Package 'kernscr'

April 17, 2023

Type Package

Title Kernel Machine Score Test for Semi-Competing Risks

Version 1.0.6

Date 2023-04-17

Maintainer Boris P Hejblum <boris.hejblum@u-bordeaux.fr>

Description Kernel Machine Score Test for Pathway Analysis in the Presence of Semi-Competing Risks. Method is detailed in: Neykov, Hejblum & Sinnott (2018) <doi:10.1177/0962280216653427>.

Depends R (>= 3.0)

Imports graphics, MASS, mvtnorm, stats

Suggests readxl, org.Hs.eg.db

License GPL-2 | file LICENSE

LazyData TRUE

BugReports https://github.com/borishejblum/kernscr/issues

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Matey Neykov [aut],
Boris P Hejblum [aut, cre],
Jennifer A Sinnot [aut]

Repository CRAN

Date/Publication 2023-04-17 16:10:02 UTC

R topics documented:

kernscr-package *kernscr: a package to perform Kernel Machine Score Test for Pathway Analysis in the Presence of Semi-Competing Risks*

Description

Kernel Machine Score Test for Pathway Analysis in the Presence of Semi-Competing Risks

Details

Package:	kernscr
Type:	Package
Version:	1.0.6
Date:	2023-04-17
License:	GPL-2

The main function of the kernscr package is `compute_all_tests`

Author(s)

Matey Neykov, Boris P. Hejblum, Jennifer A. Sinnott — Maintainer: Boris P. Hejblum

References

Neykov M, Hejblum BP, Sinnot JA, Kernel Machine Score Test for Pathway Analysis in the Presence of Semi-Competing Risks, *Stat Methods in Med Res*, , 27(4): 1099-1114 (2018). <doi: 10.1177/0962280216653427>.

cancer_pathways *70 pathways from MSigDB c2CP*

Description

70 pathways from MSigDB c2CP

Usage

```
data("cancer_pathways")
```

Format

a list of 70 relevant pathways from an old version of MSigDB c2CP containing the Entrez IDs.

References

MJ van de Vijver,YD He, LJ van't Veer, H Dai, AAM Hart, DW Voskuil, A gene-expression signature as a predictor of survival in breast cancer, *The New England Journal of Medicine*, 347(25):1999-2009, 2002.

T Cai, G Tonini, X Lin, Kernel Machine Approach to Testing the Significance of Multiple Genetic Markers for Risk Prediction, *Biometrics*, 67(3):975-986, 2011.

