Package 'staccuracy'

February 23, 2025

Title Standardized Accuracy and Other Model Performance Metrics

Version 0.2.2

Language en-US

Description Standardized accuracy (staccuracy) is a framework for expressing accuracy scores such that 50% represents a reference level of performance and 100% is a perfect prediction. The 'staccuracy' package provides tools for creating staccuracy functions as well as some recommended staccuracy measures. It also provides functions for some classic performance metrics such as mean absolute error (MAE), root mean squared error (RMSE), and area under the receiver operating characteristic curve (AUCROC), as well as their winsorized versions when applicable.

License MIT + file LICENSE

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests autogam, testthat (>= 3.0.0)

Imports cli, dplyr, methods, purrr, rlang, stringr, tidyr

Depends R (>= 4.2.0)

URL https://github.com/tripartio/staccuracy,
https://tripartio.github.io/staccuracy/

BugReports https://github.com/tripartio/staccuracy/issues

NeedsCompilation no

Author Chitu Okoli [aut, cre] (<https://orcid.org/0000-0001-5574-7572>)

Maintainer Chitu Okoli <Chitu.Okoli@skema.edu>

Repository CRAN

Date/Publication 2025-02-23 14:40:02 UTC

Contents

`aucroc` *Area under the ROC curve*

Description

Returns the area under the ROC curve based on comparing the predicted scores to the actual binary values. Tied predictions are handled by calculating the optimistic AUC (positive cases sorted first, resulting in higher AUC) and the pessimistic AUC (positive cases sorted last, resulting in lower AUC) and then returning the average of the two. For the ROC, a "tie" means at least one pair of `pred` predictions whose value is identical yet their corresponding values of `actual` are different. (If the value of `actual` are the same for identical predictions, then these are unproblematic and are not considered "ties".)

Usage

```
aucroc(
  actual,
  pred,
  na.rm = FALSE,
  positive = NULL,
  sample_size = 10000,
  seed = 0
)
```

Arguments

`actual`	any atomic vector. Actual label values from a dataset. They must be binary; that is, there must be exactly two distinct values (other than missing values, which are allowed). The "true" or "positive" class is determined by coercing `actual` to logical `TRUE` and `FALSE` following the rules of `as.logical()`. If this is not the intended meaning of "positive", then specify which of the two values should be considered `TRUE` with the argument `positive`.
`pred`	numeric vector. Predictions corresponding to each respective element in `actual`. Any numeric value (not only probabilities) are permissible.
`na.rm`	logical(1). `TRUE` if missing values should be removed; `FALSE` if they should be retained. If `TRUE`, then if any element of either `actual` or `pred` is missing, its paired element will be also removed.

positive	any single atomic value. The value of actual that is considered TRUE; any other value of actual is considered FALSE. For example, if 2 means TRUE and 1 means FALSE, then set positive = 2.
sample_size	single positive integer. To keep the computation relatively rapid, when actual and pred are longer than sample_size elements, then a random sample of sample_size of actual and pred will be selected and the ROC and AUC will be calculated on this sample. To disable random sampling for long inputs, set sample_size = NA.
seed	numeric(1). Random seed used only if length(actual) > sample_size.

Value

List with the following elements:

- roc_opt: tibble with optimistic ROC data. "Optimistic" means that when predictions are tied, the TRUE/positive actual values are ordered before the FALSE/negative ones.
- roc_pess: tibble with pessimistic ROC data. "Pessimistic" means that when predictions are tied, the FALSE/negative actual values are ordered before the TRUE/positive ones. Note that this difference is not merely in the sort order: when there are ties, the way that true positives, true negatives, etc. are counted is different for optimistic and pessimistic approaches. If there are no tied predictions, then roc_opt and roc_pess are identical.
- auc_opt: area under the ROC curve for optimistic ROC.
- auc_pess: area under the ROC curve for pessimistic ROC.
- auc: mean of auc_opt and auc_pess. If there are no tied predictions, then auc_opt, auc_pess, and auc are identical.
- ties: TRUE if there are two or more tied predictions; FALSE if there are no ties.

