Package ‘pcaMethods’

May 21, 2019

Maintainer Henning Redestig <henning.red@gmail.com>
License GPL (>= 3)
Title A collection of PCA methods
LinkingTo Rcpp
LazyLoad Yes
Author Wolfram Stacklies, Henning Redestig, Kevin Wright
SystemRequirements Rcpp
Description Provides Bayesian PCA, Probabilistic PCA, Nipals PCA,
Inverse Non-Linear PCA and the conventional SVD PCA. A cluster
based method for missing value estimation is included for
comparison. BPCA, PPCA and NipalsPCA may be used to perform PCA
on incomplete data as well as for accurate missing value
estimation. A set of methods for printing and plotting the
results is also provided. All PCA methods make use of the same
data structure (pcaRes) to provide a common interface to the
PCA results. Initiated at the Max-Planck Institute for
Molecular Plant Physiology, Golm, Germany.
Version 1.76.0
URL https://github.com/hredestig/pcamethods
BugReports https://github.com/hredestig/pcamethods/issues
Encoding UTF-8
Depends Biobase, methods
Imports BiocGenerics, Rcpp (>= 0.11.3), MASS
Suggests matrixStats, lattice, ggplot2
Collate 'derrorHierarchic.R' 'errorHierarchic.R' 'AllClasses.R'
 'AllGenerics.R' 'BPCA_dostep.R' 'BPCA_initmodel.R' 'bpca.R'
 'checkData.R' 'forkNlpcaNet.R' 'kEstimate.R' 'kEstimateFast.R'
 'lineSearch.R' 'llsImpute.R' 'methods-ExpressionSet.R'
 'methods-nniRes.R' 'methods-pcaRes.R' 'nipalsPca.R' 'nlpca.R'
 'optiAlgCgd.R' 'orth.R' 'pca.R' 'pcaMethods-package.R' 'ppca.R'
 'prep.R' 'repmat.R' 'robustPca.R' 'sortFeatures.R'
 'svdImpute.R' 'vector2matrices.R' 'xval.R'
biocViews Bayesian
RoxygenNote 6.1.1
topics documented:

- `asExprSet` ................................................................. 3
- `biplot-methods` ............................................................ 4
- `b pca` .................................................................. 5
- `BPCA_dostep` ................................................................. 7
- `BPCA_initmodel` ............................................................... 8
- `center,pcaRes-method` ......................................................... 9
- `centered,pcaRes-method` ....................................................... 9
- `checkData` .................................................................. 10
- `completeObs,nniRes-method` ............................................... 11
- `cvseg` ........................................................................ 11
- `cvstat,pcaRes-method` ....................................................... 12
- `deletediagonals` ............................................................... 13
- `derrorHierarchic` ............................................................. 13
- `dim,pcaRes` ................................................................. 14
- `DModX,pcaRes-method` ....................................................... 14
- `errorHierarchic` .............................................................. 15
- `fitted-methods` ................................................................ 16
- `forkNlpcaNet` ................................................................. 17
- `getHierarchicIdx` ............................................................. 17
- `helix` ........................................................................ 18
- `kEstimate` ................................................................. 18
- `kEstimateFast` .............................................................. 20
- `leverage,pcaRes-method` ..................................................... 22
- `lineSearch` ................................................................. 23
- `linr` .......................................................................... 23
- `listPcaMethods` .............................................................. 24
- `llsImpute` ................................................................. 24
- `loadings,ANY-method` ...................................................... 26
- `loadings,pcaRes-method` ................................................... 26
- `loadings,pcaRes` ............................................................. 27
- `metaboliteData` .............................................................. 28
- `metaboliteDataComplete` ................................................... 28
- `method,pcaRes-method` ..................................................... 29
- `nipalsPca` ................................................................. 29
- `nlpca` ...................................................................... 31
- `nmissing,pcaRes-method` ................................................ 32
- `nni` ........................................................................ 33
- `nniRes` ...................................................................... 34
- `nObs,pcaRes-method` ....................................................... 34
- `nP,pcaRes-method` ........................................................... 35
- `nPcs,pcaRes-method` ....................................................... 35
- `nVar,pcaRes-method` ....................................................... 36
asExprSet

**Convert pcaRes object to an expression set**

**Description**

This function can be used to conveniently replace the expression matrix in an ExpressionSet with the completed data from a pcaRes object.

**Usage**

```r
asExprSet(object, exprSet)
```
Arguments

- `object` pcaRes – The object containing the completed data.
- `exprSet` ExpressionSet – The object passed on to `pca` for missing value estimation.

Details

This is not a standard as function as pcaRes object alone not can be converted to an ExpressionSet (the pcaRes object does not hold any phenoData for example).

Value

An object without missing values of class ExpressionSet.

Author(s)

Wolfram Stacklies
CAS-MPG Partner Institute for Computational Biology, Shanghai, China

---

**Description**

Visualize two-components simultaneously

**Usage**

```r
## S3 method for class 'pcaRes'
between(x, choices = 1:2, scale = 1, pc.biplot = FALSE, ...)

## S4 method for signature 'pcaRes'
between(x, choices = 1:2, scale = 1, pc.biplot = FALSE, ...)
```

**Arguments**

- `x` a pcaRes object
- `choices` which two pcs to plot
- `scale` The variables are scaled by $\lambda^{\text{scale}}$ and the observations are scaled by $\lambda^{\text{scale}}$ where $\lambda$ are the singular values as computed by `princomp`. Normally $0 \leq \text{scale} \leq 1$, and a warning will be issued if the specified 'scale' is outside this range.
- `pc.biplot` If true, use what Gabriel (1971) refers to as a "principal component biplot", with $\lambda = 1$ and observations scaled up by sqrt(n) and variables scaled down by sqrt(n). Then the inner products between variables approximate covariances and distances between observations approximate Mahalanobis distance.
- `...` optional arguments to be passed to `biplot.default`
Details

This is a method for the generic function ‘biplot’. There is considerable confusion over the precise definitions: those of the original paper, Gabriel (1971), are followed here. Gabriel and Odoroff (1990) use the same definitions, but their plots actually correspond to `pc.biplot = TRUE`.