Examples

```
data("cancer_pathways")

if(interactive()){
##get the data from Vijver publication

#clinical data
import_xls_from_zip <- function(urlPath, filename, zipname, skip=0){
 zipFile <- paste0(zipname, ".zip")
 download.file(paste0(urlPath, zipFile), zipFile)
 unzip(zipFile, exdir="./temp_unzip")
 xlsFile <- paste0("./temp_unzip/", filename, ".xls")
 res <- readxl::read_xls(xlsFile, skip=skip)
 unlink(zipFile)
 unlink("./temp_unzip", recursive=TRUE)
 return(res)
}

BC_dat_clin <- import_xls_from_zip2(urlPath="http://ccb.nki.nl/data/",
                                    filename="Table1_ClinicalData_Table",
                                    zipname="nejm_table1",
                                    skip=2
                                    )
BC_dat_clin <- BC_dat_clin[order(BC_dat_clin$SampleID), ]
col2rmv <- 1:ncol(BC_dat_clin)
BC_dat_clin$ID <- paste0("S", BC_dat_clin$SampleID)
rownames(BC_dat_clin) <- BC_dat_clin$ID
BC_dat_clin$evdeath <- BC_dat_clin$EVENTdeath
BC_dat_clin$tsurv <- BC_dat_clin$TIMEsurvival
BC_dat_clin$evmeta <- BC_dat_clin$EVENTmeta
BC_dat_clin$tmeta<- pmin(BC_dat_clin$TIMEsurvival, BC_dat_clin$TIMEmeta, na.rm=TRUE)
samples2rmv <- c("S28", "S122", "S123", "S124", "S133", "S138", "S139", "S141", "S221", "S222",
            "S224", "S226", "S227", "S228", "S229", "S230", "S231", "S237", "S238", "S240",
            "S241", "S248", "S250", "S251", "S252", "S254", "S292", "S317", "S342", "S371",
                 "S379", "S380", "S397", "S398", "S401")
BC_dat_clin <- BC_dat_clin[-which(BC_dat_clin$ID %in% samples2rmv), -col2rmv]
head(BC_dat_clin)



#import genomics data
urlPath="http://ccb.nki.nl/data/"
zipFile <- paste0("ZipFiles295Samples", ".zip")
```

```
download.file(paste0(urlPath, zipFile), zipFile)
unzip(zipFile, exdir="./temp_unzip")
unlink(zipFile)
unlink("./temp_unzip/Readme.txt", recursive=FALSE)
txtfiles <- list.files("./temp_unzip/")
BC_dat_exp <- NULL
for(f in txtfiles){
 temp_exp <- read.delim(paste0("./temp_unzip/", f))
 if(f==txtfiles[1]){
   gene_id <- as.character(temp_exp[-1, 1])
   gene_symbol <- as.character(temp_exp[-1, 2])
 }
 temp_exp <- temp_exp[-1, grep("Sample.", colnames(temp_exp))]
 colnames(temp_exp) <- gsub("Sample.", "S", colnames(temp_exp))
 if(f==txtfiles[1]){
   BC_dat_exp <- temp_exp
 }else{
   BC_dat_exp <- cbind(BC_dat_exp, temp_exp)
 }
}
BC_dat_exp_all <- cbind.data.frame("SYMBOL"=gene_symbol, BC_dat_exp[,  BC_dat_clin$ID])
unlink("./temp_unzip", recursive=TRUE)

# translating the pathways from Entrez ID to gene symbol
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)){
 library(org.Hs.eg.db)
 x <- org.Hs.egSYMBOL
 mapped_genes <- mappedkeys(x)
 xx <- as.list(x[mapped_genes])
 cancer_pathways_Symbol <- lapply(cancer_pathways, function(v){unlist(xx[v])})
 sapply(cancer_pathways, function(x){length(intersect(x, rownames(BC_dat_exp)))/length(x)})
}
}
```

`compute_all_tests` *Testing pathway risk association*

Description

This functions computes p-values `frm` score tests of genetic pathway risk association in 5 different models

Usage

```
compute_all_tests(
  data,
  ind_gene = 7:ncol(data),
  num_perts = 1000,
  Ws = NULL,
```

```
  rho = NA,
  kernel = c("linear", "gaussian", "poly"),
  d = 2,
  pca_thres = 0.9,
  get_ptb_pvals = FALSE,
  ...
)
```

Arguments

data	a data.frame of N rows and set up as the output from sim_SCR_data with columns: • XR: time to recurrence / death / censoring • XD: time to death / censoring • DeltaR: Indicator of censoring (0), recurrence (1), or death (2) for this earliest time XR • DeltaD: Indicator of censoring (0) or death (1) • XPFS: time to recurrence / death / censoring (=XR) • DeltaPFS: Indicator of censoring (0) or recurrence or death, whichever came first (1) • Z_1,...,Z_P: genomic variables
ind_gene	columns indices of genes in the pathway of interest. Default is 7:ncol(data)).
num_perts	number of perturbations used. Default is 1000.
Ws	optional inputed perturbations, should be a vector of length N x num_perts containing i.i.d. realization of a random variable with mean=0 and variance=1.
rho	a vector of rhos, such as one found created from the range returned by findRhoInterval, used for tuning non-linear kernel. Only used if kernel is not "linear". Default is NA. Currently not available for use by user-defined kernels.
kernel	a character string indicating which kernel is used. Possible values (currently implemented) are "linear", "gaussian" or "poly". Otherwise, this can also be a user defined kernel function. See genericKernelEval.
d	if kernel is "poly", the polynomial power. Default is 2 (quadratic kernel).
pca_thres	a number between 0 and 1 giving the threshold to be used for PCA. Default is 0.9. If NULL, no PCA is performed.
get_ptb_pvals	a logical flag indicating whether perturbed p-values should be returned as part of the results. Default is FALSE.
...	extra parameters to be passed to a user-defined kernel.

