Package ‘epca’

Type Package
Title Exploratory Principal Component Analysis
Version 1.0.0
Date 2020-06-20
Description Exploratory principal component analysis for large-scale dataset, including sparse principal component analysis and sparse matrix approximation.

URL https://github.com/fchen365/epca
BugReports https://github.com/fchen365/epca/issues
License GPL-3
Depends R (>= 3.5),
Imports clue, irlba, imager, matlab, Matrix, RSpectra, tidyverse, GPArotation,
Suggests elasticnet, ggcorrplot, rmarkdown, markdown, reshape2, matlapr, knitr, PMA,
VignetteBuilder knitr, rmarkdown
Encoding UTF-8
RoxygenNote 7.1.0
NeedsCompilation no
Author Fan Chen [aut, cre] (<https://orcid.org/0000-0003-4508-6023>)
Maintainer Fan Chen <fan.chen@wisc.edu>
Repository CRAN
Date/Publication 2020-06-26 10:10:06 UTC

R topics documented:

epca-package .......................................................... 2
absmin ................................................................. 3
absmin.criteria ........................................................ 3
cpve ................................................................. 4
epca-package

Description

epca is for comprehending any data matrix that contains low-rank and sparse underlying signals of interest. The package currently features two key tools: (1) sca for sparse principal component analysis and (2) sma for sparse matrix approximation, a two-way data analysis for simultaneously row and column dimensionality reductions.

References

Chen, F. and Rohe K. (2020) "A New Basis for Sparse PCA".
absmin

---

**Absmin Rotation**

**Description**

Given a $p \times k$ matrix $x$, finds the orthogonal matrix (rotation) that minimizes the absmin.criteria.

**Usage**

```r
absmin(L, Tmat = diag(ncol(L)), normalize = FALSE, eps = 1e-05, maxit = 1000L)
```

**Arguments**

- **L**: a matrix or Matrix.
- **Tmat**: matrix, initial rotation matrix.
- **normalize**: logical. Should Kaiser normalization be performed? If so the rows of $x$ are re-scaled to unit length before rotation, and scaled back afterwards.
- **eps**: The tolerance for stopping: the relative change in the sum of singular values.
- **maxit**: integer, maximum number of iteration (default to 1,000).

**Value**

A list with three elements:

- **rotated**: the rotated matrix.
- **rotmat**: the (orthogonal) rotation matrix.
- **n.iter**: the number of iteration taken.

**See Also**

GPArotation::GPForth

--

absmin.criteria

---

**Absmin Criteria**

**Description**

Calculate the absmin criteria. This is a helper function for absmin.

**Usage**

```r
absmin.criteria(L)
```

**Arguments**

- **L**: a matrix or Matrix.
**cpve**  
*Cumulative Proportion of Variance Explained (CPVE)*

**Description**
Calculate the CPVE.

**Usage**
\[
\text{cpve}(\text{mat}, V, \text{is.cov} = \text{FALSE})
\]

**Arguments**
- **mat**
  - matrix or Matrix, the original data matrix \(X\) or \(\text{cov}(X) = \text{crossprod}(X) / (nrow(X) - 1)\).
- **V**
  - matrix, coefficients of linear transformation, e.g., loadings (in PCA).
- **is.cov**
  - logical, whether the input matrix is a covariance matrix or a Gram matrix.

**Value**
a numeric vector of length \(\text{ncol}(V)\), the \(i\)-th value is the CPVE of the first \(i\) columns in \(V\).

**See Also**
pve

**Examples**
```r
## use the "swiss" data
## find two sparse PCs
s.sca <- sca(swiss, 2, gamma = sqrt(ncol(swiss)))
ld <- loadings(s.sca)
cpve(as.matrix(swiss), ld)
```

---

**dist.matrix**  
*Matrix Column Distance*

**Description**
Compute the distance between two matrices. The distance between two matrices is defined as the sum of distances between column pairs. This function matches the columns of two matrices, such that the matrix distance (i.e., the sum of paired column distances) is minimized. This is accomplished by solving an optimization over column permutation. Given two matrices, \(x\) and \(y\), find permutation \(p()\) that minimizes \(\sum_i \text{similarity}(x[,p(i)], y[,i])\), where the \text{similarity}() can be "euclidean" distance, \(1 - \text{"cosine"},\) or "maximum" difference (manhattan distance). The solution is computed by \text{clue::solve_LSAP}().
distance

Usage

dist.matrix(x, y, method = "euclidean")

Arguments

x, y matrix or Matrix, of the same number of rows. The columns of x and y will be scaled to unit length.

method distance measure, "maximum", "cosine", or "euclidean" are implemented.

