Package 'KRIS'

May 7, 2026

Type Package

Title Keen and Reliable Interface Subroutines for Bioinformatic Analysis

Version 1.1.6

Description Provides useful functions which are needed for bioinformatic analysis such as calculating linear principal components from numeric data and Single-nucleotide polymorphism (SNP) dataset, calculating fixation index (Fst) using Hudson method, creating scatter plots in 3 views, handling with PLINK binary file format, detecting rough structures and outliers using unsupervised clustering, and calculating matrix multiplication in the faster way for big data.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports rARPACK,grDevices,graphics,stats,utils

Suggests testthat

BugReports https://gitlab.com/kris.ccp/kris/-/issues

URL https://gitlab.com/kris.ccp/kris

NeedsCompilation no

Author Kridsadakorn Chaichoompu [aut, cre],
Kristel Van Steen [aut],
Fentaw Abegaz [aut],
Sissades Tongsima [aut],
Philip James Shaw [aut],
Anavaj Sakuntabhai [aut],
Luisa Pereira [aut]

Maintainer Kridsadakorn Chaichoompu <kridsadakorn@biostatgen.org>

Repository CRAN

Date/Publication 2021-01-21 12:10:02 UTC

Contents

`cal.pc.linear` *Calculate linear principal component analysis (PCA) from numeric data and Single-nucleotide polymorphism (SNP) dataset*

Description

Available for two types of data; numeric data and Single-nucleotide polymorphism (SNP) dataset in additive coding (0, 1, and 2).

Usage

```
cal.pc.linear(X, PCscore = TRUE, no.pc = NA, data.type = "linear", XXT = TRUE)
```

Arguments

`X`	A data matrix which rows represent samples and columns represent features.
`PCscore`	To specify whether scaled PCs will be calculated or not. If FALSE, eigenvectors are returned instead. Default = TRUE.
`no.pc`	A number of PCs to be calculated. If no.pc is set, PCs are patially calculated. Otherwise all PCs are obtained after calculation. Default = NA.
`data.type`	To specify a type of data matrix X. It can be set to "linear" and "snp". Default = "linear".
`XXT`	To specify how pricipal components (PCs) are calculated. If TRUE, PCs are calculated from X.t(X), otherwise X is used directly. XXT is useful option especially an input matrix X contains many columns. Enabling this option, it helps to reduce computation complexity. Regardless the option XXT is enable or not, optained PCs are the same. Default = TRUE.

Value

The returned value is a list with 2 objects, $PC, $evalue:

- $PC is a PC matrix which rows represent samples and columns represent PCs.
- $evalue is a vector of eigen values.

See Also

cal.pc.projection

Examples

```
#Load simulated dataset
data(example_SNP)

#Using default parameters
PCs <- cal.pc.linear(simsnp$snp)
summary(PCs)

#Preview $PC
print(PCs$PC[1:5,1:3])

#Preview $evalue
print(PCs$evalue[1:3])

plot3views(PCs$PC[,1:3], sample_labels)

#Calculate PCs without PC scores

PCs <- cal.pc.linear(simsnp$snp, PCscore = FALSE)
summary(PCs)

#Preview $PC
print(PCs$PC[1:5,1:3])

#Preview $evalue
print(PCs$evalue[1:3])

plot3views(PCs$PC[,1:3], sample_labels)

#Calculate the top 3 PCs
PCs <- cal.pc.linear(simsnp$snp, no.pc = 3)
summary(PCs)

#Preview $PC
print(PCs$PC[1:5,1:3])

#Preview $evalue
print(PCs$evalue[1:3])

plot3views(PCs$PC[,1:3], sample_labels)
```

`cal.pc.projection` *Calculate linear principal component analysis (PCA) with a projection method for Single-nucleotide polymorphism (SNP) dataset.*

Description

In order to perform the projection method, disease status for all individuals are required. First, PCA is performed only in control group, then project the scores from control group into case group.

