Package ‘RSNPset’

December 14, 2017

Type Package

Title Efficient Score Statistics for Genome-Wide SNP Set Analysis

Version 0.5.3

Date 2017-12-11

Author Chanhee Yi, Alexander Sibley, and Kouros Owzar

Maintainer Alexander Sibley <alexander.sibley@dm.duke.edu>

Description An implementation of the use of efficient score statistics in genome-wide SNP set analysis with complex traits. Three standard score statistics (Cox, binomial, and Gaussian) are provided, but the package is easily extensible to include others. Code implementing the inferential procedure is primarily written in C++ and utilizes parallelization of the analysis to reduce runtime. A supporting function offers simple computation of observed, permutation, and FWER and FDR adjusted p-values.

License GPL-3

Imports fastmatch (>= 1.0-4), foreach (>= 1.4.1), doRNG (>= 1.5.3), qvalue (>= 1.34), Rcpp (>= 0.10.4)

LinkingTo Rcpp, RcppEigen

Suggests knitr

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-12-14 19:55:23 UTC

R topics documented:

RSNPset-package *Efficient Score Statistics for Genome-Wide SNP Set Analysis*

Description

An implementation of three standard efficient score statistics (Cox, binomial, and Gaussian) for use in genome-wide SNP set analysis with complex traits.

Details

Package:	RSNPset
Type:	Package
Version:	0.5.3
Date:	2017-12-11
License:	GPL-3

This package is designed for the analysis of sets of related SNPs, using genes as the loci of interest, but the methodology can naturally be applied to other genomic loci, including bands and pathways. The core function, `rsnpset()`, provides options for three efficient score statistics, binomial, Gaussian, and Cox, for the analysis of binary, quantitative, and time-to-event outcomes, but is easily extensible to include others. Code implementing the inferential procedure is primarily written in *C++* and utilizes parallelization to reduce runtime. A supporting function, `rsnpset.pvalue()`, offers easy computation of observed, resampling, FWER-adjusted, and FDR-adjusted p-values, and summary functions provide diagnostic measures and results metadata.

Note

The inferential procedures are written primarily in *C++* and utilize linear algebra routines from the **Eigen** library. This implementation is facilitated using the templates provided by the **Rcpp** and **RcppEigen** packages. Parallelization of the analysis, with reproducible randomization, is enabled by using the **doRNG** package to add parallel backends to looping constructs provided by the **foreach** package. The FDR-adjusted p-values are obtained using the **qvalue** package. Use of the **fastmatch** package allows efficient cross-referencing of SNP rsIDs in the data with the SNP sets.

Author(s)

Chanhee Yi, Alexander Sibley, and Kouros Owzar

Maintainer: Alexander Sibley <alexander.sibley@dm.duke.edu>

See Also

Functions available in this package: `rsnpset`, `rsnpset.pvalue`, and supporting summary functions `summary.RSNPset`, `summary.RSNPset.pvalue`

For more information on supporting packages, see: `Eigen`, `Rcpp`, `RcppEigen`, `doRNG`, `foreach`, `qvalue`, `fastmatch`

The snplist package can be used to generate sets of SNPs for analysis with this package.

Examples

```
n <- 200    # Number of patients
m <- 1000   # Number of SNPs

set.seed(123)

G <- matrix(rnorm(n*m), n, m)   # Normalized SNP expression levels
rsids <- paste0("rs", 1:m)      # SNP rsIDs
colnames(G) <- rsids

K <- 10                         # Number of SNP sets
genes <- paste0("XYZ", 1:K)     # Gene names
gsets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsids)
names(gsets) <- genes

# Survival outcome
time <- rexp(n, 1/10)           # Survival time
event <- rbinom(n, 1, 0.9)      # Event indicator

res <- rsnpset(Y=time, delta=event, G=G, snp.sets=gsets, score="cox")
head(res)
summary(res)
rsnpset.pvalue(res, qfun=function(x) NA) # q-values suppressed for small sample size (K = 10)

## Not run:
# Optional parallel backend
library(doParallel)
registerDoParallel(cores=8)

res <- rsnpset(Y=time, delta=event, G=G, snp.sets=gsets, score="cox", B=1000)
rsnpset.pvalue(res)
## End(Not run)

# Binary outcome
set.seed(123)
Y <- rbinom(n, 1, 0.5)

head(rsnpset(Y=Y, G=G, snp.sets=gsets, score="binomial", v.method="empirical"))
head(rsnpset(Y=Y, G=G, snp.sets=gsets, score="binomial", v.method="asymptotic"))
```

