Package ‘scoringutils’

August 17, 2022

Title Utilities for Scoring and Assessing Predictions

Version 1.0.1

Language en-GB

Description Provides a collection of metrics and proper scoring rules
(Tilmann Gneiting & Adrian E Raftery (2007)
<doi:10.1198/016214506000001437>, Jordan, A., Krüger, F., & Lerch, S. (2019)
<doi:10.18637/jss.v090.i12>) within a consistent framework for
evaluation, comparison and visualisation of forecasts.
In addition to 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 (e.g. binary, discrete, continuous) which may
come in different formats (e.g. forecasts represented by predictive samples
or by quantiles of the predictive distribution) can be evaluated.
Scoring metrics can be used either through a convenient data.frame format,
or can be applied as individual functions in a vector / matrix format.
All functionality has been implemented with a focus on performance and is
robustly tested.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports data.table, ggdist (>= 3.1.0), ggplot2, methods, rlang,
scoringRules, stats

Suggests kableExtra, knitr, magrittr, rmarkdown, testthat, vdiffr

Config/Needs/website r-lib/pkgdown, amirmasoudabdol/preferably

Config/testthat/edition 3

RoxygenNote 7.2.0

URL `https://epiforecasts.io/scoringutils/`,
`https://github.com/epiforecasts/scoringutils`

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

VignetteBuilder knitr

Depends R (>= 3.5)

NeedsCompilation no

Author Nikos Bosse [aut, cre] (<https://orcid.org/0000-0002-7750-5280>),
Sam Abbott [aut] (<https://orcid.org/0000-0001-8057-8037>),
Hugo Gruson [aut] (<https://orcid.org/0000-0002-4094-1476>),
Johannes Bracher [ctb] (<https://orcid.org/0000-0002-3777-1410>),
Sebastian Funk [ctb]

Maintainer Nikos Bosse <nikosbosse@gmail.com>

Repository CRAN

Date/Publication 2022-08-16 22:00:02 UTC

R topics documented:

`abs_error` *Absolute Error*

Description

Calculate absolute error as

$$\text{abs}(\text{true\_value} - \text{median\_prediction})$$

Usage

```
abs_error(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	numeric vector with predictions, corresponding to the quantiles in a second vector, `quantiles`.

Value

vector with the absolute error

See Also

`ae_median_sample()`, `ae_median_quantile()`

Examples

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

---

add_coverage *Add coverage of central prediction intervals*

Description

Adds a column with the coverage of central prediction intervals to unsummarised scores as produced by `score()`

Usage

```
add_coverage(scores, by, ranges = c(50, 90))
```

Arguments

`scores`	A data.table of scores as produced by `score()`.
`by`	character vector with column names to add the coverage for.
`ranges`	numeric vector of the ranges of the central prediction intervals for which coverage values shall be added.

Details

The coverage values that are added are computed according to the values specified in by. If, for example, by = `"model"`, then there will be one coverage value for every model and `add_coverage()` will compute the coverage for every model across the values present in all other columns which define the unit of a single forecast.

Value

a data.table with unsummarised scores with columns added for the coverage of the central prediction intervals. While the overall data.table is still unsummarised, note that for the coverage columns some level of summary is present according to the value specified in by.

Examples

```
library(magrittr) # pipe operator
score(example_quantile) %>%
  add_coverage(by = c("model", "target_type")) %>%
  summarise_scores(by = c("model", "target_type")) %>%
  summarise_scores(fun = signif, digits = 2)
```

ae_median_quantile — *Absolute Error of the Median (Quantile-based Version)*

Description

Absolute error of the median calculated as

$$\text{abs}(\text{true\_value} - \text{prediction})$$

The function was created for internal use within `score()`, but can also used as a standalone function.

Usage

```
ae_median_quantile(true_values, predictions, quantiles = NULL)
```

Arguments

`true_values`	A vector with the true observed values of size n
`predictions`	numeric vector with predictions, corresponding to the quantiles in a second vector, `quantiles`.
`quantiles`	numeric vector that denotes the quantile for the values in `predictions`. Only those predictions where `quantiles == 0.5` will be kept. If `quantiles` is `NULL`, then all `predictions` and `true_values` will be used (this is then the same as `abs_error()`)

Value

vector with the scoring values

See Also

`ae_median_sample()`, `abs_error()`

Examples

```
true_values <- rnorm(30, mean = 1:30)
predicted_values <- rnorm(30, mean = 1:30)
ae_median_quantile(true_values, predicted_values, quantiles = 0.5)
```

ae_median_sample — *Absolute Error of the Median (Sample-based Version)*

Description

Absolute error of the median calculated as

$$\mathrm{abs}(\mathrm{true\_value} - \mathrm{median\_prediction})$$

Usage

```
ae_median_sample(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

See Also

`ae_median_quantile()`, `abs_error()`

Examples

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

available_metrics — *Available metrics in scoringutils*

Description

Available metrics in scoringutils

Usage

```
available_metrics()
```

Value

A vector with the name of all available metrics `avail_forecasts` *Display Number of Forecasts Available*

Description

Given a data set with forecasts, count the number of available forecasts for arbitrary grouping (e.g. the number of forecasts per model, or the number of forecasts per model and location). This is useful to determine whether there are any missing forecasts.

