Package 'locStra'

June 14, 2020

Type Package

Title Fast Implementation of (Local) Population Stratification Methods

Version 1.4

Date 2020-06-11

Author Georg Hahn [aut,cre], Sharon M. Lutz [ctb], Christoph Lange [ctb]

Maintainer Georg Hahn <ghahn@hsph.harvard.edu>

Description Fast and fully sparse 'cpp' implementations to compute the genetic covariance matrix, the genomic relationship matrix, the Jaccard matrix, and the s-matrix of an input matrix. Full support for sparse matrices from the R-package 'Matrix'. Additionally, a 'cpp' implementation of the power method (von Mises iteration) algorithm to compute the largest eigenvector of a matrix is included, and a function to compute sliding windows.

License GPL (>= 2)

Imports Rcpp (>= 0.12.13), Rdpack, Matrix

RdMacros Rdpack

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-14 16:30:05 UTC

R topics documented:

covMatrix *Cpp implementation of a function to compute the covariance matrix for a (sparse) matrix. The function is equivalent to the R command 'cov' applied to matrices.*

Description

Cpp implementation of a function to compute the covariance matrix for a (sparse) matrix. The function is equivalent to the R command 'cov' applied to matrices.

Usage

```
covMatrix(m, dense = FALSE)
```

Arguments

m	A (sparse) matrix for which the covariance matrix is sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
dense	Flag to switch between purpose-built dense or sparse implementations. Default is dense=FALSE.

Value

The covariance matrix of m.

References

R Core Team (2014). R: A Language and Environment for Statistical Computing. R Foundation for Stat Comp, Vienna, Austria.

Examples

```
library(locStra)
library(Matrix)
m <- matrix(sample(0:1,15,replace=TRUE),ncol=3)
sparseM <- Matrix(m,sparse=TRUE)
print(covMatrix(sparseM))
```

fullscan	*Main function: A full scan of the input data* m *using a collection of windows given by the two-column matrix* windows. *For each window, the data is processed using the function* matrixFunction *(this could be e.g. the* covMatrix *function), then the processed data is summarised using the function* summaryFunction *(e.g., the largest eigenvector computed with the function* powerMethod*), and finally the global and local summary scores (e.g., the largest eigenvectors) are compared using the function* comparisonFunction *(e.g., the vector correlation with R's function* cor*). The function returns a two-column matrix which contains per row the global (e.g., the correlation between global and local eigenvectors) and local (e.g., the correlation between the local eigenvector for the current window and the eigenvector for the last window) summary statistics for each window.*

Description

Main function: A full scan of the input data m using a collection of windows given by the two-column matrix windows. For each window, the data is processed using the function matrixFunction (this could be e.g. the covMatrix function), then the processed data is summarised using the function summaryFunction (e.g., the largest eigenvector computed with the function powerMethod), and finally the global and local summary scores (e.g., the largest eigenvectors) are compared using the function comparisonFunction (e.g., the vector correlation with R's function cor). The function returns a two-column matrix which contains per row the global (e.g., the correlation between global and local eigenvectors) and local (e.g., the correlation between the local eigenvector for the current window and the eigenvector for the last window) summary statistics for each window.

