# Package ‘PheVis’

October 20, 2023

**Type** Package

**Title** Automatic Phenotyping of Electronic Health Record at Visit Resolution

**Version** 1.0.4

**Date** 2023-10-20

**Description** Using Electronic Health Record (EHR) is difficult because most of the time the true characteristic of the patient is not available. Instead we can retrieve the International Classification of Disease code related to the disease of interest or we can count the occurrence of the Unified Medical Language System. None of them is the true phenotype which needs chart review to identify. However chart review is time consuming and costly. ‘PheVis’ is an algorithm which is phenotyping (i.e identify a characteristic) at the visit level in an unsupervised fashion. It can be used for chronic or acute diseases. An example of how to use ‘PheVis’ is available in the vignette. Basically there are two functions that are to be used: `train_phevis()` which trains the algorithm and `test_phevis()` which get the predicted probabilities. The detailed method is described in preprint by Ferté et al. (2020) <doi:10.1101/2020.06.15.20131458>.

**License** GPL (>= 2)

**Depends** R (>= 3.5.0)

**Imports** dplyr, ggplot2, glmnet, knitr, lme4, purrr, randomForest, Rcpp (>= 1.0.3), stats, tidyR, viridis, zoo

**Suggests** PRROC, rmarkdown, testthat

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Author** Thomas Ferte [aut, cre], Boris P. Hejblum [aut]

**Maintainer** Thomas Ferte <thomas.ferte@u-bordeaux.fr>

**Repository** CRAN

**Date/Publication** 2023-10-20 08:30:02 UTC
R topics documented:

- boot_df .................................................. 2
- build_qantsur ........................................... 3
- build_quali ............................................. 3
- check_arg_test_phevis ................................ 4
- check_arg_train_phevis ................................ 5
- cum_lag .................................................. 6
- data_perf ............................................... 6
- data_phevis ............................................ 7
- expcorrectC ............................................. 7
- fct_surrogate_quanti ................................ 8
- gindividual_plot ....................................... 9
- matrix_exp_smooth ..................................... 9
- noising ................................................. 10
- norm_var ............................................... 10
- phenorm_longit_fit ................................... 11
- phenorm_longit_simpl ................................ 12
- pred_lme4model ......................................... 13
- pretty_cv.glmnet ...................................... 13
- rolling_var ............................................. 14
- roll_time_sum ......................................... 15
- safe_selection ......................................... 16
- sur_exp_smooth ......................................... 17
- test_phevis ............................................ 17
- train_phevis ........................................... 19

Index 21

---

**boot_df**

___

**Description**

Sample rows with replacement from a matrix

**Usage**

```r
boot_df(x_matrix, y_sur, ID = NULL, size = 10^5, seed = 1, prob = NULL)
```

**Arguments**

- `x_matrix` : matrix to perform sampling on
- `y_sur` : The numeric vector of the qualitative surrogate.
- `ID` : The patient ID
- `size` : size of matrix returned
- `seed` : seed for sampling
- `prob` : Vector for weight sampling
**build_qantsur**

**Value**
A list with the sampled explanatory matrix and the sampled qualitative surrogate (y_sur)
check_arg_test_phevis

Description
Function to check arguments passed to test_phevis()

Usage
check_arg_test_phevis(
  train_param,
  df_test,
  surparam,
  model,
  START_DATE,
  PATIENT_NUM,
  ENCOUNTER_NUM
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>train_param</td>
<td>Parameters for the model training (variables used, main ICD and CUIS, half_life, gold standard, omega). Usually obtained from train_phevis() function.</td>
</tr>
<tr>
<td>df_test</td>
<td>The dataframe on which to make the prediction.</td>
</tr>
<tr>
<td>surparam</td>
<td>The parameters used to compute the surrogate. Usually obtained by train_phevis() function.</td>
</tr>
<tr>
<td>model</td>
<td>The random intercept logistic regression. Usually obtained by train_phevis() function.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>Column name of the time column. The time column should be numeric</td>
</tr>
<tr>
<td>PATIENT_NUM</td>
<td>Column name of the patient id column.</td>
</tr>
<tr>
<td>ENCOUNTER_NUM</td>
<td>Column name of the encounter id column.</td>
</tr>
</tbody>
</table>

