

# Package: measures (via r-universe)

October 31, 2024

**Title** Performance Measures for Statistical Learning

**Version** 0.3

**Description** Provides the biggest amount of statistical measures in the whole R world. Includes measures of regression, (multiclass) classification and multilabel classification. The measures come mainly from the 'mlr' package and were programed by several 'mlr' developers.

**Depends** R (>= 3.0), stats

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**Suggests** testthat

**Repository** <https://philipppro.r-universe.dev>

**RemoteUrl** <https://github.com/philipppro/measures>

**RemoteRef** HEAD

**RemoteSha** f52743da652b4008f6898ef6b6ab6100e178729c

## Contents

ACC . . . . .	3
ARSQ . . . . .	3
AUC . . . . .	4
BAC . . . . .	5
BER . . . . .	5
Brier . . . . .	6
BrierScaled . . . . .	7
EXPVAR . . . . .	7
F1 . . . . .	8
FDR . . . . .	8
FN . . . . .	9
FNR . . . . .	10

FP	10
FPR	11
GMEAN	12
GPR	12
KAPPA	13
KendallTau	13
listAllMeasures	14
Logloss	14
LSR	15
MAE	16
MAPE	16
MCC	17
MEDAE	17
MEDSE	18
MMCE	18
MSE	19
MSLE	19
multiclass.AUIP	20
multiclass.AUIU	21
multiclass.AUNP	21
multiclass.AUNU	22
multiclass.Brier	23
MultilabelACC	23
MultilabelF1	24
MultilabelHamloss	24
MultilabelPPV	25
MultilabelSubset01	25
MultilabelTPR	26
NPV	26
PPV	27
QSR	27
RAE	28
RMSE	28
RMSLE	29
RRSE	30
RSQ	30
SAE	31
SpearmanRho	31
SSE	32
SSR	32
TN	33
TNR	33
TP	34
TPR	35
WKAPPA	35

---

ACC	<i>Accuracy</i>
-----	-----------------

---

**Description**

Defined as: `mean(response == truth)`

**Usage**

```
ACC(truth, response)
```

**Arguments**

truth	vector of true values
response	vector of predicted values

**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
response = as.factor(sample(c(1,2,3), n, replace = TRUE))
ACC(truth, response)
```

---

ARSQ	<i>Adjusted coefficient of determination</i>
------	--

---

**Description**

Defined as:  $1 - (1 - \text{rsq}) * (p / (n - p - 1L))$ . Adjusted R-squared is only defined for normal linear regression.

**Usage**

```
ARSQ(truth, response, n, p)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values
n	[numeric] number of observations
p	[numeric] number of predictors

**Examples**

```
n = 20
p = 5
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
ARSQ(truth, response, n, p)
```

---

AUC

*Area under the curve*

---

**Description**

Integral over the graph that results from computing fpr and tpr for many different thresholds.

**Usage**

```
AUC(probabilities, truth, negative, positive)
```

**Arguments**

probabilities	[numeric] vector of predicted probabilities
truth	vector of true values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
AUC(probabilities, truth, negative, positive)
```

---

BAC	<i>Balanced accuracy</i>
-----	--------------------------

---

**Description**

Mean of true positive rate and true negative rate.

**Usage**

```
BAC(truth, response, negative, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
BAC(truth, response, negative, positive)
```

---

BER	<i>Balanced error rate</i>
-----	----------------------------

---

**Description**

Mean of misclassification error rates on all individual classes.

**Usage**

```
BER(truth, response)
```

**Arguments**

truth	vector of true values
response	vector of predicted values

**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
response = as.factor(sample(c(1,2,3), n, replace = TRUE))
BER(truth, response)
```

---

 Brier

*Brier score*


---

**Description**

The Brier score is defined as the quadratic difference between the probability and the value (1,0) for the class. That means we use the numeric representation 1 and 0 for our target classes. It is similar to the mean squared error in regression. `multiclass.brier` is the sum over all one vs. all comparisons and for a binary classification  $2 * \text{brier}$ .

