Package: crctStepdown (via r-universe)

October 29, 2026

Title Univariate Analysis of Cluster Trials with Multiple Outcomes

Version 0.5.2

Date 2024-02-01

Description Frequentist statistical inference for cluster randomised trials with multiple outcomes that controls the family-wise error rate and provides nominal coverage of confidence sets. A full description of the methods can be found in Watson et al. (2023) <doi:10.1002/sim.9831>.

License CC BY-SA 4.0

Encoding UTF-8

RoxygenNote 7.2.3

LinkingTo Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), fastglm (>= 0.0.3)

Imports fastglm, Rcpp, ggplot2, ggpubr, methods, stringr, lme4, reshape2

Depends R (>= 3.0.2)

Suggests testthat

NeedsCompilation yes

Author Sam Watson [aut, cre] (<https://orcid.org/0000-0002-8972-769X>)

Maintainer Sam Watson <s.i.watson@bham.ac.uk>

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

Date/Publication 2024-02-03 02:27:57 UTC

RemoteUrl https://github.com/cran/crctStepdown

RemoteRef HEAD

RemoteSha 134a0f14b1d8f0099a0f37e9f43b40f2a72c8976

Contents

confint_search 2
est_null_model 3

gen_rand_order 4
outname_fit 5
perm_dist 5
permutation_test_impl 6
qscore_impl 7
setParallelCRT 7
simpleLM 8
stepdown 8
twoarm_sim 10

Index **12**

`confint_search` *Confidence interval search procedure*

Description

Search for the bound of a confidence interval using permutation test statistics

Usage

```
confint_search(
  start,
  b,
  n,
  nmodel,
  Xnull_,
  y,
  tr_,
  new_tr_mat,
  invS,
  family,
  family2,
  Z,
  type,
  nsteps = 1000L,
  weight = TRUE,
  alpha = 0.05,
  verbose = TRUE
)
```

Arguments

`start`	Numeric value indicating the starting value for the search procedure
`b`	Numeric value indicating the parameter estimate
`n`	Integer indicating the sample size
`nmodel`	Integer. The number of models

Xnull_	Numeric matrix. The covariate design matrix with the treatment variable removed
y	Numeric vector of response variables
tr_	Numeric vector. The original random allocation (0s and 1s)
new_tr_mat	A matrix. Each column is a new random treatment allocation with 1s (treatment group) and 0s (control group)
invS	A matrix. If using the weighted statistic then it should be the inverse covariance matrix of the observations
family	A statsfamily object
family2	A string naming the link function
Z	Matrix. Random effects design matrix describing cluster membership
type	String. Either "rw" for Romano-Wolf, "b" or "br" for bonferroni, "h" or "hr" for Holm, or "none"
nsteps	Integer specifying the number of steps of the search procedure
weight	Logical indicating whether to use the weighted (TRUE) or unweighted (FALSE) test statistic
alpha	The function generates (1-alpha)*100% confidence intervals. Default is 0.05.
verbose	Logical indicating whether to provide detailed output.

Value

The estimated confidence interval bound

`est_null_model` *Estimates null model*

Description

Given an lme4 model object and the value of the treatment effect parameter under the null hypothesis, the function returns a glm or lm object fitted under the null model with no cluster effects. For linear models (lmer) the value of the null is subtracted from the value of the outcome for those in receipt of the treatment and an lm model is fitted with no treatment effect. For generalised linear models (glmer) the model is refitted as a glm model with the treatment effect specified as an offset.

Usage

```
est_null_model(fit, data, tr_var = "treat", null_par)
```

Arguments

fit	An lme4 model object
data	The data frame used to fit model fit
tr_var	A string indicating the name of the column in data that is a binary indicator for whether the observation was under the treatment (1=treatment, 0=control)
null_par	Numeric the value of tr_var parameter under the null hypothesis

Value

An lm or glm model fit under the null model

Examples

```
out <- twoarm_sim()
data <- out[[1]]
  fit1 <- lme4::glmer(y1 ~ treat + (1|cl) ,
data=data,
family="poisson")

fit2 <- lme4::glmer(y2 ~ treat + (1|cl),
                    data=data,
                    family="poisson")
fitlist <- list(fit1,fit2)
nullfitlist <- list()
for(i in 1:length(fitlist)){
  nullfitlist[[i]] <- est_null_model(fitlist[[i]],
                                     data,
                                     tr_var = "treat",
                                     null_par = 0)
}
```

---

gen_rand_order *Function to generate a stepped-wedge cRCT randomisation allocation*

Description

Function to generate a stepped-wedge cRCT randomisation allocation. Assumes a baseline and endline period in which no clusters and all clusters have the intervention, respectively.