Value

a list of four components:

dist dist, the distance matrix.

match solve_LSAP, the column matches.

value numeric vector, the distance between pairs of columns.

method character, the distance measure used.

nrow integer, the dimension of the input matrices, i.e., nrow(x).

See Also

clue::solve_LSAP

Examples

x <- diag(4)
y <- x + rnorm(16, sd = 0.05) # add some noise
y = t(t(y) / sqrt(colSums(y ^ 2))) ## normalize the columns
## euclidian distance between column pairs, with minimal matches
dist.matrix(x, y, "euclidean")
exp.frac

Arguments

x matrix or Matrix, of the same number of rows. The columns of x and y will be scaled to unit length.

y matrix or Matrix, of the same number of rows. The columns of x and y will be scaled to unit length.

method distance measure, "maximum", "cosine", or "euclidean" are implemented.

Value

numeric, the distance between two matrices.

Description

Calculate fractional exponent/power, \( a^{\frac{\text{num}}{\text{den}}} \), where a could be negative.

Usage

## S3 method for class 'frac'
exp(a, num, den)

Arguments

a numeric(1), base (could be negative).

num a positive integer, numerator of the exponent.

den a positive integer, denominator of the exponent.

Value

numeric, the evaluated \( a^{\frac{\text{num}}{\text{den}}} \)
**fftshift**

**Description**
Rearrange the matrix so that zero frequency component is in the middle of the matrix. This is similar to the `fftshift` function in MATLAB.

**Usage**
```
fftshift(x)
```

**Arguments**
- `x` a matrix of image.

**References**

---

**hard**

**Hard-thresholding**

**Description**
Perform hard-thresholding given the cut-off value.

**Usage**
```
hard(x, t)
```

**Arguments**
- `x` any numerical matrix or vector.
- `t` numeric, the amount to hard-threshold, i.e., $sgn(x_{ij})(|x_{ij} - t|)_+$. 
inner

Matrix Inner Product

Description

Calculate the custom matrix inner product, $Z = \text{crossprod}(X,Y)$, where $Z[i,j] = \text{FUN}(X[i], Y[j])$.

Usage

inner(X, Y, FUN = "crossprod", ...)

Arguments

- **X, Y**: matrix or Matrix.
- **FUN**: function or a character(1) name of base function. The function should take in two vectors as input and output a numeric(1) result.
- **...**: additional parameters for FUN.

Value

matrix, inner product of X and Y.

Examples

```r
x <- matrix(1:6, 2, 3)
y <- matrix(7:12, 2, 3)
## The default is equivalent to `crossprod(x, y)`
inner(x, y)
## We can compute the pair-wise Euclidean distance of columns.
EuclideanDistance = function(x, y) crossprod(x, y)^2
inner(x, y, EuclideanDistance)
```

labelCluster

Label Cluster

Description

Assign cluster labels to each row from the membership matrix.

Usage

labelCluster(x, ties.method = "random")
misClustRate

Arguments

x  matrix with non-negative entries, where x[i,j] is the estimated likelihood (or any equivalent measure) of node i belongs to block j. The higher the more likely.

ties.method  character, how should ties be handled, "random", "first", "last" are allowed. See base::rank() for details.

Value

integer vector of the same length as x. Each entry is one of 1, 2, ..., ncol(x).

misClustRate  Mis-Classification Rate (MCR)

Description

Compute the empirical MCR, assuming that #cluster = #block. This calculation allows a permutation on clusters.

Usage

misClustRate(cluster, truth)

Arguments

cluster  vector of integer or factor, estimated cluster membership.

truth  a vector of the same length as clusters, the true cluster labels.

Value

numeric, the MCR.

