi_square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean(
  StanS4class,
  summary = TRUE,
  dl_is_an_array_of_C_only_and_not_C_M_Q = TRUE
)

Arguments

StanS4class An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC().

summary Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

dl_is_an_array_of_C_only_and_not_C_M_Q A Boolean, if TRUE, then false rate lambda simply denoted by $l$ in R script ($\lambda$) is an vector $l[C]$. If false, then the false alarm rate is an array $l[C,M,Q]$.

Details

This function is implemented by vectorizations and further technics. When the author review this, I find my past work is great,... I forget that I made this. But this function is great.

Revised 2019 Nov 1
A list, calculated by each modality reader and confidence level, and MCMC samples. A one the component of list contains \( \chi^2(Data|\theta_i) ; i=1,2,3,...n \), where \( n \) is the number of MCMC iterations.

Each component of list is an array whose index indicates \([\text{MCMC,Confidence,Modality,Reader}]\). Each component of list is an array whose index indicates \([\text{MCMC,C,M,Q}]\).

To be passed to the calculation of Posterior predictive p value, I need the sum of return value, that is, sum of C,M,Q and resulting quantities construct a vector whose length is the same as the number of MCMC iterations. I love you. I need you. So, to calculate such quantities, the author ... will make a new function.

Also, retains the posterior mean of chi square:

\[
\chi^2(Data) = \int \chi^2(Data|\theta) \pi(\theta|D) d\theta
\]

### Examples

```r
# Not run:
#
#========================================================================================
# 1) Create a fitted model object for data named dd
#========================================================================================
fit <- fit_Bayesian_FROC( ite = 1111, # Number of MCMC iterations
                          cha = 1,
                          dataList = BayesianFROC::dd # This is a MRMC dataset.
                          )

#========================================================================================
# 2) Calculate a chi square and meta data
#========================================================================================
a <- Chi_square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean(fit)

#========================================================================================
# 3) Extract a chi square
#========================================================================================
```
```r
chi.square <- a$chi.square

#========================================================================================
# A case of single reader is special in the programming perspective
# 2020 Feb 24
#========================================================================================

f <- fit_Bayesian_FROC( ite = 1111,
  cha = 1,
  summary = TRUE,
  dataList = dddd,
  see = 123)

Chi_square_goodness_of_fit_in_case_of_MRMC_Posterior_Mean(f)

# Revised 2019 August 19
# 2019 Nov 1

## End(Not run)# dontrun
```

**clearWorkspace**

*Clear Work Space*

**Description**

If functions are masked in global environment, I use this. This function has no variables.

**Usage**

```r
clearWorkspace()
```

**Author(s)**

Issei Tsunoda
Close_all_graphic_devices

Close the Graphic Device

Description

Close the graphic device to avoid errors in R CMD check.

Usage

Close_all_graphic_devices()

Examples

## Not run:
# Open the graphic devices

grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))

# Close the graphic device

## Close_all_graphic_devices()

#'
## End(Not run)# dottest

compare

model comparison

Description

This is a model comparison.

Usage

compare(NI, ite = 1111)
Comparison

Arguments

NI images
ite iteration

Comparison

Description

This is a model comparison.

Usage

comparison(
  Number.of.variation.of.NL,
  Number.of.images,
  ite = 1111,
  DrawCurve = FALSE,
  dig = 3,
  e = 0
)

Arguments

Number.of.variation.of.NL Lesion
Number.of.images images
ite iteration
DrawCurve Logical: TRUE of FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve = TRUE, if not then DrawCurve = FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.
dig To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...???. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.
e exp for false alarms
Description

Calculates the maximum and the minimal values of R hat over all parameters. In addition, it returns a loginal R object whether R hat is good (TRUE) or bad (FALSE).

Usage

ConfirmConvergence(StanS4class, summary = TRUE, digits = 2)

Arguments

- StanS4class: An S4 object of the class stanfit. No need that it is the S4 class stanfitExtended.
- summary: Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.
- digits: A positive integer, indicating digits for R hat statistics.

Details

Evaluates convergence criterion based on only the R hat statistics for a fitted model object. Revised Nov 23.

Value

Logical: TRUE of FALSE. If model converges (all R hat are closed to 1) then it is TRUE, and if not(some R hat is far from 1), then FALSE.

References


See Also

check_rhat(), which is made by Betanalpha.

Examples

## Not run:
#================The first example======================================
#((Primitive way)).
#1) Build the data for a singler reader and a single modality case.
dat <- list(c=c(3,2,1), #Confidence level
# where, c denotes Confidence level, 
# h denotes number of Hits for each confidence level, 
# f denotes number of False alarms for each confidence level, 
# NL denotes Number of Lesions, 
# NI denotes Number of Images,

#2) Fit the FROC model.  
#Since the above dataset "dat" are single reader and single modality, 
#the following function fit the non hierarchical model.  
fit <- BayesianFROC::fit_Bayesian_FROC(dat,ite=1111)

# Where, the variable "ite" specifies the iteration of MCMC samplings. 
# Larger iteration is better.

#3.1) Confirm whether our estimates converge.

ConfirmConvergence(fit)

# By the above R script,  
# the diagnosis of convergence will be printed in the R (R-studio) console.  
# The diagnosis is based on only the R hat statistic.  
# It also return the logical vector indicating whether or not the MCMC converge, 
# if MCMC converges, then the return value is TRUE and if not, then FALSE.  
# This logical return value is used in this package development  
# and the user should not be interested.  
# The following was useful for programming.
Confirm hit rates are correctly made in case of MRMC

#3.2) The return value is TRUE or FALSE.
   x <- ConfirmConvergence(fit)

#3.3) If you do not want to print the results in the R (Studio) console, then
   x <- ConfirmConvergence(fit,summary=FALSE)

# 2019.05.21 Revised.
# 2019.12.02 Revised.

## End(Not run)# dontrun

Confirm_hit_rates_are_correctly_made_in_case_of_MRMC

Check hit rate is defined correctly

Description

Each hit rate is defined by dividing the area under the probability density function into C+1 regions. Thus, the sum of hit rates over all confidence level must be less than 1 which is checked by this function.

This function checks the sum of all hit rate over all confidence levels are less than 1 in case of MRMC

This code confirm the following inequality:
\[ \sum_{cd} \text{ppp}[cd,md,qd] < 1 \]
for each \( cd,md \) ( \( cd =1,2,...,C \), \( md =1,2,...,M \) ).

The return value is an array consisting of logical R objects indicating whether the above inequality is TRUE or FALSE.

2020 Jam

Usage

Confirm_hit_rates_are_correctly_made_in_case_of_MRMC(
   StanS4class.or.An.array.of.ppp
)

Arguments

StanS4class.or.An.array.of.ppp
   A stanfitExtended object or an array of component of hit rate namely ppp
**Value**

A array with logical components. Its dimension costructed by number of readers and modalities.

**Examples**

```r
#========================================================================================
# array: ppp
#========================================================================================

p.truth.array <- hits_rate_creator()

Confirm_hit_rates_are_correctly_made_in_case_of_MRMC(p.truth.array)

## Not run:
#========================================================================================
# fitted model object
#========================================================================================

f <- fit_Bayesian_FROC(dd,ite = 1111)

Confirm_hit_rates_are_correctly_made_in_case_of_MRMC(f)

## End(Not run)
```

---

**convertFromJafroc**

*Convert .xlsx File of Jafroc into R Object*

**Description**

*Convert an FROC dataset*

*from* .xlsx file of *Jafroc*

*into* R object

**Usage**

```r
convertFromJafroc(No.of.Modalities, No.of.readers, No.of.confidence.levels)
```

**Arguments**

- **No.of.Modalities**
  - A positive integer, indicating the number of modalities which is used in an .xlsx file.
- **No.of.readers**
  - A positive integer, indicating the number of readers which is used in an .xlsx file.
- **No.of.confidence.levels**
  - A positive integer, indicating the number of confidence levels which is used in an .xlsx file.
Format

The .xlsx file of Jafroc consists of three sheets named TP, FP, Truth, precisely! Correctly! (other names never be permitted !!)

----------  TP  ----------

A sheet named TP consists of five columns precisely named from the right hand side:

ReaderID, ModalityID, CaseID, LesionID, TP_Rating. NOTE

CaseID  Note that the above word CaseID means the Image ID vectors indicating the ID of radiographs. That is "case = image = radiograph".

the first row  Note that the first row of each sheet of .xlsx file is constructed by the names of column as follows:

An Example of the sheet named TP in a .xlsx file for the Jafroc software

Interpretation of table

Throughout this explanation, we follow the convention that readers are male.

For example, the first row means the first reader (ReaderID=1) correctly find the first lesion (LesionID = 1) in the first image (CaseID = 1) taken by the first modality (ModalityID = 1) with his rating 5 (TP_Rating =5).

Similarly the second row means the first reader (ReaderID=1) correctly find the 4-th lesion (LesionID = 4) in the second image (CaseID = 2) taken by the 2-nd modality (ModalityID = 2) with his rating 4 (TP_Rating = 4).

<table>
<thead>
<tr>
<th>ReaderID</th>
<th>ModalityID</th>
<th>CaseID</th>
<th>LesionID</th>
<th>TP_Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>11</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
A sheet named **FP** consists of four columns *precisely* named from the right hand side: ReaderID, ModalityID, CaseID, FP_Rating. An Example of a sheet named FP in a `.xlsx` file for the Jafroc software.

**Interpretation of table**

For example, the first row means the first reader (ReaderID = 1) makes a false alarm location in the first image (CaseID = 1) taken by the first modality (ModalityID = 1) with his rating 2 (TP_Rating = 2).

Similarly the second row means the first reader (ReaderID = 1) makes a false alarm location in the second image (CaseID = 2) taken by the 2-nd modality (ModalityID = 2) with his rating 1 (TP_Rating = 1).

Similarly the 6-th and 7-th rows mean that the first reader (ReaderID = 1) makes two false alarm location in the second patient (CaseID = 2). The first false alarm is in the image taken by the 1-st modality (ModalityID = 1) with his rating 1 (TP_Rating = 1). The second false alarm is in the image taken by the 3-rd modality (ModalityID = 3) with his rating 2 (TP_Rating = 2).

<table>
<thead>
<tr>
<th>ReaderID</th>
<th>ModalityID</th>
<th>CaseID</th>
<th>FP_Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>21</td>
<td>2</td>
</tr>
</tbody>
</table>

A sheet named **Truth** consists of three columns *precisely* named from the right hand side: CaseID, LesionID, Weight.

An Example of a sheet named Truth in a `.xlsx` file for the Jafroc software.

**Interpretation of table**

<table>
<thead>
<tr>
<th>CaseID</th>
<th>LesionID</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>2</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>3</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>4</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>5</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>6</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>7</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>8</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>9</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>10</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>11</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>12</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>13</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>14</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>15</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>16</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>17</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>18</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>19</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>20</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>21</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>22</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>23</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>24</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>25</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>26</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>27</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>28</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>29</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>30</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>31</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>32</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>33</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>34</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>35</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>36</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>37</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>38</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>39</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>40</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>41</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>42</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>43</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>44</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>45</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>46</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>47</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>48</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>49</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>50</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>51</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>52</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>53</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>54</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>55</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>56</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>57</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>58</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>59</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>60</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>61</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>62</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>63</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>64</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>65</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>66</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>67</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>68</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>69</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>70</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>71</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>72</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>73</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>74</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>75</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>76</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>77</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>78</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>79</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>80</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>81</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>82</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>83</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>84</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>85</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>86</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>87</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>88</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>89</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>90</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>91</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>92</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>93</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>94</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>95</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>96</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>97</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>98</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>99</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>100</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
For example, the first image (CaseID = 1) contains three lesions each of which is named 1,2,3, namely LesionID = 1,2,3. For example, the second image (CaseID = 2) contains two lesions each of which is named 1,2, namely LesionID = 1,2. For example, the third image (CaseID = 3) contains a single lesion named 1, namely LesionID = 1.

<table>
<thead>
<tr>
<th>CaseID</th>
<th>LesionID</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.3333...</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.3333...</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.3333...</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.25</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.3333...</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.3333...</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0.3333...</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.3333...</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0.3333...</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0.3333...</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the weight are used such that each image influences a same effect on the estimates. Without weight, the images including many targets (lesions) has very strong effect on the estimates. To avoid such bias, Jafroc uses weight. In another context, weight would be used to specify more important lesions in each image.

Revised 2019 Dec 13

However, in this package, we do not use the information of weight. Since the theory of the author of this package did not consider such weight. In the future I have to include the notion of weight. Jafroc use the notion of figure of metric as non parametric manner. So, it seems difficult to include it in the Bayesian model, since generally speaking, Bayesian methodology is parametric.

Details

Create a dataset to be passed into the function `fit_Bayesian_FROC`. Convert an Excel file whose extension is `.xlsx` of Jafroc format to an R object representing FROC data to which we will apply functions in this package such as `fit_Bayesian_FROC()`.
The return values include a list which can be passed to the function `fit_Bayesian_FROC`. For data of Jafroc, running this function, we immediately can fit the author’s Bayesian FROC model to this return value.

The Jafroc software’s format consists of suspicious locations marked by readers and true locations. Such data is redundant for our Bayesian statistical models. So, we reduce the information of data to the number of false positives and number of hits for each confidence levels by this function.

Data can be calculated from the following Jafroc data, in which there are more information than TP and FP. In fact, in the Jafroc data, the FP and TP are counted for each images, each lesions etc. So, it has more information.

It causes limitation of our model. So, our model start to fit a model to the reduced data from Jafroc. So, the redunction will cause the non accuracy evaluation of the observer performance. The future research I should start the Jafroc formulation.

### Value

A list, representing FROC data.

### References

Bayesian Models for Free-response Receiver Operating Characteristic Analysis, pre-print

### See Also

Rjafroc, which is unfortunately not on CRAN, now 2019 Jun 19. Or JAFROC software in the Chakarboty’s Web page. Unfortunately, this software is no longer supported.

### Examples

```r
## Not run:
## Only run examples in interactive R sessions
if (interactive()) {
  #========================================================================================
  # Example for convert the Jafroc data to the BayesianFROC
  #========================================================================================

  # Work Flow of this example

  # step 0) Prepare Jafroc .xlsx file contained in this package
  # step 1) Convert the .xlsx file obtained in step 0)
  # step 2) Fit a model to data object obtained in step 1)
```
# step 0) Make a Jafroc data as an example dataset

# If you can search the xlsx file named JAFROC_data.xlsx
# in the director "inst/extdata" of this package,
# Then this step 0) is redundant. The author prepare this example for the people who
# cannot search the xlsx file in the "inst/extdata" of this package.

# By an xlsx file named JAFROC_data.xlsx in the director "inst/extdata" of this package,
# we can reconstruct it as follows:(If someone can obtain the Excel file
# from the path BayesianFROC/inst/extdata/JAFROC_data.xlsx, then the following code
# is not required to run. If searching bother you, then run the R script to obtain the
# Excel file.)
# I do not know how to users refer the JAFROC_data.xlsx in this package,
# so I provide it by making the same xlsx file as the JAFROC_data.xlsx.

# Note that JAFROC_data.xlsx cannot remove,
# if it is removed, then devtools::run_examples() make an error.

Truth <- readxl::read_excel(system.file("extdata",
  "JAFROC_data.xlsx",
  package="BayesianFROC"),
  sheet = "Truth")
##### utils::View(Truth)

TP <- readxl::read_excel( system.file("extdata",
  "JAFROC_data.xlsx",
  package="BayesianFROC"),
  sheet = "TP")
##### utils::View(TP)

FP <- readxl::read_excel( system.file("extdata",
  "JAFROC_data.xlsx",
  package="BayesianFROC"),
  sheet = "FP")
##### utils::View(FP)

sample <- list(TP=TP,FP=FP,Truth=Truth)
openxlsx::write.xlsx(sample,"JafrocDatasetExample.xlsx")
convertFromJafroc

tcltk::tkmessageBox(
message="A file named
JafrocDatasetExample.xlsx

is created in the working directory")

# Now, we obtain an excel file named "JafrocDatasetExample.xlsx", which is same as
# the JAFROC_data.xlsx.
# whose format is available in the Jafroc software developed by Chakraborty.
# If you use your data, your data must has same format of "JafrocDatasetExample.xlsx".
# Note that other excel data must comply with the above format.

# Note that if you have an excel file
# which is formulated correctly for our package,
# this process does not need.

# (0) From the above, we obtain "JafrocDatasetExample.xlsx"
# which is the multiple reader and multiple modality dataset
# for Jfroc analysis which is NOT implemented in our package,
# but Chakraborty's software called Jafroc or the R package Rjafroc.

#========================================================================================
# step 1) Convert a Jafroc data
#========================================================================================

# (1) Using "JafrocDatasetExample.xlsx" as an example excel file,
# we run the function to convert the excel file from Jafroc format
# to our format:

dataList <- convertFromJafroc(
    No.of.Modalities =5,
    No.of.readers =4,
    No.of.confidence.levels = 5
   )

# In the variable, there is no xlsx file, since it is selected by interactive manner.
# So, please select the xlsx file obtained in step 0) or your own Jafroc
# .xlsx file.

#========================================================================================
# step 2) Fitting a model to data converted from Jafroc
(2) Now, we obtain a list of an FROC dataset as an R object named "dataList". Using this, we can fit a model to the dataset by the following code.

```r
fit <- fit_Bayesian_FROC(dataList)
```

### Only run examples in interactive R sessions

## End(Not run)

# Revised 2019. Jun 19
# Revised 2019. Dec 13
# Revised 2020 Feb
# Revised 2020 April

CoronaVirus_Disease_2019

**Who should be inspected?**

**Description**

Even if a diagnosis test with respect to "all" said that it is positive, however the result cannot be correct in high probability. If we test no suspicious people, then it reduce our resource of diagnosis test and when some suspicious people needs the test, we cannot do the test.

So, the diagnosis test should be done for the suspicious people only. Not should be done for all people including no suspicious people. The medical resource is finite, we should use it for more optimal way.

**Usage**

```r
CoronaVirus_Disease_2019(N, n, se, sp)
```

**Arguments**

- `N`: The number of population, including diseased and non-diseased people
- `n`: The number of diseased population
- `se`: Sensitivity of a diagnostic test
- `sp`: Specificity of a diagnostic test
Details

<table>
<thead>
<tr>
<th>Diagnosis \ truth</th>
<th>Diseased</th>
<th>Non-diseased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>se*n</td>
<td>(N - n)(1 - sp)</td>
</tr>
<tr>
<td>Negative</td>
<td>(1-se)*n</td>
<td>(N - n)sp</td>
</tr>
</tbody>
</table>

For example, if prevalence is 0.0001, population is 10000, specificity = 0.8, sensitivity = 0.9, then the table is the following.

We can calculates the probability of the event that a one whose diagnosis is positive is really diseased is

\[
\frac{9}{1998 + 9} = \frac{9}{1998 + 9} = 0.00448
\]

<table>
<thead>
<tr>
<th>Diagnosis \ truth</th>
<th>Diseased</th>
<th>Non-diseased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>9</td>
<td>1998</td>
</tr>
<tr>
<td>Negative</td>
<td>1</td>
<td>7992</td>
</tr>
</tbody>
</table>

\[
n = 10 \quad N - n = 10000 - 10
\]

**Value**

Probability which is between 0 and 1. I you want to get percent, then it is 100 times the return value.

\[
Prob(Truth = diseased|Diagnosis = Positive) = \frac{Se \times n}{Se \times n + (N - n) \times (1 - sp)}
\]

**Examples**
CoronaVirus_Disease_2019_prevalence

\[ \text{CoronaVirus\_Disease\_2019(10000,10,0.9,0.8)} \]
\[ 9/(1998+9) \]

---

**Description**

Even if we test all people, the result is true with very low probabilities.

**Usage**

\[ \text{CoronaVirus\_Disease\_2019\_prevalence(pre, se, sp)} \]

**Arguments**

- **\( \text{pre} \)**: Prevalence of population

\[
\text{Prob(Truth = diseased|Diagnosis = Positive)} = \frac{\text{Se} \times \text{pre}}{\text{Se} \times \text{pre} + (1 - \text{pre}) \times (1 - \text{sp})}
\]

- **\( \text{se} \)**: Sensitivity of a diagnostic test

- **\( \text{sp} \)**: Specificity of a diagnostic test

**Details**

<table>
<thead>
<tr>
<th>Diagnosis \ truth</th>
<th>Diseased</th>
<th>Non-diseased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>se*n</td>
<td>(N - n)(1 - sp)</td>
</tr>
<tr>
<td>Negative</td>
<td>(1-se)*n</td>
<td>(N - n)sp</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>N - n</td>
</tr>
</tbody>
</table>

For example,

if prevalence is 0.0001,
population is 10000,
specificity = 0.8,
sensitivity = 0.9, then the table is the following.

We can calculates the probability of the event that a one whose diagnosis is positive is really diseased is

\[
\frac{9}{1998 + 9} = \frac{9}{(1998 + 9)} = 0.00448 \text{ percent}
\]

<table>
<thead>
<tr>
<th>Diagnosis \ truth</th>
<th>Diseased</th>
<th>Non-diseased</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>9</td>
<td>1998</td>
</tr>
<tr>
<td>Negative</td>
<td>1</td>
<td>7992</td>
</tr>
</tbody>
</table>

\[n = 10\quad N - n = 10000 - 10\]

Value
same as \texttt{CoronaVirus\_Disease\_2019()} \n
See Also
\texttt{CoronaVirus\_Disease\_2019()} \n
Examples
\texttt{CoronaVirus\_Disease\_2019\_prevalence(0.0001, 0.9, 0.8)}
\texttt{CoronaVirus\_Disease\_2019\_prevalence(0.03, 0.9, 0.8)}
\texttt{CoronaVirus\_Disease\_2019\_prevalence(0.3, 0.9, 0.8)}

# If Sensitivity and Specificity is larger, then, the probability is also larger

x <- \texttt{stats::runif(100, 0, 1)}
y <- \texttt{CoronaVirus\_Disease\_2019\_prevalence(0.1, x, x)}
dark\_theme(4)
plot(x, y)

# If Sensitivity and Specificity is larger, then, the probability is also larger
create_dataList_MRMC

# If the prevalence is larger, then, the probability is also larger
#==================================================================================================

x <- stats::runif(100,0,1)
y <- CoronaVirus_Disease_2019_prevalence(x,0.9,0.9)
dark_theme(4)
plot(x,y)

create_dataList_MRMC  Creates a Single Dataset in Case of MRMC

Description

From a given model parameter, creates a FROC dataset in case of multiple readers and multiple modality, breafly MRMC. The dataset consists of the number of hits and false alarms and ID vectors of readers, modalites, confidences, etc.

The created dataset is a list (which can be passed to fit_Bayesian_FROC()). Model parameters are thresholds, mean and standard deviation of signal Gaussian.

Usage

create_dataList_MRMC(
  z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  NI = 57,
  NL = 142,
  ModifiedPoisson = FALSE,
  seed = 123,
  summary = FALSE
)

Arguments

z.truth  Vector ( of dimension C) represents the thresholds.
mu.truth array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.
v.truth  array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.

NI  The number of images,
NL  The number of lesions,
Logical, that is TRUE or FALSE.

If \( \text{ModifiedPoisson} = \text{TRUE} \), then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.

Similarly,

If \( \text{ModifiedPoisson} = \text{TRUE} \), then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes , now, it is omitted from this package, because the size of vignettes are large.)

If \( \text{ModifiedPoisson} = \text{TRUE} \), then the False Positive Fraction (FPF) is defined as follows (\( F_c \) denotes the number of false alarms with confidence level \( c \))

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \) (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]
\[
\frac{F_5}{N_I},
\]
where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the \textit{False Positive Fraction (FPF)} \textbf{per image}.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textbf{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

\begin{description}
\item[seed] The seed for creating hits which are synthesized by the binomial distributions with the specified seed.
\item[summary] Logical: \texttt{TRUE} of \texttt{FALSE}. Whether to print the verbose summary. If \texttt{TRUE} then verbose summary is printed in the \texttt{R} console. If \texttt{FALSE}, the output is minimal. I regret, this variable name should be verbose.
\end{description}

Details
Specifying model parameters, we can replicates fake datasets. Different seed gives different fake data. Model parameters are the following.

\begin{description}
\item[\texttt{z.truth}] \texttt{mu.truth} \texttt{v.truth}.
\item[Probablity law of hits] Random variables of hits are distributed as follows.
\[
H_{5,m,r} \sim Binomial(p_{5,m,r}, N_L),
\]
then \( H_{4,m,r} \) should be drawn from the binomial distribution with remaining targets.
\[ H_{4,m,r} \sim \text{Binomial}(\frac{p_{4,m,r}}{1 - p_{5,m,r}}, N_L - H_{5,m,r}). \]

Similarly, because we already found \( H_{4,m,r} \) and \( H_{5,m,r} \), the remained targets are \( N_L - H_{5,m,r} - H_{4,m,r} \). Thus it natural to assume the following. Note that the hit rate is defined so that the resulting model satisfy certain equations which is not explained here.

\[ H_{3,m,r} \sim \text{Binomial}(\frac{p_{3,m,r}}{1 - p_{5,m,r} - p_{4,m,r}}, N_L - H_{5,m,r} - H_{4,m,r}). \]

\[ H_{2,m,r} \sim \text{Binomial}(\frac{p_{2,m,r}}{1 - p_{5,m,r} - p_{4,m,r} - p_{3,m,r}}, N_L - H_{5,m,r} - H_{4,m,r} - H_{3,m,r}). \]

\[ H_{1,m,r} \sim \text{Binomial}(\frac{p_{1,m,r}}{1 - p_{5,m,r} - p_{4,m,r} - p_{3,m,r} - p_{2,m,r}}, N_L - H_{5,m,r} - H_{4,m,r} - H_{3,m,r} - H_{2,m,r}). \]

**Probability law of false alarms**

\[ F_{5,m,r} \sim \text{Poisson}(q_{5,m,r}N_X), \]

\[ F_{4,m,r} \sim \text{Poisson}(q_{4,m,r}N_X), \]

\[ F_{3,m,r} \sim \text{Poisson}(q_{3,m,r}N_X), \]

\[ F_{2,m,r} \sim \text{Poisson}(q_{2,m,r}N_X), \]

\[ F_{1,m,r} \sim \text{Poisson}(q_{1,m,r}N_X), \]

where subscripts \( m, r \) mean the \( m \)-th modality and the \( r \)-th reader, respectively. Note that \( N_X \) is the following two cases.

1) \( N_X = N_L \) (The number of lesions), if \( \text{ModifiedPoisson} = \text{TRUE} \).
2) \( N_X = N_I \) (The number of images), if \( \text{ModifiedPoisson} = \text{FALSE} \).

We fix the \( N_X = N_L \) or \( N_X = N_I \) through out this paper.

The rate \( p_{c,m,r} \) and \( q_{c,m,r} \) are calculated from the model parameters.

\( z \).truth

\( mu \).truth

\( v \).truth.

By specifying these model parameters we can make a fake dataset consisting of hit data \( H_{c,m,r} \) false alarm data \( F_{c,m,r} \) for each \( c, m, r \).
See Also

- `chi_square_at_replicated_data_and_MCMC_samples_MRMC()`
- `replicate_MRMC_dataList()`

(To make many MRMC datasets, see `replicate_MRMC_dataList()`)

Examples

```r
## Not run:

dataList <- create_dataList_MRMC()

fit_Bayesian_FROC(dataList, 
                   summary = FALSE, 
                   ite = 1111)

# In the above example, we use a default values for true parameters for 
# the distributions. The reason why the default values exists is difficulty 
# for the user who is not familiar with FROC data nor konws the resions 
# in which parameters of FROC model move. 
# So, in the Bayesian model is merely model for FROC data. 
# If user input the abnormal data, then the model does not fit nor converge 
# in the Hamiltonian Monte Carlo simulations.

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC())
```

```r
#========================================================================================
# plot various MRMC datasets with fixed signal distribution but change thresholds
#========================================================================================

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(0.1,
                                         0.2,
                                         0.3,
                                         0.4)
                                         ))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-0.1,
                                         0.2,
                                         0.3,
                                         0.4)
                                         ))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1,
                                         0.2,
                                         0.3,
                                         0.4)
                                         ))
```
create_dataList_MRMC

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, 0.2, -0.3, 0.4))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, 0.2, 0.3))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, 1.2, 2.3))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, -0.5, 0, 1.2, 2.3, 3.3, 4))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, -0.5, 0, 1.2, 2.3, 3.3, 4, 5, 6))

plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, -0.5, 0, 1.2, 2.3, 3.3, 4, 5, 6))
create_dataset

```r
plot_FPF_and_TPF_from_a_dataset(create_dataList_MRMC( z.truth = c(-1, -0.5, 0, 1.2, 2.3, 3.3, 4, 5, 6, 7, 8, 9, 10)
))
```

#========================================================================================
# Smoothing of Scatter Plot for FPF and TPF
#========================================================================================

```r
v <- v_truth_creator_for_many_readers_MRMC_data(M=1, Q=17)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1, Q=17)
d <- create_dataList_MRMC(mu.truth = m, v.truth = v)
d <- metadata_to_fit_MRMC(d)
df <- data.frame(FPF = d$ffN, TPF = d$hhN)
# require(graphics)
dark_theme()
graphics::plot(df, main = "lowess(cars)")
graphics::lines(stats::lowess(df), col = 2)
graphics::lines(stats::lowess(df, f = .2), col = 3)
graphics::legend(5, 120, c(paste("f = ", c("2/3", ", .2"))), lty = 1, col = 2:3)
```

## End(Not run)

create_dataset

Create a dataset

Description

Create a dataset to apply the function `fit_Bayesian_FROC`.
Usage

create_dataset()

Details

This is an interactive creator of an FROC dataset. Using this return value, we can fit a FROC model to data by applying the function `fit_Bayesian_FROC` in this package.

To tell the truth, the author never use this function to create datset. So,... this function is not so good.

Value

A list of FROC data to which we fit a FROC model.

2019 Dec 12

Examples

```r
## Not run:
## Only run examples in interactive R sessions
if (interactive()) {

    create_dataset()

}### Only run examples in interactive R sessions
## End(Not run)
```

Credible_Interval_for_curve

`Credible_Interval_for_curve` is a function to draw FROC curves which means credible interval.

Description

Plot FROC curves based on two parameters a and b.

Usage

```r
Credible_Interval_for_curve(
    dataList,
    StanS4class.fit_MRMC_versionTWO,
    mesh.for.drawing.curve = 10000,
    upper_x = upper_x,
    upper_y = upper_y,
    lower_y = lower_y
)
```
Arguments

dataList  A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data. For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
    h = c(41, 22, 14, 8, 1), # number of hits for each confidence level
    f = c(1, 2, 5, 11, 13), # number of false alarms for each confidence level
    NL = 124, # number of lesions (signals)
    NI = 63, # number of images (trials)
    C = 5) # number of confidence, the author thinks it can be calculated as the length of h or f ...? ha, why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).

To make this R object dataList representing FROC data, this package provides three functions:

- convertFromJafroc() If data is a JAFROC xlsx formulation.
- dataset_creator_new_version() Enter TP and FP data by table.
- create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

f  Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

h  Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

NL  A positive integer, representing Number of Lesions.

NI  A positive integer, representing Number of Images.

C  A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified,
will be ignored, since it is created by \( c <-c(rep(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

**Data Format:**

*A single reader and a single modality case*

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( c )</td>
<td>( f )</td>
<td>( h )</td>
</tr>
<tr>
<td>definitely present</td>
<td>( c[1] = 5 )</td>
<td>( f[1] = F_5 = 1 )</td>
<td>( h[1] = H_5 = 41 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>( c[5] = 1 )</td>
<td>( f[5] = F_1 = 13 )</td>
<td>( h[5] = H_1 = 1 )</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, *thus* each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c <-c(rep(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c <-c(rep(C:1)) \) via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an \( R \) list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

\( C \) A positive integer, representing the highest number of confidence level, this is a scalar.

\( M \) A positive integer vector, representing the number of modalities.

\( Q \) A positive integer, representing the number of readers.
Credible Interval for curve

m A vector of positive integers, representing the modality ID vector.
q A vector of positive integers, representing the reader ID vector.
c A vector of positive integers, representing the confidence level. This vector must be made by rep(rep(C:1),M*Q)
h A vector of non-negative integers, representing the number of hits.
f A vector of non-negative integers, representing the number of false alarms.
NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP
*hits = True Positives = TP

StanS4class.fit_MRMC_versionTWO
A return value of fit_MRMC_versionTWO.
mesh.for.drawing.curve
A positive large integer, indicating number of dots drawing the curves, Default =10000.
upper_x  A positive real number, indicating the frame size of drawing picture.
upper_y  A positive real number, indicating the frame size of drawing picture.
lower_y  A positive real number, indicating the frame size of drawing picture.

d                     Data: A Single Reader and A Single Modality

Description
A list, representing FROC data. This is used to build a hierarchical FROC model. This data is exactly same as dataList.Chakra.1.

Details
This data is same as dataList.Chakra.1.with.explantation. The author name it d for the sake of simplicity, that is, it is easy to write, because only one character!!

Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also
dataList.Chakra.1.with.explantation which is exactly same in this data d.

dark_theme                     Dark Theme

Description
Executing this function before plotting, the plot region becomes the dark theme.

Usage
dark_theme(type = 1)

Arguments
  type  An integer
data.bad.fit

Details

A function specifies the color in graphic devices.

Value

Nothing

Examples

## Not run:

```r
dark_theme(1)
graphics::plot(c(1,2,3),c(1,2,3))

dark_theme(2)
graphics::plot(c(1,2,3),c(1,2,3))

# 2019.05.21 Revised.

dark_theme(3)
graphics::plot(c(1,2,3),c(1,2,3))

dark_theme(4)
graphics::plot(c(1,2,3),c(1,2,3))

# 2019 Oct 19 Revised

## End(Not run)
```

---

data.bad.fit  
**Data:** Single reader and Single modality

Description

A list, representing FROC data consisting of hits, false alarms, number of lesions, number of images, to which we fit a FROC model.
Format

A list consists of two integer vectors \( f, h \) and three integers \( NL, NI, C \).

- \( f \) Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- \( h \) Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- \( NL \) A positive integer, representing Number of Lesions.
- \( NI \) A positive integer, representing Number of Images.
- \( C \) A positive integer, representing Number of Confidence level.

Contents:

A single reader and single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>4 11 11</td>
<td></td>
</tr>
<tr>
<td>probably present</td>
<td>3 1 97</td>
<td></td>
</tr>
<tr>
<td>subtle</td>
<td>2 14 32</td>
<td></td>
</tr>
<tr>
<td>very subtle</td>
<td>1 74 31</td>
<td></td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(rep(C:1)) \) automatically in the program and it does not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function \texttt{viewdata()}.

Note that The format for the above example data must be made by the following forms:

\[
\text{dat} <- \text{list(}
\text{h} = \text{c}(11, 97, 32, 31),
\text{f} = \text{c}(11, 1, 14, 74),
\text{NL} = 259,
\text{NI} = 57,
\text{C} = 4)
\]
This object `dat` can be passed to the function `fit_Bayesian_FROC()` as the following manner: `fit_Bayesian_FROC(dat)`.

**Details**

This data-set is very bad fitting. Even if the MCMC sampling is very good, however, the FPF and TPF are not on the FROC curve.

Note that the maximal number of confidence level, denoted by `C`, are included, however, confidence level vector `c` should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` in the program and it does not refer from user input data, where `C` is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector `c`.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

I love you.

**See Also**

`viewdata()`, which shows your data confortably by `knitr::kable()`.

---

**Description**

This is used to build a hierarchical FROC model.

**Details**

This data is fictitious.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

The author’s preprint
data.MultiReaderMultiModality

*Multiple reader and Multiple modality data*

**Description**

This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.Web.

**Details**

This data appeared in Chakraborty’s paper (1988)

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

---

data.nonconverge.srsc  
**Non-Convergent Data: Single reader and Single modality**

**Description**

A list, representing **non-convergent** FROC data (which does not converge in the sense of R hat) of hits and false alarms. This is used to build a non-hierarchical FROC model.

**Format**

A list consists of two integer vectors \( f, h \) and three integers \( NL, NI, C \).

- \( f \) Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- \( h \) Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- \( NL \) A positive integer, representing Number of Lesions.
- \( NI \) A positive integer, representing Number of Images.
- \( C \) A positive integer, representing Number of Confidence level.

**Contents:**

* A single reader and single modality case*
**false alarms** = False Positives = FP  
**hits** = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset.  

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored , since it is created by \( c<-c(rep(C:1)) \) automatically in the program and it does not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Note that The format for the above example data must be made by the following forms:

```r
dat <- list(h = c(99, 0, 0),  
f = c(88, 0, 0),  
NL = 111,  
NI = 111,  
C = 3)
```

This object `dat` can be passed to the function `fit_Bayesian_FROC()` as the following manner

```r
fit_Bayesian_FROC(dat).
```

**Details**

Note that the maximal number of confidence level, denoted by \( C \), are included, however, confidence level vector \( c \) should not be specified. If specified, will be ignored , since it is created by \( c<-c(rep(C:1)) \) in the program and it does not refer from user input data, where \( C \) is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector \( c \).  

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**See Also**

`dataList.Chakra.1.with.explanation`
**dataList.Chakra.1**

**Data: A Single Reader and A Single Modality**

**Description**

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

**Details**

This data appeared in Chakraborty’s paper (1988)

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

**See Also**

[datavist.Chakra.1.with.explanation](#)
Contents:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probably present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>questionable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NI=57, NL=259

In R console ->

c f h

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>97</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>74</td>
<td>31</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c<-c(rep(C:1)) automatically in the program and it does not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector c.

Note that the format for the above example data must be made by the following forms:

dat <- list(h=c(97,32,31), f=c(1,14,74), NL=259, NI=57, C=3)

This object dat can be passed to the function *fit_Bayesian_FROC()* as the following manner

fit_Bayesian_FROC(dat).

Details

Note that the maximal number of confidence level, denoted by C, are included, however, confidence level vector c should not be specified. If specified, will be ignored, since it is created by c<-c(rep(C:1)) in the program and it does not refer from user input data, where C is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector c.

This data appeared in Chakraborty’s paper (1988).
Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also
dataList.Chakra.1.with.explanation

dataList.Chakra.1.with.explanation

Data: A Single Reader and A Single Modality

Description
A list, representing an FROC dataset consisting of hits, false alarms, number of lesions, number of images. We fit a FROC model to the data.

Format
A list consists of two integer vectors \( f, h \) and three integers \( NL, NI, C \).

\( f \) Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

\( h \) Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

\( NL \) A positive integer, representing Number of Lesions.

\( NI \) A positive integer, representing Number of Images.

\( C \) A positive integer, representing Number of Confidence level.

Contents:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probably present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>questionable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NI=57, NL=259</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In R console ->

\[
\begin{array}{ccc}
\text{confidence level} & \text{No. of false alarms} & \text{No. of hits} \\
\hline
\text{definitely present} & 3 & 1 & 97 \\
\text{probably present} & 2 & 14 & 32 \\
\text{questionable} & 1 & 74 & 31 \\
\end{array}
\]
* false alarms = False Positives = FP
* hits = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) automatically in the program and it does not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function viewdata().

Note that The format for the above example data must be made by the following forms:

```r
dat <- list(
  h = c(97, 32, 31),
  f = c(1, 14, 74),
  NL = 259,
  NI = 57,
  C = 3)
```

This object dat can be passed to the function fit_Bayesian_FROC() as the following manner

```r
fit_Bayesian_FROC(dat).
```

**Details**

Note that the maximal number of confidence level, denoted by C, are included, however, confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) in the program and it does not refer from user input data, where C is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector c.

This data appeared in Chakraborty’s paper (1988). This dataset is same as dataList.Chakra.1. The difference between two dataset is only explanations for vectors. That is I attached the name for each vector by names(). I hope it help user for understanding what it is.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**Source**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.
Data: A Single Reader and A Single Modality

Description

A list, representing FROC data consisting of hits, false alarms, number of lesions, number of images. We fit a FROC model to the data.

Format

A list consists of two integer vectors \( f \), \( h \) and three integers \( NL, NI, C \).

\( f \)  Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

\( h \)  Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

\( NL \)  A positive integer, representing Number of Lesions.

\( NI \)  A positive integer, representing Number of Images.

\( C \)  A positive integer, representing Number of Confidence level.

Contents:

A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=57, NL=269</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td></td>
<td>( c )</td>
<td>( f )</td>
</tr>
<tr>
<td>definitely present</td>
<td>3</td>
<td>4</td>
<td>122</td>
</tr>
<tr>
<td>probably present</td>
<td>2</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>questionable</td>
<td>1</td>
<td>44</td>
<td>20</td>
</tr>
</tbody>
</table>

* false alarms = False Positives = FP

* hits = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicous location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c <- c(rep(C:1)) \) automatically in the program and it does not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function \texttt{viewdata}().
Note that The format for the above example data must be made by the following forms:

```r
dat <- list(
  h = c(122, 31, 20),
  f = c(4, 13, 44),
  NL = 269,
  NI = 57,
  C = 3)
```

This object `dat` can be passed to the function `fit_Bayesian_FROC()` as the following manner `fit_Bayesian_FROC(dat)`.

**Details**

Note that the maximal number of confidence level, denoted by `C`, are included, however, confidence level vector `c` should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` in the program and it does not refer from user input data, where `C` is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector `c`.

This data appeared in Chakraborty’s paper (1988).

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

**See Also**

dataList.Chakra.1.with.explanation

dataList.Chakra.3  Data: A Single Reader and A Single Modality

**Description**

A list, representing FROC data consisting of hits, false alarms, number of lesions, number of images. We fit a FROC model to the data.
Format

A list consists of two integer vectors \( f \), \( h \) and three integers \( NL \), \( NI \), \( C \).

\( f \)  Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

\( h \)  Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

\( NL \)  A positive integer, representing Number of Lesions.

\( NI \)  A positive integer, representing Number of Images.

\( C \)  A positive integer, representing Number of Confidence level.

Contents:

A single reader and a single modality case

\[
\begin{array}{ccc}
\text{confidence level} & \text{No. of false alarms} & \text{No. of hits} \\
\text{c} & \text{f} & \text{h} \\
\text{definitely present} & 3 & 2 & 96 \\
\text{probably present} & 2 & 16 & 39 \\
\text{questionable} & 1 & 48 & 13 \\
\end{array}
\]

* false alarms = False Positives = FP

* hits = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored , since it is created by \( c <-c(\text{rep}(C:1)) \) automatically in the program and it does not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function \text{viewdata}().

Note that The format for the above example data must be made by the following forms:

\[
\text{dat <- list(} \\
\text{ h = c(96,39,13) ,} \\
\text{ f = c(2,16,48),} \\
\text{ NL = 269,} \\
\text{ NI = 57,} \\
\text{ C = 3)} \\
\text{)}
\]
This object `dat` can be passed to the function `fit_Bayesian_FROC()` as the following manner:

`fit_Bayesian_FROC(dat)`.

**Details**

Note that the maximal number of confidence level, denoted by `C`, are included, however, confidence level vector `c` should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` in the program and it does not refer from user input data, where `C` is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector `c`.

This data appeared in Chakraborty’s paper (1988).

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

**See Also**

dataList.Chakra.1.with.explanation

dataList.Chakra.4

---

**Description**

A list, representing FROC data consisting of hits, false alarms, number of lesions, number of images. We fit a FROC model to the data.

**Format**

A list consists of two integer vectors `f`, `h` and three integers `NL`, `NI`, `C`.

- `f` Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- `h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- `NL` A positive integer, representing Number of Lesions.
- `NI` A positive integer, representing Number of Images.
- `C` A positive integer, representing Number of Confidence level.

**Contents:**

A single reader and a single modality case
NI=50, NL=397

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>probably present</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>160</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` automatically in the program and it does not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Note that The format for the above example data must be made by the following forms:

```r
dat <- list(
  h = c(160, 25, 15, 7),
  f = c(8, 16, 18, 13),
  NL = 397,
  NI = 50,
  C = 4)
```

This object dat can be passed to the function `fit_Bayesian_FROC()` as the following manner `fit_Bayesian_FROC(dat)`.

**Details**

Note that the maximal number of confidence level, denoted by C, are included, however, confidence level vector c should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` in the program and it does not refer from user input data, where C is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector c.

This data appeared in Chakraborty’s paper (1988).

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>
References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

---

An FROC Data of Multiple-Reader and Multiple-Modality

Description

A list, representing FROC data in case of MRMC.

Details

This data is based on an example data of Chakraborty’s JAFROC software. The author have calculated hits and false alarms from this example data formulated for Jafroc.

Contents:

Multiple readers and Multiple modalities case, i.e., MRMC case

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>m</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>17</td>
<td>14</td>
</tr>
</tbody>
</table>
2 1 5 1 52
2 1 4 1 25
2 1 3 21 13
2 1 2 24 4
2 1 1 23 1
2 2 5 1 27
2 2 4 1 28
2 2 3 5 29
2 2 2 30 1
2 2 1 40 0
2 3 5 2 53
2 3 4 19 29
2 3 3 31 13
2 3 2 56 2
2 3 1 42 4
2 4 5 2 9
2 4 4 0 16
2 4 3 2 22
2 4 2 30 43
2 4 1 32 14
3 1 5 1 43
3 1 4 7 29
3 1 3 13 11
3 1 2 28 6
3 1 1 19 0
3 2 5 0 18
3 2 4 1 29
3 2 3 7 21
3 2 2 7 0
3 2 1 31 0
3 3 5 7 43
3 3 4 15 29
3 3 3 28 6
3 3 2 41 7
3 3 1 9 1
3 4 5 0 10
3 4 4 2 14
3 4 3 5 19
3 4 2 24 32
3 4 1 31 23
4 1 5 1 61
4 1 4 4 19
4 1 3 18 12
4 1 2 21 9
4 1 1 23 3
4 2 5 1 16
4 2 4 1 29
4 2 3 0 34
<table>
<thead>
<tr>
<th>Value</th>
<th>Identification</th>
<th>Value</th>
<th>Identification</th>
<th>Value</th>
<th>Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>11</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>35</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>6</td>
<td>52</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>14</td>
<td>29</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>37</td>
<td>10</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>36</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>23</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>18</td>
<td>43</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>25</td>
<td>15</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>35</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>19</td>
<td>18</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>23</td>
<td>9</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>18</td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>27</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>6</td>
<td>24</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>30</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>34</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>25</td>
<td>33</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>40</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>29</td>
<td>13</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>24</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>21</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>24</td>
<td>35</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>32</td>
<td>15</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**References**

Example data of Jafroc software

**See Also**

`dataList.Chakra.Web.orderd d`
### Description
To be fitted an FROC model.

### Details
This data was calculated from an example dataset which appears in Chakraborty’s JAFROC. The author has ordered the dataset `dataList.Chakra.Web` (or `dd`) so that the modality ID means the order of AUC. For example modality ID = 1 means its AUC is the highest. modalityID = 2 means that its AUC is the secondly high AUC.

So, let $A_1, A_2, A_3, A_4, A_5$ be the AUCs for the modality ID 1, 2, 3, 4, 5, respectively.

Then it follows that

\[ A_1 > A_2 > A_3 > A_4 > A_5. \]

So, modality ID in this dataset corresponds the modality ID of `dataList.Chakra.Web` (or `dd`) as (4 2 1 5 3).

That is, let us denote the modality ID of this dataset (1’,2’,3’,4’,5’) and let modality ID of the dataset named `dataList.Chakra.Web` (or `dd`) be (1,2,3,4,5).

Then we can write the correspondence as follows;

\[ (1’,2’,3’,4’,5’) = (4,2,1,5,3). \]

### Contents:
*Multiple readers and Multiple modalities case, i.e., MRMC case*

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q )</td>
<td>( m )</td>
<td>( c )</td>
<td>( f )</td>
<td>( h )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>61</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>52</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>37</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>18</td>
<td>43</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>19</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>31</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>25</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>40</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>29</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>24</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>32</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>41</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>31</td>
<td>23</td>
</tr>
</tbody>
</table>
An FROC Dataset with Divergent Transitions in case of A Single reader and A Single modality

Description

A list, representing an FROC dataset with divergent transitions.

Note that the maximal number of confidence level, denoted by C, are included, however, confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) in the program and it does not refer from user input data, where C is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector c.

Format

A list consists of the following integer vectors f, h and integers NL, NI, C.

f  Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

h  Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

NL  A positive integer, representing Number of Lesions.

NI  A positive integer, representing Number of Images.

C  A positive integer, representing Number of Confidence level.

Contents:
A single reader and single modality case

<table>
<thead>
<tr>
<th>NI=57, NL=269</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
</table>
In R console ->

<table>
<thead>
<tr>
<th></th>
<th>c</th>
<th>f</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>3</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>probably present</td>
<td>2</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>questionable</td>
<td>1</td>
<td>36</td>
<td>3</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

Note that in FROC data, the confidence level means present (deceased, positive) case only. Since each reader marks their suspicious location only and it generate the hits and false alarms for his confidence level representing that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset.