**Usage**

```
Brier(probabilities, truth, negative, positive)
```

**Arguments**

<code>probabilities</code>	[numeric] vector of predicted probabilities
<code>truth</code>	vector of true values
<code>negative</code>	negative class
<code>positive</code>	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
Brier(probabilities, truth, negative, positive)
```

---

BrierScaled	<i>Brier scaled</i>
-------------	---------------------

---

**Description**

Brier score scaled to [0,1], see <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3575184/>.

**Usage**

```
BrierScaled(probabilities, truth, negative, positive)
```

**Arguments**

probabilities	[numeric] vector of predicted probabilities
truth	vector of true values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
BrierScaled(probabilities, truth, negative, positive)
```

---

EXPVAR	<i>Explained variance</i>
--------	---------------------------

---

**Description**

Similar to RSQ (R-squared). Defined as explained\_sum\_of\_squares / total\_sum\_of\_squares.

**Usage**

```
EXPVAR(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
EXPVAR(truth, response)
```

---

F1	<i>F1 measure</i>
----	-------------------

---

**Description**

Defined as:  $2 * tp / (\text{sum}(\text{truth} == \text{positive}) + \text{sum}(\text{response} == \text{positive}))$

**Usage**

```
F1(truth, response, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
F1(truth, response, positive)
```

---

FDR	<i>False discovery rate</i>
-----	-----------------------------

---

**Description**

Defined as:  $fp / (tp + fp)$

**Usage**

```
FDR(truth, response, positive)
```



**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
FDR(truth, response, positive)
```

---

FN	<i>False negatives</i>
----	------------------------

---

**Description**

Sum of misclassified observations in the negative class. Also called misses.

**Usage**

```
FN(truth, response, negative)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
negative = 0
FN(truth, response, negative)
```

---

FNR *False negative rate*

---

**Description**

Percentage of misclassified observations in the negative class.

**Usage**

```
FNR(truth, response, negative, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
FNR(truth, response, negative, positive)
```

---

FP *False positives*

---

**Description**

Sum of misclassified observations in the positive class. Also called false alarms.

**Usage**

```
FP(truth, response, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
FPR(truth, response, positive)
```

---

FPR

*False positive rate*

---

**Description**

Percentage of misclassified observations in the positive class. Also called false alarm rate or fall-out.

**Usage**

```
FPR(truth, response, negative, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
FPR(truth, response, negative, positive)
```

GMEAN

*G-mean*

---

**Description**

Geometric mean of recall and specificity.

**Usage**

```
GMEAN(truth, response, negative, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class
positive	positive class

**References**

He, H. & Garcia, E. A. (2009) \*Learning from Imbalanced Data.\* IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9. pp. 1263-1284.

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
GMEAN(truth, response, negative, positive)
```

---

GPR*Geometric mean of precision and recall.*

---

**Description**

Defined as:  $\sqrt{\text{ppv} * \text{tpr}}$

**Usage**

```
GPR(truth, response, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
GPR(truth, response, positive)
```

---

KAPPA	<i>Cohen's kappa</i>
-------	----------------------

---

**Description**

Defined as:  $1 - (1 - p_0) / (1 - p_e)$ . With:  $p_0$  = 'observed frequency of agreement' and  $p_e$  = 'expected agreement frequency under independence'

**Usage**

```
KAPPA(truth, response)
```

**Arguments**

truth	vector of true values
response	vector of predicted values <code>n = 20 set.seed(122) truth = as.factor(sample(c(1,2,3), n, replace = TRUE)) response = as.factor(sample(c(1,2,3), n, repla KAPPA(truth, response)</code>

---

KendallTau	<i>Kendall's tau</i>
------------	----------------------

---

**Description**

Defined as: Kendall's tau correlation between truth and response. Only looks at the order. See Rosset et al.: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.1398&rep=rep1&type=pdf>.

**Usage**

```
KendallTau(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
 response        [numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
KendallTau(truth, response)
```

---

listAllMeasures	<i>List all measures</i>
-----------------	--------------------------

---

**Description**

Lists all measures that are available in the package with their corresponding task.

**Usage**

```
listAllMeasures()
```

**Value**

Dataframe with all available measures and the corresponding task

**Examples**

