Package 'scoringutils'

November 17, 2020

Title Utilities for Scoring and Assessing Predictions

Version 0.1.4

Description Combines a collection of metrics and proper scoring rules (Tilmann Gneiting & Adrian E Raftery (2007) <doi:10.1198/016214506000001437>) with an easy to use wrapper that can be used to automatically evaluate predictions. Apart from proper scoring rules functions are provided to assess bias, sharpness and calibration (Sebastian Funk, Anton Camacho, Adam J. Kucharski, Rachel Lowe, Rosalind M. Eggo, W. John Edmunds (2019) <doi:10.1371/journal.pcbi.1006785>) of forecasts. Several types of predictions can be evaluated: probabilistic forecasts (generally predictive samples generated by Markov Chain Monte Carlo procedures), quantile forecasts or point forecasts. Observed values and predictions can be either continuous, integer, or binary. Users can either choose to apply these rules separately in a vector / matrix format that can be flexibly used within other packages, or they can choose to do an automatic evaluation of their forecasts. This is implemented with 'data.table' and provides a consistent and very efficient framework for evaluating various types of predictions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports data.table, forcats, ggplot2, goftest, graphics, scoringRules, stats

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.1.1

URL https://github.com/epiforecasts/scoringutils

BugReports https://github.com/epiforecasts/scoringutils/issues

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Nikos Bosse [aut, cre] (<https://orcid.org/0000-0002-7750-5280>),
Sam Abbott [aut] (<https://orcid.org/0000-0001-8057-8037>),
Joel Hellewell [ctb] (<https://orcid.org/0000-0003-2683-0849>),
Sophie Meakins [ctb],
James Munday [ctb],
Katharine Sherratt [ctb],
Sebastian Funk [aut]

Maintainer Nikos Bosse <nikosbosse@gmail.com>

Repository CRAN

Date/Publication 2020-11-17 10:30:02 UTC

R topics documented:

`ae_median` *Absolute Error of the Median*

Description

Absolute error of the median calculated as

$$abs(true_value - median_prediction)$$

Usage

```
ae_median(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples. Alternatively, predictions can just be a vector of size n

Value

vector with the scoring values

Examples

```
true_values <- rnorm(30, mean = 1:30)
predicted_values <- rnorm(30, mean = 1:30)
ae_median(true_values, predicted_values)
```

`bias` *Determines bias of forecasts*

Description

Determines bias from predictive Monte-Carlo samples. The function automatically recognises, whether forecasts are continuous or integer valued and adapts the Bias function accordingly.

Usage

```
bias(true_values, predictions)
```

Arguments

true_values	A vector with the true observed values of size n
predictions	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples

Details

For continuous forecasts, Bias is measured as

$$B_t(P_t, x_t) = 1 - 2 * (P_t(x_t))$$

where P_t is the empirical cumulative distribution function of the prediction for the true value x_t. Computationally, $P_t(x_t)$ is just calculated as the fraction of predictive samples for x_t that are smaller than x_t.

For integer valued forecasts, Bias is measured as

$$B_t(P_t, x_t) = 1 - (P_t(x_t) + P_t(x_t + 1))$$

to adjust for the integer nature of the forecasts.

In both cases, Bias can assume values between -1 and 1 and is 0 ideally.

Value

vector of length n with the biases of the predictive samples with respect to the true values.

Author(s)

Nikos Bosse <nikosbosse@gmail.com>

References

The integer valued Bias function is discussed in Assessing the performance of real-time epidemic forecasts: A case study of Ebola in the Western Area region of Sierra Leone, 2014-15 Funk S, Camacho A, Kucharski AJ, Lowe R, Eggo RM, et al. (2019) Assessing the performance of real-time epidemic forecasts: A case study of Ebola in the Western Area region of Sierra Leone, 2014-15. PLOS Computational Biology 15(2): e1006785. https://doi.org/10.1371/journal.pcbi.1006785

Examples

```
## integer valued forecasts
true_values <- rpois(30, lambda = 1:30)
predictions <- replicate(200, rpois(n = 30, lambda = 1:30))
bias(true_values, predictions)

## continuous forecasts
true_values <- rnorm(30, mean = 1:30)
predictions <- replicate(200, rnorm(30, mean = 1:30))
```

```
bias(true_values, predictions)
```

`binary_example_data` *Binary Example Data*

Description

A toy dataset for a probability forecast of a binary outcome variable

Usage

```
binary_example_data
```

Format

A data.table with 120 rows and 5 variables:

id unique identifier for true observed values

true_value true observed values

model name of the model that generated the forecasts

horizon forecast horzin (e.g. 1 day ahead forecast)

prediction predicted probability that the corresponding true value will be 1

`brier_score` *Brier Score*

Description

Computes the Brier Score for probabilistic forecasts of binary outcomes.

Usage

```
brier_score(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	A vector with a predicted probability that true_value = 1.

Details

The Brier score is a proper score rule that assesses the accuracy of probabilistic binary predictions. The outcomes can be either 0 or 1, the predictions must be a probability that the true outcome will be 1.

