Package 'minPtest'

February 20, 2015

Type Package

Title Gene region-level testing procedure for SNP data, using the min P test resampling approach

Version 1.7

Date 2013-12-17

Depends scrime, Epi

Suggests parallel, snowfall

Author Stefanie Hieke

Maintainer Stefanie Hieke <hieke@imbi.uni-freiburg.de>

Description Package minPtest is designed for estimating a gene region-level summary for SNP data from case-control studies using a permutation-based resampling method, called min P test, allowing execution on a compute cluster or multicore computer.

License GPL (>= 2.14)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-19 19:05:00

R topics documented:

`generateSNPs` *Simulation of SNP data*

Description

Simulates SNP data with genotypes coded by 0, 1 and 2 as well as a binary and a continuous covariate, together with case-control status specified by logistic regression.

Usage

```
generateSNPs(n, gene.no, block.no, block.size, p.same,
             p.different = NULL, p.minor, n.sample, SNPtoBETA)
```

Arguments

`n`	an integer specifying the number of observations (cases and controls with 1:1 match) that should be generated. `n` should be an even number.
`gene.no`	an integer specifying the number of genes that should be generated.
`block.no`	an integer specifying the number of blocks per gene.
`block.size`	an integer specifying the number of SNPs per block.
`p.same`	either a numeric value specifying the probability for neighborhood SNPs within a block or a numeric vector of length `block.size`. In the latter case the argument `p.different` is ignored and has to be specified in the first item of `p.same`. The remaining items in the `p.same` vector specify the probabilities for neighborhood SNPs within the blocks, i.e. the probability that two neighboring alleles are equal within a block. If a numeric value, all SNPs, except the first item of each a block, will have the same neighborhood probability. If a vector of length `block.size`, each SNP of each block will have the neighborhood probability specified in the corresponding entry in `p.same`.
`p.different`	a numeric value specifying the probability for neighborhood blocks within a gene which is used if `p.same` is a scalar. The argument is ignored if `p.same` is a numeric vector and has to be specified in the first entry in `p.same`.
`p.minor`	a vector of length `block.no` containing the allele frequencies of the SNPs within a block. All SNPs in a block will have the same allele frequency.
`n.sample`	an integer specifying the number of simulated subjects from which the observations (case-control status) `n` are drawn.
`SNPtoBETA`	a matrix of non-negative numeric values of dimension `m * 2` consisting of the SNP index (first column) with `m <= snp.no` and the parameters (size of effect) of these SNPs (second column) for generating of case-control status.

Details

generateSNPs generates a matrix consisting of n observations, snp.no=gene.no*block.no*block.size SNPs with genotypes coded by 0, 1 and 2, two automatically generated covariates for adjustment or matching and the matchset numbers. The neighborhood probabilities for SNPs is given by p.different and/or p.same and the allele frequencies for SNPs is given by p.minor. The allele frequencies (p.minor) and the probabilities for neighborhood blocks (p.different) and/or p.same, respectively, can differ between the blocks on a gene but are repeated similar over all genes gene.no. The simulated SNP data structure is similar as in Schwender et al. (2011).

The response is determined by a logistic regression model given the SNPs, the binary covariate and the continuous covariate in the sim.cov matrix:

P(Y=1|sim.cov)=exp(sim.cov*beta)/(1+exp(sim.cov*beta))

Using the the model P(Y=1|sim.cov) is computed for each subject in n.sample, then the case and one control status for each of the n.sample subjects are determined by drawn randomly from a Bernoulli distribution using the probability P(Y=1|sim.cov). From these n.sample subjects one case and one control observation is randomly drawn. This algorithm is repeated n/2 times for each randomly sampled value from the continuous covariate, i.e. one case and one control is randomly drawn from each of n/2 times to generate the complete response vector of length(n).

As output generateSNPs provides a response vector y, a SNP matrix x, a covariate matrix cov and a matchset vector matchset which can directly be used as input for the minPtest, see the example of the minPtest function.