```
listAllMeasures()
```

---

Logloss	<i>Logarithmic loss</i>
---------	-------------------------

---

**Description**

Defined as:  $-\text{mean}(\log(p_i))$ , where  $p_i$  is the predicted probability of the true class of observation  $i$ . Inspired by <https://www.kaggle.com/wiki/MultiClassLogLoss>.

**Usage**

```
Logloss(probabilities, truth)
```

**Arguments**

probabilities	[numeric] vector (or matrix with column names of the classes) of predicted probabilities
truth	vector of true values

**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
probabilities = matrix(runif(60), 20, 3)
probabilities = probabilities/rowSums(probabilities)
colnames(probabilities) = c(1,2,3)
Logloss(probabilities, truth)
```

---

 LSR

*Logarithmic Scoring Rule*


---

**Description**

Defined as:  $\text{mean}(\log(p_i))$ , where  $p_i$  is the predicted probability of the true class of observation  $i$ . This scoring rule is the same as the negative logloss, self-information or surprisal. See: Bickel, J. E. (2007). Some comparisons among quadratic, spherical, and logarithmic scoring rules. *Decision Analysis*, 4(2), 49-65.

**Usage**

```
LSR(probabilities, truth)
```

**Arguments**

probabilities	[numeric] vector (or matrix with column names of the classes) of predicted probabilities
truth	vector of true values <code>n = 20 set.seed(122) truth = as.factor(sample(c(1,2,3), n, replace = TRUE)) probabilities = matrix(runif(60), 20, 3) probabilities = probabilities/rowSums(probabilities) colnames(probabilities) = c(1,2,3) LSR(probabilities, truth)</code>

---

MAE	<i>Mean of absolute errors</i>
-----	--------------------------------

---

**Description**

Defined as: `mean(abs(response - truth))`

**Usage**

```
MAE(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
MAE(truth, response)
```

---

MAPE	<i>Mean absolute percentage error</i>
------	---------------------------------------

---

**Description**

Defined as the `abs(truth_i - response_i) / truth_i`. Won't work if any truth value is equal to zero. In this case the output will be NA.

**Usage**

```
MAPE(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
MAPE(truth, response)
```



---

MCC *Matthews correlation coefficient*

---

**Description**

Defined as  $(tp * tn - fp * fn) / \sqrt{(tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)}$ , denominator set to 1 if 0.

**Usage**

```
MCC(truth, response, negative, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
negative = 0
MCC(truth, response, negative, positive)
```

---

MEDAE *Median of absolute errors*

---

**Description**

Defined as: `median(abs(response - truth))`.

**Usage**

```
MEDAE(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
MEDAE(truth, response)
```

---

MEDSE

*Median of squared errors*

---

**Description**

Defined as:  $\text{median}((\text{response} - \text{truth})^2)$ .

**Usage**

```
MEDSE(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
response        [numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
MEDSE(truth, response)
```

---

MMCE

*Mean misclassification error*

---

**Description**

Defined as:  $\text{mean}(\text{response} \neq \text{truth})$

**Usage**

```
MMCE(truth, response)
```

**Arguments**

truth            vector of true values  
response        vector of predicted values

**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
response = as.factor(sample(c(1,2,3), n, replace = TRUE))
MMCE(truth, response)
```

---

MSE

*Mean of squared errors*

---

**Description**

Defined as:  $\text{mean}((\text{response} - \text{truth})^2)$

**Usage**

```
MSE(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
response        [numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
MSE(truth, response)
```

---

MSLE

*Mean squared logarithmic error*

---

**Description**

Defined as:  $\text{mean}((\log(\text{response} + 1, \text{exp}(1)) - \log(\text{truth} + 1, \text{exp}(1)))^2)$ . This is mostly used for count data, note that all predicted and actual target values must be greater or equal '-1' to compute the mean squared logarithmic error.

**Usage**

```
MSLE(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
response        [numeric] vector of predicted values

**Examples**

```
n = 20  
set.seed(123)  
truth = abs(rnorm(n))  
response = abs(rnorm(n))  
MSLE(truth, response)
```

---

multiclass.AU1P            *Weighted average 1 vs. 1 multiclass AUC*

---

**Description**

Computes AUC of  $c(c - 1)$  binary classifiers while considering the a priori distribution of the classes. See Ferri et al.: <https://www.math.ucdavis.edu/~saito/data/roc/ferri-class-perf-metrics.pdf>.