Value

a plot is produced on the current graphics device.

Author(s)

Kevin Wright, Adapted from `biplot.prcomp`

See Also

`prcomp`, `pca`, `princomp`

Examples

data(iris)
pcIr <- pca(iris[,1:4])
biplo\(t(pci)r)
Details

Scores and loadings obtained with Bayesian PCA slightly differ from those obtained with conventional PCA. This is because BPCA was developed especially for missing value estimation. The algorithm does not force orthogonality between factor loadings, as a result factor loadings are not necessarily orthogonal. However, the BPCA authors found that including an orthogonality criterion made the predictions worse.

The authors also state that the difference between real and predicted Eigenvalues becomes larger when the number of observation is smaller, because it reflects the lack of information to accurately determine true factor loadings from the limited and noisy data. As a result, weights of factors to predict missing values are not the same as with conventional PCA, but the missing value estimation is improved.

BPCA works iteratively, the complexity is growing with \(O(n^3)\) because several matrix inversions are required. The size of the matrices to invert depends on the number of components used for re-estimation.

Finding the optimal number of components for estimation is not a trivial task; the best choice depends on the internal structure of the data. A method called kEstimate is provided to estimate the optimal number of components via cross validation. In general few components are sufficient for reasonable estimation accuracy. See also the package documentation for further discussion about on what data PCA-based missing value estimation makes sense.

It is not recommended to use this function directly but rather to use the pca() wrapper function.

There is a difference with respect the interpretation of rows (observations) and columns (variables) compared to matlab implementation. For estimation of missing values for microarray data, the suggestion in the original bpc is to interpret genes as observations and the samples as variables. In pcaMethods however, genes are interpreted as variables and samples as observations which arguably also is the more natural interpretation. For bpca behavior like in the matlab implementation, simply transpose your input matrix.

Details about the probabilistic model underlying BPCA are found in Oba et. al 2003. The algorithm uses an expectation maximization approach together with a Bayesian model to approximate the principal axes (eigenvectors of the covariance matrix in PCA). The estimation is done iteratively, the algorithm terminates if either the maximum number of iterations was reached or if the estimated increase in precision falls below \(1e^{-4}\).

**Complexity:** The relatively high complexity of the method is a result of several matrix inversions required in each step. Considering the case that the maximum number of iteration steps is needed, the approximate complexity is given by the term

\[
\text{maxSteps} \cdot \text{row}_{\text{miss}} \cdot O(n^3)
\]

Where \(\text{row}_{\text{miss}}\) is the number of rows containing missing values and \(O(n^3)\) is the complexity for inverting a matrix of size \(\text{components}\). Components is the number of components used for re-estimation.

Value

Standard PCA result object used by all PCA-based methods of this package. Contains scores, loadings, data mean and more. See pcaRes for details.

Note

Requires MASS.
BPCA_dostep

Author(s)
Wolfram Stacklies

References

See Also
ppca, svdImpute, prcomp, nipalsPca, pca, pcaRes, kEstimate.

Examples
## Load a sample metabolite dataset with 5% missing values (metaboliteData)
data(metaboliteData)
## Perform Bayesian PCA with 2 components
pc <- pca(t(metaboliteData), method="bpca", nPcs=2)
## Get the estimated principal axes (loadings)
loadings <- loadings(pc)
## Get the estimated scores
scores <- scores(pc)
## Get the estimated complete observations
cObs <- completeObs(pc)
## Now make a scores and loadings plot
slplot(pc)

BPCA_dostep  Do BPCA estimation step

Description
The function contains the actual implementation of the BPCA component estimation. It performs one step of the BPCA EM algorithm. It is called 'maxStep' times from within the main loop in BPCAestimate.

Usage
BPCA_dostep(M, y)

Arguments
M  Data structure containing all needed information. See the source documentation of BPCA_initmodel for details
y  Numeric original data matrix

Details
This function is NOT intended to be run standalone.
**Value**

Updated version of the data structure

**Author(s)**

Wolfram Stacklies

---

**Description**

Model initialization for Bayesian PCA. This function is NOT inteded to be run separately!

**Usage**

```r
BPCA_initmodel(y, components)
```

**Arguments**

- **y**  numeric matrix containing missing values. Missing values are denoted as `NA`
- **components**  Number of components used for estimation

**Details**

The function calculates the initial Eigenvectors by use of SVD from the complete rows. The data structure M is created and initial values are assigned.

**Value**

List containing:

- **rows**  Row number of input matrix
- **cols**  Column number of input matrix
- **comps**  Number of components to use
- **yest**  (working variable) current estimate of complete data
- **row_miss**  (Array) Indizes of rows containing missing values
- **row_nomiss**  (Array) Indices of complete rows (such with no missing values)
- **nans**  Matrix of same size as input data. TRUE if input == NA, false otherwise
- **mean**  Column wise data mean
- **PA**  (d x k) Estimated principal axes (eigenvectors, loadings) The matrix ROWS are the vectors
- **tau**  Estimated precision of the residual error
- **scores**  Estimated scores

Further elements are: galpha0, balpha0, alpha, gmu0, btau0, gtau0, SigW. These are working variables or constants.

**Author(s)**

Wolfram Stacklies
**center.pcaRes-method**  
*Get the centers of the original variables*

**Description**  
Get the centers of the original variables

**Usage**  
```r
center(object, ...)  
```

**Arguments**  
- `object`: pcaRes object  
- `...`: Not used

**Value**  
Vector with the centers

**Author(s)**  
Henning Redestig

---

**centered.pcaRes-method**  
*Check centering was part of the model*

**Description**  
Check centering was part of the model

**Usage**  
```r
centered(object, ...)  
```

**Arguments**  
- `object`: pcaRes object  
- `...`: Not used

**Value**  
TRUE if model was centered

**Author(s)**  
Henning Redestig
checkData

Do some basic checks on a given data matrix

Description

Check a given data matrix for consistency with the format required for further analysis. The data must be a numeric matrix and not contain:

- Inf values
- NaN values
- Rows or columns that consist of NA only

Usage

checkData(data, verbose = FALSE)

Arguments

data matrix – Data to check.
verbose boolean – If TRUE, the function prints messages whenever an error in the data set is found.

