Package 'yardstick'

November 12, 2017

Type Package

Title Tidy Characterizations of Model Performance

Version 0.0.1

Maintainer Max Kuhn <max@rstudio.com>

Description Tidy tools for quantifying how well model fits to a data set such as confusion matrices, class probability curve summaries, and regression metrics (e.g., RMSE).

URL https://github.com/topepo/yardstick

BugReports https://github.com/topepo/yardstick/issues

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Depends R (>= 2.10), broom

Suggests testthat, covr

Imports dplyr, tibble, tidyr, utils, pROC, MLmetrics, rlang, tidyselect

NeedsCompilation no

Author Max Kuhn [aut, cre], RStudio [cph]

Repository CRAN

Date/Publication 2017-11-12 13:45:42 UTC

R topics documented:

`accuracy` *Classification Metrics on Predited Classes*

Description

General metrics for classification models

Usage

```
accuracy(data, ...)

## S3 method for class 'data.frame'
accuracy(data, truth, estimate, na.rm = TRUE, ...)

## S3 method for class 'table'
accuracy(data, ...)

## S3 method for class 'matrix'
accuracy(data, ...)
```

Arguments

`data`	For the default functions, a factor containing the discrete measurements. For the `table` or `matrix` functions, a table or matrix object, respectively, where the true class results should be in the columns of the table.
`...`	Not currently used.
`truth`	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
`estimate`	The column identifier for the predicted class results (that is also factor). As with `truth` this can be specified different ways but the primary method is to use an unquoted variable name.
`na.rm`	A logical value indicating whether `NA` values should be stripped before the computation proceeds

Author(s)

Max Kuhn

See Also

conf_mat()

---

conf_mat *Confusion Matrix for Categorical Data*

Description

Calculates a cross-tabulation of observed and predicted classes.

For conf_mat() objects, the tidy method collapses the cell counts by cell into a data frame for each manipulation.

Usage

```
conf_mat(data, ...)

## S3 method for class 'data.frame'
conf_mat(data, truth, estimate, dnn = c("Prediction",
  "Truth"), ...)

## S3 method for class 'table'
conf_mat(data, ...)

## S3 method for class 'conf_mat'
tidy(x, ...)
```

Arguments

data	A data frame or a base::table().
...	Options to pass to base::table() (not including dnn). This argument is not currently used for the tidy method.
truth	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
estimate	The column identifier for the predicted class results (that is also factor). As with truth this can be specified different ways but the primary method is to use an unquoted variable name.
dnn	a character vector of dimnames for the table
x	A object of class conf_mat().

Details

The function requires that the factors have exactly the same levels.

Value

`conf_mat` produces a object with class `conf_mat`. This contains the table and other objects. `tidy.conf_mat` generates a tibble with columns `name` (the cell identifier) and `value` (the cell count).

Examples

```
library(dplyr)
data("hpc_cv")

# The confusion matrix from a single assessment set (i.e. fold)
hpc_cv %>%
  filter(Resample == "Fold01") %>%
  conf_mat(obs, pred)

# Now compute the average confusion matrix across all folds in
# terms of the proportion of the data contained in each cell.
# First get the raw cell counts per fold using the `tidy` method
cells_per_resample <- hpc_cv %>%
  group_by(Resample) %>%
  do(tidy(conf_mat(., obs, pred)))

# Get the totals per resample
counts_per_resample <- hpc_cv %>%
  group_by(Resample) %>%
  summarize(total = n()) %>%
  left_join(cells_per_resample, by = "Resample") %>%
  # Compute the proportions
  mutate(prop = value/total) %>%
  group_by(name) %>%
  # Average
  summarize(prop = mean(prop))

counts_per_resample

# Now reshape these into a matrix
mean_cmat <- matrix(counts_per_resample$prop, byrow = TRUE, ncol = 4)
rownames(mean_cmat) <- levels(hpc_cv$obs)
colnames(mean_cmat) <- levels(hpc_cv$obs)

round(mean_cmat, 3)
```

---

hpc_cv *Class Probability Predictions*

Description

Class Probability Predictions

Details

This data frame contains the predicted classes and class probabilities for a linear discriminant analysis model fit to the HPC data set from Kuhn and Johnson (2013). These data are the assessment sets from a 10-fold cross-validation scheme. The data column columns for the true class (`obs`), the class prediction (`pred`) and columns for each class probability (columns `VF`, `F`, `M`, and `L`). Additionally, a column for the resample indicator is included.

