Package ‘sca’

January 14, 2023

Title Simple Component Analysis

Version 0.9-1

Date 2023-01-14

Description Simple Component Analysis (SCA) often provides much more interpretable components than Principal Components (PCA) while still representing much of the variability in the data.

Author Valentin Rousson `<rousson@ifspm.unizh.ch>` and Martin Maechler

Maintainer Martin Maechler `<maechler@stat.math.ethz.ch>`

Imports grDevices, graphics, stats

Suggests MASS, Matrix

BuildResaveData no

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-14 13:20:02 UTC

R topics documented:

agglomblock *Agglomerate Two Block-Components in SCA*

Description

Agglomerate the two block-components which are *closest* according to the specified `clustering` method.

Usage

```
agglomblock(S, P, cluster = c("median","single","complete"))
```

Arguments

`S`	correlation/covariance matrix
`P`	component matrix
`cluster`	character specifying the clustering method; default `"median"`, see `sca(*, cluster=)`.

Value

New component matrix with one block component less.

See Also

`sca`, also for references

allcrit *Simple Component Quality Criterion Computation*

Description

Compute simple component criterion for components `P` on cor.matrix `S` (cumulative), using `sccrit()`.

Function `allcrit()` computes even more criteria, some derived from `sccrit()`.

Usage

```
allcrit(S, P, criterion, sortP = TRUE)
 sccrit(S, P, criterion, sortP = TRUE)
```

Arguments

`S`	correlation/covariance matrix
`P`	component matrix
`criterion`	character string specifying the optimality criterion to be used in `sccrit()` for evaluating a system of simple components. One of `"csv"` (corrected sum of variances) or `"blp"` (best linear predictor).
`sortP`	logical indicating if `P` should be sorted; if true, `sortmatrix(S,P)` used in lieu of `P`.

Value

`sccrit()` returns a numeric vector, the criterion computed (cumulatively).

`allcrit()` returns a list with components `varpc`, `varsc`, `cumpc`, `cumsc`, `opt`, `corsc`, and `maxcor`; see the description of the `allcrit` component in the return value of `sca()`.

Author(s)

Valentin Rousson `<rousson@ifspm.unizh.ch>` and Martin Maechler `<maechler@stat.math.ethz.ch>`.

See Also

`quickcrit`, `sca`, also for references.

`corcomp` *Covariance and Correlation Matrix of Components P on S*

Description

`covcomp` returns the variance-covariance matrix of the components P on S, and `corcomp` returns the correlation matrix.

Usage

```
corcomp(S, P)
covcomp(S, P)
```

Arguments

`S`	correlation/covariance matrix of the p original variables.
`P`	component matrix of dimension $p \times b$.

Value

a square $b \times b$ matrix.

Author(s)

Valentin Rousson <rousson@ifspm.unizh.ch> and Martin Maechler <maechler@stat.math.ethz.ch>.

See Also

sca, also for references

Examples

```
data(USJudgeRatings)
S.jr <- cor(USJudgeRatings)
sca.jr <- sca(S.jr, b=4, inter=FALSE)
Vr <- covcomp(S.jr, P = sca.jr$simplemat)
Vr
Cr <- corcomp(S.jr, P = sca.jr$simplemat)
Cr
```

---

firstpcres *First Principal Component of Residuals given Components*

Description

Return the first principal component of residuals of S given the components P.

Usage

```
firstpcres(S, P)
```

Arguments

S	correlation/covariance matrix
P	component matrix

Value

numeric vector; actually, the first eigenvector of $S - A'(AP)^{-1}A$ where $A := P'S$.

See Also

sca, also for references

hearlossC *Hearing Loss Correlation Data*

Description

The data consist of eight measurements of hearing loss taken on 100 males, aged 39, who had no indication of hearing difficulties. These measurements are decibel loss (in comparison to a reference standard) at frequencies 500Hz, 1000Hz, 2000Hz and 4000Hz for the left and the right ear, respectively.

Usage

```
data(hearlossC)
```

Format

Eight Variables, first the ones for "Left", than for the "Right". The frequences are abbreviated, e.g., `2k` for *2000 Hz* or `5c` for *500 Hz*. The variable names are (in this order) `"Left5c"`, `"Left1k"`, `"Left2k"`, `"Left4k"`, `"Right5c"`, `"Right1k"`, `"Right2k"`, `"Right4k"`.