Value

An object of class 'generateSNPs', which is a list containing the following components:

sim.data	a matrix with n rows and (snp.no+4) columns containing response (case-control status) values, simulated SNP values, continuous matching covariate, binary matching covariate and matchset numbers.
y	a numeric response vector coded with 0 (coding for controls) and 1 (coding for cases) of length n.
x	a numeric n * snp.no matrix containing the simulated SNP data with genotypes coded by 0, 1 and 2.
cov	a n * 2 matrix containing the continuous matching covariate (likewise to age) and the binary matching covariate (likewise to gender).
matchset	a numeric vector of length n containing the matching numbers (1:1 match).
snp.no	number of SNPs in the simulated data set.
SNPtoGene	the mapping matrix of dimension p x 2 comprising of SNP names (first column) and the name of the genes (second column) on which the SNPs are located.
call	call.

Author(s)

Stefanie Hieke <hieke@imbi.uni-freiburg.de>

References

Schwender, H. et al. (2011). Testing SNPs and sets of SNPs for importance in association studies. Biostatistics, 12, 18-32.

See Also

minPtest

Examples

```
# Generate a data set consisting of 100 subjects and 200 SNPs on 5 genes,
# with 4 blocks per gene with block size of 10, i.e. 10 SNPs per block
# yielding 40 SNPs per gene:

# specifying the matrix for 6 SNPs and corresponding parameters (effect size)
# for the generation of case-control status

SNP <- c(6,26,54,135,156,186)
BETA <- c(0.9,0.7,1.5,0.5,0.6,0.8)
SNPtoBETA <- matrix(c(SNP,BETA),ncol=2,nrow=6)
colnames(SNPtoBETA) <- c("SNP.item","SNP.beta")

set.seed(191)
sim1 <- generateSNPs(n=100,gene.no=5,block.no=4,block.size=10,p.same=0.9,
p.different=0.75,p.minor=c(0.1,0.4,0.1,0.4),n.sample=80,SNPtoBETA=SNPtoBETA)

# to reconstruct how to adopt the output from generateSNPs,
# see the example of the minPtest function.
```

`minPtest` *A gene region-level testing procedure for each candidate gene based on resampling using the min P test*

Description

Permutation-based p-values estimation via min P test, a gene region-level summary for each candidate gene. The gene region-level summary assesses the smallest p-trend within each gene region comparing cases and controls. The min P test is permutation-based method that can be based on different univariate tests per SNP. Inference is based on the permutation distribution of the ordered p-values from the marginal tests of each SNP. Potentially accelerated by parallelization, if a compute cluster or a multicore computer is available.

Usage

```
minPtest(y, x, SNPtoGene, formula = NULL, cov = NULL, matchset = NULL,
         permutation = 1000, seed = NULL, subset = NULL,
         parallel = FALSE, ccparallel = FALSE,
         trace = FALSE, aggregation.fun = min,
```

```
          adj.method=c("bonferroni","holm","hochberg",
                       "hommel","BH","BY","fdr","none"),
          ...)
```

Arguments

y	a numeric response vector coded with 0 (coding for controls) and 1 (coding for cases) of length n.
x	a numeric n * p matrix of covariates (i.e. SNPs) containing the genotypes coded by 0, 1 and 2. Thus, each column is assumed to represent one of the SNPs with corresponding column names. Detail for SNP coding are given below.
SNPtoGene	a mapping matrix of dimension p x 2 comprising SNP names (first column) which are same as the column names of x, and the gene names (second column) on which the SNPs are located.
formula	(optional) for unconditional or conditional logistic regression, respectively, with or without covariates other than SNPs. A symbolic description of the model to be fitted including covariates only, see glm or clogistic, the latter requires library **Epi**, else the default method Cochran Armitage Trend Test, which requires library **scrime**, is fitted. Details of model specification are given below.
cov	(optional) a n * q matrix containing the covariates for adjustment with corresponding column names.
matchset	(optional) a numeric vector of length n containing matching numbers, needed for conditional logistic regression.
permutation	number of permutations employed to obtain a null distribution.
seed	(optional) vector of length permutation. Allows reproducibility even when running in parallel and for different numbers of parallel processes.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
parallel	indicates whether computation in the permuted data sets should be performed in parallel using package **parallel**. If TRUE, the parallelization requires at least two cores. A value larger than 1 is taken to be the number of cores.
ccparallel	logical value indicates whether computation should be performed in parallel on a compute cluster, using package **snowfall**. If TRUE the initialization function of this package, sfInit(), should be called before calling minPtest. See Details.
trace	logical value indicating whether progress in estimation should be indicated by printing the number of permutation that is currently used. (ignored if running in parallel via **snowfall**).
aggregation.fun	function that is used to combine the trend p-values over multiple loci within a gene region. By default the minimum ("min") is applied to obtain candidate gene region-level summaries. Any other function to integrate the p-values into one single test statistic can be used, e.g., median or different functions designed by the user.
adj.method	correction method for multiple hypothesis testing. By default the Bonferroni method ("bonferroni") is used. Any other correction method as in p.adjust can be used.

