# Package ‘easyVerification’

August 15, 2023

**Title**  Ensemble Forecast Verification for Large Data Sets

**Version**  0.4.5

**Description**  Set of tools to simplify application of atomic forecast verification metrics for (comparative) verification of ensemble forecasts to large data sets. The forecast metrics are imported from the ‘SpecsVerification’ package, and additional forecast metrics are provided with this package. Alternatively, new user-defined forecast scores can be implemented using the example scores provided and applied using the functionality of this package.

**Depends**  R (>= 3.0), SpecsVerification (>= 0.5), stats, utils

**Imports**  pbapply, Rcpp (>= 0.12.9)

**Suggests**  testthat, knitr, rmarkdown, parallel, R.rsp, verification

**LinkingTo**  Rcpp

**License**  GPL-3

**Copyright**  MeteoSwiss


**VignetteBuilder**  rmarkdown, R.rsp

**RoxygenNote**  7.2.3

**Encoding**  UTF-8

**Config/build/clean-inst-doc**  FALSE

**BugReports**  https://github.com/jonasbhend/easyVerification/issues

**NeedsCompilation**  yes

**Author**  MeteoSwiss [aut, cph], Jonas Bhend [cre], Jacopo Ripoldi [ctb], Claudia Mignani [ctb], Irina Mahlstein [ctb], Rebecca Hiller [ctb], Christoph Spirig [ctb],
Mark Liniger [ctb],
Andreas Weigel [ctb],
Joaquín Bedia Jimenez [ctb],
Matteo De Felice [ctb],
Stefan Siegert [ctb],
Katrin Sedlmeier [ctb]

Maintainer Jonas Bhend <jonas.bhend@meteoswiss.ch>
Repository CRAN
Date/Publication 2023-08-15 05:20:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>changearg</td>
<td>2</td>
</tr>
<tr>
<td>climFairRpss</td>
<td>3</td>
</tr>
<tr>
<td>convert2prob</td>
<td>4</td>
</tr>
<tr>
<td>count2prob</td>
<td>5</td>
</tr>
<tr>
<td>easyVerification</td>
<td>6</td>
</tr>
<tr>
<td>Ens2AFC</td>
<td>7</td>
</tr>
<tr>
<td>EnsCorr</td>
<td>8</td>
</tr>
<tr>
<td>EnsError</td>
<td>8</td>
</tr>
<tr>
<td>EnsErrors</td>
<td>9</td>
</tr>
<tr>
<td>EnsIgn</td>
<td>10</td>
</tr>
<tr>
<td>EnsRoca</td>
<td>11</td>
</tr>
<tr>
<td>EnsSprErr</td>
<td>12</td>
</tr>
<tr>
<td>FairSprErr</td>
<td>13</td>
</tr>
<tr>
<td>generateRef</td>
<td>14</td>
</tr>
<tr>
<td>size</td>
<td>15</td>
</tr>
<tr>
<td>toymodel</td>
<td>16</td>
</tr>
<tr>
<td>veriApply</td>
<td>17</td>
</tr>
<tr>
<td>veriUnwrap</td>
<td>20</td>
</tr>
<tr>
<td>weisheimer</td>
<td>21</td>
</tr>
</tbody>
</table>

Index 23

---

### changearg

**Change Function Default Arguments**

**Description**

Override default arguments of functions. This functionality is used to deal with the updated default representation in the SpecsVerification package (>= v0.5).

**Usage**

```r
c Changearg(FUN, ...)
```
climFairRpss

Arguments

FUN
name of function
...
arguments to be overridden (e.g. format = 'member')

climFairRpss Calculate Fair Ranked Probability Skill Score Against Climatological Reference Forecast.

Description

Calculate the fair ranked probability skill score (fair RPSS) between an ensemble forecasts and a climatological reference forecast derived from the observations. The categories of the climatological reference forecast have been defined based on the distribution of the observations and the exact forecast probabilities are known. The 'fair' correction therefore should not be applied to the reference forecast.