Source

This is the correlation matrix of data described in Chapter 5 of Jackson (1991).

References

Jackson, J.E. (1991) *A User's Guide to Principal Components*. John Wiley, New York.

Examples

```
data(hearlossC)
symnum(hearlossC)
sca(hearlossC) # -> explains 89.46% instead of 91.62
```

maxmatrix *Largest Element in Correlation Matrix*

Description

return position and value of the largest element of a correlation matrix R (without taking into account the diagonal elements)

Usage

```
maxmatrix(R)
```

Arguments

`R`	a square symmetric numeric matrix

Value

a list with components

`row`	row index of maximum
`col`	col index of maximum
`val`	value of maximum, i.e. `val` == `R[row,col]`.

See Also

`sca`, also for references

Examples

```
data(reflexesC)
maxmatrix(reflexesC) # ->  0.98 at  [1, 2]
```

`nextdiff` *Compute the Next Simple Difference-Component for SCA*

Description

Compute the next simple difference-component; this is an auxiliary function for `sca`.

Usage

```
nextdiff(S, P, withinblock, criterion)
```

Arguments

`S`	correlation/covariance matrix
`P`	component matrix
`withinblock`	logical indicating whether any given difference-component should only involve variables belonging to the same block-component.
`criterion`	character string specifying the optimality criterion, see `sccrit` for details.

Details

Uses `firstpcres(S,P)` and subsequently `shrinkdiff()`, the latter in a loop when `withinblock` is true.

In order to ensure uniqueness, we ensure that the first (non zero) entry of the principal component is always *positive*.

Value

a list with components

`P`	the new component matrix, i.e. the input `P` with one new column appended.
`nextpc`	the next principal component with many entries set to 0.

Author(s)

Valentin Rousson <`rousson@ifspm.unizh.ch`> and Martin Maechler <`maechler@stat.math.ethz.ch`>.

See Also

`shrinkdiff`; `sca`, also for references

`percent` *Simple Formatting of Percentages*

Description

Returns strings of the same length as `p`, displaying the `100 * p` percentages.

Usage

```
percent(p, d = 0, sep = " ")
```

Arguments

`p`	number(s) in $[0, 1]$ – to be “displayed” as percentage(s).
`d`	number of digits after decimal point.
`sep`	separator to use before the final `"%"`.

Value

character vector of the same length as `p`.

Author(s)

Martin Maechler

Examples

```
percent(0.25)
noquote(percent((1:10)/10))
(pc <- percent((1:10)/30, 1, sep=""))
noquote(pc)
```

`pitpropC`	*Pitprops Strength Correlation Data*

Description

This correlation matrix was published in Jeffers (1967) and was calculated from 180 observations. The 13 variables were used as explanatory variables in a regression problem which arised from a study on the strength of pitprops cut from home-grown timber.

Usage

```
data(pitpropC)
```

Format

Its a correlation matrix of 13 variables which have the following meaning:

[,1]	TOPDIAM	Top diameter of the prop in inches
[,2]	LENGTH	Length of the prop in inches
[,3]	MOIST	Moisture content of the prop, expressed as a percentage of the dry weight
[,4]	TESTSG	Specific gravity of the timber at the time of the test
[,5]	OVENSG	Oven-dry specific gravity of the timber
[,6]	RINGTOP	Number of annual rings at the top of the prop
[,7]	RINGBUT	Number of annual rings at the base of the prop
[,8]	BOWMAX	Maximum bow in inches
[,9]	BOWDIST	Distance of the point of maximum bow from the top of the prop in inches
[,10]	WHORLS	Number of knot whorls
[,11]	CLEAR	Length of clear prop from the top of the prop in inches
[,12]	KNOTS	Average number of knots per whorl
[,13]	DIAKNOT	Average diameter of the knots in inches

Details

Jeffers (1967) replaced these 13 variables by their first six principal components. As noted by Vines (2000), this is an example where simple structure has proven difficult to detect in the past.

References

Jeffers, J.N.R. (1967) Two case studies in the application of principal components analysis. *Appl. Statist.* **16**, 225–236.

Vines, S.K. (2000) Simple principal components. *Appl. Statist.* **49**, 441–451.

Examples

```
data(pitpropC)
symnum(pitpropC)
```

quickcrit *Additional Contribution of New Component to the SC System*

Description

Compute the additional contribution of a new component to the simple component system P on S.

