Package ‘easyVerification’

October 15, 2015

Title Ensemble Forecast Verification for Large Datasets
Version 0.1.8
Date 2015-10-15

Description Set of tools to simplify application of atomic forecast verification metrics for (comparative) verification of ensemble forecasts to large datasets. 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, stats, utils
Imports RCurl, pbapply
Suggests testthat, knitr, parallel, R.rsp
License GPL-3
Copyright MeteoSwiss
LazyData true

URL http://www.meteoswiss.ch

BugReports https://github.com/MeteoSwiss/easyVerification/issues

VignetteBuilder knitr, R.rsp

Author Jonas Bhend [aut, cre],
Jacopo Ripoldi [ctb],
Claudia Mignani [ctb],
Irina Mahlstein [ctb],
Rebecca Hiller [ctb],
Christoph Spirig [ctb],
Mark Liniger [ctb],
Andreas Weigel [ctb],
Joaquin Bedia Jimenez [ctb],
Matteo De Felice [ctb]

Maintainer Jonas Bhend <jonas.bhend@meteoswiss.ch>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-15 18:09:15
R topics documented:

convert2prob ................................................................. 2
count2prob ................................................................. 3
easyVerification .......................................................... 4
Ens2AFC ................................................................. 4
EnsCorr ................................................................. 5
EnsError ................................................................. 6
EnsErrors ............................................................. 7
EnsRoca ................................................................. 8
EnsRoess ............................................................... 8
EnsSprErr .............................................................. 9
FairSprErr ............................................................. 10
toymodel ............................................................... 11
veriApply ............................................................... 13
veriUnwrap ............................................................ 14

Index 16

| convert2prob | Convert to probability / categorical forecast |

Description

Convert to probability / categorical forecast

Usage

convert2prob(x, prob = NULL, threshold = NULL)

Arguments

x input vector or matrix
prob thresholds for categorical forecasts (defaults to NULL)
threshold absolute thresholds for categorical forecasts (defaults to NULL)

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 supported.

Value

Matrix of occurrences per class (i.e. the number of ensemble members per class, or an indicator for the observations)
count2prob

See Also

veriApply

Examples

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,]

count2prob

Convert 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

Arguments

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

tm < toymodel()

## 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,]

easyVerification

description

easyVerification.

Ensemble
generalized discrimination score

Description

Computes the generalized discrimination score for ensemble forecasts

Usage

Ens2AFC(ens, obs, ...)

rank.ensembles(ens)
EnsCorr

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

tm <- toymodel()
Ens2AFC(tm$fcst, tm$obs)

<table>
<thead>
<tr>
<th>EnsCorr</th>
<th>Ensemble mean correlation</th>
</tr>
</thead>
</table>

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

```r
tm <- toymodel()

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

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

---

**EnsError**

*Compute various ensemble mean error metrics*

**Description**

`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).

**Usage**

```r
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**

```r
# forecast and observations
tm <- toymodel()

# compute the mean bias
EnsError(tm$fctst, tm$obs, type='me')
# equivalently
EnsMe(tm$fctst, tm$obs)
```
Compute various ensemble mean error skill scores

Description

EnsMess computes the mean error, EnsMaess the mean absolute error, EnsMsess the mean squared error, and EnsRmsess the square root of the mean squared error (for consistency with the veri package).

Usage

EnsErrorss(ens, ens.ref, obs, type)
EnsMess(ens, ens.ref, obs)
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)
**EnsRoca**

