

# Package ‘RRI’

December 19, 2019

**Type** Package

**Title** Residual Randomization Inference for Regression Models

**Version** 1.1

**Author** Panos Toulis

**Maintainer** Panos Toulis <panos.toulis@chicagobooth.edu>

**Description** Testing and inference for regression models using residual randomization methods. The basis of inference is an invariance assumption on the regression errors, e.g., clustered errors, or doubly-clustered errors.

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** Rcpp (>= 1.0.1)

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.0.2

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-12-19 06:40:03 UTC

## R topics documented:

check_model . . . . .	2
example_clustering . . . . .	2
example_model . . . . .	3
fastLm . . . . .	3
get_clustered_eps . . . . .	4
OLS_c . . . . .	5
one_sided_test . . . . .	5
out_pval . . . . .	6
restricted_OLS_c . . . . .	6
rrinf . . . . .	7
rrinfBase . . . . .	8

rrinf_clust . . . . .	9
rrtest . . . . .	11
rrtest_clust . . . . .	12
r_test_c . . . . .	14
two_sided_test . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

check_model	<i>Checks whether the input model is valid.</i>
-------------	---

---

### Description

Checks whether the input model is valid.

### Usage

```
check_model(model)
```

### Arguments

model	A model object. See <a href="#">example_model</a> for details.
-------	--

---

example_clustering	<i>An example clustering object. A clustering is a List that splits indexes 1..#num_datapoints to clusters. Each List element corresponds to one cluster. The clustering is not necessarily a partition but it usually is.</i>
--------------------	--

---

### Description

An example clustering object. A clustering is a List that splits indexes 1..#num\_datapoints to clusters. Each List element corresponds to one cluster. The clustering is not necessarily a partition but it usually is.

### Usage

```
example_clustering()
```

### Value

A List for the clustering of indexes 1..#num\_datapoints.

---

example_model	<i>Example regression model and H0.</i>
---------------	---

---

**Description**

Example regression model and H0.

**Usage**

```
example_model(n = 100)
```

**Arguments**

n                      Number of datapoints.

**Value**

List of (y, X, lam, lam0) that corresponds to regression model and null hypothesis:

- y = n-length vector of outcomes
- X = n x p covariate matrix;
- lam = p-vector of coefficients
- lam0 = real number.

The null we are testing through this specification is

H0:  $\text{lam}' \beta = \text{lam}[1] * \beta[1] + \dots + \text{lam}[p] * \beta[p] = \text{lam0}$ ,

where beta are the model parameters in the regression,  $y = X \beta + e$ . By default this example sets  $p = 2$ -dim model,  $\text{lam} = (0, 1)$  and  $\text{lam0} = 0$ . In this specification, H0:  $\beta[2] = 0$ .

**Examples**

```
model = example_model()
lm(model$y ~ model$X + 0)
```

---

fastLm	<i>Fast least squares</i>
--------	---------------------------

---

**Description**

This functions fits the regression  $y \sim X$  using Armadillo solve function.

**Usage**

```
fastLm(y, X)
```

**Arguments**

y	Vector of outcomes.
x	Matrix of covariates (first column should be 1's)

**Value**

List of regression output with elements coef, stderr.

---

get_clustered_eps	<i>Calculate residuals restricted under H0</i>
-------------------	--

---

**Description**

Given regression model and clustering, this function calculates the OLS residuals under the linear null hypothesis, and assigns them to the specified clusters.

**Usage**

```
get_clustered_eps(model, clustering)
```

**Arguments**

model	A regression model. See <a href="#">example_model</a> for details.
clustering	A List that specifies a clustering of indexes 1...n (#datapoints). See <a href="#">example_clustering</a> for details.

**Value**

A List of the restricted residuals clustered according to clustering.

**Examples**

```
m = example_model(n=100)
cl = list(1:50, 51:100)
er = get_clustered_eps(m, cl)
stopifnot(length(er) == length(cl))
stopifnot(length(er[[1]]) == 50)
```

---

OLS_c	<i>Fast least squares</i>
-------	---------------------------

---

**Description**

Fast OLS as in [fastLm](#) but returns only the fitted coefficients.

**Usage**

```
OLS_c(y, X)
```

**Arguments**

y	Vector of outcomes.
X	Matrix of covariates (first column should be 1's)

**Value**

Vector of coefficients.

---

one_sided_test	<i>One-sided testing</i>
----------------	--------------------------

---

**Description**

Decides to reject or not based on observed test statistic value `tobs` and randomization values `tvals`.

