Package ‘randPedPCA’

April 24, 2025

Type Package

Title Fast PCA for Large Pedigrees

Version 1.0.1

Description Carry out principal component analysis (PCA) of very large pedigrees such as found in breeding populations! This package, 'randPedPCA', exploits sparse matrices and randomised linear algebra to deliver a gazillion-times speed-up compared to naive singular value decoposition (SVD) (and eigen decomposition).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends spam, Matrix, methods, pedigreeTools, R (>= 3.5)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), rgl

RoxygenNote 7.3.2

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Hanbin Lee [aut],
Hannes Becher [cre],
Ros Craddock [aut],
Gregor Gorjanc [aut]

Maintainer Hannes Becher <`h.becher@ed.ac.uk`>

Repository CRAN

Date/Publication 2025-04-24 11:50:06 UTC

Contents

`dspc` *Add downsampling index to rppca object*

Description

This index is used by plot.rppca to downsample the col (colour) values. It is stored in the rppca object's ds slot.

Usage

```
dspc(pc, to = 10000)
```

Arguments

`pc`	an object of class rppca
`to`	The down-sampling parameter. A numeric > 0 or a vector or NA. Interpreted as a proportion or integer or a index vector, see details.

Details

The parameter `to` is used to specify and possibly which individuals are sampled. If NA, all individuals are retained. If `to` is of length one and is between 0 and 1, then it is interpreted as a proportion. If it is greater than 1, it is taken to be the number of individuals to be sampled (possibly rounded by `sample.int`). If `to` is a logical or an integer vector, it is used for logical or integer indexing, respectively. The integer indices of the sample individuals are written to the `ds` slot. If `ds` exists, it is overwritten with a warning.

Value

An (invisible) object of class `rppca` with a slot `ds` added.

`getNumVectorsHutchinson` *Compute the number of vectors to use for Hutchinson trace estimation*

Description

Follows Skorski, M. (2021). Modern Analysis of Hutchinson's Trace Estimator. 2021 55th Annual Conference on Information Sciences and Systems (CISS), 1–5. https://doi.org/10.1109/CISS50987.2021.9400306

Usage

```
getNumVectorsHutchinson(e, d)
```

Arguments

`e`	A `numeric` denoting the relative error margin
`d`	A `numeric`. 1-d is the probability the the relative error is bounded by e.

Value

a scalar

`hutchpp` *Hutch++ trace estimation*

Description

Hutch++ trace estimation

Usage

```
hutchpp(
  B,
  num_queries = 10,
  sketch_frac = 2/3,
  center = FALSE,
  oraculum = oraculumLi
)
```

Arguments

`B`	An object related to the matrix A for which the trace is to be estimated
`num_queries`	Number of random vectors to draw
`sketch_frac`	Hutch++ detail
`center`	Whether or not to implicitly centre
`oraculum`	The oracle function to use

Details

The Hutch++ algorithm (Meyer et al. 2021, https://doi.org/10.48550/arXiv.2010.09649) estimates the trace of a matrix A by evaluating matrix vector products of A and (sub-gaussian) random vectors. This is used on a matrix B which is related to A through some function. The oracle function has to be chosen so that oracle(B, G) returns the product A the oracle function is set to work on a pedigree's L inverse matrix. But this implementation is general and should work - given a custom oracle function - on other input too.

In the context of pedigree PCA, this is used to estimate the trace of an (implicitly) centred additive relationship matrix.

There logical parameter center allows for a pedigree's L matrix to be (implicitly) centred. This is important because centring changes the total variance of the data and thus the trace of A.

Value

An estimate of A's trace - numeric

Examples

```
hutchpp(pedLInv)
hutchpp(pedLInv, center=TRUE)
```

`importLinv` *Generate spam object from L inverse file*

Description

RandPedPCA relies in the `spam` onject format. But matrices are commonly stored in other formats.

Usage

```
importLinv(pth)
```

Arguments

`pth`	path to matrix market file for L inverse matrix in dgTMatrix format

Value

A `spam` sparse matrix

`pedLInv` *Example pedigree L inverse matrix*

Description

An L inverse matrix generated from an AlphaSimR simulation of 20 generations. An original population splits into sub-populations A and B. After a number of generations, crossbreeding starts.

Usage

```
pedLInv
```

Format

‘pedLInv‘ Matrix object of class ’spam’ of dimension 2650x2650, with 7750 (row-wise) nonzero elements. Density of the matrix is 0.11 Class ’spam’ (32-bit)

Source

Simulation

`pedLInv2` *Example pedigree L inverse matrix*

Description

An L inverse matrix generated from an AlphaSimR simulation of 20 generations. Two diverged populations A and B. After a number of generations, crossbreeding starts.