Usage

climFairRpss(ens, ens.ref, obs, format = c("category", "member"))

Arguments

ens N*K matrix. ens[i,j] is the number of ensemble members that predict category j at time i.
ens.ref N*K matrix, similar to ens
obs N*K matrix. obs[i,j] = 1 if category j is observed at time i, 0 otherwise.
format additional argument for use with SpecsVerification >= 0.5. Do not change this argument manually (except when using climFairRpss, as standalone function).

Value

A list with the following elements: rpss|skillscore: The value of the skill score. sigma.rpss|skillscore.sd: The standard deviation of the skill score, approximated by propagation of uncertainty. Please note that the naming changes with the new version of SpecsVerification.

See Also

veriApply

Examples

tm <- toymodel()

## compute RPSS using veriApply
veriApply("climFairRpss", tm$fcst, tm$obs, prob = 1:2 / 3)
Description

convert2prob Converts the continuous ensemble forecast to counts of ensemble members per category. The categories can be defined relative to the ensemble distribution (using prob) or relative to absolute values for the category thresholds (using threshold, see details). prob2thresh converts the relative threshold to absolute thresholds for later processing. expandthresh expands the vector or matrix of thresholds to fit the input data.

Usage

convert2prob(
  x,
  prob = NULL,
  threshold = NULL,
  ref.ind = NULL,
  multi.model = FALSE
)

prob2thresh(x, prob, ref.ind = NULL, multi.model = FALSE)

expandthresh(threshold, x)

Arguments

x input vector or matrix
prob thresholds for categorical forecasts (defaults to NULL)
threshold absolute thresholds for categorical forecasts (defaults to NULL)
ref.ind list of forecast/obs instances to be used to estimate percentile thresholds
multi.model logical, are we dealing with initial condition (the default) or multi-model ensembles (see details)?

Details

In case both prob and threshold are set to NULL, the function returns the input x without modification. If prob is set, a matrix with the number of occurrences per class for a given quantile of the full distribution (e.g. temperature above/below the median). If threshold is set, the classes are defined based on the absolute value (e.g. temperature above/below 13 deg. C). Multiple classes are

Only certain formats of threshold and prob are supported. prob has to be a vector with percentile thresholds separating the different classes. threshold can be a vector, matrix or array with the first entry corresponding to the different classes, and the last to the different ensemble members (if present). Thereby, time/forecast varying thresholds can potentially be supplied (although I am not sure this is useful or needed).
count2prob

If `ref.ind` is specified, only the specified indices of the input variables are used to estimate the percentile thresholds (`prob`). If used with `threshold`, or without anything, `ref.ind` has no effect. If `multi.model = TRUE`, the relative thresholds supplied by `prob` are ensemble member specific, i.e. are estimated for each ensemble member separately. This is in particular applicable for multi-model ensembles with model dependent biases.

Value

Matrix of occurrences per class (i.e. the number of ensemble members per class, or an indicator for the observations)