**Usage**

```
one_sided_test(tobs, tvals, alpha, tol = 1e-14)
```

**Arguments**

tobs	The observed value of the test statistic (scalar).
tvals	Vector of randomization values of the test statistic (to compare with tobs).
alpha	Desired level of the test (between 0 to 1).
tol	Used to check whether tobs is equal to the 1-alpha quantile of tvals.

**Details**

The test may randomize to achieve the specified level `alpha` when there are very few randomization values.

**Value**

Test decision (binary).

**See Also**

Testing Statistical Hypotheses (Ch. 15, Lehman and Romano, 2006)

---

out_pval	<i>Calculates p-value or test decision</i>
----------	--

---

**Description**

Depending on `ret_pval` this function returns either a p-value for the test or the binary decision.

**Usage**

```
out_pval(rtest_out, ret_pval, alpha)
```

**Arguments**

<code>rtest_out</code>	A List with elements <code>tobs</code> , <code>tvals</code> (see <a href="#">one_sided_test</a> for details.)
<code>ret_pval</code>	A Boolean indicating whether to return a p-value (TRUE) or not.
<code>alpha</code>	Desired test level (from 0 to 1).

**Details**

Returns 1 if the test rejects, 0 otherwise.

**Value**

Binary decision if `ret_pval` is TRUE, or the p-value otherwise.

---

restricted_OLS_c	<i>Fast least squares with linear constraint</i>
------------------	--

---

**Description**

This functions fits the regression  $y \sim X$  under a linear constraint on the model parameters. The constraint is  $Q * \beta = c$  where  $\beta$  are the regression model parameters, and  $Q, c$  are inputs.

**Usage**

```
restricted_OLS_c(y, X, bhat, Q, c)
```

**Arguments**

y	Vector of outcomes.
X	Matrix of covariates (first column should be 1's)
bhat	Unconstrained OLS-fitted coefficients.
Q	Matrix of linear constraints (k x p).
c	Vector of constraint values (k x 1).

**Value**

Vector of fitted OLS coefficients under linear constraint.

**See Also**

Advanced Econometrics (Section 1.4, Takeshi Amemiya, 1985)

---

rrinf	<i>Generic residual randomization confidence intervals</i>
-------	--

---

**Description**

This function is a wrapper over [rrtest](#) and gives confidence intervals for all parameters.

**Usage**

```
rrinf(
  y,
  X,
  g_invar,
  cover = 0.95,
  num_R = 999,
  control = list(num_se = 6, num_breaks = 60)
)
```

**Arguments**

y	Vector of outcomes (length n)
X	Covariate matrix (n x p). First column should be ones to include intercept.
g_invar	Function that transforms residuals. Accepts n-vector and returns n-vector.
cover	Number from [0, 1] that denotes the confidence interval coverage (e.g., 0.95 denotes 95%)
num_R	Number of test statistic values to calculate in the randomization test (similar to no. of bootstrap samples).
control	A List that controls the scope of the test inversion.

## Details

This function has similar functionality as standard `confint`. It generates confidence intervals by testing plausible values for each parameter. The plausible values are generated as follows. For some parameter  $\beta_i$  we test successively

$$H_0: \beta_i = \hat{\beta}_i - \text{num\_se} * \text{se}_i$$

...up to...

$$H_0: \beta_i = \hat{\beta}_i + \text{num\_se} * \text{se}_i$$

broken in `num_breaks` intervals. Here,  $\hat{\beta}_i$  is the OLS estimate of  $\beta_i$  and  $\text{se}_i$  is the standard error. We then report the minimum and maximum values in this search space which we cannot reject at level  $\alpha$ . This forms the desired confidence interval.

## Value

Matrix that includes the confidence interval endpoints, and the interval midpoint estimate.

## Note

If the confidence interval appears to be a point or is empty, then this means that the nulls we consider are implausible. We can try to improve the search through `control.tinv`. For example, we can both increase `num_se` to increase the width of search, and increase `num_breaks` to make the search space finer.

## See Also

Life after bootstrap: residual randomization inference in regression models (Toulis, 2019)

<https://www.ptoulis.com/residual-randomization>

## Examples

```
set.seed(123)
X = cbind(rep(1, 100), runif(100))
beta = c(-1, 1)
y = X %*% beta + rnorm(100)
g_invar = function(e) sample(e) # Assume exchangeable errors.
M = rrinf(y, X, g_invar, control=list(num_se=4, num_breaks=20))
M # Intervals cover true values
```

---

rrinfBase

*Generic residual randomization inference This function provides the basis for all other rrinf\* functions.*

---

## Description

Generic residual randomization inference This function provides the basis for all other rrinf\* functions.