`...` Further arguments for `aggregation.fun`. In case of `NA/NaN` within the evaluated marginal trend p-values for the SNPs from the original data (`psnp`) or within the permuted trend p-values in the permutation samples (`psnpperm`), `na.rm=TRUE` has to be specified.

Details

The idea of the gene region-level summary, using the min P test procedure (Westfall and Young, 1993; Westfall et al., 2002; Chen et al., 2006), is to identify candidate genes by assessing the statistical significance of the smallest p-trend from a set of SNPs (single nucleotide polymorphisms) within each gene region comparing cases and controls by permutation-based resampling methods. A SNP occurs when a single nucleotide, (A), (T), (C) or (G), in the genome differs between individuals and, in addition, this variation, substitution of one nucleotide for another, occurs in more than 1% of a population. A SNP can take three possible values (genotypes): either there is no SNP variant in comparison to some reference coding (homozygous reference (0)) or the SNP variant occurs on one of the two base pair positions (heterozygous (1)), or both base pairs have a variant comparing to the reference coding. **minPtest** permits to include, instead of the genotypes 0, 1 and 2, also combined carrier SNPs, e.g. coding 0 and 1 (1 + 2).

Computation of the min P test is based on the marginal trend p-values for a set of univariate SNP disease association and the trend p-values for the permutation samples for each SNP. The **minPtest** package brings together three different kinds of tests to compute such p-values that are scattered over several R packages, and automatically selects the one most appropriate for the design at hand. In any case a response vector `y`, a SNP matrix `x` and a mapping matrix `SNPtoGene` are required. Then the default, a Cochran Armitage Trend Test (Cochran, 1954; Armitage, 1955), is automatically fitted to compute p-values. The Cochran Armitage Trend Test does not depend on covariates and matching scenario. Additionally adding a formula, see also `glm` from package **base**, and a covariate matrix `cov` an unconditional logistic regression is fitted. Unconditional logistic regression can be used without or with covariates for adjustment; either `formula=y~1` or `formula=y~cov1+cov2+...`. The former does not need any information relative to covariates and matching scenario. However, the latter is general for frequency matching with the inclusion of matching variables for adjustment specified in the covariate matrix `cov`. Providing a matchset, as in the case of 1:1; 1:2 etc. matching, and a formula, see also `clogistic` from package **Epi**, a conditional logistic regression is fitted. Conditional logistic regression can be used without or with covariates for adjustment; either `formula=y~1` or `formula=y~cov1+cov2...`. In the latter case covariates other than matching variables can be used and have to be specified in the covariate matrix `cov`. In general, there are two possibilities to specify the formula, first if no covariates are used for adjustment, the formula has to be written as `y~1` without specifying the covariate matrix `cov`. Second if covariates other than SNPs are used for adjustment, the formula has to be written as response vector `y` on the left of a `~` operator, and the clinical covariates on the right, as well as a covariate matrix has to be specified.

If SNPs genotypes are coded by 0, 1 and 2, they are included as continuous variables in the logistic regression models. If SNPs are coded as carrier SNPs 0 and 1, they are included as binary variables in the logistic regression models. If covariates are used for adjustment, the column names of the covariate matrix `cov` have to be specified as used in the formula specification, to link the formula with the covariate matrix `cov`.

Missing SNP genotypes in `x` or, if used, missing values in `cov` are accounted for, as each marginal test makes use of the available data for that SNP in `x` and for that covariate in `cov` only. The **minPtest** uses all subjects with available data for each SNP (and covariates) when fitting Cochran

Armitage Trend Test or unconditional logistic regression. Note that in conditional logistic regression, the matched subjects are removed together in case of 1:1 matching. In the 1:2 matching scenario, matched subjects are removed when the missing occurs in a case, otherwise when a missing occurs in one control, only that control is removed.