See Also

veriApply

Examples

```r
tm <- toymodel()

## convert to tercile forecasts (only display first forecast and obs)
convert2prob(tm$fcst, prob = 1:2 / 3)[1, ,]
convert2prob(tm$obs, prob = 1:2 / 3)[1, ,]

## convert to category forecasts (smaller and larger than 1)
convert2prob(tm$fcst, threshold = 1)[1, ,]
convert2prob(tm$obs, threshold = 1)[1, ,]
```

```{r}

count2prob Count Ensemble Counts to Probabilities

Description

Using plotting positions as described in Wilks (2011), counts of occurrences per forecast category are converted to probabilities of occurrence. For ensembles of size 1 (e.g. verifying observations), the count vector is returned unaltered (corresponding to occurrence probabilities of 0 or 1).

Usage

count2prob(x, type = 3)

Arguments

x input matrix of counts from `convert2prob`

Argument

type selection of plotting positions (default to 3, see Types)

Value

Matrix of probabilities per category
Types

The types characterize the plotting positions as specified in Wilks (2011). The plotting positions are computed using the following relationship:

\[ p(x_i) = \frac{i + 1 - a}{n + 1 - a} \]

where \( i \) is the number of ensemble members not exceeding \( x \), and \( n \) is the number of ensemble members. The types are characterized as follows:

<table>
<thead>
<tr>
<th>type</th>
<th>description</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Weibull</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Bernard and Bos-Levenbach</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>Tukey</td>
<td>1/3</td>
</tr>
<tr>
<td>4</td>
<td>Gumbel</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Hazen</td>
<td>1/2</td>
</tr>
<tr>
<td>6</td>
<td>Cunnane</td>
<td>2/5</td>
</tr>
</tbody>
</table>

References


See Also

`convert2prob` for conversion of continuous forecasts to ensemble counts

Examples

tm <- toymodel()

```r
## convert to tercile forecasts (only display first forecast and obs)
count2prob(convert2prob(tm$fcst, prob = 1:2 / 3))[1, ]
count2prob(convert2prob(tm$obs, prob = 1:2 / 3))[1, ]
```

Description

Set of tools to simplify application of atomic forecast verification metrics for (comparative) verification of ensemble forecasts to large data sets. The forecast metrics are imported from the ’SpecsVerification’ package, and additional forecast metrics are provided with this package. Alternatively, new user-defined forecast scores can be implemented using the example scores provided and applied using the functionality of this package.
Ens2AFC

**Generalized Discrimination Score**

---

**Description**
Computes the generalized discrimination score for ensemble forecasts after (Weigel and Mason, 2011).

**Usage**

```
Ens2AFC(ens, obs, ...)  
rank.ensembles(ens)
```

**Arguments**
- `ens`: n x m matrix of n forecasts for m ensemble members
- `obs`: vector of n verifying observations
- `...`: additional arguments not used in function (for compatibility)

**Details**
This function computes the generalized discrimination score for ensemble forecasts with continuous observations as described in Weigel and Mason (2011).

**References**

**See Also**

`veriApply`

**Examples**