Usage

```
avail_forecasts(data, by = NULL, collapse = c("quantile", "sample"))
```

Arguments

`data`	data.frame with predictions in the same format required for `score()`.
`by`	character vector or `NULL` (the default) that denotes the categories over which the number of forecasts should be counted. By default (by = `NULL`) this will be the unit of a single forecast (i.e. all available columns (apart from a few "protected" columns such as 'prediction' and 'true value') plus "quantile" or "sample" where present).
`collapse`	character vector (default is `c("quantile", "sample")`) with names of categories for which the number of rows should be collapsed to one when counting. For example, a single forecast is usually represented by a set of several quantiles or samples and collapsing these to one makes sure that a single forecast only gets counted once.

Value

A data.table with columns as specified in by and an additional column with the number of forecasts.

Examples

```
avail_forecasts(example_quantile,
  collapse = c("quantile"),
  by = c("model", "target_type")
)
```

`bias_quantile` *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

```
bias_quantile(predictions, quantiles, true_value)
```

Arguments

`predictions`	vector of length corresponding to the number of quantiles that holds predictions
`quantiles`	vector of corresponding size with the quantiles for which predictions were made
`true_value`	a single true value

Details

For quantile forecasts, bias is measured as

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

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

```
predictions <- c(
  705.500, 1127.000, 4006.250, 4341.500, 4709.000, 4821.996,
  5340.500, 5451.000, 5703.500, 6087.014, 6329.500, 6341.000,
  6352.500, 6594.986, 6978.500, 7231.000, 7341.500, 7860.004,
  7973.000, 8340.500, 8675.750, 11555.000, 11976.500
)

quantiles <- c(0.01, 0.025, seq(0.05, 0.95, 0.05), 0.975, 0.99)

true_value <- 8062

bias_quantile(predictions, quantiles, true_value = true_value)
```

`bias_range` *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

```
bias_range(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

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

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 $q_{t,1} = \infty$. $\mathbf{1}()$ 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

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

---

bias_sample *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_sample(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.

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. doi: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_sample(true_values, predictions)

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

`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 with all values equal to either 0 or 1
`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.

$$\text{Brier\_Score} = \frac{1}{N} \sum_{t=1}^{n} (\text{prediction}_t - \text{outcome}_t)^2$$

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)
```

---

check_forecasts — *Check forecasts*

Description

Function to check the input data before running `score()`.

The data should come in one of three different formats:

- A format for binary predictions (see example_binary)
- A sample-based format for discrete or continuous predictions (see example_continuous and example_integer)
- A quantile-based format (see example_quantile)

Usage

```
check_forecasts(data)
```

Arguments

`data`	data.frame with predictions in the same format required for `score()`.

Value

A list with elements that give information about what `scoringutils` thinks you are trying to do and potential issues.

- `target_type` the type of the prediction target as inferred from the input: 'binary', if all values in `true_value` are either 0 or 1 and values in `prediction` are between 0 and 1, 'discrete' if all true values are integers. and 'continuous' if not.
- `prediction_type` inferred type of the prediction. 'quantile', if there is a column called 'quantile', else 'discrete' if all values in `prediction` are integer, else 'continuous.
- `forecast_unit` unit of a single forecast, i.e. the grouping that uniquely defines a single forecast. This is assumed to be all present columns apart from the following protected columns: `c("prediction", "true_value", "sample", "quantile","range", "boundary")`. It is important that you remove all unnecessary columns before scoring.
- `rows_per_forecast` a data.frame that shows how many rows (usually quantiles or samples there are available per forecast. If a forecast model has several entries, then there a forecasts with differing numbers of quantiles / samples.

- `unique_values` A data.frame that shows how many unique values there are present per model and column in the data. This doesn't directly show missing values, but rather the maximum number of unique values across the whole data.
- `warnings` A vector with warnings. These can be ignored if you know what you are doing.
- `errors` A vector with issues that will cause an error when running `score()`.
- `messages` A verbal explanation of the information provided above.

Author(s)

Nikos Bosse <nikosbosse@gmail.com>

See Also

Function to move from sample-based to quantile format: `sample_to_quantile()`

Examples

```
check <- check_forecasts(example_quantile)
print(check)
check_forecasts(example_binary)
```

---

correlation *Correlation Between Metrics*

Description

Calculate the correlation between different metrics for a data.frame of scores as produced by `score()`.

Usage

```
correlation(scores, metrics = NULL)
```

Arguments

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

Value

A data.table with correlations for the different metrics

Examples

```
scores <- score(example_quantile)
correlation(scores)
```

crps_sample *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_sample(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

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts with scoringRules, https://www.jstatsoft.org/article/view/v090i12

Examples

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

dss_sample *Dawid-Sebastiani Score*

Description

Wrapper around the `dss_sample()` function from the **scoringRules** package.