Value

either a vector of p-values for 5 different models with names:

- "SCR": Semi-Competing Risks
- "PFS": Progression Free Survival
- "CR": Competing Risks

- "OS": Overall Survival
- "SCR_alt": SCR allowing different tuning parameters for the two event time processes

or else if get_ptb_pvals is TRUE, a list with 2 elements:

- "obs_pvals": a vector containing the observed p-values for each of the 5 models as described above
- "null_pvals_perts": a matrix of dimensions num_perts x 5 containing the corresponding perturbed p-values

References

Neykov M, Hejblum BP, Sinnot JA, Kernel Machine Score Test for Pathway Analysis in the Presence of Semi-Competing Risks, submitted, 2016.

Examples

```
## First generate some Data
feat_m_fun <- function(X){
 sin(X[,1]+X[,2]^2)-1
}
feat_d_fun <-  function(X){
 (X[,4]-X[,5])^2/8
}
mydata <- sim_SCR_data(data_size = 400, ncol_gene_mat = 20, feat_m = feat_m_fun,
                       feat_d = feat_d_fun, mu_cen = 40, cov=0.5)

#initial range
ind_gene <- c(7:ncol(mydata))
my_rho_init <- seq(0.01, 20, length=300)*length(ind_gene)
range(my_rho_init)

if(interactive()){
# compute the interval for rho
rho_set <- findRhoInterval(tZ=t(mydata[,ind_gene]), rho_init = my_rho_init, kernel="gaussian")
rho_set
range(my_rho_init) # good to check that the interval produced here is strictly contained in rho_init
# otherwise, expand rho.init and rerun

rhos <- exp(seq(log(rho_set[1]),log(rho_set[2]), length=50))

# run the tests with Gaussian kernel
compute_all_tests(data = mydata, num_perts=1000, rho=rhos, kernel="gaussian")
# run the tests with linear kernel
compute_all_tests(data=mydata, num_perts=1000, kernel="linear")
}
```

`findRhoInterval` *Find an interval constraining the rho parameter for a non linear kernel*

Description

Find an interval constraining the rho parameter for a non linear kernel

Usage

```
findRhoInterval(
  tZ,
  rho_init = seq(0.01, 20, length = 300) * nrow(tZ),
  kernel = c("gaussian", "poly"),
  d = NA,
  rate_range = c(1.5, 4),
  pca_thres = 0.9,
  warning_suppress = TRUE
)
```

Arguments

`tZ`	a `P` x `N` matrix of genomic covariates (i.e., the usual data array Z transposed)
`rho_init`	an initial large range of possible rhos, which will be considered to see if they are reasonable tuning parameters for the kernel. Default is `seq(0.01, 20, length=300)*P`. See Details.
`kernel`	character string specifying a nonlinear kernel. Currently supported options are: `"gaussian"` or `"poly"`
`d`	if `kernel` is `"poly"`, the polynomial power (e.g. d=2 for quadratic kernel). Default is `NA`.
`rate_range`	a vector of length 2 indicating the range of alpha in the paper. Default is `c(1.5,4)`.
`pca_thres`	a number between `0` and `1` giving the threshold to be used for PCA. Default is `0.9`. If `NULL`, no PCA is performed.
`warning_suppress`	logical flag. Indicating whether the warnings should be suppress during the linear model fitting step. Default is `TRUE`. See details.