Value

`hpc_cv` a data frame

Source

Kuhn, M., Johnson, K. (2013) *Applied Predictive Modeling*, Springer

Examples

```
data(hpc_cv)
str(hpc_cv)
```

---

mcc *Other Metrics for 2x2 Tables*

Description

General metrics for two class problems that are not already in `sens()` or `recall()` are here, such as the Matthews correlation coefficient, Youden's J.

There is no common convention on which factor level should automatically be considered the "event" or "positive" results. In `yardstick`, the default is to use the *first* level. To change this, a global option called `yardstick.event_first` is set to `TRUE` when the package is loaded. This can be changed to `FALSE` if the last level of the factor is considered the level of interest.

Usage

```
mcc(data, ...)

## S3 method for class 'data.frame'
mcc(data, truth, estimate, na.rm = TRUE, ...)

## S3 method for class 'table'
mcc(data, ...)

j_index(data, ...)
```

```
## S3 method for class 'data.frame'
j_index(data, truth, estimate, na.rm = TRUE, ...)

## S3 method for class 'table'
j_index(data, ...)
```

Arguments

data	For the default functions, a factor containing the discrete measurements. For the table or matrix functions, a table or matrix object, respectively, where the true class results should be in the columns of the table.
...	Not currently used.
truth	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
estimate	The column identifier for the predicted class results (that is also factor). As with truth this can be specified different ways but the primary method is to use an unquoted variable name.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds

Details

If more than one statistic is required, it is more computationally efficient to create the confusion matrix using conf_mat() and applying the corresponding summary method (summary.conf_mat()) to get the values at once.

Author(s)

Max Kuhn

See Also

conf_mat(), summary.conf_mat(), recall(), sens()

Examples

```
data("two_class_example")

mcc(two_class_example, truth, predicted)

j_index(two_class_example, truth, predicted)
```

`metrics` *General Function to Estimate Performance*

Description

This function estimates one or more common performance estimates depending on the class of `truth` (see **Value** below) and returns them in a single row tibble.

Usage

```
metrics(data, ...)

## S3 method for class 'data.frame'
metrics(data, truth, estimate, ..., options = list(),
  na.rm = TRUE)
```

Arguments

`data`	A data frame
`...`	For classification: a set of unquoted column names or one or more `dplyr` selector functions to choose which variables contain the class probabilities. See the examples below. For `roc_auc` and `pr_auc`, only one value is required. If more are given, the functions will try to match the column name to the appropriate factor level of `truth`. If this doesn't work, an error is thrown. For `mnLogLoss`, there should be as many columns as factor levels of `truth`. It is **assumed** that they are in the same order as the factor levels.
`truth`	The column identifier for the true results (that is numeric or factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
`estimate`	The column identifier for the predicted results (that is also numeric or factor). As with `truth` this can be specified different ways but the primary method is to use an unquoted variable name.
`options`	Options to pass to `roc()` such as `direction` or `smooth`. These options should not include `response`, `predictor`, or `levels`.
`na.rm`	A logical value indicating whether `NA` values should be stripped before the computation proceeds.