The Brier Score is then computed as the mean squared error between the probabilistic prediction and the true outcome.

$$Brier_Score = \frac{1}{N} \sum_{t=1}^{n} (prediction_t - outcome_t)^2$$

Value

A numeric value with the Brier Score, i.e. the mean squared error of the given probability forecasts

Examples

```
true_values <- sample(c(0,1), size = 30, replace = TRUE)
predictions <- runif(n = 30, min = 0, max = 1)

brier_score(true_values, predictions)
```

---

continuous_example_data

Continuous Example Data

Description

A toy dataset for a probabilistic forecast of a continuous outcome variable

Usage

```
continuous_example_data
```

Format

A data frame with 6000 rows and 6 variables:

id unique identifier for true observed values

model name of the model that generated the forecasts

horizon forecast horzin (e.g. 1 day ahead forecast)

true_value true observed values

sample number that identifies the predictive sample generated by a specific model for a specific observed value

prediction predictive sample for the corresponding true value

correlation_plot *Plot Correlation Between Metrics*

Description

Plots a coloured table of summarised scores obtained using `eval_forecasts`

Usage

```
correlation_plot(scores, select_metrics = NULL)
```

Arguments

`scores`	A data.frame of scores as produced by `eval_forecasts`
`select_metrics`	A character vector with the metrics to show. If set to `NULL` (default), all metrics present in `summarised_scores` will be shown

Value

A ggplot2 object showing a coloured matrix of correlations between metrics

Examples

```
scores <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_wide,
                                       by = c("model", "id", "horizon"),
                                       summarise_by = c("model", "id"))
scoringutils::correlation_plot(scores)
```

crps *Ranked Probability Score*

Description

Wrapper around the `crps_sample` function from the `scoringRules` package. Can be used for continuous as well as integer valued forecasts

Usage

```
crps(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples

Value

vector with the scoring values

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts withscoringRules, https://arxiv.org/pdf/1709.04743.pdf

Examples

```
true_values <- rpois(30, lambda = 1:30)
predictions <- replicate(200, rpois(n = 30, lambda = 1:30))
crps(true_values, predictions)
```

---

dss *Dawid-Sebastiani Score*

Description

Wrapper around the `dss_sample` function from the `scoringRules` package.

Usage

```
dss(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples

Value

vector with scoring values

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts withscoringRules, https://arxiv.org/pdf/1709.04743.pdf

Examples

```
true_values <- rpois(30, lambda = 1:30)
predictions <- replicate(200, rpois(n = 30, lambda = 1:30))
dss(true_values, predictions)
```

`eval_forecasts` *Evaluate forecasts*

Description

The function `eval_forecasts` is an easy to use wrapper of the lower level functions in the `scoringutils` package. It can be used to score probabilistic or quantile forecasts of continuous, integer-valued or binary variables.

Usage

```
eval_forecasts(
  data,
  by = NULL,
  summarise_by = by,
  quantiles = c(),
  sd = FALSE,
  pit_plots = FALSE,
  pit_arguments = list(plot = FALSE),
  interval_score_arguments = list(weigh = TRUE),
  summarised = TRUE,
  verbose = TRUE
)
```

Arguments

`data` A data.frame or data.table with the predictions and observations. Note: it is easiest to have a look at the example files provided in the package and in the examples below. The following columns need to be present:

- `true_value` - the true observed values
- `prediction` - predictions or predictive samples for one true value. (You only don't need to provide a prediction column if you want to score quantile forecasts in a wide range format.)

For integer and continuous forecasts a `sample` column is needed:

- `sample` - an index to identify the predictive samples in the prediction column generated by one model for one true value. Only necessary for continuous and integer forecasts, not for binary predictions.

For quantile forecasts the data can be provided in variety of formats. You can either use a range-based format or a quantile-based format. (You can convert between formats using `quantile_to_range`, `range_to_quantile`, `sample_to_range`, `sample_to_quantile`) For a quantile-format forecast you should provide:

- prediction - prediction to the corresponding quantile
- qunaitle - quantile to which the prediction corresponds

For a range format (long) forecast you need

- `prediction` the quantile forecasts

- boundary values should be either "lower" or "upper", depending on whether the prediction is for the lower or upper bound of a given range
- range the range for which a forecast was made. For a 50% interval the range should be 50. The forecast for the 25% quantile should have the value in the prediction column, the value of range should be 50 and the value of boundary should be "lower". If you want to score the median (i.e. range = 0), you still need to include a lower and an upper estimate, so the median has to appear twice.

Alternatively you can also provide the format in a wide range format. This format needs

- pairs of columns called something like 'upper_90' and 'lower_90', or 'upper_50' and 'lower_50', where the number denotes the interval range. For the median, you need to proivde columns called 'upper_0' and 'lower_0'

by — character vector of columns to group scoring by. This should be the lowest level of grouping possible, i.e. the unit of the individual observation. This is important as many functions work on individual observations. If you want a different level of aggregation, you should use summarise_by to aggregate the individual scores. Also not that the pit will be computed using summarise_by instead of by

summarise_by — character vector of columns to group the summary by. By default, this is equal to 'by' and no summary takes place. But sometimes you may want to to summarise over categories different from the scoring. summarise_by is also the grouping level used to compute (and possibly plot) the probability integral transform(pit).

quantiles — numeric vector of quantiles to be returned when summarising. Instead of just returning a mean, quantiles will be returned for the groups specified through 'summarise_by'. By default, no quantiles are returned.

sd — if TRUE (the default is FALSE) the standard deviation of all metrics will be returned when summarising.

pit_plots — if TRUE (not the default), pit plots will be returned. For details see pit.

pit_arguments — pass down additional arguments to the pit function.

interval_score_arguments — pass down additional arguments to the interval_score function, e.g. weigh = FAlSE.

summarised — Summarise arguments (i.e. take the mean per group specified in group_by. Default is TRUE.

verbose — print out additional helpful messages (default is TRUE)

Details

the following metrics are used where appropriate:

- Interval Score for quantile forecasts. Smaller is better. See interval_score for more information. By default, the weighted interval score is used.
- Brier Score for a probability forecast of a binary outcome. Smaller is better. See brier_score for more information.