Usage

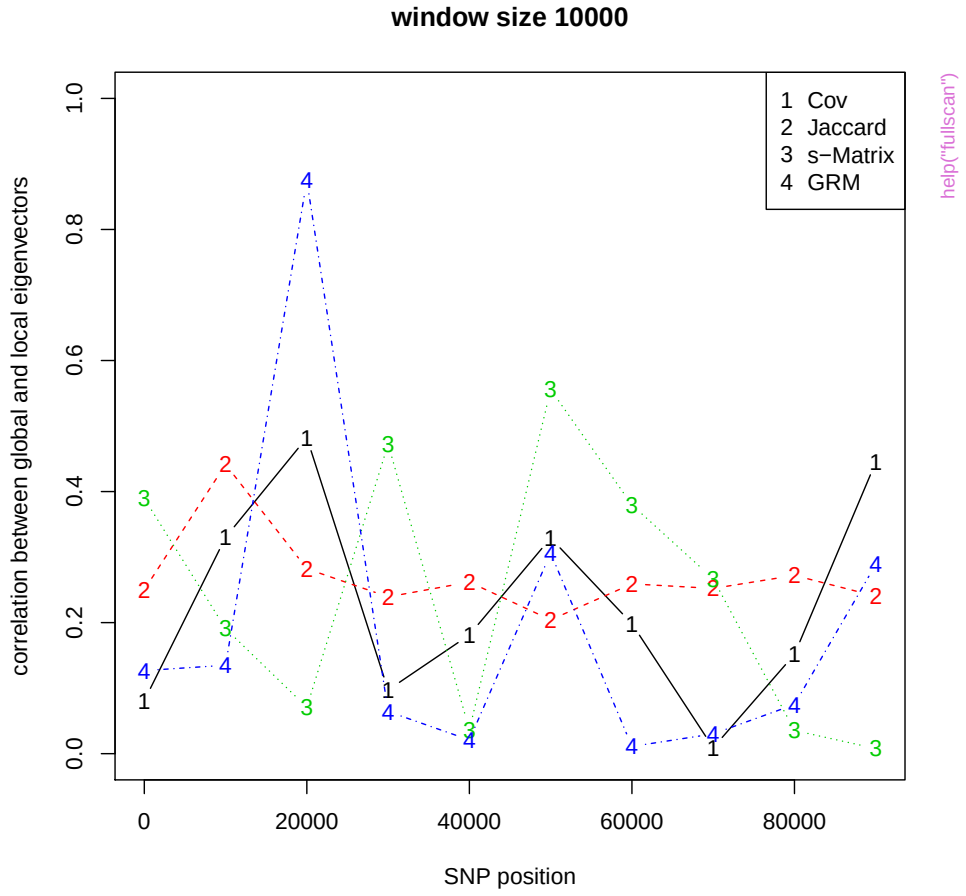
```
fullscan(m, windows, matrixFunction, summaryFunction, comparisonFunction)
```

Arguments

m	A (sparse) matrix for which the full scan is sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
windows	A two-column matrix containing per column the windows on which the data is scanned. The windows can be overlapping. The windows can be computed using the function makeWindows.
matrixFunction	Function on one matrix argument to process the data for each window (e.g., the covariance matrix).
summaryFunction	Function on one argument to summarise the output of the function matrixFunction (e.g., the largest eigenvector).
comparisonFunction	Function on two inputs to compute some kind of comparison measure for the output of the function summaryFunction (e.g., vector correlation, or matrix norm).

Value

A two-column matrix containing per row the global and local summary statistics for each window. Plotting the correlation data of the returned matrix gives a figure analogously to the figure shown here, which was generated with the example code below.



References

Dmitry Prokopenko, Julian Hecker, Edwin Silverman, Marcello Pagano, Markus Noethen, Christian Dina, Christoph Lange and Heide Fier (2016). Utilizing the Jaccard index to reveal population stratification in sequencing data: a simulation study and an application to the 1000 Genomes Project. Bioinformatics, 32(9):1366-1372.

Examples

```
library(locStra)
library(Matrix)
data(testdata)
cor2 <- function(x,y) ifelse(sum(x)==0 | sum(y)==0, 0, cor(x,y))
```

```
windowSize <- 10000
w <- makeWindows(nrow(testdata),windowSize,windowSize)
resCov <- fullscan(testdata,w,covMatrix,powerMethod,cor2)
resJac <- fullscan(testdata,w,jaccardMatrix,powerMethod,cor2)
resSMx <- fullscan(testdata,w,sMatrix,powerMethod,cor2)
resGRM <- fullscan(testdata,w,grMatrix,powerMethod,cor2)
resAll <- cbind(resCov[,1], resJac[,1], resSMx[,1], resGRM[,1])
xlabel <- "SNP position"
ylabel <- "correlation between global and local eigenvectors"
mainlabel <- paste("window size",windowSize)
matplot(w[,1],abs(resAll),type="b",xlab=xlabel,ylab=ylabel,ylim=c(0,1),main=mainlabel)
legend("topright",legend=c("Cov","Jaccard","s-Matrix","GRM"),pch=paste(1:ncol(resAll)))
```

`grMatrix` *Cpp implementation of the genomic relationship matrix (grm) for a (sparse) input matrix as defined in Yang et al. (2011).*

Description

Cpp implementation of the genomic relationship matrix (grm) for a (sparse) input matrix as defined in Yang et al. (2011).

Usage

```
grMatrix(m, dense = FALSE, robust = TRUE)
```

Arguments

`m`	A (sparse) matrix for which the genomic relationship matrix is sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`dense`	Flag to switch between purpose-built dense or sparse implementations. Default is `dense=FALSE`.
`robust`	Flag to indicate if the classic (`robust=FALSE`) or robust (`robust=TRUE`) version of the grm matrix is sought. Default is `robust=TRUE`.

Value

The genomic relationship matrix of `m`.

References

Yang J, Lee SH, Goddard ME, Visscher PM (2011). GCTA: a tool for genome-wide complex trait analysis. Am J Hum Genet, 88(1):76-82.