```r
tm <- toymodel()
Ens2AFC(tm$fcst, tm$obs)
```
EnsCorr

**Correlation with Ensemble Mean**

**Description**

Computes the ensemble mean correlation (Pearson) with the verifying observations.

**Usage**

EnsCorr(ens, obs)

**Arguments**

- **ens**: n x k matrix of n forecasts from k ensemble members
- **obs**: n verifying observations

**See Also**

veriApply

**Examples**

tm <- toymodel()

## compute correlation directly
EnsCorr(tm$fcst, tm$obs)

## compute correlation using veriApply
veriApply("EnsCorr", tm$fcst, tm$obs)

EnsError

**Ensemble Mean Error**

**Description**

Computes various ensemble mean error scores. EnsMe computes the mean error, EnsMae the mean absolute error, EnsMse the mean squared error, and EnsRmse the square root of the mean squared error (for consistency with the veri package).
EnsErrors

Usage

EnsError(ens, obs, type)
EnsMe(ens, obs)
EnsMae(ens, obs)
EnsMse(ens, obs)
EnsRmse(ens, obs)

Arguments

ens n x k matrix of n forecasts from k ensemble members
obs n verifying observations
type specifying what error metric to compute, one of [me, mae, mse, rmse]

See Also

veriApply, EnsErrors

Examples

# forecast and observations
tm <- toymodel()

# compute the mean bias
EnsError(tm$fcst, tm$obs, type = "me")
# equivalently
EnsMe(tm$fcst, tm$obs)

EnsErrors

Ensemble Mean Error Skill scores

Description

Computes various ensemble mean error skill scores. EnsMaess computes the mean absolute error, EnsMsess the mean squared error, and EnsRmsess the square root of the mean squared error.

Usage

EnsErrors(ens, ens.ref, obs, type)
EnsMaess(ens, ens.ref, obs)
EnsMsess(ens, ens.ref, obs)
EnsRmsess(ens, ens.ref, obs)
Arguments
ens n x k matrix of n forecasts from k ensemble members
ens.ref n x l matrix of m reference forecasts from l ensemble members
obs n verifying observations
type specifying what error metric to compute, one of [me, mae, mse, rmse]

See Also
veriApply, EnsError

Examples
tm <- toymodel()

## compute RMSE skill score against reference forecast with a bias of +2
EnsErrorss(ens = tm$fcst, ens.ref = tm$fcst + 2, obs = tm$obs, type = "rmse")

## compute skill score using veriApply
veriApply("EnsRmsess", fcst = tm$fcst, obs = tm$obs, fcst.ref = tm$fcst + 2)

EnsIgn

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ignorance Score</strong></td>
<td></td>
</tr>
</tbody>
</table>

Description

Computes the ignorance score EnsIgn and skill score EnsIgnss for an interpretation of the ensemble as a probability forecast

Usage

EnsIgn(ens, obs, type = 3, ...)
EnsIgnss(ens, ens.ref, obs, type = 3)

Arguments
ens n x j matrix of n probability forecasts for j categories
obs n x j matrix of occurrence of n verifying observations in j categories
type selection of plotting positions to convert ensemble counts to probabilities (default to 3, see count2prob
... additional arguments for consistency with other functions (not used)
ens.ref n x j matrix of n probability forecasts for j categories
References


See Also

veriApply, count2prob

Examples

tm <- toymodel()

## compute ignorance score for tercile forecasts
veriApply("EnsIgn", fcst = tm$fcst, obs = tm$obs, prob = 1:2 / 3)

## compute skill score
veriApply("EnsIgnss", fcst = tm$fcst, obs = tm$obs, prob = 1:2 / 3)

EnsRoca

Area Under the ROC Curve

Description

Computes the area under the ROC curve given the observations. EnsRoca computes the Area Under the Curve (AUC). For ease of interpretation, EnsRocss converts the AUC to the range from -1 to 1 with zero indicating a forecast with no discrimination.

Usage

EnsRoca(ens, obs, use.easy = FALSE)
EnsRocss(ens, obs, use.easy = FALSE)

Arguments

ens n x j matrix of n probability forecasts for j categories
obs n x j matrix of occurrence of n verifying observations in j categories
use.easy logical, should implementation of standard errors as implemented in easyVerification be used (see below)?
EnsSprErr

Standard Error

If used with SpecsVerification >= 0.5, the standard errors as implemented in the function SpecsVerification::Auc are used.

If use.easy = TRUE or when used with an older version of SpecsVerification, the standard error $\sigma$ of the ROC area skill score is given by the following formula after Broecker (2012).

$$\sigma^2 = \frac{1}{3} \left( \frac{1}{N_0} + \frac{1}{N_1} + \frac{1}{N_0N_1} \right)$$

Where $\sigma$ is the standard error, $N_1$ the number of events, and $N_0$ the number of non-events in category 1.

References


See Also

veriApply, EnsRocss

Examples

tm <- toymodel()

## compute ROC area for tercile forecasts using veriApply
veriApply("EnsRoca", fcst = tm$fcst, obs = tm$obs, prob = 1:2 / 3)

EnsSprErr

Spread to Error Ratio

Description

Computes the spread to error ratio (SPR) for probabilistic forecasts - not unlike the functions in SpecsVerification. SPR > 1 indicates overdispersion (underconfidence), whereas SPR < indicates overconfidence in the forecasts.

Usage

EnsSprErr(ens, obs)

Arguments

ens n x k matrix of n forecasts for k ensemble members
obs vector with n verifying observations
Details

Here we define the spread-error rate as the square root of the ratio of mean ensemble variance to the mean squared error of the ensemble mean with the verifying observations.

See Also

veriApply, FairSprErr

Examples