- aem Absolute error of the median prediction
- Bias 0 is good, 1 and -1 are bad. See bias for more information.
- Sharpness Smaller is better. See sharpness for more information.
- Calibration represented through the p-value of the Anderson-Darling test for the uniformity of the Probability Integral Transformation (PIT). For integer valued forecasts, this p-value also has a standard deviation. Larger is better. See pit for more information.
- DSS Dawid-Sebastiani-Score. Smaller is better. See dss for more information.
- CRPS Continuous Ranked Probability Score. Smaller is better. See crps for more information.
- Log Score Smaller is better. Only for continuous forecasts. See logs for more information.

Value

A data.table with appropriate scores. For binary predictions, the Brier Score will be returned, for quantile predictions the interval score, as well as adapted metrics for calibration, sharpness and bias. For integer forecasts, Sharpness, Bias, DSS, CRPS, LogS, and pit_p_val (as an indicator of calibration) are returned. For integer forecasts, pit_sd is returned (to account for the randomised PIT), but no Log Score is returned (the internal estimation relies on a kernel density estimate which is difficult for integer-valued forecasts). If summarise_by is specified differently from by, the average score per summary unit is returned. If specified, quantiles and standard deviation of scores can also be returned when summarising.

Author(s)

Nikos Bosse <nikosbosse@gmail.com>

References

Funk S, Camacho A, Kucharski AJ, Lowe R, Eggo RM, Edmunds WJ (2019) Assessing the performance of real-time epidemic forecasts: A case study of Ebola in the Western Area region of Sierra Leone, 2014-15. PLoS Comput Biol 15(2): e1006785. https://doi.org/10.1371/journal.pcbi.1006785

Examples

```
## Probability Forecast for Binary Target
binary_example <- data.table::setDT(scoringutils::binary_example_data)
eval <- scoringutils::eval_forecasts(binary_example,
                                     by = c("id", "model", "horizon"),
                                     summarise_by = c("model"),
                                     quantiles = c(0.5), sd = TRUE)
eval <- scoringutils::eval_forecasts(binary_example,
                                     by = c("id", "model", "horizon"))

## Quantile Forecasts
# wide format
quantile_example <- data.table::setDT(scoringutils::quantile_example_data_wide)
eval <- scoringutils::eval_forecasts(quantile_example,
                                     by = c("model", "horizon", "id"),
```

```
                                    summarise_by = "model",
                                    quantiles = c(0.05, 0.95),
                                    sd = TRUE)
eval <- scoringutils::eval_forecasts(quantile_example,
                                    by = c("model", "horizon", "id"))

#long format

eval <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_long,
                                    by = c("model", "horizon", "id"),
                                    summarise_by = c("model", "range"))

## Integer Forecasts
integer_example <- data.table::setDT(scoringutils::integer_example_data)
eval <- scoringutils::eval_forecasts(integer_example,
                                    by = c("model", "id", "horizon"),
                                    summarise_by = c("model"),
                                    quantiles = c(0.1, 0.9),
                                    sd = TRUE,
                                    pit_plots = TRUE,
                                    pit_arguments = list(n_replicates = 30,
                                                         plot = TRUE))
eval <- scoringutils::eval_forecasts(integer_example)

## Continuous Forecasts
continuous_example <- data.table::setDT(scoringutils::continuous_example_data)
eval <- scoringutils::eval_forecasts(continuous_example,
                                    by = c("model", "id", "horizon"))
eval <- scoringutils::eval_forecasts(continuous_example,
                                    quantiles = c(0.5, 0.9),
                                    sd = TRUE,
                                    summarise_by = c("model"))
```

`hist_PIT` *PIT Histogram*

Description

Make a simple histogram of the probability integral transformed values to visually check whether a uniform distribution seems likely.

Usage

```
hist_PIT(PIT_samples, num_bins = NULL, caption = NULL)
```

Arguments

`PIT_samples`	A vector with the PIT values of size n
`num_bins`	the number of bins in the PIT histogram.

caption	provide a caption that gets passed to the plot If not given, the square root of n will be used

Value

vector with the scoring values

`hist_PIT_quantile` *PIT Histogram Quantile*

Description

Make a simple histogram of the probability integral transformed values to visually check whether a uniform distribution seems likely.

Usage

```
hist_PIT_quantile(PIT_samples, num_bins = NULL, caption = NULL)
```

Arguments

PIT_samples	A vector with the PIT values of size n
num_bins	the number of bins in the PIT histogram.
caption	provide a caption that gets passed to the plot If not given, the square root of n will be used

Value

vector with the scoring values

`integer_example_data` *Integer Example Data*

Description

A toy dataset for a probabilistic forecast of an integer outcome variable

Usage

```
integer_example_data
```

Format

A data frame with 6000 rows and 5 variables:

id unique identifier for true observed values

model name of the model that generated the forecasts

horizon forecast horzin (e.g. 1 day ahead forecast)

true_value true observed values

sample number that identifies the predictive sample generated by a specific model for a specific observed value

prediction predictive sample for the corresponding true value

---

`interval_coverage` *Plot Interval Coverage*

Description

Plot interval coverage

Usage

```
interval_coverage(
  summarised_scores,
  colour = "model",
  facet_formula = NULL,
  facet_wrap_or_grid = "facet_wrap",
  scales = "free_y"
)
```

Arguments

`summarised_scores`	Summarised scores as produced by `eval_forecasts`. Make sure that "range" is included in `summarise_by` when producing the summarised scores
`colour`	According to which variable shall the graphs be coloured? Default is "model".
`facet_formula`	formula for facetting in ggplot. If this is `NULL` (the default), no facetting will take place
`facet_wrap_or_grid`	Use ggplot2's `facet_wrap` or `facet_grid`? Anything other than "facet_wrap" will be interpreted as `facet_grid`. This only takes effect if `facet_formula` is not `NULL`
`scales`	scales argument that gets passed down to ggplot. Only necessary if you make use of facetting. Default is "free_y"

Value

ggplot object with a plot of interval coverage

Examples

```
example1 <- scoringutils::quantile_example_data_long
scores <- scoringutils::eval_forecasts(example1,
                                       summarise_by = c("model", "range"))
interval_coverage(scores)
```

`interval_score` *Interval Score*

Description

Proper Scoring Rule to score quantile predictions, following Gneiting and Raftery (2007). Smaller values are better.