Usage

```
quickcrit(newcomp, S, P, criterion)
```

Arguments

`newcomp`	numeric vector, typically the result of `simpvector()`.
`S`	correlation/covariance matrix
`P`	component matrix
`criterion`	character string specifying the optimality criterion, see `sccrit` for details.

Value

...

See Also

`sccrit`; further `sca`, also for references.

reflexesC *Human Reflexes Correlation Data*

Description

This correlation matrix was published in Jolliffe (2002, p.58). The data consist of measurements of strength of reflexes at ten sites of the body, taken for 143 individuals. The variables come in five pairs, corresponding to right and left measurements on triceps, biceps, wrists, knees and ankles, respectively.

Usage

```
data(reflexesC)
```

Format

It is a $10x10$ correlation matrix, i.e. symmetric, and diagonal `1`. The five pairs of variables are (in this order) `"triceps.R"`, `"triceps.L"`, `"biceps.R"`, `"biceps.L"`, `"wrist.R"`, `"wrist.L"`, `"knee.R"`, `"knee.L"`, `"ankle.R"`, `"ankle.L"`.

References

Jolliffe, I.T. (2002) Principal Component Analysis (2nd ed.). Springer, New York.

Examples

```
data(reflexesC)
symnum(reflexesC)
sca(reflexesC)  # sca gets 97.95%  of PCA
```

---

sca *Simple Component Analysis – Interactively*

Description

A system of simple components calculated from a correlation (or variance-covariance) matrix is built (interactively if `interactive = TRUE`) following the methodology of Rousson and Gasser (2003).

Usage

```
sca(S, b = if(interactive) 5, d = 0, qmin = if(interactive) 0 else 5,
    corblocks = if(interactive) 0 else 0.3,
    criterion = c("csv", "blp"), cluster = c("median","single","complete"),
    withinblock = TRUE, invertsigns = FALSE,
    interactive = dev.interactive())
## S3 method for class 'simpcomp'
print(x, ndec = 2, ...)
```

Arguments

`S`	the correlation (or variance-covariance) matrix to be analyzed.
`b`	the number of block-components initially proposed.
`d`	the number of difference-components initially proposed.
`qmin`	if larger than zero, the number of difference-components is chosen such that the system contains at least `qmin` components (overriding argument `d`!).
`corblocks`	if larger than zero, the number of block-components is chosen such that correlations among them are all smaller than `corblocks` (overriding argument `b`).
`criterion`	character string specifying the optimality criterion to be used for evaluating a system of simple components. One of `"csv"` (corrected sum of variances) or `"blp"` (best linear predictor); can be abbreviated.
`cluster`	character string specifying the clustering method to be used in the definition of the block-components. One of `"single"` (single linkage), `"median"` (median linkage) or `"complete"` (complete linkage) can be abbreviated.
`withinblock`	a logical indicating whether any given difference-component should only involve variables belonging to the same block-component.

`invertsigns`	a logical indicating whether the sign of some variables should be inverted initially in order to avoid negative correlations.
`interactive`	a logical indicating whether the system of simple components should be built interactively. If `interactive=FALSE`, an optimal system of simple components is automatically calculated without any intervention of the user (according to `b` or `corblocks`, and to `d` or `qmin`). By default, `interactive = dev.interactive()` (which is true if `interactive()` and `.Device` is an interactive graphics device).
`x`	an object of class `sca`, typically the result of `sca(..)`.
`ndec`	number of decimals *after* the dot, for the percentages printed.
`...`	further arguments, passed to and from methods.

Details

When confronted with a large number p of variables measuring different aspects of a same theme, the practitionner may like to summarize the information into a limited number q of components. A *component* is a linear combination of the original variables, and the weights in this linear combination are called the *loadings*. Thus, a system of components is defined by a p times q dimensional matrix of loadings.

Among all systems of components, principal components (PCs) are optimal in many ways. In particular, the first few PCs extract a maximum of the variability of the original variables and they are uncorrelated, such that the extracted information is organized in an optimal way: we may look at one PC after the other, separately, without taking into account the rest.

Unfortunately PCs are often difficult to interpret. The goal of Simple Component Analysis is to replace (or to supplement) the optimal but non-interpretable PCs by suboptimal but interpretable *simple components*. The proposal of Rousson and Gasser (2003) is to look for an optimal system of components, but only among the simple ones, according to some definition of optimality and simplicity. The outcome of their method is a simple matrix of loadings calculated from the correlation matrix S of the original variables.