```r
tm <- toymodel()
EnsSprErr(tm$fcst, tm$obs)

## compute spread to error ratio using veriApply
veriApply("EnsSprErr", fcst = tm$fcst, obs = tm$obs)
```

---

**FairSprErr**  
*Fair Spread to Error Ratio*

Description

Compute the spread to error ratio (SPR) for probabilistic forecasts - not unlike the functions in SpecsVerification. SPR > 1 indicates overdispersion (underconfidence), whereas SPR < 1 indicates overconfidence in the forecasts.

Usage

```
FairSprErr(ens, obs)
```

Arguments

- `ens`  
n x k matrix of n forecasts for k ensemble members
- `obs`  
vector with n verifying observations

Details

Here we define the spread-error rate as the square root of the ratio of mean ensemble variance to the mean squared error of the ensemble mean with the verifying observations. We inflate the intra ensemble sample variance to account for the finite ensemble size as in Weigel (2011).

References

See Also

\texttt{veriApply, FairSprErr}

Examples

tm <- toymodel()
FairSprErr(tm$fcst, tm$obs)

## compute spread to error ratio using \texttt{veriApply}
veriApply("FairSprErr", fcst = tm$fcst, obs = tm$obs)

## compare with 'unfair' spread to error ratio
veriApply("EnsSprErr", fcst = tm$fcst, obs = tm$obs)

\begin{verbatim}
generateRef
Generate Probabilistic Climatological Ensemble Forecast from Observations
\end{verbatim}

Description

To generate reference ensemble forecasts for forecast evaluation based on the available observations, \texttt{indRef} implements the out-of-sample or in-sample protocol to be used and \texttt{generateRef} produces the corresponding ensemble forecast given the actual observations.

Usage

\begin{verbatim}
indRef(
  nfcst,
  type = c("none", "forward", "crossval", "block"),
  indices = 1:nfcst,
  blocklength = 1
)
\end{verbatim}

generateRef(obs, ind)

Arguments

\begin{tabular}{ll}
  \textbf{nfcst} & number of forecast instances to be produce \\
  \textbf{type} & type of out-of-sample protocol to be applied (see below) \\
  \textbf{indices} & Subset of the observations / forecast times to be used for reference forecasts \\
  \textbf{blocklength} & for cross-validation and split-sample \\
  \textbf{obs} & vector of observations \\
  \textbf{ind} & list or matrix of dimension (n x nref) of indices of the observations to be used for each forecast instance \\
\end{tabular}
Value

ind

A list of indices to be used for each forecast from 1 to nfcst

Cross-validation

Leave-one-out and leave-n-out cross-validation reference forecasts can be produced by setting type = "crossval". By default, the blocklength is set to 1, but moving blocks of length n can be specified by setting blocklength = n.

Split sample

In contrast to type="crossval", type="block" is used for split-sample validation with non-overlapping blocks of length blocklength retained for validation.

Forward

Correspondingly, reference forecasts that are only based on past (future) observations can be produced using type = "forward". For this, the first half of the reference forecasts only uses future information, i.e. observations 2:n for forecast 1, 3:n for 2 and so forth. The second half of the reference forecasts use only past observations, i.e. observations 1:(n-1) for forecast n, 1:(n-2) for n-1, etc.

Subsetting

In combination with the above, a subset of the observations can be specified for use as reference forecasts by providing the explicit indices of the observations to be used via indices=1:k. In combination with the forward method, all observations in indices will be used to construct the reference forecast for forecasts not included in indices (i.e. if nfcst > max(indices)).

---

size

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return dimension of array or length of vector.</td>
</tr>
</tbody>
</table>

Usage

size(x)

Arguments

x array or vector

See Also

veriApply
toymodel

Create Example Forecast-Observation Pairs

Description

This toy model lets you create forecast-observation pairs with specified ensemble and forecast size, correlation skill, and overconfidence (underdispersion) for application with the verification functionality provided as part of the easyVerification package.

Usage

toymodel(N = 35, nens = 51, alpha = 0.5, beta = 0)

toyarray(dims = c(10, 5), ...)

Arguments

N number of forecast instances
nens number of ensemble members
alpha nominal correlation skill of forecasts
beta overconfidence parameter (see details)
dims independent (e.g. spatial) dimensions for the toy model
... additional arguments passed to toymodel

Details

The toy model is the TM2 model as introduced by Weigel and Bowler (2009) with a slight modification to allow for forecasts with negative correlation skill. In this toy model, the observations \( x \) and forecasts \( f_i \) are defined as follows:

\[
x = \mu_x + \epsilon_x \\
f_i = \alpha/|\alpha| \mu_x + \epsilon_\beta + \epsilon_i
\]

where

\[
\begin{align*}
\mu_x & \sim N(0, \alpha^2) \\
\epsilon_x & \sim N(0, 1 - \alpha^2) \\
\epsilon_\beta & \sim N(0, \beta^2) \\
\epsilon_i & \sim N(0, 1 - \alpha^2 - \beta^2) \\
\alpha^2 & \leq 1 \\
0 & \leq \beta \leq 1 - \alpha^2
\end{align*}
\]
Note

This toy model is intended to provide example forecast observation pairs and not to serve as a conceptual model to study real forecasts. For models to do the latter, please refer to Siegert et al. (2015).

References


Examples

