Package ‘SpecsVerification’

February 26, 2020

Version 0.5-3
Date 2020-02-26
Title Forecast Verification Routines for Ensemble Forecasts of Weather and Climate
Maintainer Stefan Siegert <s.siegert@exeter.ac.uk>
Description A collection of forecast verification routines developed for the SPECS FP7 project. The emphasis is on comparative verification of ensemble forecasts of weather and climate.
License GPL (>= 2)
Imports Rcpp, methods
LinkingTo Rcpp, RcppArmadillo
Suggests testthat
LazyData true
RoxygenNote 7.0.2
NeedsCompilation yes
Author Stefan Siegert [aut, cre], Jonas Bhend [ctb], Igor Kroener [ctb], Matteo De Felice [ctb]
Repository CRAN
Date/Publication 2020-02-26 15:40:06 UTC

R topics documented:

AbsErr .............................................. 3
Auc ..................................................... 3
AucDiff .............................................. 4
aucdiff_cpp ......................................... 6
auc_cpp ............................................. 6
BrierDecomp ....................................... 7
ClimEns ............................................ 8
R topics documented:

Corr ................................................................. 9
CorrDiff .......................................................... 10
Detrend ............................................................ 11
DressCrps .......................................................... 11
DressCrpsDiff ...................................................... 12
DressCrpss ......................................................... 13
dresscrps_cpp ..................................................... 14
DressEnsemble ..................................................... 14
DressIgn ........................................................... 16
DressIgnDiff ......................................................... 17
EnsBrier ............................................................. 17
EnsBrierDiff ......................................................... 18
EnsBrierSs .......................................................... 19
EnsCorr ............................................................. 20
EnsCrps ............................................................. 21
EnsCrpsDiff ......................................................... 22
EnsCrpss ........................................................... 22
enscrps_cpp ........................................................ 23
EnsQs ............................................................... 23
EnsRps ............................................................. 24
EnsRpsDiff .......................................................... 25
EnsRpss ............................................................. 26
eurotempforecast .................................................... 27
FairBrierDiff ......................................................... 28
FairBrierSs .......................................................... 28
FairCrpsDiff ......................................................... 29
FairCrpss ........................................................... 30
FairRpsDiff .......................................................... 30
FairRpss ............................................................ 31
FitAkPdParameters .................................................. 32
GaussCrps .......................................................... 33
GaussCrpsDiff ....................................................... 34
GaussCrpss ........................................................ 34
GenerateToyData ..................................................... 35
GetDensity .......................................................... 36
PlotDressedEns ...................................................... 38
PlotRankhist ......................................................... 39
Rankhist ........................................................... 40
ReliabilityDiagram .................................................. 41
ScoreDiff ........................................................... 42
SkillScore ........................................................... 43
SpecsVerification .................................................... 44
SqErr ............................................................... 44
TestRankhist ......................................................... 45

Index 47
**AbsErr**  
*Calculate the absolute error between forecast and observation*

**Description**
Calculate the absolute error between forecast and observation

**Usage**
```
AbsErr(fcst, obs)
```

**Arguments**
- `fcst` a N-vector representing N time instances of real-valued forecasts
- `obs` a N-vector representing N time instances of real-valued observations

**Value**
numeric N-vector of absolute errors |fcst - obs|

**See Also**
SqErr, ScoreDiff, SkillScore

**Examples**
```
data(eurotempforecast)
mean(AbsErr(rowMeans(ens), obs))
```

---

**Auc**  
*Calculate area under the ROC curve (AUC) for a forecast and its verifying binary observation, and estimate the variance of the AUC*

**Description**
Calculate area under the ROC curve (AUC) for a forecast and its verifying binary observation, and estimate the variance of the AUC

**Usage**
```
Auc(
    fcst, obs,
    handle.na = c("na.fail", "only.complete.pairs"),
    use_fn = c("C++", "R")
)
```
Arguments

- `fcst`: vector of forecasts
- `obs`: vector of binary observations (0 for non-occurrence, 1 for occurrence of the event)
- `handle.na`: how should missing values in forecasts and observations be handled; possible values are 'na.fail' and 'only.complete.pairs'; default: 'na.fail'
- `use_fn`: the function used for the calculation: 'C++' (default) for the fast C++ implementation, or 'R' for the slow (but more readable) R implementation

Value

vector containing AUC and its estimated sampling standard deviation

References


See Also

AucDiff

Examples

```r
data(eurotempforecast)
Auc(rowMeans(ens.bin), obs.bin)
```

AucDiff

*Calculate difference between areas under the ROC curve (AUC) between a forecast and a reference forecast for the same observation, and estimate the variance of the AUC difference*

Description

Calculate difference between areas under the ROC curve (AUC) between a forecast and a reference forecast for the same observation, and estimate the variance of the AUC difference
**AucDiff**

**Usage**

AucDiff(
  fcst,  
  fcst.ref,  
  obs, 
  handle.na = c("na.fail", "only.complete.triplets"), 
  use_fn = c("C++", "R")
)

**Arguments**

- **fcst**: vector of forecasts
- **fcst.ref**: vector of reference forecasts
- **obs**: vector of binary observations (0 for non-occurrence, 1 for occurrence of the event)
- **handle.na**: how should missing values in forecasts and observations be handled; possible values are 'na.fail' and 'only.complete.triplets'; default: 'na.fail'
- **use_fn**: the function used for the calculation: 'C++' (default) for the fast C++ implementation, or 'R' for the slow (but more readable) R implementation

**Value**

vector with AUC difference, and estimated standard deviation

**References**


**See Also**

Auc

**Examples**

data(eurotempforecast)
AucDiff(rowMeans(ens.bin), ens.bin[, 1], obs.bin)
aucdiff_cpp

Calculate AUC difference ‘AUC(fcst,obs) - AUC(fcst_ref, obs)’ of two forecasts for the same observations, and the sampling standard deviation of the AUC difference (Internal C++ implementation)