Note that the first column of confidence level vector $c$ should not be specified. If specified, will be ignored, since it is created by $c <- c(rep(C:1))$ automatically in the program and it does not refer from user input data even if it is specified explicitly, where $C$ is the highest number of confidence levels. So you should check the compatibility of your data and the program’s generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Note that The format for the above example data must be made by the following forms:

```r
dat <- list(
c=c(3,2,1),  # Confidence level
  h=c(21,4,3),  # Number of hits for each confidence level
  f=c(0,7,36),  # Number of false alarms for each confidence level
  NL=60,       # Number of lesions
  NI=30,       # Number of images
  C=3)         # Number of confidence level
```

This R object `dat` can be passed to the function `fit_Bayesian_FROC()` as the following manner:

```r
fit_Bayesian_FROC(dat)
```

Examples

```r
## Not run:
#========================================================================================
# Change the zero cell to 1,
# then The number of divergent transitions are significantly decrease
# Thus, the divergent transitions is not rigid.
#========================================================================================
data <- dataList.divergent.transition.in.case.of.srsc
data$f <- c(1,7,36)
f <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = data )
```
### dataList.High

**Data: Single reader and Single modality**

#### Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

#### Details

This data-set is fictitious.

#### Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

#### References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

#### See Also

[dataList.Chakra.1.with.explanation](#)

---

### dataList.high.ability

**Data: A Single Reader and A Single Modality**

#### Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

#### Details

This data-set is fictitious.

#### Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>
dataList.low.ability

References
Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also
dataList.Chakra.1.with.explanation

dataList.Low  Data: Single reader and Single modality

Description
A list, representing FROC data to which we fit a FROC model. This data is same as dataList.Chakra.1.

Details
This data-set is fictitious.

Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also
dataList.Chakra.1.with.explanation

dataList.low.ability  Data: A Single Reader and A Single Modality

Description
A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details
This data-set is fictitious.
**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com >

**References**

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

**See Also**

dataList.Chakra.1.with.explantation

dataList.one.modality

---

**Description**

This is used to build a hierarchical FROC model.

**Details**

This data contains only one modality. If see = 12, then the model has converged.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com >

**References**

Nothing in 2018

---

**dataset_creator_by_specifying_only_M_Q**

*Creates dataset*

**Description**

creates dataset

**Usage**

dataset_creator_by_specifying_only_M_Q(M = 2, Q = 15)

**Arguments**

M A positive integer, indicating number of modalities.

Q A positive integer, indicating number of readers.
dataset_creator_by_specifying_only_M_Q

Value
An MRMC dataset.

Examples

```r
###1### ###2### ###3### ###4### ###5### ###6### ###7### ###8### ###9###
#========================================================================================
# make a data of a single modality and 36 readers
#========================================================================================

d<- dataset_creator_by_specifying_only_M_Q(M=1,Q=36)

check_hit_is_less_than_NL(d)

# plot_FPF_and_TPF_from_a_dataset(d)
plot_FPF_TPF_via_dataframe_with_split_factor(d)

###1### ###2### ###3### ###4### ###5### ###6### ###7### ###8### ###9###
#========================================================================================
# make a data of 2 modalities and 36 readers
#========================================================================================

d<- dataset_creator_by_specifying_only_M_Q(M=2,Q=36)

check_hit_is_less_than_NL(d)

# plot_FPF_and_TPF_from_a_dataset(d)
plot_FPF_TPF_via_dataframe_with_split_factor(d)

#========================================================================================
# make a data of 2 modalities and 6 readers
#========================================================================================

d<- dataset_creator_by_specifying_only_M_Q(M=2,Q=6)

check_hit_is_less_than_NL(d)

# plot_FPF_and_TPF_from_a_dataset(d)
```
dataset_creator_for_many_Readers

create data for MRMC

Description

create data for MRMC

Usage

dataset_creator_for_many_Readers(M, Q)

Arguments

- **M**: a positive integer, specifies the number of modalities
- **Q**: a positive integer, specifies the number of readers

Value

data, to which fit a model

Examples

d <- dataset_creator_for_many_Readers(1, 11)
dataset_creator_new_version

*Create a Dataset (version 2) Interactively*

**Description**

Create the Passing data to the function `fit_Bayesian_FROC`.

This is an interactive creator of dataset for FROC data.

**Usage**

```r
dataset_creator_new_version()
```

**Details**

This provide the interactive making of FROC dataset by using table to summarize hits and false alarm data.

Using this return value, you can build the FROC model for your data by applying the function `fit_Bayesian_FROC()` in this package.

Should carefully for the order of confidence levels.

**Value**

A list representing FROC data, to build FROC fitted model object by `fit_Bayesian_FROC()`.

**Examples**

```r
## Not run:
## Only run examples in interactive R sessions
if (interactive()) {

    dataset_creator_new_version()

}### Only run examples in interactive R sessions
## End(Not run)
```

Description
Example data-set which has small samples.

Details
the number of modalities, denoted by $M$. $M = 2$ modalities  
the number of Confidences, denoted by $C$. $C = 3$ Confidence levels  
the number of readers, denoted by $Q$. $Q = 2$ readers

Contents
$NL = 142$ (Number of Lesions)  
$NI = 57$ (Number of Images)

Contents:
Multiple readers and multiple modalities case, i.e., MRMC case

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Example data of Jafroc software
data_of_36_readers_and_a_single_modality

See Also


Examples

```r
#========================================================================================
# Show data by table
#========================================================================================

viewdata(data_of_36_readers_and_a_single_modality)

plot_FPF_and_TPF_from_a_dataset(data_of_36_readers_and_a_single_modality)

#========================================================================================
# make this data from functions in this package
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)

# The last object named d is the desired dataset.
```

Description

An example data-set whose sample size is large.

Details

Frequentist methods fails when a sample size is large. Namely, p value monotonically decreases when the sample size tends to large.

On the other hands, in Bayesian methods, the large samples such as large readers in FROC context fails the MCMC algorithm. Thus Bayesian methods is also not free from such large sample problem in this sense.

This dataset is made for validation that whether Bayes factor well work which is a subset of `data dataList.Chakra.Web.orderd`
the number of modalities, denoted by $M$ which is now 1 modality
the number of Confidences, denoted by $C$ which is now 5 Confidence levels
the number of readers, denoted by $Q$ which is now 36 readers

Contents of data_of_36_readers_and_a_single_modality

$NL = 142$ (Number of Lesions)
$NI = 57$ (Number of Images)

Contents:

Multiple readers and multiple modalities case, i.e., MRMC case

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>2</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>
|   | 7  | 1  | 5  | 17 | 8  | 5  | 1  | 11 | 8  | 4  | 5  | 19 | 8  | 3  | 10 | 16 | 8  | 2  | 16 | 17 | 8  | 1  | 12 | 15 | 9  | 5  | 0  | 15 | 9  | 4  | 1  | 26 | 9  | 3  | 3  | 20 | 9  | 2  | 6  | 18 | 9  | 1  | 4  | 12 | 10 | 5  | 0  | 31 | 10 | 4  | 4  | 40 | 10 | 3  | 8  | 22 | 10 | 2  | 13 | 16 | 10 | 1  | 9  | 5  | 11 | 5  | 0  | 13 | 11 | 4  | 2  | 23 | 11 | 3  | 5  | 19 | 11 | 2  | 9  | 19 | 11 | 1  | 6  | 17 | 12 | 5  | 0  | 8  | 12 | 4  | 3  | 16 | 12 | 3  | 7  | 15 | 12 | 2  | 11 | 17 | 12 | 1  | 8  | 22 | 13 | 5  | 0  | 13 | 13 | 4  | 1  | 23 | 13 | 3  | 4  | 19 | 13 | 2  | 7  | 21 | 13 | 1  | 4  | 20 | 14 | 5  | 0  | 36 | 14 | 4  | 4  | 45 | 14 | 3  | 9  | 22 | 14 | 2  | 14 | 13 | 14 | 1  | 10 | 3  | 15 | 5  | 0  | 17 | 15 | 4  | 2  | 27 | 15 | 3  | 5  | 20 | 15 | 2  | 9  | 18 | 15 | 1  | 6  | 10 | 16 | 5  | 0  | 8  | 16 | 4  | 4  | 15 | 16 | 3  | 8  | 13 | 16 | 2  | 13 | 16 | 16 | 1  | 9  | 22 | 17 | 5  | 0  | 9  | 17 | 4  | 1  | 16 | 118

*data_of_36_readers_and_a_single_modality*
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>2</td>
<td>8</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>1</td>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>5</td>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>4</td>
<td>2</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>3</td>
<td>6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>2</td>
<td>10</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>5</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>4</td>
<td>3</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>3</td>
<td>8</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>2</td>
<td>12</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>1</td>
<td>9</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>2</td>
<td>6</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>1</td>
<td>4</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>5</td>
<td>0</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>4</td>
<td>2</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>3</td>
<td>5</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>2</td>
<td>9</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>4</td>
<td>3</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>3</td>
<td>7</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>2</td>
<td>12</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>1</td>
<td>8</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>4</td>
<td>4</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>2</td>
<td>12</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>1</td>
<td>9</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>4</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>2</td>
<td>6</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>1</td>
<td>4</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>4</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>2</td>
<td>6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>1</td>
<td>4</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>5</td>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>4</td>
<td>1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>3</td>
<td>4</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>2</td>
<td>8</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>1</td>
<td>5</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>
data_of_36_readers_and_a_single_modality

<table>
<thead>
<tr>
<th>1</th>
<th>27</th>
<th>5</th>
<th>0</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>4</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>3</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>2</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>4</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>3</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>2</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>1</td>
<td>11</td>
<td>26</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>5</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>4</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>3</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>2</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>1</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>5</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>4</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>3</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>2</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>1</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>4</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>3</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>2</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>1</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>5</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>4</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>3</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>2</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>1</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>5</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>4</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>3</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>2</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>1</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>5</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>4</td>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>3</td>
<td>8</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>2</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>1</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>4</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>3</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>2</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>1</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>5</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>4</td>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>3</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>
data_of_36_readers_and_a_single_modality

<table>
<thead>
<tr>
<th>1</th>
<th>36</th>
<th>2</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36</td>
<td>1</td>
<td>12</td>
<td>9</td>
</tr>
</tbody>
</table>

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

References

Example data of Jafroc software

See Also


Examples

```r
#========================================================================================
# Show data by table
#========================================================================================

viewdata(data_of_36_readers_and_a_single_modality)

plot_FPF_and_TPF_from_a_dataset(data_of_36_readers_and_a_single_modality)

#========================================================================================
# make this data from functions in this package
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)

# The last object named d is the desired dataset.
```
Description

A list, representing FROC data of MRMC. This is same as `dataList.Chakra.Web`.

Details

This data is based on Chakraborty’s JAFROC software in which example data exists. The author have calculated hits and false alarms from this Jafroc example data.

Contents:

*Multiple readers and multiple modalities case, i.e., MRMC case*

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>m</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>19</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>31</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>41</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>31</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>37</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
dd

| 4 | 4 | 4 | 2 | 16 |
| 4 | 4 | 3 | 4 | 23 |
| 4 | 4 | 2 | 18 | 43 |
| 4 | 4 | 1 | 25 | 15 |
| 5 | 1 | 5 | 0 | 35 |
| 5 | 1 | 4 | 2 | 29 |
| 5 | 1 | 3 | 19 | 18 |
| 5 | 1 | 2 | 23 | 9 |
| 5 | 1 | 1 | 18 | 0 |
| 5 | 2 | 5 | 0 | 17 |
| 5 | 2 | 4 | 2 | 27 |
| 5 | 2 | 3 | 6 | 24 |
| 5 | 2 | 2 | 10 | 0 |
| 5 | 2 | 1 | 30 | 0 |
| 5 | 3 | 5 | 2 | 34 |
| 5 | 3 | 4 | 25 | 33 |
| 5 | 3 | 3 | 40 | 7 |
| 5 | 3 | 2 | 29 | 13 |
| 5 | 3 | 1 | 24 | 2 |
| 5 | 4 | 5 | 1 | 12 |
| 5 | 4 | 4 | 1 | 16 |
| 5 | 4 | 3 | 4 | 21 |
| 5 | 4 | 2 | 24 | 35 |
| 5 | 4 | 1 | 32 | 15 |

Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Example data of Jafroc software

See Also
dataList.Chakra.Web

Examples

viewdata(BayesianFROC::dd)

#========================================================================================
# dd is same as dataList.Chakra.Web, since the following code is all TRUE
#========================================================================================
#========================================================================================
# Code to make the dataset dd
#========================================================================================

h <- c(50,30,11,5,15,29,29,1,0,39,31,8,10,3,10,8,25,45,14, # modality 1
      52,25,13,4,1,27,28,29,1,0,53,29,13,2,4,9,16,22,43,14, # modality 2
      43,29,11,6,0,18,29,21,0,0,43,29,6,7,1,10,14,19,32,23, # modality 3
      61,19,12,9,3,16,29,34,1,0,52,29,10,4,3,10,16,23,43,15, # modality 4
      35,29,18,9,0,17,27,24,0,0,34,33,7,13,2,12,16,21,35,15) # modality 5

f <- c(0,4,20,21,0,0,6,15,22,1,15,18,31,19,1,2,4,16,17, # modality 1
      1,1,21,24,23,1,1,5,30,40,2,19,31,56,42,2,0,2,30,32, # modality 2
      1,7,13,28,19,0,1,7,7,31,7,15,28,41,9,0,2,5,24,31, # modality 3
      1,4,18,21,23,1,1,0,11,35,6,14,37,36,18,0,2,4,18,25, # modality 4
      0,2,19,23,18,0,2,6,10,30,2,25,40,29,24,1,1,4,24,32)# modality 5

a <- m_q_c_vector_from_M_Q_C(5,4,5)

m <- a$m
c <- a$c
q <- a$q

NI <- 199
NL <- 142
C <- 5
M <- 5
Q <- 4

dd <- list(h=h,
f=f,
m=m,
c=c,
q=q,
NI=NI,
NL=NL,
M=M,

dd$f=dataList.Chakra.Web$f
dd.orderd

Multiple Reader and Multiple Modality Data

Description

A list, representing FROC data of MRMC. This is same as dataList.Chakra.Web.

Details

This data is based on in Chakraborty’s JAFROC software in which example data exists. The author have calculated hits and false alarms from this Jafroc example data. Moreover the author ordered it such that the modality ID also means its observer performance, namely Modality ID = 1 means it has the most high AUC.

contents

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of hits</th>
<th>No. of false alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>h</td>
<td>f</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>52</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>37</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>43</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>53</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>29</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>42</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>43</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>29</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>45</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>27</td>
<td>2</td>
</tr>
</tbody>
</table>
4  2  3  24  6
4  2  2  0  10
4  2  1  0  30
4  3  5  34  2
4  3  4  33  25
4  3  3  7  40
4  3  2  13  29
4  3  1  2  24
4  4  5  12  1
4  4  4  16  1
4  4  3  21  4
4  4  2  35  24
4  4  1  15  32
5  1  5  43  1
5  1  4  29  7
5  1  3  11  13
5  1  2  6  28
5  1  1  0  19
5  2  5  18  0
5  2  4  29  1
5  2  3  21  7
5  2  2  0  7
5  2  1  0  31
5  3  5  43  7
5  3  4  29  15
5  3  3  6  28
5  3  2  7  41
5  3  1  1  9
5  4  5  10  0
5  4  4  14  2
5  4  3  19  5
5  4  2  32  24
5  4  1  23  31

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

References

Example data of Jafroc software

Examples

viewdata(BayesianFROC::dd.orderd)
# Code to make the dataset dd

```r
h <- c(61,19,12,9,3,16,29,34,1,0,52,29,10,4,3,10,16,23,43,15, # modality 4 of dataset dd
      52,25,13,4,1,27,28,29,1,0,53,29,13,2,4,9,16,22,43,14, # modality 2 of dataset dd
      50,30,11,5,1,15,29,29,1,0,39,31,8,10,3,10,8,25,45,14, # modality 1 of dataset dd
      35,29,18,9,0,17,27,24,0,0,34,33,7,13,2,12,16,21,35,15. # modality 5 of dataset dd
      43,29,11,6,0,18,29,21,0,0,43,29,6,7,1,10,14,19,32,23 # modality 3 of dataset dd
    )

f <- c(1,4,18,21,23,1,1,0,11,35,6,14,37,36,18,0,2,4,18,25,# modality 4 of dataset dd
       1,4,18,21,23,1,1,0,30,40,2,19,31,56,42,2,0,2,30,32,# modality 2 of dataset dd
       0,4,20,29,21,0,0,6,15,22,1,15,18,31,19,1,2,4,16,17,# modality 1 of dataset dd
       1,1,29,18,0,2,6,10,30,2,25,40,29,24,1,1,4,24,32,# modality 5 of dataset dd
       1,7,13,28,19,0,1,7,7,31,7,15,28,41,9,0,2,5,24,31# modality 3 of dataset dd
    )

a <- m_q_c_vector_from_M_Q_C(5,4,5)

m <- a$m
c <- a$c
q <- a$q

NI <- 199
NL <- 142
C <- 5
M <- 5
Q <- 4

dd.orderd <- list(
  h = h,
  f = f,
  m = m,
  c = c,
  q = q,
  NI = NI,
  NL = NL,
  M = M,
  Q = Q,
)
```

```
Description

This is a subset of dd.

This dataset has different dimensions for each modality and reader and confidence levels. To confirm my program is correct, the author made this.

In the following I emphasize that this data set has distinct \( C, M, Q \):

- \( \text{ddd}$C \) 5 Confidence levels
- \( \text{ddd}$M \) 3 modalities
- \( \text{ddd}$Q \) 4 readers

So, all number, i.e. \( M, C, Q \) is different each other and this is the reason why the author made this dataset.

Details

The WAIC is finite which surprises me, because a dataset dd has no finite WAIC. Why??

I forgot when I wrote this and what model was fitted to this data, so I am not sure the current model has finite WAIC.

Revised 2019 Nov. 21

Contents of dd

- NL = 142 (Number of Lesions)
- NI = 199 (Number of Images)

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>19</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>31</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>41</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>24</td>
<td>32</td>
</tr>
</tbody>
</table>
Author(s)
Issei Tsunoda <tsunoda.issei1111@gmail.com>

References
Nothing in 2018

Examples
# make an object ddd from an object dd

```r
ddd <- data.frame(m=dd$m,q=dd$q,c=dd$c,h=dd$h,f=dd$f)
```
```
dddd <- ddd[ddd$m <4,]  # Reduce the dataset ddd, i.e., dd
```
```
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
ddd <- list(
m=dddd$m,
q=dddd$q,
c=dddd$c,
h=dddd$h,
f=dddd$f,
NL=142,
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)
```
```
```r
# Reduce the dataset ddd, i.e., dd
```
```
```r
# make an object ddd from an object dd
```
```
```r
```
Description
This is a subset of ddd. For this dataset, the function fit_Bayesian_FROC() well works. So, even if the number of reader is one, my program is available. Even if not available, I think it does not cause my model but my programming.

\[ \text{dddd} \]

- **M**: 5 modalities
- **C**: 5 Confidence levels
- **Q**: 1 readers

Details

Contents of dddd

- NL = 142 (Number of Lesions)
- NI = 199 (Number of Images)

Contents:
Multiple readers and multiple modalities case, i.e., MRMC case

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>m</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>23</td>
<td>9</td>
</tr>
</tbody>
</table>
The reason why the author made this data dddd is it has only one reader. My program well works for more than two reader and more than two modality case. However, the only one modality or only one reader case is very special for programming perspective, and thus the author had to confirm whether my program well works in such cases. For this dataset, the function `fit_Bayesian_FROC()` well works. So, even if in a single reader case, my programm is available. Even if not available, I think it does not cause my model but my programming.

References

Example data of Jafroc software

See Also


Examples

```r
# Show data by table
viewdata(BayesianFROC::dd)
```

```r
# make an object dddd from an object dd

ddd <- data.frame(m=dd$m,q=dd$q,c=dd$c,h=dd$h,f=dd$f)

ddd <- dddd[ddd$q < 2,] # Reduce the dataset ddd, i.e., dd

ddd <- list(
m=ddd$m,
q=ddd$q,
c=ddd$c,
h=ddd$h,
f=ddd$f,
NL=142,
```
NI=199, # 2020 April 6
C=max(dddd$c),
M=max(dddd$m),
Q=max(dddd$q)
)

dddd <- ddd

#========================================================================================
# Fit model to the object ddd
#========================================================================================
# Unfortunately, R CMD check require running time to be less than 5 which is difficult
# for rstan::sampling(), thus, we cannot run the following from roxygen2 example.
# #
# # For Fitting, execute the following R code;
#
# #

ddd
ddd = Data of MRMC; Model does converge.

Description

This is a subset of dd. In the past, this model did not converge in the Model_MRMC.stan, thus I made a new stan file to get convergence estimates. The stan file named Model_Hiera_OneModalityMultipleReader_TargetFormulation.stan. Thus, even if the number of modality is 1, we can pool the AUCs over all readers by using this new model. The author believes this pooling is the most natural, primitive, simple way.

dddd$M 1 modality <-- ATTENTION!!
dddd$C 5 Confidence levels
dddd$Q 4 readers

Details

The model did not converge both null model and alternative model in 2019 Jun 21.

Contents of dddd

NL = 142 (Number of Lesions)
NI = 199 (Number of Images)

Multiple readers and multiple modalities case, i.e., MRMC case
### Examples

```r
# Show data by table
viewdata(BayesianFROC::ddddd)
```
### Description ###

This is a subset of `dd`

This dataset is made, as a toy data, which is a subset of data `dd`

- `dddddd$M` 2 modalities
- `dddddd$C` 3 Confidence levels
- `dddddd$Q` 2 readers
Details

The model did not converge both null model and alternative model in 2019 Jun 21.

Contents of ddddd

NL = 142 (Number of Lesions)
NI = 199 (Number of Images)

Contents:

Multiple readers and multiple modalities case, i.e., MRMC case

<table>
<thead>
<tr>
<th>ModalityID</th>
<th>ReaderID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>m</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

References

Example data of Jafroc software

See Also


Examples

```r
# Show data by table
```

#===============================================================
# Show data by table
#===============================================================

```r
data = read.csv("example.csv")
summary(data)
```
ddd <- data.frame(m=dd$m,q=dd$q,c=dd$c,h=dd$h,f=dd$f)

# The following code extract the first and the second modality from dd
ddd <- ddd[ddd$q < 3,]  # Reduce the dataset ddd, i.e., dd

# This dataset is made in 2019 July 6, for the aim of easy exhibition
# This dataset is very minimum, and it is easy to view

Multiple reader and 2 modalities data such that all modalities have same AUC.

Description

This is a subset of dataList.Chakra.Web.orderd
Details

The author made this dataset to validate the scheme of Bayes factor well works in our Bayesian FROC models.

This dataset is made for validation that whether Bayes factor well work which is a subset of data `dataList.Chakra.Web.orderd`.

- **M**: 2 modalities of almost same AUC
- **C**: 3 Confidence levels
- **Q**: 2 readers

If Bayes factor admit the null hypothesis that all modality are same, that is, 1-st and 2-nd modality of `dataList.Chakra.Web.orderd` are same, then, the Bayes factor well works.

Contents of `dddddd`

- **NL**: 142 (Number of Lesions)
- **NI**: 199 (Number of Images)

Contents:

- Multiple readers and multiple modalities case, i.e., MRMC case

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

References

Example data of Jafroc software

See Also


Examples

```r
# Show data by table
viewdata(ddddddd)
```

```r
# make an object `ddddd` from an object `dataList.Chakra.Web.orderd`
```
### Description

demonstration

### Usage

demo_Bayesian_FROC()

### Details

The author often forget the R script for execute the demos or bother to write the code to execute demo, thus I made this.
Value

none

Examples

## Not run:

demo_Bayesian_FROC()

# 2019.05.21 Revised.

## End(Not run)# dottest

demo_Bayesian_FROC_without_pause

demonstration without pause

Description

demonstration without pause

Usage

demo_Bayesian_FROC_without_pause()

Value

none

draw.CFP.CTP.from.dataList

Plot the pairs of CFPs and CTPs

Description

It plot the emipirical FROC curves (not depicted the line).

Usage

draw.CFP.CTP.from.dataList(
  dataList, 
  ModifiedPoisson = FALSE, 
  new.imaging.device = TRUE
)
Arguments

dataList  
A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data. For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
  h = c(41,22,14,8,1),# number of hits for each confidence level
  f = c(1,2,5,11,13),# number of false alarms for each confidence level

  NL = 124,# number of lesions (signals)
  NI = 63,# number of images (trials)
  C = 5) # number of confidence, .. the author thinks it can be calculated as the length of h or f ...? ha,why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).

To make this R object dataList representing FROC data, this package provides three functions:

  convertFromJafroc()  If data is a JAFROC xlsx formulation.
  dataset_creator_new_version()  Enter TP and FP data by table .
  create_dataset()  Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

  f  Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

  h  Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

  NL  A positive integer, representing Number of Lesions.

  NI  A positive integer, representing Number of Images.

  C  A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. "Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified,
draw.CFP.CTP.from.dataList

will be ignored, since it is created by \( c <- \text{c(rep}(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

**data Format:**
*A single reader and a single modality case*

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>( c[1] = 5 )</td>
<td>( f[1] = F_5 = 1 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>( c[5] = 1 )</td>
<td>( f[5] = F_1 = 13 )</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP  
*hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*diseased, lesion*) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are *present*, and marked positions generates the hits or false alarms, *thus* each confidence level represents that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c <- \text{c(rep}(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c <- \text{c(rep}(C:1)) \) via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an R list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

- \( C \) A positive integer, representing the **highest** number of confidence level, this is a scalar.
- \( M \) A positive integer vector, representing the number of **modalities**.
- \( Q \) A positive integer, representing the number of **readers**.
\( m \) A vector of positive integers, representing the \textbf{modality} ID vector.

\( q \) A vector of positive integers, representing the \textbf{reader} ID vector.

\( c \) A vector of positive integers, representing the \textbf{confidence level}. This vector must be made by \texttt{rep(rep(C:1),M*Q)}

\( h \) A vector of non-negative integers, representing the number of \textbf{hits}.

\( f \) A vector of non-negative integers, representing the number of \textbf{false alarms}.

\( \text{NL} \) A positive integer, representing the Total number of \textbf{lesions} for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from \( C \). To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function \texttt{viewdata()} shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

\textbf{Example data.}

\textit{Multiple readers and multiple modalities (i.e., MRMC)}

\begin{tabular}{llllll}
\hline
Modality ID & Reader ID & Confidence levels & No. of false alarms & No. of hits. \\
\hline
\textbf{m} & \textbf{q} & \textbf{c} & \textbf{f} & \textbf{h} \\
\hline
1 & 1 & 3 & 20 & 111 \\
1 & 1 & 2 & 29 & 55 \\
1 & 1 & 1 & 21 & 22 \\
1 & 2 & 3 & 6 & 100 \\
1 & 2 & 2 & 15 & 44 \\
1 & 2 & 1 & 22 & 11 \\
2 & 1 & 3 & 6 & 66 \\
2 & 1 & 2 & 24 & 55 \\
2 & 1 & 1 & 23 & 1 \\
2 & 2 & 3 & 5 & 66 \\
2 & 2 & 2 & 30 & 55 \\
2 & 2 & 1 & 40 & 44 \\
\hline
\end{tabular}

* \textit{false alarms} = False Positives = FP

* \textit{hits} = True Positives = TP

\texttt{ModifiedPoisson}

Logical, that is \texttt{TRUE} or \texttt{FALSE}.

If \texttt{ModifiedPoisson} = \texttt{TRUE}, then Poisson rate of false alarm is calculated \textit{per lesion}, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF \textit{per lesion} and FPF \textit{per lesion}. 
Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows \( (F_c \text{ denotes the number of false alarms with confidence level } c) \)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the False Positive Fraction (FPF) per lesion.
On the other hand,
if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]
where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the *False Positive Fraction (FPF) per image.*

The model is fitted so that the estimated FROC curve can be ragraded as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \)) or as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{TRUE} \)).

If \( \text{ModifiedPoisson} = \text{TRUE} \), then FROC curve means the expected pair of FPF *per lesion* and TPF.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \), then FROC curve means the expected pair of FPF *per image* and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

**Value**

CFPs and CTPs

**See Also**

`plot_FPF_and_TPF_from_a_dataset()`

`plot_FPF_TPF_via_dataframe_with_split_factor()`

**Examples**

```r
draw.CFP.CTP.from.dataList(dataList.Chakra.1)
```

---

**Description**

The function makes a plot of the FROC curve, the AFROC curve and *FPF* and *TPF.*
Usage

DrawCurves(
  StanS4class,
  modalityID,
  readerID,
  title = TRUE,
  indexCFPCTP = FALSE,
  upper_x,
  upper_y,
  new.imaging.device = TRUE,
  Colour = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  DrawAUC = TRUE,
  DrawCFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE,
  summary = TRUE,
  type = 4,
  color_is_changed_by_each_reader = FALSE,
  Draw.inner.circle.for.CFPCTPs = TRUE
)

Arguments

StanS4class   An S4 object of class \textit{stanfitExtended} which is an inherited class from the S4 class \textit{stanfit}. This R object is a fitted model object as a return value of the function \textit{fit-Bayesian-FROC}(). It can be passed to \textit{DrawCurves()}, \textit{ppp()} and \ldots etc

modalityID    A positive integer vector indicating modalityID. If it is not given, then the first modality is chosen.

readerID      A positive integer vector indicating readerID. If it is not given, then the first reader is chosen.

title         Logical: TRUE of FALSE. If TRUE (default), then title of curves are drawn.

indexCFPCTP   TRUE of FALSE. If TRUE, then the cumulative false and hits are specified with its confidence level.

upper_x       This is a upper bound for the axis of the horizontal coordinate of FROC curve.

upper_y       This is a upper bound for the axis of the vertical coordinate of FROC curve.

ew.imaging.device   Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

Colour        Logical: TRUE of FALSE. whether Colour of curves is dark theme or not.

DrawFROCcurve Logical: TRUE of FALSE. Whether or not FROC curves are shown.

DrawAFROCcurve Logical: TRUE of FALSE. Whether or not AFROC curves are shown.

DrawAUC       TRUE of FALSE. If TRUE then area under the AFROC curves are painted.
DrawCurves

DrawCFPCTP Logical: TRUE of FALSE. Whether or not the pairs of $FPF$ and $TPF$ are shown.

Draw.Flexible.upper_y Logical: TRUE of FALSE. Whether or not the upper bounds of vertical axis are determined automatically.

Draw.Flexible.lower_y Logical: TRUE of FALSE. Whether or not the lower bounds of vertical axis are determined automatically.

summary Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

type An integer, for the color of background and etc.

color_is_changed_by_each_reader A logical, if TRUE, then the FROC curves, AFROC curves, and FPF, TPF are colored accordingly by each reader. The aim of FROC analysis is to compare the modality and not reader, so the default value is false, and curves and FPF and TPF are colored by each modalities.

Draw.inner.circle.for.CFPCTPs TRUE or FALSE. If true, then to plot the cumulative false positives and true positives the plot points is depicted by two way, one is a large circle and one is a small circle. By see the small circle, user can see the more precise position of these points.

Details
The function makes a plot of the FROC curves and AFROC curves for user’s specified modality and user’s specified reader. Using this function repeatedly, we can draw the different reader and modality in a same plane simultaneously. So, we can visualize the difference of modality (reader).

Examples

#=============The first example========================================
# Not run:
#1) Fit a model to data by the following:

fit <- fit_Bayesian_FROC(
  BayesianFROC::dataList.Chakra.Web, # data to which fit the model
  ite=1111 # iteration of MCMC is too small
)

#Note that the return value "fit" is an object of an inherited S4 class from stanfit

#2) Using the above S4 class object, we draw the curves.
DrawCurves(
    fit,
    modality = 1,
    reader = 4)

# From this code, an FROC curve for the first modality and the fourth reader is drawn.

# 3) By changing, e.g., the modality,
   # we can draw the curves for different modalities.
   # This shows the comparison of modalities.
   # In the following,
   # the first script plots a curve for the 2nd modality and the fourth reader,
   # and the second script plots a curve for the 3rd modality and the 4th reader,
   # respectively.

DrawCurves(fit, modality = 2, reader = 4)
DrawCurves(fit, modality = 3, reader = 4)

# Curves are overwritten in a single imaging device for the comparison.

# 4) By applying the function with respect to different modalities
   # in this manner, we can draw AFROC (FROC) curves in the same plain.

# 5) If you want to draw the FROC curves
   # for reader ID = 1,2,3,4 and modality ID = 1,2, then the code is as follows;

    DrawCurves(
        fit,
        modalityID = c(1,2,3,4),
        readerID = c(1,2)
    )

    # Each color of curves corresponds to the modality ID.
    # So, the curves of "different" readers will have the "same" color,
    # if their modalities are "same".
# 6) To show only data points, i.e. FPF and TPF, use DrawFROCcurve = F as follows:

```r
DrawCurves(fit,
    DrawCFPCTP = TRUE,  # This implies data points are plotted.
    DrawFROCcurve = FALSE,  # From this, the curves are not drawn.
    modalityID = c(1,2,3,4),
    readerID = c(1)
)
```

# 7) If you use the plot in submission and it is not allowed to use color, then by Colour = FALSE, you can get black and white plots, e.g.,

```r
DrawCurves(fit,
    DrawCFPCTP = TRUE,
    DrawFROCcurve = TRUE,
    modalityID = c(1,2,3,4),
    readerID = c(1),
    Colour = FALSE  # From this, you can get plots without colors.
)
```

# 8) For AFROC, use DrawAFROCcurve = T

```r
DrawCurves(fit,
    DrawFROCcurve = FALSE,
    DrawAFROCcurve = TRUE,
    modalityID = c(1,2,3,4),
    readerID = c(1)
)
```

# 9)
# In order to compare modality, we draw curves by each modality
# The 1-st modality with all readers 1,2,3,4:

DrawCurves(fit, modalityID = 1, readerID = 1:4, new.imaging.device = TRUE)

# The 2-nd modality with all readers 1,2,3,4:
DrawCurves(fit, modalityID = 2, readerID = 1:4, new.imaging.device = FALSE)

# The 3-rd modality with all readers 1,2,3,4:
DrawCurves(fit, modalityID = 3, readerID = 1:4, new.imaging.device = FALSE)

# The 4-th modality with all readers 1,2,3,4:
DrawCurves(fit, modalityID = 4, readerID = 1:4, new.imaging.device = FALSE)

# The 5-th modality with all readers 1,2,3,4:
DrawCurves(fit, modalityID = 5, readerID = 1:4, new.imaging.device = FALSE)

# Draw for all pairs of modalities and readers:

DrawCurves(
    modalityID = 1:fit@dataList$M, 
    readerID = 1:fit@dataList$Q, 
    StanS4class = fit 
)

# Changes the color by

DrawCurves(fit, type = 2)
DrawCurves(fit, type = 3)
DrawCurves(fit, type = 4)
DrawCurves(fit, type = 5)
DrawCurves(fit, type = 6)
DrawCurves(fit, type = 7)
# Second Example

This function is available in the case of a single reader and a single modality. The reason why the maintainer separate the function for two processes, one is the fitting and the second is to plot curves is, in MRMC case, it tooks a time to drawing, but in the a single reader and a single modality case, drawing the curve is very fast, so in fitting process the curves are also depicted, however by this function user can draw the FROC curves.

First, we prepare the data endowed with this package.

```r
dat <- get(data("dataList.Chakra.1"))
```

Second, we fit a model to data named "dat"

```r
fit <- fit_srsc(dat)
```

Drawing the curves by

```r
DrawCurves(fit)
```

Changes the color by

```r
DrawCurves(fit, type = 2)
DrawCurves(fit, type = 3)
DrawCurves(fit, type = 4)
DrawCurves(fit, type = 5)
```
DrawCurves_MRMC

DrawCurves(fit, type = 6)
DrawCurves(fit, type = 7)

# Close the graphic device to avoid errors in R CMD check.
Close_all_graphic_devices()

## End(Not run)# dottest

DrawCurves_MRMC

Draw the FROC curves for all modalities and readers

Description

Draw the FROC curves and AFROC curves for all specified modalities and readers.

Usage

DrawCurves_MRMC(StanS4class, type = 1)

Arguments

StanS4class An S4 object of class StanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC().
It can be passed to DrawCurves(), ppp() and etc.
type An integer, for the color of the background and etc.

Examples

## Not run:
fit <- fit_Bayesian_FROC(
    dataList.Chakra.Web.orderd,
    ite = 1111,
    summary = FALSE
)

DrawCurves_MRMC(fit)

# 2019.05.21 Revised.

## End(Not run)# dottest
**DrawCurves_MRMC_pairwise**

*Draw the FROC curves with Colour*

**Description**

Draw **FROC curves** and **AFROC curves** for user's specified modalities and user's specified readers. Using this function **repeatedly**, we can draw the different reader and modality in a **same** plane simultaneously.

**Usage**

```r
DrawCurves_MRMC_pairwise(
  StanS4class,
  modalityID,
  readerID,
  Colour = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE,
  new.imaging.device = TRUE,
  summary = TRUE,
  color_is_changed_by_each_reader = FALSE,
  type = 1
)
```

**Arguments**

- **StanS4class**
  An S4 object of class **stanfitExtended** which is an inherited class from the S4 class **stanfit**. This R object is a fitted model object as a return value of the function **fit_Bayesian_FROC()**.
  It can be passed to **DrawCurves()**, **ppp()** and ... etc

- **modalityID**
  This is a vector indicating modalityID whose component is natural number.

- **readerID**
  This is a vector indicating readerID whose component is natural number.

- **Colour**
  Logical, that is TRUE or FALSE. Whether plot of curves are with dark theme. Default is TRUE indicating dark theme.

- **DrawFROCcurve**
  Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.

- **DrawAFROCcurve**
  Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.

- **DrawCFPCTP**
  Logical: TRUE of FALSE. Whether the CFP and CTP points are to be drawn. CFP: Cumulative false positive per lesion (or image) which is also called False Positive Fraction (FPF). CTP Cumulative True Positive per lesion which is also called True Positive Fraction (TPF).
DrawFlexible.upper_y
Logical, that is TRUE or FALSE. Whether or not the upper bounds of vertical axis are determined automatically.

DrawFlexible.lower_y
Logical, that is TRUE or FALSE. Whether or not the lower bounds of vertical axis are determined automatically.

new.imaging.device
Logical: TRUE or FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

summary
Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

color_is_changed_by_each_reader
A logical, if TRUE, then the FROC curves, AFROC curves, and FPF, TPF are colored accordingly by each reader. The aim of FROC analysis is to compare the modality and not reader, so the default value is false, and curves and FPF and TPF are colored by each modalities.

type
An integer, for the color of background and etc.

Details
By drawing different modality FROC curves in the same plane, we can compare the modality. E.g., if some modality FROC curve is upper then other modality curves, then we may say that the upper modality is better observer performance, i.e., higher AUC.

Author(s)
Issei Tsunoda

Examples

```r
## Not run:
#1) Fit a model to data by the following:

fit <- fit_Bayesian_FROC(dataList.Chakra.Web, ite = 1111)

#Note that the return value "fit" is an object of an inherited S4 class from stanfit

#2) Using the above S4 class object, we draw the curves.

DrawCurves_MRMC_pairwise(fit,
    modality = 1,
    reader = 4
)
```

#3) By changing the modality (or reader),
#we can draw the curves with respect to different modalities.
#This shows the comparison of modalities.

\[
\text{DrawCurves\_MRMC\_pairwise(fit,}
\begin{align*}
\text{modality} &= 2, \\
\text{reader} &= 4
\end{align*}
\]
\[
\text{DrawCurves\_MRMC\_pairwise(fit,}
\begin{align*}
\text{modality} &= 3, \\
\text{reader} &= 4
\end{align*}
\]

#4) By repeating in this manner for different modalities or readers,
#\ we can draw AFROC (FROC) curves in a single imaging device.
# Revised 2019 Nov 27

#5) If you want to draw the FROC curves
#for reader ID = 1, 2, 3, 4 and modality ID = 1, 2, then the code is as follows;

\[
\text{DrawCurves\_MRMC\_pairwise(}
\begin{align*}
\text{fit,}
\text{modalityID} &= c(1, 2, 3, 4), \\
\text{readerID} &= c(1, 2)
\end{align*}
\]

# Each color of curves corresponds to the modality ID.
# So, even if curves are different readers and same modality, then color is same.

# Close the graphic device
Close\_all\_graphic\_devices()
DrawCurves_MRMC_pairwise_BlackWhite

Draw the FROC curves without colour

Description
Plot curves without colors (dark theme), that is, black and white (white background with black curves). Draw FROC curves and AFROC curves for user's specified modality and user's specified reader. Using this function repeatedly, we can draw curves simultaneously, and we compare observer performance of the different reader and modality intuitively. So, we can visualize the difference of modality (reader).

Usage

DrawCurves_MRMC_pairwise_BlackWhite(
  StanS4class,
  modalityID,
  readerID,
  new.imaging.device = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  Draw.CFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE,
  summary = TRUE,
  type = 1
)

Arguments

StanS4class An S4 object of class StanS4class which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and etc

modalityID This is a vector indicating modalityID whose component is natural number.

readerID This is a vector indicating readerID whose component is natural number.

new.imaging.device Logical: TRUE or FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

DrawFROCcurve Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.

DrawAFROCcurve Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.
DrawCurves_MRMC_pairwise_col

Draw the FROC curves with Colour

Description

Draw an FROC curves and an AFROC curves for user’s specified modality and user’s specified reader. Using this function repeatedly, we can draw the different reader and modality in a same plane simultaneously. So, we can visualize the difference of modality (reader).

Usage

```r
DrawCurves_MRMC_pairwise_col(
  StanS4class,
  modalityID,
  readerID,
  type = 1,
  color_is_changed_by_each_reader = FALSE,
  new.imaging.device = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE,
  summary = TRUE
)
```
Arguments

StanS4class  
An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and etc.

modalityID  
This is a vector indicating modalityID whose component is natural number.

readerID  
This is a vector indicating readerID whose component is natural number.

type  
An integer, for the color of background and etc.

color_is_changed_by_each_reader  
A logical, if `TRUE`, then the FROC curves, AFROC curves, and FPF, TPF are colored accordingly by each reader. The aim of FROC analysis is to compare the modality and not reader, so the default value is false, and curves and FPF and TPF are colored by each modalities.

new.imaging.device  
Logical: `TRUE` of `FALSE`. If `TRUE` (default), then open a new device to draw curve. Using this we can draw curves in same plain by `new.imaging.device=FALSE`.

DrawFROCcurve  
Logical: `TRUE` of `FALSE`. Whether the FROC curve is to be drawn.

DrawAFROCcurve  
Logical: `TRUE` of `FALSE`. Whether the AFROC curve is to be drawn.

DrawCFPCTP  
Logical: `TRUE` of `FALSE`. Whether the CFP and CTP points are to be drawn. CFP: Cumulative false positive per lesion (or image) which is also called False Positive Fraction (FPF). CTP Cumulative True Positive per lesion which is also called True Positive Fraction (TPF).

Draw.Flexible.upper_y  
Logical, that is `TRUE` or `FALSE`. Whether or not the upper bounds of vertical axis are determined automatically.

Draw.Flexible.lower_y  
Logical, that is `TRUE` or `FALSE`. Whether or not the lower bounds of vertical axis are determined automatically.

summary  
Logical: `TRUE` of `FALSE`. Whether to print the verbose summary. If `TRUE` then verbose summary is printed in the R console. If `FALSE`, the output is minimal. I regret, this variable name should be verbose.

---

**DrawCurves_srsc**  
*Draw the FROC curves*

Description

Draw an FROC curves and an AFROC curves.
Usage

```r
DrawCurves_srsc(
  StanS4class,
  type = 4,
  title = TRUE,
  indexCFPCTP = FALSE,
  upper_x,
  upper_y,
  new.imaging.device = TRUE,
  Drawcol = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE,
  Draw.inner.circle.for.CFPCTPs = TRUE,
  DrawAUC = TRUE
)
```

Arguments

- **StanS4class**: An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc.
- **type**: An integer, for the color of background and etc.
- **title**: Logical: TRUE or FALSE. If TRUE (default), then title of curves are drawn.
- **indexCFPCTP**: TRUE or FALSE. If TRUE, then the cumulative false and hits are specified with its confidence level.
- **upper_x**: This is a upper bound for the axis of the horizontal coordinate of FROC curve.
- **upper_y**: This is a upper bound for the axis of the vertical coordinate of FROC curve.
- **new.imaging.device**: Logical: TRUE or FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
- **Drawcol**: Logical: TRUE or FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The Default value is a TRUE.
- **DrawFROCcurve**: Logical: TRUE or FALSE. Whether or not FROC curves are shown.
- **DrawAFROCcurve**: Logical: TRUE or FALSE. Whether or not AFROC curves are shown.
- **DrawCFPCTP**: Logical: TRUE or FALSE. Whether or not the pairs of FPF and TPF are shown.
- **Draw.inner.circle.for.CFPCTPs**: TRUE or FALSE. If true, then to plot the cumulative false positives and true positives the plot points is depicted by two way, one is a large circle and one is a small circle. By see the small circle, user can see the more precise position of these points.
- **DrawAUC**: TRUE or FALSE. If TRUE then area under the AFROC curves are painted.
Draw_an_area_of_AUC_for_srsc

*Draw a Region of the area under the AFROC curve*

**Description**

Draw a Region of the area under the AFROC curve

**Usage**

`Draw_an_area_of_AUC_for_srsc(StanS4class)`

**Arguments**

- **StanS4class**: An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc

**Value**

None

**Examples**

```r
## Not run:

fit <- fit_Bayesian_FROC(dataList.Chakra.1)

Draw_an_area_of_AUC_for_srsc(fit)
```

```r
## End(Not run)# dottest
```

**Draw_AUC**

*Draw the Region of AUC of AFROC*
Description

An AFROC curve has two parameter denoted by $a, b$. By specifying $a, b$, we can draw an AFROC curve.

Def of AFROC

$$(\xi(t), \eta(t)) = (1 - e^{-t}, \Phi(b\Phi^{-1}(\exp(-t)) - a)).$$

Def of AUC of AFROC

$$AUC = \int \eta d\xi = \frac{a}{\sqrt{1 + b^2}}.$$  

Usage

```r
draw_auc(a = 0.13, b = 0.19, mesh.for.drawing.curve = 2222)
```

Arguments

- **a**: One of the parameter of model which characterize AFROC curve
- **b**: One of the parameter of model which characterize AFROC curve
- **mesh.for.drawing.curve**: A positive large integer, indicating number of dots drawing the curves, Default =10000.

Details

We define the so-called FROC curve as a map from 1-dimensional Euclidean space to 2-dimensional Euclidean space, mapping each $t > 0$ to

$$(x(t), y(t)) = (t, \Phi(\frac{\Phi^{-1}(\exp(-t)) - \mu}{\sigma}))$$

Sine $x(t) = t, t > 0$ is not bounded, the area under the FROC curve is infinity.

To calculates alternative notion of AUC in the ordinal ROC theory, we define the so-called AFROC curve:

$$(\xi(t), \eta(t)) = (1 - e^{-t}, \Phi(\frac{\Phi^{-1}(\exp(-t)) - \mu}{\sigma}))$$

which contained in the rectangular space $[0, 1]^2$. Introducing new parameter $a := \mu/\sigma$ and $b := 1/\sigma$, we also write

$$(\xi(t), \eta(t)) = (1 - e^{-t}, \Phi(b\Phi^{-1}(\exp(-t)) - a))$$

The area Under the (AFROC) curve (breifly, we call it AUC) represents the observer performance. For example, if radiologist detects more lesions with small False Positives (FPs), then AUC would be high.

Using the parameter of the signal distribution, we express AUC as follows,
\[ AUC = \frac{\mu/\sigma}{\sqrt{1 + 1/\sigma^2}}. \]

Using new parameter \( a := \mu/\sigma \) and \( b := 1/\sigma \), we also write

\[ AUC = \frac{a}{\sqrt{1 + b^2}}. \]

**Value**

none.

**Examples**

```
Draw_AUC()
```

---

**Description**

Draw One Sample from Prior

**Usage**

```
Draw_a_prior_sample(sd = 5, C = 5, seed.for.drawing.a.prior.sample = 1111)
```

**Arguments**

- **sd**  
  Standard deviation of priors. Very large number.
- **C**  
  No. of Confidence level
- **seed.for.drawing.a.prior.sample**
- **seed**

**Value**

w, v, m, dz, z
Draw a simulated dataset from model distributions with specified parameters from priors

**Usage**

```r
Draw_a_simulated_data_set(
    sd = 5,
    C = 5,
    seed.for.drawing.a.prior.sample = 1111,
    fun = stats::var,
    NI = 259,
    NL = 259,
    initial.seed.for.drawing.a.data = 1234,
    ModifiedPoisson = FALSE,
    ite = 1111
)
```

**Arguments**

- **sd**  Standard Deviation of priors
- **C**   No. of Confidence levels
- **seed.for.drawing.a.prior.sample**
- **seed**
- **fun**  An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.
- **NI**  No. of images
Draw a simulated data set

No. of Lesions

initial.seed.for.drawing.a.data

seed

ModifiedPoisson

Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.

Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_4 + F_5}{N_L},$$

$$\frac{F_5}{N_L},$$

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_I},$$

where $N_I$ is a number of images (signal).

For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]
\[
\frac{F_5}{N_I},
\]

where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of \textbf{FPF per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

tie

To be passed to the function \texttt{rstan::sampling} in \texttt{rstan}. An argument of \texttt{rstan::sampling} in which it is named \texttt{iter}. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

### Value

A single synthesized data-set

### Examples

```r
## Not run:
one.dataList <- Draw_a_simulated_data_set()

## End(Not run)# dotest
```
**Description**

Draw a dataset and MCMC samples.

1. draw a model parameter from prior distribution,
2. draw a dataset from the model with the parameter drawn in step 1,
3. draw a collection of posterior samples for the dataset drawn in step 2.

**Usage**

```r
Draw_a_simulated_data_set_and_Draw_posterior_samples(
  sd = 5,
  C = 5,
  seed.for.drawing.a.prior.sample = 1111,
  fun = stats::var,
  NI = 259,
  NL = 259,
  initial.seed.for.drawing.a.data = 1234,
  ModifiedPoisson = FALSE,
  PreciseLogLikelihood = TRUE,
  ite = 1111,
  DrawCurve = FALSE
)
```

**Arguments**

- `sd` : Standard Deviation of priors
- `C` : No. of Confidence levels
- `seed.for.drawing.a.prior.sample` : seed
- `fun` : An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.
- `NI` : No. of images
- `NL` : No. of Lesions
- `initial.seed.for.drawing.a.data` : seed
- `seed` :
ModifiedPoisson

Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per lesion*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF *per lesion*.

Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per image*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF *per image*.

For more details, see the author’s paper in which I explained *per image* and *per lesion*. (for details of models, see vignettes , now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows (F_c denotes the number of false alarms with confidence level c )

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the False Positive Fraction (FPF) *per lesion*.

On the other hand,
if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[ \frac{F_4 + F_5}{N_I}, \]
\[ \frac{F_5}{N_I}, \]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE})
or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE})
If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.
On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of \textbf{FPF per image} and TPF.
So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.
Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

\texttt{PreciseLogLikelihood}

Logical, that is \texttt{TRUE} or \texttt{FALSE}. If \texttt{PreciseLogLikelihood = TRUE}(default), then Stan calculates the precise log likelihood with target formulation. If \texttt{PreciseLogLikelihood = FALSE}, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

\texttt{iter}

To be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. An argument of \texttt{rstan::sampling()} in which it is named \texttt{iter}. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

\texttt{DrawCurve}

Logical: \texttt{TRUE} of \texttt{FALSE}. Whether the curve is to be drawn. \texttt{TRUE} or \texttt{FALSE}. If you want to draw the FROC and AFROC curves, then you set \texttt{DrawCurve =TRUE}, if not then \texttt{DrawCurve =FALSE}. The reason why the author make this variable \texttt{DrawCurve} is that it takes long time in MRMC case to draw curves, and thus Default value is \texttt{FALSE} in the case of MRMC data.
Value

Draw.a.prior.sample The Return value of Draw_a_prior_sample
A dataList and an object of the stanfit S4 class with respect to the dataList

See Also

hits_false_alarms_creator_from_thresholds

Examples

```r
## Not run:

# Draw a curve for various seeds and various number of confidence levels.
# Changing the seed, we can draw a parameter from priors and using this sample,
# we can draw the datasets from our model whose parameters are
# the priors samples.

# 1. draw a model parameter from prior distribution,
# 2. draw a dataset from the model with the parameter drawn in step 1,
# 3. draw a collection of posterior samples for the dataset drawn in step 2.

Draw_a_simulated_data_set_and_Draw_posterior_samples(
  seed.for.drawing.a.prior.sample = 1234,
  C=8)

Draw_a_simulated_data_set_and_Draw_posterior_samples(
  seed.for.drawing.a.prior.sample = 12345,
  C=7)

Draw_a_simulated_data_set_and_Draw_posterior_samples(
  seed.for.drawing.a.prior.sample = 123456,
  C=6)

Draw_a_simulated_data_set_and_Draw_posterior_samples(
  seed.for.drawing.a.prior.sample = 1234567,
  C=5)

## End(Not run)# dottest
```

draw_latent_noise_distribution

Visualization of the Latent Gaussian for false rates
draw_latent_noise_distribution

Description

Plot the posterior mean of model parameter $\theta$ and and the latent function, i.e. the differential logarithmic Gaussian $d \log \Phi(z)$.

Usage

draw_latent_noise_distribution(
  StanS4class,
  dark_theme = TRUE,  # TRUE or FALSE
  dig = 3,            # To be passed to function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing Significant digits, used in stan Cancellation. Default = 5,
  mesh = 1000,        # Mesh for painting the area
  new.imaging.device = TRUE,  # Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
  hit.rate = FALSE,   # whether draws it. Default is TRUE.
  false.alarm.rate = TRUE,  # whether draws it. Default is TRUE.
  both.hit.and.false.rate = TRUE,  # whether draws it. Default is TRUE.
  density = 22,       # A natural number, indicating the density of shading lines, in lines per inch.
  color = TRUE,       # A color region is selected from black and white only. For more colors, put FALSE. For publication, the mono color is allowed in many case, so the author made this for such publication.
  mathmatical.symbols = TRUE,  # A logical, whether legend is in plot.
  type = 3             # An integer, for the color of background and etc.
)

Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc

dark_theme  # TRUE or FALSE

dig  # To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing Significant digits, used in stan Cancellation. Default = 5,

mesh  # Mesh for painting the area

new.imaging.device  # Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

hit.rate  # whether draws it. Default is TRUE.

false.alarm.rate  # whether draws it. Default is TRUE.

both.hit.and.false.rate  # whether draws it. Default is TRUE.

density  # A natural number, indicating the density of shading lines, in lines per inch.

color  # A color region is selected from black and white only. For more colors, put FALSE. For publication, the mono color is allowed in many case, so the author made this for such publication.

mathmatical.symbols  # A logical, whether legend is in plot.

type  # An integer, for the color of background and etc.
Details

Our FROC model uses a latent Gaussian random variable to determine false rates which are defined as follows:

\[
q_5(z_1, \ldots, z_C) = \int_{z_5}^{\infty} d \log \Phi(z) dz
\]

\[
q_4(z_1, \ldots, z_C) = \int_{z_4}^{z_5} d \log \Phi(z) dz
\]

\[
q_3(z_1, \ldots, z_C) = \int_{z_3}^{z_4} d \log \Phi(z) dz
\]

\[
q_2(z_1, \ldots, z_C) = \int_{z_2}^{z_3} d \log \Phi(z) dz
\]

\[
q_1(z_1, \ldots, z_C) = \int_{z_1}^{z_2} d \log \Phi(z) dz
\]

For example, in the following data, the number of false alarm data with confidence level 5 is 41, which is considered as a sample from the Poisson distribution of its rate.

\[
q_5(z_1, \ldots, z_C) = \int_{z_5}^{\infty} d \log \Phi(z) dz
\]

So, this Gaussian distribution determines false rate, and this function \texttt{draw_latent_noise_distribution()} plot this Gaussian distribution \(d \log \Phi\) and the density \(\text{Gaussian}(z|\mu, \sigma)\) is also plotted to compare hit rates and false rates. Thus, the author implements it in the \texttt{draw_latent_signal_distribution()}.  

Example data:

A single reader and single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>probably present</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>equivocal</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

* false alarms = False Positives = FP

* hits = True Positives = TP
Value

Information of Latent Gaussians, such as mean and S.D. of the signal distributions and thresholds.

See Also

draw_latent_signal_distribution()

Examples

```r
## Not run:
#========================================================================================
# Shape of signal distribution strongly influences the value of AUC, so in the following
# the author shows how it affects the estimates of AUCs.
# We consider two data examples, one is a low AUC and the other is a high AUC.
# In the high AUC case, the Signal Gaussian will be low variance and
# in the low AUC case, the variance will desperse. 2019 August 4, 2019 Dec 17
#========================================================================================
#
# ----- High AUC case --------
viewdata(dataList.High)
fit.High <- fit_Bayesian_FROC(dataList.High,ite=111)
draw_latent_signal_distribution(fit.High)

#
# ----- Low AUC case --------
viewdata(dataList.Low)
fit.Low <- fit_Bayesian_FROC(dataList.Low)
draw_latent_signal_distribution(fit.Low)

## End(Not run)# dottest
```

draw_latent_signal_distribution

Visualization of Latent Gaussians (Signal Distribution)

Description

Plot the posterior mean of model parameter \( \theta \) and the parameter of the latent function, i.e. the normal distribution denoted by \( \text{Gaussian}(z|\mu, \sigma) \) with posterior mean estimates of its mean \( \mu \) and standard deviation \( \sigma \).
Usage

draw_latent_signal_distribution(
  StanS4class,
  dark_theme = TRUE,
  dig = 3,
  mesh = 1000,
  new.imaging.device = TRUE,
  hit.rate = TRUE,
  false.alarm.rate = FALSE,
  both.hit.and.false.rate = FALSE,
  density = 22,
  color = TRUE,
  mathmatical.symbols = TRUE,
  type = 3
)

Arguments

StanS4class
  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and etc.
dark_theme
  TRUE or FALSE
dig
  An positive integer, indicating the digit for numbers in the R console.
mesh
  Mesh for painting the area
new.imaging.device
  Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
hit.rate
  whether draws it. Default is TRUE.
false.alarm.rate
  whether draws it. Default is TRUE.
both.hit.and.false.rate
  whether draws it. Default is TRUE.
density
  A natural number, indicating the density of shading lines, in lines per inch.
color
  A color region is selected from black and white only. For more colors, put FALSE. For publication, the mono color is allowed in many case, so the author made this for such publication.
mathmatical.symbols
  A logical, whether legend is in plot.
type
  An integer, for the color of background and etc.

Details

Our FROC model use a latent Gaussian random variable to determine hit rates. That is, each hit rate is defined as follows:
For example, in the following data, the number of hit data with the most highest confidence level 5 is regarded as an sample from the Binomial distribution of hit rate
\[ p(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_5}^{\infty} \text{Gaussian}(z|\mu, \sigma) \, dz \]
with Bernoulli trial number is NL=142.

So, this Gaussian distribution determines hit rate, and this function `draw_latent_signal_distribution()` plot this Gaussian distribution \( \text{Gaussian}(z|\mu, \sigma) \). And a reference distribution is the standard Gaussian and do not confuse that it is not the noise distribution, but only reference.

The noise distribution (denoted by \( d \log \Phi() \)) determines the False alarm rates in the similar manner and plotted by using a line of dots. The author thinks the standard Gaussian is more comfortable to compare or confirm the shape of \( \text{Gaussian}(z|\mu, \sigma) \) and thus, the author implement it in the `draw_latent_signal_distribution()`.

One would want to see the signal distribution and noise distribution simultaneously, then use the function `draw_latent_noise_distribution()`.

**Value**

Information of Latent Gaussians, such as mean and S.D. of the signal distributions and thresholds.

**See Also**

`draw_latent_noise_distribution()` Note that the difference of `draw_latent_noise_distribution()` and `draw_latent_signal_distribution()` is that the lator use the standard Gaussian for the reference distribution and former uses the \( d \log \Phi() \) for the reference distribution.

So, the old version `draw_latent_signal_distribution()` is also important and I like this old version also. Anyway who read this, I think my package size is very large.....ha,,,I have to reduce it.....but how?

**Examples**

## Not run:
```
#========================================================================================
# Shape of signal distribution strongly influences the value of AUC, so in the following
# the author shows how it affects the estimates of AUCs.
# We consider two data examples, one is a low AUC and the other is a high AUC.
```
# In the high AUC case, the Signal Gaussian will be low variance and 
# in the low AUC case, the variance will despere. 2019 August 4, 2019 Dec 17 
#========================================================================================

# ----- High AUC case --------
viewdata(dataList.High)
fit.High <- fit_Bayesian_FROC(dataList.High,ite=111)
draw_latent_signal_distribution(fit.High)

# ----- Low AUC case --------
viewdata(dataList.Low)
fit.Low <- fit_Bayesian_FROC(dataList.Low)
draw_latent_signal_distribution(fit.Low)

#--------------------------------------------------------------------------------------
# 2) For submission (without color) 
#--------------------------------------------------------------------------------------

fit <- fit_Bayesian_FROC(
        dataList = dataList.Chakra.1.with.explantation
)

# With legends

draw_latent_signal_distribution(fit,
        dark_theme = FALSE,
        color = TRUE,
        density = 11
)
Empirical_FROC_via_ggplot

```r
without legends
draw_latent_signal_distribution(fit,
dark_theme = FALSE,
color = TRUE,
mathmatical.symbols = FALSE)
```

# 2019 Sept. 5
# 2020 March 12

## End(Not run)## dotest

dz

*Threshold: parameter of an MRMC model*

### Description

A posterior mean of the model parameter for data ddd as an example of truth parameter.

### Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

### See Also

- make_true_parameter_MRMC

---

**Empirical_FROC_via_ggplot**

*Empirical FROC curve via ggplot*

### Description

Empirical FROC curve via ggplot

### Usage

Empirical_FROC_via_ggplot(dataList)
Arguments

dataList A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TP, FP, lesions, images. In addition, if in the case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. This is a variable in the function \texttt{rstan::sampling()} in which it is named \texttt{data}.

For the single reader and a single modality data, the dataList is made by the following manner:

```r
dataList.Example <- list(
  h = c(41, 22, 14, 8, 1),  # number of hits for each confidence level
  f = c(1, 2, 5, 11, 13),  # number of false alarms for each confidence level
  NL = 124,  # number of lesions (signals)
  NI = 63,  # number of images (trials)
  C = 5)  # number of confidence, the author thinks it can be calculated as the length of \texttt{h} or \texttt{f}...? ha, why I included this. ha should be omitted.
```

And using this object \texttt{dataList.Example}, we can apply \texttt{fit_Bayesian_FROC()} such as \texttt{fit_Bayesian_FROC(dataList.Example)}.

To make this \texttt{R} object \texttt{dataList} representing FROC data, this package provides three functions:

- \texttt{convertFromJafroc()} If data is \texttt{JAFROC xlsx} formulation.
- \texttt{dataset_creator_new_version()} Enter TP and FP data by table.
- \texttt{create_dataset()} Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function \texttt{viewdata()}.  

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), \texttt{dataList} is a list consisting of \texttt{f, h, NL, NI, C} where \texttt{f, h} are numeric vectors and \texttt{NL, NI, C} are positive integers.

\texttt{f} Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

\texttt{h} Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

\texttt{NL} A positive integer, representing Number of Lesions.

\texttt{NI} A positive integer, representing Number of Images.

\texttt{C} A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by \texttt{C}, are included, however, Note that confidence level vector \texttt{c} should not be specified. If specified,
will be ignored, since it is created by \( c <- c(rep(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

**data Format:**

*A single reader and a single modality case*

<table>
<thead>
<tr>
<th></th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>( c[1] = 5 )</td>
<td>( f[1] = F_3 = 1 )</td>
<td>( h[1] = H_3 = 41 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>( c[5] = 1 )</td>
<td>( f[5] = F_1 = 13 )</td>
<td>( h[5] = H_1 = 1 )</td>
</tr>
</tbody>
</table>

* **false alarms** = False Positives = FP
* **hits** = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c <- c(rep(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c <- c(rep(C:1)) \) via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an \( \mathbb{R} \) list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

\( C \) A positive integer, representing the highest number of confidence level, this is a scalar.

\( M \) A positive integer vector, representing the number of modalities.

\( Q \) A positive integer, representing the number of readers.
m  A vector of positive integers, representing the modality ID vector.
q  A vector of positive integers, representing the reader ID vector.
c  A vector of positive integers, representing the confidence level. This vector must be made by `rep(rep(C:1),M*Q)`
h  A vector of non-negative integers, representing the number of hits.
f  A vector of non-negative integers, representing the number of false alarms.
NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector,

the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

**Multiple readers and multiple modalities (i.e., MRMC)**

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP
*hits = True Positives = TP

**Value**

none
Examples

```r
Empirical_FROC_via_ggplot(
  dataList = d
)
```

Close_all_graphic_devices()

<table>
<thead>
<tr>
<th>error_message</th>
<th>Error Message for Data Format</th>
</tr>
</thead>
</table>

Description

Plot error messages to let user know his or her data format is wrong.

Usage

```r
error_message(h, NL)
```

Arguments

- `h`: A non-negative integer vector
- `NL`: A positive integer, indicating Number of lesions

Details

If \( \text{sum}(h) > \text{NL} \), then an error message will appear. The reason why the author uses the generic function `plot()` for error messages instead of such as `message()` or `cat()` is to preserve GUIs in `Shiny`. So, this error message is shown in some plot plane in the Graphical User Interface of `Shiny` in which `message()` or `cat()` cannot use.

Value

Plot of an error message by the generic function `plot()` for Shiny GUI.

See Also

`fit_GUI()`
Examples

# If number of hits > number of lesion, then an error message appears.
# Make an example such that sum(h) > NL, that is, the sum of the number of hits is
greater than the number of lesion, then, it launches an error message.

```r
h <- c(50,30,20)
NL <- 3
error_message(h,NL)
```

# Then, in an imaging device, an error message appears, because sum(h) = 100 > 3 = NL.
# In Shiny, even if plot cannot be done causing some error, Graphical User Interface
can not change (now,... I can but.), so I have to use the graphical user interface.
# Thus. in such case, I use this function rather than the message() or cat().

# Who read this? My heart will be more empty when I wrote this manuual.

# This function is made in 2019 July, 6.
# Doc is reviesed in 2020 Feb

---

**error_message_on_imaging_device_rhat_values**

*Error message on a plot plane (imaging device)*

**Description**

Since, shiny board fix user interface, and it let me make this; in graphical device, the error message
should be shown on its device. So, usual functions such as message() or cat() cannot use in Shiny
board. Since, the UI is already made and it is graphical device!

If a fitted model converges, then the error message is none and thus only in R console, the message
is printed such as "A model converged." and does not print error message on a plot plane.

**Usage**

```r
error_message_on_imaging_device_rhat_values(
    StanS4class,
    verbose = TRUE,
    digits = 3
)
```
Arguments

StanS4class  An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc

verbose  A logical. if TRUE, then the maximal R hat is printed in the R console.

digits  digits to round r hat

Details

This is for non-convergent fitted model object, where convergence criteiron is R hat statistics for each model parameters.

Examples

```
# Non convergent fitting and error on it via a graphic device
## Not run:
# Creat a fitted model object which does not converge with R hat criterion:
fit <- fit_Bayesian_FROC( ite = 111,
                        cha = 1,
                        summary = TRUE,
                        Null.Hypothesis = FALSE,
                        dataList = dd  # Here, non convergent data )

# Nothing is plotted:
plot(0,0,
     type ="n",
     axes =FALSE,
     ann=FALSE
    )

# Error message on the above graphic device:
error_message_on_imaging_device_rhat_values(fit)

# Plot
DrawCurves(fit)
```
# It does not work, and it is ,,, Ok since when non converges I will want to see
# plot, so this function is no need.

# 2019 August 18

## End(Not run)#dontrun

---

**error_MRMC**  
*Comparison of Estimates and Truth in case of MRMC*

**Description**

In order to describe what this function calculates explicitly, let us denote a specified true model parameter by $\theta_0$, from which fake datasets are replicated and denoted by:

$$D_1, D_2, ..., D_k, ..., D_K.$$  

We obtain estimates

$$\theta(D_1), ..., \theta(D_K)$$

for each replicated dataset. Using these estimates, we calculate the mean of the absolute errors (an absolute difference between estimates and a true parameter $\theta_0$), namely,

$$\frac{1}{K} \sum_{k=1}^{K} |\theta(D_k) - \theta_0|,$$

or the variance of estimates:

$$\frac{1}{K} \sum_{k=1}^{K} (\theta(D_k) - \frac{1}{K} \sum_{k=1}^{K} \theta(D_k))^2.$$

Revised 2019 Nov 1  
Revised 2020 Jan  
Revised 2020 March

**Usage**

```r
error_MRMC(
  replication.number = 2,
  initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth,
```


NI = 200,
NL = 1142,
ModifiedPoisson = FALSE,
summary = FALSE,
ite = 1111
)

Arguments

replication.number
For fixed number of lesions, images, the dataset of hits and false alarms are
replicated, and the number of replicated datasets are specified by this variable.

initial.seed
The variable initial.seed is used to replicate datasets. That is, if you take
initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, etc are for
the first replication, the second replication, the third replication, etc. If the n-th
model does not converge for some n, then such model has no mean and thus the
non-convergent models are omitted to calculate the errors.

mu.truth
array of dimension (M,Q). Mean of the signal distribution of bi-normal assump-
tion.

v.truth
array of dimension (M,Q). Standard Deviation of represents the signal distribu-
tion of bi-normal assumption.

z.truth
This is a parameter of the latent Gaussian assumption for the noise distribution.

NI
Number of Images.

NL
Number of Lesions.

ModifiedPoisson
Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per
lesion, and model is fitted so that the FROC curve is an expected curve of points
consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per
image, and model is fitted so that the FROC curve is an expected curve of points
consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and
per lesion. (for details of models, see vignettes , now, it is omitted from this
package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined
as follows (F_c denotes the number of false alarms with confidence level c )
\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_3 + F_4 + F_5}{N_L},
\]
where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if \texttt{ModifiedPoisson = FALSE} (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is a redundant parameter for the FROC trial and thus the author try to exclude it.
Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

2019 Sept 6 I found this program, I made this in several month ago? I forgot when this function is made. It well works, so it helps me now.

Print for a given true parameter, a errors of estimates from replicated dataset. Also print a standard error which is the variance of estimates.

Suppose that \( \theta_0 \) is a given true model parameter with a given number of images \( N_I \) and a given number of lesions \( N_L \), specified by user.

(I)

(I.1) Synthesize a collection of dataset \( D_k \) \( (k = 1, 2, ..., K) \) from a likelihood (model) at a given parameter \( \theta_0 \), namely \( D_k \sim \text{likelihood}(\theta_0) \).

(I.2) Replicates \( K \) models fitted to each dataset \( D_k \) \( (k = 1, 2, ..., K) \), namely, draw MCMC samples \( \{\theta_i(D_k); i = 1, ..., I\} \) from each posterior of the dataset \( D_k \), namely \( \theta_i(D_k) \sim \pi(\cdot|D_k) \).

(I.3) Calculate posterior means for the set of data \( D_k \) \( (k = 1, 2, ..., K) \), namely \( \bar{\theta}(D_k) := \frac{1}{I} \sum_i \theta_i(D_k) \).

(I.4) Calculates error for each dataset \( D_k \) \( \epsilon_k := \text{Truth - estimates} = \theta_0 - \bar{\theta}(D_k) \).

(II) Calculates mean of errors over all datasets \( D_k \) \( (k = 1, 2, ..., K) \) mean of errors \( \bar{\epsilon}(\theta_0, N_I, N_L) = \frac{1}{K} \sum \epsilon_k \).

NOTE We note that if a fitted model does not converge,( namely R hat is far from one), then it is omitted from this calculation.

(III) Calculates mean of errors for various number of lesions and images mean of errors \( \bar{\epsilon}(\theta_0, N_I, N_L) \)
For example, if $(N_1^1, N_L^1), (N_2^2, N_L^2), (N_3^3, N_L^3), \ldots, (N_m^m, N_L^m)$, then $\bar{\epsilon}((\theta_0, N_1^1, N_L^1), \bar{\epsilon}((\theta_0, N_2^2, N_L^2), \bar{\epsilon}((\theta_0, N_3^3, N_L^3), \ldots, \bar{\epsilon}((\theta_0, N_m^m, N_L^m)$ are calculated.

To obtain precise error, The number of replicated fitted models (denoted by $K$) should be large enough. If $K$ is small, then it causes a bias. $K = \text{replicate.datset}$: a variable of the function `error_srsc`.

Running this function, we can see that the error $\bar{\epsilon}(\theta_0, N_I, N_L)$ decreases monotonically as a given number of images $N_I$ or a given number of lesions $N_L$ increases.

Also, the scale of error also will be found. Thus this function can show how our estimates are correct. Scale of error differs for each component of model parameters.

Revised 2019 August 28

Usage

```r
error_srsc(
    NLvector = c(100L, 10000L, 1000000L),
    ratio = 2,
    replicate.datset = 3,
    ModifiedPoisson = FALSE,
    mean.truth = 0.6,
    sd.truth = 5.3,
    z.truth = c(-0.8, 0.7, 2.38),
    ite = 2222,
    cha = 1
)
```

Arguments

- **NLvector**: A vector of positive integers, indicating a collection of numbers of Lesions.
- **ratio**: A positive rational number, with which Number of Images is determined by the formula: (number of images) = ratio times number of lesions. Note that in calculation, it rounds ratio * NLvector to an integer.
- **replicate.datset**: A Number indicate that how many you replicate dataset from user’s specified dataset.
- **ModifiedPoisson**: Logical, that is TRUE or FALSE.
  - If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
  - Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If `ModifiedPoisson = TRUE`, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_4 + F_5}{N_L},$$

$$\frac{F_5}{N_L},$$

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if `ModifiedPoisson = FALSE` (Default), then False Positive Fraction (FPF) is given by

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_4 + F_5}{N_I},$$

$$\frac{F_5}{N_I},$$

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (`ModifiedPoisson = FALSE`).
or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE)

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

mean.truth This is a parameter of the latent Gaussian assumption for the noise distribution.

sd.truth This is a parameter of the latent Gaussian assumption for the noise distribution.

z.truth This is a parameter of the latent Gaussian assumption for the noise distribution.

ite To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

cha To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.

Details

In Bayesian inference, if sample size is large, then posterior tends to the Dirac measure. So, the error and variance of estimates should be tends to zero as sample size tends to infinity.

This function check this phenomenon.

If model has problem, then it contains some non-decreasing vias with respect to sample size.

Revised 2019 Nov 1

Provides a reliability of our posterior mean estimates. Using this function, we can find what digit makes sense.

In the real world, the data for modality comparison or observer performan evaluation is 100 images or 200 images. In such scale data, any estimate of AUC will contain error at most 0.0113... So, the value of AUC should round in 0.XXX and not 0.XXXX or 0.XXXXX or more. Since error is 0.00113... and hence 4 digit or more digit is meaningless. In such manner, we can analyze the errors.

We note that if we increase the number of images or lesions, the errors decrease.
For example, if we use 20000 images in FROC trial, then the error of AUC will be 0.0005... and thus, and so on. Thus large number of images gives us more reliable AUC. However the radiologist cannot read such large (20000) images.

Thus, the error will be 0.00113...

If the number of images are given before hand and moreover if we obtains the estimates, then we can run this function using these two, we can find the estimated errors by simulation. Of course, the estimates is not the truth, but roughly speaking, if we assume that the estimates is not so far from truth, and the error analysis is rigid with respect to changing the truth, then we can say using estimates as truth, the result of this error analysis can be regarded as an actual error.

I want to go home. Unfortunatly, my house is ...

Value

Replicated datasets, estimates, errors,...etc I made this program 1 years ago? and now I forget ... the precise return values. When I see today, 2019 August. It retains too many return values to explain all of them.

Examples

```r
## Not run:
#========================================================================================
# 0) 0-th example
#========================================================================================
# By the following, we can extract only datasets whose
# model has converged.
datasets$convergent.dataList.as.dataframe

#========================================================================================
# 1) 1-st example
#========================================================================================
# Long width is required in R console.

datasets <- error_srsc(
  NLvector = c(100,10000,1000000),
  ite = 2222
)

# By the following, we can extract only datasets whose
# model has converged.
datasets$convergent.dataList.as.dataframe
```

```r
# 1) 1-st example
#========================================================================================
# Long width is required in R console.

datasets <- error_srsc(NLvector = c(50L, 111L, 11111L),
                       # NIvector,
```
error_srsc
	nratio=2,
replicate.dataset =3,
ModifiedPoisson = FALSE,
mean.truth=0.6,
sd.truth=5.3,
z.truth =c(-0.8,0.7,2.38),
ite =2222
)

#========================================================================================
# 2) Plot the error of AUC with respect to NI
#========================================================================================

a <- error_srsc(NLvector = c(
33L,
50L,
111L,
11111L
),
# NIvector,
ratio=2,
replicate.dataset =3,
ModifiedPoisson = FALSE,
mean.truth=0.6,
sd.truth=5.3,
z.truth =c(-0.8,0.7,2.38),
ite =2222
)

aa <- a$Bias.for.various.NL

error.of.AUC <- aa[8,]
y <- subset(aa[8,], select = 2:length(aa[8,]))
y <- as.numeric(y)
y <- abs(y)
upper_y <- max(y)
lower_y <- min(y)
x <- 1:length(y)

plot(x,y, ylim=c(lower_y, upper_y))
# From this plot, we cannot see whether the error has decreased or not.
# Thus, we replot with the log y-axis, the we will see that the error
# has decreased with respect to number of images and lesions.

ggplot(data.frame(x=x, y=y), aes(x = x, y = y)) +
  geom_line() +
  geom_point() +
  scale_y_log10()

# Revised 2019 Sept 25

# General print of log scale
df <- data.frame(x=c(10, 100, 1000, 10, 100, 1000),
                 y=c(1100, 220000, 33000000, 1300, 240000, 36000000),
                 group=c("1", "1", "1", "2", "2", "2")
)

ggplot2::ggplot(df, aes(x = x, y = y, shape = group)) +
  ggplot2::geom_line(position = position_dodge(0.2)) +  # Dodge lines by 0.2
  ggplot2::geom_point(position = position_dodge(0.2), size = 4) +  # Dodge points by 0.2
  ggplot2::scale_y_log10()+
  ggplot2::scale_x_log10()

#========================================================================================
# 2) Add other param into plot plain of the error of AUC with respect to NI
#========================================================================================

a <- error_srsc(NLvector = c(111L, 11111L),
                 # NIvector,
                 ratio = 2,
                 replicate.dataset = 3,
                 ModifiedPoisson = FALSE,
                 mean.truth = 0.6,
                 sd.truth = 5.3,
                 z.truth = c(-0.8, 0.7, 2.38),
                 ite = 2222)

   aa <- a$Bias.for.various.NL
error.of.AUC <- aa[8,]
y1 <- subset(aa[8,], select = 2:length(aa[8,]))
y1 <- as.numeric(y1)
y1 <- abs(y1)

LLL <-length(y1)
y2 <- subset(aa[7,], select = 2:length(aa[7,]))
y2 <- as.numeric(y2)
y2 <- abs(y2)
y <- c(y1,y2)

upper_y <- max(y)
lower_y <- min(y)

group <- rep(seq(1,2,1), 1 , each=LLL)
x <- rep(seq(1,LLL,1), 2 , each=1)
group <- as.character(group)
df <- data.frame(x=x,y=y,group=group)

ggplot2::ggplot(df, aes(x = x, y = y, shape = group)) + 
ggplot2::geom_line(position = position_dodge(0.2)) + 
  ggplot2::geom_point(position = position_dodge(0.2), size = 4)+ 
  ggplot2::scale_y_log10() 

# Confidence level = 4
# ==-----------------------------------------------------------------------------------------------

datasets <-error_srsc(NLvector = c(111L,11111L), 
  ratio=2, 
  # NIvector, 
  # ratio=2,
replicate.dataset = 3,
ModifiedPoisson = FALSE,
mean.truth = -0.22,
sd.truth = 5.72,
z.truth = c(-0.46, -0.20, 0.30, 1.16),
ite = 2222
)

error_ssrc_variance_visualization(datasets)

# The parameter of model is 7 in which the ggplot2 fails with the following warning:
# The shape palette can deal with a maximum of 6 discrete values because more than 6
# becomes difficult to
# discriminate; you have 7. Consider specifying shapes manually if you must have them.

## End(Not run)# dontrun

error_ssrc_error_visualization

Visualization for Error of Estimator

Description
The function plot the graph of errors with respect to sample sizes.

Error plot

x-axis Sample sizes

y-axis Error for each parameter

Usage

error_ssrc_error_visualization(
  return.value.of_error_ssrc,
  log_scale_x.axis = TRUE
)
error_srsc_error_visualization

**Arguments**

- **return.value.of_error_srsc**
  A return value of the function `error_srsc()`.

- **log_scale_x.axis**
  A logical, whether x axis is log scale or not.

**Value**

A long format dataframe of error and its parameter name

**See Also**

`error_srsc_variance_visualization`

**Examples**

```r
# General plot

df <- data.frame(x=runif(100),y=runif(100),g= as.factor(rep(1:5,10)))

ggplot(df, aes(x = x, y = y, shape = g)) +
  geom_point(size = 3) +
  scale_shape_manual(values = c(1,2,3,4,5,6,7,8,9))


df <- data.frame(x=runif(100),y=runif(100),g= as.factor(rep(1:25,4)))

  # Use slightly larger points and use custom values for the shape scale

  gggplot(df, aes(x = x, y = y, shape = g)) +
  geom_point(size = 3) +
  scale_shape_manual(values = c(1,2,3,4,5,6,7,8,9,10,
  11,12,13,14,15,16,17,18,19,20,21,22,23,24,25))

  # Not run:
  a <- error_srsc()

  error_srsc_error_visualization(a)

  #========================================================================================
  # In case of C = 4, arbitrary C is available.
error_srsc_error_visualization

#========================================================================================

a <- error_srsc(NLvector = c(100, 10000, 1000000),
                 ratio = 2, replicate.dataset = 2,
                 ModifiedPoisson = FALSE,
                 mean.truth = 0.6, sd.truth = 5.3,
                 z.truth = c(-0.8, 0.7, 2.38, 3), # Here we use the C=4
                 ite = 500)

error_srsc_error_visualization(a)
error_srsc_variance_visualization(a)

#========================================================================================

# In case of C = 7, arbitrary C is available.

#========================================================================================

a <- error_srsc(NLvector = c(100, 10000, 1000000),
                 ratio = 2, replicate.dataset = 2,
                 ModifiedPoisson = FALSE,
                 mean.truth = 0.6, sd.truth = 5.3,
                 z.truth = c(-0.8, 0.7, 2.38, 3, 3.4, 3.6, 3.8), # Here we use the C=7
                 ite = 500)
Description

Visualization Of variance Analysis

Usage

```r
error_srsc_variance_visualization(  
  return.value.of_error_srsc,  
  log_scale_x.axis = TRUE
)
```

Arguments

- `return.value.of_error_srsc`
  A return value of the function `error_srsc()`.
- `log_scale_x.axis`
  A logical, whether x axis is log scale.

Value

A long format dataframe of error and its parameter name
Examples

### Not run:

```r
a <- error_srsc()
error_srsc_variance_visualization(a)
```

```r
a <- error_srsc(replicate.dataset = 10)
error_srsc_variance_visualization(a)
```

### End(Not run)

---

**explanation_about_package_BayesianFROC**

*Explanation of this package*

---

**Description**

In R console, explanation are shown.

**Usage**

```r
explanation_about_package_BayesianFROC()
```

**Examples**

```r
explanation_about_package_BayesianFROC()
```

---

**explanation_for_what_curves_are_drawn**

*Print out about what curves are drawn*

---

**Description**

For package developer.

**Usage**

```r
explanation_for_what_curves_are_drawn(modalityID, readerID)
```
**Arguments**

- **modalityID**: A vector.
- **readerID**: A vector.

**Value**

Nothing

**Examples**

```r
## Not run:
#================The first example======================================
modalityID <- c(1,2)
readerID <- c(1,2,3)
explanation_for_what_curves_are_drawn(modalityID, readerID)

#================The second example======================================
modalityID <- 1
readerID <- c(1,2,3)
explanation_for_what_curves_are_drawn(modalityID, readerID)

## End(Not run)
```

**extractAUC**

*Extract AUC*

**Description**

Extract AUC for both srsc and MRMC data.

**Usage**

```r
extractAUC(
  StanS4class,
  dig = 3,
  summary = TRUE,
  new.imaging.device = TRUE,
)```
extract_data_frame_from_dataList_MRMC

Extract sub data frame from list of FROC data

Description

Make a dataframe from a list consisting of vectors m, q, c, h, f and positive integers NL, C, M, Q, NI. So, resulting data-frame is constructed by vectors m, q, c, h, f.

Usage

extract_data_frame_from_dataList_MRMC(dataList)

Arguments

dataList A list of MRMC data.
Value

A data frame consisting of vectors \( m, q, c, h, f \).

\( m \) A vector of positive integers, representing the modality ID vector.

\( q \) A vector of positive integers, representing the reader ID vector.

\( c \) A vector of positive integers, representing the confidence level. This vector must be made by 
\[
\text{rep(rep(C:1),M*Q)}
\]

\( h \) A vector of non-negative integers, representing the number of hits.

\( f \) A vector of non-negative integers, representing the number of false alarms.

Examples

```r
## Not run:

#========================================================================================
# From example dataset named ddddd
#========================================================================================

## Only run examples in interactive R sessions
if (interactive()) {
  fit_GUI_Shiny_MRMC(DF=extract_data_frame_from_dataList_MRMC(dddddd))
}
## Only run examples in interactive R sessions

## End(Not run)
```

---

**extract_EAP_by_array**  
Extract Estimates Preserving Array Format.

Description

Extract posterior mean estimates (EAP) by array format.

Usage

```
extract_EAP_by_array(StanS4class, name.of.parameter)
```
extract_EAP_by_array

Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc.

name.of.parameter  An parameter name (given as a character string, should not surround by ""). The name of parameter which user want to extract. Parameters are contained in the parameter block of each Stan file in the path: inst/extdata.

Details

If an estimate is an array, then this function extract estimated parameters preserving an array format. The rstan also has such function, i.e., rstan::get_posterior_mean(). However this function does not extract parameter as an array but coerce to the class matrix.

Value

A list of datalists from the posterior predictive distribution

Examples

## Not run:
#=================================The first example: MRMC case ========================
#========================================================================================
# MRMC case: Extract a estimates from fitted model objects
#========================================================================================

# Make a fitted model object of class stanfitExtended
# which is inherited from the S4class stanfit.
# The following example, fitted model is the hierarchical Bayesian FROC model
# which is used to compare modality.

fit <- fit_Bayesian_FROC( ite = 1111 ,
                         summary = FALSE ,
                         dataList = dataList.Chakra.Web.orderd,
                         cha=1 )

# Extract one dimensional array "z = z[]",

z <- extract_EAP_by_array( fit,  # The above fitted model object
                           z     # One of the parameter in "fit"
                     )

# Extract two dimensional array "AA = AA[ , ]",
AA <- extract_EAP_by_array(
  fit,
  AA
)

# Extract three dimensional array "ppp = ppp[, , ]",
ppp <- extract_EAP_by_array(fit, ppp)

#================= The second example: singler reader and single modality ==============
#========================================================================================
# srsc case: Extract a estimates from fitted model objects
#========================================================================================

# Of course, for the case of srsc, it is also available.
# We shall show the case of srsc in which case the parameters are not array,
# but in such a case we can extract estimates preserving its format such as vector.
fit <- fit_Bayesian_FROC( ite = 1111 ,
  summary = FALSE ,
  dataList = dataList.Chakra.1,
  cha=2
)

# To extract the posterior mean for parameter "A" representing AUC, we run the following;

A <- extract_EAP_by_array(
  fit,
  A
)

# To extract the posterior mean for parameter "z" indicating decision thresholds;

z <- extract_EAP_by_array(
  fit,
  z
)

# 2019.05.21 Revised.
# name.of.parameter surrounded by double quote is also available
#========================================================================================

# Let fit be the above fitted model object.
# Then the following two codes are same.

extract_EAP_by_array( fit, "A" )

extract_EAP_by_array( fit, A )

# Unfortunately, the later case sometimes cause the R CMD check error which said
# that no visible binding, since object A is not defined.
# For example, if we use the later in the function: metadata_to_DrawCurve_MRMC
# Then R command said some NOTE that

# > checking R code for possible problems ... NOTE
# metadata_to_DrawCurve_MRMC: no visible binding for global variable 'A'
# Undefined global functions or variables: A

# Revised 2019 Oct 19

# I am not sure, does this package development make me happy?
# Back pain being due to an abnormality in my immune system, which is caused
# my exposure to surfactants or latex (not LaTeX).

## End(Not run)# Revised 2019 Jun 19

---

extract_EAP_CI  

MRMC: Extract Estimates of a vector from stanfitExtended object

**Description**

We extract the EAPs and CIs from the stanfitExtended S4 class which is an inherited class of the stanfit S4 class.

**Usage**

```r
extract_EAP_CI(
    StanS4class,
    parameter.name,
```
Arguments

StanS4class  An S4 object of the class stanfit. No need that it is the S4 class stanfitExtended.
parameter.name  character vector. E.g., use as "aaa". for names of parameter described in the parameter block of stan file.
dimension.of.parameter  If parameter aaa is vector, i.e.,aaa[1],aaa[2],...aaa[6] then dimension.of.parameter = 6
dig  To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5,
summary  Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Details

To validate our model has no bias, that is comparison of true parameters of distributions and EAPs, we have to extract the estimates from the stanfitExtended object. And this function do it.

Value

EAPs, CI.

See Also

extract_estimates_MRMC

Examples

```r
## Not run:
# First we create the following fitted model object of class stanfitExtend.

fit <- fit_Bayesian_FROC(
  dataList.Chakra.Web.orderd, # data
  ite = 1111,               # MCMC iteration
  summary = FALSE          # verbose
)

# Second, to extract the EAPs of the parameter z,
# we also have to specify the dimension of vector z as follows.
```
extract_EAP_CI(
    fit, # The above fitted model object
    "z", # The parameter name described in parameter block of stan file
    5  # The dimension of vector z
)

# One more example: to extract the EAPs of the parameter dz,
# we also have to specify its dimension of vector dz as follows.
list.of.dz <- extract_EAP_CI(fit,"dz",4)

# One more example: to extract the EAPs of the parameter w,
# we also have to specify its dimension of vector w as follows.
list.w <- extract_EAP_CI(fit,"w",1)

# Note that this function can extract only parameter of "vector" and not "array" !!
# To extract such array please use "extract_estimates_MRMC()"
# which extract all parameters from a hierarchical Bayesian model
# estimated from user data. So, this function is no longer meaningless,
# and I will delete this.

# I forgot where I use this function
# 2019.05.21 Revised.

# the following gives convergence seed 2019 Oct 12
f <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = ddd ,see = 123456)
z <- extract_EAP_CI(f,"z",f@dataList$C )$z.EAP
#usethis::use_data(z)
#usethis package cannot be to use since it is not declared in NAMESPACE.

dz <- extract_EAP_CI(f,"dz",f@dataList$C-1 )$dz.EAP
#usethis::use_data(dz)
#usethis package cannot be to use since it is not declared in NAMESPACE.

## End(Not run)# dottest
MRMC: Extract All Posterior Mean Estimates from stanfitExtended object

Description
Extract Posterior Mean estimates, preserving its format, such as array, vector. From MRMC models, it extract the EAPs and CIs.

Usage
extract_estimates_MRMC(StanS4class, dig = 3)

Arguments
StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and etc
dig  To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

Details
To validate our model has no bias, that is comparison of true parameters of distributions and EAPs, we have to extract the estimates from the stanfitExtended object. And this function do it.

Value
EAPs, CIs which preserving its format, such as array, vector.

See Also
extract_EAP_CI() is used in the function extract_estimates_MRMC().

Examples
## Not run:

```r
fit <- fit_Bayesian_FROC(
    BayesianFROC::dataList.Chakra.Web.orderd, 
    summary = FALSE, 
    ite=111) 
EAPs <- extract_estimates_MRMC(fit)
```
Extract Estimates From Replicated MRMC Model

**Description**

Extract Estimates From Replicated MRMC Model

**Usage**

```r
extract_parameters_from_replicated_models(
  initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth,
  NI = 200,
  NL = 142,
  ModifiedPoisson = FALSE,
  replication.number = 2,
  summary = FALSE,
  ite = 1111
)
```

**Arguments**

- `initial.seed`: The variable `initial.seed` is used to replicate datasets. That is, if you take `initial.seed = 1234`, then the seed 1234, 1235, 1236, 1237, 1238, etc are for the first replication, the second replication, the third replication, etc. If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.
- `mu.truth`: array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.
- `v.truth`: array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
- `z.truth`: This is a parameter of the latent Gaussian assumption for the noise distribution.
- `NI`: Number of Images.
- `NL`: Number of Lesions.
- `ModifiedPoisson`: Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$ )

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]
\[ \frac{F_0}{N_I} \]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the *False Positive Fraction (FPF) per image.*

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \))

or as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{TRUE} \))

If \( \text{ModifiedPoisson} = \text{TRUE} \), then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \), then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

**replication.number**

For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.

**summary**

Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**iter**

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

**Value**

A list of estimates, posterior means and posterior credible intervals for each model parameter. EAPs and CI interbals.

**Examples**

```r
## Not run:
```
false_and_its_rate_creator

False Alarm Creator for both cases of MRMC and srsc

Description

From threshold, mean and S.D., data of False Alarm are created.

Usage

false_and_its_rate_creator(
  z.truth = BayesianFROC::z_truth,
  NI = 333,
  NL = 111,
  ModifiedPoisson = FALSE,
  seed = 12345
)

Arguments

z.truth Vector of dimension = C represents the thresholds of bi-normal assumption.
NI The number of images.
NL The number of lesions.
ModifiedPoisson Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_4 + F_5}{N_L},$$

$$\frac{F_5}{N_L},$$

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion. On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_4 + F_5}{N_I},$$

$$\frac{F_5}{N_I},$$

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be graded as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = FALSE).
false_and_its_rate_creator

or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE).

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed, thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is a redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

seed

The seed for creating a collection of the number of false alarms synthesized by the Poisson distributions using the specified seed.

Details

From threshold, mean and S.D. of the latent Gaussian noise distribution in the bi-normal assumption, data of False Alarm are created. For the process of this drawing false alarm samples, its rate are also created. So, in the return values of the function, the rates for each confidence level is also attached.

Value

A list of vectors, indicating a true parameter and a sample.

A vector indicating a true parameter: False rate from thresholds.

A vector indicating a sample, more precisely, The truth parameter of false alarm rate calculated by true thresholds $z$ and also, one-time drawn samples of false alarms from the calculated false rates.

Examples

```r
## Not run:
false.rate <- false_and_its_rate_creator()

#========================================================================================
# In SBC, Poisson rate = 0,...so,... i have to investigate.
#========================================================================================

set.seed( 1234 )

dz <- runif( 3, 0.01, 1 )
```
false_and_its_rate_creator

w <- rnorm(1,
0,
1
)

z <- z_from_dz(w,dz )

false_and_its_rate_creator(z )

#========================================================================================
# Poisson rate is OK
#========================================================================================

set.seed( 1234 )
dz <-runif(3, # sample size
0.01, # lower bound
1 # upper bound
)

w <- rnorm(1,
0,
10 # It cause the poisson rate become small
)

z <- z_from_dz(w,dz )

false_and_its_rate_creator(z )

#========================================================================================
# In SBC, Poisson rate is small
#========================================================================================

set.seed( 1234 )
dz <-runif(3, # sample size
0.01, # lower bound
1 # upper bound
)

w <- rnorm(1,
0,
false_and_its_rate_creator_MRMC

MRMC: False Alarm Creator For each Modality and each Reader.

Description

From threshold, mean and S.D., data of False Alarm are created.

Usage

false_and_its_rate_creator_MRMC(
  z.truth = BayesianFROC::z_truth,
  NI = 333,
  NL = 111,
  ModifiedPoisson = FALSE,
  seed = 12345,
  M = 5,
)
\[
Q = 4, \\
\text{summary} = \text{TRUE}
\]

Arguments

\text{ntr}
Vector of dimension = C represents the thresholds of bi-normal assumption.
\text{NI}
The number of images.
\text{NL}
The number of lesions.
\text{ModifiedPoisson}
Logical, that is \text{TRUE} or \text{FALSE}.

If \text{ModifiedPoisson} = \text{TRUE}, then Poisson rate of false alarm is calculated \textit{per lesion}, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF \textit{per lesion}.

Similarly,
If \text{ModifiedPoisson} = \text{TRUE}, then Poisson rate of false alarm is calculated \textit{per image}, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF \textit{per image}.