```r
## compute the correlation for a toy forecast with default parameters
tm <- toyarray()
f.corr <- veriApply("EnsCorr", fcst = tm$fcst, obs = tm$obs)
```

---

**veriApply**

*Apply Verification Metrics to Large Datasets*

**Description**

This wrapper applies verification metrics to arrays of forecast ensembles and verifying observations. Various array-based data formats are supported. Additionally, continuous forecasts (and observations) are transformed to category forecasts using user-defined absolute thresholds or percentiles of the long-term climatology (see details).

**Usage**

```r
veriApply(
    verifun,
    fcst,
    obs,
    fcst.ref = NULL,
    tdim = length(dim(fcst)) - 1,
    ensdim = length(dim(fcst)),
    prob = NULL,
    threshold = NULL,
    strategy = "none",
    na.rm = FALSE,
    fracmin = 0.8,
    nmin = NULL,
    parallel = FALSE,
```
maxncpus = 16,
ncpus = NULL,
...
)

Arguments

verifun   Name of function to compute verification metric (score, skill score)
fcest     array of forecast values (at least 2-dimensional)
obs       array or vector of verifying observations
fcest.ref array of forecast values for the reference forecast (skill scores only)
tdim      index of dimension with the different forecasts
ensdim    index of dimension with the different ensemble members
prob      probability threshold for category forecasts (see below)
threshold absolute threshold for category forecasts (see below)
strategy  type of out-of-sample reference forecasts or namelist with arguments as in indRef
          or list of indices for each forecast instance
na.rm     logical, should incomplete forecasts be used?
fracmin   fraction of forecasts that are not-missing for forecast to be evaluated. Used to
determine nmin when is.null(nmin)
nmin      number of forecasts that are not-missing for forecast to be evaluated. If both
          nmin and fracmin are set, nmin takes precedence
parallel   logical, should parallel execution of verification be used (see below)?
maxncpus  upper bound for self-selected number of CPUs
ncpus     number of CPUs used in parallel computation, self-selected number of CPUs is
          used when is.null(ncpus) (the default).
...       additional arguments passed to verifun

List of functions to be called

The selection of verification functions supplied with this package and as part of SpecsVerification
can be enquired using ls(pos='package:easyVerification') and ls(pos='package:SpecsVerification')
respectively. Please note, however, that only some of the functions provided as part of SpecsVerification
can be used with veriApply. Functions that can be used include for example the (fair) ranked prob-
ability score EnsRps, FairRps, and its skill score EnsRpss, FairRpss, or the continuous ranked
probability score EnsCrps, etc.

Conversion to category forecasts

To automatically convert continuous forecasts into category forecasts, absolute (threshold) or
relative thresholds (prob) have to be supplied. For some scores and skill scores (e.g. the ROC
area and skill score), a list of categories will be supplied with categories ordered. That is, if prob =
1:2/3 for tercile forecasts, cat1 corresponds to the lower tercile, cat2 to the middle, and cat3 to
the upper tercile.