Description

Calculate AUC difference ‘AUC(fcst,obs) - AUC(fcst_ref, obs)’ of two forecasts for the same observations, and the sampling standard deviation of the AUC difference (Internal C++ implementation)

Usage

aucdiff_cpp(fcst, fcst_ref, obs)

Arguments

fcst numeric vector of forecasts (NAs are not allowed)
fcst_ref numeric vector of reference forecasts (NAs are not allowed)
obs vector of binary observations (obs[t] evaluates to TRUE if event happens at instance t, to FALSE otherwise)

Value

AUC values, their sampling standard deviations, the AUC difference, and their sampling standard deviations

See Also

Auc AucDiff

auc_cpp

Calculate AUC and its sampling standard deviation (Internal C++ implementation)

Description

Calculate AUC and its sampling standard deviation (Internal C++ implementation)

Usage

auc_cpp(fcst, obs)
Arguments

fcst numeric vector of forecasts (NAs are not allowed)
obsv vector of binary observations (obs[t] evaluates to TRUE if event happens at instance t, to FALSE otherwise)

Value

AUC and its sampling standard deviation

See Also

Auc AucDiff

BrierDecomp

Brier Score decomposition

Description

Return decomposition of the Brier Score into Reliability, Resolution and Uncertainty, and estimated standard deviations

Usage

BrierDecomp(p, y, bins = 10, bias.corrected = FALSE)

Arguments

p vector of forecast probabilities
y binary observations, y[t]=1 if an event happens at time t, and y[t]=0 otherwise
bins binning to estimate the calibration function (see Details), default: 10
bias.corrected logical, default=FALSE, whether the standard (biased) decomposition of Murphy (1973) should be used, or the bias-corrected decomposition of Ferro (2012)

Details

To estimate the calibration curve, the unit line is categorised into discrete bins, provided by the ‘bins’ argument. If ‘bins’ is a single number, it specifies the number of equidistant bins. If ‘bins’ is a vector of values between zero and one, these values are used as the bin-breaks.

Value

Estimators of the three components and their estimated standard deviations are returned as a 2*3 matrix.
ClimEns

Construct a climatological ensemble from a vector of observations.

Description

Construct a climatological ensemble from a vector of observations. Optionally, the climatological ensemble observation at time t can be constructed without the observation at time t (leave-one-out).

Usage

ClimEns(obs, leave.one.out=FALSE)

Arguments

obs vector of length N. The observations.
leave.one.out logical, default=FALSE. If TRUE, the n-th observation is removed from the n-th row of the ensemble matrix.

Value

matrix with N rows and N-1 columns (if leave.one.out==TRUE) or N columns otherwise.

Examples

data(eurotempforecast)
ClimEns(obs)
Corr

Correlation between forecasts and observations, and assess uncertainty

Description

Calculate correlation between forecasts and observations, and assess uncertainty

Usage

Corr(fcst, obs, N.eff = NA, conf.level = 0.95, handle.na = "na.fail")

Arguments

- fcst: vector of forecasts
- obs: vector of observations
- N.eff: user-defined effective sample size to be used in hypothesis test and for confidence bounds; if NA, the length of ‘obs’ is used after removing missing values; default: NA
- conf.level: confidence level used the confidence interval; default = 0.95
- handle.na: how should missing values in forecasts and observations be handled; possible values are 'na.fail' and 'use.pairwise.complete'; default: 'na.fail'

Value

vector with correlation, one-sided p-value, and central confidence interval at the user-defined confidence level

References


See Also

CorrDiff

Examples

data(eurotempforecast)
Corr(rowMeans(ens), obs)
CorrDiff

*Calculate correlation difference between a forecast and a reference forecast, and assess uncertainty*

---

**Description**

Calculate correlation difference between a forecast and a reference forecast, and assess uncertainty

**Usage**

```r
CorrDiff(
  fcst,  # vector of forecasts
  fcst.ref,  # vector of reference forecasts
  obs,  # vector of observations
  N.eff = NA,  # user-defined effective sample size to be used in hypothesis test and for confidence bounds; if NA, the length of 'obs' is used after removing missing values; default: NA
  conf.level = 0.95,  # confidence level for the confidence interval; default = 0.95
  handle.na = "na.fail"  # how should missing values in forecasts and observations be handled; possible values are 'na.fail' and 'only.complete.triplets'; default: 'na.fail'
)
```

**Arguments**

- `fcst`: vector of forecasts
- `fcst.ref`: vector of reference forecasts
- `obs`: vector of observations
- `N.eff`: user-defined effective sample size to be used in hypothesis test and for confidence bounds; if NA, the length of ‘obs’ is used after removing missing values; default: NA
- `conf.level`: confidence level for the confidence interval; default = 0.95
- `handle.na`: how should missing values in forecasts and observations be handled; possible values are ‘na.fail’ and ‘only.complete.triplets’; default: ‘na.fail’

**Value**

vector with correlation difference, one-sided p-value, and central confidence interval at the user-defined confidence level

**References**


**See Also**

Corr
Detrend

**Examples**

```r
data(eurotempforecast)
CorrDiff(rowMeans(ens), ens[, 1], obs)
```

**Description**

Detrend fits a linear function to a time-series of observations or to the time-series of ensemble means of an ensemble matrix. The linear trend is removed, and if option demean is true, the total mean is removed as well.

**Usage**

```r
Detrend(x, demean = TRUE)
```

**Arguments**

- `x` A vector, matrix, or data.frame.
- `demean` logical; if true, the total mean is removed from x

**Value**

The function returns an object of the same dimensions as the argument 'x', but with its linear trend and (possibly) its mean removed.

