Package: Metrics (via r-universe)

September 10, 2024

Version 0.1.4

Title Evaluation Metrics for Machine Learning

Description An implementation of evaluation metrics in R that are commonly used in supervised machine learning. It implements metrics for regression, time series, binary classification, classification, and information retrieval problems. It has zero dependencies and a consistent, simple interface for all functions.

Maintainer Michael Frasco <mfrasco6@gmail.com>

Suggests testthat

URL https://github.com/mfrasco/Metrics

BugReports https://github.com/mfrasco/Metrics/issues

License BSD_3_clause + file LICENSE

RoxygenNote 6.1.0

Repository https://mfrasco.r-universe.dev

RemoteUrl https://github.com/mfrasco/metrics

RemoteRef HEAD

RemoteSha ca12765d3e284e907d4c493b617e2b7e8056128d

Contents

`accuracy` *Accuracy*

Description

`accuracy` is defined as the proportion of elements in `actual` that are equal to the corresponding element in `predicted`

Usage

```
accuracy(actual, predicted)
```

Arguments

`actual`	The ground truth vector, where elements of the vector can be any variable type.
`predicted`	The predicted vector, where elements of the vector represent a prediction for the corresponding value in `actual`.

See Also

ce

Examples

```
actual <- c('a', 'a', 'c', 'b', 'c')
predicted <- c('a', 'b', 'c', 'b', 'a')
accuracy(actual, predicted)
```

---

ae *Absolute Error*

Description

ae computes the elementwise absolute difference between two numeric vectors.

Usage

```
ae(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mae mdae mape

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
ae(actual, predicted)
```

ape *Absolute Percent Error*

Description

ape computes the elementwise absolute percent difference between two numeric vectors

Usage

```
ape(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

ape is calculated as (actual - predicted) / abs(actual). This means that the function will return -Inf, Inf, or NaN if actual is zero.

See Also

mape smape

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
ape(actual, predicted)
```

apk *Average Precision at k*

Description

apk computes the average precision at k, in the context of information retrieval problems.

Usage

```
apk(k, actual, predicted)
```

Arguments

k	The number of elements of predicted to consider in the calculation.
actual	The ground truth vector of relevant documents. The vector can contain any numeric or character values, order does not matter, and the vector does not need to be the same length as predicted.
predicted	The predicted vector of retrieved documents. The vector can contain any numeric of character values. However, unlike actual, order does matter, with the most documents deemed most likely to be relevant at the beginning.

Details

apk loops over the first k values of predicted. For each value, if the value is contained within actual and has not been predicted before, we increment the number of sucesses by one and increment our score by the number of successes divided by k. Then, we return our final score divided by the number of relevant documents (i.e. the length of actual).

apk will return NaN if length(actual) equals 0.

See Also

apk f1

Examples

```
actual <- c('a', 'b', 'd')
predicted <- c('b', 'c', 'a', 'e', 'f')
apk(3, actual, predicted)
```

---

auc *Area under the ROC curve (AUC)*

Description

auc computes the area under the receiver-operator characteristic curve (AUC).

Usage

```
auc(actual, predicted)
```

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	A numeric vector of predicted values, where the smallest values correspond to the observations most believed to be in the negative class and the largest values indicate the observations most believed to be in the positive class. Each element represents the prediction for the corresponding element in actual.

Details

auc uses the fact that the area under the ROC curve is equal to the probability that a randomly chosen positive observation has a higher predicted value than a randomly chosen negative value. In order to compute this probability, we can calculate the Mann-Whitney U statistic. This method is very fast, since we do not need to compute the ROC curve first.

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(0.9, 0.8, 0.4, 0.5, 0.3, 0.2)
auc(actual, predicted)
```

---

bias *Bias*

Description

bias computes the average amount by which actual is greater than predicted.

Usage

```
bias(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

If a model is unbiased bias(actual, predicted) should be close to zero. Bias is calculated by taking the average of (actual - predicted).

See Also

percent_bias

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
bias(actual, predicted)
```

ce *Classification Error*

Description

ce is defined as the proportion of elements in actual that are not equal to the corresponding element in predicted.

Usage

```
ce(actual, predicted)
```

Arguments

actual	The ground truth vector, where elements of the vector can be any variable type.
predicted	The predicted vector, where elements of the vector represent a prediction for the corresponding value in actual.

See Also

accuracy

Examples

```
actual <- c('a', 'a', 'c', 'b', 'c')
predicted <- c('a', 'b', 'c', 'b', 'a')
ce(actual, predicted)
```

explained_variation *Explained Variation*

Description

explained_variation computes the percentage of variation in one numeric vector explained by another, also known as the coefficient of determination.

Usage

```
explained_variation(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

`explained_variation` subtracts the relative squared error, `rse(actual, predicted)`, from 1, meaning it can return negative values if the predictions are on average further from the actual values than predictions from a naive model that predicts the mean for every data point.