Examples

truth = rep(1:3, each = 30)
cluster = rep(3:1, times = c(25, 32, 33))
misClustRate(cluster, truth)
**norm.Lp**

*Element-wise Matrix Norm*

**Description**

Compute element-wise matrix Lp-norm. This is a helper function to `shrinkage()`.

**Usage**

```r
norm.Lp(mat, p = 1)
```

**Arguments**

- `mat`: a matrix or `Matrix`.
- `p`: numeric(1), the p for defining the Lp norm.

**Value**

numeric(1), the absolute sum of all elements.

---

**permColumn**

*Permute columns of a block membership matrix*

**Description**

Perform column permutation of block membership matrix for aesthetic visualization. That is, the k-th column gives k-th cluster. This is done by ranking the column sums of squares (by default).

**Usage**

```r
permColumn(x, s = 2)
```

**Arguments**

- `x`: a non-negative matrix, nNode x nBlock,
- `s`: integer, order of non-linear
pitprops

Pitprops correlation data

Description

The pitprops data is a correlation matrix that was calculated from 180 observations. There are 13 explanatory variables. Jeffers (1967) tried to interpret the first six PCs. This is a classical example showing the difficulty of interpreting principal components.

References


Examples

```r
## NOT TEST
data(pitprops)
ggcorrplot::ggcorrplot(pitprops)
```

polar

Polar Decomposition

Description

Perform the polar decomposition of an n x p (n > p) matrix X into U P, where U is an n x p matrix with orthogonal columns (i.e. `crossprod(U)` is the identity matrix), and P is a p x p positive-semidefinite Hermitian matrix. The function returns the U matrix. This is a helper function of `prs()`.

Usage

```r
polar(x)
```

Arguments

- `x` a matrix or `Matrix`, which is presumed full-rank.

Value

A matrix of the unitary part of the polar decomposition.
References

Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

Examples

```r
x <- matrix(1:6, nrow = 3)
polar_x <- polar(x)
```

---

**prewhiten**

*Pre-whiten Image*

Description

Mimic/simulate the processing in retina, the lateral geniculate nucleus (LGN), and V1 (1) scale all pixels to constant variance (e.g., 1). (2) combined low-pass/whitening filter

Usage

```r
prewhiten(img)
```

Arguments

- `img` a matrix of square image.

References


---

**print.sca**

*Print SCA*

Description

Print SCA

Usage

```r
## S3 method for class 'sca'
print(x, verbose = FALSE, ...)
```

Arguments

- `x` an sca object.
- `verbose` logical(1), whether to print out loadings.
- `...` additional input to generic `print`. 
**Value**

Print an `sma` object interactively.

---

**Description**

Print SMA

**Usage**

```
## S3 method for class 'sma'
print(x, verbose = FALSE, ...)
```

**Arguments**

- `x` : an `sma` object.
- `verbose` : logical(1), whether to print out loadings.
- `...` : additional input to generic `print`.

**Value**

Print an `sma` object interactively.

---

**prs**

**Polar-Rotate-Shrink**

**Description**

This function is a helper function of `sma()`. It performs polar docomposition, orthogonal rotation, and soft-thresholding shrinkage in order. The three steps together enable sparse estimates of the SMA and SCA.

**Usage**

```
prs(X, Z.hat, gamma, rotate, shrink, normalize, order, flip, epsilon)
```
Arguments

X, Z.hat
the matrix product A <- crossprod(X, Z.hat) is the input. X and Z.hat are separated because the former is additionally used to compute the proportion of variance explained, in the case when order = TRUE.

gamma
numeric, the sparsity parameter.

rotate
character(1), rotation method. Two options are currently available: "varimax" (default) or "absmin" (see details).

shrink
character(1), shrinkage method, either "soft"- (default) or "hard"-thresholding (see details).

normalize
logical, whether to rows normalization should be done before and undone after the rotation (see details).

order
logical, whether to re-order the columns of the estimates (see details).

flip
logical, whether to flip the signs of the columns of estimates such that all columns are positive-skewed (see details).

epsilon
numeric, tolerance of convergence precision (default to 0.00001).