Value

isValid boolean – TRUE if no errors were found, FALSE otherwise. isValid contains a set of attributes, these are:

- isNumeric - TRUE if data is numeric, false otherwise
- isInfinite - TRUE if data contains ’Inf’ values, false otherwise
- isNaN - TRUE if data contains ’NaN’ values, false otherwise
- isMatrix - TRUE if the data is in matrix format, FALSE otherwise
- naRows - TRUE if data contains rows in which all elements are ’NA’, FALSE otherwise
- naCols - TRUE if data contains columns in which all elements are ’NA’, FALSE otherwise

Author(s)

Wolfram Stacklies
**completeObs,nniRes-method**

Get the original data with missing values replaced with predicted values.

**Description**

Get the original data with missing values replaced with predicted values.

**Usage**

```r
completeObs(object, ...)```

**Arguments**

- `object`: object to fetch complete data from
- `...`: Not used

**Value**

Completed data (matrix)

**Author(s)**

Henning Redestig

---

**cvseg**

Get CV segments

**Description**

Get cross-validation segments that have (as far as possible) the same ratio of all classes (if classes are present)

**Usage**

```r
cvseg(x, fold = 7, seed = NULL)
```

**Arguments**

- `x`: a factor, character or numeric vector that describes class membership of a set of items, or, a numeric vector indicating unique indices of items, or, a numeric of length 1 that describes the number of items to segment (without any classes)
- `fold`: the desired number of segments
- `seed`: randomization seed for reproducibility

**Value**

a list where each element is a set of indices that defines the CV segment.
Author(s)

Henning Redestig

See Also

the cvsegments function in the pls package

Examples

```r
seg <- cvseg(iris$Species, 10)
sapply(seg, function(s) table(iris$Species[s]))
cvseg(20, 10)
```

---

**cvstat.pcaRes-method**  
*Get cross-validation statistics (e.g. $Q^2$).*

Description

Get cross-validation statistics (e.g. $Q^2$).

Usage

```r
cvstat(object, ...)
```

Arguments

- `object`  
  pcaRes object

- `...`  
  not used

Value

vector CV statistics

Author(s)

Henning Redestig
**deletediagonals**  
*Delete diagonals*

**Description**  
Replace a diagonal of elements of a matrix with NA

**Usage**  
`deletediagonals(x, diagonals = 1)`

**Arguments**
- `x`  
The matrix
- `diagonals`  
The diagonal to be replaced, i.e. the first, second and so on when looking at the fat version of the matrix (transposed or not) counting from the bottom. Can be a vector to delete more than one diagonal.

**Details**  
Used for creating artificial missing values in matrices without causing any full row or column to be completely missing

**Value**  
The original matrix with some values missing

**Author(s)**  
Henning Redestig

---

**derrorHierarchic**  
*Later*

**Description**  
Later

**Usage**  
`derrorHierarchic(nlnet, trainIn, trainOut)`

**Arguments**
- `nlnet`  
the nlnet
- `trainIn`  
training data
- `trainOut`  
fitted data
### dim.pcaRes

**Description**

Dimensions of a PCA model

**Usage**

```r
## S3 method for class 'pcaRes'
dim(x)
```

**Arguments**

- `x` a pcaRes object

**Value**

Get the dimensions of this PCA model

**Author(s)**

Henning Redestig

---

### DModX.pcaRes-method

**Description**

Distance to the model of X-space.

**Usage**

```r
DModX(object, dat, newdata=FALSE, type=c("normalized","absolute"), ...)
```

**Arguments**

- `object` a pcaRes object
- `dat` the original data, taken from `completeObs` if left missing.
- `newdata` logical indicating if this data was part of the training data or not. If it was, it is adjusted by a near one factor $v = (N/(N - A - A0))^{-1}$
- `type` if absolute or normalized values should be given. Normalized values are adjusted to the the total RSD of the model.
- `...` Not used
**Details**

Measures how well described the observations are, i.e. how well they fit in the mode. High DModX indicate a poor fit. Defined as:

\[
\sqrt{\frac{SSE_i}{SSE}} \sqrt{\frac{K - A}{(N - A)}}
\]

For observation \(i\), in a model with \(A\) components, \(K\) variables and \(N\) observations. \(SSE\) is the squared sum of the residuals. \(A_0\) is 1 if model was centered and 0 otherwise. DModX is claimed to be approximately F-distributed and can therefore be used to check if an observation is significantly far away from the PCA model assuming normally distributed data.

Pass original data as an argument if the model was calculated with `completeObs=FALSE`.

**Value**

A vector with distances from observations to the PCA model

**Author(s)**

Henning Redestig

**References**


**Examples**

```r
data(iris)
pcIr <- pca(iris[,1:4])
with(iris, plot(DModX(pcIr)~Species))
```

---

**Description**

Later

**Usage**

`errorHierarchic(nlnet, trainIn, trainOut)`

**Arguments**

- `nlnet`: The nlnet
- `trainIn`: training data
- `trainOut`: fitted data

**Value**

`error`
Author(s)
Henning Redestig, Matthias Scholz

**fitted-methods**

*Extract fitted values from PCA.*

**Description**

Fitted values of a PCA model

**Usage**

```r
# S3 method for class 'pcaRes'
fitted(object, data = NULL, nPcs = nP(object),
        pre = TRUE, post = TRUE, ...)