**Examples**

```r
data(eurotempforecast)
Detrend(ens)
Detrend(obs, demean=FALSE)
```

**DressCrps**

*Calculate the Continuous Ranked Probability Score (CRPS) for a mixture of Normal distributions, for example generated by ensemble dressing*

**Description**

Calculate the Continuous Ranked Probability Score (CRPS) for a mixture of Normal distributions, for example generated by ensemble dressing
DressCrpsDiff

Calculate DressCrps Difference (deprecated, use function ScoreDiff instead)

Description

Calculate DressCrps Difference (deprecated, use function ScoreDiff instead)

Usage

DressCrpsDiff(dressed.ens, dressed.ens.ref, obs, probs = NA)

Arguments

dressed.ens the ensemble
dressed.ens.ref the reference ensemble
obs the observation
probs not used

References


See Also

EnsCrps, ScoreDiff, SkillScore

Examples

data(eurotempforecast)
dressed.ens <- DressEnsemble(ens)
mean(DressCrps(dressed.ens, obs))

DressCrpss

**Value**

mean DressCrps difference

**See Also**

ScoreDiff DressCrps DressEnsemble

| DressCrpss | Calculate DressCrps Skill Score (deprecated, use function SkillScore instead) |

**Description**

Calculate DressCrps Skill Score ( deprecated, use function SkillScore instead)

**Usage**

DressCrpss(dressed.ens, dressed.ens.ref, obs)

**Arguments**

- dressed.ens: the ensemble
- dressed.ens.ref: the reference ensemble
- obs: the observation

**Value**

DressCrps Skill Score

**See Also**

SkillScore DressCrps DressEnsemble
**dresscrps_cpp**  
*Dress CRPS*

**Description**

Dress CRPS

**Usage**

dresscrps_cpp(m, s, y)

**Arguments**

- **m**  
  vector of kernel means
- **s**  
  vector of kernel standard deviations
- **y**  
  observation

**Value**

crps

---

**DressEnsemble**  
*Transform an ensemble forecast to a continuous forecast distribution by kernel dressing.*

**Description**

Transform an ensemble forecast to a continuous forecast distribution by kernel dressing.

**Usage**

DressEnsemble(ens, dressing.method = "silverman", parameters = NA)

**Arguments**

- **ens**  
  a N*R matrix representing N time instances of real-valued R-member ensemble forecasts
- **dressing.method**  
  One of "silverman" (default), "akd", "akd.fit". See Details.
- **parameters**  
  A list, containing the parameters for the dressing method. See Details.
Details

The dressing methods currently implemented and their required parameters are:

"silverman" (default) No parameters are given. At time instance 'n' each ensemble member is replaced by a Gaussian kernel with mean ens[n, k] and variance (4 / 3 / K)^0.4 * var(ens[n, ]). This method is called "Silverman’s rule of thumb" and provides a simple non-parametric method for smoothing a discrete ensemble.

"akd" Affine Kernel Dressing. The required parameters are list(r1, r2, a, s1, s2). The 'k'-th ensemble member at time instance 'n' is dressed with a Gaussian kernel with mean r1 + r2 * mean(ens[n,]) + a * ens[n, k] and variance (4 / 3 / K)^0.4 * (s1 + s2 * a^2 * var(ens[n,])). Negative variances are set to zero. Note that parameters = list(r1=0, r2=0, a=1, s1=0, s2=1) yields the same dressed ensemble as dressing.method="silverman".

"akd.fit" Affine Kernel Dressing with fitted parameters. The required parameters is list(obs), where 'obs' is a vector of observations which are used to optimize the parameters r1, r2, a, s1, s2 by CRPS minimization. See ?FitAkdParameters for more information.

Value

The function returns a list with elements 'ens' (a N*R matrix, where ens[t,r] is the mean of the r-th kernel at time instance t) and 'ker.wd' (a N*R matrix, where ker.wd[t,r] is the standard deviation of the r-th kernel at time t)

References


See Also

DressCrps, DressIgn, GetDensity, FitAkdParameters

Examples

data(eurotempforecast)
d.silverman <- DressEnsemble(ens)
d.akd <- DressEnsemble(ens, dressing.method="akd",
                     parameters=list(r1=0, r2=0, a=1,
                               s1=0, s2=0))
d.akd.fit <- DressEnsemble(ens, dressing.method="akd.fit",
                          parameters=list(obs=obs))
DressIgn

*Calculate the Logarithmic (Ignorance) Score for a mixture of Normal distributions, for example generated by ensemble dressing*

**Description**

Calculate the Logarithmic (Ignorance) Score for a mixture of Normal distributions, for example generated by ensemble dressing

**Usage**

```r
DressIgn(dressed.ens, obs)
```

**Arguments**

- `dressed.ens`: a list with elements 'ens', a N*R matrix representing N time instances of kernel centers, and 'ker.wd', a N*R matrix with corresponding kernel standard deviations. See function 'DressEnsemble'
- `obs`: a numeric vector of length N with real-valued observations

**Value**

numeric vector of length N with the Ignorance score values

**References**


**See Also**

DressEnsemble, DressCrps

**Examples**

```r
data(eurotempforecast)
d.ens <- DressEnsemble(ens)
DressIgn(d.ens, obs)
```
DressIgnDiff

Calculate DressIgn Difference (deprecated, use function ScoreDiff instead)

Description

Calculate DressIgn Difference (deprecated, use function ScoreDiff instead)

Usage

DressIgnDiff(dressed.ens, dressed.ens.ref, obs, probs = NA)

Arguments

dressed.ens  the ensemble
dressed.ens.ref  the reference ensemble
obs  the observation
probs  not used

Value

mean DressIgn difference

See Also

ScoreDiff DressIgn

EnsBrier

Calculate the ensemble-adjusted Brier Score

Description

Calculate the ensemble-adjusted Brier Score

Usage

EnsBrier(ens, obs, R.new = NA)

FairBrier(ens, obs)
Arguments

ens  a N*R matrix representing N time instances of R-member ensemble forecasts of binary events; ens[t,r]=1 if the r-th ensemble member at time t predicted the event, otherwise ens[t,r]=0

obs  a numeric vector of length N with binary observations; obs[t]=1 if the event happens at time t, otherwise obs[t]=0

R.new  ensemble size for which the scores should be adjusted

Details

‘FairBrier(ens, obs)’ returns ‘EnsBrier(ens, obs, R.new=Inf)’