Usage

```
gen_rand_order(nJ, nT)
```

Arguments

nJ	Number of clusters
nT	Number of time points

Value

A data frame with columns cl and t indicating the time

Examples

```
gen_rand_order(10,7)
```

outname_fit *Extracts the dependent variable name from glm, lm, or mer model*

Description

Extracts the dependent variable name from glm, lm, or mer model

Usage

```
outname_fit(fit)
```

Arguments

fit	A fitted model object of class glm, lm, or *merMod

Value

A string with the name of the dependent variable from the model

Examples

```
out <- twoarm_sim()
data <- out[[1]]
fit1 <- lme4::glmer(y1 ~ treat + (1|cl) ,
                    data=data,
                    family="poisson")
outname_fit(fit1)
```

perm_dist *Extracts the test statistics*

Description

Extracts the test statistics from the output of the permute function. Returns the largest value from a specified subset of rows, each row is the test statistic from a different null hypothesis.

Usage

```
perm_dist(out, positions)
```

Arguments

out	Array output by the permute function
positions	Vector indicating which rows of out to use

Value

Vector of numeric values of length ncol(out)

`permutation_test_impl` *Generates realisations of the permutational test statistic distribution*

Description

Generates realisations of the permutational test statistic distribution from a given matrix of permutations

Usage

```
permutation_test_impl(
  resids,
  tr_mat,
  xb,
  invS,
  family2,
  Z,
  weight,
  iter = 1000L,
  verbose = TRUE
)
```

Arguments

`resids`	A numeric vector of generalised residuals
`tr_mat`	A matrix. Each column is a new random treatment allocation with 1s (treatment group) and 0s (control group)
`xb`	A numeric vector of fitted linear predictors
`invS`	A matrix. If using the weighted statistic then it should be the inverse covariance matrix of the observations
`family2`	A string naming the link function
`Z`	A matrix with columns indicating cluster membership
`weight`	Logical value indicating whether to use the weighted statistic (TRUE) or the unweighted statistic (FALSE)
`iter`	Integer. Number of permutation test iterations.
`verbose`	Logical indicating whether to report detailed output

Value

A numeric vector of quasi-score test statistics for each of the permutations

qscore_impl *The quasi-score statistic for a generalised linear mixed model*

Description

Generates the quasi-score statistic for a generalised linear mixed model

Usage

```
qscore_impl(resids, tr, xb, invS, family2, Z, weight = TRUE)
```

Arguments

resids	A numeric vector of generalised residuals
tr	A numeric vector of 1s (treatment group) and -1s (control group)
xb	A numeric vector of fitted linear predictors
invS	A matrix. If using the weighted statistic then it should be the inverse covariance matrix of the observations
family2	A string naming the link function
Z	A matrix with columns indicating cluster membership
weight	Logical value indicating whether to use the weighted statistic (TRUE) or the unweighted statistic (FALSE)

Value

A scalar value with the value of the statistic

setParallelCRT *Disable or enable parallelised computing*

Description

By default, the package will use multithreading for many calculations if OpenMP is available on the system. For multi-user systems this may not be desired, so parallel execution can be disabled with this function.

Usage

```
setParallelCRT(parallel_, cores_ = 2L)
```

Arguments

parallel_	Logical indicating whether to use parallel computation (TRUE) or disable it (FALSE)
cores_	Number of cores for parallel execution

Value

None, called for effects

`simpleLM` *A very basic linear model solver*

Description

Returns the OLS paramter estimates and fitted values. Used internally for quick fitting of null models.

Usage

```
simpleLM(y_, X_)
```

Arguments

`y_`	A vector of outcome values
`X_`	The design matrix of fixed effects

Value

A list with the parameter values and fitted values

`stepdown` *Conduct the randomisation-based stepdown procedure*

Description

For a set of models fit with lme4, base R, or glmmrBase, the function will conduct the randomisation tests and generate p-values for the null hypotheses of no treatment effect that controls the family-wise error rate, and generates a 100(1-alpha)% confidence set for the treatment effect model parameters.