**Usage**

```
multiclass.AU1P(probabilities, truth)
```

**Arguments**

probabilities    [numeric] matrix of predicted probabilities with columnnames of the classes  
truth            vector of true values

**Examples**

```
n = 20  
set.seed(122)  
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))  
probabilities = matrix(runif(60), 20, 3)  
probabilities = probabilities/rowSums(probabilities)  
colnames(probabilities) = c(1,2,3)  
multiclass.AU1P(probabilities, truth)
```

---

multiclass.AU1U	<i>Average 1 vs. 1 multiclass AUC</i>
-----------------	---------------------------------------

---

**Description**

Computes AUC of  $c(c - 1)$  binary classifiers (all possible pairwise combinations) while considering uniform distribution of the classes. See Ferri et al.: <https://www.math.ucdavis.edu/~saito/data/roc/ferri-class-perf-metrics.pdf>.

**Usage**

```
multiclass.AU1U(probabilities, truth)
```

**Arguments**

probabilities [numeric] matrix of predicted probabilities with columnnames of the classes  
truth vector of true values

**Examples**

```
n = 20  
set.seed(122)  
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))  
probabilities = matrix(runif(60), 20, 3)  
probabilities = probabilities/rowSums(probabilities)  
colnames(probabilities) = c(1,2,3)  
multiclass.AU1U(probabilities, truth)
```

---

multiclass.AUNP	<i>Weighted average 1 vs. rest multiclass AUC</i>
-----------------	---

---

**Description**

Computes the AUC treating a  $c$ -dimensional classifier as  $c$  two-dimensional classifiers, taking into account the prior probability of each class. See Ferri et al.: <https://www.math.ucdavis.edu/~saito/data/roc/ferri-class-perf-metrics.pdf>.

**Usage**

```
multiclass.AUNP(probabilities, truth)
```

**Arguments**

probabilities [numeric] matrix of predicted probabilities with columnnames of the classes  
truth vector of true values

## Examples

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
probabilities = matrix(runif(60), 20, 3)
probabilities = probabilities/rowSums(probabilities)
colnames(probabilities) = c(1,2,3)
multiclass.AUNP(probabilities, truth)
```

---

multiclass.AUNU	<i>Average 1 vs. rest multiclass AUC</i>
-----------------	--

---

## Description

Computes the AUC treating a c-dimensional classifier as c two-dimensional classifiers, where classes are assumed to have uniform distribution, in order to have a measure which is independent of class distribution change. See Ferri et al.: <https://www.math.ucdavis.edu/~saito/data/roc/ferri-class-perf-metrics.pdf>.

## Usage

```
multiclass.AUNU(probabilities, truth)
```

## Arguments

`probabilities` [numeric] matrix of predicted probabilities with columnnames of the classes  
`truth` vector of true values

## Examples

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
probabilities = matrix(runif(60), 20, 3)
probabilities = probabilities/rowSums(probabilities)
colnames(probabilities) = c(1,2,3)
multiclass.AUNU(probabilities, truth)
```

---

multiclass.Brier	<i>Multiclass Brier score</i>
------------------	-------------------------------

---

**Description**

Defined as:  $(1/n) \sum_i \sum_j (y_{ij} - p_{ij})^2$ , where  $y_{ij} = 1$  if observation  $i$  has class  $j$  (else 0), and  $p_{ij}$  is the predicted probability of observation  $i$  for class  $j$ . From <http://docs.lib.noaa.gov/rescue/mwr/078/mwr-078-01-0001.pdf>.