Value

numeric vector of length N with the ensemble-adjusted Brier scores

References


See Also

EnsRps, EnsCrps, ScoreDiff, SkillScore

Examples

data(eurotempforecast)
mean(EnsBrier(ens.bin, obs.bin, R.new=Inf))

EnsBrierDiff  Calculate EnsBrier Difference (deprecated, use function ScoreDiff instead)

Description

Calculate EnsBrier Difference (deprecated, use function ScoreDiff instead)

Usage

EnsBrierDiff(ens, ens.ref, obs, tau = NA, probs = NA)
**EnsBrierSs**

**Arguments**
- `ens` the ensemble
- `ens.ref` the reference ensemble
- `obs` the observation
- `tau` not used
- `probs` not used

**Value**
mean EnsBrier difference

**See Also**
- ScoreDiff EnsBrier

---

**EnsBrierSs**  
`Calculate EnsBrier Skill Score (deprecated, use function SkillScore instead)`

**Description**
Calculate EnsBrier Skill Score (deprecated, use function SkillScore instead)

**Usage**
```
EnsBrierSs(ens, ens.ref, obs, tau = NA)
```

**Arguments**
- `ens` the ensemble
- `ens.ref` the reference ensemble
- `obs` the observation
- `tau` not used

**Value**
EnsBrier skill score

**See Also**
- SkillScore EnsBrier
EnsCorr

*Correlation skill analysis for ensemble forecasts*

**Description**

Calculate correlation between forecasts and observations for an ensemble forecast, including an adjustment for finite ensemble sizes.

**Usage**

`EnsCorr(ens, obs, R.new = NA)`

**Arguments**

- `ens`: a N*R matrix representing N time instances of real-valued R-member ensemble forecasts.
- `obs`: a numeric vector of length N with real-valued observations.
- `R.new`: positive number, can be Inf, ensemble size for which correlation skill should be estimated, default is NA for using the actual size R of the ensemble.

**Value**

A vector with 4 entries:

- `cmy`: Correlation skill of the ensemble mean forecast.
- `cmy_adj`: Correlation skill of the ensemble mean forecast adjusted to ensemble size R.new.
- `cxx`: Average correlation between ensemble members.
- `cxy`: Average correlation between individual ensemble members and observation.

**References**


**See Also**

`Corr`, `CorrDiff`

**Examples**

```R
data(eurotempforecast)
EnsCorr(ens, obs, R.new=Inf)
```
**EnsCrps**

*Calculate the ensemble-adjusted Continuous Ranked Probability Score (CRPS)*

**Description**

Calculate the ensemble-adjusted Continuous Ranked Probability Score (CRPS)

**Usage**

```r
EnsCrps(ens, obs, R.new = NA)

FairCrps(ens, obs)
```

**Arguments**

- `ens`: a N*R matrix representing N time instances of real-valued R-member ensemble forecasts
- `obs`: a numeric vector of length N with real-valued observations
- `R.new`: positive number, can be ‘Inf’, ensemble size for which the scores should be adjusted, default is NA for no adjustment

**Details**

‘FairCrps(ens, obs)’ returns ‘EnsCrps(ens, obs, R.new=Inf)’

**Value**

numeric vector of length N with the ensemble-adjusted CRPS values

**References**


**See Also**

EnsBrier, EnsRps, DressCrps, GaussCrps, ScoreDiff, SkillScore

**Examples**

```r
data(eurotempforecast)
mean(EnsCrps(ens, obs, R.new=Inf))
```
### EnsCrpsDiff

**Calculate EnsCrps Difference (deprecated, use function ScoreDiff instead)**

**Description**

Calculate EnsCrps Difference (deprecated, use function ScoreDiff instead)

**Usage**

```
EnsCrpsDiff(ens, ens.ref, obs, probs = NA)
```

**Arguments**

- `ens`: the ensemble
- `ens.ref`: the reference ensemble
- `obs`: the observation
- `probs`: not used

**Value**

mean EnsCrps difference

**See Also**

ScoreDiff EnsCrps

### EnsCrpss

**Calculate EnsCrps Skill Score (deprecated, use function SkillScore instead)**

**Description**

Calculate EnsCrps Skill Score (deprecated, use function SkillScore instead)

**Usage**

```
EnsCrpss(ens, ens.ref, obs)
```

**Arguments**

- `ens`: the ensemble
- `ens.ref`: the reference ensemble
- `obs`: the observation
knitr::opts_chunk$set(eval=TRUE)

## enscrps_cpp

**Value**

EnsCrps skill score

**See Also**

SkillScore EnsCrps

### enscrps_cpp

**Usage**

```r
enscrps_cpp(ens, obs, R_new)
```

**Arguments**

- `ens`: Ensemble members as columns of a matrix
- `obs`: The verifying observations
- `R_new`: Size for ensemble adjustment

**Value**

vector of crps values

### EnsQs

**Description**

Calculate the ensemble-adjusted Quadratic Score (QS) for categorical forecasts

**Usage**

```r
EnsQs(ens, obs, R.new = NA)
FairQs(ens, obs)
```
EnsRps

Arguments

ens a N*R matrix of integers, representing N time instances of categorical ensemble forecasts; ens[t,r] indicates the category index that the r-th ensemble member forecasts at time t

obs a vector of length N, obs[t] is the category that occurred at time t

R.new ensemble size for which the scores should be adjusted

Details

‘FairQs(ens, obs)’ returns ‘EnsQs(ens, obs, R.new=Inf)’

It is assumed that the smallest class index is 1, and the largest class index is calculated by max(c(ens,obs))

Value

numeric vector of length N with the ensemble-adjusted quadratic score values

See Also

EnsBrier, EnsRps, EnsCrps, ScoreDiff, SkillScore

Examples

data(eurotempforecast)
EnsQs(ens.cat, obs.cat, R.new=Inf)
EnsRpsDiff  

Calculate EnsRps Difference (deprecated, use function ScoreDiff instead)