Value
No return value, stop the code execution if one condition is not met.
Description

Function to check arguments passed to train_phevis()

Usage

check_arg_train_phevis(
  half_life,
  df,
  START_DATE,
  PATIENT_NUM,
  ENCOUNTER_NUM,
  var_vec,
  main_icd,
  main_cui,
  rf,
  p.noise,
  bool_SAFE,
  omega,
  GS
)

Arguments

half_life Duration of cumulation. For a chronic disease you might chose Inf, for acute disease you might chose the duration of the disease.
df data.frame containing all the variables.
START_DATE Column name of the time column. The time column should be numeric
PATIENT_NUM Column name of the patient id column.
ENCOUNTER_NUM Column name of the encounter id column.
var_vec Explanatory variables used for the prediction, including the main variables.
main_icd Character vector of the column names of the main ICD codes.
main_cui Character vector of the column names of the main CUIs.
rf should pseudo-labelisation with random forest be used (default is true)
p.noise percentage of noise introduced during the noising step (default is 0.3)
bool_SAFE A boolean. If TRUE, SAFE selection is done, else it is not (default is TRUE)
omega Constant for the extrema population definition (default is 2)
GS Character string corresponding to the name of the gold-standard variable (default is null for which a vector of 0 will be taken).
**Value**

No return value, stop the code execution if one condition is not met.

---

<table>
<thead>
<tr>
<th>cum_lag</th>
<th>cum_lag</th>
</tr>
</thead>
</table>

**Description**

helpful function to cumulate information.

**Usage**

cum_lag(x, n_lag)

**Arguments**

- **x**: numeric vector for which lag variable should be computed
- **n_lag**: size of lag window

**Value**

return numeric vector.

---

<table>
<thead>
<tr>
<th>data_perf</th>
<th>Control data for test</th>
</tr>
</thead>
</table>

**Description**

Simulated dataset for PheVis phenotyping.

**Usage**

data(data_perf)

**Format**

An object of class numeric of length 2.
**data_phevis**

*PheVis simulated dataset*

---

**Description**

Simulated dataset for PheVis phenotyping.

**Usage**

```r
data(data_phevis)
```

**Format**

An object of class `data.frame` with 19659 rows and 15 columns.

---

**expcorrectC**

*expcorrectC*

---

**Description**

c++ function to compute exponential cumulation of information.

**Usage**

```r
expcorrectC(mat, diffdate, lambda)
```

**Arguments**

- `mat` A matrix where each column is a variable to be cumulated.
- `diffdate` Number of days between each sojourn. NA for switch of patient and restart cumulation.
- `lambda` A double to set the exponential cumulation.

**Details**

expcorrectC

**Value**

A matrix corresponding to the mat argument with cumulated exponential decay
Description

Compute the quantitative surrogate and then apply thresholds to get the qualitative surrogate.

Usage

```r
fct_surrogate_quanti(
  main_icd,  # Character vector of the column names of the main ICD codes.
  main_cui,  # Character vector of the column names of the main CUIs.
  df,        # Dataframe containing all variables.
  half_life, # Duration of accumulation. For a chronic disease you might chose Inf, for acute
date,      # disease you might chose the duration of the disease.
  patient_id, # Column name of the patient id column.
  encounter_id, # Column name of the encounter id column.
  omega = 2,  # Constant for the extrema population definition.
  param = NULL  # param of a previous train_phevis() result.
)
```

Arguments

- `main_icd`: Character vector of the column names of the main ICD codes.
- `main_cui`: Character vector of the column names of the main CUIs.
- `df`: Dataframe containing all variables.
- `half_life`: Duration of accumulation. For a chronic disease you might chose Inf, for acute disease you might chose the duration of the disease.
- `date`: Column name of the time column. The time column should be numeric.
- `patient_id`: Column name of the patient id column.
- `encounter_id`: Column name of the encounter id column.
- `omega`: Constant for the extrema population definition.
- `param`: param of a previous train_phevis() result.

Value

A list

- table - Main result: data.frame with the rolling variables and the surrogates
- param - the parameters for the standardisation of ICD and CUI
- roll_all - a subset of table with the rolling variables only
- quantile_vec - the quantile defining the extrema populations
**ggindividual_plot**

**Description**

Plot individual predictions.