# S4 method for signature 'pcaRes'
fitted(object, data = NULL, nPcs = nP(object),
        pre = TRUE, post = TRUE, ...)
```

**Arguments**

- `object`: the pcaRes object of interest.
- `data`: For standard PCA methods this can safely be left null to get scores x loadings but if set, then the scores are obtained by projecting provided data onto the loadings. If data contains missing values the result will be all NA. Non-linear PCA is an exception, here if data is NULL then data is set to the completeObs and propagated through the network.
- `nPcs`: The number of PC’s to consider
- `pre`: pre-process data based on the pre-processing chosen for the PCA model
- `post`: unpre-process the final data (add the center back etc to get the final estimate)
- `...`: Not used

**Details**

This function extracts the fitted values from a pcaRes object. For PCA methods like SVD, Nipals, PPCA etc this is basically just the scores multiplied by the loadings and adjusted for pre-processing. for non-linear PCA the original data is propagated through the network to obtain the approximated data.

**Value**

A matrix representing the fitted data

**Author(s)**
Henning Redestig

**Examples**

```r
pc <- pca(iris[,1:4], nPcs=4, center=TRUE, scale="uv")
sum( (fitted(pc) - iris[,1:4])^2 )
```
forkNlpcaNet

Description
Complete copy of nlpca net object

Usage
forkNlpcaNet(nlnet)

Arguments
nlnet a nlnet

Value
A copy of the input nlnet

Author(s)
Henning Redestig

getHierarchicIdx

Description
Index in hierarchy

Usage
getHierarchicIdx(hierarchicNum)

Arguments
hierarchicNum A number

Value
...

Author(s)
Henning Redestig, Matthias Scholz
helix  

*A helix structured toy data set*

**Description**

Simulated data set looking like a helix

**Usage**

```r
data(helix)
```

**Details**

A matrix containing 1000 observations (rows) and three variables (columns).

**Author(s)**

Henning Redestig

**References**


---

**kEstimate**  

*Estimate best number of Components for missing value estimation*

**Description**

Perform cross validation to estimate the optimal number of components for missing value estimation. Cross validation is done for the complete subset of a variable.

**Usage**

```r
kEstimate(Matrix, method = "ppca", evalPcs = 1:3, segs = 3, nruncv = 5, em = "q2", allVariables = FALSE, verbose = interactive(), ...)
```

**Arguments**

- **Matrix**  
  matrix – numeric matrix containing observations in rows and variables in columns

- **method**  
  character – of the methods found with pcaMethods() The option llsImputeAll calls llsImpute with the allVariables = TRUE parameter.

- **evalPcs**  
  numeric – The principal components to use for cross validation or the number of neighbour variables if used with llsImpute. Should be an array containing integer values, eg. evalPcs = 1:10 or evalPcs = c(2, 5, 8). The NRMSEP or Q2 is calculated for each component.

- **segs**  
  numeric – number of segments for cross validation

- **nruncv**  
  numeric – Times the whole cross validation is repeated
kEstimate

character – The error measure. This can be nrmsep or q2

boolean – If TRUE, the NRMSEP is calculated for all variables. If FALSE, only the incomplete ones are included. You maybe want to do this to compare several methods on a complete data set.

boolean – If TRUE, some output like the variable indexes are printed to the console each iteration.

Further arguments to pca or nni

Details

The assumption hereby is that variables that are highly correlated in a distinct region (here the non-missing observations) are also correlated in another (here the missing observations). This also implies that the complete subset must be large enough to be representative. For each incomplete variable, the available values are divided into a user defined number of cv-segments. The segments have equal size, but are chosen from a random equal distribution. The non-missing values of the variable are covered completely. PPCA, BPCA, SVDimpute, Nipals PCA, llsImpute an NLPCA may be used for imputation.

The whole cross validation is repeated several times so, depending on the parameters, the calculations can take very long time. As error measure the NRMSEP (see Feten et. al, 2005) or the Q2 distance is used. The NRMSEP basically normalises the RMSD between original data and estimate by the variable-wise variance. The reason for this is that a higher variance will generally lead to a higher estimation error. If the number of samples is small, the variable - wise variance may become an unstable criterion and the Q2 distance should be used instead. Also if variance normalisation was applied previously.

The method proceeds variable - wise, the NRMSEP / Q2 distance is calculated for each incomplete variable and averaged afterwards. This allows to easily see for wich set of variables missing value imputation makes senes and for wich set no imputation or something like mean-imputation should be used. Use kEstimateFast or Q2 if you are not interested in variable wise CV performance estimates.

Run time may be very high on large data sets. Especially when used with complex methods like BPCA or Nipals PCA. For PPCA, BPCA, Nipals PCA and NLPCA the estimation method is called \((v_{miss} \cdot segs \cdot nruncv)\) times as the error for all numbers of principal components can be calculated at once. For LLSimpute and SVDimpute this is not possible, and the method is called \((v_{miss} \cdot segs \cdot nruncv \cdot length(evalPcs))\) times. This should still be fast for LLSimpute because the method allows to choose to only do the estimation for one particular variable. This saves a lot of iterations. Here, \(v_{miss}\) is the number of variables showing missing values.

As cross validation is done variable-wise, in this function Q2 is defined on single variables, not on the entire data set. This is Q2 is calculated as as \(\frac{\sum (x-xe)^2}{\sum (x^2)}\), where \(x\) is the currently used variable and \(xe\) it’s estimate. The values are then averaged over all variables. The NRMSEP is already defined variable-wise. For a single variable it is then \(\sqrt{\frac{\sum (x-xe)^2}{n \cdot \text{var}(x)}}\), where \(x\) is the variable and \(xe\) it’s estimate, \(n\) is the length of \(x\). The variable wise estimation errors are returned in parameter variableWiseError.

Value

A list with:

number of PCs or k for which the minimal average NRMSEP or the maximal Q2 was obtained.
**kEstimateFast**

Estimate best number of Components for missing value estimation

**Description**

This is a simple estimator for the optimal number of components when applying PCA or LLSimpute for missing value estimation. No cross validation is performed, instead the estimation quality is defined as Matrix[!missing] - Estimate[!missing]. This will give a relatively rough estimate, but the number of iterations equals the length of the parameter evalPcs. Does not work with LLSimpute!! As error measure the NRMSEP (see Feten et. al, 2005) or the Q2 distance is used. The NRMSEP basically normalises the RMSD between original data and estimate by the variable-wise variance. The reason for this is that a higher variance will generally lead to a higher estimation error. If the number of samples is small, the gene-wise variance may become an unstable criterion and the Q2 distance should be used instead. Also if variance normalisation was applied previously.

**eError**
an array of of size length(evalPcs). Contains the average error of the cross validation runs for each number of components.

**variableWiseError**
Matrix of size incomplete_variables x length(evalPcs). Contains the NRMSEP or Q2 distance for each variable and each number of PCs. This allows to easily see for which variables imputation makes sense and for which one it should not be done or mean imputation should be used.

**evalPcs**
The evaluated numbers of components or number of neighbours (the same as the evalPcs input parameter).

**variableIx**
Index of the incomplete variables. This can be used to map the variable wise error to the original data.

**Author(s)**
Wolfram Stacklies

**See Also**
kEstimateFast, Q2, pca, nni.

**Examples**