Description

Calculate EnsRps Difference (deprecated, use function ScoreDiff instead)

Usage

EnsRpsDiff(ens, ens.ref, obs, probs = NA, format = c("category", "members"))

Arguments

 ens  the ensemble  
 ens.ref  the reference ensemble  
 obs  the observation  
 probs  not used  
 format  see ‘EnsRps’
EnsRps

Value

mean EnsRps difference

See Also

ScoreDiff EnsRps

| EnsRps | Calculate EnsRps Skill Score (deprecated, use function SkillScore instead) |

Description

Calculate EnsRps Skill Score (deprecated, use function SkillScore instead)

Usage

EnsRps(ens, ens.ref, obs, format = c("category", "members"))

Arguments

ens the ensemble
ens.ref the reference ensemble
obs the observation
format see ‘EnsRps’

Value

EnsRps skill score

See Also

SkillScore EnsRps
eurotempforecast

Seasonal ensemble forecast of European average summer temperature

Description

A hindcast dataset of average European (30N,75N,12.5W,42.5E) summer (June/July/August) surface temperatures. Forecasts were initialised in May the same year. Observations and 15-member ensemble forecasts were derived from the publicly available NCEP Reanalysis (Suranjana, 2010) and the NCEP Climate Forecast System Version 2 (Suranjana, 2014), respectively. The data was downloaded through the ECOMS User Data Gateway (Santander Meteorology Group, 2015).

Usage

data(eurotempforecast)

Format

Variables contained in the data set:

- ‘obs’ average European summer temperature observations
- ‘ens’ mean-debiased ensemble forecast data, i.e. mean(ens) == mean(obs)
- ‘obs.lag’ the observations lagged by one year, same length as ‘obs’
- ‘obs.bin’ binary observations (0 or 1), obs[i] = 1 indicates that the temperature of year i exceeded the temperature of year i-1
- ‘ens.bin’ binary ensemble forecast (each member is either 0 or 1), ens[i, j] = 1 if the j-th ensemble member in year i exceeded the observed temperature of year i-1
- ‘obs.cat’ categorical observations. obs.cat[i] is either 1, 2, and 3, indicating that the temperature in year i was lower, similar, higher than temperature in year i-1. Similar is defined as within a half degree interval centered around last years temperature.
- ‘ens.cat’ categorical ensemble forecast. ens.cat[i, j] is either 1, 2, or 3. The categories are defined as for ‘obs.cat’.

References

FairBrierDiff

Calculate FairBrier Difference (deprecated, use function ScoreDiff instead)

Description

Calculate FairBrier Difference (deprecated, use function ScoreDiff instead)

Usage

FairBrierDiff(ens, ens.ref, obs, tau = NA, probs = NA)

Arguments

- ens: the ensemble
- ens.ref: the reference ensemble
- obs: the observation
- tau: not used
- probs: not used

Value

mean FairBrier difference

See Also

ScoreDiff EnsBrier

FairBrierSs

Calculate FairBrier Skill Score (deprecated, use function SkillScore instead)

Description

Calculate FairBrier Skill Score (deprecated, use function SkillScore instead)

Usage

FairBrierSs(ens, ens.ref, obs, tau = NA)

Arguments

- ens: the ensemble
- ens.ref: the reference ensemble
- obs: the observation
- tau: not used
**FairCrpsDiff**

**Value**

FairBrier skill score

**See Also**

SkillScore EnsBrier

---

**FairCrpsDiff**  
*Calculate FairCrps Difference (deprecated, use function ScoreDiff instead)*

**Description**

Calculate FairCrps Difference (deprecated, use function ScoreDiff instead)

**Usage**

`FairCrpsDiff(ens, ens.ref, obs, probs = NA)`

**Arguments**

- `ens` the ensemble
- `ens.ref` the reference ensemble
- `obs` the observation
- `probs` not used

**Value**

mean FairCrps difference

**See Also**

ScoreDiff EnsCrps
**FairCrps**  
Calculate FairCrps Skill Score (deprecated, use function SkillScore instead)

**Description**  
Calculate FairCrps Skill Score (deprecated, use function SkillScore instead)

**Usage**  
FairCrps(ens, ens.ref, obs)

**Arguments**  
- **ens**  
  the ensemble
- **ens.ref**  
  the reference ensemble
- **obs**  
  the observation

**Value**  
FairCrps skill score

**See Also**  
SkillScore EnsCrps

**FairRpsDiff**  
Calculate FairRps Difference (deprecated, use function ScoreDiff instead)

**Description**  
Calculate FairRps Difference (deprecated, use function ScoreDiff instead)

**Usage**  
FairRpsDiff(ens, ens.ref, obs, probs = NA, format = c("category", "members"))

**Arguments**  
- **ens**  
  the ensemble
- **ens.ref**  
  the reference ensemble
- **obs**  
  the observation
- **probs**  
  not used
- **format**  
  see ‘EnsRps’
**Value**

mean FairRps difference

**See Also**

ScoreDiff EnsRps

---

| FairRps | Calculate FairRps Skill Score (deprecated, use function SkillScore instead) |

**Description**

Calculate FairRps Skill Score (deprecated, use function SkillScore instead)

**Usage**

```r
FairRps(ens, ens.ref, obs, format = c("category", "members"))
```

**Arguments**

- `ens` : the ensemble
- `ens.ref` : the reference ensemble
- `obs` : the observation
- `format` : see ‘EnsRps’

**Value**

FairRps skill score

**See Also**

SkillScore EnsRps
FitAkdParameters

Fit the 5 parameters used for affine kernel dressing by minimum CRPS estimation.

Description

Fit the 5 parameters used for affine kernel dressing by minimum CRPS estimation.

Usage

FitAkdParameters(ens, obs)

Arguments

ens  
a N*R matrix. An archive of R-member ensemble forecasts for N time instances.
obs  
a vector of length N. The verifying observations corresponding to the N ensemble forecasts.