Concerning parallelization on a compute cluster, i.e. with argument `ccparallel=TRUE`, there are two possibilities to run **minPtest**:

1. Start R on a commandline with sfCluster (Knaus et al., 2009) and preferred options, e.g. number of cpus. The initialization function of package **snowfall**, `sfInit()`, should be called before calling minPtest.
2. Use any other solutions supported by **snowfall**. Argument `ccparallel` has to be set to TRUE and number of cpus can be chosen in the `sfInit()` function.

sfCluster is a Unix tool for convenient management of R parallel processes. It is available at www.imbi.uni-freiburg.de/parallel, with detailed information.

A print function returns a short overviews of the results. The print function describes the number of subjects included in the analysis, which method is used by the package, briefing of the number of genes, the number of SNPs, the number of missings in the SNP matrix `x` and the number of permutations used for the fit. A `summary.minPtest` and a `plot.minPtest` function are available.

Value

An object of class 'minPtest', which is a list containing the following components:

`minp`	`nrgene * 1` matrix of permutation-based p-values of the min P test for each candidate gene.
`p.adj.minp`	`nrgene * 1` matrix of corrected permutation-based p-values for each candidate gene.
`psnp`	`nrsnp * 1` matrix of marginal trend p-values for each SNP from the original data set.
`p.adj.psnp`	`nrsnp * 1` matrix of corrected marginal trend p-values for each SNP from the original data set.
`psnpperm`	`nrsnp * n.permute` matrix of permuted trend p-values for each SNP in each permutation step.
`zgen`	`nrgene * 1` matrix of min P test statistics for each candidate gene from the original data set.
`zgenperm`	`nrgene * n.permute` matrix of permuted min P test statistics for each candidate gene in each permutation step.
`n`	number of subjects in the original data set.
`nrsnp`	number of SNPs in the original data set.
`nrgene`	number of genes in the original data set.
`snp.miss`	number of missings in the SNP matrix `x`.
`n.permute`	number of permutations.
`method`	used method.
`call`	call.
`SNPtoGene`	the mapping matrix of dimension `p x 2` comprising of SNP names (first column) and names of the genes (second column) on which the SNPs are located.

Author(s)

Stefanie Hieke <hieke@imbi.uni-freiburg.de>

References

Armitage,P. (1955). Tests for linear trends in proportions and frequencies. Biometrics, 11(3), 375-386.

Chen,B.E. et al. (2006). Resampling-based multiple hypothesis testing procedures for genetic case-control association studies. Genetic Epidemiology, 30, 495-507.

Cochran,W.G. (1954). Some methods for strengthening the common chi-squared tests. Biometrics, 10(4), 417-451.

Knaus,J. et al. (2009). Easier parallel computing in R with snowfall and sfCluster. The R Journal, 1, 54-59.

Westfall,P.H. et al.(2002). Multiple tests for genetic effects in association studies. Methods Mol Biol, 184, 143-168.

Westfall,P.H. and Young,S.S. (1993). Resampling-Based Multiple Testing: Example and Methods for p-Value Adjustment. Wiley, New York.

See Also

summary.minPtest, plot.minPtest

Examples

```
# generate a simulated data set as in the example of the function generateSNPs
# consisting of 100 subjects and 200 SNPs on 5 genes.

SNP <- c(6,26,54,135,156,186)
BETA <- c(0.9,0.7,1.5,0.5,0.6,0.8)
SNPtoBETA <- matrix(c(SNP,BETA),ncol=2,nrow=6)
colnames(SNPtoBETA) <- c("SNP.item","SNP.beta")

set.seed(191)
sim1 <- generateSNPs(n=100,gene.no=5,block.no=4,block.size=10,p.same=0.9,
p.different=0.75,p.minor=c(0.1,0.4,0.1,0.4),n.sample=80,SNPtoBETA=SNPtoBETA)

# Cochran Armitage Trend Test without covariates and default permutations.
# Example: Run R sequential

### Seed
set.seed(10)
seed1 <- sample(1:1e7,size=1000)
###
minPtest.object <- minPtest(y=sim1$y, x=sim1$x, SNPtoGene=sim1$SNPtoGene,
                    seed=seed1)
```

plot.minPtest *Plot method for "minPtest" object*

Description

plot method for an object of class 'minPtest'. Plots allowing to get an impression of important genes or/and SNPs.