Details

rotate: The rotate option specifies the rotation technique to use. Currently, there are two build-in options—"varimax" and "absmin". The "varimax" rotation maximizes the element-wise L4 norm of the rotated matrix. It is faster and computationally more stable. The "absmin" rotation minimizes the absolute sum of the rotated matrix. It is sharper (as it directly minimizes the L1 norm) but slower and computationally less stable.

shrink: The shrink option specifies the shrinkage operator to use. Currently, there are two build-in options—"soft"- and "hard"-thresholding. The "soft"-thresholding universally reduce all elements and sets the small elements to zeros. The "hard"-thresholding only sets the small elements to zeros.

normalize: The argument normalize gives an indication of if and how any normalization should be done before rotation, and then undone after rotation. If normalize is FALSE (the default) no normalization is done. If normalize is TRUE then Kaiser normalization is done. (So squared row entries of normalized A sum to 1.0. This is sometimes called Horst normalization.) For rotate="absmin", if normalize is a vector of length equal to the number of indicators (i.e., the number of rows of A), then the columns are divided by normalize before rotation and multiplied by normalize after rotation. Also, If normalize is a function then it should take A as an argument and return a vector which is used like the vector above.

order: In PCA (and SVD), the principal components (and the singular vectors) are ordered. For this, we order the sparse components (i.e., the columns of Z or Y) by their explained variance in the data, which is defined as sum((A %*% y)^2), where y is a column of the sparse component. Note: not to be confused with the cumulative proportion of variance explained by Y (and Z), particularly when Y (and Z) is may not be strictly orthogonal.

flip: The argument flip gives an indication of if and the columns of estimated sparse component should be flipped. Note that the estimated (sparse) loadings, i.e., the weights on original variables, are column-wise invariant to a sign flipping. This is because flipping of a principal direction does not influence the amount of the explained variance by the component. If flip=TRUE, then the columns of loadings will be flip accordingly, such that each column is positive-skewed. This means that for each column, the sum of cubic elements (i.e., sum(x^3)) are non-negative.
pve

Value

a matrix of the sparse estimate, of the same dimension as crossprod(X,Z.hat).

References

Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

See Also

sma, sca, polar, rotation, shrinkage

pve(mat, V, is.cov = FALSE)

Arguments

mat matrix or Matrix, the original data matrix X or cov(X) = crossprod(X) / (nrow(X) -1)
V matrix, coefficients of linear transformation, e.g., loadings (in PCA).
is.cov logical, whether the input matrix is a covariance matrix or a Gram matrix.

Value

a numeric value between 0 and 1, the proportion of variance in mat explained by Y.

Examples

## use the "swiss" data
## find two sparse PCs
s.sca <- sca(swiss, 2, gamma = sqrt(ncol(swiss)))
ld <- loadings(s.sca)
pve(as.matrix(swiss), ld)
rootmatrix

*Find root matrix*

**Description**

Find X from the Gram matrix (i.e., `crossprod(X)`).

**Usage**

```r
rootmatrix(x)
```

**Arguments**

- `x`: a symmetric matrix.

---

rotation

*Varimax Rotation*

**Description**

Perform varimax rotation. Flip the signs of columns so that the resulting matrix is positive-skewed.

**Usage**

```r
rotation(
  x,
  rotate = c("varimax", "absmin"),
  normalize = FALSE,
  flip = TRUE,
  eps = 1e-06
)
```

**Arguments**

- `x`: a matrix or `Matrix`.
- `rotate`: character(1), rotation method. Two options are currently available: "varimax" (default) or "absmin" (see details).
- `normalize`: logical, whether to rows normalization should be done before and undone afterward the rotation (see details).
- `flip`: logical, whether to flip the signs of the columns of estimates such that all columns are positive-skewed (see details).
- `eps`: numeric precision tolerance.
Details

rotate: The rotate option specifies the rotation technique to use. Currently, there are two built-in options—"varimax" and "absmin". The "varimax" rotation maximizes the element-wise L4 norm of the rotated matrix. It is faster and computationally more stable. The "absmin" rotation minimizes the absolute sum of the rotated matrix. It is sharper (as it directly minimizes the L1 norm) but slower and computationally less stable.