Details

Affine Kernel Dressing transforms the discrete K-member forecast ensemble at time instance n, ‘ens[n, j]’, to a continuous distribution function for the target ‘y’ by the equation:

\[ p(y|\text{ens}) = \frac{1}{K} \sum \text{dnorm}(y, z.i, s) \]

where \[ s = \left(\frac{4}{3}/K\right)^{0.4} \left(s_1 + s_2 \cdot a^2 \cdot \text{var(ens)}\right) \]
and \[ z.i = r1 + r2 \cdot \text{mean(ens)} + a \cdot \text{ens} \]

The parameters \( r1, r2, a, s1, s2 \) are fitted by minimizing the continuously ranked probability score (CRPS). The optimization is carried out using the R function ‘optim(...)’.

Since the evaluation of the CRPS is numerically expensive, the optimization can take a long time. Speed can be increased by optimizing the parameters only for a part of the forecast instances.

Value

The function returns a list of 5 parameters for affine kernel dressing.

References


See Also

DressEnsemble, DressCrps, DressIgn, PlotDressedEns, GetDensity

Examples

data(eurotempforecast)
FitAkdParameters(ens, obs)
GaussCrps

Calculate the Continuous Ranked Probability Score (CRPS) for forecasts issued as Normal distributions

Description

Calculate the Continuous Ranked Probability Score (CRPS) for forecasts issued as Normal distributions

Usage

GaussCrps(mean, sd, obs)

Arguments

mean A vector of length N. The forecast means.

sd A vector of length N. The forecast standard deviations.

obs A numeric vector of length N of real-valued verifying observations

Value

numeric vector of length N with the CRPS values

References


See Also

EnsCrps, DressCrps, ScoreDiff, SkillScore

Examples

data(eurotempforecast)
mean <- rowMeans(ens)
sd <- apply(ens, 1, sd)
mean(GaussCrps(mean, sd, obs))
**GaussCrpsDiff**  
*Calculate GaussCrps Difference (deprecated, use function ScoreDiff instead)*

**Description**  
Calculate GaussCrps Difference (deprecated, use function ScoreDiff instead)

**Usage**  
`GaussCrpsDiff(mean, sd, mean.ref, sd.ref, obs, probs = NA)`

**Arguments**  
- `mean`: forecast means  
- `sd`: forecast standard deviations  
- `mean.ref`: reference forecast means  
- `sd.ref`: reference forecast standard deviations  
- `obs`: the observation  
- `probs`: not used

**Value**  
mean GaussCrps difference

**See Also**  
ScoreDiff GaussCrps

---

**GaussCrpss**  
*Calculate GaussCrps Skill Score (deprecated, use function SkillScore instead)*

**Description**  
Calculate GaussCrps Skill Score (deprecated, use function SkillScore instead)

**Usage**  
`GaussCrpss(mean, sd, mean.ref, sd.ref, obs)`
GenerateToyData

Generate artificial data for ensemble verification using a signal-plus-noise model

Usage

GenerateToyData(
  N = 20,
  mu.y = 0,
  s.s = 7,
  s.eps = 6,
  mu.x = 0,
  beta = 0.2,
  s.eta = 8,
  K = 10,
  mu.x.ref = NA,
  beta.ref = NA,
  s.eta.ref = NA,
  K.ref = NA
)

Arguments

N       number of forecasts and observations
mu.y    expectation value of the observations
s.s     standard deviation of the predictable signal
Details

The function simulates data from the latent variable model:

\[ y_t = \mu_y + s_t + \varepsilon_t \]

\[ x_{t,r} = \mu_x + \beta \times s_t + \eta_{t,r} \]

where \( y_t \) is the observation at time \( t \), and \( x_{t,r} \) is the \( r \)-th ensemble member at time \( t \). The latent variable \( s_t \) is to be understood as the "predictable signal" that generates correlation between observations and ensemble members. If all arguments that end in ".ref" are specified, a reference ensemble is returned to also test comparative verification.

Value

A list with elements:

- **obs** N-vector of observations
- **ens** N*K matrix of ensemble members
- **ens.ref** N*K.ref matrix of reference ensemble members

Examples

```r
l <- GenerateToyData()
with(l, EnsCrps(ens, obs))
```

---

GetDensity

Calculate density and integrated density function of a dressed ensemble forecast at a matrix of values

Description

Calculate density and integrated density function of a dressed ensemble forecast at a matrix of values

Usage

```r
GetDensity(dressed.ens, x, integrated = FALSE)
```
GetDensity

Arguments

- **dressed.ens**: A list returned by the function ‘DressEnsemble’. See ‘?DressEnsemble’ for details.
- **x**: A matrix with either 1 row or nrow(dressed.ens["ens"]) rows and an arbitrary number of columns, holding the arguments at which the forecast distributions are to be evaluated. See Details.
- **integrated**: logical, (default=FALSE): If ‘integrated’ is TRUE, the integrated density (i.e. the value of the cumulative distribution function) is returned, otherwise the value of the density is returned.

Details

If you want to evaluate each forecast distribution function at the same x-values, a matrix with one row can be provided, e.g. ‘x = matrix(c(-1, 0, 1), nrow=1)’

If the N individual forecast distributions are to be evaluated at different x-values, a matrix with N rows must be provided, where N is the number of time instances.

To calculate the PIT values for the dressed ensemble and observations ‘obs’, use ‘GetDensity(dressed.ens, x = matrix(obs, ncol=1), integrated=TRUE)’

Value

The function returns a matrix, whose rows correspond to the individual ensemble forecasts and whose columns correspond to the values provided by the argument ‘x’.