Usage

```
pedLInv2
```

Format

‘pedLInv2‘ Matrix object of class ’spam’ of dimension 2650x2650, with 7750 (row-wise) nonzero elements. Density of the matrix is 0.11 Class ’spam’ (32-bit)

Source

Simulation

`pedLInv4` *Example pedigree L inverse matrix*

Description

An L inverse matrix generated from an AlphaSimR simulation of 20 generations. One population, ABCD, is split into four, A, B, C, D.

Usage

`pedLInv4`

Format

‘pedLInv4‘ Matrix object of class ’spam’ of dimension 4200x4200, with 12200 (row-wise) nonzero elements. Density of the matrix is 0.0692 Class ’spam’ (32-bit)

Source

Simulation

`pedMeta` *Metadata associated with the example pedigree*

Description

A dataframe.

Usage

`pedMeta`

Format

‘pedMeta‘ A ’data.frame’ of 2650 individuals (rows) with 12 variables (cols):

id Integer individual ID

population Population code. A, B or AB

generation Generation of the individual

mid dam ID

fid sire ID

gv1 genetic value

pv1 phenotypic value

gv2 genetic value

pv2 phenotypic value
gv genetic value
pv phenotypic value
generationPlotShift for plotting

Source

Simulation

`pedMeta2`	*#' Example genotype data #' #' An genotype matrix (0/1/2) generated from an AlphaSimR simulation of 20 generations. #' An original population splits into sub-populations A and B. After a number of #' generations, crossbreeding starts. #' #' #' @format ## 'pedGeno' #' Matrix of 2650 rows (inidividuals) and 11000 columns (SNPs) #' @source Simulation "pedGeno" Metadata associated with the 2nd example pedigree*

Description

A dataframe.

Usage

```
pedMeta2
```

Format

'pedMeta2' A 'data.frame' of 2650 individuals (rows) with 12 variables (cols):

id Integer individual ID
population Population code. A, B or AB
generation Generation of the individual
mid dam ID
fid sire ID
gv1 genetic value
pv1 phenotypic value
gv2 genetic value
pv2 phenotypic value
gv genetic value
pv phenotypic value
generationPlotShift for plotting

Source

Simulation

`pedMeta4` *#' Example genotype data 2 #' #' An genotype matrix (0/1/2) generated from an AlphaSimR simulation of 20 generations. #' Two diverged populations A and B. After a number of #' generations, crossbreeding starts. #' #' #' @format ## ‘pedGeno2‘ #' Matrix of 2650 rows (inidividuals) and 11000 columns (SNPs) #' @source Simulation "pedGeno2" Metadata associated with the 2nd example pedigree*

Description

A dataframe.

Usage

`pedMeta4`

Format

‘pedMeta4‘ A ’data.frame’ of 4200 individuals (rows) with 9 variables (cols):

id Integer individual ID

population Population code. A, B, C, D, or ABCD

generation Generation of the individual

mid dam ID

fid sire ID

gv1 genetic value

pv1 phenotypic value

gv2 genetic value

pv2 phenotypic value

Source

Simulation

`plot3D` *3D plot using rgl*

Description

A simple wrapper around rgl's pot3d function.

Usage

```
plot3D(x, dims = c(1, 2, 3), xlab = NULL, ylab = NULL, zlab = NULL, ...)
```

Arguments

`x`	an rppca object
`dims`	vector of length 3 - indices of the PCs to plot
`xlab`	(optional) x axis label
`ylab`	(optional) yaxis label
`zlab`	(optional) xz axis label
`...`	additional arguments passed to rgl::plot3d

Details

Note, different to `plot.rppca`, which is relatively slow, `plot3D` does not down-sample the principal components and it ignores the `ds` slot of an `rppca` object if present.

Value

No return value, called for its side effects.

Examples

```
pc <- rppca(pedLInv)
plot3D(pc)

ped <- pedigree(sire=pedMeta$fid, dam=pedMeta$mid, label=pedMeta$id)
pc2 <- rppca(ped)
plot3D(pc2)
```

`plot3DWithProj` *3D plot of PC scores with projections on coordinate planes*

Description

3D plot of PC scores with projections on coordinate planes

Usage

```
plot3DWithProj(
  pc,
  dims = c(1, 2, 3),
  plotProj = TRUE,
  grid = TRUE,
  col = 1,
  ff = 0.5,
  theta = -45,
  phi = 25
)
```

Arguments

`pc`	An `rppca` object
`dims`	`integer vector`, which PCs to plot, defauts to 1:3
`plotProj`	`logical`, whether to plot the projections
`grid`	`logical`, wheter to plot grids
`col`	the dot colours, integer or string, scalar or vector
`ff`	`numeric`, offset for projection (proportion of the orthogonal axis's range)
`theta, phi`	polar coordinates in degrees. `theta` rotates round the vertical axis. `phi` rotates round the horizontal axis.