The score is computed as

$$score = (upper - lower) + 2/alpha * (lower - true_value) * 1(true_values < lower) + 2/alpha * (true_value - upper) * 1(tru$$

where $1()$ is the indicator function and alpha is the decimal value that indicates how much is outside the prediction interval. To improve usability, the user is asked to provide an interval range in percentage terms, i.e. interval_range = 90 (percent) for a 90 percent prediction interval. Correspondingly, the user would have to provide the 5% and 95% quantiles (the corresponding alpha would then be 0.1). No specific distribution is assumed, but the range has to be symmetric (i.e you can't use the 0.1 quantile as the lower bound and the 0.7 quantile as the upper).

The interval score is a proper scoring rule that scores a quantile forecast

Usage

```
interval_score(
  true_values,
  lower,
  upper,
  interval_range = NULL,
  weigh = TRUE,
  separate_results = FALSE
)
```

Arguments

`true_values`	A vector with the true observed values of size n
`lower`	vector of size n with the lower quantile of the given range
`upper`	vector of size n with the upper quantile of the given range

`interval_range` the range of the prediction intervals. i.e. if you're forecasting the 0.05 and 0.95 quantile, the interval_range would be 90. Can be either a single number or a vector of size n, if the range changes for different forecasts to be scored. This corresponds to (100-alpha)/100 in Gneiting and Raftery (2007). Internally, the range will be transformed to alpha.

`weigh` if TRUE, weigh the score by alpha / 4, so it can be averaged into an interval score that, in the limit, corresponds to CRPS. Default: FALSE.

`separate_results` if TRUE (default is FALSE), then the separate parts of the interval score (sharpness, penalties for over- and under-prediction get returned as separate elements of a list)

Value

vector with the scoring values, or a list with separate entries if `separate_results` is TRUE.

References

Strictly Proper Scoring Rules, Prediction,and Estimation, Tilmann Gneiting and Adrian E. Raftery, 2007, Journal of the American Statistical Association, Volume 102, 2007 - Issue 477

Evaluating epidemic forecasts in an interval format, Johannes Bracher, Evan L. Ray, Tilmann Gneiting and Nicholas G. Reich, <arXiv:2005.12881v1>

Bracher J, Ray E, Gneiting T, Reich, N (2020) Evaluating epidemic forecasts in an interval format. https://arxiv.org/abs/2005.12881

Examples

```
true_values <- rnorm(30, mean = 1:30)
interval_range = 90
alpha = (100 - interval_range) / 100
lower = qnorm(alpha/2, rnorm(30, mean = 1:30))
upper = qnorm((1- alpha/2), rnorm(30, mean = 1:30))

interval_score(true_values = true_values,
               lower = lower,
               upper = upper,
               interval_range = interval_range)
```

---

logs *LogS*

Description

Wrapper around the `logs_sample` function from the `scoringRules` package. Used to score continuous predictions. While the Log Score is in theory also applicable to integer forecasts, the problem lies in the implementation: The Log Score needs a kernel density estimation, which is not well defined with integer-valued Monte Carlo Samples. The Log Score can be used for specific integer valued probabiliy distributions. See the scoringRules package for more details.

Usage

```
logs(true_values, predictions)
```

Arguments

true_values	A vector with the true observed values of size n
predictions	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples

Value

vector with the scoring values

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts withscoringRules, https://arxiv.org/pdf/1709.04743.pdf

Examples

```
true_values <- rpois(30, lambda = 1:30)
predictions <- replicate(200, rpois(n = 30, lambda = 1:30))
logs(true_values, predictions)
```

---

mse *Mean Squared Error*

Description

Mean Squared Error MSE of point forecasts. Calculuated as

$$mean((true_values - predicted_values)^2)$$

Usage

```
mse(true_values, predictions)
```

Arguments

true_values	A vector with the true observed values of size n
predictions	A vector with predicted values of size n

Value

vector with the scoring values

Examples

```
true_values <- rnorm(30, mean = 1:30)
predicted_values <- rnorm(30, mean = 1:30)
mse(true_values, predicted_values)
```

---

pit *Probability Integral Transformation*

Description

Uses a Probability Integral Transformation (PIT) (or a randomised PIT for integer forecasts) to assess the calibration of predictive Monte Carlo samples. Returns a p-values resulting from an Anderson-Darling test for uniformity of the (randomised) PIT as well as a PIT histogram if specified.

Usage

```
pit(
  true_values,
  predictions,
  plot = TRUE,
  full_output = FALSE,
  n_replicates = 20,
  num_bins = NULL,
  verbose = FALSE
)
```

Arguments

true_values	A vector with the true observed values of size n
predictions	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples
plot	logical. If TRUE, a histogram of the PIT values will be returned as well
full_output	return all individual p_values and computed u_t values for the randomised PIT. Usually not needed.
n_replicates	the number of tests to perform, each time re-randomising the PIT
num_bins	the number of bins in the PIT histogram (if plot == TRUE) If not given, the square root of n will be used
verbose	if TRUE (default is FALSE) more error messages are printed. Usually, this should not be needed, but may help with debugging.