Usage

```
dss_sample(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 scoring values

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts with scoringRules, https://www.jstatsoft.org/article/view/v090i12

Examples

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

---

example_binary *Binary Forecast Example Data*

Description

A data set with binary predictions for COVID-19 cases and deaths constructed from data submitted to the European Forecast Hub.

Usage

```
example_binary
```

Format

A data frame with 346 rows and 10 columns:

location the country for which a prediction was made

location_name name of the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

true_value true observed values

forecast_date the date on which a prediction was made

model name of the model that generated the forecasts

horizon forecast horizon in weeks

prediction predicted value

Details

Predictions in the data set were constructed based on the continuous example data by looking at the number of samples below the mean prediction. The outcome was constructed as whether or not the actually observed value was below or above that mean prediction. This should not be understood as sound statistical practice, but rather as a practical way to create an example data set.

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

`example_continuous` *Continuous Forecast Example Data*

Description

A data set with continuous predictions for COVID-19 cases and deaths constructed from data submitted to the European Forecast Hub.

Usage

```
example_continuous
```

Format

A data frame with 13,429 rows and 10 columns:

location the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

true_value true observed values

location_name name of the country for which a prediction was made

forecast_date the date on which a prediction was made

model name of the model that generated the forecasts

horizon forecast horizon in weeks

prediction predicted value

sample id for the corresponding sample

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

`example_integer` *Integer Forecast Example Data*

Description

A data set with integer predictions for COVID-19 cases and deaths constructed from data submitted to the European Forecast Hub.

Usage

```
example_integer
```

Format

A data frame with 13,429 rows and 10 columns:

location the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

true_value true observed values

location_name name of the country for which a prediction was made

forecast_date the date on which a prediction was made

model name of the model that generated the forecasts

horizon forecast horizon in weeks

prediction predicted value

sample id for the corresponding sample

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

`example_point` *Point Forecast Example Data*

Description

A data set with predictions for COVID-19 cases and deaths submitted to the European Forecast Hub. This data set is like the quantile example data, only that the median has been replaced by a point forecast.

Usage

```
example_point
```

Format

A data frame with

location the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

true_value true observed values

location_name name of the country for which a prediction was made

forecast_date the date on which a prediction was made

quantile quantile of the corresponding prediction

prediction predicted value

model name of the model that generated the forecasts

horizon forecast horizon in weeks

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

`example_quantile` *Quantile Example Data*

Description

A data set with predictions for COVID-19 cases and deaths submitted to the European Forecast Hub.

Usage

```
example_quantile
```

Format

A data frame with

- **location** the country for which a prediction was made
- **target_end_date** the date for which a prediction was made
- **target_type** the target to be predicted (cases or deaths)
- **true_value** true observed values
- **location_name** name of the country for which a prediction was made
- **forecast_date** the date on which a prediction was made
- **quantile** quantile of the corresponding prediction
- **prediction** predicted value
- **model** name of the model that generated the forecasts
- **horizon** forecast horizon in weeks

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

`example_quantile_forecasts_only`

Quantile Example Data - Forecasts only

Description

A data set with quantile predictions for COVID-19 cases and deaths submitted to the European Forecast Hub.

Usage

```
example_quantile_forecasts_only
```

Format

A data frame with 7,581 rows and 9 columns:

location the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

forecast_date the date on which a prediction was made

quantile quantile of the corresponding prediction

prediction predicted value

model name of the model that generated the forecasts

horizon forecast horizon in weeks

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

example_truth_only *Truth data only*

Description

A data set with truth values for COVID-19 cases and deaths submitted to the European Forecast Hub.

Usage

```
example_truth_only
```

Format

A data frame with 140 rows and 5 columns:

location the country for which a prediction was made

target_end_date the date for which a prediction was made

target_type the target to be predicted (cases or deaths)

true_value true observed values

location_name name of the country for which a prediction was made

Details

The data was created using the script create-example-data.R in the inst/ folder (or the top level folder in a compiled package).

Source

https://github.com/covid19-forecast-hub-europe/covid19-forecast-hub-europe/commit/a42867b1ea152c57e25b04f9faa26cfd4bfd8fa6/

find_duplicates *Find duplicate forecasts*

Description

Helper function to identify duplicate forecasts, i.e. instances where there is more than one forecast for the same prediction target.

Usage

```
find_duplicates(data)
```

Arguments

`data`	A data.frame as used for `score()`

Value

A data.frame with all rows for which a duplicate forecast was found

Examples

```
example <- rbind(example_quantile, example_quantile[1000:1010])
find_duplicates(example)
```

`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

$$\text{score} = (\text{upper} - \text{lower}) + \frac{2}{\alpha}(\text{lower} - \text{true\_value}) * \mathbf{1}(\text{true\_value} < \text{lower}) + \frac{2}{\alpha}(\text{true\_value} - \text{upper}) * \mathbf{1}(\text{true\_value} > \text{upper})$$

where $\mathbf{1}()$ is the indicator function and indicates how much is outside the prediction interval. α 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). Non-symmetric quantiles can be scored using the function `quantile_score()`.

Usage

```
interval_score(
  true_values,
  lower,
  upper,
  interval_range,
  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 prediction for the lower quantile of the given range
- `upper` vector of size n with the prediction for 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 / 2, so it can be averaged into an interval score that, in the limit, corresponds to CRPS. Alpha is the decimal value that represents how much is outside a central prediction interval (e.g. for a 90 percent central prediction interval, alpha is 0.1) Default: `TRUE`.
- `separate_results` if `TRUE` (default is `FALSE`), then the separate parts of the interval score (dispersion penalty, penalties for over- and under-prediction get returned as separate elements of a list). If you want a `data.frame` instead, simply call `as.data.frame()` on the output.