**Usage**

`ggindividual_plot(subject, time, gold_standard, prediction)`

**Arguments**

- `subject`: numeric vector subject id
- `time`: numeric vector time or date
- `gold_standard`: numeric vector of gold standard
- `prediction`: numeric vector of prediction

**Value**

a ggplot graph

**Examples**

```r
ggindividual_plot(subject = rep(1,10),
                  time = 1:10,
                  gold_standard = c(0,0,1,1,0,0,1,1,0,0),
                  prediction = runif(n = 10, min = 0, max = 1))
```

**matrix_exp_smooth**

**Description**

Function to accumulate the information with exponential decay.

**Usage**

`matrix_exp_smooth(half_life, df, date, patient_id, encounter_id)`
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>half_life</td>
<td>Duration of accumulation. For a chronic disease you might chose Inf, for acute disease you might chose the duration of the disease.</td>
</tr>
<tr>
<td>df</td>
<td>Dataframe of the explanatory variables.</td>
</tr>
<tr>
<td>date</td>
<td>Vector of date. The date should be in a numeric format.</td>
</tr>
<tr>
<td>patient_id</td>
<td>The vector of patient id</td>
</tr>
<tr>
<td>encounter_id</td>
<td>The vector of visit id</td>
</tr>
</tbody>
</table>

Value

A data.frame object with both the raw variables and the accumulated ones.

Description

Noise a matrix

Usage

noising(X_boot, p = 0.3)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_boot</td>
<td>matrix to perform noise on</td>
</tr>
<tr>
<td>p</td>
<td>amount of noise</td>
</tr>
</tbody>
</table>

Value

A noised matrix

Description

Standardize a numeric variable

Usage

norm_var(x)
Arguments

x A numeric variable

Value

The standardized variable

Description

Apply simplified 'PheNorm' algorithm on longitudinal data with bootstrap and noise.

Usage

phenorm_longit_fit(x_matrix, y_sur, ID, size = 10^5, seed = 1, p.noise = 0.3, do_sampling = TRUE, do_noise = TRUE, prob = NULL, calc.prob = TRUE, nAGQ = 0, glmer.control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e+05)) )

Arguments

x_matrix x matrix to sample, noise and predict on
y_sur surrogate with 3 values (0 and 1 the extremes and 3 middle patients)
ID Vector of patient ID
size size of sampling. default is 10^5
seed seed. default is 1.
p.noise noise probability parameter. default is .3.
do_sampling should algorithm do sampling. default is TRUE.
do_noise should algorithm do noise. default is TRUE.
prob sampling probability during noising denoising step
calc.prob should the 'prob' argument be calculated
nAGQ glmer parameter
glmer.control glmer parameter
Value
A list with the fixed effects, the predicted responses and the model used (mixed effect or logistic regression)

Description
'PheNorm' like function adapted to longitudinal data.

Usage
phenorm_longit_simpl(
  df,
  var_surrogate,
  surrogates_quali,
  id_rnd,
  rf = FALSE,
  ntree = 100,
  bool_weight = FALSE,
  p.noise = 0.3,
  bool SAFE = TRUE,
  size = 10^5
)

Arguments

df            dataframe
var_surrogate variables used for building the surrogates
surrogates_quali numeric vector of the qualitative surrogate
id_rnd        ID for random effect
rf             should pseudo-labelisation with random forest be used (default is FALSE)
nTree          number of tree for randomforest (default is 100)
bool_weight    should the sampling probability balance the number of positive and negative extrema.
p.noise        percentage of noise introduced during the noising step
bool SAFE      A boolean. If TRUE, SAFE selection is done, else it is not (default is TRUE)
size          minimum size of sampling

Value
A list with the logistic model, the random forest model, the variables selected for prediction and the predictions
**pred_lme4model**

**Description**

function to predict probability from 'lme4' or 'glm' objects

**Usage**

pred_lme4model(model = NULL, fe.model = NULL, df)

**Arguments**

- **model**: lme4 model
- **fe.model**: the fixed effect of a model
- **df**: dataframe for prediction

**Value**

A vector of the predictions

---

**pretty_cv.glmnet**

**Description**

Train a 'glmnet' with cross validation (cv) model and return convenient results (model and results with non zero coefficients)

**Usage**

pretty_cv.glmnet(
    x_glmnet,
    y,
    alpha = 1,
    family = "binomial",
    s = "lambda.1se",
    weights = rep(1, nrow(x_glmnet)),
    ...)