See Also

DressEnsemble, DressCrps, DressIgn, PlotDressedEns, FitAkdParameters

Examples

data(eurotempforecast)
dressed.ens <- DressEnsemble(ens)
# calculate each density at the same x-values
x1 <- matrix(seq(-3, 3, 0.1), nrow=1)
dens1 <- GetDensity(dressed.ens, x1)
# get the densities that the forecast
# distributions assign to the observations
x2 <- matrix(obs, ncol=1)
dens2 <- GetDensity(dressed.ens, x2)
# get the integrated densities that the forecast
# distributions assign to the observations (useful
# for constructing a PIT histogram)
pit <- GetDensity(dressed.ens, x2, integrated=TRUE)
PlotDressedEns

Plot a series forecast distributions of dressed ensembles

Description

Plot a series forecast distributions of dressed ensembles

Usage

PlotDressedEns(
  dressed.ens,
  add = FALSE,
  obs = NULL,
  plot.ens = FALSE,
  plot.ker = FALSE
)

Arguments


add  logical, default=FALSE. If TRUE, no new plotting device is created and everything is added to an existing device.

obs  A vector of length N, default=NULL. The verifying observations corresponding to the individual ensemble forecasts. If a vector of length N is provided (N = nrow(dressed.ens["ens"])), the values are added to the plot as markers.

plot.ens  logical, default=FALSE. If TRUE, the centers of the individual dressing kernels are indicated by markers.

plot.ker  logical, default=FALSE. If TRUE, the individual dressing kernels are plotted.

Value

none

See Also

DressEnsemble

Examples

data(eurotempforecast)
d.ens <- DressEnsemble(ens)
PlotDressedEns(d.ens, add=FALSE, obs=obs, plot.ens=FALSE, plot.ker=TRUE)
PlotRankhist

Plotting function for rank histograms

Description

Plots a rank histogram in different modes.

Usage

PlotRankhist(rank.hist, mode = "raw")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank.hist</td>
<td>A vector or rank counts.</td>
</tr>
<tr>
<td>mode</td>
<td>Either &quot;raw&quot; (default) or &quot;prob.paper&quot;. Whether to draw the raw rank histogram, or the rank histogram on probability paper. See Details.</td>
</tr>
</tbody>
</table>

Details

The plotting modes currently implemented are:

- raw (the default): A simple bar plot of the counts provided by the `rank.hist` argument.
- prob.paper: The individual counts given by `rank.hist` are transformed to their cumulative probabilities under the binomial distribution with parameters `N` and `1/K`, where `N=sum(rank.hist)` and `K=length(rank.hist)`. This transformation makes possible an assessment of the observed rank counts under the hypothesis of equally likely ranks. The y-axis on the left indicates the cumulative probabilities. The intervals on the right of the plot indicate central 90, 95, and 99 percent simultaneous confidence intervals. That is, if all ranks were equally likely on average, approximately 90 percent of all rank histograms would be completely contained in the 90 percent interval and approximately 10 percent of all rank histograms would have at least one bar that falls outside this interval.

References


See Also

Rankhist, TestRankhist

Examples

data(eurotempforecast)
rank.hist <- Rankhist(ens, obs)
PlotRankhist(rank.hist, mode="prob.paper")
Rankhist

Rank histogram for ensemble forecasts

Description

Calculate the rank histogram for an archive of ensemble forecasts and their corresponding verifying observations.

Usage

Rankhist(ens, obs, reduce.bins = 1, handle.na = "na.fail")

Arguments

- **ens**: matrix of dimension (N,K). An archive of K-member ensemble forecasts for N time instances.
- **obs**: vector of length N. The corresponding verifying observations.
- **reduce.bins**: number of adjacent bins that will be merged into one bin; has to be a divisor of K+1
- **handle.na**: how should missing values in ensemble and observation data be handled; possible values are 'na.fail' (fails if any data is missing) and 'use.complete' (only uses times where all ensemble members and obs are available); default: 'na.fail'

Value

a vector of length (K+1)/reduce.bins containing the rank counts

References


See Also

PlotRankhist, TestRankhist

Examples

data(eurotempforecast)
rh <- Rankhist(ens, obs)
ReliabilityDiagram  

Reliability diagram for probability forecasts

Description

Reliability diagram for probability forecasts

Usage

ReliabilityDiagram(
  probs,
  obs,
  bins = 10,
  nboot = 500,
  plot = FALSE,
  plot.refin = TRUE,
  cons.probs = 0.95,
  attributes = FALSE,
  handle.na = c("na.fail", "use.pairwise.complete")
)

Arguments

probs    vector of N probability forecasts for the event obs=1
obs      vector of N binary observations, event/no event are coded as 0/1
bins     binning to estimate the calibration function (see Details), default: 10
nboot    number of bootstrap resamples to calculate the consistency bars, default: 500
plot     logical, whether to plot the reliability diagram, default: FALSE
plot.refin    Whether to add the frequency distribution of the forecasts to the reliability dia-
               gram. default: TRUE
cons.probs    The width of the consitency intervals. default: 0.95. Set to NA for no consis-
               tency bars.
attributes    locical, whether attributes lines are included in the diagram. default: FALSE
handle.na     how should missing values be handled; possible values are 'na.fail' and 'use.pairwise.complete';
               default: 'na.fail'

Details

To estimate the reliability curve, the unit line is categorised into discrete bins, provided by the
'bins' argument. If 'bins' is a single number, it specifies the number of equidistant bins. If 'bins' is
a vector of values between zero and one, these values are used as the bin-breaks.
ScoreDiff

Value

A data.frame with nrows equal to the number of bins (given by the ‘bins’ argument), with columns: average forecast probability per bin, conditional event frequency per bin, lower and upper limit of the consistency bar per bin, number of forecast probabilities per bin, lower and upper bin limit.

References


Examples

data(eurotempforecast)
p <- rowMeans(ens.bin)
ReliabilityDiagram(p, obs.bin, plot=TRUE)

ScoreDiff Calculate average score difference and assess uncertainty

Description

Calculate the difference (mean score of the reference forecast) minus (mean score of the forecast). Uncertainty is assessed by the Diebold-Mariano test for equality of predictive accuracy.