Absolute and relative thresholds can be supplied in various formats. If a vector of thresholds is supplied with the threshold argument, the same threshold is applied to all forecasts (e.g. lead times, spatial locations). If a vector of relative thresholds is supplied using prob, the category boundaries to be applied are computed separately for each space-time location. Relative boundaries specified using prob are computed separately for the observations and forecasts, but jointly for all available ensemble members.

Location specific thresholds can also be supplied. If the thresholds are supplied as a matrix, the number of rows has to correspond to the number of forecast space-time locations (i.e. same length as `length(fcst)/prod(dim(fcst)[c(tdim, ensdim)])`). Alternatively, but equivalently, the thresholds can also be supplied with the dimensionality corresponding to the obs array with the difference that the forecast dimension in obs contains the category boundaries (absolute or relative) and thus may differ in length.

### Out-of-sample reference forecasts

strategy specifies the set-up of the climatological reference forecast for skill scores if no explicit reference forecast is provided. The default is strategy = "none", that is all available observations are used as equiprobable members of a reference forecast. Alternatively, strategy = "crossval" can be used for leave-one-out crossvalidated reference forecasts, or strategy = "forward" for a forward protocol (see indRef).

Alternatively, a list with named parameters corresponding to the input arguments of indRef can be supplied for more fine-grained control over standard cases. Finally, also a list with observation indices to be used for each forecast can be supplied (see generateRef).

### Parallel processing

Parallel processing is enabled using the parallel package. Parallel verification is using ncpus FORK clusters or, if ncpus are not specified, one less than the autodetected number of cores. The maximum number of cores used for parallel processing with auto-detection of the number of available cores can be set with the maxncpus argument.

Progress bars are available for non-parallel computation of the verification metrics. Please note, however, that the progress bar only indicates the time of computation needed for the actual verification metrics, input and output re-arrangement is not included in the progress bar.

### Note

If the forecasts and observations are only available as category probabilities (or ensemble counts as used in SpecsVerification) as opposed to as continuous numeric variables, veriApply cannot be used but the atomic verification functions for category forecasts have to be applied directly.

Out-of-sample reference forecasts are not fully supported for categorical forecasts defined on the distribution of forecast values (e.g. using the argument prob). Whereas only the years specified in strategy are used for the reference forecasts, the probability thresholds for the reference forecasts are defined on the collection of years specified in strategy.

### See Also

- convert2prob for conversion of continuous into category forecasts (and observations)
Examples

```r
tm <- toyarray()
f.me <- veriApply("EnsMe", tm$fcst, tm$obs)

## find more examples and instructions in the vignette
## Not run:
devtools::install_github("MeteoSwiss/easyVerification", build_vignettes = TRUE)
library("easyVerification")
vignette("easyVerification")

## End(Not run)
```

---

**veriUnwrap**

Unwrap Arguments and Hand Over to Verification Function

### Description

Decomposes input arguments into forecast, verifying observations, and reference forecast and hands these over to the function provided.

### Usage