Usage

```
stepdown(
  fitlist,
  tr_var = "treat",
  cl_var = "cl",
  data,
  alpha = 0.05,
  plots = TRUE,
  n_permute = 1000,
  nsteps = 1000,
```

```
  type = "rw",
  rand_func = NULL,
  confint = TRUE,
  sigma = NULL,
  ci_start_values = NULL,
  verbose = TRUE
)
```

Arguments

fitlist	A list of models fitted with lme4, base R (lm or glm), or glmmrBase. All models should be fit using the same data frame.
tr_var	String indicating the name of the column in data that is a binary indicator for whether the observation was under the treatment (1=treatment, 0=control)
cl_var	String specifying the name of the column identifying the clusters/cluster-time
data	A data frame containing the data used to fit the models in fitlist
alpha	Numeric. 100(1-alpha)% confidence intervals are calculated. Default it 0.05
plots	Logical indicating whether to plot permutational distributions and confidence interval search during running of function. Default is TRUE
n_permute	Number of permutations of the randomisation test to run
nsteps	Number of steps of the confidence interval search process
type	Method of correction: options are "rw" = Romano-Wolf randomisation test based stepdown, "h" = Holm standard stepdown, "h" = Holm stepdown using randomisation test, "b" = standard Bonferroni, "br" = Bonerroni using randomisation test, or "none" = randomisation test with no correction.
rand_func	String of the name of a function that re-randomises the clusters. The function must take the arguments nJ for the number of clusters and nT for the number of time periods. The function should produce a data frame that identifies the clusters in the treatment group under the new randomisation scheme. The data frame can either have a single column with name cl_var or two columns of cl_var and t identifying the cluster ID and time period a cluster joins the treatment group. If NULL then clusters are randomised in a 1:1 ratio to treatment and control
confint	Logical indicating whether to run the confidence interval search process
sigma	optional, list of estimated covariance matrices of the observations from the models in fitlist. If provided then the weighted q-score statistic is used.
ci_start_values	Optional list. The list should contain named vectors "upper" and/or "lower" that provide a set of starting values for the upper and/or lower confidence interval searches, respectively. Alternatively, a named scalar scale can be provided such that the starting values of the confidence interval search procedure are est +/- scale*SE.
verbose	Logical indicating whether to provide detailed output

Value

A data frame with the point estimates, p-values, and confidence intervals

Examples

```
out <- twoarm_sim()
data <- out[[1]]
  fit1 <- lme4::glmer(y1 ~ treat + (1|cl) ,
data=data,
family="poisson")

fit2 <- lme4::glmer(y2 ~ treat + (1|cl),
                    data=data,
                    family="poisson")
 stepdown(fitlist=list(fit1,fit2),
          data=data,
          n_permute = 100,
          nsteps=100,
          plots=FALSE,
          verbose=TRUE)
```

---

twoarm_sim *Simulates data from a two-arm parallel cluster randomised trial*

Description

Simple simulation of two Poisson distributed outcomes for a two-arm parallel cluster randomised trial with no baseline measures. A log-linear model is specified y~Poisson(lambda) with lambda = exp(mu + beta*D + theta) where D is the treatment effect indicator equal to one in clusters with the treatment and zero otherwise, and theta~N(0,sigma^2) is the cluster random effect. Used for testing error rates of the methods.

Usage

```
twoarm_sim(
  nJ = c(7, 7),
  N = 20,
  mu = rep(1, 2),
  beta = c(0, 0),
  sig_cl = rep(0.05, 2)
)
```

Arguments

nJ	Vector of two integers with the number of clusters in treatment and control arms
N	Number of individuals per cluster
mu	Vector of two numeric values with the intercept terms for the two models on the log scale

beta	Vector of two numeric values that are the treatment effect parameters in the two models
sig_cl	Vector of two values equal to the variance of the random effect in each model

Value

A list consisting of: (1) data frame with the cluster IDs (cl), treatment effect indicators (treat), and two outcomes (y1, y2), and (2) the values of the treatment effect parameters used in the simulation.

Examples

```
out <- twoarm_sim()
data <- out[[1]]
```

Index

confint_search, 2

est_null_model, 3

family, *3*

gen_rand_order, 4

outname_fit, 5

perm_dist, 5
permutation_test_impl, 6

qscore_impl, 7

setParallelCRT, 7
simpleLM, 8
stats, *3*
stepdown, 8

twoarm_sim, 10