Package 'locStra'

February 17, 2022

Type Package

Title Fast Implementation of (Local) Population Stratification Methods

Version 1.8

Date 2022-02-17

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

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

Description

Fast implementations to compute the genetic covariance matrix, the Jaccard similarity matrix, the s-matrix (the weighted Jaccard similarity matrix), and the (classic or robust) genomic relationship matrix of a (dense or sparse) input matrix (see Hahn, Lutz, Hecker, Prokopenko, Cho, Silverman, Weiss, and Lange (2020) <doi:10.1002/gepi.22356>). Full support for sparse matrices from the R-package 'Matrix'. Additionally, an implementation of the power method (von Mises iteration) to compute the largest eigenvector of a matrix is included, a function to perform an automated full run of global and local correlations in population stratification data, a function to compute sliding windows, and a function to invert minor alleles and to select those variants/loci exceeding a minimal cut-off value. New functionality in locStra allows one to extract the k leading eigenvectors of the genetic covariance matrix, Jaccard similarity matrix, s-matrix, and genomic relationship matrix without actually computing the similarity matrices.

License GPL (>= 2)

Imports Rcpp (>= 0.12.13), Rdpack, Matrix, RSpectra

RdMacros Rdpack

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-02-17 18:12:11 UTC

R topics documented:

covMatrix 2

fastCovEVs . . . 3
fastGrmEVs . . . 4
fastJaccardEVs . . . 5
fastSMatrixEVs . . . 6
fullscan . . . 7
grMatrix . . . 10
jaccardMatrix . . . 11
makeWindows . . . 12
powerMethod . . . 12
selectVariants . . . 13
sMatrix . . . 14
testdata . . . 15

Index **16**

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

Description

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

Usage

```
covMatrix(m, useCpp = TRUE, sparse = TRUE)
```

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.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.

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

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

`fastCovEVs` *Computation of the k leading eigenvectors of the covariance matrix for a (sparse) input matrix.*

Description

Computation of the k leading eigenvectors of the covariance matrix for a (sparse) input matrix.

Usage

```
fastCovEVs(m, k, useCpp = TRUE, sparse = TRUE, q = 2)
```

Arguments

`m`	A (sparse) matrix for which the eigenvectors of its covariance matrix are sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`k`	The number of leading eigenvectors.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.
`q`	The number of power iteration steps (default is `q=2`).

Value

The k leading eigenvectors of the covariance matrix of `m` as a column matrix.

References

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

N. Halko, P.G. Martinsson, and J.A. Tropp (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review: 53(2), pp. 217–288.

Examples

```
require(locStra)
require(Matrix)
m <- matrix(sample(0:1,100,replace=TRUE),ncol=5)
sparseM <- Matrix(m,sparse=TRUE)
print(fastCovEVs(sparseM,k=2,useCpp=FALSE))
```

`fastGrmEVs` *Computation of the k leading eigenvectors of the genomic relationship matrix, defined in Yang et al. (2011), for a (sparse) input matrix.*

Description

Computation of the k leading eigenvectors of the genomic relationship matrix, defined in Yang et al. (2011), for a (sparse) input matrix.

Usage

```
fastGrmEVs(m, k, useCpp = TRUE, sparse = TRUE, robust = TRUE, q = 2)
```

Arguments

`m`	A (sparse) matrix for which the eigenvectors of its genomic relationship matrix are sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`k`	The number of leading eigenvectors.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.
`robust`	Flag to indicate if the classic (`robust=FALSE`) or robust (`robust=TRUE`) version of the genomic relationship matrix is desired. Default is `robust=TRUE`.
`q`	The number of power iteration steps (default is `q=2`).

Value

The k leading eigenvectors of the genomic relationship matrix of `m` as a column matrix.

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.

N. Halko, P.G. Martinsson, and J.A. Tropp (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review: 53(2), pp. 217–288.

Examples

```
require(locStra)
require(Matrix)
m <- matrix(sample(0:1,100,replace=TRUE),ncol=5)
sparseM <- Matrix(m,sparse=TRUE)
print(fastGrmEVs(sparseM,k=2,useCpp=FALSE))
```

`fastJaccardEVs` *Computation of the k leading eigenvectors of the Jaccard similarity matrix for a (sparse) input matrix. Note that this computation is only approximate and does not necessarily coincide with the result obtained by extracting the k leading eigenvectors of the Jaccard matrix computed with the function* `jaccardMatrix`.