Details

Calibration or reliability of forecasts is the ability of a model to correctly identify its own uncertainty in making predictions. In a model with perfect calibration, the observed data at each time point look as if they came from the predictive probability distribution at that time.

Equivalently, one can inspect the probability integral transform of the predictive distribution at time t,

$$u_t = F_t(x_t)$$

where x_t is the observed data point at time $tint_1, \ldots, t_n$, n being the number of forecasts, and F_t is the (continuous) predictive cumulative probability distribution at time t. If the true probability distribution of outcomes at time t is G_t then the forecasts eqnF_t are said to be ideal if eqnF_t = G_t at all times t. In that case, the probabilities ut are distributed uniformly.

In the case of discrete outcomes such as incidence counts, the PIT is no longer uniform even when forecasts are ideal. In that case a randomised PIT can be used instead:

$$u_t = P_t(k_t) + v * (P_t(k_t) - P_t(k_t - 1))$$

where k_t is the observed count, $P_t(x)$ is the predictive cumulative probability of observing incidence k at time t, eqnP_t (-1) = 0 by definition and v is standard uniform and independent of k. If P_t is the true cumulative probability distribution, then u_t is standard uniform.

The function checks whether integer or continuous forecasts were provided. It then applies the (randomised) probability integral and tests the values u_t for uniformity using the Anderson-Darling test.

As a rule of thumb, there is no evidence to suggest a forecasting model is miscalibrated if the p-value found was greater than a threshold of p >= 0.1, some evidence that it was miscalibrated if 0.01 < p < 0.1, and good evidence that it was miscalibrated if p <= 0.01. In this context it should be noted, though, that uniformity of the PIT is a necessary but not sufficient condition of calibration.

Value

a list with the following components:

- `p_value`: p-value of the Anderson-Darling test on the PIT values. In case of integer forecasts, this will be the mean p_value from the 'n_replicates' replicates
- `sd`: standard deviation of the p_value returned. In case of continuous forecasts, this will be NA as there is only one p_value returned.
- `hist_PIT` a plot object with the PIT histogram. Only returned if `plot == TRUE`. Call `plot(PIT(...)$hist_PIT)` to display the histogram.
- `p_values`: all p_values generated from the Anderson-Darling tests on the randomised PIT. Only returned for integer forecasts and if `full_output = TRUE`
- `u`: the u_t values internally computed. Only returned if `full_output = TRUE`

References

Sebastian Funk, Anton Camacho, Adam J. Kucharski, Rachel Lowe, Rosalind M. Eggo, W. John Edmunds (2019) Assessing the performance of real-time epidemic forecasts: A case study of Ebola in the Western Area region of Sierra Leone, 2014-15, <doi:10.1371/journal.pcbi.1006785>

Examples

```
## continuous predictions
true_values <- rnorm(30, mean = 1:30)
predictions <- replicate(200, rnorm(n = 30, mean = 1:30))
pit(true_values, predictions)

## integer predictions
true_values <- rpois(100, lambda = 1:100)
predictions <- replicate(5000, rpois(n = 100, lambda = 1:100))
pit(true_values, predictions,  n_replicates = 5)
```

---

plot_predictions *Plot Predictions vs True Values*

Description

Make a plot of observed and predicted values

Usage

```
plot_predictions(
  data,
  x = "date",
  add_truth_data = NULL,
  range = c(0, 50, 90),
  facet_formula = NULL,
  facet_wrap_or_grid = "facet_wrap",
  scales = "free_y",
  xlab = x,
  ylab = "True and predicted values"
)
```

Arguments

data	a data.frame that follows the same specifications outlined in eval_forecasts. The data.frame needs to have columns called "true_value", "prediction" and then either a column called sample, or one called "quantile" or two columns called "range" and "boundary".
x	character vector of length one that denotes the name of the variable on the x-axis. Usually, this will be "date", but it can be anything else.
add_truth_data	additional truth data, e.g. past values for which no predictions are available. This should be a data.frame with a column called "true_value" and another column corresponding to the x variable selected for the plot
range	numeric vector indicating the interval ranges to plot. If 0 is included in range, the median prediction will be shown.

facet_formula	formula for facetting in ggplot. If this is NULL (the default), no facetting will take place
facet_wrap_or_grid	Use ggplot2's facet_wrap or facet_grid? Anything other than "facet_wrap" will be interpreted as facet_grid. This only takes effect if facet_formula is not NULL
scales	scales argument that gets passed down to ggplot. Only necessary if you make use of facetting. Default is "free_y"
xlab	Label for the x-axis. Default is the variable name on the x-axis
ylab	Label for the y-axis. Default is "True and predicted values"

Value

ggplot object with a plot of true vs predicted values

Examples

```
example1 <- scoringutils::continuous_example_data
example2 <- scoringutils::quantile_example_data_long

scoringutils::plot_predictions(example1, x = "id",
                               facet_formula = ~ horizon)
scoringutils::plot_predictions(example2, x = "id",
                               facet_formula = ~ horizon)
```

---

quantile_bias *Determines Bias of Quantile Forecasts*

Description

Determines bias from quantile forecasts. For an increasing number of quantiles this measure converges against the sample based bias version for integer and continuous forecasts.