**Usage**

```
rrinfBase(y, X, g_or_clust, cover, num_R, control.tinv)
```

**Arguments**

<code>y</code>	Vector of outcomes (length $n$ )
<code>X</code>	Covariate matrix ( $n \times p$ ). First column should be ones to include intercept.
<code>g_or_clust</code>	Either <code>clustering</code> or an invariance function that transforms residuals.
<code>cover</code>	Number from $[0, 1]$ that denotes the confidence interval coverage (e.g., 0.95 denotes 95%)
<code>num_R</code>	Number of test statistic values to calculate in the randomization test (similar to no. of bootstrap samples).
<code>control.tinv</code>	A List that determines the test inversion.

**Details**

This function has similar functionality as standard `confint`. It does so by testing plausible values for each parameter. The plausible values can be controlled as follows. For some parameter  $\beta_i$  we will test successively

$H_0: \beta_i = \text{hat\_beta}_i - \text{num\_se} * \text{se}_i$

...up to...

$H_0: \beta_i = \text{hat\_beta}_i + \text{num\_se} * \text{se}_i$

broken in `num_breaks` intervals. Here, `hat_beta_i` is the OLS estimate of  $\beta_i$  and `se_i` is the standard error.

The `g_or_clust` object should either be (i) a g-invariance function  $R^n \rightarrow R^n$ ; or (ii) a list(`type`, `cl`) where `type`=c("perm", "sign", "double") and `cl`=`clustering` (see [example\\_clustering](#) for details).

<https://www.ptoullis.com/residual-randomization>

**Value**

Matrix that includes the confidence interval endpoints, and the interval midpoint estimate.

---

```
rrinf_clust
```

*Residual randomization inference based on cluster invariances*

---

**Description**

This function is a wrapper over `rrtest_clust` and gives confidence intervals for all parameters assuming a particular cluster invariance on the errors.

**Usage**

```
rrinf_clust(
  y,
  X,
  type,
  clustering = NULL,
  cover = 0.95,
  num_R = 999,
  control = list(num_se = 6, num_breaks = 60)
)
```

**Arguments**

y	Vector of outcomes (length n)
X	Covariate matrix (n x p). First column should be ones to include intercept.
type	A string, either "perm", "sign" or "double".
clustering	A List that specifies a clustering of datapoint indexes 1, ..., n. See <a href="#">example_clustering</a> for details.
cover	Number from [0, 1] that denotes the confidence interval coverage (e.g., 0.95 denotes 95%)
num_R	Number of test statistic values to calculate in the randomization test (similar to no. of bootstrap samples).
control	A List that controls the scope of the test inversion.

**Details**

This function has similar functionality as standard [confint](#). It generates confidence intervals by testing plausible values for each parameter. The plausible values are generated as follows. For some parameter  $\beta_i$  we test successively

$H_0: \beta_i = \hat{\beta}_i - \text{num\_se} * \text{se}_i$

...up to...

$H_0: \beta_i = \hat{\beta}_i + \text{num\_se} * \text{se}_i$

broken in `num_breaks` intervals. Here,  $\hat{\beta}_i$  is the OLS estimate of  $\beta_i$  and  $\text{se}_i$  is the standard error. We then report the minimum and maximum values in this search space which we cannot reject at level  $\alpha$ . This forms the desired confidence interval.

**Value**

Matrix that includes the OLS estimate, and confidence interval endpoints.

**Note**

If the confidence interval appears to be a point or is empty, then this means that the nulls we consider are implausible. We can try to improve the search through `control.tinv`. For example, we can both increase `num_se` to increase the width of search, and increase `num_breaks` to make the search space finer.

See [rrtest\\_clust](#) for a description of `type` and `clustering`.

**See Also**

Life after bootstrap: residual randomization inference in regression models (Toulis, 2019)  
<https://www.ptoulis.com/residual-randomization>

**Examples**

```
# Heterogeneous example
set.seed(123)
n = 200
X = cbind(rep(1, n), 1:n/n)
beta = c(-1, 0.2)
ind = c(rep(0, 0.9*n), rep(1, .1*n)) # cluster indicator
y = X %*% beta + rnorm(n, sd= (1-ind) * 0.1 + ind * 5) # heteroskedastic
confint(lm(y ~ X + 0)) # normal OLS CI is imprecise

cl = list(which(ind==0), which(ind==1)) # define the clustering
rrinf_clust(y, X, "perm", cl) # improved CI through clustered errors
```

rrtest

*Generic residual randomization test***Description**

This function tests the specified linear hypothesis in model assuming the errors are distributionally invariant with respect to stochastic function `g_invar`.