Usage

```
## S3 method for class 'minPtest'
plot(x, type=c("gene" ,"SNP" ,"both"), level=0.05, lambda=1, gene.name=FALSE,
     sigPch=pch, nonsigPch=pch, pch=20,
     sigLty=lty, nonsigLty=lty, lty=1,
     sigCol=col, nonsigCol=col, col=NULL, xlab, ...)
```

Arguments

x	an object of class minPtest.
type	by default, permutation-based p-values for each gene are plotted ("gene"). "SNP": marginal p-values for each SNP are plotted. "both": marginal p-values for each SNP and the transformed permutation-based p-values for each gene are displayed in a combined plot, see Details.
level	a numeric threshold that specifies which genes or/and SNPs are highlighted in the plot. I.e. not depending on the used type argument, the genes or/and SNPs with adjusted permutation-based p-values or/and marginal p-value, respectively, which are smaller than or equal to that threshold are by default highlighted in red. Default is 0.05.
lambda	only useful for type="both". A numeric value to scale the y-axis for the permutation-based p-values of the genes (indicated at the right hand side). Default is 1.
gene.name	only useful for type="SNP" and type="both". A logical value, if TRUE, the gene names are shown at the x-axis. Default is FALSE.
sigPch	Type of plotting for significant permutation-based p-values (type="gene") or for significant marginal p-values (type="SNP" and type="both") (if neither sigPch nor pch set: points)
nonsigPch	Type of plotting for non significant permutation-based p-values (type="gene") or non significant marginal p-values (type="SNP" and type="both") (if neither nonsigPch nor pch set: points)
pch	Set type of plotting for both sigPch and nonsigPch (but can be overwritten by sigPch and nonsigPch if set)
sigLty	only used for type="both". Type of plotting for significant permutation-based p-values (if neither sigLty nor lty set: solid lines)
nonsigLty	only used for type="both". Type of plotting for non significant permutation-based p-values (if neither nonsigLty nor lty set: solid lines)

lty	only used for type="both". Set type of plotting for both sigLty and nonsigLty (but can be overwritten by sigLty and nonsigLty if set)
sigCol	Color for significant genes or/and significant SNPs (if neither sigCol nor col set: red)
nonsigCol	Color for non significant genes or/and non significant SNPs (if neither nonsigCol nor col set: black)
col	Set color for both sigCol and nonsigCol (but can be overwritten by sigCol and nonsigCol if set)
xlab	xlab (Default: Gene if type="both") and SNP if type=SNP or type="both", respectively
...	Further arguments for the plot function.

Details

The function plots either (-log_{10}) transformed permutation-based p-values for each gene or (-log_{10}) transformed marginal p-values for each SNP in a basic scatterplot. The y-axis is (-log_{10}) transformed to obtain a disposition as a Manhattan plot for the points of the marginal p-values of the SNPs. Furthermore, an alternative given by the function is to display the marginal p-values for each SNP and the transformed permutation-based p-values for each gene in a combined plot. The (-log_{10}) transformed marginal p-values for each SNP are plotted as points. In addition, horizontal lines of (-lambda*log_{10}) transformed permutation-based p-values of each gene, covering all SNPs located on that gene, are plotted. The composed plot is indicated by two separated y-axes ((-log_{10}(psnp)) at left hand side and (-lambda*log_{10}(minp)) at the right hand side). After correction for multiple hypothesis testing depending on the level and the argument adj.method in the minPtest function, but not depending on the used type of plot, significant genes and SNPs are by default highlighted in red, i.e. each permutation-based p-value or/and marginal p-value smaller than or equal to the level, respectively, is highlighted in red.

Value

No value returned

Note

The default for gene.name=FALSE, used for type="SNP" and type="both", should kept for performance reasons, if a large number of genes are included in the fit. For type="both" no ylim should be specified as the plot is indicated by two separate y-axes.