normalize: The argument normalize gives an indication of if and how any normalization should be done before rotation, and then undone after rotation. If normalize is FALSE (the default) no normalization is done. If normalize is TRUE then Kaiser normalization is done. (So squared row entries of normalized A sum to 1.0. This is sometimes called Horst normalization.) For rotate="absmin", if normalize is a vector of length equal to the number of indicators (i.e., the number of rows of A), then the columns are divided by normalize before rotation and multiplied by normalize after rotation. Also, if normalize is a function then it should take A as an argument and return a vector which is used like the vector above.

flip: The argument flip gives an indication of if and the columns of estimated sparse component should be flipped. Note that the estimated (sparse) loadings, i.e., the weights on original variables, are column-wise invariant to a sign flipping. This is because flipping of a principal direction does not influence the amount of the explained variance by the component. If flip=TRUE, then the columns of loadings will be flip accordingly, such that each column is positive-skewed. This means that for each column, the sum of cubic elements (i.e., sum(x^3)) are non-negative.

Value

the rotated matrix of the same dimension as x.

References

Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

See Also

prs, varimax

Examples

## use the "swiss" data
fa <- factanal(~., 2, data = swiss, rotation = "none")
rotation(loadings(fa))
Usage

sca(
  A,
  k = min(5, dim(A)),
  gamma = NULL,
  is.cov = FALSE,
  rotate = c("varimax", "absmin"),
  shrink = c("soft", "hard"),
  center = TRUE,
  scale = TRUE,
  normalize = FALSE,
  order = TRUE,
  flip = TRUE,
  max.iter = 1000,
  epsilon = 1e-05,
  quiet = TRUE
)

Arguments

A  matrix or Matrix to be analyzed.

k  integer, rank of approximation.

gamma numeric(1), sparsity parameter, default to $\sqrt{pk}$, where n x p is the dimension of A.

is.cov logical, whether the A is a covariance matrix or Gram matrix (i.e., crossprod(X)). This function presumes that A is not covariance matrix.

rotate character(1), rotation method. Two options are currently available: "varimax" (default) or "absmin" (see details).

shrink character(1), shrinkage method, either "soft"-(default) or "hard"-thresholding (see details).

center logical, whether to center columns of A (see scale()).

scale logical, whether to scale columns of A (see scale()).

normalize logical, whether to rows normalization should be done before and undone afterward the rotation (see details).

order logical, whether to re-order the columns of the estimates (see details).

flip logical, whether to flip the signs of the columns of estimates such that all columns are positive-skewed (see details).

max.iter integer, maximum number of iteration (default to 1,000).

epsilon numeric, tolerance of convergence precision (default to 0.00001).

quiet logical, whether to mute the process report (default to TRUE)
Details

**rotate**: The `rotate` option specifies the rotation technique to use. Currently, there are two built-in options—“varimax” and “absmin”. The “varimax” rotation maximizes the element-wise L4 norm of the rotated matrix. It is faster and computationally more stable. The “absmin” rotation minimizes the absolute sum of the rotated matrix. It is sharper (as it directly minimizes the L1 norm) but slower and computationally less stable.

**shrink**: The `shrink` option specifies the shrinkage operator to use. Currently, there are two built-in options—“soft”- and “hard”-thresholding. The “soft”-thresholding universally reduce all elements and sets the small elements to zeros. The “hard”-thresholding only sets the small elements to zeros.

**normalize**: The argument `normalize` gives an indication of if and how any normalization should be done before rotation, and then undone after rotation. If `normalize` is `FALSE` (the default) no normalization is done. If `normalize` is `TRUE` then Kaiser normalization is done. (So squared row entries of normalized `A` sum to 1.0. This is sometimes called Horst normalization.) For `rotate=“absmin”`, if `normalize` is a vector of length equal to the number of indicators (i.e., the number of rows of `A`), then the columns are divided by `normalize` before rotation and multiplied by `normalize` after rotation. Also, if `normalize` is a function then it should take `A` as an argument and return a vector which is used like the vector above.

