

# Package: RPttests (via r-universe)

September 8, 2024

**Type** Package

**Title** Goodness of Fit Tests for High-Dimensional Linear Regression Models

**Version** 0.1.5

**Description** Performs goodness of fits tests for both high and low-dimensional linear models. It can test for a variety of model misspecifications including nonlinearity and heteroscedasticity. In addition one can test the significance of potentially large groups of variables, and also produce p-values for the significance of individual variables in high-dimensional linear regression.

**License** GPL (>= 2)

**Imports** glmnet, parallel, randomForest, Rcpp, stats

**LinkingTo** Rcpp

**URL** <https://rss.onlinelibrary.wiley.com/doi/10.1111/rssb.12234>

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Author** Rajen Shah [aut, cre], Peter Buhlmann [aut]

**Maintainer** Rajen Shah <r.shah@statslab.cam.ac.uk>

**Encoding** UTF-8

**Date/Publication** 2021-07-10 20:10:02 UTC

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

**RemoteUrl** <https://github.com/cran/RPttests>

**RemoteRef** HEAD

**RemoteSha** 6296adc362435eb9cc6045ece64094cc2d5360e4

## Contents

pval . . . . .	2
RPtest . . . . .	3

RPtest_single . . . . .	5
sparse_proj . . . . .	6
sqrt_lasso . . . . .	7

<b>Index</b>	<b>10</b>
--------------	-----------

---

pval *Compute p-values for RPtest output*

---

## Description

Produces p-values given a list of simulated test statistics and the true test statistic (which may be a vector if it is the output of multiple RP functions).

## Usage

```
pval(test, test_sim)
```

## Arguments

test	A numeric vector of test statistics.
test_sim	A list of test statistics, each component of which is a numeric vector.

## Details

In the case where the individual test statistics are vectors, the first component of test is compared against the first components of test\_sim[[1]], test\_sim[[2]] etc. and the results of these multiple comparisons are combined into a single p-value (see the reference). When the lengths of the test statistics differ, the final components are first discarded to make all the test statistics have equal length.

## Value

A single p-value.

## References

Shah, R. D., Buhlmann, P. (2017) *Goodness of fit tests for high-dimensional linear models* <https://rss.onlinelibrary.wiley.com/doi/10.1111/rssb.12234>

## See Also

[RPtest](#) the output of which this would typically be applied to.

**Description**

Can test for the significance of (potentially large) groups of predictors and the presence of nonlinearity or heteroscedasticity in the context of both low and high-dimensional linear models. Outputs a p-value. Also allows for the calibration of arbitrary goodness of fit tests via specification of RPfunction.

**Usage**

```
RPtest(
  x,
  y,
  resid_type = c("Lasso", "OLS"),
  test = c("nonlin", "group", "hetero"),
  x_alt,
  RPfunction = NULL,
  B = 49L,
  rand_gen = rnorm,
  noise_matrix = NULL,
  mc.cores = 1L,
  nfolds = 5L,
  nperms = 2L,
  beta_est = NULL,
  resid_only = FALSE,
  output_all = FALSE,
  verbose = FALSE
)
```

**Arguments**

x	Input matrix with nobs rows, each an observation vector.
y	Response vector.
resid_type	Type of residuals used for the test (see details below). Use Lasso when the null model is high-dimensional; otherwise use OLS.
test	Type of departure from the linear model to test for (see details below). Ignored if RPfunction is given.
x_alt	If test is group, this gives the set of variables whose significance we wish to ascertain, after controlling for those in x. If RPfunction is given, it is the input matrix passed to the function RPfunction.
RPfunction	A residual prediction (RP) function that must permit calling as RPfunction(x_alt, resid) where resid is a numeric vector with nobs components. The output must be either a single number or a numeric vector (in the latter case RPfunction would encode a number of RP functions).

B	The number of bootstrap samples to use - note the p-value produced will always be at least 1/B.
rand_gen	A function to generate the simulated errors up to an unknown scale factor. It must permit calling as <code>rand_gen(nobs*B)</code> . Determines the form of errors in the null model. The default <code>rnorm</code> equates to a null of a (sparse) Gaussian linear model. Setting <code>rand_gen=NULL</code> resamples residuals to generate simulated errors and approximates a null of i.i.d. errors with unknown distribution.
noise_matrix	An optional matrix whose columns are the simulated errors to use. Note that B and <code>rand_gen</code> will be ignored if this is supplied.
mc.cores	The number of cores to use. Will always be 1 in Windows.
nfolds	Number of folds to use when performing cross-validation to obtain <code>beta_est</code> , the initial estimate of the vector of regression coefficients, via Lasso estimation.
nperms	Number of permutations of the data for which <code>nfolds</code> cross-validation is to be performed. Thus in total prediction errors on <code>nfolds*nperms</code> folds are averaged over.
beta_est	An optional user-supplied estimate.
resid_only	If TRUE only outputs the residuals without applying an RP function.
output_all	In addition to the p-value, gives further output (see Value below).
verbose	Whether to print addition information.