Usage

```
cal.pc.projection(
  X,
  status,
  individual_id = NULL,
  labels = NULL,
  no.pc = NA,
  data.type = "linear"
)
```

Arguments

`X`	A data matrix which rows represent samples and columns represent features.
`status`	A vector of numbers that contains disease status for all individuals. For control group, the status is "1", and "2" for case group. Individuals with unknown status (other numbers) are ignored and excluded from the result.
`individual_id`	A vector of charactors that contains individuals IDs
`labels`	A vector of charactors that contains labels for all lindividuals
`no.pc`	A number of PCs to be calculated. If no.pc is set, PCs are patially calculated. Otherwise all PCs are obtained after calculation. Default = NA.
`data.type`	To specify a type of data matrix X. It can be set to "linear" and "snp". Default = "linear".

Value

The returned value is a list with 4 objects, `$PC`, `$id`, `$label`,and `$status`. Individuals with unknown status are excluded.

- `$PC` is a PC matrix which rows represent samples and columns represent PCs.
- `$individual_id` is a vector of charactors that contains individuals IDs.
- `$label` is a vector of charactors that contains labels for all lindividuals.
- `$status` is a vector of numbers that contains disease status for all. individuals.

See Also

`cal.pc.linear`

Examples

```
data(example_SNP)

#Create a random list of disease status, 1 = Control and 2 = Case

ind_status <- sample(c(1,2), size = length(sample_labels), replace = TRUE)

PCs <- cal.pc.projection(simsnp$snp, status = ind_status,
labels = sample_labels)
summary(PCs)

#Preview $PC
print(PCs$PC[1:5,1:3])

#Preview $status
print(PCs$status[1:3])

plot3views(PCs$PC[,1:3], PCs$label)

#Calculate the top 3 PCs

PCs <- cal.pc.projection(simsnp$snp, status = ind_status,
labels = sample_labels, no.pc = 3)
summary(PCs)

#Preview $PC
print(PCs$PC[1:5,1:3])

plot3views(PCs$PC[,1:3], PCs$label)
```

`fst.each.snp.hudson` *Calculate the fixation index (Fst) for all SNPs between two groups of individuals from Single-nucleotide polymorphism (SNP)*

Description

Fixation index (Fst) calculation was implemented using Hudson method as in Bhatia (2013) and Hudson (1992).

Fixation index (Fst) calculation was implemented using Hudson method as in Bhatia (2013) and Hudson (1992).

Usage

```
fst.each.snp.hudson(X, idx.p1, idx.p2)

fst.each.snp.hudson(X, idx.p1, idx.p2)
```

Arguments

X	A matrix contains the number 0, 1, and 2 representing SNP in additive coding. Rows represent individuals and columns represent SNP.
idx.p1	An integer vector contains the row indices of first population in the matrix X.
idx.p2	An integer vector contains the row indices of second population in the matrix X.

Value

The function returns a matrix of pairwise Fst values for all SNPs between 2 specified groups.

The function returns a matrix of pairwise Fst values for all SNPs between 2 specified groups.

References

Bhatia, G., Patterson, N., Sankararaman, S., and Price, A.L. (2013). Estimating and interpreting FST: The impact of rare variants. Genome Res. 23, 1514-1521.

Hudson, R.R., Slatkin, M., and Maddison, W.P. (1992). Estimation of levels of gene flow from DNA sequence data. Genetics 132, 583-589.

Bhatia, G., Patterson, N., Sankararaman, S., and Price, A.L. (2013). Estimating and interpreting FST: The impact of rare variants. Genome Res. 23, 1514-1521.

Hudson, R.R., Slatkin, M., and Maddison, W.P. (1992). Estimation of levels of gene flow from DNA sequence data. Genetics 132, 583-589.