**Usage**

```
multiclass.Brier(probabilities, truth)
```

**Arguments**

probabilities	[numeric] matrix of predicted probabilities with columnnames of the classes
truth	vector of true values

**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
probabilities = matrix(runif(60), 20, 3)
probabilities = probabilities/rowSums(probabilities)
colnames(probabilities) = c(1,2,3)
multiclass.Brier(probabilities, truth)
```

---

MultilabelACC	<i>Accuracy (multilabel)</i>
---------------	------------------------------

---

**Description**

Averaged proportion of correctly predicted labels with respect to the total number of labels for each instance, following the definition by Charte and Charte: <https://journal.r-project.org/archive/2015-2/charte-chart.pdf>. Fractions where the denominator becomes 0 are replaced with 1 before computing the average across all instances.

**Usage**

```
MultilabelACC(truth, response)
```

**Arguments**

truth	matrix of true values
response	matrix of predicted values <code>n = 20</code> <code>set.seed(122)</code> <code>truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)</code> <code>response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)</code> <code>MultilabelACC(truth, response)</code>

---

MultilabelF1                      *F1 measure (multilabel)*

---

### Description

Harmonic mean of precision and recall on a per instance basis (Micro-F1), following the definition by Montanes et al.: <http://www.sciencedirect.com/science/article/pii/S0031320313004019>. Fractions where the denominator becomes 0 are replaced with 1 before computing the average across all instances.

### Usage

```
MultilabelF1(truth, response)
```

### Arguments

truth	matrix of true values
response	matrix of predicted values n = 20 set.seed(122) truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) MultilabelF1(truth, response)

---

MultilabelHamloss                      *Hamming loss*

---

### Description

Proportion of labels that are predicted incorrectly, following the definition by Charte and Charte: <https://journal.r-project.org/archive/2015-2/charte-charte.pdf>.

### Usage

```
MultilabelHamloss(truth, response)
```

### Arguments

truth	matrix of true values
response	matrix of predicted values

### Examples

```
n = 20
set.seed(122)
truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)
response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)
MultilabelHamloss(truth, response)
```



---

MultilabelPPV	<i>Positive predictive value (multilabel)</i>
---------------	---

---

**Description**

Also called precision. Averaged ratio of correctly predicted labels for each instance, following the definition by Charte and Charte: <https://journal.r-project.org/archive/2015-2/charte-chart.pdf>. Fractions where the denominator becomes 0 are ignored in the average calculation.

**Usage**

```
MultilabelPPV(truth, response)
```

**Arguments**

truth	matrix of true values
response	matrix of predicted values n = 20 set.seed(122) truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) MultilabelPPV(truth, response)

---

MultilabelSubset01	<i>Subset-0-1 loss</i>
--------------------	------------------------

---

**Description**

Proportion of observations where the complete multilabel set (all 0-1-labels) is predicted incorrectly, following the definition by Charte and Charte: <https://journal.r-project.org/archive/2015-2/charte-chart.pdf>.

**Usage**

```
MultilabelSubset01(truth, response)
```

**Arguments**

truth	matrix of true values
response	matrix of predicted values

**Examples**

```
n = 20
set.seed(122)
truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)
response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3)
MultilabelSubset01(truth, response)
```

---

MultilabelTPR	<i>TPR (multilabel)</i>
---------------	-------------------------

---

**Description**

Also called recall. Averaged proportion of predicted labels which are relevant for each instance, following the definition by Charte and Charte: <https://journal.r-project.org/archive/2015-2/charte-chart.pdf>. Fractions where the denominator becomes 0 are ignored in the average calculation.

**Usage**

```
MultilabelTPR(truth, response)
```

**Arguments**

truth	matrix of true values
response	matrix of predicted values

```
n = 20 set.seed(122) truth = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) response = matrix(sample(c(0,1), 60, replace = TRUE), 20, 3) MultilabelTPR(truth, response)
```

---

NPV	<i>Negative predictive value</i>
-----	----------------------------------

---

**Description**

Defined as:  $tn / (tn + fn)$ .

**Usage**

```
NPV(truth, response, negative)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
negative = 0
NPV(truth, response, negative)
```

---

PPV *Positive predictive value*

---

**Description**

Defined as:  $tp / (tp + fp)$ . Also called precision. If the denominator is 0, PPV is set to be either 1 or 0 depending on whether the highest probability prediction is positive (1) or negative (0).

**Usage**

```
PPV(truth, response, positive, probabilities = NULL)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class
probabilities	[numeric] vector of predicted probabilities