Examples

```
library(locStra)
library(Matrix)
m <- matrix(sample(0:1,15,replace=TRUE),ncol=3)
sparseM <- Matrix(m,sparse=TRUE)
print(grMatrix(sparseM))
```

`jaccardMatrix` *Cpp implementation of the Jaccard similarity matrix computation for a (sparse) input matrix.*

Description

Cpp implementation of the Jaccard similarity matrix computation for a (sparse) input matrix.

Usage

```
jaccardMatrix(m, dense = FALSE)
```

Arguments

`m`	A (sparse) matrix for which the Jaccard similarity matrix is sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`dense`	Flag to switch between purpose-built dense or sparse implementations. Default is `dense=FALSE`.

Value

The Jaccard matrix of `m`.

References

Dmitry Prokopenko, Julian Hecker, Edwin Silverman, Marcello Pagano, Markus Noethen, Christian Dina, Christoph Lange and Heide Fier (2016). Utilizing the Jaccard index to reveal population stratification in sequencing data: a simulation study and an application to the 1000 Genomes Project. Bioinformatics, 32(9):1366-1372.

Examples

```
library(locStra)
library(Matrix)
m <- matrix(sample(0:1,15,replace=TRUE),ncol=3)
sparseM <- Matrix(m,sparse=TRUE)
print(jaccardMatrix(sparseM))
```

`makeWindows` *Auxiliary function to generate a two-column matrix of window sizes to be used in the function 'fullscan'.*

Description

Auxiliary function to generate a two-column matrix of window sizes to be used in the function 'fullscan'.

Usage

```
makeWindows(len, size, offset)
```

Arguments

`len`	The overall length of the data which is to be scanned in windows.
`size`	The window size.
`offset`	The offset of the generated windows (e.g., if `offset=1` then sliding window, if `offset=size` then blocks).

Value

A two-column matrix of sliding windows, with one window per row defined through start and end value.

Examples

```
library(locStra)
print(makeWindows(100,10,5))
```

`powerMethod` *Cpp implementation of the power method (von Mises iteration) to compute the largest eigenvector for a (sparse) input matrix.*

Description

Cpp implementation of the power method (von Mises iteration) to compute the largest eigenvector for a (sparse) input matrix.

Usage

```
powerMethod(m, initvector = 0)
```

Arguments

m	Symmetric matrix for which the largest eigenvector is sought.
initvector	Optional vector compatible with the input matrix which serves as a starting value for the iteration. Default is zero.

Value

The largest eigenvector of m.

References

Richard von Mises and Hilda Pollaczek-Geiringer (1929). Praktische Verfahren der Gleichungsaufloesung. ZAMM Zeitschrift fuer Angewandte Mathematik und Mechanik, 9:152-164.

Examples

```
library(locStra)
m <- matrix(1:9,3)
print(powerMethod(m))
```

---

sMatrix *Cpp implementation of the s-matrix function (which computes the weighted Jaccard similarity matrix) for a (sparse) input matrix as in the 'Stego' package on https://github.com/dschlauch/stego.*

Description

Cpp implementation of the s-matrix function (which computes the weighted Jaccard similarity matrix) for a (sparse) input matrix as in the 'Stego' package on https://github.com/dschlauch/stego.

Usage

```
sMatrix(m, dense = FALSE, phased = FALSE, minVariants = 0)
```

Arguments

m	A (sparse) matrix for which the s-matrix is sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
dense	Flag to switch between purpose-built dense or sparse implementations. Default is dense=FALSE.
phased	Boolean flag to indicate if input matrix is phased. Default is phased=FALSE.
minVariants	Integer cutoff value for minimal number of variants. Default is minVariants=0

Value

The s-matrix (the weighted Jaccard matrix) of m.

References

Daniel Schlauch (2016). Implementation of the stego algorithm - Similarity Test for Estimating Genetic Outliers. https://github.com/dschlauch/stego

Examples

```
library(locStra)
library(Matrix)
m <- matrix(sample(0:1,15,replace=TRUE),ncol=3)
sparseM <- Matrix(m,sparse=TRUE)
print(sMatrix(sparseM))
```

---

testdata *Simulated test data.*

Description

An artificial dataset containing 100,000 RVs for 100 subjects (one per column). The dataset was generated by resampling the data for chromosome 1 of the 1,000 Genome Project with replacement for randomly chosen subjects.

Usage

```
data(testdata)
```

Format

A matrix with 100,000 rows and 100 columns.

References

A global reference for human genetic variation, The 1000 Genomes Project Consortium, Nature 526, 68-74 (01 October 2015) doi:10.1038/nature15393.

Index