---

rsnpset *RSNPset Analysis Function*

Description

Compute the (binomial, Gaussian, or Cox) efficient score for genome-wide SNP set analysis with binary, uncensored quantitative, or right-censored time-to-event traits.

Usage

```
rsnpset(Y, delta=NULL, G, X=NULL, snp.sets,
        score=c("cox", "binomial", "gaussian"), B=0,
        r.method="monte carlo", v.method="empirical",
        v.permute=FALSE, ret.rank=FALSE, pinv.check=FALSE,
        pinv.method="specdecomp", pinv.tol=7.8e-8)
```

Arguments

`Y` Vector of outcomes. Can be binary, continuous, or non-negative (time-to-event), dictating the type of score to request in argument `score`. Required.

`delta` Vector of event indicators for survival outcomes. Must be binary: 1 indicates event, 0 indicates censored. Required when `score="cox"`, unused otherwise.

`G` Matrix of SNP allele counts or expression levels. One row per patient and one column per SNP. Must be type `double`. Required.

`X` Optional matrix of covariates. One row per patient and one column per cofactor. Must be numeric with non-colinear columns. If used, an intercept column will be appended automatically. Currently implemented only for `score="gaussian"`.

`snp.sets` List of sets of SNPs to be analyzed together. Each element should be a vector of values (usually rsIDs) corresponding to names of columns of `G`. Required.

`score` Score statistic to be calculated. (Must be appropriate for the type of outcome given in `Y`). Required to be one of `c("cox", "binomial", "gaussian")`.

`B` Number of replication sets used to calculate the empirical distribution of the score statistics and empirical p-values. Default is 0 (no replications will be generated).

`r.method` Resampling method. Default is `"monte carlo"`. `r.method="permutation"` is allowed when the model does not include covariates (`X=NULL`).

`v.method` Method for calculating the variance of the score statistic. Default is `"empirical"` for all values of `score`, and is the only method currently implemented for `score="gaussian"` or `score="cox"`. `score="binomial"` also allows `"asymptotic"`.

`v.permute` Boolean indicating whether or not to recalculate the variance of the score statistics for each permutation replicate. If `FALSE`, variance matrices are computed only for the observed data and then reused for the permutation replicates. `FALSE` when `r.method="monte carlo"` (does not apply). `r.method="permutation"` also allows `TRUE`.

`ret.rank` Boolean indicating whether or not to return the rank of the variance matrices for the permutation replicates. `FALSE` when `r.method="monte carlo"` (does not apply). `r.method="permutation"` also allows `TRUE`.

`pinv.check` Boolean. If `TRUE`, the function returns several diagnostic measures (see below) of the Penrose-Moore inverse of the variance matrices. Default is `FALSE`.

`pinv.method` Method for calculating the inverse of the variance matrices. Currently, only the default of `"specdecomp"` is implemented.

`pinv.tol` Number indicating the tolerance for determining the rank of the variance matrix (see below). Default is `7.8e-8`.

Details

For each SNP set, the function computes the statistic W as

$$W = \boldsymbol{U}_{\bullet}^{T} \Sigma^{+} \boldsymbol{U}_{\bullet}$$

where Σ^{+} is the Penrose-Moore inverse of the variance matrix $\Sigma = \boldsymbol{U}^{T}\boldsymbol{U}$, and $\boldsymbol{U}_{\bullet} = \boldsymbol{U}^{T}\mathbf{1}$. Here, $\boldsymbol{U}$ is an n by J matrix where entry *i,j* corresponds to the *i*th patient's contribution to the score statistic for SNP *j*. Statistical performance is improved by centering the values of `G` for each SNP prior to calculating $\boldsymbol{U}$.