Value

A single row tibble. When `truth` is a factor, there is an `accuracy()` column. If a full set of class probability columns are passed to `...`, then there is also a column for `mnLogLoss()`. When `truth` has two levels and there are class probabilities, `roc_auc()` is appended. When `truth` is numeric, there are columns for `rmse()` and `rsq()`,

A number or `NA`

pathology *Liver Pathology Data*

Description

Liver Pathology Data

Details

These data have the results of a *x*-ray examination to determine whether liver is abnormal or not (in the `scan` column) versus the more extensive pathology results that approximate the truth (in `pathology`).

Value

`pathology` a data frame

Source

Altman, D.G., Bland, J.M. (1994) "Diagnostic tests 1: sensitivity and specificity," *British Medical Journal*, vol 308, 1552.

Examples

```
data(pathology)
str(pathology)
```

recall *Calculate recall, precision and F values*

Description

These functions calculate the recall, precision or F values of a measurement system for finding/retrieving relevant documents compared to reference results (the truth regarding relevance). The measurement and "truth" data must have the same two possible outcomes and one of the outcomes must be thought of as a "relevant" results.

Usage

```
recall(data, ...)

## S3 method for class 'table'
recall(data, ...)

## S3 method for class 'data.frame'
recall(data, truth, estimate, na.rm = TRUE, ...)
```

```
precision(data, ...)

## S3 method for class 'data.frame'
precision(data, truth, estimate, na.rm = TRUE, ...)

## S3 method for class 'table'
precision(data, ...)

f_meas(data, ...)

## Default S3 method:
f_meas(data, truth, estimate, beta = 1, na.rm = TRUE, ...)

## S3 method for class 'table'
f_meas(data, beta = 1, ...)
```

Arguments

data	For the default functions, a factor containing the discrete measurements. For the table or matrix functions, a table or matrix object, respectively, where the true class results should be in the columns of the table.
...	Not currently used.
truth	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
estimate	The column identifier for the predicted class results (that is also factor). As with truth this can be specified different ways but the primary method is to use an unquoted variable name.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds
beta	A numeric value used to weight precision and recall. A value of 1 is traditionally used and corresponds to the harmonic mean of the two values but other values weight recall beta times more important than precision.

Details

The recall (aka specificity) is defined as the proportion of relevant results out of the number of samples which were actually relevant. When there are no relevant results, recall is not defined and a value of NA is returned.

The precision is percentage of predicted truly relevant results of the total number of predicted relevant results and characterizes the "purity in retrieval performance" (Buckland and Gey, 1994).

The measure "F" is a combination of precision and recall (see below).

There is no common convention on which factor level should automatically be considered the relevant result. In yardstick, the default is to use the *first* level. To change this, a global option called yardstick.event_first is set to TRUE when the package is loaded. This can be changed to FALSE if the last level of the factor is considered the level of interest.

Suppose a 2x2 table with notation

	Reference	
Predicted	relevant	Irrelevant
relevant	A	B
Irrelevant	C	D

The formulas used here are:

$$recall = A/(A + C)$$

$$precision = A/(A + B)$$

$$F_i = (1 + i^2) * prec * recall/((i^2 * precision) + recall)$$

See the references for discussions of the statistics.

If more than one statistic is required, it is more computationally efficient to create the confusion matrix using `conf_mat()` and applying the corresponding `summary` method (`summary.conf_mat()`) to get the values at once.

References

Buckland, M., & Gey, F. (1994). The relationship between Recall and Precision. *Journal of the American Society for Information Science*, 45(1), 12-19.

Powers, D. (2007). Evaluation: From Precision, Recall and F Factor to ROC, Informedness, Markedness and Correlation. Technical Report SIE-07-001, Flinders University

See Also

`conf_mat()`, `summary.conf_mat()`, `sens()`, `mcc()`

Examples

```
data("two_class_example")

# Different methods for calling the functions:
precision(two_class_example, truth = truth, estimate = predicted)

recall(two_class_example, truth = "truth", estimate = "predicted")

truth_var <- quote(truth)
f_meas(two_class_example, !! truth_var, predicted)
```

rmse *Calculate Metrics for Numeric Outcomes*

Description

These functions are appropriate for cases where the model outcome is a number. The root mean squared error (rmse) and mean absolute error (mae) are error measures that are in the same units as the original data. The two estimates for the coefficient of determination, rsq and rsq_trad, differ by their formula. The former guarantees a value on (0, 1) while the latter can generate inaccurate values when the model is non-informative (see the examples). Both are measures of consistency/correlation and not of accuracy. The concordance correlation coefficient (ccc) is a measure of both.