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, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008618 # nolint

Examples

```
true_values <- rnorm(30, mean = 1:30)
interval_range <- rep(90, 30)
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
)

# example with missing values and separate results
interval_score(
```

```
  true_values = c(true_values, NA),
  lower = c(lower, NA),
  upper = c(NA, upper),
  separate_results = TRUE,
  interval_range = 90
)
```

`logs_binary` *Log Score for Binary outcomes*

Description

Computes the Log Score for probabilistic forecasts of binary outcomes.

Usage

```
logs_binary(true_values, predictions)
```

Arguments

`true_values`	A vector with the true observed values of size n with all values equal to either 0 or 1
`predictions`	A vector with a predicted probability that true_value = 1.

Details

The Log Score is a proper score rule suited to assessing 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 Log Score is then computed as the negative logarithm of the probability assigned to the true outcome. Reporting the negative logarithm means that smaller values are better.

Value

A numeric value with the Log 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)
logs_binary(true_values, predictions)
```

logs_sample *Logarithmic score*

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 probability distributions. See the scoringRules package for more details.

Usage

```
logs_sample(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

References

Alexander Jordan, Fabian Krüger, Sebastian Lerch, Evaluating Probabilistic Forecasts with scoringRules, https://www.jstatsoft.org/article/view/v090i12

Examples

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

`mad_sample` *Determine dispersion of a probabilistic forecast*

Description

Determine dispersion of a probabilistic forecast

Usage

```
mad_sample(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. Alternatively, predictions can just be a vector of size n.

Details

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

Dispersion 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() and the explanations given in Funk et al. (2019)

Value

vector with dispersion 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. doi:10.1371/journal.pcbi.1006785

Examples

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

make_NA *Make Rows NA in Data for Plotting*

Description

Filters the data and turns values into `NA` before the data gets passed to `plot_predictions()`. The reason to do this is to this is that it allows to 'filter' prediction and truth data separately. Any value that is NA will then be removed in the subsequent call to `plot_predictions()`.

Usage

```
make_NA(data = NULL, what = c("truth", "forecast", "both"), ...)

make_na(data = NULL, what = c("truth", "forecast", "both"), ...)
```

Arguments

`data`	A data.frame or data.table with the predictions and observations. For examples, look at the example_quantile, example_continuous, example_integer, and example_binary data sets. For scoring using `score()`, 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 scoring 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 scoring predictions in a quantile-format forecast you should provide a column called `quantile`: • `quantile`: quantile to which the prediction corresponds
`what`	character vector that determines which values should be turned into `NA`. If `what = "truth"`, values in the column 'true_value' will be turned into `NA`. If `what = "forecast"`, values in the column 'prediction' will be turned into `NA`. If `what = "both"`, values in both column will be turned into `NA`.
`...`	logical statements used to filter the data

Value

A data.table

Examples

```
make_NA (
    example_continuous,
    what = "truth",
    target_end_date >= "2021-07-22",
    target_end_date < "2021-05-01"
  )
```

---

merge_pred_and_obs *Merge Forecast Data And Observations*

Description

The function more or less provides a wrapper around `merge` that aims to handle the merging well if additional columns are present in one or both data sets. If in doubt, you should probably merge the data sets manually.

Usage

```
merge_pred_and_obs(
  forecasts,
  observations,
  join = c("left", "full", "right"),
  by = NULL
)
```

Arguments

`forecasts`	data.frame with the forecast data (as can be passed to `score()`).
`observations`	data.frame with the observations
`join`	character, one of `c("left", "full", "right")`. Determines the type of the join. Usually, a left join is appropriate, but sometimes you may want to do a full join to keep dates for which there is a forecast, but no ground truth data.
`by`	character vector that denotes the columns by which to merge. Any value that is not a column in observations will be removed.

Value

a data.frame with forecasts and observations

Examples

```
forecasts <- example_quantile_forecasts_only
observations <- example_truth_only
merge_pred_and_obs(forecasts, observations)
```

metrics *Summary information for selected metrics*

Description

A data set with summary information on selected metrics implemented in **scoringutils**

Usage

```
metrics
```

Format

An object of class `data.table` (inherits from `data.frame`) with 22 rows and 8 columns.

Details

The data was created using the script create-metric-tables.R in the inst/ folder (or the top level folder in a compiled package).

pairwise_comparison *Do Pairwise Comparisons of Scores*

Description

Make pairwise comparisons between models. The code for the pairwise comparisons is inspired by an implementation by Johannes Bracher.

The implementation of the permutation test follows the function `permutationTest` from the `surveillance` package by Michael Höhle, Andrea Riebler and Michaela Paul.

Usage