```r
veriUnwrap(
  x, 
  verifun, 
  nind = c(nens = ncol(x) - 1, nref = 0, nobs = 1, nprob = 0, nthresh = 0),
  ref.ind = NULL,
  ...
)
```

### Arguments

- **x**: n x k + 1 matrix with n forecasts of k ensemble members plus the verifying observations
- **verifun**: character string with function name to be executed
- **nind**: named vector with number of ensemble members, ensemble members of reference forecasts, observations (defaults to 1), probability or absolute thresholds (see details)
- **ref.ind**: list with specifications for the reference forecast (see details)
- **...**: additional arguments passed on to verifun
Details

Forecast verification metrics are only computed for forecasts with non-missing verifying observation and at least one non-missing ensemble member. Metrics for all other forecasts are set to missing. For aggregate metrics (e.g. skill scores) the metric is computed over non-missing observation/forecast pairs only.

For computation of skill scores, reference forecasts can be provided. That is, the first nens columns of x contain the forecasts, the (nens + 1):(ncol(x) - 1) following columns contain the reference forecast, and the final column contains the observations. If no reference forecast is provided (i.e. ncol(x) == nens + 1), a climatological forecast is constructed from the n verifying observations.

The elements of vector nind have to be named with nens containing the number of ensemble members, nref the number of ensemble members in the reference forecast for skill scores, nobs the number of observations (only one supported), nprob the number of probability thresholds, and nthresh the number of absolute threshold for conversion of continuous forecasts to category forecasts.

ref.ind specifies the set-up of the climatological reference forecast for skill scores if no explicit reference forecast is provided (see indRef). Also, ref.ind is used to determine the baseline to estimate the percentile-based category boundaries to convert continuous forecasts to category probabilities.

Note

Out-of-sample reference forecasts are now fully supported.

See Also

veriApply

weisheimer

Compute Reliability Categories as in Weisheimer et al. (2014)

Description

This function implements the reliability categorisation for forecasts of binary events as documented in Weisheimer et al. (2014). It has only been implemented for category forecasts with categories defined relative to the forecast and observed climatological distribution (i.e. without systematic bias).

Usage

weisheimer(
  ens,
  obs,
  pthresh = 2/3,
  nboot = 100,
  brier.thresholds = seq(0, 1, 0.2),
  ...
)
Arguments

- **ens**: $n \times k$ matrix of $n$ forecasts from $k$ ensemble members
- **obs**: $n$ verifying observations
- **pthresh**: probability threshold to convert to category forecasts. If negative, event falling below threshold is used, else, event above threshold is used.
- **nboot**: number of bootstrap replicates to estimate 75 percent confidence interval
- **brier.thresholds**: Thresholds used to bin the forecasts (see `brier`)
- **...**: additional arguments for compatibility with other scores
Index

* utilities
  convert2prob, 4
  count2prob, 5
  generateRef, 14
  size, 15
  toymodel, 16
  veriApply, 17
  weisheimer, 21

brier, 22
changearg, 2
climFairRpss, 3
convert2prob, 4, 5, 6, 19
count2prob, 5, 10, 11
easyVerification, 6
Ens2AFC, 7
EnsCorr, 8
EnsCrops, 18
EnsError, 8, 10
EnsErrorss, 9, 9
EnsIgn, 10
EnsIgnss (EnsIgn), 10
EnsMae (EnsError), 8
EnsMaess (EnsErrorss), 9
EnsMe (EnsError), 8
EnsMse (EnsError), 8
EnsMsess (EnsErrorss), 9
EnsRmse (EnsError), 8
EnsRmsess (EnsErrorss), 9
EnsRoca, 11
EnsRocss, 12
EnsRocss (EnsRoca), 11
EnsRps, 18
EnsRpss, 18
EnsSprErr, 12
expandthresh (convert2prob), 4

FairRpss, 18
FairSprErr, 13, 13, 14
generateRef, 14, 19
indRef, 18, 19, 21
indRef (generateRef), 14
oldEnsRoca (EnsRoca), 11
parallel, 19
prob2thresh (convert2prob), 4
rank.ensembles (Ens2AFC), 7
size, 15
toyarray (toymodel), 16
toymodel, 16, 16
veriApply, 3, 5, 7–15, 17, 18, 21
veriUnwrap, 20
weisheimer, 21