```r
## Load a sample metabolite dataset with 5\% missing values (metaboliteData)
data(metaboliteData)
# Do cross validation with ppca for component 2:4
esti <- kEstimate(metaboliteData, method = "ppca", evalPcs = 2:4, nruncv=1, em="nrmsep")
# Plot the average NRMSEP
barplot(drop(estim$eError), xlab = "Components",ylab = "NRMSEP (1 iterations)")
# The best result was obtained for this number of PCs:
print(estim$bestNPcs)
# Now have a look at the variable wise estimation error
barplot(drop(estim$variableWiseError[, which(estim$evalPcs == estim$bestNPcs)]), 
       xlab = "Incomplete variable Index", ylab = "NRMSEP")
```
**Usage**

```r
kEstimateFast(Matrix, method = "ppca", evalPcs = 1:3, em = "nrmsep", allVariables = FALSE, verbose = interactive(), ...)
```

**Arguments**

- **Matrix**: matrix – numeric matrix containing observations in rows and variables in columns
- **method**: character – a valid pca method (see `pca`).
- **evalPcs**: numeric – The principal components to use for cross validation or cluster sizes if used with llsImpute. Should be an array containing integer values, e.g. `evalPcs = 1:10` or `evalPcs = c(2,5,8)`. The NRMSEP is calculated for each component.
- **em**: character – The error measure. This can be `nrmsep` or `q2`.
- **allVariables**: boolean – If TRUE, the NRMSEP is calculated for all variables. If FALSE, only the incomplete ones are included. You might want to do this to compare several methods on a complete data set.
- **verbose**: boolean – If TRUE, the NRMSEP and the variance are printed to the console each iteration.
- **...**: Further arguments to `pca`.

**Value**

- **list**: Returns a list with the elements:
  - `minNPcs` - number of PCs for which the minimal average NRMSEP was obtained
  - `eError` - an array of size `length(evalPcs)`. Contains the estimation error for each number of components.
  - `evalPcs` - The evaluated numbers of components or cluster sizes (the same as the `evalPcs` input parameter).

**Author(s)**

Wolfram Stacklies

**See Also**

- `kEstimate`

**Examples**

```r
data(metaboliteData)
# Estimate best number of PCs with ppca for component 2:4
esti <- kEstimateFast(t(metaboliteData), method = "ppca", evalPcs = 2:4, em="nrmsep")
barplot(drop(esti$eError), xlab = "Components", ylab = "NRMSEP (1 iterations)"
# The best k value is:
print(esti$minNPcs)
```
leverage.pcaRes-method

Extract leverages of a PCA model

Description

The leverages of PCA model indicate how much influence each observation has on the PCA model. Observations with high leverage has caused the principal components to rotate towards them. It can be used to extract both "unimportant" observations as well as picking potential outliers.

Usage

```r
## S4 method for signature 'pcaRes'
leverage(object)
```

Arguments

- `object` a `pcaRes` object

Details

Defined as $Tr(T(T' T)^{-1} T')$

Value

The observation leverages as a numeric vector

Author(s)

Henning Redestig

References

Introduction to Multi- and Megavariate Data Analysis using Projection Methods (PCA and PLS), L. Eriksson, E. Johansson, N. Kettaneh-Wold and S. Wold, Umetrics 1999, p. 466

Examples

```r
data(iris)
pcIr <- pca(iris[,1:4])
## versicolor has the lowest leverage
with(iris, plot(leverage(pcIr)~Species))
```
**lineSearch**  

*Line search for conjugate gradient*

**Description**

Line search for conjugate gradient

**Usage**

```r
lineSearch(nlnet, dw, e0, ttGuess, trainIn, trainOut, verbose)
```

**Arguments**

- `nlnet`: The nlnet
- `dw`: ...
- `e0`: ...
- `ttGuess`: ...
- `trainIn`: Training data
- `trainOut`: Fitted data
- `verbose`: logical, print messages

**Value**

...

**Author(s)**

Henning Redestig, Matthias Scholz

---

**linr**  

*Linear kernel*

**Description**

Linear kernel

**Usage**