Simplicity is not a guarantee for interpretability (but it helps in this regard). Thus, the user may wish to partly modify an optimal system of simple components in order to enhance interpretability. While PCs are by definition 100% optimal, the optimal system of simple components proposed by the procedure `sca` may be, say, 95%, optimal, whereas the simple system altered by the user may be, say, 93% optimal. It is ultimately to the user to decide if the gain in interpretability is worth the loss of optimality.

The interactive procedure `sca` is intended to assist the user in his/her choice for an interptetable system of simple components. The algorithm consists of three distinct stages and proceeds in an interative way. At each step of the procedure, a simple matrix of loadings is displayed in a window. The user may alter this matrix by clicking on its entries, following the instructions given there. If all the loadings of a component share the same sign, it is a “block-component”. If some loadings are positive and some loadings are negative, it is a “difference-component”. Block-components are arguably easier to interpret than difference-components. Unfortunately, PCs almost always contain only one block-component. In the procedure `sca`, the user may choose the number of block-components in the system, the rationale being to have as many block-components such that correlations among them are below some cut-off value (typically .3 or .4).

Simple block-components should define a partition of the original variables. This is done in the first stage of the procedure `sca`. An agglomerative hierarchical clustering procedure is used there.

The second stage of the procedure `sca` consists in the definition of simple difference-components. Those are obtained as simplified versions of some appropriate “residual components”. The idea is to retain the large loadings (in absolute value) of these residual components and to shrink to zero the small ones. For each difference-component, the interactive procedure `sca` displays the loadings of the corresponding residual component (at the right side of the window), such that the user may know which variables are especially important for the definition of this component.

At the third stage of the interactive procedure `sca`, it is possible to remove some of the difference-components from the system.

For many examples, it is possible to find a simple system which is 90% or 95% optimal, and where correlations between components are below 0.3 or 0.4. When the structure in the correlation matrix is complicated, it might be advantageous to invert the sign of some of the variables in order to avoid as much as possible negative correlations. This can be done using the option ‘invertsigns=TRUE’.

In principle, simple components can be calculated from a correlation matrix or from a variance-covariance matrix. However, the definition of simplicity used is not well adapted to the latter case, such that it will result in systems which are far from being 100% optimal. Thus, it is advised to define simple components from a correlation matrix, not from a variance-covariance matrix.

Value

An object of class `simpcomp` which is basically as list with the following components:

`simplemat`	an integer matrix defining a system of simple components. The rows correspond to variables and the columns correspond to components.
`loadings`	loadings of simple components. This is a version of `simplemat`, normalized by a version of `scale`.
`allcrit`	a `list` containing the following components:

- **varpc** a vector containing the percentage of total variability accounted by each of the the first `nblock + ndiff` principal components of `S`.
- **varsc** a vector containing the percentage of total variability accounted by each of the simple components defined by `simplemat`.
- **cumpc** the sum of varpc, indicating the percentage of total variability accounted by the first `nblock + ndiff` principal components of `S`.
- **cumsc** a score indicating the percentage of total variability accounted by the system of simple components. `cumsc` is calculated according to `criterion`.
- **opt** indicates the optimality of the system of simple components and is computed as `cumsc/cumpc`.
- **corsc** correlation matrix of the simple components defined by `simplemat`.
- **maxcor** a list with the following components:
  - **row** label of the row of the maximum value in `corsc`.
  - **col** label of the column of the maximum value in `corsc`.
  - **val** maximum value in `corsc` (in absolute value).

`nblock`	number of block-components in `simplemat`.

`ndiff`	number of difference-components in `simplemat`.
`criterion`	as above.
`cluster`	as above.
`withinblock`	as above.
`invertsigns`	as above
`vardata`	the correlation (or variance-covariance) matrix which was analyzed. In principle it should be equal to argument `S` above, except if it has been transformed in order to avoid negative correlations.

Note

PCA already is known to be "non-unique" in the sense that the principal directions (eigen vectors, `eigen`) are only determined up to a factor ± 1, i.e., sign change.

Consequently results may change depending e.g., only on the Lapack / BLAS library used.

This is even more the case for SCA, notably in artificial situations such as the '`tests/artif3.R`' in the sources of **sca**.