For more details, see the author’s paper in which I explained \textit{per image} and \textit{per lesion}. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If \text{ModifiedPoisson} = \text{TRUE}, then the False Positive Fraction (FPF) is defined as follows (\(F_c\) denotes the number of false alarms with confidence level \(c\))

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where \(N_L\) is a number of lesions (signal). To emphasize its denominator \(N_L\), we also call it the \textit{False Positive Fraction (FPF)} \textit{per lesion}.

On the other hand,

if \text{ModifiedPoisson} = \text{FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by
\[ \frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I}, \]
\[ \frac{F_2 + F_3 + F_4 + F_5}{N_I}, \]
\[ \frac{F_3 + F_4 + F_5}{N_I}, \]
\[ \frac{F_4 + F_5}{N_I}, \]
\[ \frac{F_5}{N_I}, \]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \)) or as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{TRUE} \)).

If \( \text{ModifiedPoisson} = \text{TRUE} \), then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \), then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

**seed**

The seed for creating a collection of the number of false alarms synthesized by the Poisson distributions using the specified seed.

**M**

Number of modalities

**Q**

Number of readers

**summary**

Logical: \( \text{TRUE} \) or \( \text{FALSE} \). Whether to print the verbose summary. If \( \text{TRUE} \) then verbose summary is printed in the \( \texttt{R} \) console. If \( \text{FALSE} \), the output is minimal. I regret, this variable name should be verbose.
Details
In our model, false alarm rate does not depend on the readers or modalities. Thus this sampling function merely synthesizes samples from the Poisson distribution of the same false alarm rate. Of course, this same false rate of the Poisson distributions is not desired one. Since we should assume that each reader with different modality should differ. To accomplish this, we have to assume that threshold parameter of Gaussian assumption should depend on the reader and modality. However, such model does not converge in the Hamiltonian Monte Carlo simulation.

Value
Vector for false alarms as an element of list of MRMC data.

Examples
```r
## Not run:

false_and_its_rate_creator_MRMC()

## End(Not run)
```

Package Development tools and memo.

Description
This is for the author of this package. project option build and reload

Usage
`fffaaabbb()`

Fit a model to data

Description
Fit a model to data.
fit_a_model_to

Usage

```
fit_a_model_to(
  dataList,
  number_of_chains_for_MCMC = 1,
  number_of_iterations_for_MCMC = 1111,
  seed_for_MCMC = 1234,
  ...
)
```

Arguments

dataList: A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also.

The dataList will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`.

For the single reader and a single modality data, the dataList is made by the following manner:

```
dataList.Example <- list(  
  h = c(41,22,14,8,1),# number of hits for each confidence level  
  f = c(1,2,5,11,13),# number of false alarms for each confidence level  
  NL = 124,# number of lesions (signals)  
  NI = 63,# number of images (trials)  
  C = 5) # number of confidence, .. the author thinks it can be calculated as the length of h or f ...? ha,why I included this. ha .. should be omitted.
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList` representing FROC data, this package provides three functions:

- `convertFromJafroc()` If data is a JAFROC xlsx formulation.
- `dataset_creator_new_version()` Enter TP and FP data by table.
- `create_dataset()` Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function `viewdata()`.

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), `dataList` is a list consisting of `f`, `h`, `NL`, `NI`, `C` where `f`, `h` are numeric vectors and `NL`, `NI`, `C` are positive integers.

- `f` Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

A positive integer, representing Number of Lesions.

A positive integer, representing Number of Images.

A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by \( C \), are included, however. Note that confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(\text{rep}(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

data Format:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>definitely present</td>
<td>c[1] = 5</td>
<td>f[1] = ( F_1 ) = 1</td>
<td>h[1] = ( H_1 ) = 41</td>
</tr>
</tbody>
</table>

\* false alarms = False Positives = FP

\* hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(\text{rep}(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c \leftarrow c(\text{rep}(C:1)) \) via a table which can be displayed by the function \text{viewdata}().
Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an \( \mathbb{R} \) list object representing FROC data must contain components \( m, q, c, h, f, N_{L}, C, M, Q \).

- \( C \): A positive integer, representing the highest number of confidence level, this is a scalar.
- \( M \): A positive integer vector, representing the number of modalities.
- \( Q \): A positive integer, representing the number of readers.
- \( m \): A vector of positive integers, representing the modality ID vector.
- \( q \): A vector of positive integers, representing the reader ID vector.
- \( c \): A vector of positive integers, representing the confidence level. This vector must be made by `\text{rep}(\text{rep}(C:1),M*Q)`
- \( h \): A vector of non-negative integers, representing the number of hits.
- \( f \): A vector of non-negative integers, representing the number of false alarms.
- \( N_{L} \): A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above \( \mathbb{R} \) object. However, each confidence level vector is not included in the data, because it is created automatically from \( C \). To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

Example data.

Multiple readers and multiple modalities (i.e., MRMC)
---

*false alarms* = False Positives = FP  
*hits* = True Positives = TP

number_of_chains_for_MCMC
A positive integer, indicating the number of chains for MCMC. To be passed to the function `rstan::sampling()` in `rstan`.

number_of_iterations_for_MCMC
A positive integer, indicating the number of iterations for MCMC. To be passed to the function `rstan::sampling()` in `rstan`.

seed_for_MCMC
A positive integer, indicating the seed for MCMC. To be passed to the function `rstan::sampling()` in `rstan`.

...  
Additional arguments

Details
The author made a function `fit_Bayesian_FROC()` which has very redundant variables. So, `fit_a_model_to()` is made by simplifying `fit_Bayesian_FROC()` so that its variables is minimum. To access full details, see the help of `fit_Bayesian_FROC()`.

This function aims to give a simple interface by ignoring unnecessarily parameters of `fit_Bayesian_FROC()`.

Value
An fitted model object of the S4 class named `stanfitExtended` which is an inherited class from `stanfit`.

See Also

`fit_Bayesian_FROC()`

Examples

```r
## Not run:
#========================================================================================
# 1) Build a data-set
#========================================================================================
# For a single reader and a single modality case.

data <- list(c=c(3,2,1), # Confidence level. Note that c is ignored.
h=c(97,32,31), # Number of hits for each confidence level
f=c(1,14,74), # Number of false alarms for each confidence level
NL=259, # Number of lesions
NI=57, # Number of images
C=3) # Number of confidence level
```
viewdata(data)

# where,
# c denotes confidence level, i.e., rating of reader.
#   3 = Definitely diseased,
#   2 = subtle,... diseased
#   1 = very subtle
# h denotes number of hits (True Positives: TP) for each confidence level,
# f denotes number of false alarms (False Positives: FP) for each confidence level,
# NL denotes number of lesions,
# NI denotes number of images,

# For example, in the above example data,
# the number of hits with confidence level 3 is 97,
# the number of hits with confidence level 2 is 32,
# the number of hits with confidence level 1 is 31,
#
# the number of false alarms with confidence level 3 is 1,
# the number of false alarms with confidence level 2 is 14,
# the number of false alarms with confidence level 1 is 74,

#========================================================================================
# 2) Fit an FROC model to the above dataset.
#========================================================================================

fit <- BayesianFROC::fit_a_model_to(
  # Dataset to be fitted
dataList = data,

  # To run in time <5s, MCMC iterations too small to obtain reliable estimates
number_of_iterations_for_MCMC = 1111,

  # The number of chains, it is better if larger.
number_of_chains_for_MCMC = 1
)

## End(Not run)#dontrun
Description

Creates a fitted model object of class `stanfitExtended`: an inherited class from the S4 class `stanfit` in `rstan`.

Usage

```r
fit_Bayesian_FROC(
    dataList,
    ModifiedPoisson = FALSE,
    prior = -1,
    zz = 1,
    verbose = TRUE,
    print_CI_of_AUC = TRUE,
    model_reparametrized = FALSE,
    Model_MRMN_non_hierarchical = TRUE,
    prototype = FALSE,
    PreciseLogLikelihood = TRUE,
    DrawCurve = length(dataList$m) == 0,
    Drawcol = TRUE,
    summary = TRUE,
    make.csv.file.to.draw.curve = FALSE,
    mesh.for.drawing.curve = 10000,
    significantLevel = 0.7,
    new.imaging.device = TRUE,
    cha = 1,
    ite = 10000,
    DrawFROCcurve = TRUE,
    DrawAFROCCurve = FALSE,
    DrawCFPCTP = TRUE,
    dig = 5,
    war = floor(ite/5),
    see = 1234567,
    Null.Hypothesis = FALSE,
    ...
)
```

Arguments

dataList  A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also.

The `dataList` will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and a single modality data, the `dataList` is made by the following manner:

```r
dataList.Example <- list(
    h = c(41,22,14,8,1), # number of hits for each confidence level
    ...)
```
fit_Bayesian_FROC

\[ f = c(1, 2, 5, 11, 13), \# \text{number of false alarms for each confidence level} \]

\[ NL = 124, \# \text{number of lesions (signals)} \]
\[ NI = 63, \# \text{number of images (trials)} \]
\[ C = 5, \# \text{number of confidence, the author thinks it can be calculated as the length of } h \text{ or } f \ldots \] ha, why I included this. ha \ldots \text{should be omitted.} \]

And using this object \texttt{dataList.Example}, we can apply \texttt{fit_Bayesian_FROC()} such as \texttt{fit_Bayesian_FROC(dataList.Example)}.

To make this \texttt{R} object \texttt{dataList} representing FROC data, this package provides three functions:

- \texttt{convertFromJafroc()} (If data is a \texttt{JAFROC xlsx} formulation).
- \texttt{dataset_creator_new_version()} (Enter TP and FP data \texttt{by table}).
- \texttt{create_dataset()} (Enter TP and FP data \texttt{by interactive} manner).

Before fitting a model, we can confirm our dataset is correctly formulated by using the function \texttt{viewdata()}.

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), \texttt{dataList} is a list consisting of \( f, h, NL, NI, C \) where \( f, h \) are numeric vectors and \( NL, NI, C \) are positive integers.

- \( f \) Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- \( h \) Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- \( NL \) A positive integer, representing Number of Lesions.
- \( NI \) A positive integer, representing Number of Images.
- \( C \) A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by \( C \), are included, however, Note that confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c <- c(rep(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

**data Format:**

*A single reader and a single modality case*

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>
definitely present  \( c[1] = 5 \)  \( f[1] = F_5 = 1 \)  \( h[1] = H_5 = 41 \)
very subtle  \( c[5] = 1 \)  \( f[5] = F_1 = 13 \)  \( h[5] = H_1 = 1 \)

* false alarms = False Positives = FP
* hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(rep(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c \leftarrow c(rep(C:1)) \) via a table which can be displayed by the function \texttt{viewdata()}.

### Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function \texttt{fit_Bayesian_FROC()}, dataset represented by an \texttt{R} list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

- \( C \) A positive integer, representing the highest number of confidence level, this is a scalar.
- \( M \) A positive integer vector, representing the number of modalities.
- \( Q \) A positive integer, representing the number of readers.
- \( m \) A vector of positive integers, representing the modality ID vector.
- \( q \) A vector of positive integers, representing the reader ID vector.
- \( c \) A vector of positive integers, representing the confidence level. This vector must be made by \( rep(rep(C:1),M*Q) \)
- \( h \) A vector of non-negative integers, representing the number of hits.
- \( f \) A vector of non-negative integers, representing the number of false alarms.
- \( NL \) A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above \texttt{R} object. However, each confidence level vector is not included in the
data, because it is created automatically from c. To confirm false positives and
hits are correctly ordered with respect to the automatically generated confidence
vector,
the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019
Dec 5

*Example data.*

*Multiple readers and multiple modalities (i.e., MRMC)*

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

ModifiedPoisson

Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per lesion*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF *per lesion*.

Similarly,

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per image*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF *per image*.

For more details, see the author’s paper in which I explained *per image* and *per lesion*. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows (*F_c* denotes the number of false alarms with confidence level *c*)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]
where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \) (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \)) or as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{TRUE} \)).

If \( \text{ModifiedPoisson} = \text{TRUE} \), then FROC curve means the expected pair of FPF per lesion and TPF.
On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

prior  positive integer, to select the prior
zz    A real number specifying one of the parameter of prior
verbose  A logical, if TRUE, then the redundant summary is printed in \texttt{R} console. If FALSE, it suppresses output from this function.

print_CI_of_AUC  Logical, if TRUE then Credible intervals of AUCs for each modality are plotted.
model_reparametrized  A logical, if TRUE, then a model under construction is used.
Model_MRM_non_hierarchical  A logical. If TRUE, then the model of multiple readers and multiple modalities consists of no hyper parameters. The reason why the author made this parameter is that the hyper parameter make the MCMC posterior samples be unstable. And also, my hierarchical model is not so good in theoretical perspective. Thus, I made this. The Default is TRUE.
prototype  A logical, if TRUE then the model is no longer a generative model. Namely, in generally speaking, a dataset drawn from the model cannot satisfy the condition that the sum of the numbers of hits over all confidence levels is bounded from the above by the number of lesions, namely,

$$\sum c H_c \leq N_L$$

However, this model (TRUE ) is good in the sense that it admits various initial values of MCMC sampling.

if FALSE, then the model is precisely statistical model in the sense that any dataset drawn from the model satisfies that the sum of the number of hits is not greater than the number of lesions, namely,

$$\sum c H_c \leq N_L.$$

This model is theoretically perfect. However, in the practically, the calculation will generates some undesired results which caused by the so-called flood .... I forget English :-D. The flood point??! I forgseeeeeeexesss!! Ha. So, prior synthesizes very small hit rates such as 0.0000000000000001234 and it cause the non accurate calculation such as 0.00000,..0000123/0.000....00012345=
0.0012 which becomes hit rate and thus OH No!. Then it synthesizes Bernoulli success rate which is not less than 1 !! To avoid this, the author should develop the theory of prior to avoid this very small numbers, however the author has idea but now it does not success.

If \texttt{prototype = TRUE}, then the model for hits is the following:

\[
\begin{align*}
H_5 &\sim \text{Binomial}(p_5, N_L) \\
H_4 &\sim \text{Binomial}(p_4, N_L) \\
H_3 &\sim \text{Binomial}(p_3, N_L) \\
H_2 &\sim \text{Binomial}(p_2, N_L) \\
H_1 &\sim \text{Binomial}(p_1, N_L)
\end{align*}
\]

On the other hand, if \texttt{prototype = FALSE}, then the model for hits is the following:

\[
\begin{align*}
H_5 &\sim \text{Binomial}(p_5, N_L) \\
H_4 &\sim \text{Binomial}(\frac{p_4}{1 - p_5}, N_L - H_5) \\
H_3 &\sim \text{Binomial}(\frac{p_3}{1 - p_5 - p_4}, N_L - H_5 - H_4) \\
H_2 &\sim \text{Binomial}(\frac{p_2}{1 - p_5 - p_4 - p_3}, N_L - H_5 - H_4 - H_3) \\
H_1 &\sim \text{Binomial}(\frac{p_1}{1 - p_5 - p_4 - p_3 - p_2}, N_L - H_5 - H_4 - H_3 - H_2)
\end{align*}
\]

Each number of lesions is adjusted so that the sum of hits \(\sum c H_c\) is less than the number of lesions (signals, targets) \(N_L\). And hence the model in case of \texttt{prototype = FALSE} is a generative model in the sense that it can replicate datasets of FROC arises. Note that the adjustment of the number of lesions in the above manner leads us the adjustment of hit rates. The reason why we use the hit rates such as \(\frac{p_2}{1 - p_5 - p_4 - p_3 - p_2}\) instead of \(p_c\) is that it ensures the equality \(E[H_c/N_L] = p_c\). This equality is very important. To establish Bayesian FROC theory so that it is compatible to the classical FROC theory, we need the following two equations,

\[
\begin{align*}
E[H_c/N_L] &= p_c, \\
E[F_c/N_X] &= q_c,
\end{align*}
\]

where \(E\) denotes the expectation and \(N_X\) is the number of lesion or the number of images and \(q_c\) is a false alarm rate, namely, \(F_c \sim \text{Poisson}(q_c N_X)\).

Using the above two equations, we can establish the alternative Bayesian FROC theory preserving classical notions and formulas. For the details, please see the author’s pre print:

Bayesian Models for ... for?? I forget my paper title .... :'-(D. What the hell!!? I forget,... My health is so bad to forget , ... I forget.
The author did not notice that the prototype is not a generative model. And hence the author revised the model so that the model is exactly generative model. But the reason why the author remains the prototype model (prototype = TRUE) is that the convergence of MCMC sampling in case of MRMC is not good in the current model (prototype = FALSE). Because it uses fractions $\frac{p_1}{1-p_5-p_4-p_3-p_2}$ and which is very dangerous to numerical perspective. For example, if $p_1$ is very small, then the numerator and denominator of $\frac{p_1}{1-p_5-p_4-p_3-p_2}$ is very small. Both of them is like 0.000000000000000123.... and such small number causes the non accurate results. So, sometimes, it occurs that $\frac{p_1}{1-p_5-p_4-p_3-p_2} > 1$ which never occur in the theoretical perspective but unfortunately, in numerically occurs.

SO, now, the author try to avoid such phenomenon by using priors but it now does not success.

Here of course we interpret the terms such as $N_L - H_5 - H_4 - H_3$ as the remained targets after reader get hits. The author thinks it is another manner to do so like $N_L - H_1 - H_2 - H_3$, but it does not be employed. Since the author thinks that the reader will assign his suspicious lesion location from high confidence level and in this view point the author thinks it should be considered that targets are found from the highest confidence suspicious location.

**PreciseLogLikelihood**
Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE (default), then Stan calculates the precise log likelihood with target formulation. If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

**DrawCurve**
Logical: TRUE or FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve = TRUE, if not then DrawCurve = FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.

**Drawcol**
Logical: TRUE or FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The Default value is a TRUE.

**summary**
Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**make.csv.file.to.draw.curve**
Logical: TRUE or FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.

**mesh.for.drawing.curve**
A positive large integer, indicating number of dots drawing the curves, Default = 10000.

**significantLevel**
This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.
null.Hypothesis Logical, that is TRUE or FALSE. If Null.or.Alternative.Hypothesis = FALSE(default), then fit the alternative model to dataList (for details of models, see vignettes). If Null.or.Alternative.Hypothesis = TRUE, then fit the null model to dataList.(for details of models, see vignettes). Note that the null model is constructed under the null hypothesis that all modality are same observer performance ability. The alternative model is made under the assumption that all modality are not same. The reason why author creates this parameter is to test the null hypothesis by the Bayes factor. But the result of test is not desired one for me. Thus the test is under construction.

Details

Draw MCMC samples using R package: rstan
It also plots FROC curves if a single reader and a single modality case. For details, see vignettes
Build the S4 object by Stan to fit the author’s Bayesian models introduced in the author’s paper (for details of models, see vignettes). The output of the rstan::sampling() is an object of the S4 class.
called `stanfit`. But, in this package, we extended the `stanfit` class to an S4 class named `stanfitExtended`. The new S4 class `stanfitExtended` included new slots for sequential analysis. So, the return value of the function is not the S4 class `stanfit` but the new S4 class `stanfitExtended`. Thus, to apply the functions in the `rstan` package for fitted model objects, we have to change the class of the S4 fitted model objects using the function `methods::as()` such as by the code `methods::as(object = fitted.model.object,"stanfit").

The following items are main substances of this function.

This function `fit_Bayesian_FROC` is available both a single reader and a single modality case and multiple readers and multiple modality case. Confidence level vector is not required but it is implicitly referred as the decreasing order. For example, if C=3, then it would be a form c=c(3,2,1,3,2,1,...). Even if you write your data according to the order c=c(1,2,3,1,2,3,...), the program does not consider as your order, but c=c(3,2,1,3,2,1,...) instead.

**Value**

An object of class `stanfitExtended` which is an inherited S4 class from the S4 class `stanfit` By `rstan::sampling`, the function fit the author’s FROC Bayesian models to user data.

Use this fitted model object for sequential analysis, such as drawing the FROC curve and alternative FROC (AFROC) curves.

Notations and symbols for the **Outputs of a single reader and a single modality case**

In the following, the notations for estimated parameters are shown.

* $w$ A real number representing the **lowest threshold** of the Gaussian assumption (bi-normal assumption). so $w=z[1]$.


* ..., $m$ A real number representing the The **mean** of the Latent Gaussian distribution for diseased images. In TeX, it denoted by $\mu$.

* $v$ A positive real number representing the **standard deviation** of the Latent Gaussian distribution for diseased images. In TeX, it will be denoted by $\sigma$, not the square of $\sigma$.

* $p[1]$ A real number representing the Hit rate with confidence level 1.

* $p[2]$ A real number representing the Hit rate with confidence level 2.


* ..., $l[1]$ A positive real number representing the (Cumulative) False positive rate with confidence level 1. In TeX, it will be denoted by $\lambda_1$. 
A positive real number representing the (Cumulative) False positive rate with confidence level 2. In TeX, it will be denoted by $\lambda_2$.

A positive real number representing the (Cumulative) False positive rate with confidence level 3. In TeX, it will be denoted by $\lambda_3$.

A positive real number representing the (Cumulative) False positive rate with confidence level 4. In TeX, it will be denoted by $\lambda_4$.

... 


... 

z[1] A real number representing the lowest threshold of the (Gaussian) bi-normal assumption.
z[2] A real number representing the 2nd threshold of the (Gaussian) bi normal assumption.
z[3] A real number representing the 3rd threshold of the (Gaussian) bi normal assumption.
z[4] A real number representing the fourth threshold of the (Gaussian) bi-normal assumption.

a A real number defined by $m/v$, please contact the author’s paper for detail.
b A real number representing defined by $1/v$, please contact the author’s paper for detail.

A A positive real number between 0 and 1, representing AUC, i.e., the area under the alternative ROC curve.

1p__ The logarithmic likelihood of our model for your data.

--- Notations and symbols: Outputs of Multiple Reader and Multiple Modality case ---

w The lowest threshold of the Gaussian assumption (bi-normal assumption). so $w = z[1]$.

... 

mu The mean of the Latent Gaussian distribution for diseased images.
v The variance of the Latent Gaussian distribution for diseased images.
ppp[1,1,1] Hit rate with confidence level 1, modality 1, reader 1.
ppp[2,1,1] Hit rate with confidence level 2, modality 1, reader 1.
ppp[3,1,1] Hit rate with confidence level 3, modality 1, reader 1.

... 

l[1] (Cumulative) False positive rate with confidence level 1.
l[2] (Cumulative) False positive rate with confidence level 2.
l[3] (Cumulative) False positive rate with confidence level 3.
... 
\( d_1[1] \) This is defined by the difference \( 1[1] - 1[2] \).
\( d_1[2] \) This is defined by the difference \( 1[2] - 1[3] \).
\( d_1[3] \) This is defined by the difference \( 1[3] - 1[4] \).
...
\( z[1] \) The lowest threshold of the (Gaussian) bi-normal assumption.
\( z[2] \) The 2nd threshold of the (Gaussian) bi-normal assumption.
\( z[3] \) The 3rd threshold of the (Gaussian) bi-normal assumption.
\( z[4] \) The fourth threshold of the (Gaussian) bi-normal assumption.
\( aa \) This is defined by \( m/v \), please see the author’s paper for more detail.
\( bb \) This is defined by \( 1/v \), please see the author’s paper for more detail.
\( AA \) The area under alternative FROC curve associated to reader and modality.
\( A \) The area under alternative FROC curve associated to modality.
\( hyper_v \) Standard deviation of \( AA \) around \( A \).
\( 1p_{--} \) The logarithmic likelihood of our model for your data.

References

Bayesian Models for Free-response Receiver Operating Characteristic Analysis; Pre-print See vignettes

See Also

——— Before fitting: create a dataset

convertFromJafroc Convert from JAFROC format xlsx file to the author’s format
dataset_creator_new_version Create an \( \text{R} \) object which represent user data.
create_dataset Create an \( \text{R} \) object which represent user data.

——— Further sequential analysis: Plot curves Using the result of fitting a Bayesian FROC model, we can go sequential analysis.

drawCurves for drawing free response ROC curves.

——— Further sequential analysis: Validation of the Model

ppp Calculation of a p-value in the Bayesian paradigm.

——— \( \text{R} \) objects of example datasets from real world or fictitious:
dataList.Chakra.1 A list for an example dataset of a single reader and a single modality data.
The word Chakra in the dataset name means that it appears in the paper of Chakraborty.
dataList.Chakra.2 A list for an example dataset of a single reader and a single modality data.
The word Chakra in the dataset name means that it appears in the paper of Chakraborty.
dataList.Chakra.3 A list for an example dataset of a single reader and a single modality data.
The word Chakra in the dataset name means that it appears in the paper of Chakraborty.
dataList.Chakra.4 A list for an example dataset of a single reader and a single modality data.
The word Chakra in the dataset name means that it appears in the paper of Chakraborty.
**Examples**

```r
## Not run:
\dontrun{
#========================================================================================
# The 1-st example
#========================================================================================
#
#
# Making FROC Data and Fitting a Model to the data
#
# Notations
#
# h = hits = TP = True Positives
# f = False alarms = FP = False Positives
#
#========================================================================================
# 1) Build a data-set
```

- **dataList.high.ability**  A list for an example dataset of a single reader and a single modality data
- **dataList.low.ability**  A list for an example dataset of a single reader and a single modality data
- **dataList.Chakra.Web**  A list for an example dataset of multiple readers and multiple modalities data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.
- **data.hier.ficitious**  A list for an example dataset of multiple readers and multiple modalities data
- **dataList.High**  A list for an example dataset of a single reader and a single modality data whose AUC is high.
- **dataList.Low**  A list for an example dataset of a single reader and a single modality data whose AUC is low.
- **data.bad.fit**  A list for an example dataset of a single reader and a single modality data whose fitting is bad, that is chi square is very large. However the MCMC convergence criterion is satisfied with very high quality. Thus the good MCMC convergence does not mean the model is correct. So, to fit a model to this data, we should change the latent Gaussian and differential logarithmic Gaussian to more appropriate distributions for hit and false alarm rate. In theoretically perspective, there is no a a prior distribution for hit and false alarm rate. So, if we encounter not good fitting data, then we should change the model, and such change will occur in the latent distributions. The reason why the author saved this data is to show that our model is not unique nor good and gives a future research directions. To tell the truth the author is not interested the FROC theory. My background is mathematics, geometry, pure mathematics. So, I want to go back to my home ground. This program are made to show my skill for programming or my ability. But, now, I do not think to get job. I want to go back mathematics. Soon, my paper is published which is related Gromov Hausdorff topology. Of course, I will publish this package’s theory soon. Please wait.

The other datasets, the author like these datasets because name is very simple.
# For a single reader and a single modality case.

dat <- list(c=c(3,2,1), # Confidence level. Note that c is ignored.
h=c(97,32,31), # Number of hits for each confidence level
f=c(1,14,74), # Number of false alarms for each confidence level
NL=259, # Number of lesions
NI=57, # Number of images
C=3) # Number of confidence level

if (interactive()) { viewdata(dat)}

# where,
# c denotes confidence level, i.e., rating of reader.
# 3 = Definitely diseased,
# 2 = subtle... diseased
# 1 = very subtle
# h denotes number of hits (True Positives: TP) for each confidence level,
# f denotes number of false alarms (False Positives: FP) for each confidence level,
# NL denotes number of lesions,
# NI denotes number of images,

# For example, in the above example data,
# the number of hits with confidence level 3 is 97,
# the number of hits with confidence level 2 is 32,
# the number of hits with confidence level 1 is 31,
# the number of false alarms with confidence level 3 is 1,
# the number of false alarms with confidence level 2 is 14,
# the number of false alarms with confidence level 1 is 74,

#============================================================
# 2) Fit an FROC model to the above dataset.
#============================================================

fit <- fit_Bayesian_FROC(
  dat, # dataset
  ite = 1111, # To run in time <5s.
  cha = 1, # number of chains, it is better more large.
  summary = FALSE
)
The return value "fit" is an S4 object of class "stanfitExtended" which is inherited from the S4 class "stanfit".

# 3) Change the S4 class of fitted model object
# Change the S4 class from "stanfitExtended" to "stanfit" to apply other packages.
# The fitted model object of class "stanfit" is available for the package ggmcmc, rstan
# Thus, to use such package, we coerce the class into "stanfit" as follows:

# Changing the class from stanfitExtended to stanfit,
# we can apply other package's functions to the resulting object.

fit.stan <- methods::as(fit,"stanfit")

# Then, return value "fit.stan" is no longer an S4 object of class "stanfitExtended" but the S4 object of class "stanfit".

# 3.1) Apply the functions for the class stanfit

grDevices::dev.new();rstan::stan_hist(fit.stan, bins=33,pars = c("A"))
grDevices::dev.new();rstan::stan_hist(fit.stan, bins=22,pars = c("A"))
grDevices::dev.new();rstan::stan_hist(fit.stan, bins=11,pars = c("A"))
grDevices::dev.off()

# I am not sure why the above stan_hist also works for the new S4 class "stanfitExtended"

# Get pipe operator

'\+%\+' <- utils::getFromNamespace("\+%", "magrittr")

# Plot about MCMC samples of parameter name "A", representing AUC

# Trace-plot density for parameter "A"
grDevices::dev.new()
  ggmcmc::ggs(fit.stan) %>% ggmcmc::ggs_traceplot(family = "A")
grDevices::dev.off()
# Posterior density for parameter "A"
grDevices::dev.new()
  ggmcmc::ggs(fit.stan) %>% ggmcmc::ggs_density(family = "A")
grDevices::dev.off()
# Auto-correlation for parameter "A"
grDevices::dev.new()
  ggmcmc::ggs(fit.stan) %>% ggmcmc::ggs_autocorrelation(family = "A")
grDevices::dev.off()

# The author does not think the inherited class "stanfitExtended" is good,
# Since the size of object is very redundant and large,
# which caused by the fact that inherited class contains plot data for FROC curve.
# To show the difference of size for the fitted model object of class
# stanfitExtended and stanfit, we execute the following code;

size_of_return_value(fit) - size_of_return_value(methods::as(fit,"stanfit"))

#4) Using the S4 object fit, we can go further step, such as calculation of the
# Chisquare and the p value of the Bayesian version for testing the goodness of fit.
# I think p value has problems that it relies on the sample size with monotonicity.
# But it is well used, thus I hate but I implement the p value.

#========================================================================================
# REMARK
#========================================================================================
#
# Should not write the above data as follows:
#
# MANNER (A)  dat <- list(c=c(1,2,3),h=c(31,32,97),f=c(74,14,1),NL=259,NI=57,C=3)

# Even if user writes data in the above MANNER (A),
# the program interprets it as the following MANNER (B);

# MANNER (B)  dat <- list(c=c(3,2,1),h=c(31,32,97),f=c(74,14,1),NL=259,NI=57,C=3)
# Because the vector c is ignored in the program,
# and it is generated by the code rep(C:1) automatically in the internal of the function.
# So, we can omit the vector c from the list.

# This package is very rigid format, so please be sure that your format is
# exactly same to the data in this package.
# More precisely, the confidence level vector should be denoted rep(C:1) (Not rep(1:C)).
# Note that confidence level vector c should not be specified.
# If specified, will be ignored,
# since it is created by c <-c(rep(C:1)) in the program and
# do not refer from user input confidence level vector,
# where C is the highest number of confidence levels.

#========================================================================================
# The 2-nd example
#========================================================================================
#
# (1) First, we prepare the data from this package.

    dat <- BayesianFROC::dataList.Chakra.1

# (2) Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
# with data named "dat" and the author's Bayesian model.

    fit <- fit_Bayesian_FROC(dat,
        ite = 1111  #To run in time <5s.
    )

# Now, we get the stan's output, i.e., an S4 class object named "fit".
#
# << Minor Comments>>
# More precisely, this is an S4 object of some inherited class (named stanfitExtended)
# which is extended using stan's S4 class named "stanfit".

fit.stan <- methods::as(fit,"stanfit")
# Using the output "fit.stan",

# we can use the functions in the "rstan" package, for example, as follows;

grDevices::dev.new();
   rstan::stan_trace(fit.stan, pars = c("A"))  # stochastic process of a posterior estimate
   rstan::stan_hist(fit.stan, pars = c("A"))  # Histogram of a posterior estimate
   rstan::stan_rhat(fit.stan, pars = c("A"))  # Histogram of rhat for all parameters
   rstan::summary(fit.stan, pars = c("A"))   # summary of fit.stan by rstan

grDevices::dev.off()

#========================================================================================
# The 3-rd example
#========================================================================================

# Fit a model to a hand made data

# 1) Build the data for a single reader and a single modality case.

dat <- list(
    c=c(3,2,1),  # Confidence level, which is ignored.
    h=c(97,32,31),  # Number of hits for each confidence level
    f=c(1,14,74),  # Number of false alarms for each confidence level
    NL=259,  # Number of lesions
    NI=57,  # Number of images
    C=3)  # Number of confidence level

# where,
#  c denotes confidence level, , each components indicates that
#  3 = Definitely lesion,
#  2 = subtle,
#  1 = very subtle
#  That is the high number indicates the high confidence level.
#  h denotes number of hits
#  (True Positives: TP) for each confidence level,
#  f denotes number of false alarms
#  (False Positives: FP) for each confidence level,
#  NL denotes number of lesions,
#  NI denotes number of images,
2) Fit and draw FROC and AFROC curves.

```r
fit <- fit_Bayesian_FROC(dat, DrawCurve = TRUE)
```

# (( REMARK ))
# Changing the hits and false alarms denoted by h and f
# in the above dataset denoted by dat,
# user can fit a model to various datasets and draw corresponding FROC curves.
# Enjoy drawing the curves for various datasets in case of
# a single reader and a single modality data

# For Prior and Bayesian Update:

# Calculates a posterior mean and variance
# for each parameter

# Mean values of posterior samples are used as a point estimates, and
# Although the variance of posteriors receives less attention,
# but to make a prior, we will need the it.
# For, example, if we assume that model parameter m has prior distributed by
# Gaussian, then we have to know the mean and variance to characterize prior.

```r
e <- rstan::extract(fit)
```

# model parameter m and v is a number,
# indicating the mean and variance of signal distribution, respectively.

```r
stats::var(e$m)
mean(e$m)

stats::var(e$v)
mean(e$v)
```
# The model parameter \( z \) or \( dz \) is a vector, and thus we execute the following;

\[
\begin{align*}
  & z = ( z[1], z[2], z[3] ) \\
  & dz = ( z[2]-z[1], z[3]-z[2] )
\end{align*}
\]

# `Posterior mean of posterior MCMC samples for parameter \( z \) and \( dz \)

```r
apply(e$dz, 2, mean)
apply(e$z, 2, mean)
```

# `Posterior variance of posterior MCMC samples for parameter \( z \) and \( dz \)

```r
apply(e$dz, 2, var)
apply(e$z, 2, var)
apply(e$dl, 2, mean)
apply(e$l, 2, mean)
apply(e$p, 2, mean)
apply(e$p, 2, var)
```
# Revised 2019 Sept 6

### The 4-th example

## Only run examples in interactive R sessions
if (interactive()) {

# 1) Build the data interactively,

dataList <- create_dataset()

#Now, as a return value of create_dataset(), we get the FROC data (list) named dataList.

# 2) Fit an MRMC or ssrc FROC model.

fit <- fit_Bayesian_FROC(dataList)

}

### The 5-th example

## Comparison of the posterior probability for AUC

# In the following, we calculate the probability of the events that
# the AUC of some modality is greater than the AUC of another modality.

### Posterior Probability for some events of AUCs by using posterior MCMC samples

# This example shows how to use the stanfit (stanfit.Extended) object.
# Using stanfit object, we can extract posterior samples and using these samples,
# we can calculate the posterior probability of research questions.

fit <- fit_Bayesian_FROC(dataList.Chakra.Web.orderd, ite = 1111, summary = FALSE)
# For example, we shall show the code to compute the posterior probability of the ever
# that the AUC of modality 1 is larger than that of modality 2:

e <- extract(fit)

# This code means that the MCMC samples are retained in the object e for all parameters.
# For example, the AUC is extracted by the code e$A and it is a two dimensional array.
# The first component indicates the MCMC samples and
# the second component indicate the modality ID.

# For example, the code e$A[,1] means the vector of MCMC samples of the 1st modality.
# For example, the code e$A[,2] means the vector of MCMC samples of the 2nd modality.
# For example, the code e$A[,3] means the vector of MCMC samples of the 3rd modality.
# To calculate the posterior probability of the ever
# that the AUC of modality 1 is larger than that of modality 2,
# we execute the following R script:

mean(e$A[,1] > e$A[,2])

# Similarly, to compute the posterior probability that
# the AUC of modality 1 is larger than that of modality 3:

mean(e$A[,1] > e$A[,3])

# Similarly, to compute the posterior probability that
# the AUC of modality 1 is larger than that of modality 4:

mean(e$A[,1] > e$A[,4])

# Similarly, to compute the posterior probability that
# the AUC of modality 1 is larger than that of modality 5:

mean(e$A[,1] > e$A[,5])

# Similarly, to compute the posterior probability that
# the AUC of modality 1 is larger than that of modality 5 at least 0.01

mean(e$A[,1] > e$A[,5]+0.01)

# Similarly,

mean( e$A[,1] > e$A[,5] + 0.01 )
mean( e$A[,1] > e$A[,5] + 0.02 )
mean( e$A[,1] > e$A[,5] + 0.03 )
mean( e$A[,1] > e$A[,5] + 0.04 )
mean( e$A[,1] > e$A[,5] + 0.05 )
mean( e$A[,1] > e$A[,5] + 0.06 )
mean( e$A[,1] > e$A[,5] + 0.07 )
mean( e$A[,1] > e$A[,5] + 0.08 )

# Since any posterior distribution tends to the Dirac measure whose center is
# true parameter under the assumption that the model is correct in the sense that the
# true distribution is belongs to a family of models.
# Thus using this procedure, we will get
# the true parameter if any more large sample size we can take.

# Close the graphic device to avoid errors in R CMD check.

Close_all_graphic_devices()

#========================================================================================
# The 6-th Example for MRMC data
#========================================================================================

# To draw FROC curves for each modality and each reader, the author provides codes.
# First, we make a fitted object of class stanfitExtended as following manner.

fit <- fit_Bayesian_FROC( ite = 1111,
                           cha = 1,
                           summary = FALSE,
                           Null.Hypothesis = FALSE,
                           dataList = dd # This is a MRMC dataset. )

# Using this fitted model object called fit, we can draw FROC curves for the
# 1-st modality as following manner:
DrawCurves(
  # This is a fitted model object
  fit,
  # Here, the modality is specified
  modalityID = 1,
  # Reader is specified 1,2,3,4
  readerID = 1:4,
  # If TRUE, the new imaging device is created and curves are drawn in it.
  new.imaging.device = TRUE
)

# The next codes are quite same, except modality ID and new.imaging.device
# The code that "new.imaging.device = F" means that the curves are drawn using
# the previous imaging device to plot the 1-st and 2-nd modality curves draw in the same
# Plot plain. Drawing in different curves in same plain, we can compare the curve
# of modality. Of course, the interpretation of FROC curve is the ordinal ROC curve,
# that is,
# if curve is upper then the observer performance with its modality is more greater.
# So, please enjoy drawing curves.

DrawCurves(fit,modalityID = 2,readerID = 1:4, new.imaging.device = FALSE)
DrawCurves(fit,modalityID = 3,readerID = 1:4, new.imaging.device = FALSE)
DrawCurves(fit,modalityID = 4,readerID = 1:4, new.imaging.device = FALSE)
DrawCurves(fit,modalityID = 5,readerID = 1:4, new.imaging.device = FALSE)

Close_all_graphic_devices()

#========================================================================================
# The 7-th example NON-CONVERGENT CASE 2019 OCT.
#========================================================================================

# ff <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = ddd )

dat <- list(
  c=c(3,2,1),  #Confidence level
  h=c(73783933,15661264,12360003),  #Number of hits for each confidence level
  f=c(1738825,53666125,254965774),  #Number of false alarms for each confidence level
NL=100000000,  #Number of lesions
NI=200000000,  #Number of images
C=3)  #Number of confidence level

# From the examples of the function mu_truth_creator_for_many_readers_MRMC_data()
#========================================================================================
# Large number of readers cause non-convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=4,Q=6)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=4,Q=6)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 111, cha = 1, summary = TRUE, dataList = d )
plot_FPF_and_TPF_from_a_dataset(d)

#========================================================================================
# convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=2,Q=21)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=2,Q=21)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 200, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# non-convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=5,Q=6)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=5,Q=6)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 111, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# convergence
#========================================================================================
v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
d <- create_dataList_MRMC(mu.truth = m, v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# non-convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
d <- create_dataList_MRMC(mu.truth = m, v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 111, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# convergence A single modality and 11 readers
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=11)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=11)
d <- create_dataList_MRMC(mu.truth = m, v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 1111,
# cha = 1,
# summary = TRUE,
# dataList = d,
# see = 123455)
#
## f <- fit
# DrawCurves( summary = FALSE,
# modalityID = c(1:f@dataList$M),
# readerID = c(1:f@dataList$Q),
# StanS4class = f )
#
##
##
#========================================================================================
v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=17)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=17)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = d,see = 123455)
#
# f <-fit
# DrawCurves( summary = FALSE, modalityID = c(1:f@dataList$M),
# readerID = c(1:f@dataList$Q),f )
#
#
# DrawCurves( summary = FALSE, modalityID = 1,
# readerID = c(8,9),f )
#
## For readerID 8,9, this model is bad
#
Close_all_graphic_devices()

#========================================================================================
# convergence 37 readers, 1 modality
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC(see = 2345678, ite = 2000, cha = 1, summary = TRUE, dataList = d)
#
# # f <-fit
# DrawCurves( summary = FALSE, modalityID = c(1:f@dataList$M),
# readerID = c(1:f@dataList$Q),f )
#
# DrawCurves( summary = FALSE, modalityID = 1,
# readerID = c(8,9),f )
#
# In the following, consider two readers whose ID are 8 and 15, respectively.
# Obviously, one of them will have high performance than the other,
# however,
# Sometimes, the FROC curve dose not reflect it,
# Namely, one of the FROC curve is upper than the other
# even if the FPF and TPF are not.... WHY???


# DrawCurves( summary = FALSE, modalityID = 1,
#     readerID = c(8,15), f )
#
Close_all_graphic_devices()

Close_all_graphic_devices()
}

## End(Not run)

fit_GUI

---

**fit_GUI**

*Fit with GUI via Shiny*

---

**Description**

First, please execute, then user will understand what it is. This function is the one of the most important function in this package. I do not assume that the user is familiar with R script but FROC analysis. So, I made this function to provide the Graphical User Interface (GUI) for users. I hope it helps someone in the world.

**Usage**

```r
fit_GUI(display.mode = FALSE)
```

**Arguments**

- `display.mode` Logical, passing to `runApp`. Default is FALSE corresponding to "normal", and if TRUE, then "showcase" which shows code. The author made this, but it did not work or ignored, that is, showcase did not work. Why???

**Value**

None

**Examples**

```r
## Not run:
## Only run examples in interactive R sessions
if (interactive()) {
  #No need to consider the variables, it is sufficient in default values.
  #fit_GUI()
}### Only run examples in interactive R sessions

## End(Not run)"
```
fit_GUI_dashboard  

Fit with GUI via Shiny (Simple version)

Description

simple is vest

Usage

```r
fit_GUI_dashboard(
  DF = data.frame(h = c(97L, 32L, 31L), f = c(1L, 14L, 74L)),
  NL.max = 1111,
  NI.max = 1111,
  NL.initial = 259,
  MCMC.chains.max = 4
)
```

Arguments

- **DF**  
  A dataframe as an initial data to be fitted a model

- **NL.max**  
  max number of bins indicating the maximal number in which the number of lesions can move

- **NI.max**  
  max number of bins indicating the maximal number in which the number of images can move

- **NL.initial**  
  Natural number indicating the initial number of lesions, Default value =259. The reason why the author made this is in example code, the author made a very strange data in it, the default value = 259 is not satisfies the data format. That is in the example, total number of hits is greater than 259 and it is impossible. So, I have to change the default value.

- **MCMC.chains.max**  
  max number of bins indicating number of MCMC chains

Details

First, please execute, then user will understand what it is. This function is the one of the most important function in this package. I do not assume that the user is familiar with R script but FROC analysis. So, I made this function to provide the Graphical User Interface (GUI) for users. I hope it helps someone in the world.

Value

None

Author(s)

Issei Tsunoda
Examples

```r
## Not run:
## Only run examples in interactive R sessions
if (interactive()) {

#========================================================================================
# 1) Use the default User Interface
#========================================================================================

#No need to consider the variables, it is sufficient in default values.

#fit_GUI_dashboard()

#========================================================================================
# 2) Change the User Interface
#========================================================================================

# We can change the max imput of the number of lesions and the max of number of images
#
#fit_GUI_dashboard(NL.max = 2222,
#                  NI.max = 3333)

#========================================================================================
# 3) Change the Default value
#========================================================================================

# fit_GUI_dashboard(
# DF= data.frame( h=dataList.Chakra.4$h,
#                 f=dataList.Chakra.4$f
#                 )
# )

# Or equivalently,

# fit_GUI_dashboard(
# DF= data.frame(
# h = c(160, 25, 15, 7),
# f = c( 8, 16, 18, 13)
# )
```

fit_GUI_MRMCFit with GUI via Shiny in case of MRMC

Description

First, please execute, then user will understand what it is. This function is the one of the most important function in this package. I do not assume the user is familiar with R script but FROC analysis. So, I made this function to provide the Graphical User Interface (GUI) for users. I hope it helps someone in the world.

Usage

fit_GUI_MRMCF(M = 2, Q = 3, C = 4)
**Arguments**

- **M**  
  No. of modalities
- **Q**  
  No. of readers
- **C**  
  No. of confidence levels revised 2019 Nov. 21

**Value**

None

---

`fit_GUI_MRMC_new`  
*Fit an MRMC model to data with Shiny GUI*

**Description**

I love you.

**Usage**

`fit_GUI_MRMC_new(M = 2, Q = 3, C = 4)`

**Arguments**

- **M**  
  mo
- **Q**  
  re
- **C**  
  con

**Details**

I need you.

**Value**

ret
Description

A graphical user interface (GUI) to fit a model to data.

Usage

```r
fit_GUI_Shiny(
  DF = data.frame(h = c(97L, 32L, 31L), f = c(1L, 14L, 74L)),
  NL.max = 1111,
  NI.max = 1111,
  NL.initial = 259,
  MCMC.chains.max = 4
)
```

Arguments

- **DF**: A dataframe as an initial data to be fitted a model.
- **NL.max**: max number of bins indicating the maximal number in which the number of lesions can move.
- **NI.max**: max number of bins indicating the maximal number in which the number of images can move.
- **NL.initial**: Natural number indicating the initial number of lesions. Default value = 259. The reason why the author made this is in example code, the author made a very strange data in it, the default value = 259 is not satisfies the data format. That is in the example, total number of hits is greater than 259 and it is impossible. So, I have to change the default value.
- **MCMC.chains.max**: max number of bins indicating number of MCMC chains.

Details

First, please execute, then user will understand what it is. This function is the one of the most important function in this package. I do not assume that the user is familiar with R script but FROC analysis. So, I made this function to provide the Graphical User Interface (GUI) for users. The GUI is made by the shiny package. I hope it helps someone in the world.

Value

None

Author(s)

Issei Tsunoda
Examples

```r
## Only run examples in interactive R sessions
if (interactive()) {
  #========================================================================================
  # 1) Use the default User Interface
  #========================================================================================
  #No need to consider the variables, it is sufficient in default values.
  
  fit_GUI_Shiny()
  
  #========================================================================================
  # 2) Change the User Interface
  #========================================================================================
  
  # We can change the max input of the number of lesions and the max of number of images
  #
  fit_GUI_Shiny(NL.max = 2222,
                NI.max = 3333)
  
  #========================================================================================
  # 3) Change the Default value
  #========================================================================================
  
  fit_GUI_Shiny(
    DF = data.frame( h = dataList.Chakra.4$h,
                     f = dataList.Chakra.4$f )
  )
  
  # Or equivalently,
  
  fit_GUI_Shiny(
    DF = data.frame( h = c(160, 25, 15, 7),
                     f = c( 8, 16, 18, 13) )
  )
```
# 4) Change the user interface

```r
fit_GUI_Shiny(
  DF= data.frame(
    h = c(160, 25, 15, 7),
    f = c( 8, 16, 18, 13)
  ),
  NL.max = 1192,
  NI.max = 794,
  MCMC.chains.max = 6
)
```

# 5) CUI rather than GUI input

```r
# How to input data using CUI?
# This example gives an answer.
#
# CUI: Characteristic user interface

# Here, I show the very strange data, that is, the number of hits is all 33
# and replicated 10 times, that is,
# h is substituted by rep(33L,10) indicating 33 33 33 33 33 33 33 33 33 33
# f is also same as h.

fit_GUI_Shiny(NL.initial=555,
  DF =data.frame(
    h= as.integer(rep(33,10)),
    f= as.integer(rep(33,10))
  )
)
```
# The author made this example since, when I check my program,  
# such as whether the color used in polygon() is appropriate or not.  
# If user thinks that it is very hard to input hits and false alarms  
# by GUI manner, then use this characteristic like manner.

}### Only run examples in interactive R sessions

---

**fit_GUI_Shiny_MRMC**

*Fit with GUI via Shiny (in case of MRMC)*  

**Description**  
Fit a Bayesian model with GUI.  
Revised 2019 Nov.

**Usage**

```r
fit_GUI_Shiny_MRMC(
  DF = data.frame(m = as.integer(BayesianFROC::dd$m), q =
                  as.integer(BayesianFROC::dd$q), c = as.integer(BayesianFROC::dd$c), h =
                  as.integer(BayesianFROC::dd$h), f = as.integer(BayesianFROC::dd$f)),
  NL.max = 1111,
  NI.max = 1111,
  NL.initial = 142,
  NI.initial = 199,
  seed.initial.of.MCMC = 237410,
  MCMC.chains.max = 4
)
```

**Arguments**

- **DF** A dataframe, consisting of five vectors: reader ID, modality ID, confidence levels, hits, false alarms. 
- **NL.max** max number of bins indicating the maximal number in which the number of lesions can move 
- **NI.max** max number of bins indicating the maximal number in which the number of images can move 
- **NL.initial** Natural number indicating the initial number of lesions, Default value =142. 
- **NI.initial** Natural number indicating the initial number of images, Default value =199. 
- **seed.initial.of.MCMC** positive integers indicating the initial seed of MCMC sampling. Default is 1234. 
- **MCMC.chains.max** max number of bins indicating number of MCMC chains
Details

In what follows, we assume that our dataset has more than two readers or modalities, namely, our dataset is MRMC case. The term *imaging modality*, we mean a set of imaging methods such as MRI, CT, PET, etc.

Revised 2019 Nov 25. Revised 2020 Jan

Value

None

Examples

```r
## Not run:

```r
# Only run examples in interactive R sessions
if (interactive()) {
  #========================================================================================
  # 1) Use the default User Interface
  #========================================================================================
  #No need to consider the variables, it is sufficient in default values.

  fit_GUI_Shiny()

  #========================================================================================
  # 2) From existing dataset, named ddddd or ddddd or ddd
  #========================================================================================

  fit_GUI_Shiny_MRMC(DF=extract_data_frame_from_dataList_MRMC(dddddd))
  fit_GUI_Shiny_MRMC(DF=extract_data_frame_from_dataList_MRMC(ddddd))
  fit_GUI_Shiny_MRMC(DF=extract_data_frame_from_dataList_MRMC(ddd))

  #========================================================================================
  # 2) data of 11 readers and a single modality
  #========================================================================================

  d <- dataset_creator_for_many_Readers(1,11)
```
fit_GUI_Shiny_MRMCDfit(GUI_Shiny_MRMCD(DF=extract_data_frame_from_dataList_MRMCD(d)),
    seed.initial.of.MCMC = 2345678,
    NL.initial = d$NL,
    NI.initial = d$NI)

#========================================================================================
# 2) From existing dataset, named dddd
#========================================================================================

fit_GUI_Shiny_MRMCD(DF=extract_data_frame_from_dataList_MRMCD(dddd))

# This dataset named dddd is a dataset consisting of
# only a single reader and multiple modality.
# Such a single reader and multiple modality case had error caused
# by some reduction of array to vector.
# So, the program was fixed so that such special case is also available
# 2020 Feb 24

# To reflect the information of the number of lesions and images,
# use the following.
fit_GUI_simple_from_apppp_file

**Fit with GUI via Shiny**

**Description**

First, please execute, then user will understand what it is. This function is the one of the most important function in this package. I do not assume that the user is familiar with R script but FROC analysis. So, I made this function to provide the Graphical User Interface (GUI) for users. I hope it helps someone in the world.
Usage

fit_GUI_simple_from_apppp_file(display.mode = FALSE)

Arguments

display.mode Logical, passing to runApp. Default is FALSE corresponding to "normal", and if TRUE, then "showcase" which shows code. The author made this, but it did not work or ignored, that is, showcase did not work. Why???