Examples

```
set.seed(0)
# Generate some simulated "actual" data
a <- sample(c(TRUE, FALSE), 50, replace = TRUE)

# Generate some simulated predictions
p <- runif(50) |> round(2)
p[c(7, 8, 22, 35, 40, 41)] <- 0.5

# Calculate AUCROC with its components
ar <- aucroc(a, p)
ar$auc
```

mae *Regression error and deviation measures*

Description

These are standard error and deviation measures for numeric data. "Deviation" means the natural variation of the values of a numeric vector around its central tendency (usually the mean or median). "Error" means the average discrepancy between the actual values of a numeric vector and its predicted values.

Usage

```
mae(actual, pred, na.rm = FALSE)

rmse(actual, pred, na.rm = FALSE)

mad(x, na.rm = FALSE, version = "mean", ...)
```

Arguments

actual	numeric vector. Actual (true) values of target outcome data.
pred	numeric vector. Predictions corresponding to each respective element in actual.
na.rm	logical(1). TRUE if missing values should be removed; FALSE if they should be retained. If TRUE, then if any element of either actual or pred is missing, its paired element will be also removed.
x	numeric vector. Values for which to calculate the MAD.
version	character(1). By default (version = 'mean'), mad() returns the mean absolute deviation (MAD) of values relative to their mean. If version = 'median', it calls the stats::mad() function instead, the median absolute deviation relative to their median (MedAD, sometimes also called MAD). Any other value gives an error. See details.
...	Arguments to pass to stats::mad() if version = 'median'. See the version argument for details.

Details

Mean absolute deviation (MAD)

mad() returns the mean absolute deviation (MAD) of values relative to their mean. This is useful as a default benchmark for the mean absolute error (MAE), as the standard deviation (SD) is a default benchmark for the root mean square error (RMSE).

NOTE: This function name overrides stats::mad() (median absolute deviation relative to their median). To maintain the functionality of stats::mad(), specify the version argument.

Value

In all cases, if any value in `actual` or `pred` is `NA` and `na.rm = FALSE`, then the function returns `NA`.

`mae()` returns the mean absolute error (MAE) of predicted values `pred` compared to the `actual` values.

`rmse()` returns the root mean squared error (RMSE) of predicted values `pred` compared to the `actual` values.

`mad()` returns either the mean absolute deviation (MAD) of values relative to their mean (default) or the median absolute deviation relative to their median. See details.

Examples

```
a <- c(3, 5, 2, 7, 9, 4, 6, 8, 1, 10)
p <- c(2.5, 5.5, 2, 6.5, 9.5, 3.5, 6, 7.5, 1.5, 9.5)
mae(a, p)

rmse(a, p)

mad(a)
```

---

reg_aucroc *Area under the ROC curve for regression target outcomes*

Description

Area under the ROC curve (AUCROC) is a classification measure. By dichotomizing the range of `actual` values, `reg_aucroc()` turns regression evaluation into classification evaluation for any regression model. Note that the model that generates the predictions is assumed to be a regression model; however, any numeric inputs are allowed for the `pred` argument, so there is no check for the nature of the source model.

Usage

```
reg_aucroc(
  actual,
  pred,
  num_quants = 100,
  ...,
  cuts = NULL,
  imbalance = 0.05,
  na.rm = FALSE,
  sample_size = 10000,
  seed = 0
)
```