### Details

The function works by first computing residuals from a regression of  $y$  on  $x$ . Next  $B$  sets of errors generated through `rand_gen` are added to a signal derived from `beta_est` and artificial residuals are computed. The option `resid_only=TRUE` then outputs these residuals along with the original residuals, scaled to have  $l_2$ -norm squared equal to `nobs`. The residuals in question are OLS residuals when `resid_type=OLS` (case a - for use when the null hypothesis is low-dimensional so the number of columns of  $x$  is smaller than `nobs-1`), and square-root / scaled Lasso residuals otherwise (case b). The options for `test` then apply different functions to the residuals as described below.

`nonlin` In case (a), the test statistic is the RSS (residual sum of squares) of a `randomForest` fit from regressing the residuals on to  $x$ ; case (b) is similar but the OOB error is used and the regression is carried out on the equicorrelation set rather than all of  $x$ .

`group`  $x_{alt}$  is first residualised with respect to  $x$  by (a) OLS or (b) `sparse_proj`. Then the RSS from Lasso fits from regressions of the residuals on to  $x_{alt}$  are used.

`hetero` Uses the RSS from Lasso fits from regressions of the squared residuals to the equicorrelation set (b) or all of  $x$  (a).

### Value

When `resid_only=FALSE` and `output_all=FALSE`, the output is a single p-value. Otherwise, a list with some of the following components is returned (`resid_only=FALSE` causes the last two components to be omitted):

p-value p-value

beta\_est estimated vector of regression coefficients `beta_est`

sigma\_est set to 1 when resid\_type=OLS; otherwise the normalised root-RSS derived from beta\_est used in generated the simulated errors

resid scaled residuals

resid\_sim simulated scaled residuals

test the test statistic(s) - may be a vector if multiple RP functions are being used such as when test=group

test\_sim a list of simulated test statistics

## References

Shah, R. D., Buhlmann, P. (2017) *Goodness-of-fit tests for high dimensional linear models* <https://rss.onlinelibrary.wiley.com/doi/10.1111/rssb.12234>

## See Also

[RPtest\\_single](#) and [sqrt\\_lasso](#)

## Examples

```
# Testing for nonlinearity
set.seed(1)
x <- scale(matrix(runif(100*200), 100, 200))
y <- x[, 1] + x[, 1]^4 + rnorm(nrow(x))
out <- RPtest(x, y, test="nonlin", B=9L, nperms=2, resid_type = "Lasso")

# Testing significance of a group
y <- x[, 1:5] %*% rep(1, 5) + x[, 151] + rnorm(nrow(x))
(out <- RPtest(x[, 1:150], y, test="group", x_alt = x[, 151:200], B=9L, nperms=1))

# Testing for heteroscedasticity
x <- scale(matrix(runif(250*100), 250, 100))
hetero_sig <- x[, 1] + x[, 2]
var_vec <- hetero_sig - min(hetero_sig) + 0.01
var_vec <- var_vec / mean(var_vec)
sd_vec <- sqrt(var_vec)
y <- x[, 1:5] %*% rep(1, 5) + sd_vec*rnorm(nrow(x))
(out <- RPtest(x, y, test="hetero", B=9L, nperms=1))
```

---

RPtest\_single

*Test significance of single predictors*

---

## Description

Compute p-values for the significance of each variable in x.

## Usage

```
RPtest_single(x, y, x_alt, B = 100L, rand_gen = rnorm, mc.cores = 1L)
```

**Arguments**

x	Input matrix with nobs rows, each an observation vector.
y	Response variable; should be a numeric vector.
x_alt	Optional: a matrix with jth column the sparse projection of the jth column of x on all its other columns i.e. the output of <code>sparse_proj</code> . If not supplied this is computed by the function.
B	Number of bootstrap samples. If set to 0, the asymptotic distribution is used for calibration.
rand_gen	A function to generate the simulated errors up to an unknown scale factor. It must permit calling as <code>rand_gen(nobs*B)</code> . Determines the form of errors in each of the null models, though the results are broadly insensitive to this choice. The default <code>rnorm</code> equates to null hypotheses of (sparse) Gaussian linear models. Setting <code>rand_gen=NULL</code> resamples residuals to generate simulated errors and approximates nulls of i.i.d. errors with unknown distributions.
mc.cores	Number of cores to use.

**Value**

A vector of p-values for each variable.