Usage

```
quantile_bias(range, lower, upper, true_value)
```

Arguments

range	vector of corresponding size with information about the width of the central prediction interval
lower	vector of length corresponding to the number of central prediction intervals that holds predictions for the lower bounds of a prediction interval
upper	vector of length corresponding to the number of central prediction intervals that holds predictions for the upper bounds of a prediction interval
true_value	a single true value

Details

For quantile forecasts, bias is measured as

$$B_t = (1 - 2 \cdot \max\{i | q_{t,i} \in Q_t \wedge q_{t,i} \leq x_t\}) 1(x_t \leq q_{t,0.5}) + (1 - 2 \cdot \min\{i | q_{t,i} \in Q_t \wedge q_{t,i} \geq x_t\}) 1(x_t \geq q_{t,0.5}),$$

where Q_t is the set of quantiles that form the predictive distribution at time t. They represent our belief about what the true value x_t will be. For consistency, we define Q_t such that it always includes the element $q_{t,0} = -\infty$ and $q_{t,1} = \infty$. $1()$ is the indicator function that is 1 if the condition is satisfied and 0 otherwise. In clearer terms, B_t is defined as the maximum percentile rank for which the corresponding quantile is still below the true value, if the true value is smaller than the median of the predictive distribution. If the true value is above the median of the predictive distribution, then B_t is the minimum percentile rank for which the corresponding quantile is still larger than the true value. If the true value is exactly the median, both terms cancel out and B_t is zero. For a large enough number of quantiles, the percentile rank will equal the proportion of predictive samples below the observed true value, and this metric coincides with the one for continuous forecasts.

Bias can assume values between -1 and 1 and is 0 ideally.

Value

scalar with the quantile bias for a single quantile prediction

Author(s)

Nikos Bosse <nikosbosse@gmail.com>

Examples

```
lower <- c(6341.000, 6329.500, 6087.014, 5703.500,
           5451.000, 5340.500, 4821.996, 4709.000,
           4341.500, 4006.250, 1127.000, 705.500)

upper <- c(6341.000, 6352.500, 6594.986, 6978.500,
           7231.000, 7341.500, 7860.004, 7973.000,
           8340.500, 8675.750, 11555.000, 11976.500)

range <- c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95, 98)

true_value <- 8062

quantile_bias(lower = lower, upper = upper,
              range = range, true_value = true_value)
```

quantile_coverage *Plot Quantile Coverage*

Description

Plot quantile coverage

Usage

```
quantile_coverage(
  summarised_scores,
  colour = "model",
  facet_formula = NULL,
  facet_wrap_or_grid = "facet_wrap",
  scales = "free_y"
)
```

Arguments

summarised_scores	Summarised scores as produced by eval_forecasts. Make sure that "quantile" is included in summarise_by when producing the summarised scores
colour	According to which variable shall the graphs be coloured? Default is "model".
facet_formula	formula for facetting in ggplot. If this is NULL (the default), no facetting will take place
facet_wrap_or_grid	Use ggplot2's facet_wrap or facet_grid? Anything other than "facet_wrap" will be interpreted as facet_grid. This only takes effect if facet_formula is not NULL
scales	scales argument that gets passed down to ggplot. Only necessary if you make use of facetting. Default is "free_y"

Value

ggplot object with a plot of interval coverage

Examples

```
example1 <- scoringutils::quantile_example_data_long
scores <- scoringutils::eval_forecasts(example1,
                                       summarise_by = c("model", "quantile"))
quantile_coverage(scores)
```

quantile_example_data_long

Quantile Example Data - Long Format

Description

A toy dataset for quantile forecasts of an outcome variable

Usage

```
quantile_example_data_long
```

Format

A data frame with 720 rows and 7 variables:

true_value true observed values

id unique identifier for true observed values

horizon forecast horizon (e.g. 1 day ahead forecast)

model name of the model that generated the forecasts

prediction quantile predictions

boundary lower or upper bound of an interval range

range interval range for which the quantile forecast was made

quantile_example_data_plain

Quantile Example Data - Plain Quantile Format

Description

A toy dataset for quantile forecasts of an outcome variable in a format that uses plain quantiles instead of interval ranges

Usage

```
quantile_example_data_plain
```

Format

A data frame with 600 rows and 6 variables:

true_value true observed values

id unique identifier for true observed values

horizon forecast horzin (e.g. 1 day ahead forecast)

model name of the model that generated the forecasts

prediction quantile predictions

quantile quantile of the corresponding prediction

`quantile_example_data_wide`

Quantile Example Data - Wide Format

Description

A toy dataset for quantile forecasts of an outcome variable

Usage

```
quantile_example_data_wide
```

Format

A data frame with 120 rows and 10 variables:

true_value true observed values

id unique identifier for true observed values

model name of the model that generated the forecasts

horizon forecast horzin (e.g. 1 day ahead forecast)

lower_90 prediction for the lower value of the 90% interval range (corresponding to the 5% quantile)

lower_50 prediction for the lower value of the 50% interval range (corresponding to the 25% quantile)

lower_0 prediction for the lower value of the 0% interval range (corresponding to the 50% quantile, i.e. the median. For computational reasons there need be a column with `lower_0` and `upper_0`)

upper_0 prediction for the upper value of the 0 (corresponding to the 50% quantile, i.e. the median)

upper_50 prediction for the upper value of the 50% interval range (corresponding to the 75% quantile)

upper_90 prediction for the lower value of the 90% interval range (corresponding to the 95% quantile)

quantile_to_long *Pivot Quantile Forecasts From Wide to Long Format*

Description

Given a data.frame that follows the structure shown in quantile_example_data_wide, the function outputs the same data in a long format as (as shown in quantile_example_data_long). This can be useful e.g. for plotting.

Usage

```
quantile_to_long(data)
```

Arguments

data	a data.frame following the specifications from eval_forecasts) for quantile forecasts. For an example, see quantile_example_data_wide)

Value

a data.frame in long format

Examples

```
wide <- scoringutils::quantile_example_data_wide
long <- scoringutils::quantile_to_long(wide)
```

quantile_to_range *Change Data from a Plain Quantile Format to a Range Format*

Description

Transform data from a format that uses quantiles only to one that uses interval ranges to denote quantiles.