Value

None

Author(s)

Issei Tsunoda

Examples

## Not run:
## Only run examples in interactive R sessions
if (interactive()) {
    #No need to consider the variables, it is sufficient in default values.
    #fit_GUI_simple_from_apppp_file()

}### Only run examples in interactive R sessions

## End(Not run)####

---

fit_MRMCM Fit and Draw the FROC models (curves)

Description

Fit and Draw the FROC models (curves).

Usage

fit_MRMCM(
    dataList,
    DrawCurve = FALSE,
    verbose = TRUE,
    print_CI_of_AUC = TRUE,
    PreciseLogLikelihood = FALSE,
    summary = TRUE,
)
fit_MRMC

dataList.Name = "",
prior = 1,
ModifiedPoisson = TRUE,
mesh.for.drawing.curve = 10000,
significantLevel = 0.7,
cha = 1,
war = floor(ite/5),
ite = 10000,
dig = 3,
see = 1234569,
Null.Hypothesis = FALSE,
prototype = FALSE,
model_reparametrized = FALSE,
Model_MRMC_non_hierarchical = TRUE,
zz = 1,
... )

Arguments

dataList A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data. For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
h = c(41,22,14,8,1),# number of hits for each confidence level
f = c(1,2,5,11,13),# number of false alarms for each confidence level
NL = 124,# number of lesions (signals)
NI = 63,# number of images (trials)
C = 5) # number of confidence levels the author thinks it can be calculated as the length of h or f...? ha, why I included this. ha... should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).

To make this R object dataList representing FROC data, this package provides three functions:

cvtColorJafroc() If data is a JAFROC xlsx formulation.
dataset_creator_new_version() Enter TP and FP data by table.
create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

———————————————————————————-
A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

f Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

h Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

NL A positive integer, representing Number of Lesions.

NI A positive integer, representing Number of Images.

C A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored , since it is created by c <-c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

**data Format:**

A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP

*hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader
think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(rep(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c \leftarrow c(rep(C:1)) \) via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an \( \mathbb{R} \) list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

- \( C \): A positive integer, representing the highest number of confidence level, this is a scalar.
- \( M \): A positive integer vector, representing the number of modalities.
- \( Q \): A positive integer, representing the number of readers.
- \( m \): A vector of positive integers, representing the modality ID vector.
- \( q \): A vector of positive integers, representing the reader ID vector.
- \( c \): A vector of positive integers, representing the confidence level. This vector must be made by \( rep(rep(C:1), M*Q) \).
- \( h \): A vector of non-negative integers, representing the number of hits.
- \( f \): A vector of non-negative integers, representing the number of false alarms.
- \( NL \): A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above \( \mathbb{R} \) object. However, each confidence level vector is not included in the data, because it is created automatically from \( C \). To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

*Multiple readers and multiple modalities (i.e., MRMC)*

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

---

*false alarms = False Positives = FP*

*hits = True Positives = TP*

**DrawCurve**

Logical: TRUE or FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.

**verbose**

A logical, if TRUE, then the redundant summary is printed in R console. If FALSE, it suppresses output from this function.

**print_CI_of_AUC**

Logical, if TRUE then Credible intervals of AUCs for each modality are plotted.

**PreciseLogLikelihood**

Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE (default), then Stan calculates the precise log likelihood with target formulation. If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

**summary**

Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**dataList.Name**

This is not for user, but the author for this package development.

**prior**

positive integer, to select the prior

**ModifiedPoisson**

Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly,

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If \( \text{ModifiedPoisson} = \text{TRUE} \), then the \textit{False Positive Fraction (FPF)} is defined as follows (\( F_c \) denotes the number of false alarms with confidence level \( c \))

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \) (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be regressed as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \))
or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE).

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

mesh.for.drawing.curve

A positive large integer, indicating number of dots drawing the curves, Default = 10000.

significantLevel

This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

cha

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.

war

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5) = 10000/5 = 2000.

ite

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

dig

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named digits. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

see

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named seed. A positive integer representing seed used in stan, Default = 1234567.

Null.Hypothesis

Logical, that is TRUE or FALSE. If Null.or.Alternative.Hypothesis = FALSE(default), then fit the alternative model to dataList (for details of models, see vignettes). If Null.or.Alternative.Hypothesis = TRUE, then fit the null model to dataList.(for details of models, see vignettes). Note that the null model is
constructed under the null hypothesis that all modality are same observer performance ability. The alternative model is made under the assumption that all modality are not same. The reason why author creates this parameter is to test the null hypothesis by the Bayes factor. But the result of test is not desired one for me. Thus the test is under construction.

A logical, if TRUE then the model is no longer a generative model. Namely, in generally speaking, a dataset drawn from the model cannot satisfy the condition that the sum of the numbers of hits over all confidence levels is bounded from the above by the number of lesions, namely,

\[ \Sigma c H_c \leq N_L \]

However, this model (TRUE) is good in the sense that it admits various initial values of MCMC sampling.

if FALSE, then the model is precisely statistical model in the sense that any dataset drawn from the model satisfies that the sum of the number of hits is not greater than the number of lesions, namely,

\[ \Sigma c H_c \leq N_L. \]

This model is theoretically perfect. However, in the practically, the calculation will generates some undesired results which caused by the so-called floo .... I forget English :’-D. The flood point??! I forvvreueeeeeeedd!! Ha. So, prior synthesizes very small hit rates such as 0.000000000000001234 and it cause the non accurate calculation such as 0.000000000000000012345= 0.0012 which becomes hit rate and thus OH No!. Then it synthesizes Bernoulli success rate which is not less than 1 !! To avoid this, the author should develop the theory of prior to avoid this very small numbers, however the author has idea but now it does not success.

If prototype = TRUE, then the model for hits is the following:

\[ H_5 \sim \text{Binomial}(p_5, N_L) \]
\[ H_4 \sim \text{Binomial}(p_4, N_L) \]
\[ H_3 \sim \text{Binomial}(p_3, N_L) \]
\[ H_2 \sim \text{Binomial}(p_2, N_L) \]
\[ H_1 \sim \text{Binomial}(p_1, N_L) \]

On the other hand, if prototype = FALSE, then the model for hits is the following:

\[ H_5 \sim \text{Binomial}(p_5, N_L) \]
\[ H_4 \sim \text{Binomial}(\frac{p_4}{1 - p_5}, N_L - H_5) \]
\[ H_3 \sim \text{Binomial}(\frac{p_3}{1 - p_5 - p_4}, N_L - H_5 - H_4) \]
Each number of lesions is adjusted so that the sum of hits $\sum c H_c$ is less than the number of lesions (signals, targets) $N_L$. And hence the model in case of prototype = FALSE is a generative model in the sense that it can replicate datasets of FROC arises. Note that the adjustment of the number of lesions in the above manner leads us the adjustment of hit rates. The reason why we use the hit rates such as $\frac{p_1}{1 - p_5 - p_4 - p_3}$ instead of $p_c$ is that it ensures the equality $E[H_c/N_L] = p_c$. This equality is very important. To establish Bayesian FROC theory so that it is compatible to the classical FROC theory, we need the following two equations,

\begin{align*}
E[H_c/N_L] &= p_c, \\
E[F_c/N_X] &= q_c,
\end{align*}

where $E$ denotes the expectation and $N_X$ is the number of lesion or the number of images and $q_c$ is a false alarm rate, namely, $F_c \sim \text{Poisson}(q_c N_X)$.

Using the above two equations, we can establish the alternative Bayesian FROC theory preserving classical notions and formulas. For the details, please see the author’s pre print:

Bayesian Models for „, for?? I forget my paper title .... :'D. What the hell!? I forget,... My health is so bad to forget , .... I forget.

The author did not notice that the prototype is not a generative model. And hence the author revised the model so that the model is exactly generative model.

But the reason why the author remains the prototype model(prototype = TRUE) is that the convergence of MCMC sampling in case of MRMC is not good in the current model (prototype = FALSE). Because it uses fractions $\frac{p_1}{1 - p_5 - p_4 - p_3}$ and which is very dangerous to numerical perspective. For example, if $p_1$ is very small, then the numerator and denominator of $\frac{p_1}{1 - p_5 - p_4 - p_3}$ is very small. Both of them is like 0.00000000000000123... and such small number causes the non accurate results. So, sometimes, it occurs that $\frac{p_1}{1 - p_5 - p_4 - p_3} > 1$ which never occur in the theoretical perspective but unfortunately, in numerically occurs.

SO, now, the author try to avoid such phenomenon by using priors but it now does not success.

Here of course we interpret the terms such as $N_L - H_5 - H_4 - H_3$ as the remained targets after reader get hits. The author thinks it is another manner to do so like $N_L - H_1 - H_2 - H_3$, but it does not be employed. Since the author thinks that the reader will assign his suspicious lesion location from high confidence level and in this view point the author thinks it should be considered that targets are found from the highest confidence suspicious location.
Model_MRMC_non_hierarchical
A logical. If TRUE, then the model of multiple readers and multiple modalities consists of no hyper parameters. The reason why the author made this parameter is that the hyper parameter make the MCMC posterior samples be unstable. And also, my hierarachical model is not so good in theoretical perspective. Thus, I made this. The Default is TRUE.

zz
A real number specifying one of the parameter of prior

Additional arguments

fit_MRMC_versionTWO  Fit and Draw the FROC models (curves) version2.

Description

Fit and Draw the FROC models (curves). This model is aimed to draw a free-response ROC curves for multiple readers, that is, resulting FROC curve is one for multiple readers and reflects their hits and false alarms.

Usage

fit_MRMC_versionTWO(
    dataList,
    DrawFROCcurve = TRUE,
    DrawCFPCTP = TRUE,
    version = 2,
    mesh.for.drawing.curve = 10000,
    significantLevel = 0.7,
    cha = 1,
    war = floor(ite/5),
    ite = 10000,
    dig = 5,
    see = 1234569
)

Arguments

dataList A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data. For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
    h = c(41,22,14,8,1),# number of hits for each confidence level
f = c(1,2,5,11,13), # number of false alarms for each confidence level

NL = 124, # number of lesions (signals)
NI = 63, # number of images (trials)
C = 5) # number of confidence, ... the author thinks it can be calculated as the length of h or f ...? ha, why I included this. ha ... should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).

To make this R object dataList representing FROC data, this package provides three functions:

- convertFromJafroc() If data is a JAFROC xlsx formulation.
- dataset_creator_new_version() Enter TP and FP data by table.
- create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

- f Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- h Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- NL A positive integer, representing Number of Lesions.
- NI A positive integer, representing Number of Images.
- C A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored , since it is created by c <-c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

data Format:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

---

*false alarms* = False Positives = FP  
*hits* = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector $c$ should not be specified. If specified, will be ignored, since it is created by $c \leftarrow c(rep(C:1))$ automatically in the program and do not refer from user input data even if it is specified explicitly, where $C$ is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector $c \leftarrow c(rep(C:1))$ via a table which can be displayed by the function `viewdata()`.

---

Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an R list object representing FROC data must contain components $m,q,c,h,f,NL,C,M,Q$.

- $C$ A positive integer, representing the highest number of confidence level, this is a scalar.
- $M$ A positive integer vector, representing the number of modalities.
- $Q$ A positive integer, representing the number of readers.
- $m$ A vector of positive integers, representing the modality ID vector.
- $q$ A vector of positive integers, representing the reader ID vector.
- $c$ A vector of positive integers, representing the confidence level. This vector must be made by $rep(rep(C:1),M*Q)$
- $h$ A vector of non-negative integers, representing the number of hits.
- $f$ A vector of non-negative integers, representing the number of false alarms.
- $NL$ A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by $C$) are included in the above R object. However, each confidence level vector is not included in the
data, because it is created automatically from C. To confirm false positives and
hits are correctly ordered with respect to the automatically generated confidence
vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019
Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms* = False Positives = FP

*hits* = True Positives = TP

**version**

A positive large integer, indicating number of dots drawing the curves, Default
= 10000.

**mesh.for.drawing.curve**

To be passed to the function `rstan::sampling()` in `rstan`. An argument of
`rstan::sampling()` in which it is named `chains`. A positive integer representing
the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.
war  To be passed to the function rstan::sampling in rstan. An argument of rstan::sampling in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000.

ite  To be passed to the function rstan::sampling in rstan. An argument of rstan::sampling in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

dig  To be passed to the function rstan::sampling in rstan. An argument of rstan::sampling in which it is named ...?? . A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

see  To be passed to the function rstan::sampling in rstan. An argument of rstan::sampling in which it is named seed. A positive integer representing seed used in stan, Default = 1234567.

Author(s)
Issei Tsunoda

See Also
Example data:
BayesianFROC::dataList.one.modality
This dataset is a single modality dataset with multiple readers.

Examples
## Not run:

### The first example

First, we prepare the data from this package.

dat <- BayesianFROC::dataList.one.modality

Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
with data named "dat" and the author's Bayesian model.

fit <- fit_MRMC_versionTWO(dat,see = 12,ite=2222)

It needs a lot of memory and so, in this example we take the small iteration, i.e., ite =2222. However if user execute this, then the ite =30000 is recommended for getting reliable estimates.
#Note that we change the seed from default to 12 to get a convergence model.
#If users encounter the convergence issues,
#then please consider changing the seed like this example.

#The resulting FROC curve means the summarizing curve over all readers

#(1) First, we prepare the data from this package.

```r
dat <- BayesianFROC::dataList.Chakra.Web
```

#(2) Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
#with data named "dat" and the author's Bayesian model.

```r
fit <- fit_MRMC_versionTWO(dataList.Chakra.Web ,ite=2222)
```

#The resulting FROC curve means the summarizing curve over all readers

# It needs a lot of memory and so, in this example we take the small iteration,
# i.e., ite =2222. However if user execute this, then the ite =30000 is recommended
# for getting reliable estimates.

```
if (!grDevices::dev.cur()>=2) {
  for (i in 1:grDevices::dev.cur()-1) {message("The",i,"-th graphic device is omitted.")
    grDevices::dev.off()
  }
}
```

## End(Not run)#
dontrun

---

**fit_Null_hypothesis_model_to_**

*Fit the null model*
Description

Fit the null model, representing the null hypothesis that all modalities are same.

Usage

```r
fit_Null_hypothesis_model_to_(
  dataList,
  DrawCurve = FALSE,
  PreciseLogLikelihood = FALSE,
  dataList.Name = "",
  ModifiedPoisson = FALSE,
  verbose = TRUE,
  summary = TRUE,
  mesh.for.drawing.curve = 10000,
  significantLevel = 0.7,
  cha = 1,
  war = floor(ite/5),
  ite = 10000,
  dig = 3,
  see = 1234569,
  ...
)
```

Arguments

- **dataList**: A list, to be fitted a model. For example, in case of a single reader and a single modality, it consists of \( f, h, NL, NI, C \). The detail of these dataset, see the example data-sets. Note that the maximal number of confidence level, denoted by \( C \), are included, however, should not include its each confidence level in `dataList`.
- **DrawCurve**: Logical, that is TRUE or FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set `DrawCurve` = TRUE, if not then `DrawCurve` = FALSE. The reason why the author make this variable `DrawCurve` is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.
- **PreciseLogLikelihood**: Logical, that is TRUE or FALSE. If `PreciseLogLikelihood` = TRUE (default), then Stan calculates the precise log likelihood with target formulation. If `PreciseLogLikelihood` = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.
- **dataList.Name**: This is not for user, but the author for this package development.
- **ModifiedPoisson**: Logical, that is TRUE or FALSE. If `ModifiedPoisson` = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]
where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the *False Positive Fraction (FPF) per image*.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion ($\text{ModifiedPoisson} = \text{FALSE}$) or as the expected pairs of FPF per image and TPF per lesion ($\text{ModifiedPoisson} = \text{TRUE}$).

If $\text{ModifiedPoisson} = \text{TRUE}$, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if $\text{ModifiedPoisson} = \text{FALSE}$, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether $\text{ModifiedPoisson} = \text{TRUE}$ or $\text{FALSE}$. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether $\text{ModifiedPoisson} = \text{TRUE}$ or $\text{FALSE}$. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

**verbose**

A logical, if TRUE, then the redundant summary is printed in R console. If FALSE, it suppresses output from this function.

**summary**

Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**mesh.for.drawing.curve**

A positive large integer, indicating number of dots drawing the curves, Default = 10000.

**significantLevel**

This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

**cha**

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.

**war**

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000.

**ite**

To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
**Description**

Build a fitted model object in case of single reader and single modality data `dataList`. FPF is per image.

**Usage**

```r
fit_srsc(
  dataList, prior = -1,
  new.imaging.device = TRUE,
  dataList.Name = "",
  ModifiedPoisson = FALSE,
  model_reparametrized = FALSE,
  verbose = TRUE,
  DrawCurve = TRUE,
  PreciseLogLikelihood = TRUE,
  Drawcol = TRUE,
  make.csv.file.to.draw.curve = FALSE,
  mesh.for.drawing.curve = 10000,
  summary = TRUE,
  DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE,
  cha = 4,
  ite = 3000,
  dig = 5,
  war = floor(ite/5),
  see = 1234,
  prototype = FALSE,
  zz = 2.55,
  ...
)
```
**Arguments**

- **dataList**: A list, to be fitted a model. For example, in case of a single reader and a single modality, it consists of \(f, h, NL, NI, C\). The detail of these dataset, see the example data-sets. Note that the maximal number of confidence level, denoted by \(C\), are included, however, should not include its each confidence level in dataList.

- **prior**: positive integer, to select the prior

- **new.imaging.device**: Logical: TRUE or FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

- **dataList.Name**: This is not for user, but the author for this package development.

- **ModifiedPoisson**: Logical, that is TRUE or FALSE.

  If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per lesion*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF *per lesion*.

  Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated *per image*, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF *per image*.

  For more details, see the author’s paper in which I explained *per image* and *per lesion*. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

  If ModifiedPoisson = TRUE, then the *False Positive Fraction (FPF)* is defined as follows (\(F_c\) denotes the number of false alarms with confidence level \(c\))

  \[
  \frac{F_1 + F_2 + F_3 + F_4 + F_5}{NL},
  \]

  \[
  \frac{F_2 + F_3 + F_4 + F_5}{NL},
  \]

  \[
  \frac{F_3 + F_4 + F_5}{NL},
  \]

  \[
  \frac{F_4 + F_5}{NL},
  \]

  \[
  \frac{F_5}{NL},
  \]

  where \(NL\) is a number of lesions (signal). To emphasize its denominator \(NL\), we also call it the *False Positive Fraction (FPF) per lesion*.

  On the other hand,
if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[ \frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I}, \]

\[ \frac{F_2 + F_3 + F_4 + F_5}{N_I}, \]

\[ \frac{F_3 + F_4 + F_5}{N_I}, \]

\[ \frac{F_4 + F_5}{N_I}, \]

\[ \frac{F_5}{N_I}, \]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = FALSE)

or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE)

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

model_reparametrized

A logical, if TRUE, then a model under construction is used.

verbose

A logical, if TRUE, then the redundant summary is printed in R console. If FALSE, it suppresses output from this function.
DrawCurve Logical: TRUE or FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve = TRUE, if not then DrawCurve = FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.

PreciseLogLikelihood

Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE (default), then Stan calculates the precise log likelihood with target formulation. If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

Drawcol Logical: TRUE or FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The Default value is TRUE.

make.csv.file.to.draw.curve Logical: TRUE or FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.

mesh.for.drawing.curve

A positive large integer, indicating number of dots drawing the curves, Default = 10000.

summary Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

DrawFROCcurve Logical: TRUE or FALSE. Whether the FROC curve is to be drawn.

DrawAFROCcurve Logical: TRUE or FALSE. Whether the AFROC curve is to be drawn.

DrawCFPCTP Logical: TRUE or FALSE. Whether the CFP and CTP points are to be drawn. CFP: Cumulative false positive per lesion (or image) which is also called False Positive Fraction (FPF). CTP Cumulative True Positive per lesion which is also called True Positive Fraction (TPF).

cha To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.

ite To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

dig To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

war To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000,
To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `seed`. A positive integer representing seed used in `stan`, Default = 1234567.

A logical, if `TRUE` then the model is no longer a generative model. Namely, in generally speaking, a dataset drawn from the model cannot satisfy the condition that the sum of the numbers of hits over all confidence levels is bounded from the above by the number of lesions, namely,

\[ \sum_c H_c \leq N_L \]

However, this model (TRUE) is good in the sense that it admits various initial values of MCMC sampling.

if `FALSE`, then the model is precisely statistical model in the sense that any dataset drawn from the model satisfies that the sum of the number of hits is not greater than the number of lesions, namely,

\[ \sum_c H_c \leq N_L. \]

This model is theoretically perfect. However, in the practically, the calculation will generates some undesired results which caused by the so-called floo .... I forget English ‘-D. The flood point??!! I forgеееееееееееееееее!! Ha. So, prior synthesizes very small hit rates such as 0.0000000000000001234 and it cause the non accurate calculation such as 0.00000000000012345/0.00012345 = 0.0012 which becomes hit rate and thus OH No!. Then it synthesizes Bernoulli success rate which is not less than 1 !! To avoid this, the author should develop the theory of prior to avoid this very small numbers, however the author has idea but now it does not success.

If `prototype = TRUE`, then the model for hits is the following:

\[
H_5 \sim \text{Binomial}(p_5, N_L) \\
H_4 \sim \text{Binomial}(p_4, N_L - H_5) \\
H_3 \sim \text{Binomial}(p_3, N_L - H_5 - H_4) \\
H_2 \sim \text{Binomial}(p_2, N_L - H_5 - H_4 - H_3)
\]

On the other hand, if `prototype = FALSE`, then the model for hits is the following:

\[
H_5 \sim \text{Binomial}(p_5, N_L) \\
H_4 \sim \text{Binomial}\left(\frac{p_4}{1 - p_5}, N_L - H_5\right) \\
H_3 \sim \text{Binomial}\left(\frac{p_3}{1 - p_5 - p_4}, N_L - H_5 - H_4\right) \\
H_2 \sim \text{Binomial}\left(\frac{p_2}{1 - p_5 - p_4 - p_3}, N_L - H_5 - H_4 - H_3\right)
\]
Each number of lesions is adjusted so that the sum of hits $\Sigma c H_c$ is less than the number of lesions (signals, targets) $N_L$. And hence the model in case of $\text{prototype} = \text{FALSE}$ is a generative model in the sense that it can replicate datasets of FROC arises. Note that the adjustment of the number of lesions in the above manner leads us the adjustment of hit rates. The reason why we use the hit rates such as $\frac{p_1}{1-p_5-p_4-p_3-p_2}$ instead of $p_c$ is that it ensures the equality $E[H_c/N_L] = p_c$. This equality is very important. To establish Bayesian FROC theory so that it is compatible to the classical FROC theory, we need the following two equations,

$$E[H_c/N_L] = p_c, \quad E[F_c/N_X] = q_c,$$

where $E$ denotes the expectation and $N_X$ is the number of lesion or the number of images and $q_c$ is a false alarm rate, namely, $F_c \sim \text{Poisson}(q_c N_X)$. Using the above two equations, we can establish the alternative Bayesian FROC theory preserving classical notions and formulas. For the details, please see the author’s pre print:

Bayesian Models for ... for?? I forget my paper title .... :'-D. What the hell!? I forget.... My health is so bad to forget , .... I forget.

The author did not notice that the prototype is not a generative model. And hence the author revised the model so that the model is exactly generative model.

But the reason why the author remains the prototype model($\text{prototype} = \text{TRUE}$) is that the convergence of MCMC sampling in case of MRMC is not good in the current model ($\text{prototype} = \text{FALSE}$). Because it uses fractions $\frac{p_1}{1-p_5-p_4-p_3-p_2}$ and which is very dangerous to numerical perspective. For example, if $p_1$ is very small, then the numerator and denominator of $\frac{p_1}{1-p_5-p_4-p_3-p_2}$ is very small. Both of them is like 0.000000000000000123.... and such small number causes the non accurate results. So, sometimes, it occurs that $\frac{p_1}{1-p_5-p_4-p_3-p_2} > 1$ which never occur in the theoretical perspective but unfortunately, in numerically occurs.

SO, now, the author try to avoid such phenomenon by using priors but it now does not success.

Here of course we interpret the terms such as $N_L - H_5 - H_4 - H_3$ as the remained targets after reader get hits. The author thinks it is another manner to do so like $N_L - H_1 - H_2 - H_3$, but it does not be employed. Since the author thinks that the reader will assign his suspicious lesion location from high confidence level and in this view point the author thinks it should be considered that targets are found from the highest confidence suspicious location.

zz

A real number specifying one of the parameter of prior

Additional arguments
Value

An S4 object of class stanfitExtended, which is an inherited S4 class from stanfit. To change the S4 class, use

Examples

```r
## Not run:
# First, prepare the example data from this package.

dat <- get(data("dataList.Chakra.1"))

# Second, fit a model to data named "dat"

fit <- fit_srsc(dat)

# Close the graphic device to avoid errors in R CMD check.

Close_all_graphic_devices()

## End(Not run)
```

flatnames from rstan package

Description

from rstan package

Usage

flatnames(names, dims, col_major = FALSE)
Arguments

names       A vector of characters
 dims      A positive integer
col_major   A logical

Value

A vector of characters

Author(s)

Some Stan developer, I am not sure,..., who?

Examples

flatnames(c("a","b"),3)

# [1] "a[1]" "a[2]" "a[3]" "b[1]" "b[2]" "b[3]"

flat_one_par       Makes array names

Description

Makes array names

Usage

flat_one_par(n, d, col_major = FALSE)

Arguments

n           A character, n is an abbreviation of name
d          A vector of integers, to be passed to seq_array_ind()
col_major    A logical, to be passed to seq_array_ind()

Value

a vector of characters

Author(s)

Some Stan developer, I am not sure,..., who?
Examples

```r
a <- flat_one_par("a", 1:3)
# > a
# [1] "a[1,1,1]" "a[1,1,2]" "a[1,1,3]" "a[1,2,1]" "a[1,2,2]" "a[1,2,3]"
```

---

**foo**  

without double quote

---

Description

wait

Usage

```r
foo(X)
```

Arguments

- **X** sequence of

---

**foo**  

taboo or

---

Description

wait

Usage

```r
fooo()
```
foo_of_a_List_of_Arrays

Variance of a List of Arrays

Description

Then the function calculates the variance over all list for each array component.

Usage

foo_of_a_List_of_Arrays(x, name.of.function)

Arguments

x

A List of Arrays. The dimension of array is fixed for all list component.

name.of.function

This is an operator, such as mean, var, sum... Note that user no need to surround the input by "". For example, mean instead of "mean".

Details

Of course variance can change to sum or mean or any other functions whose entry is a vector. One can find this function in the Stack over flow, since I ask there, and thus the example given in here can also find also there. In my hierarchical Bayesian Model, the estimates has the format arrays. For example the hit rate are array whose subscript is confidence level, modality, and reader. So, when one desire to validate the estimates, it needs to calculate such variance of arrays. When I validate the estimates, I used the function.

Value

An array being reduced form use input list of array via user input operator such as mean, var, sum,...

Examples

#Suppose that x is the following list of arrays:

```
a <- array(1,c(2,3,4));
b <- array(2,c(2,3,4));
c <- array(3,c(2,3,4));
d <- array(4,c(2,3,4));
x <- list(a=a,b=b,c=c,d=d)
```

foo_of_a_List_of_Arrays(x,sum)
foo_of_a_List_of_Arrays(x,mean)
foo_of_a_List_of_Arrays(x,stats::var)
# Note that the component of list can be vectors with fixed same length.

```r
y <- list(c(1,2,3),
          c(11,22,33),
          c(1111,2222,3333))

a <- foo_of_a_List_of_Arrays(y,sum)
```

---

### FROC_curve

**FROC curve as an embedding map**

### Description

FROC curve as an embedding map

### Usage

```r
FROC_curve(x)
```

### Arguments

- `x` A real number moves in domain of FROC curve

### Value

none

### Examples

```r
# I love you!
```
from_array_to_vector | Transform from an array to a vector

Description
Transform a vector into an array

Usage
from_array_to_vector(Three.dim.array)

Arguments
Three.dim.array
Three dimensional array, such as the number of hits for each confidence level, modality and reader. Or false alarms. Since the author construct the substituting data list as one dimensional (one index) array, it needs to reconstruct to the three indexed array from one dimensional array whose subscript is [confidence level, modality, reader] or vice versa.

Details
In stan files of this package, the number of hits, false alarms and hit rates in binomial assumption for MRMC case are written with the three indexed array format. Three index indicates confidence levels, modality ID, reader ID. However, hit data passed to the function BayesianFROC::fit_Bayesian_FROC() are written with the vector. So, in order to connect these different format, (i.e. vector and array,) the author made this function.

Value
A vector, transformed from three dimensional array.

Examples
```r
## Not run:
#========================================================================================
# Practical example
#========================================================================================

h.array.etc <- hits_from_thresholds()
h.array.etc$h
h.vector <- from_array_to_vector(h.array.etc$h)
h.vector

#========================================================================================
# Educational example 1
#========================================================================================
```
### Description

This function is under construction. I validate only the example of this function. For MRMC case, I have to write or modify code. 2019 Sept 6

### Usage

```r
get_posterior_variance(StanS4class, name.of.parameter)
```

### Arguments

- **StanS4class**: An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and etc.

- **name.of.parameter**: An parameter name (given as a character string, should not surround by ""). The name of parameter which user want to extract. Parameters are contained in the parameter block of each Stan file in the path: `inst/extdata`.

### Value

Variance or posterior parameters, if it is an array, then return is also an array.
Examples

```r
## Not run:
dontrun{

fit <- fit_Bayesian_FROC(BayesianFROC::dd,ite = 1111)

e <- rstan::extract(fit)

# Check the reotrun value is the desired one.

apply(e$z, 2, var) == get_posterior_variance(fit,z)
apply(e$mu, c(2,3), var) == get_posterior_variance(fit,mu)
apply(e$v, c(2,3), var) == get_posterior_variance(fit,v)
apply(e$ppp, c(2,3,4), var) == get_posterior_variance(fit,ppp)
}

## End(Not run)
dontrun
## End(Not run)
dontrun
```

### Description

Synthesizes samples from posterior predictive distributions.

### Usage

```r
get_samples_from_Posterior_Predictive_distribution(
  StanS4class,
  counter.plot.via.schatter.plot = TRUE,
  new.imaging.device = TRUE,
)```

---

**get_samples_from_Posterior_Predictive_distribution**

*Synthesizes Samples from Predictive Posterior Distributions (PPD).*

---

**Description**

Synthesizes samples from posterior predictive distributions.

**Usage**

```r
get_samples_from_Posterior_Predictive_distribution(
  StanS4class,
  counter.plot.via.schatter.plot = TRUE,
  new.imaging.device = TRUE,
)```
**get_samples_from_Posterior_Predictive_distribution**

```r
upper_x,
upper_y,
Colour = TRUE,
plot.replicated.points = TRUE
)
```

### Arguments

- **StanS4class**
  - An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc.

- **counter.plot.via.schatter.plot**
  - Logical: TRUE or FALSE. Whether counter plot via schatter plot is drawn. Default = TRUE.

- **new.imaging.device**
  - Logical: TRUE or FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

- **upper_x**
  - This is a upper bound for the axis of the horizontal coordinate of FROC curve.

- **upper_y**
  - This is a upper bound for the axis of the vertical coordinate of FROC curve.

- **Colour**
  - Logical: TRUE or FALSE. Whether Colour of curves is dark theme or not.

- **plot.replicated.points**
  - TRUE or FALSE. If true, then plot replicated points (hits, false alarms) by the scatter plot. This process will takes a long times. So if user has no time, then FALSE will help you.

### Details

This methods to draw from the PPD is described in Gelman book, Bayesian Data Analysis. The aim of this function is to evaluate the chi square test statistics as a Bayesian sense. According to Gelman book, the chi square test need the samples from the PPD. So, we use this function to accomplish this task.

### Value

A list of datalists from the posterior predictive distribution

### Examples

```r
## Not run:
fit <- fit_Bayesian_FROC(
ite = 1111,
summary = FALSE ,
dataList = BayesianFROC::dataList.Chakra.1 )
```
get_samples_from_Posterior_Predictive_distribution

#======= The first example ======================================================
TPs.FPs <- get_samples_from_Posterior_Predictive_distribution(fit)

#======= The Second Example: Short cut ===========================================
# If user has no time, then plot.replicated.points=FALSE will help you.
# By setting FALSE, the replicated data from the posterior predictive
# distribution does not draw, and hence the running time of function become shorter.
TPs.FPs <- get_samples_from_Posterior_Predictive_distribution(fit,
  plot.replicated.points = FALSE)

# Close the graphic device to avoid errors in R CMD check.
grDevices::dev.new();plot(stats::runif(100),stats::runif(100))

#================The third example: From Hand made data to fitting ==========
# To draw the scatter plots of hits and false alarms synthesized from the posterior
# predictive distribution for the submission to a journal,
# then the colored plot is not appropriate.
# So, by setting the argument Colour = FALSE, the scatter plot colored by black and white.
# we use the resulting plot for submission.
get_samples_from_Posterior_Predictive_distribution(fit,Colour = FALSE)
g <-get_samples_from_Posterior_Predictive_distribution(fit)
x <- g$CFP
y <- g$CTP

plot( hexbin::hexbin(unlist(x),unlist(y))  )

# Close the graphic device to avoid errors in R CMD check.
Close_all_graphic_devices()
## End(Not run)# dottest
Draw FROC curves by two parameters $a$ and $b$

**Description**

Plot FROC curves based on two parameters $a$ and $b$.

**Usage**

```r
ggplotFROC(
  a,
  b,
  mesh.for.drawing.curve = 10000,
  upper_x = 1,
  upper_y = 1,
  lower_y = 0,
  dataList,
  StanS4class
)
```

**Arguments**

- **a**: An arbitrary real number. It is no need to require any assumption, but I use such as $a=\mu/\sigma$, where $\mu$ is a mean of signal distribution and $\sigma$ is its standard deviation in the bi-normal assumption.
- **b**: An arbitrary positive real number. I use such as $b=1/\sigma$, where $\sigma$ is a standard deviation of signal distribution in the bi-normal assumption.
- **mesh.for.drawing.curve**: A positive large integer, indicating number of dots drawing the curves, Default =10000.
- **upper_x**: A positive real number, indicating the frame size of drawing picture.
- **upper_y**: A positive real number, indicating the frame size of drawing picture.
- **lower_y**: A positive real number, indicating the frame size of drawing picture.
- **dataList**: A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and a single modality data, the dataList is made by the following manner:

```r
dataList.Example <- list(
  h = c(41,22,14,8,1), # number of hits for each confidence level
  f = c(1,2,5,11,13), # number of false alarms for each confidence level
)
NL = 124, # number of lesions (signals)
NI = 63, # number of images (trials)
C = 5) # number of confidence, .. the author thinks it can be calculated
as the length of h or f ...? ha, why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).
To make this R object dataList representing FROC data, this package provides three functions:

- convertFromJafroc() If data is a JAFROC xlsx formulation.
- dataset_creator_new_version() Enter TP and FP data by table.
- create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

- f Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- h Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- NL A positive integer, representing Number of Lesions.
- NI A positive integer, representing Number of Images.
- C A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored , since it is created by c <- c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

**data Format:**

**A single reader and a single modality case**

<table>
<thead>
<tr>
<th></th>
<th>confidence level c</th>
<th>No. of false alarms f</th>
<th>No. of hits h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>( c )</td>
<td>( f )</td>
<td>( h )</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>probably present</td>
<td>4</td>
<td>( F_4 ) = 2</td>
<td>( H_4 ) = 22</td>
</tr>
<tr>
<td>equivocal</td>
<td>3</td>
<td>( F_3 ) = 5</td>
<td>( H_3 ) = 14</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>( F_2 ) = 11</td>
<td>( H_2 ) = 8</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>( F_1 ) = 13</td>
<td>( H_1 ) = 1</td>
</tr>
</tbody>
</table>

\(* false alarms = False Positives = FP  
* hits = True Positives = TP  

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector \( c \) should not be specified. If specified, will be ignored, since it is created by \( c \leftarrow c(rep(C:1)) \) automatically in the program and do not refer from user input data even if it is specified explicitly, where \( C \) is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector \( c \leftarrow c(rep(C:1)) \) via a table which can be displayed by the function \texttt{viewdata()}.

### Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function \texttt{fit_Bayesian_FROC()}, dataset represented by an \texttt{R} list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

- \( C \) A positive integer, representing the highest number of confidence level, this is a scalar.
- \( M \) A positive integer vector, representing the number of modalities.
- \( Q \) A positive integer, representing the number of readers.
- \( m \) A vector of positive integers, representing the modality ID vector.
- \( q \) A vector of positive integers, representing the reader ID vector.
- \( c \) A vector of positive integers, representing the confidence level. This vector must be made by \texttt{rep(rep(C:1),M*Q)}
- \( h \) A vector of non-negative integers, representing the number of hits.
- \( f \) A vector of non-negative integers, representing the number of false alarms.
- \( NL \) A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above \texttt{R} object. However, each confidence level vector is not included in the data, because it is created automatically from \( C \). To confirm false positives and
hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

*Example data.*

*Multiple readers and multiple modalities (i.e., MRMC)*

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP *

*hits = True Positives = TP *

**StanS4class**

An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and etc.

**Description**

Plot FROC curves based on two parameters a and b.
Usage

```r
ggplotFROC.EAP(
  a,
  b,
  mesh.for.drawing.curve = 10000,
  upper_x = 1,
  upper_y = 1,
  lower_y = 0,
  dataList,
  StanS4class
)
```

Arguments

- `a`: An arbitrary real number. It is no need to require any assumption, but I use such as \( a = \mu / \sigma \), where \( \mu \) is a mean of signal distribution and \( \sigma \) is its standard deviation in the bi-normal assumption.

- `b`: An arbitrary positive real number. I use such as \( b = 1 / \sigma \), where \( \sigma \) is a standard deviation of signal distribution in the bi-normal assumption.

- `mesh.for.drawing.curve`: A positive large integer, indicating number of dots drawing the curves, Default = 10000.

- `upper_x`: A positive real number, indicating the frame size of drawing picture.

- `upper_y`: A positive real number, indicating the frame size of drawing picture.

- `lower_y`: A positive real number, indicating the frame size of drawing picture.

- `dataList`: A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The `dataList` will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and a single modality data, the `dataList` is made by the following manner:

```r
dataList.Example <- list(
  h = c(41, 22, 14, 8, 1), # number of hits for each confidence level
  f = c(1, 2, 5, 11, 13), # number of false alarms for each confidence level
  NL = 124, # number of lesions (signals)
  NI = 63, # number of images (trials)
  C = 5) # number of confidence, the author thinks it can be calculated as the length of h or f ...? ha, why I included this. ha ... should be omitted.
)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList` representing FROC data, this package provides three functions:
Before fitting a model, we can confirm our dataset is correctly formulated by using the function `viewdata()`.

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), `dataList` is a list consisting of `f`, `h`, `NL`, `NI`, `C` where `f`, `h` are numeric vectors and `NL`, `NI`, `C` are positive integers.

- `f` Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- `h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- `NL` A positive integer, representing Number of Lesions.
- `NI` A positive integer, representing Number of Images.
- `C` A positive integer, representing Number of Confidence level.

The detail of this dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

**Data Format:**

*A single reader and a single modality case*

---

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
</table>

---

*false alarms* = False Positives = FP
* hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector c <-c(rep(C:1)) via a table which can be displayed by the function viewdata().

---

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function fit_Bayesian_FROC(), dataset represented by an R list object representing FROC data must contain components m, q, c, h, f, NL, C, M, Q.

C  A positive integer, representing the highest number of confidence level, this is a scalar.
M  A positive integer vector, representing the number of modalities.
Q  A positive integer, representing the number of readers.
m  A vector of positive integers, representing the modality ID vector.
q  A vector of positive integers, representing the reader ID vector.
c  A vector of positive integers, representing the confidence level. This vector must be made by rep(rep(C:1),M*Q)
h  A vector of non-negative integers, representing the number of hits.
f  A vector of non-negative integers, representing the number of false alarms.
NL  A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function viewdata() shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)


<table>
<thead>
<tr>
<th>m</th>
<th>q</th>
<th>c</th>
<th>f</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

---

*false alarms* = False Positives = FP  
*hits* = True Positives = TP

**StanS4class**  
An S4 object of class *stanfitExtended* which is an inherited class from the S4 class *stanfit*. This R object is a fitted model object as a return value of the function *fit_Bayesian_FROC()*.

It can be passed to *DrawCurves()*., *ppp()* and ... etc

---

**give_name_srsc_CFP_CTP_vector**  
*Give a Name For CTP CFP vector*

---

**Description**

Give a Name for a vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs).

**Usage**

```r
give_name_srsc_CFP_CTP_vector(
  vector,
  CFP.or.CTP = "CFP",
  ModifiedPoisson = FALSE
)
```
Arguments

vector
A vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs).

CFP.or.CTP
"CFP" or "CTP". Default value is “CFP”.

ModifiedPoisson
 Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows (\(F_c\) denotes the number of false alarms with confidence level \(c\))

\[
\begin{align*}
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L}, \\
\frac{F_2 + F_3 + F_4 + F_5}{N_L}, \\
\frac{F_3 + F_4 + F_5}{N_L}, \\
\frac{F_4 + F_5}{N_L}, \\
\frac{F_5}{N_L}
\end{align*}
\]

where \(N_L\) is a number of lesions (signal). To emphasize its denominator \(N_L\), we also call it the False Positive Fraction (FPF) per lesion.

On the other hand,
if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\begin{align*}
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I}, \\
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\end{align*}
\]
\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE})
or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE})

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}.

In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal.

One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

**Details**

Some function in this package give the return values of vectors representing the CFP or CTPs. Using this function, we specify what the components of vector means. This is important since its order is not deterministic, that is, its order give two case, one is decreasing and one is increasing order. So, to avoid such confusion, the name should be specified. Of course this function is no needed for user to know or to use it.

**Value**

A vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs) with its name.
Examples

```r
h <- BayesianFROC::dataList.Chakra.1$h
NL <- BayesianFROC::dataList.Chakra.1$NL
CTP.vector <- cumsum(h)/NL
CTP.vector.with.name <- give_name_srsc_CFP_CTP_vector(CTP.vector)
```

Description

By specifying the data, the names are given for each component vectors.