Usage

```
rmse(data, ...)

## S3 method for class 'data.frame'
rmse(data, truth, estimate, na.rm = TRUE, ...)

rsq(data, ...)

## S3 method for class 'data.frame'
rsq(data, truth, estimate, na.rm = TRUE, ...)

rsq_trad(data, ...)

## S3 method for class 'data.frame'
rsq_trad(data, truth, estimate, na.rm = TRUE, ...)

mae(data, ...)

## S3 method for class 'data.frame'
mae(data, truth, estimate, na.rm = TRUE, ...)

ccc(data, ...)

## S3 method for class 'data.frame'
ccc(data, truth, estimate, bias = FALSE, na.rm = TRUE,
  ...)
```

Arguments

data	A data frame
...	Not currently used.

truth	The column identifier for the true results (that is numeric). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
estimate	The column identifier for the predicted results (that is also numeric). As with truth this can be specified different ways but the primary method is to use an unquoted variable name.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds.
bias	A logical; should the biased estimate of variance be used for the concordance correlation coefficient (as is Lin (1989))?

Value

A number or NA

Author(s)

Max Kuhn

References

Kvalseth. Cautionary note about R^2. American Statistician (1985) vol. 39 (4) pp. 279-285.

Lin, L. (1989). A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45 (1), 255–268.

Nickerson, C. (1997). A note on" A concordance correlation coefficient to evaluate reproducibility". *Biometrics*, 53(4), 1503-1507.

Examples

```
rmse(solubility_test, truth = solubility, estimate = prediction)
mae(solubility_test, truth = solubility, estimate = prediction)

rsq(solubility_test, solubility, prediction)

set.seed(2291)
solubility_test$randomized <- sample(solubility_test$prediction)
rsq(solubility_test, solubility, randomized)
rsq_trad(solubility_test, solubility, randomized)

ccc(solubility_test, solubility, prediction)
```

roc_auc *Metrics Based on Class Probabilities*

Description

These functions compute the areas under the receiver operating characteristic (ROC) curve (`roc_auc`), the precision-recall curve (`pr_auc`), or the multinomial log loss (`mnLogLoss`).

Usage

```
roc_auc(data, ...)

## S3 method for class 'data.frame'
roc_auc(data, truth, ..., options = list(),
  na.rm = TRUE)

pr_auc(data, ...)

## S3 method for class 'data.frame'
pr_auc(data, truth, ..., na.rm = TRUE)

mnLogLoss(data, ...)

## S3 method for class 'data.frame'
mnLogLoss(data, truth, ..., na.rm = TRUE, sum = FALSE)
```

Arguments

`data`	A data frame with the relevant columns.
`...`	A set of unquoted column names or one or more `dplyr` selector functions to choose which variables contain the class probabilities. See the examples below. For `roc_auc` and `pr_auc`, only one value is required. If more are given, the functions will try to match the column name to the appropriate factor level of `truth`. If this doesn't work, an error is thrown. For `mnLogLoss`, there should be as many columns as factor levels of `truth`. It is **assumed** that they are in the same order as the factor levels.
`truth`	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
`options`	Options to pass to `roc()` such as `direction` or `smooth`. These options should not include `response`, `predictor`, or `levels`.
`na.rm`	A logical value indicating whether `NA` values should be stripped before the computation proceeds
`sum`	A logical. Should the sum of the likelihood contrinbutions be returned (instead of the mean value)?