See Also

fst.hudson

fst.hudson

Examples

```
#Load simulated dataset
data(example_SNP)

idx1 <- which(sample_labels == 'pop1')
idx2 <- which(sample_labels == 'pop2')
fst.pairwise <- fst.each.snp.hudson(simsnp$snp, idx1, idx2)

#Print out the Fst values of the first three SNPs between 'pop1' and 'pop2'
print(fst.pairwise[1:3])
```

```
#Load simulated dataset
data(example_SNP)

idx1 <- which(sample_labels == 'pop1')
idx2 <- which(sample_labels == 'pop2')
fst.pairwise <- fst.each.snp.hudson(simsnp$snp, idx1, idx2)
```

```
#Print out the Fst values of the first three SNPs between 'pop1' and 'pop2'
print(fst.pairwise[1:3])
```

`fst.hudson` *Calculate the average fixation index (Fst) between two groups of individuals from Single-nucleotide polymorphism (SNP)*

Description

Fixation index (Fst) calculation was implemented using Hudson method as in Bhatia (2013) and Hudson (1992).

Fixation index (Fst) calculation was implemented using Hudson method as in Bhatia (2013) and Hudson (1992).

Usage

```
fst.hudson(X, idx.p1, idx.p2)

fst.hudson(X, idx.p1, idx.p2)
```

Arguments

`X`	A matrix contains the number 0, 1, and 2 representing SNP in additive coding. Rows represent individuals and columns represent SNP.
`idx.p1`	An integer vector contains the row indices of first population in the matrix X.
`idx.p2`	An integer vector contains the row indices of second population in the matrix X.

Value

The function returns an average Fst value between 2 specified groups.

The function returns an average Fst value between 2 specified groups.

References

Bhatia, G., Patterson, N., Sankararaman, S., and Price, A.L. (2013). Estimating and interpreting FST: The impact of rare variants. Genome Res. 23, 1514-1521.

Hudson, R.R., Slatkin, M., and Maddison, W.P. (1992). Estimation of levels of gene flow from DNA sequence data. Genetics 132, 583-589.

Bhatia, G., Patterson, N., Sankararaman, S., and Price, A.L. (2013). Estimating and interpreting FST: The impact of rare variants. Genome Res. 23, 1514-1521.

Hudson, R.R., Slatkin, M., and Maddison, W.P. (1992). Estimation of levels of gene flow from DNA sequence data. Genetics 132, 583-589.

See Also

fst.each.snp.hudson

fst.each.snp.hudson

Examples

```
#Load simulated dataset
data(example_SNP)

idx1 <- which(sample_labels == 'pop1')
idx2 <- which(sample_labels == 'pop2')
fst <- fst.hudson(simsnp$snp, idx1, idx2)

#Print out the Fst value between 'pop1' and 'pop2'
print(fst)
```

```
#Load simulated dataset
data(example_SNP)

idx1 <- which(sample_labels == 'pop1')
idx2 <- which(sample_labels == 'pop2')
fst <- fst.hudson(simsnp$snp, idx1, idx2)

#Print out the Fst value between 'pop1' and 'pop2'
print(fst)
```

---

plot3views *Create scatter plots in three views.*

Description

Visualize data in X-Y plane, X-Z plane, and Y-Z plane. The input object (matrix or data.frame) must contain at least 3 columns.

Usage

```
plot3views(
  X,
  labels,
  only.row = NA,
  plot.legend = NA,
  plot.pattern = NA,
  plot.color = NA
)
```

Arguments

X	A matrix or a data.frame that contains at least 3 columns of numeric data. If there are more than 3 columns in X, only the first 3 columns will be used.
labels	A vector containing row labels of X for display. All vector elements should be of type "character" (as.character). The length of vector equals the number of rows in X.
only.row	A vector that contains subset of row numbers that are selected to be plotted. Default = NA.
plot.legend	A vector of characters representing legends, also see the note below. Default = NA.
plot.pattern	A vector of characters or integer representing patterns, also see the note below. Default = NA.
plot.color	A vector of characters or integer representing colors, also see the note below. Default = NA.

