Package 'mvrsquared'

September 1, 2026

Type Package

Title Compute the Coefficient of Determination for Vector or Matrix Outcomes

Version 0.1.6

Description Compute the coefficient of determination for outcomes in n-dimensions. May be useful for multidimensional predictions (such as a multinomial model) or calculating goodness of fit from latent variable models such as probabilistic topic models like latent Dirichlet allocation or deterministic topic models like latent semantic analysis. Based on Jones (2019) <doi:10.48550/arXiv.1911.11061>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/TommyJones/mvrsquared

BugReports https://github.com/TommyJones/mvrsquared/issues

Depends R (>= 3.0.2)

Imports Matrix, methods, Rcpp (>= 1.0.2)

Suggests dplyr, furrr, knitr, MASS, nnet, parallel, rmarkdown, stats, stringr, testthat, textmineR, tidytext, spelling

LinkingTo Rcpp, RcppArmadillo, RcppThread (>= 2.1.3)

VignetteBuilder knitr

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Tommy Jones [aut, cre] (ORCID: <https://orcid.org/0000-0001-6457-2452>), Thomas Nagler [ctb] (ORCID: <https://orcid.org/0000-0003-1855-0046>)

Maintainer Tommy Jones <jones.thos.w@gmail.com>

Repository CRAN

Date/Publication 2026-08-31 22:20:08 UTC

Contents

calc_rsquared 2

Index **4**

`calc_rsquared` *Calculate R-Squared.*

Description

Calculate R-Squared for univariate or multivariate outcomes.

Usage

```
calc_rsquared(y, yhat, ybar = NULL, return_ss_only = FALSE, threads = 1)
```

Arguments

`y`	The true outcome. This must be a numeric vector, numeric matrix, or coercible to a sparse matrix of class `dgCMatrix`. See 'Details' below for more information.
`yhat`	The predicted outcome or a list of two matrices whose dot product makes the predicted outcome. See 'Details' below for more information.
`ybar`	Numeric scalar or vector; the mean of `y`. Useful for parallel computation in batches.
`return_ss_only`	Logical. Do you want to forego calculating R-squared and only return the sums of squares?
`threads`	Integer number of threads for parallelism; defaults to 1.

Details

There is some flexibility in what you can pass as `y` and `yhat`. In general, `y` can be a numeric vector, numeric matrix, a sparse matrix of class `dgCMatrix` from the `Matrix` package, or any object that can be coerced into a `dgCMatrix`.

`yhat` can be a numeric vector, numeric matrix, or a list of two matrices whose dot product has the same dimensionality as `y`. If `yhat` is a list of two matrices you may optionally name them `x` and `w` indicating the order of multiplication (`x` left multiplies `w`). If unnamed or ambiguously named, then it is assumed that `yhat[[1]]` left multiplies `yhat[[2]]`.

Value

If `return_ss_only = FALSE`, `calc_rsqured` returns a numeric scalar R-squared. If `return_ss_only = TRUE`, `calc_rsqured` returns a vector; the first element is the error sum of squares (SSE) and the second element is the total sum of squares (SST). R-squared may then be calculated as `1 - SSE / SST`.

Note

Results do not depend on `threads`. Each observation's contribution is written to its own slot and the totals are summed sequentially afterwards, so the answer is the same however the work is scheduled. Earlier versions carried a warning that parallelism could introduce imprecision on some Linux systems; that is no longer believed to be the case, and thread counts from 1 to 12 were verified to give bit-identical results as of version 0.1.6.

Setting `return_ss_only` to `TRUE` is useful for parallel or distributed computing for large data sets, particularly when `y` is a large matrix. However if you do parallel execution you MUST pre-calculate 'ybar' and pass it to the function. If you do not, SST will be calculated based on means of each batch independently. The resulting r-squared will be incorrect.

See example below for parallel computation with `future_map` from the `furr` package.

Examples

```
# standard r-squared with y and yhat as vectors
f <- stats::lm(mpg ~ cyl + disp + hp + wt, data = datasets::mtcars)

y <- f$model$mpg

yhat <- f$fitted.values

calc_rsquared(y = y, yhat = yhat)

# standard r-squared with y as a matrix and yhat containing 'x' and linear coefficients
s <- summary(f)

x <- cbind(1, as.matrix(f$model[, -1]))

w <- matrix(s$coefficients[, 1], ncol = 1)

calc_rsquared(y = matrix(y, ncol = 1), yhat = list(x, w))

# multivariate r-squared with y and yhat as matrices
calc_rsquared(y = cbind(y, y), yhat = cbind(yhat, yhat))

# multivariate r-squared with yhat as a linear reconstruction of two matrices
calc_rsquared(y = cbind(y, y), yhat = list(x, cbind(w,w)))
```

Index

calc_rsquared, 2

future_map, *3*

Matrix, *2*