Arguments

actual	numeric vector. Actual label values from a dataset. They must be numeric.
pred	numeric vector. Predictions corresponding to each respective element in actual.
num_quants	scalar positive integer. If cuts is NULL (default), actual will be dichotomized into quants quantiles and that many ROCs will be returned in the rocs element. However, if cuts is specified, then quants is ignored.
...	Not used. Forces explicit naming of the arguments that follow.
cuts	numeric vector. If cuts is provided, it overrides quants to specify the cut points for dichotomization of actual for the creation of cuts + 1 ROCs.
imbalance	numeric(1) in (0, 0.5]. The result element mean_auc averages the AUCs over three regions (see details of the return value). imbalance is the supposed percentage of the less frequent class in the data. If not provided, defaults to 0.05 (5%).
na.rm	See documentation for aucroc()
sample_size	See documentation for aucroc(). In addition to those notes, for reg_aucroc(), any sampling is conducted before the dichotomization of actual so that all classification ROCs are based on identical data.
seed	See documentation for aucroc()

Details

The ROC data and AUCROC values are calculated with aucroc().

Value

List with the following elements:

- rocs: List of results for aucroc() for each dichotomized segment of actual.
- auc: named numeric vector of AUC extracted from each element of rocs. Named by the percentile that the AUC represents.
- mean_auc: named numeric(3). The average AUC over the low, middle, and high quantiles of dichotomization:
- lo: average AUC with imbalance% (e.g., 5%) or less of the actual target values;
- mid: average AUC in between lo and hi;
- hi: average AUC with (1 - imbalance)% (e.g., 95%) or more of the actual target values;

Examples

```
# Remove rows with missing values from airquality dataset
airq <- airquality |>
  na.omit()

# Create binary version where the target variable 'Ozone' is dichotomized based on its median
airq_bin <- airq
airq_bin$Ozone <- airq_bin$Ozone >= median(airq_bin$Ozone)
```

```
# Create a generic regression model; use autogam
req_aq   <- autogam::autogam(airq, 'Ozone', family = gaussian())
req_aq$perf$sa_wmae_mad  # Standardized accuracy for regression

# Create a generic classification model; use autogam
class_aq <- autogam::autogam(airq_bin, 'Ozone', family = binomial())
class_aq$perf$auc  # AUC (standardized accuracy for classification)

# Compute AUC for regression predictions
reg_auc_aq <- reg_aucroc(
  airq$Ozone,
  predict(req_aq)
)

# Average AUC over the lo, mid, and hi quantiles of dichotomization:
reg_auc_aq$mean_auc
```

`sa_diff` *Statistical tests for the differences between standardized accuracies (staccuracies)*

Description

Because the distribution of staccuracies is uncertain (and indeed, different staccuracies likely have different distributions), bootstrapping is used to empirically estimate the distributions and calculate the p-values. See the return value description for details on what the function provides.

Usage

```
sa_diff(
  actual,
  preds,
  ...,
  na.rm = FALSE,
  sa = NULL,
  pct = c(0.01, 0.02, 0.03, 0.04, 0.05),
  boot_alpha = 0.05,
  boot_it = 1000,
  seed = 0
)
```

Arguments

`actual`	numeric vector. The actual (true) labels.

preds	named list of at least two numeric vectors. Each element is a vector of the same length as actual with predictions for each row corresponding to each element of actual. The names of the list elements should be the names of the models that produced each respective prediction; these names will be used to distinguish the results.
...	not used. Forces explicit naming of subsequent arguments.
na.rm	See documentation for staccuracy()
sa	list of functions. Each element is the unquoted name of a valid staccuracy function (see staccuracy() for the required function signature.) If an element is named, the name will be displayed as the value of the sa column of the result. Otherwise, the function name will be displayed. If NULL (default), staccuracy functions will be automatically selected based on the datatypes of actual and preds.
pct	numeric with values from (0, 1). The percentage values on which the difference in staccuracies will be tested.
boot_alpha	numeric(1) from 0 to 1. Alpha for percentile-based confidence interval range for the bootstrapped means; the bootstrap confidence intervals will be the lowest and highest (1 - 0.05) / 2 percentiles. For example, if boot_alpha = 0.05 (default), the intervals will be at the 2.5 and 97.5 percentiles.
boot_it	positive integer(1). The number of bootstrap iterations.
seed	integer(1). Random seed for the bootstrap sampling. Supply this between runs to assure identical results.