Details

This function will print `rho_init` range and the range of valid tuning parameters. If that range butts up against either the upper or lower bound of `rho_init`, you can rerun this function with a bigger `rho_init`.

Finding the right tuning parameters includes a step of fitting a linear model which can fail because some tuning parameters yield only one eigenvector. We want to eliminate those tuning parameters, so this is OK. However, in case one want to suppress (numerous) annoying warning messages, use the `warning_suppress` argument.

Value

an upper and lower bound to look for rho

Examples

```
## First generate some Data
feat_m_fun <- function(X){
 sin(X[,1]+X[,2]^2)-1
}
feat_d_fun <-  function(X){
 (X[,4]-X[,5])^2/8
}
mydata <- sim_SCR_data(data_size = 400, ncol_gene_mat = 20, feat_m = feat_m_fun,
                       feat_d = feat_d_fun, mu_cen = 30, cov=0.5)

#initial range
ind_gene <- c(7:ncol(mydata))
my_rho_init <- seq(0.01, 20, length=300)*length(ind_gene)
range(my_rho_init)

if(interactive()){
# compute the interval for rho
rho_set <- findRhoInterval(tZ=t(mydata[,ind_gene]), rho_init = my_rho_init, kernel="gaussian")
rho_set
range(my_rho_init) # good to check that the interval produced here is strictly contained in rho_init
# otherwise, expand rho.init and rerun

#rhos <- exp(seq(log(rho_set[1]),log(rho_set[2]), length=50))
}
```

`sim_SCR_data` *Data Simulation Function*

Description

Data Simulation Function

Usage

```
sim_SCR_data(
  data_size,
  ncol_gene_mat,
  feat_m,
  feat_d,
  mu_cen,
  cov,
  lam_m = 1/15,
  lam_d = 1/20,
```

```
  norm_vcov = c(1, 0.5, 0.5, 1)
)
```

Arguments

data_size	an integer giving the simulated sample size N
ncol_gene_mat	an integer giving the simulated number of genomic covariates P
feat_m	a function that transforms the genomic features into the signal for the metastasis process. This function should a matrix of dimensions N X P as its only argument.
feat_d	a function that transforms the genomic features into the signal for the death process. This function should a matrix of dimensions N X P as its only argument.
mu_cen	mean of the exponential censoring process
cov	the correlation between the genomic covariates
lam_m	baseline hazard constant for metastasis process. Default is 1/15.
lam_d	baseline hazard constant for death process. Default is 1/20.
norm_vcov	vector of length 4 of correlation between errors between the two processes on the normal scale before being complementary-log-log-transformed. Default is c(1,.5,.5,1).

Value

a data.frame with columns:

- XR: time to recurrence / death / censoring
- XD: time to death / censoring
- DeltaR: Indicator of censoring (0), recurrence (1), or death (2) for this earliest time XR
- DeltaD: Indicator of censoring (0) or death (1)
- XPFS: time to recurrence / death / censoring (=XR)
- DeltaPFS: Indicator of censoring (0) or recurrence or death, whichever came first (1)
- Z_1,...,Z_P: genomic variables

Examples

```
feat_m_fun <- function(X){
 sin(X[,1]+X[,2]^2)-1
}
feat_d_fun <-  function(X){
 (X[,4]-X[,5])^2/8
}
mydata <- sim_SCR_data(data_size = 400, ncol_gene_mat = 20, feat_m = feat_m_fun,
                       feat_d = feat_d_fun, mu_cen = 30, cov=0.5)
head(mydata)
## how many experience both events
mean(mydata[,"DeltaR"]==1 & mydata[,"DeltaD"]==1)
## how many only recur
mean(mydata[,"DeltaR"]==1 & mydata[,"DeltaD"]==0)
```

```
## how many only die
mean(mydata[,"DeltaR"]==2 & mydata[,"DeltaD"]==1)
## how many are censored
mean(mydata[,"DeltaR"]==0 & mydata[,"DeltaD"]==0)
```

Index