Author(s)

Valentin Rousson <`rousson@ifspm.unizh.ch`> and Martin Maechler <`maechler@stat.math.ethz.ch`>.

References

Rousson, Valentin and Gasser, Theo (2004) Simple Component Analysis. *JRSS: Series C (Applied Statistics)* **53**(4), 539–555; doi:10.1111/j.14679876.2004.05359.x

Rousson, V. and Gasser, Th. (2003) *Some Case Studies of Simple Component Analysis*. Manuscript, *no longer* available as '`https://www.biostat.uzh.ch/research/manuscripts/scacases.pdf`'

Gervini, D. and Rousson, V. (2003) *Some Proposals for Evaluating Systems of Components in Dimension Reduction Problems*. Submitted.

See Also

`prcomp` (for PCA), etc.

Examples

```
data(pitpropC)
sc.pitp <- sca(pitpropC, interactive=FALSE)
sc.pitp
## to see it's low-level components:
str(sc.pitp)

## Let `X' be a matrix containing some data set whose rows correspond to
## subjects and whose columns correspond to variables. For example:


library(MASS)
SigU <- function(p, rho) { r <- diag(p); r[col(r) != row(r)] <- rho; r}
```

```
rmvN <- function(n,p, rho)
        mvrnorm(n, mu=rep(0,p), Sigma = SigU(p, rho))
X <- cbind(rmvN(100, 3, 0.7),
           rmvN(100, 2, 0.9),
           rmvN(100, 4, 0.8))


## An optimal simple system with at least 5 components for the data in `X',
## where the number of block-components is such that correlations among
## them are all smaller than 0.4, can be automatically obtained as:

(r <- sca(cor(X), qmin=5, corblocks=0.4, interactive=FALSE))

## On the other hand, an optimal simple system with two block-components
## and two difference-components for the data in `X' can be automatically
## obtained as:

(r <- sca(cor(X), b=2, d=2, qmin=0, corblocks=0, interactive=FALSE))

## The resulting simple matrix is contained in `r$simplemat'.
## A matrix of scores for such simple components can then be obtained as:

(Z <- scale(X) %*% r$loadings)

## On the other hand, scores of simple components calculated from the
## variance-covariance matrix of `X' can be obtained as:

r <- sca(var(X), b=2, d=2, qmin=0, corblocks=0, interactive=FALSE)
Z <- scale(X, scale=FALSE) %*% r$loadings

## One can also use the program interactively as follows:

if(interactive()) {
  r <- sca(cor(X), corblocks=0.4, qmin=5, interactive = TRUE)

  ## Since the interactive part of the program is active here, the proposed
  ## system can then be  modified according to the user's wishes. The
  ## result of the procedure will be contained in `r'.
}
```

---

shrinkdiff *Shrink Component Towards a Simple Difference-Component in SCA*

Description

Shrinks a (principal) component towards a simple difference-component in `sca`.

Usage

```
shrinkdiff(zcomp, S, P, criterion)
```

Arguments

`zcomp`	a component vector to be *simplified*.
`S`	correlation/covariance matrix
`P`	component matrix
`criterion`	character string specifying the optimality criterion, see `sccrit` for details.

Value

a list with the components

`scompmax`	a new simple component vector, typically result of `simpvector`.
`critmax`	the (optimal) value of the criterion, achieved for `scompmax`.

See Also

`sca`, also for references

`simpvector` *Simplify a (Principal Component) Vector to a Simple Component*

Description

Simplifies the vector `x` to become a "simple" component vector (of the same size).

Usage

```
simpvector(x)
```

Arguments

`x`	numeric vector of length `n`, say.

Value

a "simplified" version of `x`, i.e. an integer vector of the same length and each entry with the same signs.

See Also

`sca`, also for references

Examples

```
x0 <- c(-2:3, 3:-1,0:3,1,1)
cbind(x0, simpvector(x0)) # entries (-11, 0, 3)
```

`sortmatrix` *Sort Simple Component Matrix*

Description

Reorder the columns of a component matrix `P` by decreasing variances of components where the block-components come first, the difference components afterwards.

Usage

```
sortmatrix(S, P)
```

Arguments

`S`	correlation/covariance matrix
`P`	component matrix

Value

numeric matrix which is just `P` with columns reordered.

See Also

`sca`, also for references

Index