Value

tibble with staccuracy difference results:

- staccuracy: name of staccuracy measure
- pred: Each named element (model name) in the input preds. The row values give the staccuracy for that prediction. When pred is NA, the row represents the difference between prediction staccuracies (diff) instead of staccuracies themselves.
- diff: When diff takes the form 'model1-model2', then the row values give the difference in staccuracies between two named elements (model names) in the input preds. When diff is NA, the row instead represents the staccuracy of a specific model prediction (pred).
- lo, mean, hi: The lower bound, mean, and upper bound of the bootstrapped staccuracy. The lower and upper bounds are confidence intervals specified by the input boot_alpha.
- p__: p-values that the difference in staccuracies are at least the specified percentage amount or greater. E.g., for the default input pct = c(0.01, 0.02, 0.03, 0.04, 0.05), these columns would be p01, p02, p03, p04, and p05. As they apply only to differences between staccuracies, they are provided only for diff rows and are NA for pred rows. As an example of their meaning, if the mean difference for 'model1-model2' is 0.0832 with p01 of 0.012 and p02 of 0.035, then 1.2% of bootstrapped staccuracies had a model1 - model2 difference of less than 0.01 and 3.5% were less than 0.02. (That is, 98.8% of differences were greater than 0.01 and 96.5% were greater than 0.02.)

Examples

```
lm_attitude_all <- lm(rating ~ ., data = attitude)
lm_attitude__a <- lm(rating ~ . - advance, data = attitude)
lm_attitude__c <- lm(rating ~ . - complaints, data = attitude)

sdf <- sa_diff(
  attitude$rating,
  list(
    all = predict(lm_attitude_all),
    madv = predict(lm_attitude__a),
    mcmp = predict(lm_attitude__c)
  ),
  boot_it = 10
)
sdf
```

`staccuracy` *Standardized accuracy (staccuracy) functions.*

Description

Standardized accuracy (staccuracy) represents error or accuracy measures on a scale where 1 or 100% means perfect prediction and 0.5 or 50% is a reference comparison of some specified standard performance. Higher than 0.5 is better than the reference and below 0.5 is worse. 0 might or might not have a special meaning; sometimes negative scores are possible, but these often indicate modelling errors.

The core function is `staccuracy()`, which receives as input a generic error function and a reference function against which to compare the error function performance. In addition, the following recommended staccuracy functions are provided:

- `sa_mae_mad`: standardized accuracy of the mean absolute error (MAE) based on the mean absolute deviation (MAD)
- `sa_rmse_sd`: standardized accuracy of the root mean squared error (RMSE) based on the standard deviation (SD)
- `sa_wmae_mad`: standardized accuracy of the winsorized mean absolute error (MAE) based on the mean absolute deviation (MAD)
- `sa_wrmse_sd`: standardized accuracy of the winsorized root mean squared error (RMSE) based on the standard deviation (SD)

Usage

```
staccuracy(error_fun, ref_fun)

sa_mae_mad(actual, pred, na.rm = FALSE)

sa_wmae_mad(actual, pred, na.rm = FALSE)
```

```
sa_rmse_sd(actual, pred, na.rm = FALSE)

sa_wrmse_sd(actual, pred, na.rm = FALSE)
```

Arguments

error_fun	function. The unquoted name of the function that calculates the error (or accuracy) measure. This function must be of the signature function(actual, pred, na.rm = FALSE).
ref_fun	function. The unquoted name of the function that calculates the reference error, accuracy, or deviation measure. This function must be of the signature ref_fun(actual, na.rm = FALSE).
actual	numeric. The true (actual) labels.
pred	numeric. The predicted estimates. Must be the same length as actual.
na.rm	logical(1). Whether NA values should be removed (TRUE) or not (FALSE, default).