Details

Note that the vectors of plot.legend, plot.pattern, and plot.color need to be defined as the same length. All of these vectors need to be given to the function otherwise the default colors and patterns will be used. The vectors need to be set properly, see the section "Examples" for more details.

From version 1.1.5 onward, the parameter 'col.pat.table' is removed out from the function.

Examples

```
#Load simulated dataset
data(example_SNP)

PCs <- cal.pc.linear(simsnp$snp, no.pc = 3)
plot3views( PCs$PC, sample_labels)

#To change colors and patterns using symbols
all.labels <- unique(sample_labels)
my.colors <- c('pink', 'yellow', 'cyan', 'green')
my.patterns <- c(0,1,2,3)
plot3views(PCs$PC, labels = sample_labels, plot.legend = all.labels,
plot.pattern = my.patterns, plot.color = my.colors)

#To change patterns using characters
my.patterns <- c('o', 'x', '&', '#')
#To change colors using Hex code
my.colors <- c('#E74C3C', '#8E44AD', '#2ECC71', '#E67E22')
plot3views(PCs$PC, labels = sample_labels, plot.legend = all.labels,
plot.pattern = my.patterns, plot.color = my.colors)
```

read.bed *Read the binary PLINK format (BED, BIM, and FAM)*

Description

Require the complete set of 3 files in the binary PLINK format. It includes BED file, BIM file and BAM file. For more information about the binary PLINK format, please check in the manual of PLINK.

Usage

```
read.bed(bed, bim, fam, only.snp = FALSE)
```

Arguments

bed	A path of BED file
bim	A path of BIM file
fam	A path of FAM file
only.snp	If TRUE, the function to read only SNP matrix, otherwise all files are loaded. The default value is FALSE.

Details

For more details about the binary PLINK format, please check http://zzz.bwh.harvard.edu/plink/binary.shtml

Value

The list containing the matrices of $snp, $snp.info, and $ind.info.

- $snp is a SNP matrix from BED file.
- $snp.info is a data.frame of SNP information from BIM file.
- $ind.info is a data.frame of individual information from FAM file.

See Also

write.bed

Examples

```
#Use the example files embedded in the package.
bed <- system.file("extdata", "example_SNP.bed", package="KRIS")
bim <- system.file("extdata", "example_SNP.bim", package="KRIS")
fam <- system.file("extdata", "example_SNP.fam", package="KRIS")
snp <- read.bed(bed, bim, fam )

#Check the objects inside 'snp'
```

```
ls(snp)

#Preview $snp
print(snp$snp[1:10, 1:10])

#Preview $snp.info
head(snp$snp.info)

#Preview $ind.info
head(snp$ind.info)
```

`replace.missing` *(Internal) Replace missing values with other values,internally used for parallelization*

Description

(Internal) Replace missing values with other values,internally used for parallelization

Usage

```
replace.missing(X, missing = NA, rep)
```

Arguments

`X`	An input vector
`missing`	A charactors representing a missing value
`rep`	A vector of new values to replace missing values

Value

A vector with replaced values

`rubikclust` *Unsupervised clustering to detect rough structures and outliers.*

Description

Handle and operate on Nx3 matrix, where N is the number of samples and data are collected on 3 variables.

Usage

```
rubikclust(X, min.space = 0.4, rotation = TRUE)
```

Arguments

X	A data matrix for which rows represent samples and the 3 columns represent features. Missingness is not allowed.
min.space	A value to specify a minimum space between 2 consecutive projected values. Default = 0.4.
rotation	To specify if rotation is enabled or not. Default = TRUE.

Details

The function rubikClust is able to take up to 3 variables (N x 3 matrix). In case, a matrix contains more than 3 columns, only the first three columns are used; the other columns are ignored.

Value

The returned value is a vector of numbers representing cluster memberships.