**order**: In PCA (and SVD), the principal components (and the singular vectors) are ordered. For this, we order the sparse components (i.e., the columns of `Z` or `Y`) by their explained variance in the data, which is defined as `sum((A%*%y)^2)`, where `y` is a column of the sparse component. Note: not to be confused with the cumulative proportion of variance explained by `Y` (and `Z`), particularly when `Y` (and `Z`) is may not be strictly orthogonal.

**flip**: The argument `flip` gives an indication of if and the columns of estimated sparse component should be flipped. Note that the estimated (sparse) loadings, i.e., the weights on original variables, are column-wise invariant to a sign flipping. This is because flipping of a principal direction does not influence the amount of the explained variance by the component. If `flip=TRUE`, then the columns of loadings will be flip accordingly, such that each column is positive-skewed. This means that for each column, the sum of cubic elements (i.e., `sum(x^3)`) are non-negative.

Value

an `sca` object that contains:

- **loadings** matrix, sparse loadings of PCs.
- **scores** an n x k matrix, the component scores.
- **sdev** a numeric vector of length k, standard deviation of each columns of scores. These may not sum to exactly 1 because of a slight loss of orthogonality.
- **pve** a numeric vector of length k, cumulative proportion of variance in `A` explained by the top PCs.
- **center** logical, this records the center parameter.
- **scale** logical, this records the scale parameter.
- **n.iter** integer, number of iteration taken.
- **n.obs** integer, sample size, that is, `nrow(A)`.
References
Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

See Also
sma, prs

Examples

```r
## ------ example 1 ------
## simulate a low-rank data matrix with some additive Gaussian noise
n <- 300
p <- 50
k <- 5  # rank
Z <- shrinkage(polar(matrix(runif(n * k), n, k)), sqrt(n))
B <- diag(5) * 3
Y <- shrinkage(polar(matrix(runif(p * k), p, k)), sqrt(p))
E <- matrix(rnorm(n * p, sd = .01), n, p)
X <- scale(Z %*% B %*% t(Y) + E)

## perform sparse PCA
s.sca <- sca(X, k)
s.sca

## ------ example 2 ------
## use the `pitprops` data from the `elasticnet` package
data(pitprops)

## find 3 sparse PCs
s.sca <- sca(pitprops, 3, gamma = 4.5)
print(s.sca, verbose = TRUE)

## find 6 sparse PCs
s.sca <- sca(pitprops, 6, gamma = 6)
print(s.sca, verbose = TRUE)
```

---

**shrinkage**

**Shrinkage**

Description
Shrink a matrix using soft-thresholding or hard-thresholding.

Usage

```r
shrinkage(x, gamma, shrink = c("soft", "hard"), epsilon = 1e-11)
```
Arguments

- `x` : matrix or `Matrix`, to be threshold.
- `gamma` : numeric, the constraint of Lp norm, i.e. $\|x\| \leq \gamma$.
- `shrink` : character(1), shrinkage method, either "soft"-(default) or "hard"-thresholding (see details).
- `epsilon` : numeric, precision tolerance. This should be greater than `.Machine$double.eps`.

Details

A binary search to find the cut-off value.

`shrink`: The `shrink` option specifies the shrinkage operator to use. Currently, there are two build-in options—"soft"- and "hard"-thresholding. The "soft"-thresholding universally reduce all elements and sets the small elements to zeros. The "hard"-thresholding only sets the small elements to zeros.

Value

A list with two components:

- `matrix` : matrix, the matrix that results from soft-thresholding
- `norm` : numeric, the norm of the matrix after soft-thresholding. This value is close to constraint if using the second option.

References

Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

See Also

`prs`

Examples

```r
x <- matrix(1:6, nrow = 3)
shrink_x <- shrinkage(x, 1)
```

---

### sma

**Sparse Matrix Approximation**

#### Description

Perform the sparse matrix approximation (SMA) of a data matrix `X` as three components: $Z B Y'$, where $Z$ and $Y$ are sparse, and $B$ is low-rank but not necessarily diagonal.
Usage

sma(
  A,
  k = min(5, dim(A)),
  gamma = NULL,
  rotate = c("varimax", "absmin"),
  shrink = c("soft", "hard"),
  center = FALSE,
  scale = FALSE,
  normalize = FALSE,
  order = FALSE,
  flip = FALSE,
  max.iter = 1000,
  epsilon = 1e-05,
  quiet = TRUE
)