```r
linr(x)
```

**Arguments**

- `x`: datum

**Value**

Input value

**Author(s)**

Henning Redestig, Matthias Scholz
listPcaMethods  List PCA methods

Description
Vector with current valid PCA methods

Usage
listPcaMethods(which = c("all", "linear", "nonlinear"))

Arguments
which the type of methods to get. E.g. only get the PCA methods based on the classical model where the fitted data is a direct multiplication of scores and loadings.

Value
A character vector with the current methods for doing PCA

Author(s)
Henning Redestig

llsImpute  LLSimpute algorithm

Description
Missing value estimation using local least squares (LLS). First, k variables (for Microarray data usually the genes) are selected by pearson, spearman or kendall correlation coefficients. Then missing values are imputed by a linear combination of the k selected variables. The optimal combination is found by LLS regression. The method was first described by Kim et al, Bioinformatics, 21(2),2005.

Usage
llsImpute(Matrix, k = 10, center = FALSE, completeObs = TRUE, correlation = "pearson", allVariables = FALSE, maxSteps = 100, xval = NULL, verbose = FALSE, ...)

Arguments
Matrix matrix – Data containing the variables (genes) in columns and observations (samples) in rows. The data may contain missing values, denoted as NA.
k numeric – Cluster size, this is the number of similar genes used for regression.
center boolean – Mean center the data if TRUE
completeObs boolean – Return the estimated complete observations if TRUE. This is the input data with NA values replaced by the estimated values.
**llsImpute**

- **correlation** character – How to calculate the distance between genes. One out of pearson, kendall, spearman, see also help("cor").
- **allVariables** boolean – Use only complete genes to do the regression if TRUE, all genes if FALSE.
- **maxSteps** numeric – Maximum number of iteration steps if allGenes = TRUE.
- **xval** numeric – Use LLSimpute for cross validation. xval is the index of the gene to estimate, all other incomplete genes will be ignored if this parameter is set. We do not consider them in the cross-validation.
- **verbose** boolean – Print step number and relative change if TRUE and allVariables = TRUE

Details

Missing values are denoted as NA.

It is not recommended to use this function directly but rather to use the nni() wrapper function. The methods provide two ways for missing value estimation, selected by the allVariables option. The first one is to use only complete variables for the regression. This is preferable when the number of incomplete variables is relatively small.

The second way is to consider all variables as candidates for the regression. Hereby missing values are initially replaced by the columns wise mean. The method then iterates, using the current estimate as input for the regression until the change between new and old estimate falls below a threshold (0.001).

Value

- **nniRes** Standard nni (nearest neighbour imputation) result object of this package. See nniRes for details.

Note

Each step the generalized inverse of a miss x k matrix is calculated. Where miss is the number of missing values in variable j and k the number of neighbours. This may be slow for large values of k and / or many missing values. See also help("ginv").

Author(s)

Wolfram Stacklies

References


See Also

- pca, nniRes, nni.
## Load a sample metabolite dataset (metaboliteData) with already 5% of data missing
```r
data(metaboliteData)
result <- llsImpute(metaboliteData, k = 10, correlation="pearson", allVariables=TRUE)
cObs <- completeObs(result)
```

### Description
Crude way to unmask the function with the same name from stats

### Usage
```r
loadings(object, ...)
```

### Arguments
- `object` any object
- `...` not used

### Value
The loadings

### Author(s)
Henning Redestig

### Description
Get loadings from a pcaRes object

### Usage
```r
loadings(object, ...)
```
loadings.pcaRes

Arguments

object  
a pcaRes object
...
not used

Value

The loadings as a matrix

Author(s)

Henning Redestig

See Also

loadings.pcaRes

Description

Get loadings from a pcaRes object

Usage

## S3 method for class 'pcaRes'
loadings(object, ...)

Arguments

object  
a pcaRes object
...
not used

Value

The loadings as a matrix

Author(s)

Henning Redestig
**metaboliteData**

**Description**

A complete subset from a larger metabolite data set. This is the original, complete data set and can be used to compare estimation results created with the also provided incomplete data (called metaboliteData).

**Details**

A matrix containing 154 observations (rows) and 52 metabolites (columns). The data was created during an in house Arabidopsis coldstress experiment.

**Author(s)**

Wolfram Stacklies

**References**


**See Also**

metaboliteDataComplete

---

**metaboliteDataComplete**

**Description**

A incomplete metabolite data set from an Arabidopsis coldstress experiment.

**Details**

A incomplete subset from a larger metabolite data set. This is the original, complete data set and can be used to compare estimation results created with the also provided incomplete data (called metaboliteData). The data contains 5% of artificially created uniformly distributed missing values. The data was created during an in house Arabidopsis coldstress experiment.

**Author(s)**

Wolfram Stacklies
method.pcaRes-method

References

See Also

*metaboliteData*

---

**method, pcaRes-method**

*Get the used PCA method*

**Description**
Get the used PCA method

**Usage**

```
method(object, ...)
```

**Arguments**

- `object`
  pcaRes object
- `...`
  Not used

**Value**

The used pca method

**Author(s)**

Henning Redestig

---

**nipalsPca**

*NIPALS PCA*

**Description**

PCA by non-linear iterative partial least squares

**Usage**