Given a data.frame that follows the structure shown in quantile_example_data_plain, the function outputs the same data in a long format as (as shown in quantile_example_data_long).

Usage

```
quantile_to_range(data, keep_quantile_col = TRUE)
```

Arguments

data	a data.frame following the specifications shown in the example quantile_example_data_long)
keep_quantile_col	keep the quantile column in the final output after transformation (default is FALSE)

Value

a data.frame in a long interval range format

Examples

```
quantile_plain <- scoringutils::quantile_example_data_plain

long <- scoringutils::quantile_to_range(quantile_plain)
```

---

quantile_to_wide *Pivot Quantile Forecasts From Long to Wide Format*

Description

Given a data.frame that follows the structure shown in quantile_example_data_long, the function outputs the same data in a long format as (as shown in quantile_example_data_wide). This can be useful e.g. for plotting.

Usage

```
quantile_to_wide(data)
```

Arguments

data	a data.frame following the specifications from eval_forecasts) for quantile forecasts. For an example, see quantile_example_data_long)

Value

a data.frame in wide format

Examples

```
long <- scoringutils::quantile_example_data_long
wide <- scoringutils::quantile_to_wide(long)
```

range_plot *Plot Metrics by Range of the Prediction Interval*

Description

Visualise the metrics by range, e.g. if you are interested how different interval ranges contribute to the overal interval score, or how sharpness changes by range.

Usage

```
range_plot(
  scores,
  y = "interval_score",
  x = "model",
  colour = "range",
  xlab = x,
  ylab = y
)
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by eval_forecasts. Note that "range" must be included in the summarise_by argument when running eval_forecasts
y	The variable from the scores you want to show on the y-Axis. This could be something like "interval_score" (the default) or "sharpness"
x	The variable from the scores you want to show on the x-Axis. Usually this will be "model"
colour	Charachter vector of length one used to determine a variable for colouring dots. The Default is "range".
xlab	Label for the x-axis. Default is the variable name on the x-axis
ylab	Label for the y-axis. Default is "WIS contributions"

Value

A ggplot2 object showing a contributions from the three components of the weighted interval score

Examples

```
scores <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_long,
                                       by = c("model", "id", "horizon"),
                                       summarise_by = c("model", "horizon", "range"))
scoringutils::range_plot(scores, x = "model")

scoringutils::range_plot(scores, y = "sharpness", x = "model")
```

range_to_quantile *Change Data from a Range Format to a Plain Quantile Format*

Description

Transform data from a format that uses interval ranges to denote quantiles to a format that uses quantiles only.

Given a data.frame that follows the structure shown in quantile_example_data_long, the function outputs the same data in a long format as (as shown in quantile_example_data_long). This can be useful e.g. for plotting. If you're data.frame is in a different format, consider running quantile_to_wide first.

Usage

```
range_to_quantile(data, keep_range_col = FALSE)
```

Arguments

data	a data.frame following the specifications from eval_forecasts) for quantile forecasts. For an example, see quantile_example_data_long)
keep_range_col	keep the range and boundary columns after transformation (default is FALSE)

Value

a data.frame in a plain quantile format

Examples

```
wide <- scoringutils::quantile_example_data_wide
long <- scoringutils::quantile_to_long(wide)

plain_quantile <- range_to_quantile(long)
```

sample_to_quantile *Change Data from a Sample Based Format to a Quantile Format*

Description

Transform data from a format that is based on predictive samples to a format based on plain quantiles.

Usage

```
sample_to_quantile(data, quantiles = c(0.05, 0.25, 0.5, 0.75, 0.95), type = 7)
```

Arguments

data	a data.frame with samples
quantiles	a numeric vector of quantiles to extract
type	type argument passed down to the quantile function. For more information, see quantile

Value

a data.frame in a long interval range format

Examples

```
example_data <- scoringutils::integer_example_data

quantile_data <- sample_to_quantile(example_data)
```

---

sample_to_range *Change Data from a Sample Based Format to a Interval Range Format*

Description

Transform data from a format that is based on predictive samples to a format based on interval ranges

Usage

```
sample_to_range(data, range = c(0, 50, 90), type = 7, keep_quantile_col = TRUE)
```

Arguments

data	a data.frame with samples
range	a numeric vector of interval ranges to extract (e.g. c(0,50,90))
type	type argument passed down to the quantile function. For more information, see quantile
keep_quantile_col	keep quantile column, default is TRUE

Value

a data.frame in a long interval range format

Examples

```
example_data <- scoringutils::integer_example_data

quantile_data <- sample_to_range(example_data)
```

score_heatmap *Create a Heatmap of a Scoring Metric*

Description

This function can be used to create a heatmap of one metric across different groups, e.g. the interval score obtained by several forecasting models in different locations.

Usage

```
score_heatmap(scores, y = "model", x, metric, ylab = y, xlab = x)
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by eval_forecasts.
y	The variable from the scores you want to show on the y-Axis. The default for this is "model"
x	The variable from the scores you want to show on the x-Axis. This could be something like "horizon", or "location"
metric	the metric that determines the value and colour shown in the tiles of the heatmap
ylab	Label for the y-axis. Default is the variable name on the y-axis
xlab	Label for the x-axis. Default is the variable name on the x-axis