Arguments

A  matrix or Matrix to be analyzed.
k  integer, rank of approximation.
gamma numeric(2), sparsity parameters. If gamma is numeric(1), it is used for both left and right sparsity component (i.e, Z and Y). If absent, the two parameters are set as (default): \( \sqrt{nk} \) and \( \sqrt{pk} \) for Z and Y respectively, where \( n \times p \) is the dimension of A.
rotate character(1), rotation method. Two options are currently available: "varimax" (default) or "absmin" (see details).
shrink character(1), shrinkage method, either "soft"-(default) or "hard"-thresholding (see details).
center logical, whether to center columns of A (see scale()).
scale logical, whether to scale columns of A (see scale()).
normalize logical, whether to rows normalization should be done before and undone afterward the rotation (see details).
order logical, whether to re-order the columns of the estimates (see details).
flip logical, whether to flip the signs of the columns of estimates such that all columns are positive-skewed (see details).
max.iter integer, maximum number of iteration (default to 1,000).
epsilon numeric, tolerance of convergence precision (default to 0.00001).
quiet logical, whether to mute the process report (default to TRUE)

Details

rotate: The rotate option specifies the rotation technique to use. Currently, there are two build-in options—"varimax" and "absmin". The "varimax" rotation maximizes the element-wise L4 norm of the rotated matrix. It is faster and computationally more stable. The "absmin" rotation minimizes
the absolute sum of the rotated matrix. It is sharper (as it directly minimizes the L1 norm) but slower
and computationally less stable.

shrink: The shrink option specifies the shrinkage operator to use. Currently, there are two build-in
options—“soft”- and “hard”-thresholding. The “soft”-thresholding universally reduce all elements
and sets the small elements to zeros. The “hard”-thresholding only sets the small elements to zeros.

normalize: The argument normalize gives an indication of if and how any normalization should
be done before rotation, and then undone after rotation. If normalize is FALSE (the default) no nor-
malization is done. If normalize is TRUE then Kaiser normalization is done. (So squared row entries
of normalized A sum to 1.0. This is sometimes called Horst normalization.) For rotate="absmin",
if normalize is a vector of length equal to the number of indicators (i.e., the number of rows of
A), then the columns are divided by normalize before rotation and multiplied by normalize after
rotation. Also, If normalize is a function then it should take A as an argument and return a vector
which is used like the vector above.

order: In PCA (and SVD), the principal components (and the singular vectors) are ordered. For
this, we order the sparse components (i.e., the columns of Z or Y) by their explained variance in the
data, which is defined as \( \sum((A*Y^T)^2) \), where y is a column of the sparse component. Note:
not to be confused with the cumulative proportion of variance explained by Y (and Z), particularly
when Y (and Z) is not be strictly orthogonal.

flip: The argument flip gives an indication of if and the columns of estimated sparse component
should be flipped. Note that the estimated (sparse) loadings, i.e., the weights on original variables,
are column-wise invariant to a sign flipping. This is because flipping of a principal direction does
not influence the amount of the explained variance by the component. If flip=TRUE, then the
columns of loadings will be flip accordingly, such that each column is positive-skewed. This means
that for each column, the sum of cubic elements (i.e., \( \sum(x^3) \)) are non-negative.

Value

an SMA object that contains:

Z, B, Y

the three parts in the SMA (i.e., \( ZBY' \)). Z is a sparse n x k matrix that contains
the row components (loadings). The row names of Z inherit the row names of A.
B is a k x k matrix that contains the scores of SMA; the Frobenius norm of B
equals to the total variance explained by the SMA. Y is a sparse n x k matrix that
contains the column components (loadings).

The row names of Y inherit the column names of A.

score

the total variance explained by the SMA. This is the optimal objective value
obtained.

n.iter

integer, the number of iteration taken.

References

Chen, F. and Rohe, K. (2020) "A New Basis for Sparse PCA."