Under suitable regularity conditions, the distribution of W can be approximated by a chi-squared distribution with degrees of freedom equal to the rank of Σ. The rank is determined as the number of eigenvalues greater than `pinv.tol`. For more information on SNP sets and the efficient score, see the package vignette.

If `B > 0` and `r.method="monte carlo"`, `B` resampling replicates of W are obtained by replacing $\boldsymbol{U}_{\bullet} = \boldsymbol{U}^{T}\mathbf{1}$ with $\boldsymbol{U}_{\bullet} = \boldsymbol{U}^{T}\boldsymbol{Z}$, where $\boldsymbol{Z}$ is a vector of n normal random values. Replications are executed in parallel, if a backend is available.

If `B > 0` and `r.method="permutation"`, `B` permutation replicates of W are obtained by permuting the values of `Y`, or, in the case of `score="cox"`, by permuting `(Y,delta)` pairs. Permutation replicates are executed in parallel, if a backend is available. Note that `r.method="permutation"` is not appropriate when the model includes covariates.

If `pinv.check=TRUE`, the following diagnostic measures of the Penrose-Moore inverse are calculated.

Column	Absolute largest element of:
`d0`	$\Sigma - \boldsymbol{QDQ}$
`d1`	$\Sigma\Sigma^{+}\Sigma - \Sigma$
`d2`	$\Sigma^{+}\Sigma\Sigma^{+} - \Sigma^{+}$
`d2`	$(\Sigma\Sigma^{+})^{T} - \Sigma\Sigma^{+}$
`d4`	$(\Sigma^{+}\Sigma)^{T} - \Sigma^{+}\Sigma$

where $\boldsymbol{QDQ}$ is the spectral decomposition of Σ. Departure of these values from zero indicates poor performance of the Penrose-Moore inverse.

Value

An S3 class `RSNPset` object consisting of a list of objects of class `data.frame`. The first data frame in the list describes the observed statistics, with `B` additional data frames corresponding to the requested resampling replicates. The rows of each data frame correspond to the elements of the `snp.sets` argument. The first column is `W`, the calculated efficient score for that set, and the second is `rank`, the rank of the variance matrix of the computed statistic. If `ret.rank=FALSE`, the ranks are not returned for the replicates. (They are assumed to be identical). The first data frame in the list also includes a third column, `m`, giving the number of SNPs analyzed in that SNP set.

If `pinv.check=TRUE`, a list of `B+1` data frames, each with one row per SNP set and five columns of diagnostic measures of the calculated Penrose-Moore inverse (see above), is returned as an attribute. This and other attributes of the function's execution can be accessed using the class's `summary` function.

Note

I. This function does not require that all entries of an element of `snp.sets` be present in the matrix `G`. If an element contains column names that are not present in `G`, the function will execute without objection and return a value based on the subset of columns that *are* present.

II. No statistics are returned for SNP sets which include SNPs with missing values (i.e. `NA`s in the selected columns of the matrix `G`).

III. As the Cox score statistic (`method="cox"`) is not a sum of independent patient level scores, some level of pruning of SNPs is recommended. The permutation resampling facilities of the package can be utilized to assess the performance of the asymptotic inference. The development of robust methods for calculating the score statistics and approximating the covariance matrix is under way.

See Also

The function `rsnpset.pvalue` provides options for computing p-values for the returned statistics.

The function `summary.RSNPset` provides diagnostics and information about the function's execution.