Value

nothing

Examples

```
ped <- pedigree(pedMeta$fid,
pedMeta$mid,
pedMeta$id
)
pc <- rppca(ped)
plot3DWithProj(pc, col=as.numeric(factor(pedMeta$population)))
```

`randRangeFinder` *Generate range matrix for SVD*

Description

Generate range matrix for SVD

Usage

```
randRangeFinder(L, rank, depth, numVectors, cent = FALSE)
```

Arguments

`L`	a pedigree's L inverse matrix in sparse 'spam' format
`rank`	An `integer`, how many principal components to return
`depth`	`integer`, number of iterations for generating the range matrix
`numVectors`	An `integer` > `rank`, to specify the oversampling for the
`cent`	`logical` whether or not to (implicitly) 'centre' the additive relationship matrix, or more precisely, its underlying 'data matrix' L

Value

The range matrix for `randSVD`

`randSVD` *Singular value decomposition in sparse triangular matrix*

Description

Uses randomised linear algebra, see Halko et al. (2010). Singular value decomposition (SVD) decomposes a matrix $X = U\Sigma W^T$

Usage

```
randSVD(L, rank, depth, numVectors, cent = FALSE)
```

Arguments

`L`	a pedigree's L inverse matrix in sparse 'spam' format
`rank`	An `integer`, how many principal components to return
`depth`	`integer`, the number of iterations for generating the range matrix
`numVectors`	An `integer` > `rank` to specify the oversampling for the
`cent`	`logical`, whether or not to (implicitly) centre the additive relationship matrix

Value

A list of three: `u` (=U), `d` (=Sigma), and `v` (=W^T)

`randTraceHutchinson` *Trace estimation for sparse L inverse matrices*

Description

Using Hutchinson's method

Usage

```
randTraceHutchinson(L, numVectors)
```

Arguments

`L`	A pedigree's L inverse matrix
`numVectors`	an `integer` specifying how many random vectors to use If you do not have a good reason to do otherwise, use the function `hutchpp` instead. The higher `numVectors`, the higher the accuracy and the longer the running time. Accuracy can be estimated with the function `getNumVectorsHutchinson`.

Value

a scalar

`rppca` *Fast pedigree PCA using sparse matrices and randomised linear algebra*

Description

Fast pedigree PCA using sparse matrices and randomised linear algebra

Usage

```
rppca(X, ...)

## S3 method for class 'spam'
rppca(
  X,
  method = "randSVD",
  rank = 10,
  depth = 3,
```

```
  numVectors = 15,
  totVar = NULL,
  center = FALSE,
  ...
)

## S3 method for class 'pedigree'
rppca(
  X,
  method = "randSVD",
  rank = 10,
  depth = 3,
  numVectors = 15,
  totVar = NULL,
  center = FALSE,
  ...
)
```

Arguments

`X`	A representation of a pedigree, see Details.
`...`	optional arguments passed to methods
`method`	`string` only randSVD (the default) is implemented
`rank`	`integer` how many principal components to return
`depth`	`integer` number of iterations for generating the range matrix
`numVectors`	`integer` > `rank` to specify the oversampling for the range matrix
`totVar`	`scalar` (optional) the total variance, required for computation of variance proportions when using an L-inverse matrix a input
`center`	`logical` whether or not to (implicitly) centre the additive relationship matrix

Details

The output slots are named like those of R's built in `prcomp` function. Rotation is not returned by default as it is the transpose of the PC scores, which are returned in `x`. `scale` and `center` are set to `FALSE`.

Value

A `list` containing:

`x` the principal components

`sdev` the variance components of each PC. Note that the total variance is not known per se and this these components cannot be used to compute the proportion of the total variance accounted for by each PC. However, if `nVecTraceEst` is specified, `rppca` will estimate the total variance and return variance proportions.

`vProp` the estimated variance proportions accounted for by each PC. Only returned if `totVar` is set.

`scale` always `FALSE`

`center` `logical` indicating whether or not the implicit data matrix was centred

`rotation` the right singular values of the relationship matrix. Only returned if `returnRotation == TRUE`

`varProps` proportion of the total variance explained by each PC. Only returned if starting from a pedigree object without centring, or if `totVar` is supplied.

Examples

```
pc <- rppca(pedLInv)
ped <- pedigree(sire=pedMeta$fid,
                dam=pedMeta$mid,
                label=pedMeta$id
                )
pc2 <- rppca(ped)
```

`sparse2spam` *Convert generic sparse matrix to spam format*

Description

Convert generic sparse matrix to spam format

Usage

```
sparse2spam(sprs)
```

Arguments

`sprs`	A sparse matrix.

Value

A `spam` sparse matrix

Index