Usage

```r
give_name_srsc_data(dataList)
```

Arguments

dataList A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and a single modality data, the dataList is made by the following manner:

```r
dataList.Example <- list(h = c(41, 22, 14, 8, 1), # number of hits for each confidence level
                      f = c(1, 2, 5, 11, 13), # number of false alarms for each confidence level
                      NL = 124, # number of lesions (signals)
                      NI = 63, # number of images (trials)
                      C = 5) # number of confidence, ... the author thinks it can be calculated as the length of h or f ...? ha, why I included this. ha .. should be omitted.
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`. To make this R object dataList representing FROC data, this package provides three functions: `convertFromJafroc()` If data is a JAFROC xlsx formulation.
dataset_creator_new_version() Enter TP and FP data by table.
create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), dataList is a list consisting of f,h,NL,NI,C where f,h are numeric vectors and NL,NI,C are positive integers.

- **f** Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- **h** Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- **NL** A positive integer, representing Number of Lesions.
- **NI** A positive integer, representing Number of Images.
- **C** A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

**data Format:**

A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R console -&gt;</td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP

*hits = True Positives = TP
Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector c <-c(rep(C:1)) via a table which can be displayed by the function viewdata().

Multiple readers and multiple modalities case, i.e., MRMC case

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function fit_Bayesian_FROC(), dataset represented by an R list object representing FROC data must contain components m, q, c, h, f, NL, C, M, Q.

C A positive integer, representing the highest number of confidence level, this is a scalar.
M A positive integer vector, representing the number of modalities.
Q A positive integer, representing the number of readers.
m A vector of positive integers, representing the modality ID vector.
q A vector of positive integers, representing the reader ID vector.
c A vector of positive integers, representing the confidence level. This vector must be made by rep(rep(C:1), M*Q)
h A vector of non-negative integers, representing the number of hits.
f A vector of non-negative integers, representing the number of false alarms.
NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function viewdata() shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

Example data.

Multiple readers and multiple modalities (i.e., MRMC)
Details
This is only available on singler reader and single modality case, not available on MRMC case.

Examples
```r
#> dataList.Chakra.2
#$f
#[1]  4 13 44
#
#$h
#[1] 122 31 20
#
#$NL
#[1]  269
#
#$NI
#[1]  57
#
#$C
#[1]  3

dataList.with.name <- give_name_srsc_data(dataList.Chakra.2)
```

```r
#> dataList.with.name
#$f
# F(3) F(2) F(1)
```
hits_creator_from_rate

MRMC Dataset Creator From Hit Rate.

Description
From hit rates, data of hits are created.

Usage

hits_creator_from_rate(NL = 252, seed = 123, p.truth = BayesianFROC::p_truth)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>Number of Lesions.</td>
</tr>
<tr>
<td>seed</td>
<td>The seed for creating data consisting of the number of hits synthesized by</td>
</tr>
<tr>
<td></td>
<td>the binomial distributions with the specified seed.</td>
</tr>
<tr>
<td>p.truth</td>
<td>Array of dimension (C, M, Q), where C = number of confidence levels, M =</td>
</tr>
<tr>
<td></td>
<td>number of modalities, Q = number of readers.</td>
</tr>
</tbody>
</table>

Details
Random variables of hits are distributed as follows.

\[ h_{5,m,r} \sim \text{Binomial}(p_{5,m,r}, N_L), \]

then \( h_{4,m,r} \) should be drawn from the binomial distribution with remaining targets.
\[ h_{4,m,r} \sim \text{Binomial}\left( \frac{p_{4,m,r}}{1-p_{5,m,r}}, N_L - h_{5,m,r} \right). \]

Similarly,
\[ h_{3,m,r} \sim \text{Binomial}\left( \frac{p_{3,m,r}}{1-p_{5,m,r}-p_{4,m,r}}, N_L - h_{5,m,r} - h_{4,m,r} \right). \]
\[ h_{2,m,r} \sim \text{Binomial}\left( \frac{p_{2,m,r}}{1-p_{5,m,r}-p_{4,m,r}-p_{3,m,r}}, N_L - h_{5,m,r} - h_{4,m,r} - h_{3,m,r} \right). \]
\[ h_{1,m,r} \sim \text{Binomial}\left( \frac{p_{1,m,r}}{1-p_{5,m,r}-p_{4,m,r}-p_{3,m,r}-p_{2,m,r}}, N_L - h_{5,m,r} - h_{4,m,r} - h_{3,m,r} - h_{2,m,r} \right). \]

\( p.\;\text{truth} \) is an array representing \( p_{c,m,r} \). By specifying the array \( p.\;\text{truth} \) (and hence \( p_{c,m,r} \)), with the above model, we can calculate hit data \( h_{c,m,r} \) for each \( c, m, r \).

**Value**

Hits Data, an array of dimension [Confidence, Modality, Reader].

**Examples**

```r
# Not run:
#========================================================================================
# 2019 Sept 6 1) Using the default hit values, hit data are created as follows;
#========================================================================================

hits <- hits_creator_from_rate()

#========================================================================================
# 2019 Sept 6 2) If user want to use their own hit rates, then use the following codes:
#========================================================================================

h <- hits_creator_from_rate(NL=252, seed=123, p.truth =
```

```
```r
array(c(
  c(0.03, 0.13, 0.2, 0.3, 0.4, # for M=1 Q=1
      0.04, 0.23, 0.3, 0.4, 0.5), # for M=2 Q=1,
  c(0.05, 0.33, 0.4, 0.5, 0.6, # for M=1 Q=2
      0.06, 0.43, 0.5, 0.6, 0.7), # for M=2 Q=2,
  c(0.07, 0.53, 0.6, 0.7, 0.8, # for M=1 Q=3
      0.08, 0.63, 0.7, 0.8, 0.9) # for M=2 Q=3,
),

dim = c(5, 2, 3) # C M Q
)
```

# =========================================================================================
# 2019 Sept 6 3) If user want to use their own hit rates, then use the following codes:
# =========================================================================================

h <- hits_creator_from_rate(
  NL=252,
  seed = 123,
  p.truth =
    array(c(
      c(0.03, 0.1, 0.2, 0.3, 0.4, # for M=1 Q=1
          0.04, 0.2, 0.3, 0.4, 0.5), # for M=2 Q=1
          0.05, 0.3, 0.4, 0.5, 0.6), # for M=3 Q=1
      c(0.05, 0.33, 0.4, 0.5, 0.6, # for M=1 Q=2
          0.06, 0.43, 0.5, 0.6, 0.7, # for M=2 Q=2
          0.05, 0.3, 0.4, 0.5, 0.6), # for M=3 Q=2
      c(0.07, 0.53, 0.6, 0.7, 0.8, # for M=1 Q=3
          0.08, 0.63, 0.7, 0.8, 0.9, # for M=2 Q=3
          0.05, 0.3, 0.4, 0.5, 0.6) # for M=3 Q=3
    ),

dim = c(5, 3, 3) # C M Q
)
h <- hits_creator_from_rate(
  NL=252,
  seed =123,
  p.truth =
    array(c(
      c(0.03,0.1,0.2,0.3,0.4), #for M=1 Q=1
      0.04,0.2,0.3,0.4,0.5, #for M=2 Q=1
      0.05,0.3,0.4,0.5,0.6), #for M=3 Q=1
    ),
    dim = c(5,3,1) #C M Q
  )
)

# The third example

# The hits rate cannot take any values, since there is a trend that a hit rate of
# a higher confidence level is a higher. So, If it is difficult for user to create
# a true hit rates, then by taking estimates as true parameters,
# user can replicate datasets.
# To do so, work follow is first fitting, secondly extracting estimates,
# thirdly apply this function (hits_creator_from_rate() ).

# * Fitting
fit <- fit_Bayesian_FROC(
  dataList.Chakra.Web.orderd,
  ite = 1111, # For simplicity, we take small MCMC samples.
  summary =FALSE)

# * Extracting
estimates <- extract_estimates_MRMC(fit)
ppp <- estimates$ppp.EAP

# Note that ppp is an array
# whose dimension is constituted by number of confidence levels, modalities, readers.

# * Replicating as an true values is ppp

hits <- hits_creator_from_rate(p.truth = ppp)

# <<Remark>>
# ppp is an array. ignoring its indices, we can write that
# hits ~ Binomial(ppp, NL)
# Where NL is a number of lesions.
# By writing its component explicitly, we can write
# Hits[c,m,r] ~ Binomial(ppp[c,m,r], NL)
# Where c means the c-th confidence level,
# m means the m-th modality,
# r means the r-th reader.

## End(Not run)# dottest

---

**hits_false_alarms_creator_from_thresholds**

*Hits and False Alarms Creator*

**Description**

From the parameter of the bi-normal assumptions, hits and false alarms are generated.

**Usage**

```r
hits_false_alarms_creator_from_thresholds()
```
replicate.dataset = 3,  
ModifiedPoisson = FALSE,  
mean.truth = 0.6,  
sd.truth = 5.3,  
z.truth = c(-0.8, 0.7, 2.38),  
NL = 259,  
NI = 57,  
summary = TRUE,  
initial.seed = 12345  
}

Arguments

replicate.dataset

A Number indicate that how many you replicate dataset from user's specified dataset.

ModifiedPoisson

Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author's paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows (F_c denotes the number of false alarms with confidence level c)

\[ \frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L}, \]

\[ \frac{F_2 + F_3 + F_4 + F_5}{N_L}, \]

\[ \frac{F_3 + F_4 + F_5}{N_L}, \]

\[ \frac{F_4 + F_5}{N_L}, \]

\[ \frac{F_5}{N_L}. \]
where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand, if \texttt{ModifiedPoisson = FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be graded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textit{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textit{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

\begin{itemize}
\setlength\itemsep{0em}
\item \texttt{mean.truth} This is a parameter of the latent Gaussian assumption for the noise distribution.
\item \texttt{sd.truth} This is a parameter of the latent Gaussian assumption for the noise distribution.
\end{itemize}
This is a parameter of the latent Gaussian assumption for the noise distribution.

- **NL**: Number of Lesions.
- **NI**: Number of Images.

**summary**

Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**initial.seed**

Replicated datasets are created using a continuous sequence of seeds and its initial seed is specified by this argument. For example, if you choose initial.seed = 12300, then the replicated datasets are created from using the sequence of seeds: 12301, 12302, 12303, 12304, ...

**Details**

From the fixed parameters of bi-normal assumptions, we replicate data, that is, we draw the data from the distributions whose parameters are known. Especially, we interest the hits and false alarms since the number of images, lesions and confidence level is same for all replications. So, it is sufficient to check the hits and false alarms.

**Value**

Datasets Including Hits and False Alarms

**Examples**

```r
# Not run:
#======================================The first example======================================
# Replication of Data from Fixed (specified) Parameters.

a <- hits_false_alarms_creator_from_thresholds(replicate.dataset = 1)

# Extract the first replicated dataset:

da[[1]]$NL
da[[1]]$NI
da[[1]]$f
da[[1]]$h
da[[1]]$C

#======================================The second example======================================
# Replication of Data from Fixed (specified) Parameters.

b <- hits_false_alarms_creator_from_thresholds(replicate.dataset = 2)

# Extract the first replicated dataset:

hb[[1]]$NL
hb[[1]]$NI
```
# Extract the second replicated dataset:

b[[2]]$NL
b[[2]]$NI
b[[2]]$f
b[[2]]$h
b[[2]]$C

#================The Third example======================================
# Replication of Data from Fixed (specified) Parameters.

# Extract the first replicated dataset:

c[[1]]$NL
c[[1]]$NI
c[[1]]$f
c[[1]]$h
c[[1]]$C

# Extract the second replicated dataset:

c[[2]]$NL
c[[2]]$NI
c[[2]]$f
c[[2]]$h
c[[2]]$C

# Extract the third replicated dataset:

c[[3]]$NL
c[[3]]$NI
c[[3]]$f
c[[3]]$h
c[[3]]$C

## End(Not run)# dottest
hits_from_thresholds  

MRMC Hit Creator from thresholds, mean and S.D.

Description

From threshold, mean and S.D., data of hit rate are created.

Usage

```r
hits_from_thresholds(
  z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  NL = 252,
  seed = 123
)
```

Arguments

- `z.truth`: Vector of dimension = C represents the thresholds of bi-normal assumption.
- `mu.truth`: array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.
- `v.truth`: array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
- `NL`: Number of Lesions.
- `seed`: The seed for creating data consisting of the number of hits synthesized by the binomial distributions with the specified seed.

Value

Hits Data for MRMC. The reason that hits is multiple reader and multiple modalities arise from the multiple indices of mean and S.D. of signal distribution of the bi-normal assumption.

Examples

```r
## Not run:
hits.rate.p <- hits_from_thresholds()

## End(Not run)#dontrun
```
**hits_rate_creator**  
**MRMC Hit Rates Creator from Thresholds, Mean and S.D.**

**Description**

From thresholds, data of hit rate are created.

Note that the return values has changed from $p$ (in R notation: `ppp`) to

$$\text{hitrate}_c := \frac{p_c(\theta)}{1 - p_C(\theta) - p_{C-1}(\theta) - \cdots - p_{c+1}(\theta)}$$

**Usage**

```r
hits_rate_creator(
  z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  is_hit_rate_adjusted = FALSE
)
```

**Arguments**

- `z.truth` Vector of dimension = C represents the thresholds of bi-normal assumption.
- `mu.truth` array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.
- `v.truth` array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
- `is_hit_rate_adjusted` whether the return value is a vector of

  $$p_c(\theta)$$

  or

  $$\text{hitrate}_c := \frac{p_c(\theta)}{1 - p_C(\theta) - p_{C-1}(\theta) - \cdots - p_{c+1}(\theta)}$$

  The former is the default (FALSE) and the later is returned if `is_hit_rate_adjusted=TRUE`.

**Value**

A vector of the hit rate:

$$\text{hitrate}_c := \frac{p_c(\theta)}{1 - p_C(\theta) - p_{C-1}(\theta) - \cdots - p_{c+1}(\theta)}$$

Do not confuse the old version `ppp` which is an array with three indices: `ppp[C,M,Q]`. 
Examples

```
## Not run:
#------------------The first example-----------------------------------------
#
# Using default values for hit rates, we can create a data of hits as follows:
#  hits.rate <- hits_rate_creator()
#
#------------------The second example----------------------------------------
#
# Using the hit rate from the hits_rate_creator(), we can get the hits data:
#  hits_creator_from_rate(p.truth = hits_rate_creator())
#
#------------------The remark for example------------------------------------
#
# The author does not show how to specify the hit rates or thresholds.
# For the details of it, please see the default values of such quantities.
#
#------------------The 4-th example-----------------------------------------
#
# p.truth.array <- hits_rate_creator()
```

```
# 2019 Sept 6
```

```
## End(Not run)# dottest
```

---

**hit_rate_adjusted_from_the_vector_p**

*hit rate adjusted from a vector p*

### Description

hit rate adjusted from a vector p
hit_rate_adjusted_from_the_vector_p

Usage

hit_rate_adjusted_from_the_vector_p(p_vector)

Arguments

p_vector A vector

Value

A vector

Examples

p <- c(1,2,3)
a <- hit_rate_adjusted_from_the_vector_p(p)
a

# [1] -0.25 -1.00 3.00

a[3] == 3

#========================================================================================
# application in the function ppp_srsr in this package
#========================================================================================

## Not run:

f <- fit_Bayesian_FROC(dataList = d)
e <- rstan::extract(f)
q<-e$p[1,]
hit_rate_adjusted_from_the_vector_p(q)
t(apply(e$p,hit_rate_adjusted_from_the_vector_p,MARGIN = 1))[1,]
q<-e$p[2,]
hit_rate_adjusted_from_the_vector_p(q)
t(apply(e$p,hit_rate_adjusted_from_the_vector_p,MARGIN = 1))[2,]

## End(Not run)
**initial_values_specification_for_stan_in_case_of_MRCM**

*Initial values for HMC (Hamiltonian Monte Carlo Markov Chains)*

---

### Description

An internal function.

### Usage

```r
initial_values_specification_for_stan_in_case_of_MRCM(dataList)
```

### Arguments

- **dataList**: A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also.

The `dataList` will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`.

For the single reader and a single modality data, the `dataList` is made by the following manner:

```r
dataList.Example <- list(
  h = c(41, 22, 14, 8, 1),  # number of hits for each confidence level
  f = c(1, 2, 5, 11, 13),  # number of false alarms for each confidence level
  NL = 124,  # number of lesions (signals)
  NI = 63,   # number of images (trials)
  C = 5)     # number of confidence... the author thinks it can be calculated as the length of h or f...? ha, why I included this. ha.. should be omitted.
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList` representing FROC data, this package provides three functions:

- `convertFromJafroc()` If data is a JAFROC xlsx formulation.
- `dataset_creator_new_version()` Enter TP and FP data by table.
- `create_dataset()` Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function `viewdata()`.

---

### A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srg), `dataList` is a list consisting of `f, h, NL, NI, C` where `f, h` are numeric vectors and `NL, NI, C` are positive integers.
f Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.

h Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

NL A positive integer, representing Number of Lesions.

NI A positive integer, representing Number of Images.

C A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified, will be ignored, since it is created by c<-c(rep(C:1)) in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

data Format:
A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

* false alarms = False Positives = FP
* hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c<-c(rep(C:1)) automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you
should check the compatibility of your data and the confidence level vector \( c \leftarrow c(\text{rep}(C:1)) \) via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an R list object representing FROC data must contain components \( m, q, c, h, f, NL, C, M, Q \).

\( C \): A positive integer, representing the highest number of confidence level, this is a scalar.

\( M \): A positive integer vector, representing the number of modalities.

\( Q \): A positive integer, representing the number of readers.

\( m \): A vector of positive integers, representing the modality ID vector.

\( q \): A vector of positive integers, representing the reader ID vector.

\( c \): A vector of positive integers, representing the confidence level. This vector must be made by \( \text{rep}(\text{rep}(C:1), M*Q) \).

\( h \): A vector of non-negative integers, representing the number of hits.

\( f \): A vector of non-negative integers, representing the number of false alarms.

\( NL \): A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by \( C \)) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from \( C \). To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

*Multiple readers and multiple modalities (i.e., MRMC)*

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>( q )</td>
<td>( c )</td>
<td>( f )</td>
<td>( h )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
</tbody>
</table>
install_imports

Details
This attempt failed, that is, I cannot specify the initial values so that the `rstan::sampling()` does not say the following:
Rejecting initial value:
Log probability evaluates to log(0), i.e. negative infinity.
Stan can’t start sampling from this initial value.

Value
Initial values specification. See the detailed documentation for the init argument in `stan()`.

Examples

```r
init <- initial_values_specification_for_stan_in_case_of_MRMC(dataList.Chakra.Web)

# Where init is the variable of the rstan::stan() or rstan::sampling()
```

install_imports

Installer

Description
This is an installer for required packages in this package. To install this package BayesianFROC, we use the package xlsx which require the Java. So, if use buy a new computer and and it does not have installed the Java, then please install Java.

Usage

```r
install_imports()
```
inv_Phi

Inverse function of the Cumulative distribution function \( \Phi(x) \) of the Standard Gaussian, where \( x \) is a real number.

Description

The author is confused `stats::qnorm()` with `stats::pnorm()` and thus he made this.

Usage

inv_Phi(x)

Arguments

x

A real. To be passed to the function `stats::qnorm()`

Details

In Stan file, it is `inv_Phi()` and not `inv_phi`.

Since \( \Phi(x) \) is monotonic, it follows that \( \frac{d}{dx} \Phi^{-1} = (\frac{d}{dx} \Phi)^{-1} > 0 \), and thus \( \Phi^{-1}(x) \) is also monotonic.

Value

A real number: \( \Phi^{-1}(x) \)

See Also

`Phi()`, `Phi_inv()`

Examples

```r
x <- runif(100)
Phi_inv(x) == stats::qnorm(x)
inv_Phi(x) == stats::qnorm(x)
```
make_TeX

Make a TeX file for summary

Description

Under Construction... “This only inner funtion, in the future I run this in the fit_Bayesian_FROC().

Usage

make_TeX()

Value

TeX file reflected the analysis

make_true_parameter_MRMC

Make a true model parameter and include it in this package

Description

Make a true model parameter and include it in this package

Usage

make_true_parameter_MRMC(StanS4class)

Arguments

StanS4class An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC().
It can be passed to DrawCurves(), ppp() and etc
Create metadata for MRMC data.

Description

The so-called false positive fraction (FPF) and the true positive fraction (TPF) are calculated from the number of hits (True Positives: TPs) and the number of false alarms (False Positives: FPs).

Usage

metadata_srsc_per_image(dataList, ModifiedPoisson)

Arguments

dataList A list, should include m,q,c,h,f,NL,C,M,Q which means
c should be created by c <-c(rep(C:1)), where C is the number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.
h means the number of hits
f means the number of false alarm
NL means the Total number of lesions for all images
C means the highest number of confidence level

ModifiedPoisson

Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$
where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if $\text{ModifiedPoisson} = \text{FALSE}$ (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be ragraded as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = FALSE) or as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = TRUE)

If ModifiedPoisson = TRUE, then FROC curve means the expected pair of FPF per lesion and TPF. On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only
per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

Details

From data of number of hits (True Positive: TP) and false alarms (False Positive: FP), we calculate the number of cumulative false positives (FPF) and cumulative hits (TPF).

Because there are three subscripts, reader, modality, and image, we create array format and vector format etc...

Value

A metadata such as number of cumulative false alarms and hits to create and draw the curve.

Examples

```r
## Not run:
#========================================================================================
# TP and FP
#========================================================================================

dat <- BayesianFROC::dataList.Chakra.Web

#========================================================================================
# Calculates TPF and FPF from TP and FP
#========================================================================================

metadata_srsc_per_image(dat)

# Revised 2019 Nov.

## End(Not run) dottest
```
**metadata_to_DrawCurve_MRMC**  
*Create metadata for MRMC data*

**Description**

From data of number of hits and false alarms, we calculate the number of cumulative false positives and hits. Since there are three subscripts, reader, modality, and image, we create array format and vector format etc...

**Usage**

```r
metadata_to_DrawCurve_MRMC(StanS4class, mesh.for.drawing.curve = 5000)
```

**Arguments**

- **StanS4class**: An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`.  
  It can be passed to `DrawCurves()`, `ppp()` and etc.

- **mesh.for.drawing.curve**: A positive large integer, indicating number of dots drawing the curves, Default =10000.

**Value**

A metadata such as number of cumulative false alarms and hits to create and draw the curve.

---

**metadata_to_fit_MRMC**  
*Create metadata for MRMC data*

**Description**

The so-called *false positive fraction (FPF)* and the *true positive fraction (TPF)* are calculated from the number of hits (True Positives: TPs) and the number of false alarms (False Positives: FPs)

**Usage**

```r
metadata_to_fit_MRMC(dataList, ModifiedPoisson = FALSE)
```
Arguments

dataList A list, consisting of the following R objects: m, q, c, h, f, NL, C, M, Q each of which means from the right
m : A vector, indicating the modality ID = 1,2,... which does not include zero.
q : A vector, indicating the reader ID = 1,2,... which does not include zero.
c : A vector, indicating the confidence = 1,2,... which does not include zero.
h : A vector, indicating the number of hits
f : A vector, indicating the number of false alarm
NL : An positive integer, indicating the number of lesions for all images
C : An positive integer, indicating the highest number of confidence level
M : An positive integer, indicating the number of modalities
Q : An positive integer, indicating the number of readers.
The detail of these dataset, please see the example datasets, e.g. dd.

ModifiedPoisson

Logical, that is TRUE or FALSE.
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly,
If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.
For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes , now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L}.
\]
where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand,

if \texttt{ModifiedPoisson = FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textit{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textit{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.
Details

To fit a model to data, we need a hit data and false data formulated by both an array and a vector. It also calculates the so-called False Positive Fractions (FPF) (resp. True Positive Fractions (TPF)) which are cumulative sums of false alarms (resp. hits) over number of lesions or images.

From data of number of hits and false alarms, we calculate the number of cumulative false positives and hits per image or lesion, in other words, False Positive Fraction (FPF) and True Positive Fraction (TPF). Since there are three subscripts, reader, modality, and image, we can create array format or vector format etc...

Abbreviations

**FPF**: false positive fraction  
**TPF**: true positive fraction  
**hit**: True Positive = TP  
**false alarms**: False Positive = FP

The traditionaly, the so-called FPF:False Positive Fraction and TPT:True Positive Fraction are used. Recall that our data format:

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms (FP:False Positive)</th>
<th>No. of hits (TP:True Positive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>5</td>
<td>F_5</td>
</tr>
<tr>
<td>probably present</td>
<td>4</td>
<td>F_4</td>
</tr>
<tr>
<td>equivocal</td>
<td>3</td>
<td>F_3</td>
</tr>
<tr>
<td>subtle</td>
<td>2</td>
<td>F_2</td>
</tr>
<tr>
<td>very subtle</td>
<td>1</td>
<td>F_1</td>
</tr>
</tbody>
</table>

FPF is defined as follows;

\[
FPF(5) := \frac{F_5}{NI},
\]

\[
FPF(4) := \frac{F_4 + F_5}{NI},
\]

\[
FPF(3) := \frac{F_3 + F_4 + F_5}{NI},
\]

\[
FPF(2) := \frac{F_2 + F_3 + F_4 + F_5}{NI},
\]

\[
FPF(1) := \frac{F_1 + F_2 + F_3 + F_4 + F_5}{NI}.
\]

TPF is defined as follows;
The False Positive Fraction (FPF) and True Positive Fraction (TPF) are also calculated.

**The components of list** I rediscover it at 2019 Jun 18, I am not sure it is useful? 2019 Dec 8

- **harray** An array of hit, dimension $[C,M,Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.
- **farray** An array of false alarms, dimension $[C,M,Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.
- **hharray** An array of cumulative hits, dimension $[C,M,Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.
- **ffarray** An array of cumulative false alarms, dimension $[C,M,Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.
- **hhN** An array of TPF, dimension $[C*M*Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.
- **ffN** An array of FPF, dimension $[C*M*Q]$, where $C,M,Q$ are a number of confidence level, modalities, readers, respectively.

Revised Nov. 21
Examples

#========================================================================================
# First, we prepare the data endowed with this package.
#========================================================================================

dat <- get(data("dataList.Chakra.Web"))

#========================================================================================
# #Calculate FPFs and TPFs and etc.
#========================================================================================

a <- metadata_to_fit_MRMC(dat)

#Now, we get a meta-data object named "a".

#========================================================================================
# Check of Definition
#========================================================================================

a$hh/dat$NL == a$hhN

# Since all of aboves are TRUE, the hhN is a TPF per NL.

#========================================================================================
# Plot a FPFs and TPFs
#========================================================================================

FPF = a$ffN
TPF = a$hhN

dark_theme()
plot(FPF,TPF)

#========================================================================================
# Plot a FPFs and TPFs via ggplot
#========================================================================================

length(dat$f)==length(FPF)
```r
q <- dat$q
m <- dat$m
df <- data.frame(FPF, TPF, m, q

# ggplot2::ggplot(df, aes(x =FPF, y = TPF, colour = q, group = m)) + ggplot2::geom_point()

# Revised 2019 Jun 18, Revised 2019 Sept 9
```

---

**mu**

*Mean of signal: parameter of an MRMC model*

**Description**

A posterior mean of the model parameter for data ddd as an example of truth parameter.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com >

**See Also**

make_true_parameter_MRMC

**Examples**

```r
#> BayesianFROC::mu

[,]  [,2]  [,3]  [,4]
[1,]  1.914686  0.7933306  1.526482  0.9543375
[2,]  2.008008  1.2005846  2.081756  1.0197703
[3,]  1.532117  0.5851726  1.513018  0.8879678

# [modality, reader]
```
mu_truth

Mean of signal: parameter of an MRMC model

Description

A posterior mean of the model parameter for data ddd as an example of truth parameter.

Details

Mean Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

See Also

`hits_creator_from_rate`

Examples

```r
#> mu_truth
#  # [,1] [,2] [,3] [,4]
# [1,]  1.730751  0.8298189  1.334771  0.6386057
# [2,]  1.812523  1.1889223  1.883562  0.7185546
# [3,]  1.319588  0.6062924  1.248589  0.5458920
# [modality, reader]```
mu_truth_creator_for_many_readers_MRMC_data

mu of MRMC model parameter

Description

mu of MRMC model parameter

Usage

mu_truth_creator_for_many_readers_MRMC_data(M, Q)

Arguments

M
An integer, indicating a number of modalities

Q
An integer, indicating a number of readers

Value

An array, representing a mu of MRMC model parameter

Examples

m <- mu_truth_creator_for_many_readers_MRMC_data(M=4, Q=50)

## Not run:

#========================================================================================
# Large number of readers or modalities causes non-convergence MCMC     
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=4, Q=6)

m <- mu_truth_creator_for_many_readers_MRMC_data(M=4, Q=6)

d <- create_dataList_MRMC(mu.truth = m, v.truth = v)

#fit <- fit_Bayesian_FROC( ite = 1111, cha = 1, summary = TRUE, dataList = d )

#plot_FPF_and_TPF_from_a_dataset(fit@dataList)

#========================================================================================
# convergence
#========================================================================================
\begin{verbatim}
v <- v_truth_creator_for_many_readers_MRMC_data(M=2,Q=21)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=2,Q=21)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
#fit <- fit_Bayesian_FROC( ite = 200, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# non-convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=5,Q=6)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=5,Q=6)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 111, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=36)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
#fit <- fit_Bayesian_FROC( ite = 2000, cha = 1, summary = TRUE, dataList = d)

#========================================================================================
# non-convergence
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
# fit <- fit_Bayesian_FROC( ite = 111, cha = 1, summary = TRUE, dataList = d)
\end{verbatim}
**m_q_c_vector_from_M_Q_C**

Creates vectors: *m*, *q*, *c* from integers: *M*, *Q*, *C*

---

### Description

Makes *m*, *q*, *c* vectors from a collection of three integers *M*, *Q*, *C*, where three vectors *m*, *q*, *c* denotes modality ID, reader ID, confidence level, respectively.

### Usage

```r
m_q_c_vector_from_M_Q_C(M, Q, C)
```

### Arguments

- **M**
  - A positive integer, representing modality ID
- **Q**
  - A positive integer, representing reader ID
- **C**
  - A positive integer, representing confidence level

### Details

My research is not supported any found, I am completely independent and only my own or my parents are supported my research. No internet, poor condition, I made this. I must go on untill jounal accepts my manuscripts.

I am not happy to spent with FROC analysis, since it is not my interest. I want to research pure mathematics. I do not want to waste a time in hospital or plurigo nodularis. When I become happy? This program helps me? With great pain at 2019 Sept. 2019 Sept. 8

### Value

A data-frame, including three vectors, which are named *m*, *q*, *c* representing modality ID and reader ID and confidence level, respectively.

For example, the resulting object of `a <- m_q_c_vector_from_M_Q_C(2, 3, 4)` is given by

```r
> a
```
### m_q_c_vector_from_M_Q_C

<table>
<thead>
<tr>
<th>m</th>
<th>q</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Examples

```r
# Create a ID vectors
a <- m_q_c_vector_from_M_Q_C(2,3,4)
a$m
a$q
a$c
```

```r
# validation of this function
a <- m_q_c_vector_from_M_Q_C(5,4,5)
```
\begin{verbatim}
transformations: 
  a$m == dd$m 
  a$c == dd$c 
  a$q == dd$q
\end{verbatim}

\section*{p}

\textit{Hit Rate: parameter of an MRMC model}

\subsection*{Description}
A posterior mean of the model parameter for data \texttt{ddd} as an example of truth parameter.

\subsection*{Author(s)}
Issei Tsunoda <tsunoda.issei1111@gmail.com>

\subsection*{See Also}
make_true_parameter_MRMC

\subsection*{pairs_plot_if_divergent_transition_occurred}

\textit{Pairs plot for divergent transition}

\subsection*{Description}
If divergent transition occurs, the author often forget the variable \texttt{par} or \texttt{pars}. So, I made this to avoid such confusion.

\subsection*{Usage}
pairs_plot_if_divergent_transition_occurred(
  StanS4class, 
  character.representing.paramter = "z"
)

\subsection*{Arguments}
\begin{description}
  \item[StanS4class] An S4 object of class \texttt{stanfitExtended} which is an inherited class from the S4 class \texttt{stanfit}. This R object is a fitted model object as a return value of the function \texttt{fit_Bayesian_FROC()}. It can be passed to \texttt{DrawCurves()}, \texttt{ppp()} and ... etc.
  \item[character.representing.paramter] Character, surrounded by ",", indicating the parameter of model.
\end{description}
Examples

## Not run:
\dontrun{
# Create a fitted model object of class stanfitExtended inherited from stanfit.

fit <- fit_Bayesian_FROC( ite = 1111,
                           summary = FALSE,
                           cha = 1,
                           Null.Hypothesis = FALSE,
                           dataList = dd )

# Pairs plot to examine the divergent transition.

if (interactive()){grDevices::dev.new()}
   pairs_plot_if_divergent_transition_occurred(fit)
if (interactive()){grDevices::dev.off()}

}

## End(Not run)

---

**pause**

**Pause for Demo**

Description

Pause if and only if interactive() = TRUE.

Usage

```r
pause(simple = FALSE)
```

Arguments

- `simple` A logical. If false, then verbose.
  ```r
  pause()
  ```
The Cumulative distribution function $\Phi(x)$ of the Standard Gaussian, namely, mean = 0 and variance = 1.

### Description

$$\Phi(x) := \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

### Usage

Phi(x)

### Arguments

x A real. To be passed to the function stats::pnorm()

### Value

$$\Phi(x) := \int_{-\infty}^{x} \text{Gaussian}(z|0,1)dz$$

### See Also

Phi_inv()

### Examples

```r
#========================================================================================
# 1) validation of this function
#========================================================================================
#
'x<-0.2
Phi(x)==stats::pnorm(x)
```

```r
#========================================================================================
# 1) Build the data
#========================================================================================
#
'a <- 0.1;
NX <- 222;
x <- runif(100,-11,11)
y <- Phi_inv(exp(a/NX)*Phi(x))-x
plot(x,y)
'a <- 0.1;
NX <- 222;
```
\begin{verbatim}
x <- runif(100,0,11)
y <- Phi_inv(exp(a/NX) *Phi(x))-x
plot(x,y)

a <- 0.1;
NX <- 222;
x <- runif(100,2,4)
y <- Phi_inv(exp(a/NX) *Phi(x))-x
plot(x,y)

a <- 0.01;
NX <- 222;
x <- runif(100,2,4);
y <- Phi_inv(exp(a/NX) *Phi(x))-x
plot(x,y)

a <- 0.01;
NX <- 222;
x <- runif(100,3.5,4);
y <- Phi_inv(exp(a/NX) *Phi(x))-x
plot(x,y)
\end{verbatim}

\textbf{Phi_inv} \hspace{1cm} \textit{Inverse function of the Cumulative distribution function }\Phi(x)\textit{ of the Standard Gaussian. where }x\textit{ is a real number.}

\textbf{Description}

The author is confused \texttt{stats::qnorm()} with \texttt{stats::pnorm()} and thus he made this.

\textbf{Usage}

\texttt{Phi_inv(x)}

\textbf{Arguments}

\texttt{x} \hspace{1cm} A real. To be passed to the function \texttt{stats::qnorm()}

\textbf{Details}

In Stan file, it is \texttt{inv_Phi()} and not \texttt{inv_phi}.

Since \( \Phi(x) \) is monotonic, it follows that \( \frac{d}{dx} \Phi^{-1} = \left( \frac{d}{dx} \Phi \right)^{-1} > 0 \), and thus \( \Phi^{-1}(x) \) is also monotonic.
Value
A real number: $\Phi^{-1}(x)$

See Also
Phi(), inv_Phi()

Examples

```r
x <- runif(100)

Phi_inv(x) == stats::qnorm(x)

inv_Phi(x) == stats::qnorm(x)
```

A generic function plot()

Description
A generic function plot()

Usage
```r
## S4 method for signature 'stanfitExtended,missing'
plot(x, y, ...)
```

Arguments
- `x` An R object of the S4 class `stanfitExtended`
- `y` An R object of the S4 class `missing-class`
- `...` Additional arguments
**plotFROC**

*Draw FROC curves by two parameters a and b*

### Description

Plot FROC curves based on two parameters a and b.

### Usage

```r
plotFROC(
  a,
  b,
  mesh.for.drawing.curve = 10000,
  upper_x = 1,
  upper_y = 1,
  lower_y = 0
)
```

### Arguments

- **a**: An arbitrary real number. It is no need to require any assumption, but I use such as \( a = \mu / \sigma \), where \( \mu \) is a mean of signal distribution and \( \sigma \) is its standard deviation in the bi-normal assumption.

- **b**: An arbitrary positive real number. I use such as \( b = 1 / \sigma \), where \( \sigma \) is a standard deviation of signal distribution in the bi-normal assumption.

- **mesh.for.drawing.curve**: A positive large integer, indicating number of dots drawing the curves, Default =10000.

- **upper_x**: A positive real number, indicating the frame size of drawing picture.

- **upper_y**: A positive real number, indicating the frame size of drawing picture.

- **lower_y**: A positive real number, indicating the frame size of drawing picture.

### Details

FROC curve is the alternative notion of ROC curve in signal detection theory.

The definition of FROC curve is

\[
(x(t), y(t)) = (t, 1 - \Phi(b \cdot \Phi^{-1}(e^{\exp(-t)}) - a))
\]

where, \( \Phi() \) is the cumulative distribution function of the standard Gaussian distribution and \( \Phi^{-1}() \) is its inverse mapping.

Revised 2019 NOv 27
Examples

dark_theme()

plotFROC(0.1, 0.2)

Description

Draws FROC curve and signal and noise (noise distribution is the differential of the logarithmic of the cumulative standard Gaussian denoted by $d \log \Phi$) are drawn in a same plain. The author of this package developed the FROC theory, and find that the noise distribution is not the so-called bi normal assumption. But instead, we use the differential logarithmic Gaussian for the noise distribution.

*Note that MRMC data is not allowed.*

Usage

plot_curve_and_hit_rate_and_false_rate_simultaneously(StanS4class)

Arguments

StanS4class  
An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC().
It can be passed to DrawCurves(), ppp() and ... etc

Details

This function is made to pass this plot to Shiny.

With pain from all my body, but today 2019 July 23 is good. Neuralgia or muscle aches makes my feeling down and down. If I can transform into Anpanman, then I want to give my head.

I fails, this is very small plot, so I cannot use this function for my package. I will remove this function or extend plot region for more confortable exhibition.

Value

None
plot_curve_and_hit_rate_and_false_rate_simultaneously

See Also

DrawCurves
draw_latent_noise_distribution

Examples

```r
## Not run:

#========================================================================================
# 1) Build the data
#========================================================================================
# For singler reader and single modality case.

dat <- list(c=c(3,2,1), #Confidence level. Note that c is ignored.
            h=c(97,32,31), #Number of hits for each confidence level
            f=c(1,14,74), #Number of false alarms for each confidence level
            NL=259, #Number of lesions
            NI=57,  #Number of images
            C=3)    #Number of confidence level

# where,
# c denotes confidence level, i.e., rating of reader.
# 3 = Definitely deseased,
# 2 = subtle... deseased
# 1 = very subtle
# h denotes number of hits (True Positives: TP) for each confidence level,
# f denotes number of false alarms (False Positives: FP) for each confidence level,
# NL denotes number of lesions,
# NI denotes number of images,

# For example, in the above example data,
# the number of hits with confidence level 3 is 97,
# the number of hits with confidence level 2 is 32,
# the number of hits with confidence level 1 is 31,

# the number of false alarms with confidence level 3 is 1,
# the number of false alarms with confidence level 2 is 14,
# the number of false alarms with confidence level 1 is 74,

#========================================================================================
# 2) Fit a model to the above data-set
#========================================================================================
```

Since dataset named dat are a single reader and a single modality, the function build the such model by running the following code.

```r
fit <- BayesianFROC::fit_Bayesian_FROC(
  dat, # dataset
  ite=1111, #To run in time <5s.
  cha=1, # number of chains, it is better more large.
)
```

3) Draw the FROC curve and signal and noise (logarithmic Gaussian)

Using fitted model object of class stanfitExtended, we can draw curves.

```r
plot_curve_and_hit_rate_and_false_rate_simultaneously(fit)
```

## End(Not run)

---

**plot_empirical_FROC_curves**

*Plot empirical FROC Curves by traditional ways of ggplot2*

---

**Description**

Plot empirical FROC Curves.

**Usage**

```r
plot_empirical_FROC_curves(
  dataList.MRMC,
  ModifiedPoisson = FALSE,
  colored_by_modality = TRUE,
  numbered_by_modality = TRUE,
  cex = 1.3,
)```
modalityID = c(1, dataList.MRMC$M),
readerID = c(1, dataList.MRMC$Q)
)

Arguments

dataList.MRMC  A list, indicating FROC data of MRMC. See also dataList which is a variable of the function fit_Bayesian_FROC().

ModifiedPoisson  Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.

Similarly,

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignette, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand,

if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the *False Positive Fraction (FPF) per image*.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\(\text{ModifiedPoisson} = \text{FALSE}\)) or as the expected pairs of FPF per image and TPF per lesion (\(\text{ModifiedPoisson} = \text{TRUE}\)).

If \(\text{ModifiedPoisson} = \text{TRUE}\), then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \(\text{ModifiedPoisson} = \text{FALSE}\), then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \(\text{ModifiedPoisson} = \text{TRUE}\) or \(\text{FALSE}\). In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \(\text{ModifiedPoisson} = \text{TRUE}\) or \(\text{FALSE}\). This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

colored_by_modality
A logical, if TRUE, then the color in the scatter plot means modality ID. If not, then the each color in the scatter plot indicates reader ID.

numbered_by_modality
A logical, if TRUE, then the number in the scatter plot means modality ID. If not, then the each number in the scatter plot indicates reader ID.

cex
A positive real number, specifying the size of dots in the resulting plot.

modalityID
A vector of integer, specifying modality ID to be drawn.

readerID
A vector of integer, specifying modality ID to be drawn.
plot_empirical_FROC_curves

Value

An object made by ggplot2, I am not sure what it is.

Examples

#========================================================================================
# The 1-st example
#========================================================================================

plot_empirical_FROC_curves(dd,readerID = 1:4,modalityID = 1:5)
plot_empirical_FROC_curves(dd,readerID = 1,modalityID = c(4,3))
plot_empirical_FROC_curves(dd,readerID = 2,modalityID = c(4,3))
plot_empirical_FROC_curves(dd,readerID = 3,modalityID = c(4,3))
plot_empirical_FROC_curves(dd,readerID = 4,modalityID = c(4,3))

#========================================================================================
# The example
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=2,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=2,Q=37)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)

plot_empirical_FROC_curves(d,readerID = 1:14,modalityID = 1:2)
plot_empirical_FROC_curves(d,readerID = 1:24,modalityID = 1:2)
plot_empirical_FROC_curves(d,readerID = 1:34,modalityID = 1:2)

#========================================================================================
# The example
#========================================================================================

v <- v_truth_creator_for_many_readers_MRMC_data(M=2,Q=7)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=2,Q=7)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)
plot_FPF_and_TPF_from_a_dataset

Plot FPF and TPF from MRMC data

Description

From data (srsc or MRMC), empirical FROC is plotted, namely FPF and TPF.

Usage

plot_FPF_and_TPF_from_a_dataset(dataList, ModifiedPoisson = FALSE)
Arguments

dataList

A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The dataList will be passed to the function rstan::sampling() in rstan. This is a variable in the function rstan::sampling() in which it is named data.

For the single reader and a single modality data, the dataList is made by the following manner:

dataList.Example <- list(
  h = c(41,22,14,8,1), # number of hits for each confidence level
  f = c(1,2,5,11,13), # number of false alarms for each confidence level
  NL = 124, # number of lesions (signals)
  NI = 63, # number of images (trials)
  C = 5)  # number of confidence, .. the author thinks it can be calculated as the length of h or f ...? ha,why I included this. ha .. should be omitted.

And using this object dataList.Example, we can apply fit_Bayesian_FROC() such as fit_Bayesian_FROC(dataList.Example).

To make this R object dataList representing FROC data, this package provides three functions:

convertFromJafroc() If data is a JAFROC xlsx formulation.
dataset_creator_new_version() Enter TP and FP data by table.
create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

A Single reader and a single modality (SRSC) case.

In a single reader and a single modality case (srsc), dataList is a list consisting of f, h, NL, NI, C where f, h are numeric vectors and NL, NI, C are positive integers.

f Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
h Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

NL A positive integer, representing Number of Lesions.
NI A positive integer, representing Number of Images.
C A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. ’Note that the maximal number of confidence level, denoted by C, are included, however, Note that confidence level vector c should not be specified. If specified,
will be ignored, since it is created by `c <- c(rep(C:1))` in the program and do not refer from user input data, where C is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

**data Format:**

A single reader and a single modality case

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td><code>c[1] = 5</code></td>
<td><code>h[1] = H_1 = 41</code></td>
</tr>
<tr>
<td>probably present</td>
<td><code>f[1] = F_1 = 1</code></td>
<td></td>
</tr>
<tr>
<td>subtle</td>
<td><code>f[2] = F_2 = 2</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>f[5] = F_5 = 13</code></td>
<td></td>
</tr>
</tbody>
</table>

* `false alarms` = False Positives = FP
* `hits` = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored, since it is created by `c <- c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector `c <- c(rep(C:1))` via a table which can be displayed by the function `viewdata()`.

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function `fit_Bayesian_FROC()`, dataset represented by an R list object representing FROC data must contain components `m, q, c, h, f, NL, C, M, Q`.

C A positive integer, representing the **highest** number of confidence level, this is a scalar.

M A positive integer vector, representing the number of **modalities**.

Q A positive integer, representing the number of **readers**.
A vector of positive integers, representing the modality ID vector.

q A vector of positive integers, representing the reader ID vector.

c A vector of positive integers, representing the confidence level. This vector must be made by `rep(rep(C:1),M*Q)`

h A vector of non-negative integers, representing the number of hits.

f A vector of non-negative integers, representing the number of false alarms.

NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector, the function `viewdata()` shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>29</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>24</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>66</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>40</td>
<td>44</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP

*hits = True Positives = TP

**ModifiedPoisson**

Logical, that is `TRUE` or `FALSE`.

If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image. For more details, see the author’s paper in which I explained per image and per lesion. (For details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_4 + F_5}{N_L},
\]
\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]
\[
\frac{F_5}{N_I},
\]
where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the *False Positive Fraction (FPF) per image*. The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{FALSE} \)) or as the expected pairs of FPF per image and TPF per lesion (\( \text{ModifiedPoisson} = \text{TRUE} \)).

If \( \text{ModifiedPoisson} = \text{TRUE} \), then FROC curve means the expected pair of FPF per lesion and TPF. On the other hand, if \( \text{ModifiedPoisson} = \text{FALSE} \), then FROC curve means the expected pair of FPF per image and TPF. So, data of FPF and TPF are changed thus, a fitted model is also changed whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \( \text{ModifiedPoisson} = \text{TRUE} \) or \( \text{FALSE} \). This rigidity of curves means that the number of images is a redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

**Value**

TPF and FPF

**See Also**

draw.CFP.CTP.from.dataList

**Examples**

```r
#======================================================================================
# srsc
#======================================================================================
# FPF is Per image

plot_FPF_and_TPF_from_a_dataset(d)

#======================================================================================
# MRMC
#======================================================================================
# FPF is Per lesion
```
**Description**

Make a factor vector by which we plot FPF and TPF.

**Usage**

```r
plot_FPF_TPF_via_dataframe_with_split_factor(
  dataList.MRMC,
  ModifiedPoisson = FALSE,
  colored_by_modality = TRUE,
  numbered_by_modality = TRUE,
  cex = 1.3
)
```

**Arguments**

- `dataList.MRMC`: A list, indicating FROC data of MRMC. See also `dataList` which is a variable of the function `fit_Bayesian_FROC()`.
- `ModifiedPoisson`: Logical, that is TRUE or FALSE. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion. Similarly, if `ModifiedPoisson = TRUE`, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image. For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If `ModifiedPoisson = TRUE`, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_3 + F_4 + F_5}{N_L},
\]
\[
\frac{F_4 + F_5}{N_L},
\]
\[
\frac{F_5}{N_L},
\]

where \( N_L \) is a number of lesions (signal). To emphasize its denominator \( N_L \), we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand, if \texttt{ModifiedPoisson = FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]
\[
\frac{F_5}{N_I},
\]

where \( N_I \) is the number of images (trial). To emphasize its denominator \( N_I \), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be ragraded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE}) or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE})

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textit{per lesion} and TPF.
On the other hand, if ModifiedPoisson = FALSE, then FROC curve means the expected pair of FPF per image and TPF. So, data of FPF and TPF are changed thus, a fitted model is also changed whether ModifiedPoisson = TRUE or FALSE. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether ModifiedPoisson = TRUE or FALSE. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

colored_by_modality
A logical, if TRUE, then the color in the scatter plot means modality ID. If not, then the each color in the scatter plot indicates reader ID.

numbered_by_modality
A logical, if TRUE, then the number in the scatter plot means modality ID. If not, then the each number in the scatter plot indicates reader ID.

cex
A positive real number, specifying the size of dots in the resulting plot.

Value

A dataframe, which is added TPF and FPF, etc into dataList.MRMC.

**Added Vectors as Contents of the Data-frame**

**CFP** A vector of *Cumulative False Positive*

**CTP** A vector of *Cumulative True Positive*

**TPF** A vector of *True Positive Fraction*

**FPF** A vector of *False Positive Fraction* per image or per lesion according to the logical variable ModifiedPoisson

**factor** What this means is trivial.

**Vectors as Contents of the Data-frame** dataList.MRMC

**c** A vector of positive integers, representing the **confidence level**. This vector must be made by rep(rep(C:1),M*Q)

**m** A vector of positive integers, representing the **modality** ID vector.

**q** A vector of positive integers, representing the **reader** ID vector.

**h** A vector of non-negative integers, representing the number of **hits**.

**f** A vector of non-negative integers, representing the number of **false alarm**.

Examples

```
#========================================================================================
# The 1st example
```
v <- v_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=1,Q=37)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = TRUE,
numbered_by_modality = TRUE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = FALSE,
numbered_by_modality = TRUE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = TRUE,
numbered_by_modality = FALSE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = FALSE,
numbered_by_modality = FALSE)

v <- v_truth_creator_for_many_readers_MRMC_data(M=2,Q=37)
m <- mu_truth_creator_for_many_readers_MRMC_data(M=2,Q=37)
d <- create_dataList_MRMC(mu.truth = m,v.truth = v)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = TRUE,
numbered_by_modality = TRUE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = FALSE,
numbered_by_modality = TRUE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = TRUE,
numbered_by_modality = FALSE)

plot_FPF_TPF_via_dataframe_with_split_factor(d,
colored_by_modality = FALSE,
numbered_by_modality = FALSE)
v <- v_truth_creator_for_many_readers_MRMC_data(M=3,Q=7)
mu <- mu_truth_creator_for_many_readers_MRMC_data(M=3,Q=7)
d <- create_dataList_MRMC(mu.truth = mu, v.truth = v)

plot_FPF_TPF_via_dataframe_with_split_factor(d, colored_by_modality = TRUE, numbered_by_modality = TRUE)
plot_FPF_TPF_via_dataframe_with_split_factor(d, colored_by_modality = FALSE, numbered_by_modality = TRUE)
plot_FPF_TPF_via_dataframe_with_split_factor(d, colored_by_modality = TRUE, numbered_by_modality = FALSE)
plot_FPF_TPF_via_dataframe_with_split_factor(d, colored_by_modality = FALSE, numbered_by_modality = FALSE)

# The 4th example

plot_FPF_TPF_via_dataframe_with_split_factor(dlist.MRMC = dd, colored_by_modality = TRUE, numbered_by_modality = TRUE)

# The 5th example

## Not run:

a <- plot_FPF_TPF_via_dataframe_with_split_factor(dd)

p <- ggplot2::ggplot(a, ggplot2::aes(FFP, TPF, group = factor(factor), colour = factor(m))) + ggplot2::geom_line(size = 1.4)
print(p)
# The 6th example