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
PPV(truth, response, positive, probabilities = NULL)
```

---

QSR *Quadratic Scoring Rule*

---

**Description**

Defined as:  $1 - (1/n) \sum_i \sum_j (y_{ij} - p_{ij})^2$ , where  $y_{ij} = 1$  if observation  $i$  has class  $j$  (else 0), and  $p_{ij}$  is the predicted probability of observation  $i$  for class  $j$ . This scoring rule is the same as  $1 - \text{multiclass.brier}$ . See: Bickel, J. E. (2007). Some comparisons among quadratic, spherical, and logarithmic scoring rules. *Decision Analysis*, 4(2), 49-65.

**Usage**

```
QSR(probabilities, truth)
```

**Arguments**

probabilities [numeric] vector (or matrix with column names of the classes) of predicted probabilities

truth vector of true values `n = 20` `set.seed(122)` `truth = as.factor(sample(c(1,2,3), n, replace = TRUE))` `probabilities = matrix(runif(60), 20, 3)` `probabilities = probabilities/rowSums(probabilities)` `colnames(probabilities) = c(1,2,3)` `QSR(probabilities, truth)`

RAE

*Relative absolute error***Description**

Defined as `sum_of_absolute_errors / mean_absolute_deviation`. Undefined for single instances and when every truth value is identical. In this case the output will be NA.

**Usage**

```
RAE(truth, response)
```

**Arguments**

truth [numeric] vector of true values

response [numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
RAE(truth, response)
```

RMSE

*Root mean squared error***Description**

The RMSE is aggregated as `sqrt(mean(rmse.vals.on.test.sets^2))`

**Usage**

```
RMSE(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
response        [numeric] vector of predicted values

**Examples**

```
n = 20  
set.seed(123)  
truth = rnorm(n)  
response = rnorm(n)  
RMSE(truth, response)
```

---

RMSLE

*Root mean squared logarithmic error*

---

**Description**

Definition taken from: <https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError>. This is mostly used for count data, note that all predicted and actual target values must be greater or equal '-1' to compute the root mean squared logarithmic error.

**Usage**

```
RMSLE(truth, response)
```

**Arguments**

truth            [numeric] vector of true values  
response        [numeric] vector of predicted values

**Examples**

```
n = 20  
set.seed(123)  
truth = abs(rnorm(n))  
response = abs(rnorm(n))  
RMSLE(truth, response)
```

---

RRSE	<i>Root relative squared error</i>
------	------------------------------------

---

**Description**

Defined as  $\sqrt{\text{sum\_of\_squared\_errors} / \text{total\_sum\_of\_squares}}$ . Undefined for single instances and when every truth value is identical. In this case the output will be NA.

**Usage**

```
RRSE(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
RRSE(truth, response)
```

---

RSQ	<i>Coefficient of determination</i>
-----	-------------------------------------

---

**Description**

Also called R-squared, which is  $1 - \text{residual\_sum\_of\_squares} / \text{total\_sum\_of\_squares}$ .

**Usage**

```
RSQ(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
RSQ(truth, response)
```

---

SAE	<i>Sum of absolute errors</i>
-----	-------------------------------

---

**Description**

Defined as: `sum(abs(response - truth))`"

**Usage**

```
SAE(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
SAE(truth, response)
```

---

SpearmanRho	<i>Spearman's rho</i>
-------------	-----------------------

---

**Description**

Defined as: Spearman's rho correlation between truth and response. Only looks at the order. See Rosset et al.: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.1398&rep=rep1&type=pdf>.

**Usage**

```
SpearmanRho(truth, response)
```

**Arguments**

truth	[numeric] vector of true values
response	[numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
SpearmanRho(truth, response)
```

---

SSE *Sum of squared errors*

---

**Description**

Defined as:  $\text{sum}((\text{response} - \text{truth})^2)$

**Usage**

SSE(truth, response)

**Arguments**

truth            [numeric] vector of true values  
 response        [numeric] vector of predicted values

**Examples**

```
n = 20
set.seed(123)
truth = rnorm(n)
response = rnorm(n)
SSE(truth, response)
```

---

SSR *Spherical Scoring Rule*

---

**Description**

Defined as:  $\text{mean}(p_i(\sum_j(p_{ij})))$ , where  $p_i$  is the predicted probability of the true class of observation  $i$  and  $p_{ij}$  is the predicted probability of observation  $i$  for class  $j$ . See: Bickel, J. E. (2007). Some comparisons among quadratic, spherical, and logarithmic scoring rules. *Decision Analysis*, 4(2), 49-65.