**Usage**

```
rrtest(model, g_invar, num_R = 999, alpha = 0.05, val_type = "decision")
```

**Arguments**

<code>model</code>	Regression model and hypothesis. See <a href="#">example_model</a> for details.
<code>g_invar</code>	Stochastic function that transforms residuals. Accepts n-vector and returns n-vector.
<code>num_R</code>	Number of test statistic values to calculate in the randomization test.
<code>alpha</code>	Nominal test level (between 0 to 1).
<code>val_type</code>	The type of return value.

**Details**

For the regression  $y = X * \beta + e$ , this function is testing the following linear null hypothesis:

$H_0: \lambda' \beta = \lambda[1] * \beta[1] + \dots + \lambda[p] * \beta[p] = \lambda_0$ ,

where  $y$ ,  $X$ ,  $\lambda$ ,  $\lambda_0$  are specified in `model`. The assumption is that the errors,  $e$ , have some form of cluster invariance. Specifically:

$(e_1, e_2, \dots, e_n) \sim g\_invar(e_1, e_2, \dots, e_n)$ ,

where  $\sim$  denotes equality in distribution, and `g_invar` is the supplied invariance function.

**Value**

If `val_type = "decision"` (default) we get the test binary decision (1=REJECT  $H_0$ ).

If `val_type = "pval"` we get the test p-value.

If `val_type = "full"` we get the full test output, i.e., a `List` with elements `tobs`, `tvals`, the observed and randomization values of the test statistic, respectively.

**Note**

There is no guarantee that an arbitrary `g_invar` will produce valid tests. The `rrtest_clust` function has such guarantees under mild assumptions.

**See Also**

Life after bootstrap: residual randomization inference in regression models (Toulis, 2019)

<https://www.ptoulis.com/residual-randomization>

**Examples**

```
model = example_model(n = 100) # test H0: beta2 = 0 (here, H0 is true)
g_invar = function(e) sample(e) # Assume errors are exchangeable.
rrtest(model, g_invar) # same as rrtest_clust(model, "perm")
```

---

rrtest\_clust

*Residual randomization test under cluster invariances*

---

**Description**

This function tests the specified linear hypothesis in `model` assuming that the errors have some form of cluster invariance determined by `type` within the clusters determined by `clustering`.

**Usage**

```
rrtest_clust(
  model,
  type,
  clustering = NULL,
  num_R = 999,
  alpha = 0.05,
  val_type = "decision"
)
```

**Arguments**

model	Regression model and hypothesis. See <a href="#">example_model</a> for details.
type	A character, either "perm", "sign" or "double".
clustering	A List that specifies a clustering of datapoint indexes 1, ..., n. See <a href="#">example_clustering</a> . If NULL it takes default value according to type (see Note)
num_R	Number of test statistic values to calculate in the test.
alpha	Nominal test level (between 0 to 1).
val_type	The type of return value.

**Details**

For the regression  $y = X * \beta + e$ , this function is testing the following linear null hypothesis:

$H_0: \text{lam}' \beta = \text{lam}[1] * \beta[1] + \dots + \text{lam}[p] * \beta[p] = \text{lam}_0$ ,

where  $y$ ,  $X$ ,  $\text{lam}$ ,  $\text{lam}_0$  are specified in `model`. The assumption is that the errors,  $e$ , have some form of cluster invariance. Specifically:

- If `type = "perm"` then the errors are assumed exchangeable within the specified clusters:  
 $(e_1, e_2, \dots, e_n) \sim \text{cluster\_perm}(e_1, e_2, \dots, e_n)$ ,  
 where  $\sim$  denotes equality in distribution, and `cluster_perm` is any random permutation within the clusters defined by `clustering`. Internally, the test repeatedly calculates a test statistic by randomly permuting the residuals within clusters.
- If `type = "sign"` then the errors are assumed sign-symmetric within the specified clusters:  
 $(e_1, e_2, \dots, e_n) \sim \text{cluster\_signs}(e_1, e_2, \dots, e_n)$ ,  
 where `cluster_signs` is a random signs flip of residuals on the cluster level. Internally, the test repeatedly calculates a test statistic by randomly flipping the signs of cluster residuals.
- If `type = "double"` then the errors are assumed both exchangeable and sign symmetric within the specified clusters:  
 $(e_1, e_2, \dots, e_n) \sim \text{cluster\_signs}(\text{cluster\_perm}(e_1, e_2, \dots, e_n))$ ,  
 Internally, the test repeatedly calculates a test statistic by permuting and randomly flipping the signs of residuals on the cluster level.