See Also

`rse`

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
explained_variation(actual, predicted)
```

---

f1 *F1 Score*

Description

`f1` computes the F1 Score in the context of information retrieval problems.

Usage

```
f1(actual, predicted)
```

Arguments

`actual`	The ground truth vector of relevant documents. The vector can contain any numeric or character values, order does not matter, and the vector does not need to be the same length as `predicted`.
`predicted`	The predicted vector of retrieved documents. The vector can contain any numeric or character values, order does not matter, and the vector does not need to be the same length as `actual`.

Details

`f1` is defined as $2 * precision * recall / (precision + recall)$. In the context of information retrieval problems, precision is the proportion of retrieved documents that are relevant to a query and recall is the proportion of relevant documents that are successfully retrieved by a query. If there are zero relevant documents that are retrieved, zero relevant documents, or zero predicted documents, `f1` is defined as `0`.

See Also

`apk` `mapk`

Examples

```
actual <- c('a', 'c', 'd')
predicted <- c('d', 'e')
f1(actual, predicted)
```

`fbeta_score` *F-beta Score*

Description

`fbeta_score` computes a weighted harmonic mean of Precision and Recall. The `beta` parameter controls the weighting.

Usage

```
fbeta_score(actual, predicted, beta = 1)
```

Arguments

`actual`	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
`predicted`	The predicted binary numeric vector containing 1 for the positive class and 0 for the negative class. Each element represents the prediction for the corresponding element in `actual`.
`beta`	A non-negative real number controlling how close the F-beta score is to either Precision or Recall. When `beta` is at the default of 1, the F-beta Score is exactly an equally weighted harmonic mean. The F-beta score will weight toward Precision when `beta` is less than one. The F-beta score will weight toward Recall when `beta` is greater than one.

See Also

`precision` `recall`

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(1, 0, 1, 1, 1, 1)
recall(actual, predicted)
```

ll *Log Loss*

Description

ll computes the elementwise log loss between two numeric vectors.

Usage

```
ll(actual, predicted)
```

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	A numeric vector of predicted values, where the values correspond to the probabilities that each observation in actual belongs to the positive class

See Also

logLoss

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(0.9, 0.8, 0.4, 0.5, 0.3, 0.2)
ll(actual, predicted)
```

logLoss *Mean Log Loss*

Description

logLoss computes the average log loss between two numeric vectors.

Usage

```
logLoss(actual, predicted)
```

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	A numeric vector of predicted values, where the values correspond to the probabilities that each observation in actual belongs to the positive class

See Also

ll

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(0.9, 0.8, 0.4, 0.5, 0.3, 0.2)
logLoss(actual, predicted)
```

---

mae *Mean Absolute Error*

Description

mae computes the average absolute difference between two numeric vectors.

Usage

```
mae(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mdae mape

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
mae(actual, predicted)
```

mape *Mean Absolute Percent Error*

Description

mape computes the average absolute percent difference between two numeric vectors.

Usage

```
mape(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

mape is calculated as the average of (actual - predicted) / abs(actual). This means that the function will return -Inf, Inf, or NaN if actual is zero. Due to the instability at or near zero, smape or mase are often used as alternatives.

See Also

mae smape mase

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
mape(actual, predicted)
```

mapk *Mean Average Precision at k*

Description

mapk computes the mean average precision at k for a set of predictions, in the context of information retrieval problems.

Usage

```
mapk(k, actual, predicted)
```

Arguments

k	The number of elements of predicted to consider in the calculation.
actual	A list of vectors, where each vector represents a ground truth vector of relevant documents. In each vector, the elements can be numeric or character values, and the order of the elements does not matter.
predicted	A list of vectors, where each vector represents the predicted vector of retrieved documents for the corresponding element of actual. In each vector, the order of the elements does matter, with the elements believed most likely to be relevant at the beginning.

Details

mapk evaluates apk for each pair of elements from actual and predicted.

See Also

apk f1

Examples

```
actual <- list(c('a', 'b'), c('a'), c('x', 'y', 'b'))
predicted <- list(c('a', 'c', 'd'), c('x', 'b', 'a', 'b'), c('y'))
mapk(2, actual, predicted)

actual <- list(c(1, 5, 7, 9), c(2, 3), c(2, 5, 6))
predicted <- list(c(5, 6, 7, 8, 9), c(1, 2, 3), c(2, 4, 6, 8))
mapk(3, actual, predicted)
```

---

mase *Mean Absolute Scaled Error*

Description

mase computes the mean absolute scaled error between two numeric vectors. This function is only intended for time series data, where actual and numeric are numeric vectors ordered by time.