**References**

Shah, R. D., Buhlmann, P. (2017) *Goodness of fit tests for high-dimensional linear models* <https://rss.onlinelibrary.wiley.com/doi/10.1111/rssb.12234>

**See Also**

`RPtest` and `sparse_proj`

**Examples**

```
x <- scale(matrix(rnorm(50*100), 50, 100))
x <- scale(x)
y <- as.numeric(x[, 1:5] %%% rep(1, 5) + rnorm(nrow(x)))
out <- RPtest_single(x=x, y=y, B=25)
```

---

sparse\_proj

*Sparse projections using the square-root Lasso*

---

**Description**

Regresses each column of `x` against all others in turn, using the square-root Lasso, and outputs residuals from the regressions. Thus it outputs a form of sparse projection of each column on all others. Alternatively, given two matrices `x_null` and `x_alt`, it regresses each column of `x_null` on `x_alt` in a similar fashion.

**Usage**

```
sparse_proj(x, x_null, x_alt, mc.cores = 1L, rescale = FALSE, ...)
```

**Arguments**

<code>x</code>	Matrix with each row an observation vector. Need not be supplied if <code>x_alt</code> and <code>x_null</code> are given.
<code>x_null</code>	Matrix whose columns are to be regressed on to <code>x_alt</code> .
<code>x_alt</code>	Matrix which the columns of <code>x_null</code> are regressed on to. Must be specified if <code>x_null</code> is given.
<code>mc.cores</code>	The number of cores to use. Will always be 1 in Windows.
<code>rescale</code>	Should the columns of the output be rescaled to have $l_2$ -norm the square-root of the number of observations? Default is FALSE.
<code>...</code>	Additional arguments to be passed to <code>sqrt_lasso</code> .

**Value**

A matrix where each column gives the residuals.

**References**

- A. Belloni, V. Chernozhukov, and L. Wang. (2011) *Square-root lasso: pivotal recovery of sparse signals via conic programming*. *Biometrika*, 98(4):791-806.
- T. Sun and C.-H. Zhang. (2012) *Scaled sparse linear regression*. *Biometrika*, 99(4):879-898.
- T. Sun and C.-H. Zhang. (2013) *Sparse matrix inversion with scaled lasso*. *The Journal of Machine Learning Research*, 14(1):3385-3418.

**See Also**

[sqrt\\_lasso](#) and [RPtest\\_single](#).

**Examples**

```
x <- matrix(rnorm(50*100), 50, 100)
out <- sparse_proj(x)
```

---

sqrt\_lasso

*Square-root Lasso regression*

---

**Description**

Fits a linear model to potentially high-dimensional data using the square-root Lasso, also known as the scaled Lasso. The Lasso path is computed using the **glmnet** package.

**Usage**

```
sqrt_lasso(x, y, lam0 = NULL, exclude = integer(0), output_all = FALSE, ...)
```

**Arguments**

x	Input matrix of dimension nobs by nvars; each row is an observation vector.
y	Response variable; should be a numeric vector.
lam0	Tuning parameter for the square-root / scaled Lasso. If left blank (recommended) this is chosen using the method of Sun & Zhang (2013) implemented in the <b>scalreg</b> package.
exclude	Indices of variables to be excluded from the model; default is none.
output_all	In addition to the vector of coefficients, if TRUE, also outputs the intercept, an estimate of the noise standard deviation, and the output of <b>glmnet</b> .
...	Additional arguments to be passed to <b>glmnet</b> .

**Details**

First the Lasso path is computed using **glmnet** from **glmnet**. Next the particular point on the path corresponding to the square-root Lasso solution is found. As the path is only computed on a grid of points, the square-root Lasso solution is approximate.

**Value**

Either an estimated vector of regression coefficients with nvars components or, if `output_all` is true, a list with components

`beta` the vector of regression coefficients

`a0` an intercept term

`sigma_hat` an estimate of the noise standard deviation; this is calculated as square-root of the average residual sums of squares

`glmnet_obj` the fitted **glmnet** object, an S3 class "glmnet"

`lamda_index` the index of the lambda vector in the **glmnet** object corresponding to the square-root Lasso solution

**References**

A. Belloni, V. Chernozhukov, and L. Wang. (2011) *Square-root lasso: pivotal recovery of sparse signals via conic programming*. *Biometrika*, 98(4):791-806.

T. Sun and C.-H. Zhang. (2012) *Scaled sparse linear regression*. *Biometrika*, 99(4):879-898.

T. Sun and C.-H. Zhang. (2013) *Sparse matrix inversion with scaled lasso*. *The Journal of Machine Learning Research*, 14(1):3385-3418.

**See Also**

[glmnet](#) and [scalreg](#).



**Examples**

```
x <- matrix(rnorm(100*250), 100, 250)
y <- x[, 1] + x[, 2] + rnorm(100)
out <- sqrt_lasso(x, y)
```

# Index

glmnet, 8

pval, 2

randomForest, 4

RPtest, 2, 3, 6

RPtest\_single, 5, 5, 7

scalreg, 8

sparse\_proj, 4, 6, 6

sqrt\_lasso, 5, 7, 7