*Area under the ROC curve*

**Description**
Computes the area under the ROC curve given the observations.

**Usage**
EnsRoca(ens, obs)

**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

**See Also**
veriApply, EnsRocss

**Examples**

```R
tm <- toymodel()

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

---

**EnsRocss**

*Skill score for area under the ROC curve*

**Description**
Computes the skill score for the area under the ROC curve compared to an arbitrary reference forecast (generally climatological forecast).

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

**Arguments**
- **ens**: n x j matrix of n probability forecasts for j categories
- **ens.ref**: n x j matrix of reference forecast for j categories
- **obs**: n x j matrix of occurrence of n verifying observations in j categories
Details

For the traditional ROC area skill score where the reference forecast has zero association with the observations, 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 $i$. Please note the factor 2 difference to the formulation of the standard error for the ROC area in the original manuscript due to the conversion of the ROC area to the ROC area skill score.

Value

A list with the ROC area skill score for forecast category $i$ in cati and the standard deviation of this skill score for category $i$ in cati.sigma if a reference forecast with zero association is used (see details).

References


See Also

veriApply, EnsRoca

Examples

tm <- toymodel()

## compute ROC skill score for forecasts of x <= 0, 0 <= x < 1, and x > 1
## skill score is computed using climatological forecast as reference
veriApply("EnsRocss", tm$fcst, tm$obs, threshold=c(0,1))

EnsSprErr

Compute spread-error ratio

Description

Modular function that 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()
EnSprErr(tm$fcst, tm$obs)

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

---

**FairSprErr** *Fair spread-error ratio*

Description
Modular function that computes 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
```r
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).
toymodel

References


See Also

veriApply, FairSprErr

Examples

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

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

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

---

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:

$$x = \mu_x + \epsilon_x$$

$$f_i = \frac{\alpha}{|\alpha|}\mu_x + \epsilon_\beta + \epsilon_i$$

where

$$\mu_x \sim N(0, \alpha^2)$$

$$\epsilon_x \sim N(0, 1 - \alpha^2)$$

$$\epsilon_\beta \sim N(0, \beta^2)$$

$$\epsilon_i \sim N(0, 1 - \alpha^2 - \beta^2)$$

$$\alpha^2 \leq 1$$

$$0 \leq \beta \leq 1 - \alpha^2$$

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

A. Weigel and N. Bowler (2009). Comment on ‘Can multi-model combination really enhance the prediction skill of probabilistic ensemble forecasts?’.


Examples

```r
# compute the correlation for a toy forecast with default parameters
tm <- toyarray()
fcst <- veriApply("EnsCorr", fcst=tm$fcst, obs=tm$obs)
```
Apply verification metrics to large datasets

Description
This wrapper applies verification metrics to arrays of forecast ensembles and verifying observations. Various formats array-based 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
veriApply(verifun, fcst, obs, fcst.ref = NULL, tdim = length(dim(fcst)) - 1, ensdim = length(dim(fcst)), prob = NULL, threshold = NULL, na.rm = FALSE, parallel = FALSE, maxncpus = 16, ncpus = NULL, ...)

Arguments
- **verifun**: Name of function to compute verification metric (score, skill score)
- **fcst**: array of forecast values (at least 2-dimensional)
- **obs**: array or vector of verifying observations
- **fcst.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)
- **na.rm**: logical, should incomplete forecasts be used?
- **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

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 autodetection 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.
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 \(\text{length}(\text{fcst})/\text{prod}(\text{dim}(\text{fcst})[c(tdim, ensdim)])\)). 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.

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.

See Also

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

Examples

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)
Description

decomposes input argument into forecast and verifying observations and hands these over to the function provided

Usage

veriUnwrap(x, verifun, nind = c(nens = ncol(x) - 1, nref = 0, nob = 1, nprob = 0, nthresh = 0), ...)

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)
...          additional arguments passed on to verifun

Details

Only forecasts with non-missing observation and complete ensembles are computed. 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) \n 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, nob 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.

See Also

veriApply
Index

*Topic utilities
  convert2prob, 2
  count2prob, 3
  toymodel, 11
  veriApply, 13

convert2prob, 2, 3, 4, 14
count2prob, 3
easyVerification, 4
easyVerification-package
  (easyVerification), 4
Ens2AFC, 4
EnsCorr, 5
EnsError, 6, 7
EnsErrorss, 6, 7
EnsMae (EnsError), 6
EnsMaess (EnsErrorss), 7
EnsMe (EnsError), 6
EnsMess (EnsErrorss), 7
EnsMse (EnsError), 6
EnsMsess (EnsErrorss), 7
EnsRmse (EnsError), 6
EnsRmsess (EnsErrorss), 7
EnsRoca, 8, 9
EnsRocss, 8, 8
EnsSprErr, 9
FairSprErr, 10, 10, 11
parallel, 13
rank.ensembles (Ens2AFC), 4
toarray (toymodel), 11
toymodel, 11, 11

veriApply, 3, 5–11, 13, 15
veriUnwrap, 14