Examples

```
n <- 200    # Number of patients
m <- 1000   # Number of SNPs

set.seed(123)

G <- matrix(rnorm(n*m), n, m)   # Normalized SNP expression levels
rsids <- paste0("rs", 1:m)      # SNP rsIDs
colnames(G) <- rsids

K <- 10                         # Number of SNP sets
genes <- paste0("XYZ", 1:K)     # Gene names
gsets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsids)
names(gsets) <- genes

# Binary outcome
Yb <- rbinom(n, 1, 0.5)
head(rsnpset(Y=Yb, G=G, snp.sets=gsets, score="binomial", v.method="empirical"))
head(rsnpset(Y=Yb, G=G, snp.sets=gsets, score="binomial", v.method="asymptotic"))

# Quantitative outcome
Yq <- rbinom(n, 1, 0.5)
head(rsnpset(Y=Yq, G=G, snp.sets=gsets, score="gaussian"))

# Survival outcome
time <- rexp(n, 1/10)           # Survival time
event <- rbinom(n, 1, 0.9)      # Event indicator
head(rsnpset(Y=time, delta=event, G=G, snp.sets=gsets, score="cox"))

## Not run:
# Optional parallel backend
library(doParallel)
```

```
registerDoParallel(cores=8)

res <- rsnpset(Y=Yb, G=G, snp.sets=gsets, score="binomial", B=1000)
length(res) # = 1001
## End(Not run)
```

rsnpset.pvalue *RSNPset P-value Function*

Description

Calculate observed, resampling, FWER-adjusted, and FDR-adjusted p-values for statistics from the function rsnpset().

Usage

```
rsnpset.pvalue(result, pval.transform=FALSE, qfun=function(x){qvalue(x)$qvalue})
```

Arguments

result	Result from rsnpset(), an "RSNPset" S3 class object. Required.
pval.transform	Boolean indicating if the resampling p-values should be computed by comparing the observed p-value to the resampling p-values (TRUE). If not (FALSE), they are computed by comparing the observed statistics to the resampling statistics (may not be appropriate for r.method="permutation"). Note that rsnpset() must be run with B > 0 in order to use pval.transform=TRUE. Default is FALSE.
qfun	Function used to calculate false discovery rate adjusted p-values. See below. Default is function(x){qvalue(x)$qvalue}.

Details

See below.

Value

An S3 class RSNPset.pvalue object that extends data.frame, with one row for each of the K SNP sets in result, columns W, rank, m, and two or more additional columns of p-values. Two columns, p, and q are always returned. If rsnpset() was run with B > 0, the columns pB and qB are included as well. If pval.transform=TRUE, the returned p-value columns will be p, pB, PB, q, and qB.

Column	Definition
W	Observed statistic
rank	Rank of the variance matrix for the observed data
m	Number of SNPs analyzed in the SNP set

Column	P-value	Definition
p	Asymptotic*	pchisq(W,rank,lower.tail=FALSE)
pB	Resampling**	See below.
PB	Family-wise error adjusted***	See below.
q	False discovery rate adjusted	qfun(p)
qB	Resampling FDR adjusted	qfun(pB)

* For W and rank from rsnpset().

** By default, the unadjusted resampling p-values are computed by comparing the observed statistics to the replication statistics. Note that a large number of replications may be required in order to account for multiple testing. For each SNP set, the value for pB is sum(W <= Wb)/B, where W is the observed statistic for the SNP set, Wb is a vector of resampling statistics, and B is the number of replications. If pval.transform=TRUE, then for each SNP set, the value for pB is sum(p > pb)/B where p is the observed p-value, and pb is a vector of the p-values of the B resampling statistics. It is possible that pB may be 0 for some SNP sets. To prevent this, pmax(pB,1/B) is returned instead.

*** The column PB is only returned if pval.transform=TRUE. For each SNP set, the value for PB is sum(p > Zb)/B, where Zb a vector of length B. Each element of Zb is the smallest resampling p-value across all K SNP sets for the bth replication. It is possible that PB may be 0 for some SNP sets. To prevent this, pmax(PB,1/B) is returned instead.