```
nipalsPca(Matrix, nPcs = 2, varLimit = 1, maxSteps = 5000, threshold = 1e-06, ...)
```
nipalsPca

Arguments

Matrix: Pre-processed (centered, scaled) numerical matrix samples in rows and variables as columns.

nPcs: Number of components that should be extracted.

varLimit: Optionally the ratio of variance that should be explained. nPcs is ignored if varLimit < 1

maxSteps: Defines how many iterations can be done before algorithm should abort (happens almost exclusively when there were some wrong in the input data).

threshold: The limit condition for judging if the algorithm has converged or not, specifically if a new iteration is done if \((T_{old} - T)^T(T_{old} - T) > \text{limit}\).

...: Only used for passing through arguments.

Details

Can be used for computing PCA on a numeric matrix using either the NIPALS algorithm which is an iterative approach for estimating the principal components extracting them one at a time. NIPALS can handle a small amount of missing values. It is not recommended to use this function directly but rather to use the pca() wrapper function.

Value

A pcaRes object.

Author(s)

Henning Redestig

References


See Also

prcomp, princomp, pca

Examples

data(metaboliteData)
mat <- prep(t(metaboliteData))
pc <- nipalsPca(mat, nPcs=2)
## better use pca()
pc <- pca(t(metaboliteData), method="nipals", nPcs=2)
Non-linear PCA

Description

Neural network based non-linear PCA

Usage

nlpca(Matrix, nPcs = 2, maxSteps = 2 * prod(dim(Matrix)),
      unitsPerLayer = NULL, functionsPerLayer = NULL,
      weightDecay = 0.001, weights = NULL, verbose = interactive(), ...)

Arguments

Matrix matrix — Preprocessed data with the variables in columns and observations in rows. The data may contain missing values, denoted as NA.

nPcs numeric — Number of components to estimate. The preciseness of the missing value estimation depends on the number of components, which should resemble the internal structure of the data.

maxSteps numeric — Number of estimation steps. Default is based on a generous rule of thumb.

unitsPerLayer The network units, example: c(2,4,6) for two input units 2 feature units (principal components), one hidden layer for non-linearity and three output units (original amount of variables).

functionsPerLayer The function to apply at each layer eg. c("linr", "tanh", "linr")

weightDecay Value between 0 and 1.

weights Starting weights for the network. Defaults to uniform random values but can be set specifically to make algorithm deterministic.

verbose boolean — nlpca prints the number of steps and warning messages if set to TRUE. Default is interactive().

... Reserved for future use. Not passed on anywhere.

Details

Artificial Neural Network (MLP) for performing non-linear PCA. Non-linear PCA is conceptually similar to classical PCA but theoretically quite different. Instead of simply decomposing our matrix (X) to scores (T) loadings (P) and an error (E) we train a neural network (our loadings) to find a curve through the multidimensional space of X that describes a much variance as possible. Classical ways of interpreting PCA results are thus not applicable to NLPCA since the loadings are hidden in the network. However, the scores of components that lead to low cross-validation errors can still be interpreted via the score plot. Unfortunately this method depend on slow iterations which currently are implemented in R only making this method extremely slow. Furthermore, the algorithm does not by itself decide when it has converged but simply does ‘maxSteps’ iterations.

Value

Standard PCA result object used by all PCA-based methods of this package. Contains scores, loadings, data mean and more. See pcaRes for details.
Author(s)

Based on a matlab script by Matthias Scholz and ported to R by Henning Redestig

References


Examples

```r
## Data set with three variables where data points constitute a helix
data(helix)
helixNA <- helix
## not a single complete observation
helixNA <- t(apply(helix, 1, function(x) { x[sample(1:3, 1)] <- NA; x}))
## 50 steps is not enough, for good estimation use 1000
helixNlPca <- pca(helixNA, nPcs=1, method="nlpca", maxSteps=50)
fittedData <- fitted(helixNlPca, helixNA)
plot(fittedData[which(is.na(helixNA))], helix[which(is.na(helixNA))])
## compared to solution by Nipals PCA which cannot extract non-linear patterns
helixNipPca <- pca(helixNA, nPcs=2)
fittedData <- fitted(helixNipPca)
plot(fittedData[which(is.na(helixNA))], helix[which(is.na(helixNA))])
```

nmissing,pcaRes-method

Missing values

Description

Missing values

Usage

```r
nmissing(object, ...)
```

Arguments

- `object`: pcaRes object
- `...`: Not used

Value

Get the number of missing values

Author(s)

Henning Redestig
Nearest neighbour imputation

Description

Wrapper function for imputation methods based on nearest neighbour clustering. Currently llsImpute only.

Usage

nni(object, method = c("llsImpute"), subset = numeric(), ...)

Arguments

object Numerical matrix with (or an object coercible to such) with samples in rows and variables as columns. Also takes ExpressionSet in which case the transposed expression matrix is used.
method For convenience one can pass a large matrix but only use the variable specified as subset. Can be colnames or indices.
subset Currently "llsImpute" only.
... Further arguments to the chosen method.

Details

This method is wrapper function to llsImpute, See documentation for link{llsImpute}.

Value

A clusterRes object. Or a list containing a clusterRes object as first and an ExpressionSet object as second entry if the input was of type ExpressionSet.

Author(s)

Wolfram Stacklies

See Also

llsImpute, pca

Examples

data(metaboliteData)
llsRes <- nni(metaboliteData, k=6, method="llsImpute", allGenes=TRUE)
nniRes

Class for representing a nearest neighbour imputation result

Description

This is a class representation of nearest neighbour imputation (nni) result

Details

Creating Objects

new("nniRes", completeObs=[the estimated complete observations], k=[cluster size], nObs=[amount of observations], nVar=[amount of variables], centered=[was the data centered before running LLSimpute], center=[original means], method=[method used to perform clustering], missing=[amount of NAs])

Slots

- completeObs "matrix", the estimated complete observations
- nObs "numeric", amount of observations
- nVar "numeric", amount of variables
- correlation "character", the correlation method used (pearson, kendall or spearman)
- centered "logical", data was centered or not
- center "numeric", the original variable centers
- k "numeric", cluster size
- method "character", the method used to perform the clustering
- missing "numeric", the total amount of missing values in original data

Methods

- print Print function

Author(s)

Wolfram Stacklies

nObs,pcaRes-method

Get the number of observations used to build the PCA model.

Description

Get the number of observations used to build the PCA model.

Usage

nObs(object, ...)
**Arguments**

<table>
<thead>
<tr>
<th>object</th>
<th>pcaRes object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Not used</td>
</tr>
</tbody>
</table>

**Value**

Number of observations

**Author(s)**

Henning Redestig

---

**nP,pcaRes-method**  \(\text{Get number of PCs}\)

**Description**

Get number of PCs

**Usage**

\[\text{nP}(\text{object, ...})\]

**Arguments**

<table>
<thead>
<tr>
<th>object</th>
<th>pcaRes object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>not used</td>
</tr>
</tbody>
</table>

**Value**

Number of PCs

**Author(s)**

Henning Redestig

---

**nPcs,pcaRes-method**  \(\text{Get number of PCs}\)

**Description**

Get number of PCs

**Usage**

\[\text{nPcs}(\text{object, ...})\]
nVar,pcaRes-method

Arguments

object pcaRes object
...

Value

Number of PCs

Note

Try to use \link{nP} instead since nPcs tend to clash with argument names.

Author(s)

Henning Redestig

---

Get the number of variables used to build the PCA model.

Description

Get the number of variables used to build the PCA model.

Usage

nVar(object, ...)

Arguments

object pcaRes object
...

Value

Number of variables

Author(s)

Henning Redestig
**optiAlgCgd**

Conjugate gradient optimization

**Description**

Conjugate gradient optimization

**Usage**