```
pairwise_comparison(
  scores,
  by = c("model"),
  metric = "auto",
  baseline = NULL,
  ...
)
```

Arguments

`scores`	A data.table of scores as produced by `score()`.
`by`	character vector with names of columns present in the input data.frame. `by` determines how pairwise comparisons will be computed. You will get a relative skill score for every grouping level determined in `by`. If, for example, `by = c("model", "location")`. Then you will get a separate relative skill score for every model in every location. Internally, the data.frame will be split according `by` (but removing "model" before splitting) and the pairwise comparisons will be computed separately for the split data.frames.
`metric`	A character vector of length one with the metric to do the comparison on. The default is "auto", meaning that either "interval_score", "crps", or "brier_score" will be selected where available. See `available_metrics()` for available metrics.
`baseline`	character vector of length one that denotes the baseline model against which to compare other models.
`...`	additional arguments for the comparison between two models. See `compare_two_models()` for more information.

Value

A ggplot2 object with a coloured table of summarised scores

Author(s)

Nikos Bosse <nikosbosse@gmail.com>

Johannes Bracher, <johannes.bracher@kit.edu>

Examples

```
df <- data.frame(
  model = rep(c("model1", "model2", "model3"), each = 10),
  date = as.Date("2020-01-01") + rep(1:5, each = 2),
  location = c(1, 2),
  interval_score = (abs(rnorm(30))),
  ae_median = (abs(rnorm(30)))
)

res <- pairwise_comparison(df,
  baseline = "model1"
)
plot_pairwise_comparison(res)

eval <- score(example_quantile)
pairwise_comparison(eval, by = c("model"))
```

pit *Probability Integral Transformation (data.frame Format)*

Description

Wrapper around `pit()` for use in data.frames

Usage

```
pit(data, by, n_replicates = 100)
```

Arguments

`data`	a data.frame with the following columns: `true_value`, `prediction`, `sample`.
`by`	Character vector with the columns according to which the PIT values shall be grouped. If you e.g. have the columns 'model' and 'location' in the data and want to have a PIT histogram for every model and location, specify `by = c("model", "location")`.
`n_replicates`	the number of draws for the randomised PIT for integer predictions.

Details

see `pit()`

Value

a data.table with PIT values according to the grouping specified in by

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

```
result <- pit(example_continuous, by = "model")
plot_pit(result)

# example with quantile data
result <- pit(example_quantile, by = "model")
plot_pit(result)
```

`pit_sample` *Probability Integral Transformation (sample-based version)*

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_sample(true_values, predictions, n_replicates = 100)
```

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.
`n_replicates`	the number of draws for the randomised PIT for integer predictions.

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 t in $t_1, \ldots, t_n$, 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 F_t are said to be ideal if $F_t = G_t$ at all times t. In that case, the probabilities u_t 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, $P_t(-1) = 0$ 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. However, the AD-p-values may be overly strict and there actual usefulness may be questionable. In this context it should be noted, though, that uniformity of the PIT is a necessary but not sufficient condition of calibration.

Value

A vector with PIT-values. For continuous forecasts, the vector will correspond to the length of `true_values`. For integer forecasts, a randomised PIT will be returned of length `length(true_values) * n_replicates`

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

See Also

`pit()`

Examples

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

## integer predictions
true_values <- rpois(100, lambda = 1:100)
predictions <- replicate(5000, rpois(n = 100, lambda = 1:100))
pit <- pit_sample(true_values, predictions, n_replicates = 50)
plot_pit(pit)
```

`plot_avail_forecasts` *Visualise Where Forecasts Are Available*

Description

Visualise Where Forecasts Are Available

Usage

```
plot_avail_forecasts(
  avail_forecasts,
  y = "model",
  x = "forecast_date",
  make_x_factor = TRUE,
  show_numbers = TRUE
)
```

Arguments

`avail_forecasts`	data.frame with a column called `Number forecasts` as produced by `avail_forecasts()`
`y`	character vector of length one that denotes the name of the column to appear on the y-axis of the plot. Default is "model".
`x`	character vector of length one that denotes the name of the column to appear on the x-axis of the plot. Default is "forecast_date".
`make_x_factor`	logical (default is TRUE). Whether or not to convert the variable on the x-axis to a factor. This has an effect e.g. if dates are shown on the x-axis.
`show_numbers`	logical (default is `TRUE`) that indicates whether or not to show the actual count numbers on the plot

Value

ggplot object with a plot of interval coverage

Examples

```
library(ggplot2)
avail_forecasts <- avail_forecasts(
  example_quantile, by = c("model", "target_type", "target_end_date")
)
plot_avail_forecasts(
 avail_forecasts, x = "target_end_date", show_numbers = FALSE
) +
 facet_wrap("target_type")
```

---

plot_correlation *Plot Correlation Between Metrics*

Description

Plots a heatmap of correlations between different metrics

Usage

```
plot_correlation(correlations)
```

Arguments

`correlations`	A data.table of correlations between scores as produced by `correlation()`.

Value

A ggplot2 object showing a coloured matrix of correlations between metrics

Examples

```
scores <- score(example_quantile)
correlations <- correlation(
 summarise_scores(scores)
)
plot_correlation(correlations)
```

`plot_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

```
plot_heatmap(scores, y = "model", x, metric)
```

Arguments

`scores`	A data.frame of scores based on quantile forecasts as produced by `score()`.
`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

Value

A ggplot2 object showing a heatmap of the desired metric

Examples

```
scores <- score(example_quantile)
scores <- summarise_scores(scores, by = c("model", "target_type", "range"))

plot_heatmap(scores, x = "target_type", metric = "bias")
```

plot_interval_coverage
Plot Interval Coverage

Description

Plot interval coverage

Usage

```
plot_interval_coverage(scores, colour = "model")
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by `score()` or `summarise_scores()`. Note that "range" must be included in the by argument when running `summarise_scores()`
colour	According to which variable shall the graphs be coloured? Default is "model".