Usage

ScoreDiff(
    scores,
    scores.ref,
    N.eff = NA,
    conf.level = 0.95,
    handle.na = "na.fail"
)

Arguments

scores vector of verification scores
scores.ref vector of verification scores of the reference forecast, must be of the same length as ‘scores’
N.eff user-defined effective sample size to be used in hypothesis test and for confidence bounds; if NA, the length of ‘scores’ is used; default: NA
conf.level confidence level for the confidence interval; default = 0.95
handle.na how should missing values in scores vectors be handled; possible values are ‘na.fail’ and ‘use.pairwise.complete’; default: ‘na.fail’
Value
vector with mean score difference, estimated standard error of the mean, one-sided p-value of the Diebold-Mariano test, and the user-specified confidence interval

References

See Also
SkillScore

Examples
data(eurotempforecast)
ScoreDiff(EnsCrps(ens, obs), EnsCrps(ens[, 1:2], obs))

SkillScore Calculate a skill score and assess uncertainty.

Description
A skill score is defined as (mean score - mean reference score) / (perfect score - mean reference score). The skill score is zero if the mean score of the forecast equals the mean score of the reference forecast, and equals one if the mean score of the forecast equals the best possible score. Uncertainty is assessed by estimating the standard deviation of the skill score by propagation of uncertainty.

Usage
SkillScore(
  scores,
  scores.ref,
  N.eff = NA,
  score.perf = 0,
  handle.na = c("na.fail", "use.pairwise.complete")
)

Arguments
scores vector of verification scores
scores.ref vector of verification scores of the reference forecast, must be of the same length as 'scores'
N.eff user-defined effective sample size to be used to estimate the sampling uncertainty; if NA, the length of 'scores' is used; default: NA
score.perf a numeric constant, indicating the value that the score would assign to the perfect forecast
how should missing values in scores vectors be handled; possible values are 'na.fail' and 'use.pairwise.complete'; default: 'na.fail'

Value

vector with skill score and its estimated standard deviation

See Also

ScoreDiff

Examples

data(eurotempforecast)
SkillScore(EnsCrps(ens, obs), EnsCrps(ens[, 1:2], obs))

SpecsVerification - Forecast verification routines

Description

SpecsVerification - Forecast verification routines

Author(s)

: Stefan Siegert

SqErr - Calculate the squared error between forecast and observation

Description

Calculate the squared error between forecast and observation

Usage

SqErr(fcst, obs)

Arguments

fcst a N-vector representing N time instances of real-valued forecasts
obs a N-vector representing N time instances of real-valued observations

Value

numeric N-vector of squared errors
**TestRankhist**

**See Also**
AbsErr, ScoreDiff, SkillScore

**Examples**

```r
data(eurotempforecast)
mean(SqErr(rowMeans(ens), obs))
```

---

**TestRankhist**

**Statistical tests for rank histograms**

**Description**
Perform statistical tests related to the deviation from flatness of a rank histogram.

**Usage**

```r
TestRankhist(rank.hist)
```

**Arguments**

- `rank.hist` Vector of rank counts. Generated by function ‘Rankhist()’

**Details**

Given a vector of rank counts ‘x’, the Pearson Chi^2 statistic is calculated by

\[
\sum((x - \text{sum}(x)/\text{length}(x))^2 / (\text{sum}(x)/\text{length}(x)))
\]

and has a chi^2 distribution with (\text{length}(x)-1) degrees of freedom if every rank is equally likely on average. The Jolliffe-Primo test statistics are calculated by projecting the vector

\[
(x-\text{sum}(x)/\text{length}(x)) / \sqrt{\text{sum}(x)/\text{length}(x)}
\]

onto a linear, respectively squared contrast, i.e. a linear and quadratic function defined over the index set 1: length(x), who are mutually orthogonal, whose elements sum to zero, and whose squared elements sum to one. The projections independently have chi^2 distributions with 1 degree of freedom under the null hypothesis of a flat rank histogram.

**Value**

A dataframe whose columns refer to the Pearson Chi^2 statistic, the Jolliffe-Primo test statistic for slope, and the Jolliffe-Primo test statistic for convexity. The rows refer to the actual test statistic and its p-value under the null hypothesis of a flat rank histogram.

**References**

Pearson K. (1900): X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Phil. Mag. Series 5, 50(302) doi: 10.1080/14786440009463897

See Also

Rankhist, PlotRankhist

Examples

data(eurotempforecast)
rh <- Rankhist(ens, obs)
TestRankhist(rh)
Index

*Topic datasets
eurotempforecast, 27

AbsErr, 3
Auc, 3
auc_cpp, 6
AucDiff, 4
aucdiff_cpp, 6

BrierDecomp, 7
BrierScoreDecomposition (BrierDecomp), 7

ClimEns, 8
Corr, 9
CorrDiff, 10

Detrend, 11
DressCrps, 11
dresscrps_cpp, 14
DressCrpsDiff, 12
DressCrpss, 13
DressEnsemble, 14
DressIgn, 16
DressIgnDiff, 17

eurotempforecast, 27
EnsBrier, 17
EnsBrierDiff, 18
EnsBrierSs, 19
EnsCorr, 20
EnsCrps, 21
enscrps_cpp, 23
EnsCrpsDiff, 22
EnsCrpss, 22
EnsQs, 23
EnsRps, 24
EnsRpsDiff, 25
EnsRpss, 26
eurotempforecast, 27

FairBrier (EnsBrier), 17
FairBrierDiff, 28
FairBrierSs, 28
FairCrps (EnsCrps), 21
FairCrpsDiff, 29
FairCrpss, 30
FairQs (EnsQs), 23
FairRps (EnsRps), 24
FairRpsDiff, 30
FairRpss, 31
FitAkdaParameters, 32

GaussCrps, 33
GaussCrpsDiff, 34
GaussCrpss, 34
GenerateToyData, 35
GetDensity, 36

obs (eurotempforecast), 27

PlotDressedEns, 38
PlotRankhist, 39

Rankhist, 40
ReliabilityDiagram, 41

ScoreDiff, 42
SkillScore, 43
SpecsVerification, 44
SqErr, 44

TestRankhist, 45