Details

There is no common convention on which factor level should automatically be considered the "relevant" or "positive" results. In yardstick, the default is to use the *first* level. To change this, a global option called yardstick.event_first is set to TRUE when the package is loaded. This can be changed to FALSE if the last level of the factor is considered the level of interest.

Value

A number between 0 and 1 (or NA) for roc_auc or pr_auc. For mnLogLoss a number or NA.

See Also

conf_mat(), summary.conf_mat(), recall(), mcc()

Examples

```
library(tidyselect)

data("two_class_example")
prob_cols <- levels(two_class_example$truth)

roc_auc(two_class_example, truth = truth, Class1)
# warning is issued here because 2 columns are selected:
roc_auc(two_class_example, truth, starts_with("Class"))

# passing options via a list and _not_ `...`
roc_auc(two_class_example, truth = "truth", Class1,
        options = list(smooth = TRUE))

pr_auc(two_class_example, truth, prob_cols)

mnLogLoss(two_class_example, truth, starts_with("Class"))
# or
mnLogLoss(two_class_example, truth, !! prob_cols)
```

---

sens *Calculate sensitivity, specificity and predictive values*

Description

These functions calculate the sensitivity, specificity or predictive values of a measurement system compared to a reference results (the truth or a gold standard). The measurement and "truth" data must have the same two possible outcomes and one of the outcomes must be thought of as a "positive" results or the "event".

Usage

```
sens(data, ...)

## S3 method for class 'data.frame'
sens(data, truth, estimate, na.rm = TRUE, ...)

## S3 method for class 'table'
sens(data, ...)

## S3 method for class 'matrix'
sens(data, ...)

## S3 method for class 'data.frame'
spec(data, truth, estimate, na.rm = TRUE, ...)

ppv(data, ...)

## S3 method for class 'table'
ppv(data, prevalence = NULL, ...)

## S3 method for class 'matrix'
ppv(data, prevalence = NULL, ...)

npv(data, ...)

## S3 method for class 'table'
npv(data, prevalence = NULL, ...)

## S3 method for class 'matrix'
npv(data, prevalence = NULL, ...)
```

Arguments

data	For the default functions, a factor containing the discrete measurements. For the table or matrix functions, a table or matrix object, respectively, where the true class results should be in the columns of the table.
...	Not currently used.
truth	The column identifier for the true class results (that is a factor). This should an unquoted column name although this argument is passed by expression and support quasiquotation (you can unquote column names or column positions).
estimate	The column identifier for the predicted class results (that is also factor). As with truth this can be specified different ways but the primary method is to use an unquoted variable name.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds
prevalence	A numeric value for the rate of the "positive" class of the data.

Details

The sensitivity is defined as the proportion of positive results out of the number of samples which were actually positive. When there are no positive results, sensitivity is not defined and a value of `NA` is returned. Similarly, when there are no negative results, specificity is not defined and a value of `NA` is returned. Similar statements are true for predictive values.

The positive predictive value is defined as the percent of predicted positives that are actually positive while the negative predictive value is defined as the percent of negative positives that are actually negative.

There is no common convention on which factor level should automatically be considered the "event" or "positive" results. In `yardstick`, the default is to use the *first* level. To change this, a global option called `yardstick.event_first` is set to `TRUE` when the package is loaded. This can be changed to `FALSE` if the last level of the factor is considered the level of interest.

Suppose a 2x2 table with notation

	Reference	
Predicted	Event	No Event
Event	A	B
No Event	C	D

The formulas used here are:

$$Sensitivity = A/(A+C)$$

$$Specificity = D/(B+D)$$

$$Prevalence = (A+C)/(A+B+C+D)$$

$$PPV = (sensitivity*Prevalence)/((sensitivity*Prevalence)+((1-specificity)*(1-Prevalence)))$$

$$NPV = (specificity*(1-Prevalence))/(((1-sensitivity)*Prevalence)+((specificity)*(1-Prevalence)))$$

See the references for discussions of the statistics.