Value

ggplot object with a plot of interval coverage

Examples

```
library("scoringutils")
scores <- score(example_quantile)
scores <- summarise_scores(scores, by = c("model", "range"))
plot_interval_coverage(scores)
```

plot_pairwise_comparison
Plot Heatmap of Pairwise Comparisons

Description

Creates a heatmap of the ratios or pvalues from a pairwise comparison between models

Usage

```
plot_pairwise_comparison(
  comparison_result,
  type = c("mean_scores_ratio", "pval", "together"),
  smaller_is_good = TRUE
)
```

Arguments

comparison_result
: A data.frame as produced by pairwise_comparison()

type
: character vector of length one that is either "mean_scores_ratio", "pval", or "together". This denotes whether to visualise the ratio or the p-value of the pairwise comparison or both. Default is "mean_scores_ratio".

smaller_is_good
: logical (default is TRUE) that indicates whether smaller or larger values are to be interpreted as 'good' (as you could just invert the mean scores ratio). This option is not supported when type = "pval"

Examples

```
library(ggplot2)
df <- data.frame(
  model = rep(c("model1", "model2", "model3"), each = 10),
  id = rep(1:10),
  interval_score = abs(rnorm(30, mean = rep(c(1, 1.3, 2), each = 10))),
  ae_median = (abs(rnorm(30)))
)

scores <- score(example_quantile)
pairwise <- pairwise_comparison(scores, by = "target_type")
plot_pairwise_comparison(pairwise) +
  facet_wrap(~target_type)
```

---

plot_pit *PIT Histogram*

Description

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

Usage

```
plot_pit(pit, num_bins = "auto", breaks = NULL)
```

Arguments

pit
: either a vector with the PIT values of size n, or a data.frame as produced by pit()

num_bins
: the number of bins in the PIT histogram, default is "auto". When num_bins == "auto", plot_pit() will either display 10 bins, or it will display a bin for each available quantile in case you passed in data in a quantile-based format. You can control the number of bins by supplying a number. This is fine for sample-based pit histograms, but may fail for quantile-based formats. In this case it is preferred to supply explicit breaks points using the breaks argument.

breaks numeric vector with the break points for the bins in the PIT histogram. This is preferred when creating a PIT histogram based on quantile-based data. Default is NULL and breaks will be determined by num_bins.

Value

vector with the scoring values

Examples

```
# PIT histogram in vector based format
true_values <- rnorm(30, mean = 1:30)
predictions <- replicate(200, rnorm(n = 30, mean = 1:30))
pit <- pit_sample(true_values, predictions)
plot_pit(pit)

# quantile-based pit
pit <- pit(example_quantile, by = c("model"))
plot_pit(pit, breaks = seq(0.1, 1, 0.1))

# sample-based pit
pit <- pit(example_integer, by = c("model"))
plot_pit(pit)
```

---

plot_predictions *Plot Predictions vs True Values*

Description

Make a plot of observed and predicted values

Usage

```
plot_predictions(data, by = NULL, x = "date", range = c(0, 50, 90))
```

Arguments

data	a data.frame that follows the same specifications outlined in score(). To customise your plotting, you can filter your data using the function make_NA().
by	character vector with column names that denote categories by which the plot should be stratified. If for example you want to have a facetted plot, this should be a character vector with the columns used in facetting (note that the facetting still needs to be done outside of the function call)
x	character vector of length one that denotes the name of the variable
range	numeric vector indicating the interval ranges to plot. If 0 is included in range, the median prediction will be shown.

Value

ggplot object with a plot of true vs predicted values

Examples

```
library(ggplot2)
library(magrittr)

example_continuous %>%
  make_NA (
    what = "truth",
    target_end_date >= "2021-07-22",
    target_end_date < "2021-05-01"
  ) %>%
  make_NA (
    what = "forecast",
    model != 'EuroCOVIDhub-ensemble',
    forecast_date != "2021-06-07"
  ) %>%
  plot_predictions (
    x = "target_end_date",
    by = c("target_type", "location"),
    range = c(0, 50, 90, 95)
  ) +
  facet_wrap(~ location + target_type, scales = "free_y") +
  aes(fill = model, color = model)

example_continuous %>%
  make_NA (
    what = "truth",
    target_end_date >= "2021-07-22",
    target_end_date < "2021-05-01"
  ) %>%
  make_NA (
    what = "forecast",
    forecast_date != "2021-06-07"
  ) %>%
  plot_predictions (
    x = "target_end_date",
    by = c("target_type", "location"),
    range = c(0)
  ) +
  facet_wrap(~ location + target_type, scales = "free_y") +
  aes(fill = model, color = model)
```

---

plot_quantile_coverage *Plot Quantile Coverage*

Description

Plot quantile coverage

Usage

```
plot_quantile_coverage(scores, colour = "model")
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by `score()` or `summarise_scores()`. Note that "range" must be included in the by argument when running `summarise_scores()`
colour	According to which variable shall the graphs be coloured? Default is "model".

Value

ggplot object with a plot of interval coverage

Examples