Note

The qvalue() function, used by default in qfun, can fail for small numbers of replications/SNP sets. To overcome this, the qfun argument can be used to define a new q-value function, or to assign arguments for the qvalue() function. For example:

qfun=function(x){qvalue(x, robust=TRUE)$qvalue}.

See Also

This function computes p-values for the statistics from the function rsnpset.

For sorting and reviewing the p-values, see summary.RSNPset.pvalue.

More information on qvalue.

Examples

```
n <- 200    # Number of patients
m <- 1000   # Number of SNPs

set.seed(123)
G <- matrix(rnorm(n*m), n, m)   # Normalized SNP expression levels
rsids <- paste0("rs", 1:m)      # SNP rsIDs
colnames(G) <- rsids

K <- 15                         # Number of SNP sets
genes <- paste0("XYZ", 1:K)     # Gene names
gsets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsids)
names(gsets) <- genes
```

```
# Survival outcome
time <- rexp(n, 1/10)           # Survival time
event <- rbinom(n, 1, 0.9)      # Event indicator

## Not run:
# Optional parallel backend
library(doParallel)
registerDoParallel(cores=8)
## End(Not run)

# B >= 1000 is typically recommended
res <- rsnpset(Y=time, delta=event, G=G, snp.sets=gsets, score="cox",
               B=50, r.method="permutation", ret.rank=TRUE)
rsnpset.pvalue(res, pval.transform=TRUE)
```

`summary.RSNPset` *RSNPset Analysis Summary Function*

Description

Summary function to display execution information from `rsnpset()`.

Usage

```
## S3 method for class 'RSNPset'
summary(object, verbose=TRUE, ...)
```

Arguments

`object`	Result from `rsnpset()`, an "RSNPset" S3 class object. Required.
`verbose`	Boolean indicating if additional information about the results should be reported. Default is `TRUE`.
`...`	Additional arguments affecting the summary produced.

Details

If `verbose=TRUE`, prints a summary of the execution conditions of `rsnpset()`. The default report includes:

- The number of samples (the length of the `rsnpset()` argument `Y`).
- The range in sizes of the analyzed SNP sets.
- The number of SNP sets not analyzed (e.g., due to not containing any valid SNPs).
- The number of SNP sets containing SNPs not used in the analysis (e.g., due to being missing from the data).

If resampling replicates were generated, the report will also include:

- The method and number of resampling replicates computed (i.e. the value of the `rsnpset()` arguments `r.method` and `B`).
- Whether or not the ranks of the variance matrices of permutation replicates are included in the results (i.e. the value of the `rsnpset()` argument `ret.rank`).
- Whether or not the variance matrices were recomputed for each permutation replicate (i.e. the value of the `rsnpset()` argument `v.permute`).

If `rsnpset()` was run with `pinv.check=TRUE`, the value of the argument `pinv.tol` will also be reported.

Value

If `rsnpset()` was run with `pinv.check=TRUE`, a list of `data.frame` objects is returned, each containing the following diagnostic measures of the calculated Penrose-Moore inverses for the observed and permutation results.

Column	**Absolute largest element of:**
`d0`	$\Sigma - \boldsymbol{QDQ}$
`d1`	$\Sigma\Sigma^{+}\Sigma - \Sigma$
`d2`	$\Sigma^{+}\Sigma\Sigma^{+} - \Sigma^{+}$
`d2`	$(\Sigma\Sigma^{+})^{T} - \Sigma\Sigma^{+}$
`d4`	$(\Sigma^{+}\Sigma)^{T} - \Sigma^{+}\Sigma$

where $\boldsymbol{QDQ}$ is the spectral decomposition of Σ. Departure of these values from zero indicates poor performance of the Penrose-Moore inverse. If `rsnpset()` was run with `pinv.check=FALSE`, the function returns `NA`.

Note

If `pinv.check=TRUE` and the number of permutations is large, the user may wish to capture the resulting diagnostic measures in an object for examination, as in the example below, as opposed to having them printed.

See Also

The function `rsnpset` provides a description of the meaning of these reported values, as well as an explanation as to how they influence the results.