**Usage**

SSR(probabilities, truth)

**Arguments**

probabilities    [numeric] vector (or matrix with column names of the classes) of predicted probabilities  
 truth            vector of true values



**Examples**

```
n = 20
set.seed(122)
truth = as.factor(sample(c(1,2,3), n, replace = TRUE))
probabilities = matrix(runif(60), 20, 3)
probabilities = probabilities/rowSums(probabilities)
colnames(probabilities) = c(1,2,3)
SSR(probabilities, truth)
```

---

 TN

*True negatives*


---

**Description**

Sum of correctly classified observations in the negative class. Also called correct rejections.

**Usage**

```
TN(truth, response, negative)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
negative = 0
TN(truth, response, negative)
```

---

 TNR

*True negative rate*


---

**Description**

Percentage of correctly classified observations in the negative class. Also called specificity.

**Usage**

```
TNR(truth, response, negative)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
negative	negative class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
negative = 0
TNR(truth, response, negative)
```

---

TP	<i>True positives</i>
----	-----------------------

---

**Description**

Sum of all correctly classified observations in the positive class.

**Usage**

```
TP(truth, response, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
TP(truth, response, positive)
```

---

TPR	<i>True positive rate</i>
-----	---------------------------

---

**Description**

Percentage of correctly classified observations in the positive class. Also called hit rate or recall or sensitivity.

**Usage**

```
TPR(truth, response, positive)
```

**Arguments**

truth	vector of true values
response	vector of predicted values
positive	positive class

**Examples**

```
n = 20
set.seed(125)
truth = as.factor(sample(c(1,0), n, replace = TRUE))
probabilities = runif(n)
response = as.factor(as.numeric(probabilities > 0.5))
positive = 1
TPR(truth, response, positive)
```

---

WKAPPA	<i>Mean quadratic weighted kappa</i>
--------	--------------------------------------

---

**Description**

Defined as:  $1 - \frac{\sum(\text{weights} * \text{conf.mat})}{\sum(\text{weights} * \text{expected.mat})}$ , the weight matrix measures seriousness of disagreement with the squared euclidean metric.

**Usage**

```
WKAPPA(truth, response)
```

**Arguments**

truth	vector of true values
response	vector of predicted values <code>n = 20 set.seed(122) truth = as.factor(sample(c(1,2,3), n, replace = TRUE)) response = as.factor(sample(c(1,2,3), n, repla</code> WKAPPA(truth, response)

# Index

ACC, 3  
ARSQ, 3  
AUC, 4  
  
BAC, 5  
BER, 5  
Brier, 6  
BrierScaled, 7  
  
EXPVAR, 7  
  
F1, 8  
FDR, 8  
FN, 9  
FNR, 10  
FP, 10  
FPR, 11  
  
GMEAN, 12  
GPR, 12  
  
KAPPA, 13  
KendallTau, 13  
  
listAllMeasures, 14  
Logloss, 14  
LSR, 15  
  
MAE, 16  
MAPE, 16  
MCC, 17  
MEDAE, 17  
MEDSE, 18  
MMCE, 18  
MSE, 19  
MSLE, 19  
multiclass.AU1P, 20  
multiclass.AU1U, 21  
multiclass.AUNP, 21  
multiclass.AUNU, 22  
multiclass.Brier, 23  
MultilabelACC, 23  
MultilabelF1, 24  
MultilabelHamloss, 24  
MultilabelPPV, 25  
MultilabelSubset01, 25  
MultilabelTPR, 26  
  
NPV, 26  
  
PPV, 27  
  
QSR, 27  
  
RAE, 28  
RMSE, 28  
RMSLE, 29  
RRSE, 30  
RSQ, 30  
  
SAE, 31  
SpearmanRho, 31  
SSE, 32  
SSR, 32  
  
TN, 33  
TNR, 33  
TP, 34  
TPR, 35  
  
WKAPPA, 35