```r
a <- plot_FPF_TPF_via_dataframe_with_split_factor(dd, cex = 1.8)
```

# The 7th example

```r
# Plot empirical FROC curve whose modality is specified as following manner

a <- plot_FPF_TPF_via_dataframe_with_split_factor(dd)
aa <- a[a$m == c(2, 3),]
p <- ggplot2::ggplot(aa, ggplot2::aes(FPF, TPF, group = factor(factor),
                           colour = factor(m))) +
   ggplot2::geom_line(size = 1.4)
print(p)
```

```r
# Plot empirical FROC curve whose modality is specified as following manner

a <- plot_FPF_TPF_via_dataframe_with_split_factor(dd)
aa <- a[a$m %in% c(4, 3),]
p <- ggplot2::ggplot(aa, ggplot2::aes(FPF, TPF, group = factor(factor),
                           colour = factor(m))) +
   ggplot2::geom_line(size = 1.4)
print(p)
```

```r
# Plot empirical FROC curve whose modality is specified as following manner

a <- plot_FPF_TPF_via_dataframe_with_split_factor(dd)
aa <- a[a$m %in% c(3, 4),]
p <- ggplot2::ggplot(aa, ggplot2::aes(FPF, TPF, group = factor(factor),
                           colour = factor(m))) +
   ggplot2::geom_line(size = 1.4)
print(p)
```
Definition of a method for the inherited class stanfitExtended from stanfit

Description
This is a function for a method in the generic function plot.

Usage
plot_test(x)

Arguments
x
This is an object of an S4 class named stanfitExtended which is an inherited S4 class from the stanfit S4 class in the rstan package.

Description
The author is stupid, so he is confused pnorm() and qnorm().
Thu author always forget which is cumulative distribution of Gaussia, so I made this and this tells me which is mmy desired one. In this package, I often use Φ() for the standard Gaussian, and it is pnorm(). I am very confuse, since probability density has initial alphabet p, but pnorm() is not it.

Usage
pnorm_or_qnorm()
**Description**

PPP for chi square goodness of fit statistic.

**Usage**

```r
ppp(StanS4class, Colour = TRUE, dark_theme = TRUE, plot = TRUE, summary = TRUE)
```

**Arguments**

- `StanS4class`: An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc.
- `Colour`: Logical: TRUE or FALSE. Whether Colour of curves is dark theme or not.
- `dark_theme`: TRUE or FALSE
- `plot`: Logical, whether replicated datasets are drawn.
- `summary`: Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

**Details**

I hate the notion of p value and this is the motivation that I developed new FROC theory. However, I cannot overcome the traditional bitch. I hate statistics since p value is bitch, monotonically decreases when the sample size is large. In some papers, I forget the name, but in some papers, one pointed out that the frequentist p values precisely coincides some posterior probability of some event (I forget this but such as mean1 is greater than mean2).

In some suitable condition, I conjecture that Bayesian p value coincides to frequentist p value in some sense such as analytically or its expectation of a posterior or etc or large MCMC samples. So, p value is bitch and bitch and bitch. I emphasize that notion of p value is bitch and its background is unknown. In suitable condition, frequentist p value bitch is equal to a probability of some event measured by posterior. So,... Bayesian method cannot break the traditional frequentist bitch. Bayesian and frequentist are all bitch!! Of course, intuitively, it is good. But, the theoretically, it does not satisfies naturalist.

**Value**

A positive number between zero and one, indicating Posterior Predictive P value (PPP). In addition, it plots replicated datasets which are used to calculate a ppp.
## Not run:

```r
# The 1-st example: MRMC data
#========================================================================================
# 1) Fit a Model to MRMC Data
#========================================================================================

fit <- fit_Bayesian_FROC( ite = 111, dataList = ddd )

#========================================================================================
# 2) Evaluate Posterior Predictive P value for the Goodness of Fit
#========================================================================================

ppp(fit)
```

# If this quantity, namely a p value, is greater,
# then we may say that our goodness of fit is better. (accept the null hypothesis)
# In the traditional procedure, if p-value is less than 0.05 or 0.01 then we reject
# the null hypothesis that our model fit to data well.

# Of course, even if p-values is small, we should not ignore our result.
# P value bitch is not so clear what it does and in frequentist methods,
# we experienced p value is bitch with respect to sample size.
# So, in Bayesian context, this bitch might be bitch with respect to ...
# Anyway, but ha...many statisticians like this bitch.
# The 2-nd example uses data named d
#========================================================================================
# 1) Fit a Model to Data
#========================================================================================

fitt <- fit_Bayesian_FROC( ite = 111, dataList = d )

#========================================================================================
# 2) Evaluate Posterior Predictive P value for the Goodness of Fit
#========================================================================================

ppp(fitt)

# If this quantity is greater, then we may say that our model is better.
# I made this ppp at 2019 August 25.

#========================================================================================
# PPP is problematic
#========================================================================================

# Consider the dataset:

dat <- list(c=c(4,3,2,1), # Confidence level. Note that c is ignored.
            h=c(77,97,32,31), # Number of hits for each confidence level
            f=c(77,1,14,74), # Number of false alarms for each confidence level
            NL=259, # Number of lesions
            NI=57, # Number of images
            C=4) # Number of confidence level#

# Fit a model to the data

fit <- fit_Bayesian_FROC(dat)
# calculate p value

```r
ppp(fit)
```

# Then we can see that FPF and TPF are far from FROC curve, but p value is not so small, and thus in this case, ppp is not the desired one for us.

# In our model, we need monotonicity condition, namely
#  # However the above dataset is far from this condition, and it would relate the above undesired p value.
#  # Revised 2019 Sept 7
#  # Of course it is no need to satisfy this monotonicity precisely, but good data should satisfy.
#  # Since doctor should not wrong (false positive) diagnosis with his high confidence.

## End(Not run)

---

**Description**

PPP for chi square goodness of fit statistic

**Usage**

```r
ppp_MRMC(
```
StanS4class, 
summary = TRUE, 
replicate.number.from.model.for.each.MCMC.sample = 2 
)

Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc

summary       Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

replicate.number.from.model.for.each.MCMC.sample  A positive integer, representing $J$ in the following notation. Now, I think all I needed is love! tu tu tututu Love is all I need.
Suppose that

$$
\theta_1, \theta_2, \theta_3, ..., \theta_n
$$

is drawn from posterior $\pi(\theta|D)$ of given data $D$.
Let $y_1, y_2, ..., y_n$ be samples drawn from

$$
 y_1 \sim \text{likelihood}(\cdot | \theta_1), \\
 y_2 \sim \text{likelihood}(\cdot | \theta_2), \\
 y_3 \sim \text{likelihood}(\cdot | \theta_3), \\
 ..., \\
 y_n \sim \text{likelihood}(\cdot | \theta_n),
$$

Then the list of return values retains the following:

chisq_at_observed_data

$$
\chi(D|\theta_1), \chi(D|\theta_2), \chi(D|\theta_3), ..., \chi(D|\theta_n),
$$

chisq_not_at_observed_data

$$
\chi(y_1|\theta_1), \chi(y_2|\theta_2), \chi(y_3|\theta_3), ..., \chi(y_n|\theta_n),
$$

Logical  The i-th component is a logical vector indicating whether

$$
\chi(y_i|\theta_i) > \chi(D|\theta_i)
$$

is satisfied or not. Oppai ga Ippai. If TRUE, then the inequality holds.

p.value  From the component Logical, we calculate the so-called Posterior Predictive P value. Note that the author hate this notion!! I hate it!! Akkan Beeeee!!!
Details

The author hates the notion of p value and this is the motivation that he developed new theory without p values. However, he cannot overcome the traditional people. he loves mathematics, but he hates statistics. he emphasizes that notion of p value is dangerous (monotonicity w.r.t. sample size) and its background is unknown. Of course, intuitively, it is good. But, the theoretically, it does not ensure some criterion in large sample context.

So, p value said that my effort is rarely admissible, since its p value said that he is small for various datasets. So, this funcking p value said my effort is wrong, or should change model. Unfortunately, my hand aches cannot program more models. Ha,... why many peoply like p value bitch.

Value

A positive number indicates Posterior Predictive P value (ppp).

Examples

```
## Not run:
#========================================================================================
# 1) Fit a Model to MRMC Data
#========================================================================================

fit <- fit_Bayesian_FROC( ite = 111, dataList = ddd )

#========================================================================================
# 1) Evaluate Posterior Predictive P value for the Goodness of Fit
#========================================================================================

ppp_MRMC(fit)

# If this quantity is greater, then we may say that our model is better.

# I made this ppp at 2019 August 25.

## End(Not run)"
```
Description

Calculates Posterior Predictive P value for chi square (goodness of fit)

Usage

```r
ppp_srsc(
  StanS4class,
  Colour = TRUE,
  dark_theme = TRUE,
  plot = TRUE,
  summary = TRUE,
  replicate.number.from.model.for.each.MCMC.sample = 100
)
```

Arguments

StanS4class

An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`.

It can be passed to `DrawCurves()`, `ppp()` and ... etc

Colour

Logical: TRUE of FALSE. whether Colour of curves is dark theme or not.

dark_theme

TRUE or FALSE

plot

Logical, whether replicated data are drawn, in the following notation, replicated data are denoted by \( y_1, y_2, \ldots, y_N \).

summary

Logical: TRUE of FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

replicate.number.from.model.for.each.MCMC.sample

A positive integer, representing \( J \) in the following notation. Now, I think all I needed is love! tuu tuu tutu Love is all I need.

Suppose that

\[ \theta_1, \theta_2, \theta_3, \ldots, \theta_n \]

is drawn from posterior \( \pi(\theta|D) \) of given data \( D \).

Let \( y_1, y_2, \ldots, y_n \) be samples drawn from

\[ y_1 \sim \text{likelihood}(\cdot|\theta_1), \]
\[ y_2 \sim \text{likelihood}(\cdot|\theta_2), \]
\[ y_3 \sim \text{likelihood}(\cdot|\theta_3), \]
\[ \ldots, \]
\[ y_n \sim \text{likelihood}(\cdot|\theta_n), \]

Then the list of return values retains the following:

\[ \chi(D|\theta_1), \chi(D|\theta_2), \chi(D|\theta_3), \ldots, \chi(D|\theta_n), \]
chisq_not_at_observed_data

\[ \chi(y_1|\theta_1), \chi(y_2|\theta_2), \chi(y_3|\theta_3), \ldots, \chi(y_n|\theta_n), \]

Logical The i-th component is a logical vector indicating whether 
\[ \chi(y_2|\theta_2) > \chi(D|\theta_2) \]
is satisfied or not. Oppai ga Ippai. If TRUE, then the inequality holds.

p.value From the component Logical, we calculate the so-called Posterior Predictive P value. Note that the author hate this notion!! I hate it!! Akkan Beeeee!!

Details

In addition, this function plots replicated datasets from model at each MCMC sample generated by HMC. Using the Hamiltonian Monte Carlo Sampling: HMC, we can draw the MCMC samples of size \( n \), say 
\[ \theta_1, \theta_2, \theta_3, \ldots, \theta_n \]
, namely,
\[ \theta_1 \sim \pi(.,|D), \]
\[ \theta_2 \sim \pi(.,|D), \]
\[ \theta_3 \sim \pi(.,|D), \]
\[ \ldots, \]
\[ \theta_n \sim \pi(.,|D). \]

where \( \pi(\theta|D) \) is the posterior for given data \( D \).

Then, the function plots the following datasets \( y_1^1, y_2^1, \ldots, y_I^f \).

\[ \chi(y_1,1|\theta_1), \chi(y_1,2|\theta_1), \chi(y_1,3|\theta_1), \ldots, \chi(y_1,j|\theta_1), \chi(y_I,1|\theta_1), \ldots, \chi(y_I,j|\theta_I), \]
\[ \chi(y_2,1|\theta_2), \chi(y_2,2|\theta_2), \chi(y_2,3|\theta_2), \ldots, \chi(y_2,j|\theta_2), \chi(y_I,1|\theta_2), \ldots, \chi(y_I,j|\theta_I), \]
\[ \chi(y_3,1|\theta_3), \chi(y_3,2|\theta_3), \chi(y_3,3|\theta_3), \ldots, \chi(y_3,j|\theta_3), \chi(y_I,1|\theta_3), \ldots, \chi(y_I,j|\theta_I), \]
\[ \ldots, \]
\[ \chi(y_I,1|\theta_I), \chi(y_I,2|\theta_I), \chi(y_I,3|\theta_I), \ldots, \chi(y_I,j|\theta_I), \]

where \( L(.,|\theta_i) \) is a likelihood at parameter \( \theta_i \).

Let \( \chi(y|\theta) \) be a chi square goodness of fit statistics of our hierarchical Bayesian Model

\[ \chi(y|\theta) := \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{c=1}^{C} \left( \frac{(H_{c,m,r} - N_L \times p_{c,m,r})^2}{N_L \times p_{c,m,r}} + \frac{(F_{c,m,r} - (\lambda_c - \lambda_{c+1}) \times N_L)^2}{(\lambda_c - \lambda_{c+1}) \times N_L} \right). \]
and a chi square goodness of fit statistics of our non-hierarchical Bayesian Model

\[ \chi(y|\theta) := \sum_{c=1}^{C} \left( \frac{(H_c - N_L \times p_c)^2}{N_L \times p_c} + \frac{(F_c - (\lambda_c - \lambda_{c+1}) \times N_L)^2}{(\lambda_c - \lambda_{c+1}) \times N_L} \right) \]

where a dataset \( y \) denotes \((F_{c,m,r}, H_{c,m,r})\) in MRMC case and \((F_c, H_c)\) in a single reader and a single modality case, and model parameter \( \theta \).

Then we can calculate the posterior predictive \( p \) value for a given dataset \( y_0 \).

\[
\int \int I(\chi(y|\theta) > \chi(y_0|\theta)) f(y|\theta) \pi(\theta|y_0) d\theta dy \\
\approx \sum_{i=1}^{I} I(\chi(y_i|\theta_i) > \chi(y_0|\theta_i)) f(y_i|\theta_i) dy_i \\
\approx \sum_{j=1}^{J} \sum_{i=1}^{I} I(\chi(y_{i,j}|\theta_i) > \chi(y_0|\theta_i))
\]

When we plot these synthesized data-sets \( y_{i,j} \), we use the \text{jitter()} which adds a small amount of noise to avoid overlapping points. For example, \text{jitter(c(1,1,1,1))} returns values: 1.0161940 1.0175678 0.9862400 0.9986126, which is changed from 1,1,1,1 to be not exactly 1 by adding tiny errors to avoid overlapping. I love you. 2019 August 19 Nowadays, I cannot remove my self from some notion, such as honesty, or pain, or, maybe these thing is no longer with myself. This programm is made to fix previous release calculation. Now, this programm calculates correct \( p \) value.

So... I calculate the ppp for MCMC and Graphical User Interface based on Shiny for MRMC, which should be variable such as number of readers, modalities, to generate such ID vectors automatically. Ha,... tired! Boaring, I want to die...t, diet!! Tinko, tinko unko unko. Manko manko. ha.
Leberiya, he will be die, ha... he cannot overcome, very old, old guy. I will get back to meet him. Or I cannot meet him? Leberiya,...very wisdom guy, Ary you already die? I will get back with presents for you. Ball, I have to throgh ball, and he will catch it.

The reason why the author made the plot of data drawn from Posterior Predictive likelihoods with each MCMC parameters is to understand our programm is correct, that is, each drawing is very mixed. Ha,... when wright this,... I always think who read it. I love you, Ruikobach. Ruikobach is tiny and tiny, but,... cute. Ruikosan...Ruiko... But he has time only several years. He will die, he lives sufficiently so long, ha.

Using this function, user would get reliable posterior predictive \( p \) values. Cheers! Pretty Crowd!

We note that the calculation of posterior predictive \( p \) value (PPP) relies on the law of large number. Thus, in order to obtain the reliable PPP, we need to enough large MCMC samples to approximate the double integral of PPP. For example, the MCMC samples is small, then R hat is far from 1 but, the low MCMC samples leads us to incorrect \( p \) value which sometimes said that the model is correct even if the R hat criteria reject the MCMC results.

**Value**

A list, including \( p \) value and materials to calculate it.
Author(s)

Issei Tsunoda, Prof. of Curlbus University, Mosquitobus and Gostbus univ. also. My technique of catch mosquitos are excellent, so, I am a prof. ha., employ me. My health is bad, my life will be over.

Examples

```r
## Not run:

#========================================================================================
# 1) Create a fitted model object with data named "d"
#========================================================================================

fit <- fit_Bayesian_FROC( dataList = d,
                           ite = 222 # to restrict running time, but it is too small
)
#========================================================================================
# 2) Calculate p value and meta data
#========================================================================================

ppp <- ppp_srsc(fit)
#========================================================================================
# 3) Extract a p value
#========================================================================================

ppp$p.value

# Revised 2019 August 19
# Revised 2019 Nov 27

## End(Not run)
```
print.stanfitExtended-method

A method for a generic function print() for class "stanfitExtended"

Description

This is a method for print and stanfitExtended S4 class.

Usage

## S4 method for signature 'stanfitExtended'
print(x)

Arguments

x An S4 object of class stanfitExtended inherited from the class stanfit in the rstan package.

Examples

## Not run:
# How to use a new method for generic function "print".
#================================The First Example=================================

#(1)First, we prepare the example data from this package.

dat <- BayesianFROC::dataList.Chakra.1

# The R object named dat is a list which contains the hits and false alarms representing
# an FROC dataset. To confirm it, the function viewdata() can be used;

viewdata(dat)

#(2)Second, we run fit_Bayesian_FROC() in which the rstan::sampling() is implemented.
#Fit to data named "dat" the author's Bayesian model by

fit <- fit_Bayesian_FROC(dat)
#(3) Thirdly, we obtain the R object fit of S4 class named stanfitExtended that is an inherited class from the S4 class stanfit defined in the package rstan. For the S4 class stanfitExtended defined in this package, we can use the generic function print for this new S4 class.

```
print(fit)
```

# To use the generic function print() as a object of class "stanfit", we coerce class of fit into stanfit from stanfitExtended as follows;

```
fitt <- methods::as(fit,"stanfit")
```

# The R object "fitt" is a fitted model object of class stanfit, thus we can also apply the generic function print() as follows:

```
print(fitt)
```

#=========================================The Second Example=========================================

# (1) First, we prepare the example data from this package.
```
dat <- BayesianFROC::dataList.Chakra.Web
```

# (2) Second, we run fit_Bayesian_FROC() in which the rstan::sampling() is implemented. Fit to data named "dat" the author's Bayesian model by

```
fit <- fit_Bayesian_FROC(dat)
```

# (3) Thirdly, we obtain the R object fit of S4 class named stanfitExtended that is an inherited class from the S4 class stanfit defined in the package rstan.
# For the S4 class stanfitExtended defined in this package, we can use
# the generic function print for this new S4 class.

```r
print(fit)
```

# 2019.05.21 Revised.

```r
## End(Not run)# dotest
```

---

**print_minimal_reproducible_code_in_case_of_MRMC**

*Show minimal code in MRMC*

---

**Description**

Now 2020 March, it is available.

**Usage**

```r
print_minimal_reproducible_code_in_case_of_MRMC()
```

**Value**

`NULL`?

**Examples**

```r
print_minimal_reproducible_code_in_case_of_MRMC()
```
print_stanfitExtended

Definition of a method for the inherited class stanfitExtended from stanfit

Description

This is a function for a method for a generic function print() for class "stanfitExtended"

Usage

print_stanfitExtended(x)

Arguments

x

This is an R object of an S4 class named stanfitExtended inherited class from the stanfit in the rstan package.

Details

Print of stanfit has many parameters, but one of them, the AUC is the most important parameter. Thus in particular, we explain how to interpret the print out messages for AUCs.

——— Print of stanfit object ————————————————

* The AUC denoted by AA[modalityID,readerID] are shown by the function print() with a stanfit object.
* The column of 2.5% and 97.5% means the lower and upper bounds of the 95%
* For example, AA[2,3] means the AUC of the 2nd modality and the 3rd reader.

priorResearch

Research for Prior

Description

The autor investigates prior

Usage

priorResearch(z, m = 6, sd = 1, e = 0.01)

Arguments

z a real number, indicating θc.
m a real number, specifying the mean of signal Gaussian
sd a real number, specifying the standard deviation of signal Gaussian
e a positive real number, indicating ϵ.
prior_predictor

Value
A real, to investigate prior

\[
\mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_c - \mu}{\sigma})) - \Phi^{-1}(\Phi(\theta) \exp(\epsilon))
\]

where, \(m = \mu, \text{sd} = \sigma, z = \theta, e = \epsilon\).

Examples

```r
# From this plot, we can evaluate the minimum value of x such that the value is negative.

x <- runif(100, -1, 3)  # Synthesize 100 samples from Uniform(-1,3)
y <- priorResearch(x)
plot(x, y)
```

---

prior_predictor  
**Predict some estimates of parameter**

Description
Predict some estimates of parameter

Usage

```r
prior_predictor(d = d)
```

Arguments

d  
A list of data, which can be passed to the fit_Bayesian_FROC.

Value

none
prior_print_MRMC  Print What Prior Are Used

Description
Prints prior in R console

Usage
prior_print_MRMC(prior = 0)

Arguments
prior An integer, representing type of Prior

Value
none

Examples
prior_print_MRMC()

prior_print_srsc  Print What Prior Are Used

Description
Prints prior in R console

Usage
prior_print_srsc(prior = 0)

Arguments
prior An integer, representing type of Prior

Value
none

Examples
prior_print_srsc()
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit

| p_truth | Hit Rate: parameter of an MRMC model |

Description

A posterior mean of the model parameter for data ddd as an example of truth parameter.

Details

Hit Rate data of some MRMC data to use as a default value of the function hits_creator_from_rate. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

See Also

hits_creator_from_rate

p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit

P value for goodness of fit: No longer used in 2019 Oct

Description

Calculates the p value of the chi-squared test statistic for our model.

Get the Chi square values

\[ \chi(D_i|\theta_j) \]

for all possible pairs of synthesized data-sets \( D_1, D_2, \ldots, D_i, \ldots \) and MCMC samples \( \theta_1, \theta_2, \ldots, \theta_i, \ldots \).

Usage

```r
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(
  StanS4class,
  dig = 3,
  Colour = TRUE,
  plot.replicated.points = FALSE,
  head.only = FALSE,
  counter.plot.via.schatter.plot = TRUE,
  Show.table = TRUE
)
```
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit

Arguments

StanS4class  An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()`, and etc.

dig  To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `...??`. A positive integer representing the Significant digits, used in `stan` Cancellation. Default = 5.

Colour  Logical: TRUE or FALSE. Whether Colour of curves is dark theme or not.

plot.replicated.points  TRUE or FALSE. If true, then plot replicated points (hits, false alarms) by the scatter plot. This process will takes a long times. So if user has no time, then FALSE will help you.

head.only  Logical: TRUE or FALSE. Whether head part or entire of the table are shown. If TRUE, only head part are shown. Default is FALSE.

counter.plot.via.scatter.plot  Logical: TRUE or FALSE. Whether counter plot via scatter plot is drawn. Default = TRUE.

Show.table  Logical: TRUE or FALSE. Whether table includes the terms used calculation of p-value are shown.

Details

Here, we briefly review how to get the chi square samples in the Bayesian paradigm.

First, Let

\[ f(y|\theta) \]

be a model (likelihood) for a future data-set \( y \) and a model parameter \( \theta \). Let

\[ \pi(\theta|D) \]

be the posterior for given data \( D \). In this situation, the Hamiltonian Monte Carlo method is performed to obtain the MCMC samples of size \( N \). Denote MCMC samples by

\( \theta_1, \theta_2, \theta_3, \ldots, \theta_N \)

from posterior \( p(\theta|D) \) of given data \( D \). Alternatively, we get the sequence of models

\[ f(y|\theta_1), f(y|\theta_2), f(y|\theta_3), \ldots, f(y|\theta_N). \]

To get the samples

\[ y_1, y_2, \ldots, y_N \]

from the posterior predictive distribution, we merely draw the \( y_1, y_2, \ldots, y_N \) from \( f(y|\theta_1), f(y|\theta_2), f(y|\theta_3), \ldots, f(y|\theta_N) \), respectively. That is for all \( y_i \) is drawn from the distribution \( f(y|\theta_i) \). In notation, it may write;

\[ y_1 \sim f(\cdot|\theta_1) \]
\[ y_2 \sim f(\cdot|\theta_2) \]
\[ y_3 \sim f(.)|\theta_3) \]
\[ \ldots \]
\[ y_N \sim f(.)|\theta_1 N) \]

Once, we draw samples from the posterior predictive density, we can calculate an arbitrary integral with the posterior measure by the law of large number, or it is sometimes called Monte Carlo integral. Recall that the chi square goodness of fit statistics \( \chi \) is dependent of the model parameter \( \theta \) and data \( D \). that is,

\[ \chi = \chi(D|\theta). \]

Integrating \( \chi(D|\theta) \) with the posterior predictive measure, we get the

\[ \chi(D) \]

which depends only of the data \( D \), that is,

\[
p\text{value for data } D := \int I(\chi(Data|\theta) > \chi(D|\theta)) f(\theta|Data) \pi(\theta|D) d\theta d(Data)
\]

So, in the return value of this function is p value.

My hand, especially right has ache, so I quit this documentation, Good Luck, 2019 may 29. I do not have confidence whether my explanation success.

In this manner we get the two sequence of samples, one is from the posterior distribution and one is the posterior predictive distribution. Using these two kind of samples, we can calculate the test statistics as the Bayesian manner. That is, in frequentist method, the test statistics are calculated by the fixed model parameters, such as the maximal likelihood estimators. However, in Bayesian context, the parameter is not deterministic and hence we should calculate test statistics with the posterior measure. To accomplish this task, this package include the function.

**Value**

The main return is a nonnegative real number indicating p value of the Chi square goodness of fit. And the other components to calculate p values.

**See Also**

get_samples_from_Posterior_Predictive_distribution, chi_square_goodness_of_fit_from_input_all_param

**Examples**

```r
## Not run:
# First, fit the model to data. The number of sampling of the Hamiltonian Monte Carlo
# methods should be a little number, if user computer has low ability,
# since the calculation of the posterior predictive p values is heavy.

fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1, ite = 1111)
```
Next, extract the posterior predictive p value from the fitted model object "fit", and to do so, we have to make an object "output".

```
output <- p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit)
```

From the above R script, the table will appear in the R console. If the TRUE is more, then model fitting is better. Finally, we obtain the following p value;

```
p.value <- output$p.values.for.chisquare
```

The significant level of p value is 0.05 in frequentist paradigm, but, in this p value I think it should be more greater, and should use e.g., 0.6 instead of 0.05 for significant level. If significant level is 0.5, then test

```
p.value > 0.5
```

If it is FALSE, then the fitting is bad. If p value is more greater than the fitting is more better.

If user has no time, then plot.replicated.points=FALSE will help you. By setting FALSE, the replicated data from the posterior predictive distribution does not draw, and hence the running time of function become shorter.

```
TPs.FPs <- p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit, plot.replicated.points = FALSE)
```

If user want to use the scatter plots of hits and false alarms from the posterior predictive distribution for the submission, then the color plot is not appropriate. So, by setting the argument Colour = FALSE, the scatter plot become black and white. So, user can use this scatter plot for submission.

```
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit,Colour = FALSE)
```

Since p values are depend on data only, so it is better to show this dependency more explicitly as follows;

```
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit_Bayesian_FROC(dataList.High)
```

)
# Close the graphic device

`Close_all_graphic_devices()`

```
## End(Not run)# dottest
```

---

**rank_statistics_with_two_parameters**

*Rank Statistics*

**Description**

Rank Statistics

**Usage**

```r
rank_statistics_with_two_parameters(
  values.of.f.at.one.MCMC.samples,
  values.of.f.at.a.sample.from.priors
)
```

**Arguments**

- `values.of.f.at.one.MCMC.samples`
  - The value of f at a vector whose components are constructed by the all parameters at one MCMC sample.

- `values.of.f.at.a.sample.from.priors`
  - The value of f at a vector of model parameters from the prior distribution.

**Value**

The value of the Rank Statistics

**Examples**

```
## Not run:
#======== The first example ==========================================
rank_statistics_with_two_parameters(c(1,2,3,4,5),4)
#======= The Second Example ==========================================
a <- Draw_a_simulated_data_set_and_Draw_posterior_samples()
```
replicate_model_MRMC

Description

Replicate Models For Replicated Data From True Distributions.

Usage

replicate_model_MRMC(
  initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth,
  v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth,
  NI = 200,
  NL = 142,
  ModifiedPoisson = FALSE,
  replication.number = 2,
  summary = FALSE,
  ite = 1111
)

Arguments

initial.seed  The variable initial.seed is used to replicate datasets. That is, if you take initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, etc are for the first replication, the second replication, the third replication, etc. If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.

mu.truth     array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.

v.truth      array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.

z.truth      This is a parameter of the latent Gaussian assumption for the noise distribution.

NI           Number of Images.
replicate_model_MRMC

Number of Lesions.

Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated \textit{per lesion}, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF \textit{per lesion}.

Similarly, if ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated \textit{per image}, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF \textit{per image}.

For more details, see the author’s paper in which I explained \textit{per image} and \textit{per lesion}. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the \textit{False Positive Fraction (FPF)} is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_3 + F_4 + F_5}{N_L},
\]

\[
\frac{F_4 + F_5}{N_L},
\]

\[
\frac{F_5}{N_L},
\]

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand, if ModifiedPoisson = FALSE (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the \textit{False Positive Fraction (FPF) per image.}

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE})
or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE}).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textbf{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

\texttt{replication.number}

For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.

\texttt{summary}

Logical: \texttt{TRUE} of \texttt{FALSE}. Whether to print the verbose summary. If \texttt{TRUE} then verbose summary is printed in the \texttt{R} console. If \texttt{FALSE}, the output is minimal. I regret, this variable name should be verbose.

\texttt{ite}

To be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. An argument of \texttt{rstan::sampling()} in which it is named \texttt{iter}. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

\textbf{Value}

A list, each component is an S4 object of class \texttt{stanfitExtended}.

Revised 2019 Nov 7
## Examples

```r
# Plot FROC curves for a single model in the replicated models

list.of.fitted.model.objects <- replicate_model_MRMC(replication.number = 2)

DrawCurves(StanS4class = list.of.fitted.model.objects[[2]],
    modalityID = 1:list.of.fitted.model.objects[[2]]@dataList$M,
    readerID = 1:list.of.fitted.model.objects[[2]]@dataList$Q )

# Revised 2019 Sept 9
```

```r
## End(Not run)
```

---

**replicate_MRMC_dataList**

**MRMC: Replicates Datasets From Threshold, Mean and S.D.**

### Description

Make several datasets from a given model parameter.

### Usage

```r
replicate_MRMC_dataList(
    replication.number = 2,
    initial.seed = 123,
    mu.truth = BayesianFROC::mu_truth,
    v.truth = BayesianFROC::v_truth,
    z.truth = BayesianFROC::z_truth,
    NI = 200,
    NL = 142,
    ModifiedPoisson = TRUE,
    summary = FALSE
)
```

### Arguments

- **replication.number**
  
  A positive integer, specifying number of replicated datasets by this function. For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.
initial.seed  
The variable `initial.seed` is used to replicate datasets. That is, if you take `initial.seed = 1234`, then the seed 1234, 1235, 1236, 1237, 1238, ... etc are for the first replication, the second replication, the third replication, ... etc. If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.

`mu.truth`  
array of dimension (M,Q). Mean of the signal distribution of bi-normal assumption.

`v.truth`  
array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.

`z.truth`  
This is a parameter of the latent Gaussian assumption for the noise distribution.

`NI`  
Number of Images.

`NL`  
Number of Lesions.

`ModifiedPoisson`  
Logical, that is TRUE or FALSE.

If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.

Similarly,

If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If `ModifiedPoisson = TRUE`, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_4 + F_5}{N_L},$$

$$\frac{F_5}{N_L},$$

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.
On the other hand, if \texttt{ModifiedPoisson = FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_3 + F_4 + F_5}{N_I},
\]

\[
\frac{F_4 + F_5}{N_I},
\]

\[
\frac{F_5}{N_I},
\]

where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be degraded as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = FALSE})

or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE})

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textbf{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

\textbf{summary}

Logical: \texttt{TRUE} of \texttt{FALSE}. Whether to print the verbose summary. If \texttt{TRUE} then verbose summary is printed in the \texttt{R} console. If \texttt{FALSE}, the output is minimal. I regret, this variable name should be \texttt{verbose}.

\textbf{Value}

A list, each component is also a list, representing an FROC dataset.
Examples

```r
# Replicates datasets from a model with user specified parameters (now, it is default).
# a <- replicate_MRMC_dataList()

# Calculates FPF and TPF and plot it for the first replicate dataset
plot_FPF_and_TPF_from_a_dataset(a[[1]])

# Calculates FPF and TPF and plot it for the second replicate dataset
plot_FPF_and_TPF_from_a_dataset(a[[2]])

# Revised 2019 Oct 9
```

---

**sbcc**

**SBC**

Description

Priors should guarantee suitable conditions such that the ...

Usage

```r
sbcc(stanmodel, data, M, iter, refresh)
```

Arguments

- `stanmodel` see ?sbc
- `data` To specify priors.
- `M` The number of samples for rank statistics
- `iter` MCMC iterations
- `refresh` ????

Value

```r
```
Author(s)

Some Stan developer, I am not sure,..., who?

Examples

```r
## Not run:
stanModel <- stan_model_of_sbc()

Simulation_Based_Calibration_single_reader_single_modality_via_rstan_sbc(
  ite = 233,
  M = 11,
  epsilon = 0.04,
  stanModel = stanModel
)

## End(Not run)# dontrun
```

seq_array_ind

Makes a Matrix from a vector of integers

Description

To make sbc funtion

Usage

```r
seq_array_ind(d, col_major = FALSE)
```

Arguments

- `d`: A vector of integers
- `col_major`: A logical, whether..... ?

Value

A matrix, dimension is `prod(d)` times `length(d)`.

Author(s)

Some Stan developer, I am not sure,..., who?
Examples

```r
a <- seq_array_ind(1:3, col_major = TRUE)
#> a
#>     [,1] [,2] [,3]
#> [1,]  1  1  1
#> [2,]  1  2  1
#> [3,]  1  1  2
#> [4,]  1  2  2
#> [5,]  1  1  3
#> [6,]  1  2  3
```

```r
b <- seq_array_ind(1:3, col_major = FALSE)
```

_showGM

Show the Graphical Model for the case of a single reader and a single modality

Description

This function shows the graphical model for a single reader and a single modality FROC statistical model.

Usage

```r
showGM()
```

Examples

```r
## Not run:
showGM()
## End(Not run)# dontrun
```
Simulation_Based_Calibration_histogram

**Description**

Show R codes used in my manuscript

**Usage**

```
show_codes_in_my_manuscript()
```

**Value**

NULL

**Examples**

```
# R codes in my manuscript

show_codes_in_my_manuscript()
```

---

Simulation_Based_Calibration_histogram

*Draw a histogram of the rank statistics*

**Description**

To validate that the MCMC procedure is correct or not, we show the histogram of rank statistics. If the resulting histogram is uniformly distributed, then we can conclude that the MCMC sampling is correct. If the histogram is far from uniformity, then the MCMC sampling or specification of priors is not correct or not appropriate.

**Usage**

```
Simulation_Based_Calibration_histogram(
  N = 3,
  sd = 5,
  C = 5,
  initial.seed.for.drawing.a.rank.statistics = 1234567,
  fun = stats::var,
  NI = 259,
  NL = 259,
)```
initial.seed.for.drawing.a.data = 1234,
ModifiedPoisson = FALSE,
ite = 1111,
DrawCurve = FALSE
)

Arguments

N samples size of the rank statistics.

sd Standard Deviation of priors

C No. of Confidence levels

initial.seed.for.drawing.a.rank.statistics

seed

fun An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.

NI No. of images

NL No. of Lesions

initial.seed.for.drawing.a.data

seed

ModifiedPoisson Logical, that is TRUE or FALSE.

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.

Similarly,

If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)

If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L}.$$
\[
\frac{F_4 + F_5}{N_L},
\]
\[
\frac{F_5}{N_L},
\]
where \(N_L\) is a number of lesions (signal). To emphasize its denominator \(N_L\), we also call it the \textit{False Positive Fraction (FPF) per lesion}.

On the other hand, if \texttt{ModifiedPoisson = FALSE} (Default), then \textit{False Positive Fraction (FPF)} is given by

\[
\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_2 + F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_3 + F_4 + F_5}{N_I},
\]
\[
\frac{F_4 + F_5}{N_I},
\]
\[
\frac{F_5}{N_I},
\]
where \(N_I\) is the number of images (trial). To emphasize its denominator \(N_I\), we also call it the \textit{False Positive Fraction (FPF) per image}.

The model is fitted so that the estimated FROC curve can be ragraded as the expected pairs of FPF per image and TPF per lesion \(\texttt{ModifiedPoisson = FALSE}\)

or as the expected pairs of FPF per image and TPF per lesion \(\texttt{ModifiedPoisson = TRUE}\).

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF per lesion and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF per image and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28
**itable**

To be passed to the function `rstan::sampling()` in *rstan*. An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

**DrawCurve**

Logical: TRUE or FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set `DrawCurve` = TRUE, if not then `DrawCurve` = FALSE. The reason why the author make this variable `DrawCurve` is that it takes long time in MRMC case to draw curves, and thus Default value is FALSE in the case of MRMC data.

**Value**

samples of rank statistics

**Examples**

```r
## Not run:
g <- Simulation_Based_Calibration_histogram(N=2,ite = 2222)
graphics::hist(g$rank.statistics)

g <- Simulation_Based_Calibration_histogram(
   NI=1111111,
   NL=1111111,
   # N =100 would be better more than N =10 
   # But this is only example, we take very small N 
   N=10,
   ite=3333,
   sd=1,
   initial.seed.for.drawing.a.rank.statistics = 123456789,
   DrawCurve = TRUE
)

g <- Simulation_Based_Calibration_histogram(
   NI=1111111,
   NL=1111111,
   # N =100 would be better more than N =10 
   # But this is only example, we take very small N 
   N=10,
   ite=3333,
   sd=1,initial.seed.for.drawing.a.rank.statistics = 123456789,
   DrawCurve = TRUE,
   C=11)
```

# The Second Example: ---------------------------------------------

# If you want to see the replicated data, then the following code is available.
In the following, I extract the dataset which is very small rank statistics, e.g. less than 10. And draw the CFP and CTP for observation of dataset.

```r
gggg <- Simulation_Based_Calibration_histogram(
  NI=1111111,
  NL=1111111,
  N=22,
  ite=2222)

a <- gggg$rank.statistics<10
aa <- the_row_number_of_logical_vector(a)

draw.CFP.CTP.from.dataList(gggg$fit.list[[ aa[1] ]])
```

## End(Not run)##

dontrun

---

**Simulation Based Calibration (SBC) for a single reader and a single modality case**

**Description**

Implements the SBC algorithm for a single reader and a single modality case.

**Prior Under Construction**

I do not use the following prior, but instead the precise prior is defined in the file: sbcVer2.stan. I am tired and not want to write this.

For sufficiently small $\epsilon$,

\[
\epsilon < \bar{p}_c(\theta) < 1 - \epsilon,
\]

\[
g_c(\theta) > c\epsilon,
\]

namely

\[
\epsilon < \log \frac{\Phi(\theta_{c+1})}{\Phi(\theta_c)},
\]

\[
\epsilon < \Phi\left(\frac{\theta_{c+1} - \mu}{\sigma}\right) - \Phi\left(\frac{\theta_c - \mu}{\sigma}\right) < 1 - \epsilon.
\]

We have to consider this equation.
To satisfy the condition \( q_c(\theta) > c\epsilon \), we propose the following priors.

\[
\begin{align*}
\theta_1 & \sim Unif(-111, \Phi^{-1}(\exp^{-5\epsilon})), \\
\theta_2 & \sim Unif(\Phi^{-1}(\Phi(\theta_1) \exp^{\epsilon}), \Phi^{-1}(\exp^{-4\epsilon})), \\
\theta_3 & \sim Unif(\Phi^{-1}(\Phi(\theta_2) \exp^{\epsilon}), \Phi^{-1}(\exp^{-3\epsilon})), \\
\theta_4 & \sim Unif(\Phi^{-1}(\Phi(\theta_3) \exp^{\epsilon}), \Phi^{-1}(\exp^{-2\epsilon})), \\
\theta_5 & \sim Unif(\Phi^{-1}(\Phi(\theta_4) \exp^{\epsilon}), \Phi^{-1}(\exp^{-1\epsilon})).
\end{align*}
\]

To satisfy the condition \( \epsilon < p_c(\theta) < 1 - \epsilon \), we propose the following priors for more general condition \( f < p_c(\theta) < g \), where \( f \) and \( g \) are function of \( \epsilon, c \), e.g., \( f = \epsilon, g = 1 - \epsilon \).

\[
\begin{align*}
\theta_1 & \sim Unif(\phi^{-1}(1 - g), \phi^{-1}(1 - f)), \\
\theta_2 & \sim Unif(\phi^{-1}\left(\frac{\phi(\theta_1)}{1 - f}\right), \phi^{-1}\left(\frac{1 - g}{1 - f}\right)), \\
\theta_3 & \sim Unif(\phi^{-1}\left(\frac{\phi(\theta_2)}{1 - f}\right), \phi^{-1}\left(\frac{1 - g}{1 - f}\right)), \\
\theta_4 & \sim Unif(\phi^{-1}\left(\frac{\phi(\theta_3)}{1 - f}\right), \phi^{-1}\left(\frac{1 - g}{1 - f}\right)), \\
\theta_5 & \sim Unif(\phi^{-1}\left(\frac{\phi(\theta_4)}{1 - f}\right), \phi^{-1}\left(\frac{1 - g}{1 - f}\right)).
\end{align*}
\]

where \( \phi(\theta) := \Phi\left(\frac{\theta - \mu}{\sigma}\right) \) and \( \phi^{-1}(\tau) := \mu + \sigma \Phi^{-1}(\tau) \).

To show that the above equations are well-defined, we have to show
(1) the support of the above uniform distribution is not empty
(2) the condition \( q_c(\theta) > c\epsilon \) holds.

To show (1), we have to verify
\[
\Phi^{-1}(\exp^{-c\epsilon}) - \Phi^{-1}(\Phi(\theta_c) \exp^{\epsilon})
\]

Suppose that we obtain \( \theta_1, \theta_2, \ldots, \theta_c \) distributed by the above.

\[
\begin{align*}
\exp^{-(C+1-c)\epsilon} - \Phi(\theta_c) \exp^{\epsilon} \\
&> \exp^{-(C+1-c-1)\epsilon} - \exp^{(C+1-c)\epsilon} \exp^{\epsilon} \\
&> 0
\end{align*}
\]

Recall that the number of false alarms is distributed by Poisson with rate

\[
q_c(\theta) = \log \frac{\Phi(\theta_{c+1})}{\Phi(\theta_c)}
\]
Because $q_c(\theta)$ cannot be zero, but if we use non-informative priors for the model parameter $\theta$, then some synthesized parameter gives $q_c(\theta) = 0$ which causes undesired results in SBC.

Thus, for sufficiently small fixed $\epsilon$, we should assume that

$$q_c(\theta) > c\epsilon,$$

namely,

$$\epsilon < \log \frac{\Phi(\theta_{c+1})}{\Phi(\theta_c)},$$

from which

$$\Phi^{-1}(\Phi(\theta_c) \exp^{\epsilon}) < \theta_{c+1},$$

where we assume $\Phi(\theta_c) \exp^{\epsilon} < 1$, namely, $\theta_c < \Phi^{-1}(\exp^{-\epsilon}).$

These assumptions are necessary restriction for the equation $q_c(\theta) > \epsilon.$

Furthermore, we should consider the Bernoulli success rate for the number of hits. Next, recall that the number of hits is distributed by the binomial distribution of rate $p_c(\theta)$ which should be in between zero and one. However, non-informative prior cannot holds this condition. Thus, we should investigate the prior such that it restricts the hit rate to be in the interval $[0, 1].$

Recall that

$$p_c(\theta) = \Phi(\frac{\theta_{c+1} - \mu}{\sigma}) - \Phi(\frac{\theta_c - \mu}{\sigma}).$$

We have to assume

$$\epsilon < p_c(\theta) < 1 - \epsilon,$$

from which, we obtain

$$\epsilon < \Phi(\frac{\theta_{c+1} - \mu}{\sigma}) - \Phi(\frac{\theta_c - \mu}{\sigma}), < 1 - \epsilon$$

$$\epsilon + \Phi(\frac{\theta_c - \mu}{\sigma}) < \Phi(\frac{\theta_{c+1} - \mu}{\sigma}), < 1 - \epsilon + \Phi(\frac{\theta_c - \mu}{\sigma})$$

To go further step, we assume that

$$\Phi(\frac{\theta_c - \mu}{\sigma}) < \epsilon,$$
from which, we can apply $\Phi^{-1}$ to $1 - \epsilon + \Phi(\frac{\theta - \mu}{\sigma})$. So,

$$\frac{\theta - \mu}{\sigma} < \Phi^{-1}(\epsilon),$$

and thus

$$\theta < \mu + \sigma \Phi^{-1}(\epsilon).$$

$$\Phi^{-1}(\epsilon + \Phi(\frac{\theta - \mu}{\sigma})) < \frac{\theta + 1 - \mu}{\sigma} < \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta - \mu}{\sigma}))$$

$$\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta - \mu}{\sigma})) < \theta + 1 < \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta - \mu}{\sigma}))$$

To accomplish the above, we should assume that

$$\theta_{c+1} \sim \text{Uniform}(\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta - \mu}{\sigma})), \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta - \mu}{\sigma}))),$$

namely,

$$\theta_1 \sim \text{Unif}(-111, 111),$$

$$\theta_2 \sim \text{Unif}(\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta_1 - \mu}{\sigma})), \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_1 - \mu}{\sigma}))),$$

$$\theta_3 \sim \text{Unif}(\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta_2 - \mu}{\sigma})), \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_2 - \mu}{\sigma}))),$$

$$\theta_4 \sim \text{Unif}(\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta_3 - \mu}{\sigma})), \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_3 - \mu}{\sigma}))),$$

$$\theta_5 \sim \text{Unif}(\mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta_4 - \mu}{\sigma})), \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_4 - \mu}{\sigma}))),$$

Combining the necessary conditions of hit rates and false alarm rates, we should assume their intersections.

Set

$$X_c := \Phi^{-1}(\Phi(\theta_c) \exp'),$$

$$Y_c := \mu + \sigma \Phi^{-1}(\epsilon + \Phi(\frac{\theta_c - \mu}{\sigma}))$$

$$Z_c := \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_c - \mu}{\sigma})),$$

then,

$$\theta_1 \sim \text{Unif}(-111, 111),$$

$$\theta_2 \sim \text{Unif}(\max(X_1, Y_1), Z_1),$$

$$\theta_3 \sim \text{Unif}(\max(X_2, Y_2), Z_2),$$

$$\theta_4 \sim \text{Unif}(\max(X_3, Y_3), Z_3).$$
\[ \theta_5 \sim \text{Unif}(\max(X_4, Y_4), Z_4). \]

To justify these priors, we have to implement the SBC algorithm.