If more than one statistic is required, it is more computationally efficient to create the confusion matrix using `conf_mat()` and applying the corresponding `summary` method (`summary.conf_mat()`) to get the values at once.

Value

A number between 0 and 1 (or NA).

References

Altman, D.G., Bland, J.M. (1994) "Diagnostic tests 1: sensitivity and specificity," *British Medical Journal*, vol 308, 1552.

Altman, D.G., Bland, J.M. (1994) "Diagnostic tests 2: predictive values," *British Medical Journal*, vol 309, 102.

See Also

`conf_mat()`, `summary.conf_mat()`, `recall()`, `mcc()`

Examples

```
data("two_class_example")

# Given that a sample is Class 1,
#   what is the probability that is predicted as Class 1?
sens(two_class_example, truth = truth, estimate = predicted)

# Given that a sample is predicted to be Class 1,
#  what is the probability that it truly is Class 1?
ppv(two_class_example, truth = truth, estimate = predicted)

# But what if we think that Class 1 only occurs 40% of the time?
ppv(two_class_example, truth, predicted, prevalence = 0.40)
```

---

solubility_test *Solubility Predictions from MARS Model*

Description

Solubility Predictions from MARS Model

Details

For the solubility data in Kuhn and Johnson (2013), these data are the test set results for the MARS model. The observed solubility (in column `solubility`) and the model results (`prediction`) are contained in the data.

Value

`solubility_test`
: a data frame

Source

Kuhn, M., Johnson, K. (2013) *Applied Predictive Modeling*, Springer

Examples

```
data(solubility_test)
str(solubility_test)
```

summary.conf_mat *Summary Statistics for Confusion Matrices*

Description

Various statistical summaries of confusion matrices are produced and returned in a easily used format. These potentially include those shown in the help pages for `sens()`, `recall()`, and `accuracy()`.

Usage

```
## S3 method for class 'conf_mat'
summary(object, prevalence = NULL, beta = 1,
  wide = FALSE, ...)
```

Arguments

`object`	An object of class `conf_mat()`.
`prevalence`	A number in `(0, 1)` for the prevalence (i.e. prior) of the event. If left to the default, the data are used to derive this value.
`beta`	A numeric value used to weight precision and recall for `f_meas()`.
`wide`	A single logical value: should there be one row and columns for each statistic (`wide = TRUE`) or a column for the statistic name (`name`) and the estimate (`value`).
`...`	Not currently used.

Details

There is no common convention on which factor level should automatically be considered the "event" or "positive" results. In `yardstick`, the default is to use the *first* level. To change this, a global option called `yardstick.event_first` is set to `TRUE` when the package is loaded. This can be changed to `FALSE` if the last level of the factor is considered the level of interest.

Value

A tibble. Note that if the argument `prevalence` was used, the value reported in the tibble reflects the argument value and not the observed rate of events.

Examples

```
data("two_class_example")

cmat <- conf_mat(two_class_example, truth = "truth", estimate = "predicted")
summary(cmat, wide = TRUE)
summary(cmat, wide = TRUE, prevalence = 0.70)

library(dplyr)
```

```
data("hpc_cv")

# Compute statistics per resample then summarize
hpc_cv %>%
  group_by(Resample) %>%
  do(summary(conf_mat(., truth = "obs", estimate = "pred"))) %>%
  group_by(name) %>%
  summarize(mean = mean(value, na.rm = TRUE),
            sd = sd(value, na.rm = TRUE))
```

`two_class_example` *Two Class Predictions*

Description

Two Class Predictions

Details

These data are a test set form a model built for two classes ("Class1" and "Class2"). There are columns for the true and predicted classes and column for the probabilities for each class.

Value

`two_class_example`
: a data frame

Examples

```
data(two_class_example)
str(two_class_example)
```

Index