Details

The core function staccuracy() receives as input a generic error function and a reference function against which to compare the error function's performance. These input functions must have the following signatures (see the argument specifications for details of the arguments):

- error_fun: function(actual, pred, na.rm = na.rm); the output must be a scalar numeric (that is, a single number).
- error_fun: function(actual, pred, na.rm = na.rm); the output must be a scalar numeric (that is, a single number).

Value

staccuracy() returns a function with signature function(actual, pred, na.rm = FALSE) that receives an actual and a pred vector as inputs and returns the staccuracy of the originally input error function based on the input reference function.

The convenience sa_*() functions return the staccuracy measures specified above.

Examples

```
# Here's some data
actual_1 <- c(2.3, 4.5, 1.8, 7.6, 3.2)

# Here are some predictions of that data
predicted_1 <- c(2.5, 4.2, 1.9, 7.4, 3.0)

# MAE measures the average error in the predictions
mae(actual_1, predicted_1)

# But how good is that?
# MAD gives the natural variation in the actual data; this is a point of comparison.
mad(actual_1)
```

```
# So, our predictions are better (lower) than the MAD, but how good, really?
# Create a standardized accuracy function to give us an easily interpretable metric:
my_mae_vs_mad_sa <- staccuracy(mae, mad)

# Now use it
my_mae_vs_mad_sa(actual_1, predicted_1)

# That's 94.2% standardized accuracy compared to the MAD. Pretty good!
```

`winsorize` *Winsorize a numeric vector*

Description

Winsorization means truncating the extremes of a numeric range by replacing extreme values with a predetermined minimum and maximum. `winsorize()` returns the input vector values with values less than or greater than the provided minimum or maximum replaced by the provided minimum or maximum, respectively.

`win_mae()` and `win_rmse()` return MAE and RMSE respectively with winsorized predictions. The fundamental idea underlying the winsorization of predictions is that if the actual data has well-defined bounds, then models should not be penalized for being overzealous in predicting beyond the extremes of the data. Models that are overzealous in the boundaries might sometimes be superior within normal ranges; the extremes can be easily corrected by winsorization.

Usage

```
winsorize(x, win_range)

win_mae(actual, pred, win_range = range(actual), na.rm = FALSE)

win_rmse(actual, pred, win_range = range(actual), na.rm = FALSE)
```

Arguments

`x`	numeric vector.
`win_range`	numeric(2). The minimum and maximum allowable values for the `pred` predictions or for `x`. For functions with `pred`, `win_range` defaults to the minimum and maximum values of the provided `actual` values. For functions with `x`, there is no default.
`actual`	numeric vector. Actual (true) values of target outcome data.
`pred`	numeric vector. Predictions corresponding to each respective element in `actual`.
`na.rm`	logical(1). `TRUE` if missing values should be removed; `FALSE` if they should be retained. If `TRUE`, then if any element of either `actual` or `pred` is missing, its paired element will be also removed.

Value

`winsorize()` returns a winsorized vector.

`win_mae()` returns the mean absolute error (MAE) of winsorized predicted values `pred` compared to the `actual` values. See `mae()` for details.

`win_rmse()` returns the root mean squared error (RMSE) of winsorized predicted values `pred` compared to the `actual` values. See `rmse()` for details.

Examples

```
a <- c(3, 5, 2, 7, 9, 4, 6, 8, 2, 10)
p <- c(2.5, 5.5, 1.5, 6.5, 10.5, 3.5, 6, 7.5, 0.5, 11.5)

a  # the original data
winsorize(a, c(2, 8))  # a winsorized on defined boundaries

# range of the original data
a
range(a)

# some overzealous predictions
p
range(p)

# MAE penalizes overzealous predictions
mae(a, p)

# Winsorized MAE forgives overzealous predictions
win_mae(a, p)

# RMSE penalizes overzealous predictions
rmse(a, p)

# Winsorized RMSE forgives overzealous predictions
win_rmse(a, p)
```

Index