```
scores <- score(example_quantile)
scores <- summarise_scores(scores, by = c("model", "quantile"))
plot_quantile_coverage(scores)
```

`plot_ranges` *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 overall interval score, or how sharpness / dispersion changes by range.

Usage

```
plot_ranges(scores, y = "interval_score", x = "model", colour = "range")
```

Arguments

scores	A data.frame of scores based on quantile forecasts as produced by `score()` or `summarise_scores()`. Note that "range" must be included in the by argument when running `summarise_scores()`
y	The variable from the scores you want to show on the y-Axis. This could be something like "interval_score" (the default) or "dispersion"
x	The variable from the scores you want to show on the x-Axis. Usually this will be "model"
colour	Character vector of length one used to determine a variable for colouring dots. The Default is "range".

Value

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

Examples

```
library(ggplot2)
scores <- score(example_quantile)
scores <- summarise_scores(scores, by = c("model", "target_type", "range"))

plot_ranges(scores, x = "model") +
  facet_wrap(~target_type, scales = "free")

# visualise dispersion instead of interval score
plot_ranges(scores, y = "dispersion", x = "model") +
  facet_wrap(~target_type)
```

---

plot_score_table *Plot Coloured Score Table*

Description

Plots a coloured table of summarised scores obtained using `score()`.

Usage

```
plot_score_table(scores, y = "model", by = NULL, metrics = NULL)
```

Arguments

`scores`	A data.table of scores as produced by `score()`.
`y`	the variable to be shown on the y-axis. Instead of a single character string, you can also 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").
`by`	A character vector that determines how the colour shading for the plot gets computed. By default (`NULL`), shading will be determined per metric, but you can provide additional column names (see examples).
`metrics`	A character vector with the metrics to show. If set to `NULL` (default), all metrics present in `scores` will be shown.

Value

A ggplot2 object with a coloured table of summarised scores

Examples

```
library(ggplot2)
library(magrittr) # pipe operator

scores <- score(example_quantile) %>%
  summarise_scores(by = c("model", "target_type")) %>%
  summarise_scores(fun = signif, digits = 2)

plot_score_table(scores, y = "model", by = "target_type") +
  facet_wrap(~target_type, ncol = 1)

# can also put target description on the y-axis
plot_score_table(scores,
                 y = c("model", "target_type"),
                 by = "target_type")
```

`plot_wis` *Plot Contributions to the Weighted Interval Score*

Description

Visualise the components of the weighted interval score: penalties for over-prediction, under-prediction and for high dispersion (lack of sharpness)

Usage

```
plot_wis(scores, x = "model", relative_contributions = FALSE, flip = FALSE)
```

Arguments

`scores`	A data.frame of scores based on quantile forecasts as produced by `score()` and summarised using `summarise_scores()`
`x`	The variable from the scores you want to show on the x-Axis. Usually this will be "model".
`relative_contributions`	show relative contributions instead of absolute contributions. Default is FALSE and this functionality is not available yet.
`flip`	boolean (default is `FALSE`), whether or not to flip the axes.

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://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008618

Examples

```
library(ggplot2)
scores <- score(example_quantile)
scores <- summarise_scores(scores, by = c("model", "target_type"))

plot_wis(scores,
  x = "model",
  relative_contributions = TRUE
) +
  facet_wrap(~target_type)
plot_wis(scores,
  x = "model",
  relative_contributions = FALSE
) +
  facet_wrap(~target_type, scales = "free_x")
```

---

print.scoringutils_check

Print output from `check_forecasts()`

Description

Helper function that prints the output generated by `check_forecasts()`

Usage

```
## S3 method for class 'scoringutils_check'
print(x, ...)
```

Arguments

x	An object of class 'scoringutils_check' as produced by `check_forecasts()`
...	additional arguments (not used here)

Examples

```
check <- check_forecasts(example_quantile)
print(check)
```

quantile_score *Quantile Score*

Description

Proper Scoring Rule to score quantile predictions. Smaller values are better. The quantile score is closely related to the Interval score (see `interval_score()`) and is the quantile equivalent that works with single quantiles instead of central prediction intervals.

Usage

```
quantile_score(true_values, predictions, quantiles, weigh = TRUE)
```

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.
`quantiles`	vector of size n with the quantile values of the corresponding predictions.
`weigh`	if TRUE, weigh the score by alpha / 2, so it can be averaged into an interval score that, in the limit, corresponds to CRPS. Alpha is the value that corresponds to the (alpha/2) or (1 - alpha/2) quantiles provided and will be computed from the quantile. Alpha is the decimal value that represents how much is outside a central prediction interval (E.g. for a 90 percent central prediction interval, alpha is 0.1). Default: `TRUE`.

Value

vector with the scoring values

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, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008618

Examples

```
true_values <- rnorm(10, mean = 1:10)
alpha <- 0.5

lower <- qnorm(alpha / 2, rnorm(10, mean = 1:10))
upper <- qnorm((1 - alpha / 2), rnorm(10, mean = 1:10))
```