Usage

```
mase(actual, predicted, step_size = 1)
```

Arguments

actual	The ground truth numeric vector ordered in time, with most recent observation at the end of the vector.
predicted	The predicted numeric vector ordered in time, where each element of the vector represents a prediction for the corresponding element of actual.

step_size	A positive integer that specifies how many observations to look back in time in order to compute the naive forecast. The default is 1, which means that the naive forecast for the current time period is the actual value of the previous period. However, if actual and predictions were quarterly predictions over many years, letting step_size = 4, would mean that the naive forecast for the current time period would be the actual value from the same quarter last year. In this way, mase can account for seasonality.

See Also

smape mape

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
step_size <- 1
mase(actual, predicted, step_size)
```

---

mdae *Median Absolute Error*

Description

mdae computes the median absolute difference between two numeric vectors.

Usage

```
mdae(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mae mape

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
mdae(actual, predicted)
```

MeanQuadraticWeightedKappa *Mean Quadratic Weighted Kappa*

Description

MeanQuadraticWeightedKappa computes the mean quadratic weighted kappa, which can optionally be weighted

Usage

```
MeanQuadraticWeightedKappa(kappas, weights = rep(1, length(kappas)))
```

Arguments

kappas	A numeric vector of possible kappas.
weights	An optional numeric vector of ratings.

See Also

ScoreQuadraticWeightedKappa

Examples

```
kappas <- c(0.3 ,0.2, 0.2, 0.5, 0.1, 0.2)
weights <- c(1.0, 2.5, 1.0, 1.0, 2.0, 3.0)
MeanQuadraticWeightedKappa(kappas, weights)
```

mse *Mean Squared Error*

Description

mse computes the average squared difference between two numeric vectors.

Usage

```
mse(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

rmse mae

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
mse(actual, predicted)
```

---

msle *Mean Squared Log Error*

Description

msle computes the average of squared log error between two numeric vectors.

Usage

```
msle(actual, predicted)
```

Arguments

actual	The ground truth non-negative vector
predicted	The predicted non-negative vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

msle adds one to both actual and predicted before taking the natural logarithm to avoid taking the natural log of zero. As a result, the function can be used if actual or predicted have zero-valued elements. But this function is not appropriate if either are negative valued.

See Also

rmsle sle

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
msle(actual, predicted)
```

params_binary *Inherit Documentation for Binary Classification Metrics*

Description

This object provides the documentation for the parameters of functions that provide binary classification metrics

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	The predicted binary numeric vector containing 1 for the positive class and 0 for the negative class. Each element represents the prediction for the corresponding element in actual.

params_classification *Inherit Documentation for Classification Metrics*

Description

This object provides the documentation for the parameters of functions that provide classification metrics

Arguments

actual	The ground truth vector, where elements of the vector can be any variable type.
predicted	The predicted vector, where elements of the vector represent a prediction for the corresponding value in actual.

params_regression *Inherit Documentation for Regression Metrics*

Description

This object provides the documentation for the parameters of functions that provide regression metrics

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

percent_bias *Percent Bias*

Description

percent_bias computes the average amount that actual is greater than predicted as a percentage of the absolute value of actual.

Usage

```
percent_bias(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

If a model is unbiased percent_bias(actual, predicted) should be close to zero. Percent Bias is calculated by taking the average of (actual - predicted) / abs(actual) across all observations.

percent_bias will give -Inf, Inf, or NaN, if any elements of actual are 0.

See Also

bias

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
percent_bias(actual, predicted)
```

precision *Precision*

Description

precision computes proportion of observations predicted to be in the positive class (i.e. the element in predicted equals 1) that actually belong to the positive class (i.e. the element in actual equals 1)

Usage

```
precision(actual, predicted)
```

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	The predicted binary numeric vector containing 1 for the positive class and 0 for the negative class. Each element represents the prediction for the corresponding element in actual.

See Also

recall fbeta_score

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(1, 1, 1, 1, 1, 1)
precision(actual, predicted)
```

---

rae *Relative Absolute Error*

Description

rae computes the relative absolute error between two numeric vectors.

Usage

```
rae(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

rae divides sum(ae(actual, predicted)) by sum(ae(actual, mean(actual))), meaning that it provides the absolute error of the predictions relative to a naive model that predicted the mean for every data point.