**Value**

If `val_type = "decision"` (default) we get the test binary decision (1=REJECT  $H_0$ ).

If `val_type = "pval"` we get the test p-value.

If `val_type = "full"` we get the full test output, i.e., a List with elements `tobs`, `tvals`, the observed and randomization values of the test statistic, respectively.

**Note**

If `clustering` is NULL then it will be assigned a default value:

- `list(1:n)` if `type = "perm"`, where  $n$  is the number of datapoints;
- `as.list(1:n)` if `type = "sign"` or `"double"`.

As in bootstrap `num_R` is usually between 1000-5000.

**See Also**

Life after bootstrap: residual randomization inference in regression models (Toulis, 2019)

<https://www.ptoulis.com/residual-randomization>

**Examples**

```
# 1. Validity example
set.seed(123)
n = 50
X = cbind(rep(1, n), 1:n/n)
beta = c(0, 0)
rej = replicate(200, {
  y = X %>% beta + rt(n, df=5)
  model = list(y=y, X=X, lam=c(0, 1), lam0=0) # H0: beta2 = 0
  rrtest_clust(model, "perm")
})
mean(rej) # Should be ~ 5% since H0 is true.

# 2. Heteroskedastic example
set.seed(123)
n = 200
X = cbind(rep(1, n), 1:n/n)
beta = c(-1, 0.2)
ind = c(rep(0, 0.9*n), rep(1, .1*n)) # cluster indicator
y = X %>% beta + rnorm(n, sd= (1-ind) * 0.1 + ind * 5) # heteroskedastic
confint(lm(y ~ X + 0)) # normal OLS does not reject H0: beta2 = 0
cl = list(which(ind==0), which(ind==1))
model = list(y=y, X=X, lam=c(0, 1), lam0=0)

rrtest_clust(model, "sign") # errors are sign symmetric regardless of cluster.
# Cluster sign test does not reject because of noise.

rrtest_clust(model, "perm", cl) # errors are exchangeable within clusters
# Cluster permutation test rejects because inference is sharper.
```

---

r\_test\_c

*Residual randomization test*

---

**Description**

Implements the residual randomization test. The hypothesis tested is

**Usage**

```
r_test_c(y, X, lam, lam0, cluster_eps_r, use_perm, use_sign, num_R)
```

**Arguments**

y	Vector of outcomes (n x 1).
X	Matrix of covariates (n x p). First column should be 1's.
lam	Vector of coefficients in linear H0 (p x 1).
lam0	Scalar value for linear H0.
cluster_eps_r	A List with restricted residuals. See <a href="#">get_clustered_eps</a> .
use_perm	Boolean flag whether to use permutations within clusters.
use_sign	Boolean flag whether to use sign flips across clusters.
num_R	Integer of how many randomization values to calculate.

**Details**

H0:  $\text{lam}' \beta = \text{lam}[1] * \beta[1] + \dots + \text{lam}[p] * \beta[p] = \text{lam0}$ .

**Value**

A List with the observed test statistic value (tobs), and the randomization values (tvals)

---

two_sided_test	<i>Two-sided testing</i>
----------------	--------------------------

---

**Description**

Decides to reject or not based on observed test statistic value tobs and randomization values tvals. The test may randomize to achieve the specified level alpha when there are very few randomization values.

**Usage**

```
two_sided_test(tobs, tvals, alpha)
```

**Arguments**

tobs	The observed value of the test statistic (scalar).
tvals	Vector of randomization values of the test statistic (to compare with tobs).
alpha	Desired level of the test (between 0 to 1).

**Value**

Test decision (binary).

**See Also**

Testing Statistical Hypotheses (Ch. 15, Lehman and Romano, 2006)

# Index

check\_model, 2  
confint, 8–10  
  
example\_clustering, 2, 4, 9, 10, 13  
example\_model, 2, 3, 4, 11, 13  
  
fastLm, 3, 5  
  
get\_clustered\_eps, 4, 15  
  
OLS\_c, 5  
one\_sided\_test, 5, 6  
out\_pval, 6  
  
r\_test\_c, 14  
restricted\_OLS\_c, 6  
rrinf, 7  
rrinf\_clust, 9  
rrinfBase, 8  
rrtest, 7, 11  
rrtest\_clust, 9, 10, 12, 12  
  
two\_sided\_test, 15