Examples

```
#Load simulated dataset
data(example_SNP)

PCs <- cal.pc.linear(simsnp$snp, no.pc = 3)

#Run rubikclust with the default parameters
groups <- rubikclust(PCs$PC)
#Check clustering results
print(groups)

#Check cluster's distribution
table(groups)

#Check the plot, highlight the points according to the clustering result
mylabels <- paste0("group", as.factor(groups))
plot3views( PCs$PC, labels = mylabels)

#Run rubikclust with min.space = 0.02
groups <- rubikclust(PCs$PC, min.space = 0.02)
#Check clustering results
print(groups)

#Check cluster's distribution
table(groups)

#Check the plot, highlight the points according to the clustering result
mylabels <- paste0("group", as.factor(groups))
plot3views( PCs$PC, labels = mylabels)
```

sample_labels *Synthetic dataset containing population labels for the dataset simsnp.*

Description

A dataset contains a character vector of 753 elements containing labels or populations of 753 individuals which they belong. Three populations and outliers were labeled as "pop1", "pop2", "pop3", and "outlier".

Usage

```
data(example_SNP)
```

Format

A vector with 753 elements.

See Also

simsnp

simsnp *Synthetic dataset containing single nucleotide polymorphisms (SNP)*

Description

The simsnp is the simulated dataset which consists of 3,000 independent SNPs and 753 individuals belonging to one of three populations (250 individuals each) and 3 outlying individuals. The pairwise genetic distance between populations was set to Fst=0.01 as in Balding (1995).

Usage

```
data(example_SNP)
```

Format

A list with 3 objects

Details

ind.info A character matrix of 753 rows and 6 columns representing individuals and individual information respectively. The columns of ind.info represents sample_ID, family_ID, father_ID, mother_ID, gender, and phenotype respectively.

snp.info A character matrix of 3,000 rows and 6 columns representing SNPs and SNP information respectively. The columns of snp.info represents SNP_CHR (chromosome), SNP_ID, centimorgan, position, allele1, and allele2 respectively.

snp A numeric matrix of 753 rows and 3,000 columns representing individuals and SNPs respectively. The matrix snp contains the number 0, 1, and 2 representing SNP in additive coding.

References

Balding, D.J., and Nichols, R.A. (1995). A method for quantifying differentiation between populations at multi-allelic loci and its implications for investigating identity and paternity. Genetica 96, 3-12.

See Also

sample_labels

`write.bed` *Write a list of SNP object to the binary PLINK format (BED, BIM, and FAM)*

Description

Write a SNP object to the files in the binary PLINK format. For more information about the binary PLINK format, please check in the manual of PLINK.

Usage

```
write.bed(object, file)
```

Arguments

`object`	An object of SNP is a list consisting of 3 matrices, see the *Details* section for more details.
`file`	A prefix of output files for BED, BIM and FAM to be saved.

Details

The `object` should contain:

- `$snp` is a SNP matrix from BED file.
- `$snp.info` is a data.frame of SNP information from BIM file.
- `$ind.info` is a data.frame of individual information from FAM file.

For more details about the binary PLINK format, please check http://zzz.bwh.harvard.edu/plink/binary.shtml

Value

`NULL`.

See Also

read.bed

Examples

```
#Load example data
data(example_SNP)

#Save 'simsnp' to the file as defined in 'save.file'
save.file <- file.path(tempdir(),"new_SNP")
write.bed(simsnp , save.file)
```

`xxt` *Calculate matrix multipication between a matrix and its transpose for large data.*

Description

Calculate matrix multiplication using "divide and conquer technique", which accelerates the computation to be faster.

Usage

```
xxt(X, window.size = 5)
```

Arguments

`X`	An input matrix to be processed.
`window.size`	The window size of matrices to be devided. The default value is 5.

Value

The multiplication matrix of `X.t(X)`.

Examples

```
#Use the example files embedded in the package.
X <-matrix(runif(100), ncol=20)
R1  <- xxt(X)

#Show the result (R1)
print(R1)
R2 <- X %*% t(X)

#Show the result (R2)
print(R2)
```

Index