See Also

rse rrse

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
rrse(actual, predicted)
```

recall *Recall*

Description

recall computes proportion of observations in the positive class (i.e. the element in actual equals 1) that are predicted to be in the positive class (i.e. the element in predicted equals 1)

Usage

```
recall(actual, predicted)
```

Arguments

actual	The ground truth binary numeric vector containing 1 for the positive class and 0 for the negative class.
predicted	The predicted binary numeric vector containing 1 for the positive class and 0 for the negative class. Each element represents the prediction for the corresponding element in actual.

See Also

precision fbeta_score

Examples

```
actual <- c(1, 1, 1, 0, 0, 0)
predicted <- c(1, 0, 1, 1, 1, 1)
recall(actual, predicted)
```

rmse *Root Mean Squared Error*

Description

rmse computes the root mean squared error between two numeric vectors

Usage

```
rmse(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mse

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
rmse(actual, predicted)
```

---

rmsle *Root Mean Squared Log Error*

Description

rmsle computes the root mean squared log error between two numeric vectors.

Usage

```
rmsle(actual, predicted)
```

Arguments

actual	The ground truth non-negative vector
predicted	The predicted non-negative vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

rmsle adds one to both actual and predicted before taking the natural logarithm to avoid taking the natural log of zero. As a result, the function can be used if actual or predicted have zero-valued elements. But this function is not appropriate if either are negative valued.

See Also

msle sle

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
rmsle(actual, predicted)
```

rrse *Root Relative Squared Error*

Description

rrse computes the root relative squared error between two numeric vectors.

Usage

```
rrse(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

rrse takes the square root of sse(actual, predicted) divided by sse(actual, mean(actual)), meaning that it provides the squared error of the predictions relative to a naive model that predicted the mean for every data point.

See Also

rse rae

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
rrse(actual, predicted)
```

rse *Relative Squared Error*

Description

rse computes the relative squared error between two numeric vectors.

Usage

```
rse(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

rse divides sse(actual, predicted) by sse(actual, mean(actual)), meaning that it provides the squared error of the predictions relative to a naive model that predicted the mean for every data point.

See Also

rrse rae

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
rse(actual, predicted)
```

ScoreQuadraticWeightedKappa

Quadratic Weighted Kappa

Description

ScoreQuadraticWeightedKappa computes the quadratic weighted kappa between two vectors of integers

Usage

```
ScoreQuadraticWeightedKappa(rater.a, rater.b, min.rating = min(c(rater.a,
  rater.b)), max.rating = max(c(rater.a, rater.b)))
```

Arguments

rater.a	An integer vector of the first rater's ratings.
rater.b	An integer vector of the second rater's ratings.
min.rating	The minimum possible rating.
max.rating	The maximum possible rating.

See Also

MeanQuadraticWeightedKappa

Examples

```
rater.a <- c(1, 4, 5, 5, 2, 1)
rater.b <- c(2, 2, 4, 5, 3, 3)
ScoreQuadraticWeightedKappa(rater.a, rater.b, 1, 5)
```

---

se *Squared Error*

Description

se computes the elementwise squared difference between two numeric vectors.

Usage

```
se(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mse rmse

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
se(actual, predicted)
```

---

sle *Squared Log Error*

Description

sle computes the elementwise squares of the differences in the logs of two numeric vectors.

Usage

```
sle(actual, predicted)
```

Arguments

actual	The ground truth non-negative vector
predicted	The predicted non-negative vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

sle adds one to both actual and predicted before taking the natural logarithm of each to avoid taking the natural log of zero. As a result, the function can be used if actual or predicted have zero-valued elements. But this function is not appropriate if either are negative valued.

See Also

msle rmsle

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
sle(actual, predicted)
```

---

smape *Symmetric Mean Absolute Percentage Error*

Description

smape computes the symmetric mean absolute percentage error between two numeric vectors.

Usage

```
smape(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

Details

smape is defined as two times the average of abs(actual - predicted) / (abs(actual) + abs(predicted)). Therefore, at the elementwise level, it will provide NaN only if actual and predicted are both zero. It has an upper bound of 2, when either actual or predicted are zero or when actual and predicted are opposite signs.

smape is symmetric in the sense that smape(x, y) = smape(y, x).

See Also

mape mase

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
smape(actual, predicted)
```

---

sse *Sum of Squared Errors*

Description

sse computes the sum of the squared differences between two numeric vectors.

Usage

```
sse(actual, predicted)
```

Arguments

actual	The ground truth numeric vector.
predicted	The predicted numeric vector, where each element in the vector is a prediction for the corresponding element in actual.

See Also

mse

Examples

```
actual <- c(1.1, 1.9, 3.0, 4.4, 5.0, 5.6)
predicted <- c(0.9, 1.8, 2.5, 4.5, 5.0, 6.2)
sse(actual, predicted)
```

Index