Value

A ggplot2 object showing a heatmap of the desired metric

Examples

```
scores <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_long,
                                       by = c("model", "id", "horizon"),
                                       summarise_by = c("model", "horizon"))

scoringutils::score_heatmap(scores, x = "horizon", metric = "bias")
```

score_table *Plot Coloured Score Table*

Description

Plots a coloured table of summarised scores obtained using eval_forecasts

Usage

```
score_table(
  summarised_scores,
  y = NULL,
  select_metrics = NULL,
  facet_formula = NULL,
  ncol = NULL,
  facet_wrap_or_grid = "facet_wrap"
)
```

Arguments

summarised_scores	A data.frame of summarised scores as produced by eval_forecasts
y	the variable to be shown on the y-axis. If NULL (default), all columns that are not scoring metrics will be used. Alternatively, you can specify a vector with column names, e.g. y = c("model","location"). These column names will be concatenated to create a unique row identifier (e.g. "model1_location1")
select_metrics	A character vector with the metrics to show. If set to NULL (default), all metrics present in summarised_scores will be shown
facet_formula	formula for facetting in ggplot. If this is NULL (the default), no facetting will take place
ncol	Number of columns for facet wrap. Only relevant if facet_formula is given and facet_wrap_or_grid == "facet_wrap"
facet_wrap_or_grid	Use ggplot2's facet_wrap or facet_grid? Anything other than "facet_wrap" will be interpreted as facet_grid. This only takes effect if facet_formula is not NULL

Value

A ggplot2 object with a coloured table of summarised scores

Examples

```
scores <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_wide,
                                       by = c("model", "id", "horizon"),
                                       summarise_by = c("model", "horizon"))
```

```
scoringutils::score_table(scores, y = "model", facet_formula = ~ horizon,
                          ncol = 1)

scoringutils::score_table(scores, y = c("model", "horizon"))

# yields the same result in this case
scoringutils::score_table(scores)
```

---

scoringutils *scoringutils*

Description

This package is designed to help with assessing the quality of predictions. It provides a collection of proper scoring rules and metrics as well that can be accessed independently or collectively through a higher-level wrapper function.

Predicitions can be either probabilistic forecasts (generally predictive samples generated by Markov Chain Monte Carlo procedures), quantile forecasts or point forecasts. The true values can be either continuous, integer, or binary.

A collection of different metrics and scoring rules can be accessed through the function `eval_forecasts`. Given a data.frame of the correct form the function will automatically figure out the type of prediction and true values and return appropriate scoring metrics.

The package also has a lot of default visualisation based on the output created by `eval_forecasts`.

- `score_table`
- `correlation_plot`
- `wis_components`
- `range_plot`
- `score_heatmap`
- `plot_predictions`
- `interval_coverage`
- `quantile_coverage`

Alternatively, the following functions can be accessed directly:

- `brier_score`
- `pit`
- `bias`
- `quantile_bias`
- `sharpness`
- `crps`
- `logs`
- `dss`

- ae_median

Predictions can be evaluated in a lot of differnent formats. If you want to convert from one format to the other, the following helper functions can do that for you:

- sample_to_range
- sample_to_quantile
- quantile_to_range
- range_to_quantile

---

sharpness *Determines sharpness of a probabilistic forecast*

Description

Determines sharpness of a probabilistic forecast

Usage

```
sharpness(predictions)
```

Arguments

predictions	nxN matrix of predictive samples, n (number of rows) being the number of data points and N (number of columns) the number of Monte Carlo samples

Details

Sharpness is the ability of the model to generate predictions within a narrow range. It is a data-independent measure, and is purely a feature of the forecasts themselves.

Shaprness of predictive samples corresponding to one single true value is measured as the normalised median of the absolute deviation from the median of the predictive samples. For details, see mad

Value

vector with sharpness values

References

Funk S, Camacho A, Kucharski AJ, Lowe R, Eggo RM, Edmunds WJ (2019) Assessing the performance of real-time epidemic forecasts: A case study of Ebola in the Western Area region of Sierra Leone, 2014-15. PLoS Comput Biol 15(2): e1006785. https://doi.org/10.1371/journal.pcbi.1006785

Examples

```
predictions <- replicate(200, rpois(n = 30, lambda = 1:30))
sharpness(predictions)
```

wis_components *Plot Contributions to the Weighted Interval Score*

Description

Visualise the components of the weighted interval score: penalties for over-predicition, under-prediction and for a lack of sharpness

Usage

```
wis_components(
  scores,
  x = "model",
  group = NULL,
  relative_contributions = FALSE,
  facet_formula = NULL,
  scales = "free_y",
  nrow = NULL,
  xlab = x,
  ylab = "WIS contributions"
)
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by eval_forecasts
x	The variable from the scores you want to show on the x-Axis. Usually this will be "model"
group	Choose a grouping variable for the plot that gets directly passed down to ggplot. Default is NULL
relative_contributions	show relative contributions instead of absolute contributions. Default is FALSE and this functionality is not available yet.
facet_formula	facetting formula passed down to ggplot. Default is NULL
scales	scales argument that gets passed down to ggplot. Only necessary if you make use of facetting. Default is "free_y"
nrow	nrow argument that gets passed down to ggplot. Specifies the number of rows to use for facet_wrap in ggplot
xlab	Label for the x-axis. Default is the variable name on the x-axis
ylab	Label for the y-axis. Default is "WIS contributions"

Value

A ggplot2 object showing a contributions from the three components of the weighted interval score

References

Bracher J, Ray E, Gneiting T, Reich, N (2020) Evaluating epidemic forecasts in an interval format. https://arxiv.org/abs/2005.12881

Examples

```
scores <- scoringutils::eval_forecasts(scoringutils::quantile_example_data_wide,
                                       by = c("model", "id", "horizon"),
                                       summarise_by = c("model", "horizon"))
scoringutils::wis_components(scores, x = "model", facet_formula = ~ horizon)
```

Index