Description

Computation of the k leading eigenvectors of the Jaccard similarity matrix for a (sparse) input matrix. Note that this computation is only approximate and does not necessarily coincide with the result obtained by extracting the k leading eigenvectors of the Jaccard matrix computed with the function `jaccardMatrix`.

Usage

```
fastJaccardEVs(m, k, useCpp = TRUE, sparse = TRUE, q = 2)
```

Arguments

`m`	A (sparse) matrix for which the eigenvectors of its Jaccard matrix are sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`k`	The number of leading eigenvectors.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.
`q`	The number of power iteration steps (default is `q=2`).

Value

The k leading eigenvectors of the Jaccard matrix of `m` as a column matrix.

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.

N. Halko, P.G. Martinsson, and J.A. Tropp (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review: 53(2), pp. 217–288.

Examples

```
require(locStra)
require(Matrix)
m <- matrix(sample(0:1,100,replace=TRUE),ncol=5)
sparseM <- Matrix(m,sparse=TRUE)
print(fastJaccardEVs(sparseM,k=2,useCpp=FALSE))
```

`fastSMatrixEVs` *Computation of the k leading eigenvectors of the s-matrix (the weighted Jaccard similarity matrix) for a (sparse) input matrix. Note that in contrast to the parameters of the function* `sMatrix`*, the choice* `phased=FALSE` *cannot be modified for the fast eigenvector computation.*

Description

Computation of the k leading eigenvectors of the s-matrix (the weighted Jaccard similarity matrix) for a (sparse) input matrix. Note that in contrast to the parameters of the function `sMatrix`, the choice `phased=FALSE` cannot be modified for the fast eigenvector computation.

Usage

```
fastSMatrixEVs(m, k, useCpp = TRUE, sparse = TRUE, Djac = FALSE, q = 2)
```

Arguments

`m`	A (sparse) matrix for which the eigenvectors of its s-matrix are sought. The input matrix is assumed to be oriented to contain the data for one individual per column.
`k`	The number of leading eigenvectors.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.
`Djac`	Flag to switch between the unweighted (`Djac=TRUE`) or weighted (`Djac=FALSE`) version. Default is `Djac=FALSE`.
`q`	The number of power iteration steps (default is `q=2`).

Value

The k leading eigenvectors of the s-matrix of m as a column matrix.

References

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

N. Halko, P.G. Martinsson, and J.A. Tropp (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review: 53(2), pp. 217–288.

Examples

```
require(locStra)
require(Matrix)
m <- matrix(sample(0:1,100,replace=TRUE),ncol=5)
sparseM <- Matrix(m,sparse=TRUE)
print(fastSMatrixEVs(sparseM,k=2,useCpp=FALSE))
```

---

fullscan — *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 summarized using the function* summaryFunction *(e.g., the largest eigenvector computed with the function* powerMethod*), and finally the global and local summaries 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 summary statistics (e.g., the correlation between the global and local eigenvectors) and the local summary statistics (e.g., the correlation between the local eigenvectors of the previous and current windows) for each window.*

Description

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 summarized using the function summaryFunction (e.g., the largest eigenvector computed with the function powerMethod), and finally the global and local summaries 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 summary statistics (e.g., the correlation between the global and local eigenvectors) and the local summary statistics (e.g., the correlation between the local eigenvectors of the previous and current windows) for each window.

Usage

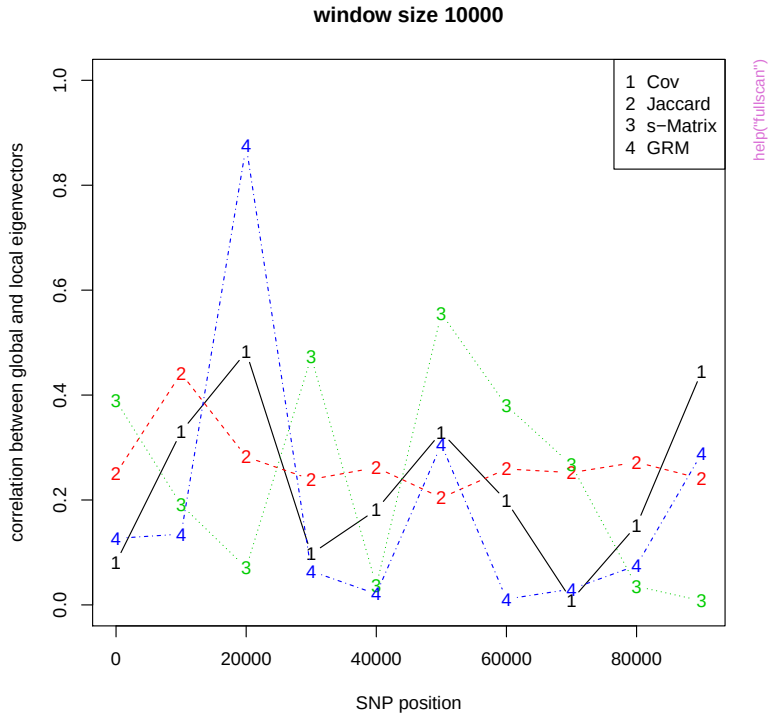
```
fullscan(m, windows, matrixFunction, summaryFunction, comparisonFunction)
```