Examples

```
n <- 200    # Number of patients
m <- 1000   # Number of SNPs

set.seed(123)
G <- matrix(rnorm(n*m), n, m)   # Normalized SNP expression levels
rsids <- paste0("rs", 1:m)      # SNP rsIDs
colnames(G) <- rsids

K <- 15                         # Number of SNP sets
```

```
genes <- paste0("XYZ", 1:K)     # Gene names
gsets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsids)
names(gsets) <- genes

# Survival outcome
time <- rexp(n, 1/10)           # Survival time
event <- rbinom(n, 1, 0.9)      # Event indicator

## Not run:
# Optional parallel backend
library(doParallel)
registerDoParallel(cores=8)
## End(Not run)

# B >= 1000 is typically recommended
res <- rsnpset(Y=time, delta=event, G=G, snp.sets=gsets,
               score="cox", B=50, r.method="permutation",
               ret.rank=TRUE, pinv.check=TRUE)

pinvcheck <- summary(res)
pinvcheck[["Observed"]]
```

summary.RSNPset.pvalue

RSNPset P-value Summary Function

Description

Summary function to sort and display p-values resulting from `rsnpset.pvalue()`.

Usage

```
## S3 method for class 'RSNPset.pvalue'
summary(object, sort="p", decreasing=FALSE,
        nrows=10, dropcols=c(""), verbose=FALSE, ...)
```

Arguments

object	Result from `rsnpset.pvalue()`, an "RSNPset.pvalue" S3 class object. Required.
sort	Character string indicating column by which to sort results. If not one of `c("W", "rank", "m", "p", "pB", "PB", "q", "qB")` results will be sorted by the row names (i.e. the names of the SNP sets) instead. Default is `"p"`.
decreasing	Boolean indicating if the sort column should be arranged in decreasing order. Default is `FALSE`.
nrows	Integer indicating number of rows to display. Default is 10.
dropcols	Character vector corresponding names of columns of to be suppressed from the summary. Default is none.

`verbose` Boolean indicating if additional information about the p-value calculations should be reported. Default is `FALSE`.

`...` Additional arguments affecting the summary produced.

Details

As a typical GWAS study may span thousands of SNPs and SNP sets, this function allows for the succinct reporting of p-values for the most significant results. For more information about the different columns reported, see the documentation for `rsnpset.pvalue()`. If `verbose=TRUE`, a note will be printed with the total number of SNP sets and replications used in the calculations, as well as the value of the `pval.transform` argument from `rsnpset.pvalue()`.

Value

A `data.frame` object subset from `object`, the result of `rsnpset.pvalue()`. Rows are selected based on the `sort`, `decreasing`, and `nrows` arguments, and columns are selected based on the `dropcols` argument.

See Also

The function `rsnpset.pvalue` provides a description of the different p-values computed, as well as the other columns in the results.

Examples

```
n <- 200    # Number of patients
m <- 1000   # Number of SNPs

set.seed(123)
G <- matrix(rnorm(n*m), n, m)   # Normalized SNP expression levels
rsids <- paste0("rs", 1:m)      # SNP rsIDs
colnames(G) <- rsids

K <- 15                         # Number of SNP sets
genes <- paste0("XYZ", 1:K)     # Gene names
gsets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsids)
names(gsets) <- genes

# Survival outcome
time <- rexp(n, 1/10)           # Survival time
event <- rbinom(n, 1, 0.9)      # Event indicator

## Not run:
# Optional parallel backend
library(doParallel)
registerDoParallel(cores=8)
## End(Not run)

# B >= 1000 is typically recommended
res <- rsnpset(Y=time, delta=event, G=G, snp.sets=gsets, score="cox",
                B=50, r.method="permutation", ret.rank=TRUE)
pvals <- rsnpset.pvalue(res, pval.transform=TRUE)

summary(pvals)

summary(pvals, sort="W", decreasing=TRUE, nrows=5, dropcols=c("p","rank"), verbose=TRUE)
```

Index