Author(s)

Stefanie Hieke <hieke@imbi.uni-freiburg.de>

See Also

minPtest, generateSNPs

Examples

```
## Continuing the example from minPtest and generateSNPs:
# generate a data set consisting of 100 subjects and 200 SNPs on 5 genes.

SNP <- c(6,26,54,135,156,186)
BETA <- c(0.9,0.7,1.5,0.5,0.6,0.8)
SNPtoBETA <- matrix(c(SNP,BETA),ncol=2,nrow=6)
colnames(SNPtoBETA) <- c("SNP.item","SNP.beta")

set.seed(191)
sim1 <- generateSNPs(n=100,gene.no=5,block.no=4,block.size=10,p.same=0.9,
             p.different=0.75,p.minor=c(0.1,0.4,0.1,0.4),
                     n.sample=80,SNPtoBETA=SNPtoBETA)


# Cochran Armitage Trend Test without covariates and default permutations.
# Example: Run R sequential

### Seed
set.seed(10)
seed1 <- sample(1:1e7,size=1000)
###
minPtest.object <- minPtest(y=sim1$y, x=sim1$x, SNPtoGene=sim1$SNPtoGene,
                    seed=seed1)
### Combined plot for permutation-based p-values and marginal p-values.
plot(minPtest.object,type="both",lambda=0.5,gene.name=TRUE)

## Combined plot for permutation-based p-values and marginal
## p-values. Plot permutation-based p-values and significant marginal
## p-values as blue dotted lines and blue points
## plot(minPtest.object,type="both",lambda=0.5,
##      gene.name=TRUE,sigCol="blue",sigLty=2)
```

`summary.minPtest` *Summary method for a "minPtest" object*

Description

Summary method for objects of class "minPtest"

Usage

```
## S3 method for class 'minPtest'
summary(object, level = 0.05, sign.SNP = FALSE, ...)
```

Arguments

`object`	an object of class `minPtest`, i.e. the output of a `minPtest` call.

`level`	a numeric threshold that specifies which genes are shown in the summary, i.e. the genes with adjusted permutation-based p-values smaller than or equal to that threshold are printed. Default is 0.05.
`sign.SNP`	a logical value; if TRUE, print, in addition to the genes selected by a `level`, the SNPs with adjusted marginal p-values smaller than or equal to the `level` (same as for genes) located on these genes. Default is FALSE, all SNP located on these genes, selected according to the `level`, are shown. Default is FALSE.
`...`	Further arguments for the `summary` method. Not used.

Details

Prints the genes with adjusted permutation-based p-value smaller than or equal to a `level`, the corresponding permutation-based p-values, the adjusted permutation-based p-values and the SNPs located on these genes, either all SNPs or SNPs selected by the `level`, sorted by the adjusted marginal p-values, with marginal p-values and adjusted marginal p-values.

Value

`summary.minPtest` returns a list. Each item characterizes a gene, selected according to a `level`, list items are named by means of these genes. Each gene item contains a list of data frames, a data frame for the permutation-based p-values and adjusted permutation-based p-values for this gene and a data frame for the marginal p-values and adjusted marginal p-values for the SNPs located on that gene, either all SNPs or SNPs selected by the `level`.

Author(s)

Stefanie Hieke <hieke@imbi.uni-freiburg.de>

See Also

`generateSNPs`, `minPtest`

Examples

```
## Continuing the example from minPtest and generateSNPs:
# generate a data set consisting of 100 subjects and 200 SNPs on 5 genes.

SNP <- c(6,26,54,135,156,186)
BETA <- c(0.9,0.7,1.5,0.5,0.6,0.8)
SNPtoBETA <- matrix(c(SNP,BETA),ncol=2,nrow=6)
colnames(SNPtoBETA) <- c("SNP.item","SNP.beta")

set.seed(191)
sim1 <- generateSNPs(n=100,gene.no=5,block.no=4,block.size=10,p.same=0.9,
             p.different=0.75,p.minor=c(0.1,0.4,0.1,0.4),
                    n.sample=80,SNPtoBETA=SNPtoBETA)

# Cochran Armitage Trend Test without covariates and default permutations.
# Example: Run R sequential
```

```
### Seed
set.seed(10)
seed1 <- sample(1:1e7,size=1000)
###
minPtest.object <- minPtest(y=sim1$y, x=sim1$x, SNPtoGene=sim1$SNPtoGene,
                    seed=seed1)
###
summary(minPtest.object)
```

Index