)
Arguments

- **x_glmnet**: Independent variable matrix (X)
- **y**: Dependent variable vector (Y)
- **alpha**: alpha parameter of glmnet (default = 1)
- **family**: family parameter of glmnet (default = "binomial")
- **s**: lambda chosen from cv.glmnet (default = "lambda.1se")
- **weights**: glmnet parameter
- **...**: additional parameters passed to glmnet

Value

A list with the model, the coefficient associated with variables and the selected variables.

---

**rolling_var**

Description

Compute rolling variables (last visit, last 5 visits, last month and last year)

Usage

`rolling_var(id, var, start_date, id_encounter)`

Arguments

- **id**: Patient id numeric vector
- **var**: Variable numeric vector
- **start_date**: Time numeric vector
- **id_encounter**: Encounter id vector

Value

A dataframe containing the rolling variables.
**Description**

Compute the cumulated information of what happened in past month and past year.

**Usage**

```r
roll_time_sum(id, id_encounter, var, start_date, win_size1 = 30, win_size2 = 365, name1 = "cum_month", name2 = "cum_year")
```

**Arguments**

- **id**: Patient id numeric vector
- **id_encounter**: Encounter id vector
- **var**: Variable numeric vector
- **start_date**: Time numeric vector
- **win_size1**: First window size (default is 30)
- **win_size2**: Second window size (default is 365)
- **name1**: name of first rolling var (default is "cum_month")
- **name2**: name of second rolling var (default is "cum_year")

**Value**

A dataframe containing the rolling variables.
safe_selection

Description

Select the variables from dataframe by removing the rare variables and apply 'SAFE' on it.

Usage

```r
safe_selection(
  df,
  var_surrogate,
  surrogate_quali,
  threshold = 0.05,
  alpha = 0.5,
  remove_var_surrogate = TRUE,
  bool_weight = FALSE,
  ...
)
```

Arguments

- `df` : dataframe
- `var_surrogate` : variables used for building the surrogates
- `surrogate_quali` : surrogate with 3 values (0 and 1 the extremes and 3 middle patients)
- `threshold` : rareness threshold (default = 0.05).
- `alpha` : glmnet parameter (default is 0.5 elastic net)
- `remove_var_surrogate` : does the glmnet algorithm should learn on features in var_surrogate (default is TRUE).
- `bool_weight` : Should the glmnet function be weighted to balance the extrema populations (default is FALSE).
- `...` : arguments to pass to pretty_cv.glmnet

Value

A list

- `glmnet_model` - A list of three elements: the cv.glmnet fitted model, the coefficients of non zero variables and the vector of non zero coefficient variables.
- `important_var` - A vector with the variables used for the surrogate and the non zero variables.
- `surrogate_quali` - The surrogate_quali argument.
**sur_exp_smooth**

Description

Function to cumulate surrogate with exponential decay

Usage

```
sur_exp_smooth(half_life, sur, date, patient_id, encounter_id)
```

Arguments

- `half_life`: Duration of cumulation. For a chronic disease you might chose Inf, for acute disease you might chose the duration of the disease.
- `sur`: The quantitative surrogate.
- `date`: A numeric vector of time of days unit.
- `patient_id`: Vector of patient ID
- `encounter_id`: Vector of encounter ID

Value

A dataframe with the cumulated surrogate.

---

**test_phevis**

Description

test_phevis

Usage

```
test_phevis(
    train_param,
    df_test,
    surpparam,
    model,
    START_DATE,
    PATIENT_NUM,
    ENCOUNTER_NUM
)
```
Arguments

- **train_param**: Parameters for the model training (variables used, main ICD and CUIS, half_life, gold standard, omega). Usually obtained from train_phevis() function.
- **df_test**: The dataframe on which to make the prediction.
- **surparam**: The parameters used to compute the surrogate. Usually obtained by train_phevis() function.
- **model**: The random intercept logistic regression. Usually obtained by train_phevis() function.
- **START_DATE**: Column name of the time column. The time column should be numeric
- **PATIENT_NUM**: Column name of the patient id column.
- **ENCOUNTER_NUM**: Column name of the encounter id column.