```
qs_lower <- quantile_score(true_values,
  predictions = lower,
  quantiles = alpha / 2
)
qs_upper <- quantile_score(true_values,
  predictions = upper,
  quantiles = 1 - alpha / 2
)
interval_score <- (qs_lower + qs_upper) / 2
```

`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

```
sample_to_quantile(example_integer)
```

score *Evaluate forecasts*

Description

The function `score` allows automatic scoring of forecasts and wraps the lower level functions in the **scoringutils** package.

It can be used to score forecasts in a quantile-based, sample-based, or binary format. To obtain an overview of what input is expected, have a look at the example_quantile, example_continuous, example_integer, and example_binary data sets.

You can (and should) check your input using the function `check_forecasts()` before scoring.

To obtain a quick overview of the evaluation metrics used, have a look at the metrics data included in the package. The column `metrics$Name` gives an overview of all available metric names that can be computed.

Usage

```
score(data, metrics = NULL, ...)
```

Arguments

`data` A data.frame or data.table with the predictions and observations. For examples, look at the example_quantile, example_continuous, example_integer, and example_binary data sets. For scoring using `score()`, 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 scoring 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 scoring predictions in a quantile-format forecast you should provide a column called `quantile`:

- `quantile`: quantile to which the prediction corresponds

`metrics` the metrics you want to have in the output. If `NULL` (the default), all available metrics will be computed. For a list of available metrics see `available_metrics()`, or check the metrics data set.

`...` additional parameters passed down to `score_quantile()` (internal function used for scoring forecasts in a quantile-based format).

Value

A data.table with unsummarised scores. There will be one score per quantile or sample, which is usually not desired, so you should always run `summarise_scores()` on the unsummarised scores.

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. doi:10.1371/journal.pcbi.1006785

Examples

```
library(magrittr) # pipe operator

check_forecasts(example_quantile)
score(example_quantile) %>%
  add_coverage(by = c("model", "target_type")) %>%
  summarise_scores(by = c("model", "target_type"))

# forecast formats with different metrics
score(example_binary)
score(example_quantile)
score(example_integer)
score(example_continuous)

# score point forecasts (marked by 'NA' in the quantile column)
score(example_point) %>%
  summarise_scores(by = "model", na.rm = TRUE)
```

`se_mean_sample` *Squared Error of the Mean (Sample-based Version)*

Description

Squared error of the mean calculated as

$$\text{mean}(\text{true\_value} - \text{prediction})^2$$

Usage

```
se_mean_sample(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

See Also

`squared_error()`

Examples

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

`squared_error` *Squared Error*

Description

Squared Error SE calculated as

$$(\text{true\_values} - \text{predicted\_values})^2$$

Usage

```
squared_error(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)
squared_error(true_values, predicted_values)
```

`summarise_scores` *Summarise scores as produced by* `score()`

Description

Summarise scores as produced by `score()`-

Usage

```
summarise_scores(
  scores,
  by = NULL,
  fun = mean,
  relative_skill = FALSE,
  metric = "auto",
  baseline = NULL,
  ...
)

summarize_scores(
  scores,
  by = NULL,
  fun = mean,
  relative_skill = FALSE,
  metric = "auto",
  baseline = NULL,
  ...
)
```

Arguments

`scores`	A data.table of scores as produced by `score()`.
`by`	character vector with column names to summarise scores by. Default is `NULL`, meaning that the only summary that takes is place is summarising over quantiles (in case of quantile-based forecasts), such that there is one score per forecast as defined by the unit of a single forecast (rather than one score for every quantile).
`fun`	a function used for summarising scores. Default is `mean`.
`relative_skill`	logical, whether or not to compute relative performance between models based on pairwise comparisons. If `TRUE` (default is `FALSE`), then a column called 'model' must be present in the input data. For more information on the computation of relative skill, see `pairwise_comparison()`. Relative skill will be calculated for the aggregation level specified in `by`.
`metric`	character with the name of the metric for which a relative skill shall be computed. If equal to 'auto' (the default), then this will be either interval score, CRPS or Brier score (depending on which of these is available in the input data)

`baseline`	character string with the name of a model. If a baseline is given, then a scaled relative skill with respect to the baseline will be returned. By default (`NULL`), relative skill will not be scaled with respect to a baseline model.
`...`	additional parameters that can be passed to the summary function provided to `fun`. For more information see the documentation of the respective function.

Examples

```
library(magrittr) # pipe operator

# summarise over samples or quantiles to get one score per forecast
scores <- score(example_quantile)
summarise_scores(scores)

# get scores by model
summarise_scores(scores, by = c("model"))

# get scores by model and target type
summarise_scores(scores, by = c("model", "target_type"))

# get standard deviation
summarise_scores(scores, by = "model", fun = sd)

# round digits
summarise_scores(scores, by = c("model")) %>%
  summarise_scores(fun = signif, digits = 2)

# get quantiles of scores
# make sure to aggregate over ranges first
summarise_scores(scores,
  by = "model", fun = quantile,
  probs = c(0.25, 0.5, 0.75)
)

# get ranges
# summarise_scores(scores, by = "range")
```

`theme_scoringutils` *Scoringutils ggplot2 theme*

Description

A theme for ggplot2 plots used in scoringutils

Usage

```
theme_scoringutils()
```

Value

A ggplot2 theme

Index