In the above uniform distribution, the support of them should not be empty. However it is not satisfied without any restriction. So, we should require the inequality that

\[ \Phi^{-1}(\Phi(\theta_c) \exp^\epsilon) < \mu + \sigma \Phi^{-1}(1 - \epsilon + \Phi(\frac{\theta_c - \mu}{\sigma})), \]

which is satisfied in sufficiently small \( \theta_c \) and the continuity of this equation implies that the set of solutions of \( \theta_c \) satifying the inequality is not empty. Thus we have to find the minimum of parameter \( \theta^*_c \) such that it satisfies the inequality.

```
Usage
Simulation_Based_Calibration_single_reader_single_modality_via_rstan_sbc(
    epsilon = 0.01,
    ite = 3333,
    NL = 259,
    NI = 57,
    C = 3,
    M = 500,
    BBB = 0.3,
    AAA = 3e-04,
    vvv = 0.3,
    vvvv = 11,
    mmm = 0,
    mmmm = 1,
    stanModel,
    sbc_from_rstan = TRUE
)
```

Arguments

- **epsilon** lower bound of Poisson for false positives.
- **ite** To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
- **NL** number of lesions
- **NI** number of images
- **C** number of confidence levels
- **M** To be passed to the function `rstan::sbc()` in `rstan`. 
- **BBB** a real
- **AAA** a real
Simulation Based Calibration single reader single modality via rstan sbc

vvv a real
vvvv a real
mm m a real
mmmm a real

stanModel An object of the class stanfit of sbc. This is for the package developer.
sbc_from_rstan A logical, whether rstan::sbc() is used

Details

The implementation is done using the rstan::sbc. The stan file is SBC.stan The implementation is done using the function rstan::sbc. The stan file is SBC.stan The variable in this function is a collection of parameters of priors

If we use non-informative prior, then from the prior the odd model parameter are synthesized. For example, If two thresholds z[c] and z[c+1] agree for some c, then the false alarm rate becomes zero with the following error from rstan::sbc:

failed to create the sampler; sampling not done

Error in new_CppObject_xp(fields$.module,fields$.pointer,...) :
Exception: poisson_rng: Rate parameter is 0, but must be > 0!

Thus, we have to use very strong prior to avoid to synthesize such odd parameters of model.

SBC is a validation algorithm for models with respect to its prior. I cannot fined the prior in which we can fit a model to various datasets.

What is SBC?

Aim of SBC is to evaluate how the computed posteriors are incorrect. To do so, SBC algorithm makes a histogram whose uniformity indicates MCMC samples contains bias.

For example,

If histogram is concave, namely there are spikes at the boundaries of histogram, then it indicates that MCMC samples is correlated. If a histogram is convex (∩-shaped), then it indicates that over-dispersed posteriors relative to the true posterior.

if histogram is concave, namely there are spikes at the boundaries of histogram, then it indicates that MCMC samples is correlated.

If a histogram is convex (∩-shaped), then it indicates that over-dispersed posteriors relative to the true posterior.

If a histogram is weighted to right or left, then posterior moves opposite direction, namely left or right respectively.

We may say that SBC is a statistical test of the null hypothesis $H_0$:

$$H_0 : MCMC\text{sampling is correct.}$$

If the histogram is far from uniformity, then we reject $H_0$ and say that MCMC sampling contains bias.

Parameters of our model
The first threshold

dz The difference of thresholds, that is, \(dz[c] := z[c+1] - z[c]\)

\(m\) Mean of signal Gaussian

\(v\) Standard deviation (Do not confuse it with Variance) of signal Gaussian

**Value**

A list of S3 class "sbc", which is an output of the function `rstan::sbc()` in `rstan`.

**References**


**Data Format:**

A single reader and a single modality case

<table>
<thead>
<tr>
<th>NI=63, NL=124</th>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c</td>
<td>f</td>
<td>h</td>
</tr>
</tbody>
</table>

Recall our model for the above data format:

\(H_5 \sim \text{Binomial}(p_5, N_L)\)

\(H_4 \sim \text{Binomial}(p_4, N_L)\)

\(H_3 \sim \text{Binomial}(p_3, N_L)\)

\(H_2 \sim \text{Binomial}(p_2, N_L)\)

\(H_1 \sim \text{Poisson}(p_1, N_L)\)

\(F_5 \sim \text{Poisson}(q_5)\)

\(F_4 \sim \text{Poisson}(q_4)\)

\(F_3 \sim \text{Poisson}(q_3)\)

\(F_2 \sim \text{Poisson}(q_2)\)

\(F_1 \sim \text{Poisson}(q_1)\)
where
\[
p_5 = p_5(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_5}^{\infty} \text{Gaussian}(z|\mu, \sigma) dz
\]
\[
p_4 = p_4(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_4}^{z_5} \text{Gaussian}(z|\mu, \sigma) dz
\]
\[
p_3 = p_3(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_3}^{z_4} \text{Gaussian}(z|\mu, \sigma) dz
\]
\[
p_2 = p_2(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_2}^{z_3} \text{Gaussian}(z|\mu, \sigma) dz
\]
\[
p_1 = p_1(z_1, \ldots, z_C; \mu, \sigma) = \int_{z_1}^{z_2} \text{Gaussian}(z|\mu, \sigma) dz
\]
\[
q_5 = q_5(z_1, \ldots, z_C) = \int_{z_5}^{\infty} d \log \Phi(z)
\]
\[
q_4 = q_4(z_1, \ldots, z_C) = \int_{z_4}^{z_5} d \log \Phi(z)
\]
\[
q_3 = q_3(z_1, \ldots, z_C) = \int_{z_3}^{z_4} d \log \Phi(z)
\]
\[
q_2 = q_2(z_1, \ldots, z_C) = \int_{z_2}^{z_3} d \log \Phi(z)
\]
\[
q_1 = q_1(z_1, \ldots, z_C) = \int_{z_1}^{z_2} d \log \Phi(z)
\]

**Priors**

\[z[c] \sim ?\]
\[m \sim ?\]
\[v \sim ?\]

In SBC, we have to specify proper priors, thus, we use the above priors. So, what reader should do is to specify the above parameters, that is, \(w, w, w, w, z, z, z, m, m, m, v, v, v\) and further a number of images \(N_I\) and a number of lesion \(N_J\) and a number of confidence levels should be specified. In the above example data format, the number of confidence level is the number of rows, and now it is 5, that is \(C=5\).

Revised 2019 August 4

I am not statistician nor researcher nor human. My leg is gotten by death who is prurigo nodularis. Death is soon. I cannot understand, I hate statistics. I do not want to waste my time to this FROC analysis. My program is volunteer, I am no money no supported. Completely my own support or my parents. Completely my own. I am tired for this no end point running. I have not money to research or place or circumstance. No healthy condition. This program is made with my blood and pain, great pain. I no longer want to live. I hate all. Honesty.
Simulation Based Calibration (SBC) for a single reader and a single modality case

Description

Implements the SBC algorithm for the a single reader and a single modality case.

Usage

```r
Simulation_Based_Calibration_via_rstan_sbc_MRMC(
  ww = -0.81,
  www = 0.001,
  mm = 0.65,
  mmm = 0.001,
  vv = 5.31,
  vvv = 0.001,
  zz = 1.55,
  zzz = 0.001,
  A_mean = 0.6,
  A_variance = 0.1,
  v_hyper_v = 0.05,
  v_hyper_v = 0.01,
  NL = 259,
  NI = 57,
  C = 3,
  M = 5,
  Q = 4
)
```

Arguments

- **ww**: A real number representing parameter of prior, indicating mean of prior for the first threshold
- **www**: A real number representing parameter of prior, variance of prior for the first threshold
- **mm**: A real number representing parameter of prior, mean of prior for the mean of signal distribution
- **mmm**: A real number representing parameter of prior, variance of prior for the variance of signal distribution
- **vv**: A real number representing parameter of prior, mean of prior for the mean of signal distribution
- **vvv**: A real number representing parameter of prior, variance of prior for the variance of signal distribution
zz A real number representing parameter of prior, mean of prior for the differences of thresholds

zzz A real number representing parameter of prior, variance of prior for the differences of thresholds

A_mean A real number representing parameter of prior, indicating mean of prior for the A

A_variance A real number representing parameter of prior, indicating mean of prior for the A

vv_hyper_v A real number representing parameter of prior, indicating mean of prior for the hyper_v

vvv_hyper_v A real number representing parameter of prior, indicating variance of prior for the hyper_v

NL number of lesions

NI number of images

C number of confidence levels

M number of modalities

Q number of readers

Details

The implementation is done using the rstan::sbc. The stan file is SBC.stan

Value

A list of S3 class "sbc", which is an outputs of the `sbc` function in rstan.

References


See Also

rstan::sbc, which implements SBC.

Stan file: SBC_MRMC.stan
size_of_return_value  Size of R object

Description

This return value can add each other or any number by the manner: return + number of R object

Usage

size_of_return_value(
  object,
  summary = TRUE,
  is_return_value = TRUE,
  base_size = 0,
  col = FALSE
)

Arguments

object  Any R object, whose size is measured.
summary  A logical, whether the result is printed.
is_return_value  A logical, printed word is used as "return value" if it is TRUE.
base_size  This value is added to the return value, namely, object size + base_size is the return value. This is for the package developer.
col  A logical, wheter print is colored.

Value

return value of utils::object.size()

small_margin  Margin

Description

If each variable is smaller, then the margin of it is smaller, so plot region become larger. But title and x axis title will be vanished.
**small_margin**

**Usage**

```r
small_margin(
  Down.oma = 1,
  Left.oma = 1,
  Top.oma = 1,
  Right.oma = 1,
  Down.mar = 1,
  Left.mar = 1,
  Top.mar = 1,
  Right.mar = 1
)
```

**Arguments**

- **Down.oma**: smaller gives larger plot region
- **Left.oma**: smaller gives larger plot region
- **Top.oma**: smaller gives larger plot region
- **Right.oma**: smaller gives larger plot region
- **Down.mar**: smaller gives larger plot region
- **Left.mar**: smaller gives larger plot region
- **Top.mar**: smaller gives larger plot region
- **Right.mar**: smaller gives larger plot region

**Details**

To show FROC curve or signal and noise distributions in Shiny Graphical devices, the author wrote down this function `small_margin`. By taking margin too small, we give more larger plot regions in Shiny Graphical devices. 2019 August 6

**Value**

NONE

**See Also**

- `draw_latent_signal_distribution`
- `draw_latent_noise_distribution`
- `DrawCurves`
- `DrawCurves_srsc`

**Examples**

```r
small_margin()
graphics::plot(1:3,1:3)
```
Description

Snippet for the package BayesianFROC. Copy and paste to the snippet edition tools in your R studio for the comfortable usage of the package BayesianFROC. This is under construction. To edit snippet, you should open the editor located in Tools > Global options > Code > Edit snippets.

Usage

snippet_for_BayesianFROC()

Value

nothing

Examples

snippet_for_BayesianFROC()
sortAUC

Prints a Ranking for AUCs for MRMC Data

Description

prints a modality ranking according to their AUCs.

Usage

sortAUC(StanS4class, digits = 3, simple = FALSE)

Arguments

StanS4class
An S4 object of class `stanfitExtended` which is an inherited class from the S4 class `stanfit`. This R object is a fitted model object as a return value of the function `fit_Bayesian_FROC()`. It can be passed to `DrawCurves()`, `ppp()` and ... etc

digits
To be passed to `round()` for AUC, to determine the significant digits of AUCs.

simple
Logical, `TRUE` or `FALSE`. If `TRUE`, then it is simple.

Details

This is a ranking. Sort a data-frame involving AUC and corresponding modality IDs.

Value

A data-frame, representing sorted ranking of modality ID and its AUC. Revised 2019 Sept 9

Examples

```r
## Not run:
#========================================================================================
# 1) Fit a model to an MRMC data-set named dd
#========================================================================================
fit <- fit_Bayesian_FROC(
    ite = 1111,
    summary = FALSE,
    cha = 1,
    dataList = dd
)

#========================================================================================
# 2) Sort the AUC and make a ranking table
```
sortAUC(fit)

# Then, a ranking table will appear.

# Revised 2019 Sept 9

## End(Not run)

---

stanfitExtended

stanfitExtended, an S4 class inherited from the S4 class stanfit

---

**Description**

Inherits from the class `stanfit` which is an S4 class defined in the package `rstan`:

**Details**


——— To read the table of R object of class `stanfit` in case of MRMC —————————

* The AUC denoted by AA[modalityID ,readerID] are shown.
  For example, AA[2,3] means the AUC of the 2nd modality and the 3rd reader.
* The column of 2.5% and 97.5% means the lower and upper bounds of the 95

**Slots**

- `plotdataMRMC` Plot data for MRMC case.
- `plotdata` This is a data frame with four components which is used to draw curves such as FROC curves and AFROC curves. So, this slot includes the component:
  ```r
  fit@plotdata$x.AFROC,
  fit@plotdata$y.AFROC,
  fit@plotdata$x.FROC,
  fit@plotdata$y.AFROC
  ```
  where `fit` is an object of class `stanfitExtended`.
  For example, we can use this slot such as `plot(fit@plotdata$x.AFROC, fit@plotdata$y.AFROC)`
  The author think this slot is not good because it increases the object size.
- `dataList` An FROC dataset. Using the dataset, the fitting has done.
studyDesign  A character, e.g., "srsc.per.image", "srsc.per.lesion", according to False Positive Fraction (FPF) is per image or per lesion.

metadata  An additional data calculated from dataList, such as cumulative hits and false alarms,...,etc.

WAIC  A WAIC calculated by the function waic.

convergence  A logical R object TRUE or FALSE. If TRUE, then it means your model is good in the R hat criterion.

PreciseLogLikelihood This is TRUE or FALSE. If TRUE, then target formulation is used in the .stan files. However, non-target formulation has warning for non-linear Jacobian issue. So, the author use target formulations for all .stan files, and thus this slot is now, redundant.

chisquare This is a chi square calculated with Expected A Posterior estimates, i.e., the posterior mean estimates. Chi square statistic is $\chi^2(\text{Data}|\theta)$, there are three simple ways to get it.

1. $\int \chi^2(\text{Data}|\theta)\pi(\theta|\text{Data})d\theta$
2. $\chi^2(\text{Data}|\int \theta\pi(\theta|\text{Data})d\theta)$
3. $\int \chi^2(\text{Data}|\theta)f(\text{Data}|\theta)\pi(\theta|\text{Data})d\theta$

where, $f(\text{Data}|\theta)$ denotes a likelihood and $\pi(\theta|\text{Data})$ is a posterior. This slot retains the (2)

See also ppp()

index  This is for programming phase.

Divergences This is the number of the divergence transitions in the MCMC simulation.

MCMC. Iterations  A MCMC iterations which does not count the burn-in period.

Divergence. rate  A divergence rate, that is the number of the divergence iterations over total MCMC iterations. Burn-in period is not included.

model_name  A slot of the stanfit which is an S4 class defined in the rstan package.

model_pars  A slot of the stanfit which is an S4 class in the package rstan.

par_dims  A slot of the stanfit which is an S4 class in the package rstan.

mode  A slot of the stanfit which is an S4 class in the package rstan.

sim  A slot of the stanfit which is an S4 class in the package rstan.

inits  A slot of the stanfit which is an S4 class in the package rstan.

stan_args  A slot of the stanfit which is an S4 class in the package rstan.

stanmodel  A slot of the stanfit which is an S4 class in the package rstan.

date  A slot of the stanfit which is an S4 class in the package rstan.

.MISC  A slot of the stanfit which is an S4 class in the package rstan.
Usage

Stan_code_validation(
  z = BayesianFROC::z,
  mu = BayesianFROC::mu,
  v = BayesianFROC::v,
  T.or.F = T
)

Arguments

z thresholds
mu mean
v standard deviation
T.or.F logical, if true hten a logical is return hit rate <1 and if false hit rate is returned.

Examples

Stan_code_validation(z=c(4.7,5,6),mu+555,v/1000000000)

Stan_code_validation(z=c(4.7,5,6),mu+5,v/10,T.or.F = FALSE)

#ppp[1,3,4]/denoo[1,3,4]

stan_model_of_sbc

Creates an object of class stanfit of SBC

Description

Creates an object of class stanfit of SBC

Usage

stan_model_of_sbc()

Value

An object of class stanfit for SBC

Examples

## Not run:
stan_model_of_sbc()

## End(Not run)
StatisticForANOVA

Description
Provides a statistic to test the null hypothesis that all modalities are same.

Usage
StatisticForANOVA()

Value
None

summarize_MRMC

Description
Summarize the estimates for MRMC case

Usage
summarize_MRMC(StanS4class, dig = 3)

Arguments
StanS4class An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc
dig To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

Value
Nothing
summary_AUC_comparison_MRMC

Print summary for AUC comparisons for MRMC

Description

It prints the results of AUC comparison for MRMC data.

Usage

summary_AUC_comparison_MRMC(StanS4class, significantLevel = 0.8, dig = 3)

Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC().

It can be passed to DrawCurves(), ppp() and ... etc

significantLevel  This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

dig  To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.
summary_AUC_comparison_MRMC_without_crayon

Print summary for AUC comparisons for MRMC without color

Description

It prints the results of AUC comparison for MRMC data.

Usage

summary_AUC_comparison_MRMC_without_crayon(
  StanS4class,
  significantLevel = 0.8,
  dig = 3
)

Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc

significantLevel  This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

dig  To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

summary_AUC_comparison_MRMC_with_crayon

Print summary for AUC comparisons for MRMC hier with color

Description

It prints the results of AUC comparison for MRMC data.

Usage

summary_AUC_comparison_MRMC_with_crayon(
  StanS4class,
  significantLevel = 0.8,
  dig = 3
)
Arguments

StanS4class  An S4 object of class stanfitExtended which is an inherited class from the S4 class stanfit. This R object is a fitted model object as a return value of the function fit_Bayesian_FROC(). It can be passed to DrawCurves(), ppp() and ... etc.

significantLevel  This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

dig  To be passed to the function rstan::sampling() in rstan. An argument of rstan::sampling() in which it is named ...??. A positive integer representing the Significant digits, used in stan Cancellation. Default = 5.

Value  The estimates

Examples

```r
### Not run:
#================The first example=======================================================
#1) Build the data for singler reader and single modality case.
```
Test_Null_Hypothesis_that_all_modalities_are_same

Test the Null hypothesis that all modalities are same

Description

Test null hypothesis that all modalities have same observer performance ability, using Bayes factor.

Usage

Test_Null_Hypothesis_that_all_modalities_are_same(
  dataList,
  ite = 1111,
  cha = 1,
  summary = FALSE
)
Arguments

- **dataList**: MRMC is the only case in which the function is available for this function.
- **ite**: To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
- **cha**: To be passed to the function `rstan::sampling()` in `rstan`. An argument of `rstan::sampling()` in which it is named `chains`. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.
- **summary**: Logical: TRUE or FALSE. Whether to print the verbose summary. If TRUE then verbose summary is printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Details

From input data (variable: dataList), the two objects of class `stanfit` are created. one is fitted to the null hypothesis model and the another one representing alternative hypothesis. These two `stanfit` objects are compared by the Bayes factor.

Value

- none

---

**the_row_number_of_logical_vector**

*Extract the row number from a logical vector*

Description

Extract the row number from a logical vector

Usage

```
the_row_number_of_logical_vector(vector.logical)
```

Arguments

- `vector.logical`: vector with logical component

Value

the row number of logical component
TRUE.Counter.in.vector

Author(s)
Issei Tsunoda

Examples

```r
a <- c(TRUE, FALSE, FALSE, TRUE, TRUE)

b <- the_row_number_of_logical_vector(a)
# Then, return value object, b is a vector of
#> b
# 1, 4, 5
# From this, we can count the TRUE, as following manner:

Number.of.TRUE <- length(b)
# Of course, it is:
#> Number.of.TRUE
# 3

length(b) == sum(a)
```

---

TRUE.Counter.in.vector

Count TRUE in a Vector whose components are all Logical R objects

Description
For the posterior predictive p value.

Usage
TRUE.Counter.in.vector(vector.logical)

Arguments
- `vector.logical` vector with logical component

Value
A positive integer.
Examples

```r
#========================================================================================
# Revised 2019 oct. This is same as sum(), I did not know this
#========================================================================================

a <-c(TRUE,FALSE,FALSE,TRUE,TRUE)
TRUE.Counter.in.vector(a)

# Of course, it is:
#> Number.of.TRUE
# 3

sum(a) == TRUE.Counter.in.vector(a)

# I did not know this equality,... no longer this function is needed
```

---

\( \nu \)

**Standard Deviation: parameter of an MRMC model**

**Description**

A posterior mean of the model parameter for data ddd as an example of truth parameter.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**See Also**

make_true_parameter_MRMC

---

**validation.dataset_srsc**

**Error between a give parameter and estimates for the parameters**

**Description**

Let us denote a model parameter by \( \theta_0 \), \( N_I \) by a number of images and number of lesions by \( N_L \), which are specified by user as the variables of the function.

1. **Replicates models for** \( D_1, D_2, \ldots, D_k, \ldots, D_K \).
2. **Draw a dataset** \( D_k \) **from a likelihood (model), namely** \( D_k \text{ likelihood}(\theta_0) \).
Draw a MCMC samples \( \{ \theta_i(D_k) \} \) from a posterior, namely \( \theta \sim \pi(\cdot | D_k) \).

Calculate a posterior mean, namely \( \bar{\theta}(D_k) := \sum \theta_i(D_k) \).

Calculate error for \( D_k \) \( \epsilon_k := \text{Truth} - \text{posterior mean estimates of } D_k = |\theta_0 - \bar{\theta}(D_k)| \) (or \( \theta_0 - \bar{\theta}(D_k) \)), according by the user specified absolute.errors.

(II) Calculate mean of errors \( \bar{\epsilon}(\theta_0, N_I, N_L) = \frac{1}{K} \sum \epsilon_k \)

Running this function, we can see that the error \( \bar{\epsilon}(\theta_0, N_I, N_L) \) decreases monotonically as a given number of images \( N_I \) or a given number of lesions \( N_L \) increases.

Also, the scale of error also will be found. Thus this function can show how our estimates are correct. Scale of error differs for each component of model parameters.

Revised 2019 August 28

Usage

validation.dataset_srsc(
  replicate.dataset = 3,
  ModifiedPoisson = FALSE,
  mean.truth = 0.6,
  sd.truth = 5.3,
  z.truth = c(-0.8, 0.7, 2.38),
  NL = 259,
  NI = 57,
  ite = 1111,
  cha = 1,
  summary = TRUE,
  serial.number = 1,
  base.size = 0,
  absolute.errors = TRUE
)

Arguments

replicate.dataset
  A Number indicate that how many you replicate dataset from user’s specified dataset.

ModifiedPoisson
  Logical, that is TRUE or FALSE.
  If ModifiedPoisson = TRUE, then Poisson rate of false alarm is calculated per lesion, and model is fitted so that the FROC curve is an expected curve of points consisting of the pairs of TPF per lesion and FPF per lesion.
  Similarly,
  If ModifiedPoisson = FALSE, then Poisson rate of false alarm is calculated per image, and model is fitted so that the FROC curve is an expected curve of points consisting of the pair of TPF per lesion and FPF per image.

For more details, see the author’s paper in which I explained per image and per lesion. (for details of models, see vignettes, now, it is omitted from this package, because the size of vignettes are large.)
If ModifiedPoisson = TRUE, then the False Positive Fraction (FPF) is defined as follows ($F_c$ denotes the number of false alarms with confidence level $c$)

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_3 + F_4 + F_5}{N_L},$$

$$\frac{F_4 + F_5}{N_L},$$

$$\frac{F_5}{N_L},$$

where $N_L$ is a number of lesions (signal). To emphasize its denominator $N_L$, we also call it the False Positive Fraction (FPF) per lesion.

On the other hand, if ModifiedPoisson = FALSE (Default), then False Positive Fraction (FPF) is given by

$$\frac{F_1 + F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_2 + F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_3 + F_4 + F_5}{N_I},$$

$$\frac{F_4 + F_5}{N_I},$$

$$\frac{F_5}{N_I},$$

where $N_I$ is the number of images (trial). To emphasize its denominator $N_I$, we also call it the False Positive Fraction (FPF) per image.

The model is fitted so that the estimated FROC curve can be regarded as the expected pairs of FPF per image and TPF per lesion (ModifiedPoisson = FALSE)
or as the expected pairs of FPF per image and TPF per lesion (\texttt{ModifiedPoisson = TRUE})

If \texttt{ModifiedPoisson = TRUE}, then FROC curve means the expected pair of FPF \textbf{per lesion} and TPF.

On the other hand, if \texttt{ModifiedPoisson = FALSE}, then FROC curve means the expected pair of FPF \textbf{per image} and TPF.

So, data of FPF and TPF are changed thus, a fitted model is also changed whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. In traditional FROC analysis, it uses only per images (trial). Since we can divide one image into two images or more images, number of trial is not important. And more important is per signal. So, the author also developed FROC theory to consider FROC analysis under per signal. One can see that the FROC curve is rigid with respect to change of a number of images, so, it does not matter whether \texttt{ModifiedPoisson = TRUE} or \texttt{FALSE}. This rigidity of curves means that the number of images is a redundant parameter for the FROC trial and thus the author try to exclude it.

Revised 2019 Dec 8 Revised 2019 Nov 25 Revised 2019 August 28

\begin{itemize}
\item \texttt{mean.truth} This is a parameter of the latent Gaussian assumption for the noise distribution.
\item \texttt{sd.truth} This is a parameter of the latent Gaussian assumption for the noise distribution.
\item \texttt{z.truth} This is a parameter of the latent Gaussian assumption for the noise distribution.
\item \texttt{NL} Number of Lesions.
\item \texttt{NI} Number of Images.
\item \texttt{ite} To be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. An argument of \texttt{rstan::sampling()} in which it is named \texttt{iter}. A positive integer representing the number of samples synthesized by Hamiltonian Monte Carlo method, and, Default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
\item \texttt{cha} To be passed to the function \texttt{rstan::sampling()} in \texttt{rstan}. An argument of \texttt{rstan::sampling()} in which it is named \texttt{chains}. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, Default = 1.
\item \texttt{summary} Logical: \texttt{TRUE} of \texttt{FALSE}. Whether to print the verbose summary. If \texttt{TRUE} then verbose summary is printed in the \texttt{R} console. If \texttt{FALSE}, the output is minimal. I regret, this variable name should be verbose.
\item \texttt{serial.number} An positive integer or Character. This is for programming perspective. The author use this to print the serial number of validation. This will be used in the validation function.
\item \texttt{base_size} An numeric for size of object, this is for the package developer.
\item \texttt{absolute.errors} A logical specifying whether mean of errors is defined by

\begin{align*}
\texttt{TRUE} & \quad \bar{\epsilon}(\theta_0, N_I, N_L) = \frac{1}{K} \sum_k |\epsilon_k| \\
\texttt{FALSE} & \quad \epsilon(\theta_0, N_I, N_L) = \frac{1}{K} \sum_k \epsilon_k
\end{align*}
\end{itemize}
Value

Return values is,

**Stanfit objects**  for each Replicated datasets

**Errors**  EAPs minus true values, in the above notations, it is \( \bar{\epsilon}(\theta_0, N_I, N_L) \)

**Variances of estimators.**  This calculates the variance of posterior means over all replicated datasets

Examples

```r
## Not run:
#=========================== The first example =================================================================

#  It is sufficient to run the function with default variable

datasets <- validation.dataset_srsc()

#============================= The second example =================================================================

#  If user do not familiar with the values of thresholds, then
#  it would be better to use the actual estimated values
#  as an example of true parameters. In the following,
#  I explain this.

#  First, to get estimates, we run the following:

fit <- fit_Bayesian_FROC(dataList.Chakra.1,ite = 1111,summary =FALSE,cha=3)

#  Secondly, extract the expected a posterior estimators (EAPs) from the object fit

z <- rstan::get_posterior_mean(fit,par=c("z"))[,"mean-all chains"]

#  Thirdly we use this z as a true values.

datasets <- validation.dataset_srsc(z.truth = z)
```
#========================================================================================
# 1) extract replicated fitted model object
#========================================================================================

# Replicates models
a <- validation.dataset_srsc(replicate.datset = 3, ite = 111)

# Check convergence, in the above MCMC iterations = 111 which is too small to get
# a convergence MCMC chain, and thus the following example will the example
# of a non-convergent model in the r hat criteria.
ConfirmConvergence(a$fit[[3]])

# Check trace plot to confirm whether MCMC chain do converge or not.
stan_trace(a$fit[[3]], pars = "A")

# Check p value
ppp(a$fit[[3]])

# In the above example, the posterior predictive p value is enough large,
# but the model did not converge in R that criteria, which will cause
# that the model does not fit to data. However p value is said
# we can not reject the null hypothesis that the model does fit.
# The author think this contradiction cause that the
# number of MCMC iterations are too small which leads us to incorrect
# Monte Carlo integral for p value. Thu p value is not correct.
# Calculation of p value relies on the law of large number and thus
# to obtain reliable posterior predictive p value, we need enough large
# MCMC samples. 2019 August 29

# Revised in 2019 August 29

#========================================================================================
# 1) Histogram of error of posterior means for replicated datasets
#========================================================================================
\texttt{validation.draw_ssrc}

\begin{verbatim}
#
# a<- validation.dataset_ssrc(replicate.dataset = 100)
hist(a$error.of.AUC,breaks = 111)
hist(a$error.of.AUC,breaks = 30)

# absolute.errors = FALSE generates negative biases
validation.dataset_ssrc(absolute.errors = FALSE)

# absolute.errors = TRUE does not generate negative biases
validation.dataset_ssrc(absolute.errors = TRUE)

## End(Not run)# dontrun

validation.draw_ssrc  \textit{Draw Curves for validation dataset}

\textbf{Description}

drawing curves.

\textbf{Red curve} indicates an FROC curve of truth parameter.

\textbf{Other curves} are drawn using replicated estimates.

\textbf{Usage}

validation.draw_ssrc(
    validation.data,
    mesh.for.drawing.curve = 11111,
    upper_y = 1,
    DrawFROCcurve = TRUE
)
Arguments

validation.data

This is a return value of the function validation.dataset_srsc.

mesh.for.drawing.curve

A positive large integer, indicating number of dots drawing the curves, Default = 10000.

upper_y

This is a upper bound for the axis of the vertical coordinate of FROC curve.

DrawFROCcurve

Logical: TRUE of FALSE. Whether or not FROC curves are shown.

Value

NULL

Examples

```r
## Not run:
# --------------------------------------------------------------------------------------
# 1) Draw the curve for each replicated dataset
# --------------------------------------------------------------------------------------

datasets <- validation.dataset_srsc()
validation.draw_srsc(datasets)

# --------------------------------------------------------------------------------------
# 1) Draw the curve for each replicated dataset
# --------------------------------------------------------------------------------------

datasets <- validation.dataset_srsc(replicate.dataset = 5)
validation.draw_srsc(datasets)

## End(Not run)# dottest
```

Description

Create a tabular representation of FROC data from FROC data object.
Usage

```r
viewdata(dataList, summary = TRUE, head.only = FALSE)
```

Arguments

dataList

A Single reader and A single modality (SRSC) case.

In a single reader and a single modality case, it should include \( f, h, NL, NI, C \).
For example data, see the datasets endowed with this package.

**data Format:**

*A single reader and a single modality case*

<table>
<thead>
<tr>
<th>confidence level</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>( c[1] = 5 )</td>
<td>( f[1] = F_1 = 1 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>( c[5] = 1 )</td>
<td>( f[5] = F_1 = 13 )</td>
</tr>
</tbody>
</table>

Multiple readers and multiple modalities case, i.e., MRMC case

In multiple readers and multiple modalities case, i.e., MRMC case, it should include \( m, q, c, h, f, NL, C, M, Q \) which means the followings:

\( C \) means the highest number of confidence level, this is a scalar.
\( M \) means the number of modalities
\( Q \) means the number of readers.
\( c \) means the confidence level vector. This vector must be made by \( \text{rep}(\text{rep}(C:1), M*Q) \).
\( m \) means the modality ID vector.
\( q \) means the reader ID vector.
\( h \) means the number of hits vector.
\( f \) means the number of false alarm vector.
\( NL \) means the Total number of lesions for all images, this is a scalar.

The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by \( C \), are included, however, its each confidence level vector also created in the program by \( C \). So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata_MRMC`. 
summary Logical: TRUE or FALSE. If true then results are printed, if FALSE this function do nothing.

head.only Logical: TRUE or FALSE. Whether it prints data of head part only (TRUE) or entire (FALSE). If TRUE, only head part are shown. Default is FALSE

Value

Nothing

In order to confirm your data, please view table before fitting. Confidence level vector are created in my program regardless of user’s confidence level vectors.

Author(s)

Issei Tsunoda

Examples

## Not run:

# The first example, we prepare the data in this package.

dat <- get(data("dataList.Chakra.1"))
viewdata(dat)

# The second examle, we consider a dataset of multiple readers and multiple modalities

dat <- get(data("dataList.Chakra.Web"))
viewdata(dat)

## End(Not run)# dottest

---

**viewdata_MRM**

*View MRM data*

**Description**

Build a table for data *dataList*. 
viewdata_MRMCM(dataList, summary = TRUE, head.only = FALSE)

Arguments

dataList

it should include \(m, q, c, h, f, NL, C, M, Q\) which means from the right:
- \(m\) means the modality ID vector
- \(q\) means the reader ID vector
- \(c\) means the confidence level
- \(h\) means the number of hits
- \(f\) means the number of false alarm
- \(NL\) means the Total number of lesions for all images
- \(C\) means the highest number of confidence level
- \(M\) means the number of modalities
- \(Q\) means the number of readers.

The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by \(C\), are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function viewdata_MRMCM.

summary

TRUE or FALSE, if true then results are printed, if FALSE this function do nothing.

head.only

Logical: TRUE of FALSE. Whether head part or entire. If TRUE, only head part are shown. Default is FALSE.

viewdata_srsC(dataList, summary = TRUE)

Description

In order to confirm that your dataset is correctly formulated, please view the data via table. my program makes new column of confidence levels which are used in my program. So, it is possible that your order of confidence level and Program’s order of confidence level are inverse. This function’s result table are the one which are used in program.

Usage

viewdata_srsC(dataList, summary = TRUE)
Arguments

dataList  
it should include f, h, NL, NI, C. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, user should confirm the orders by the function.

summary  
TRUE or FALSE, if true then results are printed, if FALSE this function do nothing.

Examples

## Not run:

# First, we prepare an example FROC data "dataList.Chakra.1" in this package.
# Note that this data should be formed as a single reader and a single modality.
# If data are multiple readers and multiple modalities, i.e., MRMC-data,
# then another function named viewdataMRMC is available for MRMC-data.

dat <- get(data("dataList.Chakra.1"))

# Show data named "dat";

viewdata_srsc(dat)

# The Reason why the author made this \code{viewdata_srsc} is
# the code does not refer your confidence level.
# More precisely, my program made the column vector of confidence levels
# from the its highest number,
# so, it may be occur the interpretation of code for hits and false alarm
# are inverse order compared with your data.

## End(Not run)

\begin{verbatim}

\end{verbatim}

\textbf{v_truth}

\textit{Standard Deviation: parameter of an MRMC model}

Description

A posterior mean of the model parameter for data \texttt{ddd} as an example of truth parameter.
v_truth_creator_for_many_readers_MRMC_data

Details

Standard Deviation Rate data of some MRMC data to use as a default value of the function hits_creator_from_rate. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

See Also

hits_creator_from_rate

Description

v of MRMC model paramter

Usage

v_truth_creator_for_many_readers_MRMC_data(M, Q)

Arguments

M An integer, indicating a number of modalities
Q An integer, indicating a number of readers

Value

An array, representing v of MRMC model paramter

Examples

v <- v_truth_creator_for_many_readers_MRMC_data(M=4, Q=50)
**waic**  

**WAIC Calculator**

**Description**

Calculates the WAIC of the fitted object of class stanfit whose stan file is described with `target +=`.

**Usage**

```r
waic(StanS4classwithTargetFormulation, dig = 4, summary = TRUE)
```

**Arguments**

- `StanS4classwithTargetFormulation`  
  This is a fitted model object built by `rstan::sampling()` whose model block is described by `target formulation` in the `rstan` package. This object is available for both S4 classes: stanfit and stanfitExtended.

- `dig`  
  The number of significant digits of WAIC.

- `summary`  
  Logical: `TRUE` or `FALSE`. Whether to print the verbose summary. If `TRUE` then verbose summary is printed in the R console. If `FALSE`, the output is minimal. I regret, this variable name should be `verbose`.

**Details**

WAIC is an abbreviation for Widely Applicable Information Criterion (Watanabe-Akaike Information Criterion)

**Value**

A real number, representing the value of WAIC of the fitted model object `StanS4classwithTargetFormulation`.

Revised 2020 Jan

**Examples**

```r
## Not run:
#========================================================================================
# Model selection based on WAIC
#========================================================================================
# First, we prepare the data endowed with this package:

dat <- get(data("dataList.Chakra.1"))
```
# Second, create a fitted model object;

    fit1 <- fit_Bayesian_FROC(dat,  
                           ModifiedPoisson = FALSE)

# Using the fitted model object "fit", we can calculate the WAIC of it

    waic(fit1)

# Furthermore,  
# the Author provides an another model for a single reader and a single modality case.  
# One is false alarm rates means "per lesion" and the other means "per image".  
# The above "fit" is "per image".  
# Now we shall consider to compare WAIC of these two models  
# To do so, next we shall fit the "per lesion" model to the data as follows:

    fit2 <- fit_Bayesian_FROC(dat,  
                           ModifiedPoisson = TRUE)

    waic(fit2)

# By compare two model's WAIC we can say which model is better.  
# Note that the smaller WAIC is better.

    waic(fit1) # per lesion model
    waic(fit2) # per image model

# For the dataset,  
# We should select one of the above two models  
# by the criteria that the smaller waic is better.  
# Namely, if the following inequality

    waic(fit2) > waic(fit1)
# is TRUE, then we should use fit1.
# Similarly, if the following inequality

\[
\text{waic(fit2)} < \text{waic(fit1)}
\]

# is TRUE, then we should use fit2.
# 2019.05.21 Revised.
# 2020 Feb Revised.

## End(Not run)# dottest

---

`z`  
*Threshold: parameter of an MRMC model*

**Description**

A posterior mean of the model parameter for data `ddd` as an example of truth parameter.

**Author(s)**

Issei Tsunoda <tsunoda.issei1111@gmail.com>

**See Also**

- `make_true_parameter_MRMC`
- `z_from_dz`

---

`z_from_dz`  
*Thresholds from its difference*

**Description**

Thresholds are created from its difference

\[
\begin{align*}
z[1] &= w \\
\end{align*}
\]

**Usage**

`z_from_dz(w, dz)`
Arguments

\( w \)  
a real number, indicating the first threshold

\( dz \)  
a vector of real numbers, indicating the difference of thresholds

Value

A vector of real numbers

Examples

\[
\text{z_from_dz}(1, c(2,3))
\]
\[
\text{z_from_dz}(1, c(0.2,0.03))
\]
\[
\text{z_from_dz}(1, c(0.2,0.03,0.004))
\]

```r
dz <- runif(3, # sample size
            0.01, # lower bound
            1    # upper bound
)

w <- rnorm(1,
            0,
            1
)

z_from_dz(w, dz)
```

---

**z_truth**  
**Threshold : parameter of an MRMC model**

Description

A posterior mean of the model parameter for data \( ddd \) as an example of truth parameter.
Details

Threshold Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com>

See Also

`hits_creator_from_rate`

Description

Fitting is done with single

Usage

dataList %>% ite

Arguments

dataList  A list, specifying an FROC data to be fitted a model. It consists of data of numbers of TPs, FPs, lesions, images. In addition, if in case of multiple readers or multiple modalities, then modality ID and reader ID are included also. The `dataList` will be passed to the function `rstan::sampling()` in `rstan`. This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and a single modality data, the `dataList` is made by the following manner:

dataList.Example <- list(h = c(41, 22, 14, 8, 1), # number of hits for each confidence level f = c(1, 2, 5, 11, 13), # number of false alarms for each confidence level NL = 124, # number of lesions (signals) NI = 63, # number of images (trials) C = 5) # number of confidence, .. the author thinks it can be calculated as the length of h or f ...? ha, why I included this. ha .. should be omitted.

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example).`

To make this R object `dataList` representing FROC data, this package provides three functions:
convertFromJafroc() If data is a JAFROC xlsx formulation.  
dataset_creator_new_version() Enter TP and FP data by table.  
create_dataset() Enter TP and FP data by interactive manner.

Before fitting a model, we can confirm our dataset is correctly formulated by using the function viewdata().

---

**A Single reader and a single modality (SRSC) case.**

In a single reader and a single modality case (srsc), dataList is a list consisting of \( f, h, NL, NI, C \) where \( f, h \) are numeric vectors and \( NL, NI, C \) are positive integers.

- \( f \) Non-negative integer vector specifying number of false alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- \( h \) Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- \( NL \) A positive integer, representing Number of Lesions.
- \( NI \) A positive integer, representing Number of Images.
- \( C \) A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. *Note that the maximal number of confidence level, denoted by \( C \), are included, however, Note that confidence level vector \( c \) should not be specified. If specified, will be ignored , since it is created by \( c \leftarrow c(rep(C:1)) \) in the program and do not refer from user input data, where \( C \) is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created \( c \) vector.

**data Format:**

*A single reader and a single modality case*

<table>
<thead>
<tr>
<th>confidence level</th>
<th>( c )</th>
<th>No. of false alarms ( f )</th>
<th>No. of hits ( h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitely present</td>
<td>( c[1] = 5 )</td>
<td>( f[1] = F_5 = 1 )</td>
<td>( h[1] = H_5 = 41 )</td>
</tr>
<tr>
<td>very subtle</td>
<td>( c[5] = 1 )</td>
<td>( f[5] = F_1 = 13 )</td>
<td>( h[5] = H_1 = 1 )</td>
</tr>
</tbody>
</table>

*false alarms = False Positives = FP*
* hits = True Positives = TP

Note that in FROC data, all confidence level means present (diseased, lesion) case only, no confidence level indicating absent. Since each reader marks his suspicious location only if he thinks lesions are present, and marked positions generates the hits or false alarms, thus each confidence level represents that lesion is present. In the absent case, reader does not mark any locations and hence, the absent confidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector c should not be specified. If specified, will be ignored, since it is created by c <-c(rep(C:1)) automatically in the program and do not refer from user input data even if it is specified explicitly, where C is the highest number of confidence levels. So you should check the compatibility of your data and the confidence level vector c <-c(rep(C:1)) via a table which can be displayed by the function viewdata().

**Multiple readers and multiple modalities case, i.e., MRMC case**

In case of multiple readers and multiple modalities, i.e., MRMC case, in order to apply the function fit_Bayesian_FROC(), dataset represented by an R list object representing FROC data must contain components m, q, c, h, f, NL, C, M, Q.

C A positive integer, representing the highest number of confidence level, this is a scalar.
M A positive integer vector, representing the number of modalities.
Q A positive integer, representing the number of readers.
m A vector of positive integers, representing the modality ID vector.
q A vector of positive integers, representing the reader ID vector.
c A vector of positive integers, representing the confidence level. This vector must be made by rep(rep(C:1),M*Q)
h A vector of non-negative integers, representing the number of hits.
f A vector of non-negative integers, representing the number of false alarms.
NL A positive integer, representing the Total number of lesions for all images, this is a scalar.

Note that the maximal number of confidence level (denoted by C) are included in the above R object. However, each confidence level vector is not included in the data, because it is created automatically from C. To confirm false positives and hits are correctly ordered with respect to the automatically generated confidence vector,

the function viewdata() shows the table. Revised 2019 Nov 27 Revised 2019 Dec 5

**Example data.**

Multiple readers and multiple modalities (i.e., MRMC)

<table>
<thead>
<tr>
<th>Modality ID</th>
<th>Reader ID</th>
<th>Confidence levels</th>
<th>No. of false alarms</th>
<th>No. of hits</th>
</tr>
</thead>
</table>
### Value

A fitted model object of class `stanfitExtended`

### Examples

```r
## Not run:
#
# In the following, d is a data-set and 1111 is a number of MCMC iterations
#

d %>% 1111

## End(Not run)
```
Index

(451

(plot,stanfitExtended,missing-method).

351

%>>%, 447

AFROC, 7

AFROC_curve, 8

aperm, 10

array, 10

array(from_array_to_vector), 294

array_easy_example, 9

array_of_hit_and_false_alarms_from_vector, 10

as, 235

Author_vs_classic_for_AUC, 12

BayesianFROC, 13

clearWorkspace, 58

Close_all_graphic_devices, 59

compare, 59

comparison, 60

Confirm_hit_rates_are_correctly_made_in_case_of_MRMCPosteriors, 63

ConfirmConvergence, 61

convertFromJafroc, 35, 51, 64, 83, 143, 179, 221, 227, 237, 266, 275, 300, 304, 309, 326, 360, 448

CoronaVirus_Disease_2019_prevalence, 73

create_dataList_MRM, 75

create_dataset, 36, 51, 81, 83, 143, 179, 221, 227, 237, 266, 275, 300, 304, 310, 326, 360, 448

Credible_Interval_for_curve, 82

d, 86, 103, 107, 124, 238

dark_theme, 86

data.bad.fit, 87, 238

data.hier.fictitious, 89, 238

data.MultiReaderMultiModality, 90

data.nonconverge.srsc, 90

data.SingleReaderSingleModality, 92

data_2modalities_2readers_3confidence, 115

data_of_36_readers_and_a_single_modality, 116

dataList.Chakra.1, 92, 237


dataList.Chakra.2, 21, 96, 237

dataList.Chakra.3, 21, 97, 237

dataList.Chakra.4, 21, 99, 237


dataList.divergent.transition.in.case.of.srsc, 107

dataList.High, 109, 238

dataList.high.ability, 109, 238

dataList.Low, 110, 238

dataList.low.ability, 110, 238

dataList.one.modality, 111

dataSet_creator_by_specifying_only_M_Q, 111

451
INDEX

ggplotFROC.EAP, 302
give_name_srsc_CFP_CTP_vector, 306
give_name_srsc_data, 309
hit_rate_adjusted_from_the_vector_p, 302
hits_creator_from_rate, 309
hits_false_alarms_creator_from_thresholds, 317
hits_from_thresholds, 322
hits_rate_creator, 323
initial_values_specification_for_stan_in_case_of_MRMC, 326
install_imports, 329
inv_Phi, 330, 349, 350
m_q_c_vector_from_M_Q_C, 345
make_TeX, 331
make_true_parameter_MRMC, 331
metadata_srsc_per_image, 332
metadata_to_DrawCurve_MRMC, 335
metadata_to_fit_MRMC, 335
missing-class, 351
mu, 341
muTruth, 342
mu_truth_creator_for_many_readers_MRMC_data, 343
p, 347
p_truth, 388
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit, 388
pairs_plot_if_divergent_transition_occurred, 347
pause, 348
Phi, 330, 349, 351
Phi_inv, 330, 349, 350
plot, ANY, ANY-method
  (plot, stanfitExtended, missing-method), 351
plot.stanfitExtended, missing-method, 351
plot_curve_and_hit_rate_and_false_rate_simultaneously, 353
plot_empirical_FROC_curves, 355
plot_FFP_and_TPF_from_a_dataset, 347, 359
plot_FFP_TPF_via_dataframe_with_split_factor, 347, 365
plot_test, 371
plotFROC, 352
pnorm_or_qnorm, 371
ppp_MRMC, 375
ppp_srsc, 377
print.stanfitExtended-method, 382
print_minimal_reproducible_code_in_case_of_MRMC, 384
print.stanfitExtended, 385
prior.predictor, 386
prior.print_MRMC, 387
prior.print_srsc, 387
priorResearch, 385
rank_statistics_with_two_parameters, 392
replicate_model_MRMC, 393
replicate_MRMC_dataList, 79, 396
sbc, 410, 415
sbbc, 399
seq_array_ind, 290, 400
show_codes_in_my_manuscript, 402
showGM, 401
Simulation_Based_Calibration_histogram, 402
Simulation_Based_Calibration_single_reader_single_modality_via_rstan_sbc, 414
Simulation_Based_Calibration_via_rstan_sbc_MRMC, 406
Simulation_Based_Calibration_single_reader_single_modality_via_rstan_sbc_MRMC, 414
size_of_return_value, 416
small_margin, 416
snippet_for_BayesianFROC, 418
sortAUC, 419
Stan_code_validation, 421
stan_model_of_sbc, 422
StatisticForANOVA, 423
summarize_MRMC, 423
summary_AUC_comparison_MRM, 424
summary_AUC_comparison_MRM_with_crayon, 425
summary_AUC_comparison_MRM_without_crayon, 425
summary_EAP_CI_srsc, 426
Test_Null_Hypothesis_that_all_modalities_are_same, 427
the_row_number_of_logical_vector, 428
to (from_array_to_vector), 294
TRUE.Counter.in.vector, 429
v, 430
v_truth, 441
v_truth_creator_for_many_readers_MRM_data, 442
validation.dataset_srsc, 430
validation.draw_srsc, 436
vector (from_array_to_vector), 294
viewdata_MRM, 439
viewdata_srsc, 440
waic, 421, 443
z, 445
z_from_dz, 445
z_truth, 446