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 summarize the output of the function `matrixFunction` (e.g., the largest eigenvector).
`comparisonFunction`	Function on two inputs to compute a 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

```
require(locStra)
require(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 *C++ implementation to compute the genomic relationship matrix (grm) for a (sparse) input matrix as defined in Yang et al. (2011).*

Description

C++ implementation to compute the genomic relationship matrix (grm) for a (sparse) input matrix as defined in Yang et al. (2011).

Usage

```
grMatrix(m, useCpp = TRUE, sparse = TRUE, 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.
useCpp	Flag to switch between R or C++ implementations. Default is useCpp=TRUE.
sparse	Flag to switch between purpose-built dense or sparse implementations. Default is sparse=TRUE.
robust	Flag to indicate if the classic (robust=FALSE) or robust (robust=TRUE) version of the genomic relationship matrix is desired. 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

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

`jaccardMatrix` *C++ implementation to compute the Jaccard similarity matrix for a (sparse) input matrix.*

Description

C++ implementation to compute the Jaccard similarity matrix for a (sparse) input matrix.

Usage

```
jaccardMatrix(m, useCpp = TRUE, sparse = TRUE)
```

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.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.

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

```
require(locStra)
require(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 windows to be used in the function 'fullscan'.*

Description

Auxiliary function to generate a two-column matrix of windows 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

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

`powerMethod` *C++ implementation of the power method (von Mises iteration) to compute the largest eigenvector of a dense input matrix.*

Description

C++ implementation of the power method (von Mises iteration) to compute the largest eigenvector of a dense 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

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

---

selectVariants *Auxiliary function to invert minor alleles and to select those variants/loci exceeding a minimal cutoff value.*

Description

Auxiliary function to invert minor alleles and to select those variants/loci exceeding a minimal cutoff value.

Usage

```
selectVariants(m, phased = FALSE, invertMinorAllele = TRUE, minVariants = 0)
```

Arguments

m	A (sparse) input matrix. The input matrix is assumed to be oriented to contain the data for one individual per column.
phased	Boolean flag to indicate if the input matrix is phased. Default is phased=FALSE.
invertMinorAllele	Boolean flag to indicate if the minor allele should be inverted. Default is invertMinorAllele=TRUE.
minVariants	Cutoff value for minimal number of variants for keeping a locus. Default is minVariants=0.

Value

The processed matrix with pruned variants/loci.

Examples

```
require(locStra)
m <- matrix(sample(0:1,100,replace=TRUE),ncol=10)
print(selectVariants(m))
```

`sMatrix` *C++ implementation to compute the s-matrix (the weighted Jaccard similarity matrix) for a (sparse) input matrix as in the 'Stego' package: https://github.com/dschlauch/stego*

Description

C++ implementation to compute the s-matrix (the weighted Jaccard similarity matrix) for a (sparse) input matrix as in the 'Stego' package: https://github.com/dschlauch/stego

Usage

```
sMatrix(m, useCpp = TRUE, sparse = TRUE, Djac = FALSE, phased = FALSE)
```

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.
`useCpp`	Flag to switch between R or C++ implementations. Default is `useCpp=TRUE`.
`sparse`	Flag to switch between purpose-built dense or sparse implementations. Default is `sparse=TRUE`.
`Djac`	Flag to switch between the unweighted (`Djac=TRUE`) or weighted (`Djac=FALSE`) version. Default is `Djac=FALSE`.
`phased`	Boolean flag to indicate if the input matrix is phased. Default is `phased=FALSE`.

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

```
require(locStra)
require(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

∗ **dataset**
testdata, 15

covMatrix, 2

fastCovEVs, 3
fastGrmEVs, 4
fastJaccardEVs, 5
fastSMatrixEVs, 6
fullscan, 7

grMatrix, 10

jaccardMatrix, 11

makeWindows, 12

powerMethod, 12

selectVariants, 13
sMatrix, 14

testdata, 15