Value

A dataframe with the predictions.

Examples

```r
library(dplyr)
library(PRROC)
PheVis::data_phevis
PheVis::data_perf

var_vec <- c(paste0("var",1:10), "mainCUI", "mainICD")
main_icd <- "mainICD"
main_cui <- "mainCUI"
GS <- "PR_state"
half_life <- Inf

df <- data_phevis %>%
  mutate(ENCOUNTER_NUM = row_number(),
         time = round(as.numeric(time)))

trainsize <- 0.8*length(unique(df$subject))
trainid <- sample(x = unique(df$subject), size = trainsize)
testid <- unique(df$subject)[!unique(df$subject) %in% trainid]

df_train <- as.data.frame(df[df$subject %in% trainid,])
df_test <- as.data.frame(df[df$subject %in% testid,])

##### train and test model #####
train_model <- PheVis::train_phevis(half_life = half_life,
                                   df = df_train,
                                   START_DATE = "time",
                                   PATIENT_NUM = "subject",
                                   ENCOUNTER_NUM = "ENCOUNTER_NUM",
                                   var_vec = var_vec,
                                   main_icd = main_icd,
                                   main_cui = main_cui)
```
train_phevis

Global function to train phevis model.

Usage

train_phevis(half_life, df, START_DATE, PATIENT_NUM, ENCOUNTER_NUM, var_vec, main_icd, main_cui, rf = TRUE, p.noise = 0.3, bool_SAFE = TRUE, omega = 2, GS = NULL)

Arguments

half_life Duration of cumulation. For a chronic disease you might chose Inf, for acute disease you might chose the duration of the disease.

df data.frame containing all the variables.

START_DATE Column name of the time column. The time column should be numeric.

PATIENT_NUM Column name of the patient id column.
ENCOUNTER_NUM  Column name of the encounter id column.
var_vec        Explanatory variables used for the prediction, including the main variables.
main_icd       Character vector of the column names of the main ICD codes.
main_cui       Character vector of the column names of the main CUIs.
rf             should pseudo-labellisation with random forest be used (default is true)
p.noise        percentage of noise introduced during the noising step (default is 0.3)
bool_SAFE      A boolean. If TRUE, SAFE selection is done, else it is not (default is TRUE)
omega          Constant for the extrema population definition (default is 2)
GS             Character string corresponding to the name of the gold-standard variable (default is null for which a vector of 0 will be taken).

Value

A list

- surparam - the parameters used to compute the surrogate
- model - the random intercept logistic regression
- df_train_result - the data.frame containing the output predictions
- train_param - parameters for the model training (variables used, main ICD and CUIS, half_life, gold standard)

Examples

library(dplyr)
PheVis::data_phevis
df <- data_phevis %>%
    mutate(ENCOUNTER_NUM = row_number(),
           time = round(as.numeric(time)))
model <- PheVis::train_phevis(half_life = Inf,
                           df = df,
                           START_DATE = "time",
                           PATIENT_NUM = "subject",
                           ENCOUNTER_NUM = "ENCOUNTER_NUM",
                           var_vec = c(paste0("var",1:10), "mainCUI", "mainICD"),
                           main_icd = "mainICD",
                           main_cui = "mainCUI")
Index

* datasets
  data_perf, 6
  data_phevis, 7

boot_df, 2
build_qantsur, 3
build_quali, 3

check_arg_test_phevis, 4
check_arg_train_phevis, 5
cum_lag, 6

data_perf, 6
data_phevis, 7

expcorrectC, 7
fct_surrogate_quanti, 8
ggindividual_plot, 9
matrix_exp_smooth, 9
noising, 10
norm_var, 10

phenorm_longit_fit, 11
phenorm_longit_simpl, 12
pred_lme4model, 13
pretty_cv.glmnet, 13

roll_time_sum, 15
rolling_var, 14

safe_selection, 16
sur_exp_smooth, 17

test_phevis, 17
train_phevis, 19