See Also

sca, prs
Examples

```r
## simulate a rank-5 data matrix with some additive Gaussian noise
n <- 300
p <- 50
k <- 5  # rank
Z <- shrinkage(polar(matrix(runif(n * k), n, k)), sqrt(n))
B <- diag(5) * 3
Y <- shrinkage(polar(matrix(runif(p * k), p, k)), sqrt(p))
E <- matrix(rnorm(n * p, sd = .01), n, p)
X <- scale(Z %*% B %*% t(Y) + E)

## perform sparse matrix approximation
s.sma <- sma(X, k)
s.sma
```

---

### soft

**Soft-thresholding**

**Description**

Perform soft-thresholding given the cut-off value.

**Usage**

```r
soft(x, t)
```

**Arguments**

- `x` any numerical matrix or vector.
- `t` numeric, the amount to soft-threshold, i.e., \( \text{sgn}(x_{ij})(|x_{ij} - t|)_+ \).

---

### trim.fringe

**Trim Image**

**Description**

Trim the fringes of image.

**Usage**

```r
trim.fringe(img, hem = 6)
```

**Arguments**

- `img` a matrix of square image.
- `hem` integer, the width of trimmed fringes.
**varimax**

**Varimax Rotation**

**Description**
This is a re-implementation of `stats::varimax`, which (1) adds a parameter for the maximum number of iterations, (2) sets the default `normalize` parameter to `FALSE`, (3) outputs the number of iteration taken, and (4) returns regular `matrix` rather than in `loadings` class.

**Usage**
```
varimax(x, normalize = FALSE, eps = 1e-05, maxit = 1000L)
```

**Arguments**
- `x` A loadings matrix, with `p` rows and `k < p` columns
- `normalize` logical. Should Kaiser normalization be performed? If so the rows of `x` are re-scaled to unit length before rotation, and scaled back afterwards.
- `eps` The tolerance for stopping: the relative change in the sum of singular values.
- `maxit` integer, maximum number of iteration (default to 1,000).

**Value**
A list with three elements:
- `rotated` the rotated matrix.
- `rotmat` the (orthogonal) rotation matrix.
- `n.iter` the number of iterations taken.

**See Also**
- `stats::varimax`

---

**varimax.criteria**

**The varimax criterion**

**Description**
Calculate the varimax criterion.

**Usage**
```
varimax.criterion(mat)
```
Arguments
mat a matrix or Matrix.

Value
a numeric of evaluated varimax criterion.

References
Varimax rotation (Wikipedia)

Examples
```r
## use the "swiss" data
fa <- factanal(~., 2, data = swiss, rotation = "none")
lds <- loadings(fa)

## compute varimax criterion:
varimax.criteria(lds)

## compute varimax criterion (after the varimax rotation):
rlds <- rotation(lds, rotate = "varimax")
varimax.criteria(rlds)
```

---

vgQ.absmin Gradient of Absmin Criterion

Description
This is a helper function for absmin and is not to be used directly by users.

Usage
vgQ.absmin(L)

Arguments
L a matrix or Matrix.

Value
a list required by GPArotation::GPForth for the absmin rotation.
Examples

```r
## Not run:
## NOT RUN
## NOT for users to call.

## End(Not run)
```
Index

*Topic datasets
  pitprops, 11

*Topic package
  ePCA-package, 2
  absmin, 3, 6
  absmin.criteria, 3, 13
  base::rank(), 9
  clue::solve_LSAP, 5
  clue::solve_LSAP(), 4
  cpve, 4
  dist.matrix, 4
  distance, 5
  ePCA-package, 2
  exp.frac, 6
  fftshift, 7
  hard, 7
  inner, 8
  labelCluster, 8
  misClustRate, 9
  norm.Lp, 10
  permColumn, 10
  pitprops, 11
  polar, 11, 15
  prewhiten, 12
  print, 12, 13
  print.sca, 12
  print.sma, 13
  prs, 13, 17, 20, 21, 23
  prs(), 11
  pve, 4, 15
  rootmatrix, 16
  rotation, 15, 16
  sca, 15, 17, 23
  scale(), 18, 22
  shrinkage, 15, 20
  shrinkage(), 10
  SMA, 15, 20, 21
  sma(), 13
  soft, 24
  stats::varimax, 25
  trim.fringe, 24
  varimax, 17, 25
  varimax.criteria, 25
  vgq.absmin, 26