```r
optiAlgCgd(nlnet, trainIn, trainOut, verbose = FALSE)
```

**Arguments**

- `nlnet`: The nlnet
- `trainIn`: Training data
- `trainOut`: fitted data
- `verbose`: logical, print messages

**Value**

...

**Author(s)**

Henning Redestig, Matthias Scholz

---

**orth**

Calculate an orthonormal basis

**Description**

ONB = orth(mat) is an orthonormal basis for the range of matrix mat. That is, ONB’ * ONB = I, the columns of ONB span the same space as the columns of mat, and the number of columns of ONB is the rank of mat.

**Usage**

```r
orth(mat, skipInac = FALSE)
```

**Arguments**

- `mat`: matrix to calculate orthonormal base
- `skipInac`: do not include components with precision below .Machine$double.eps if TRUE

**Value**

orthonormal basis for the range of matrix

**Author(s)**

Wolfram Stacklies
pca

Perform principal component analysis

Description
Perform PCA on a numeric matrix for visualisation, information extraction and missing value imputation.

Usage
pca(object, method, nPcs = 2, scale = c("none", "pareto", "vector", "uv"), center = TRUE, completeObs = TRUE, subset = NULL, cv = c("none", "q2"), ...)

Arguments
- object: Numerical matrix with (or an object coercible to such) with samples in rows and variables as columns. Also takes ExpressionSet in which case the transposed expression matrix is used. Can also be a data frame in which case all numeric variables are used to fit the PCA.
- method: One of the methods reported by listPcaMethods(). Can be left missing in which case the svd PCA is chosen for data without missing values and nipalsPca for data with missing values.
- nPcs: Number of principal components to calculate.
- scale: Scaling, see prep.
- center: Centering, see prep.
- completeObs: Sets the completeObs slot on the resulting pcaRes object containing the original data with but with all NAs replaced with the estimates.
- subset: A subset of variables to use for calculating the model. Can be column names or indices.
- cv: character naming a the type of cross-validation to be performed.
- ...: Arguments to prep, the chosen pca method and Q2.

Details
This method is wrapper function for the following set of pca methods:
- svd: Uses classical prcomp. See documentation for svdPca.
- nipals: An iterative method capable of handling small amounts of missing values. See documentation for nipalsPca.
- rnipals: Same as nipals but implemented in R.
- bpcapca: An iterative method using a Bayesian model to handle missing values. See documentation for bpcapca.
- ppca: An iterative method using a probabilistic model to handle missing values. See documentation for ppca.
- svdImpute: Uses expectation maximization to perform SVD PCA on incomplete data. See documentation for svdImpute.

Scaling and centering is part of the PCA model and handled by prep.
Value

A pcaRes object.

Author(s)

Wolfram Stacklies, Henning Redestig

References


See Also

prcomp, princomp, nipalsPca, svdPca

Examples

data(iris)
## Usually some kind of scaling is appropriate
pcIr <- pca(iris, method="svd", nPcs=2)
pcIr <- pca(iris, method="nipals", nPcs=3, cv="q2")
## Get a short summary on the calculated model
summary(pcIr)
plot(pcIr)
## Scores and loadings plot
slplot(pcIr, sl=as.character(iris[,5]))

## use an expressionset and ggplot
data(sample.ExpressionSet)
p <- pca(sample.ExpressionSet)
df <- merge(scores(p), pData(sample.ExpressionSet), by=0)
library(ggplot2)
ggplot(df, aes(PC1, PC2, shape=sex, color=type)) +
  geom_point() +
  xlab(paste("PC1", pc@R2[1] * 100, "% of the variance")) +
  ylab(paste("PC2", pc@R2[2] * 100, "% of the variance"))
Provides Bayesian PCA, Probabilistic PCA, Nipals PCA, Inverse Non-Linear PCA and the conventional SVD PCA. A cluster based method for missing value estimation is included for comparison. BPCA, PPCA and NipalsPCA may be used to perform PCA on incomplete data as well as for accurate missing value estimation. A set of methods for printing and plotting the results is also provided. All PCA methods make use of the same data structure (pcaRes) to provide a unique interface to the PCA results. Developed at the Max-Planck Institute for Molecular Plant Physiology, Golm, Germany, RIKEN Plant Science Center Yokohama, Japan, and CAS-MPG Partner Institute for Computational Biology (PICB) Shanghai, P.R. China

Author(s)

Wolfram Stacklies, Henning Redestig

---

pcaMethods-deprecated

Deprecated methods for pcaMethods

Description

plotR2 Lack of relevance for this plot and the fact that it can not show cross-validation based diagnostics in the same plot makes it redundant with the introduction of a dedicated plot function for pcaRes. The new plot only shows R2cum but the result is pretty much the same.

Author(s)

Henning Redestig

---

pcaNet

Class representation of the NLPCA neural net

Description

This is a class representation of a non-linear PCA neural network. The nlpcaNet class is not meant for user-level usage.
Details

Creating Objects

new("nlpcaNet", net=[the network structure], hierarchic=[hierarchic design], fct=[the functions at each layer], fkt=[the functions used for forward propagation], weightDecay=[incremental decrease of weight changes over iterations (between 0 and 1)], featureSorting=[sort features or not], dataDist=[represents the present values], inverse=[net is inverse mode or not], fCount=[amount of times features were sorted], componentLayer=[which layer is the 'bottleneck' (principal components)], erro=[the used error function], gradient=[the used gradient method], weights=[the present weights], maxIter=[the amount of iterations that was done], scalingFactor=[the scale of the original matrix])

Slots

net "matrix", matrix showing the representation of the neural network, e.g. (2,4,6) for a network with two features, a hidden layer and six output neurons (original variables).

hierarchic "list", the hierarchic design of the network, holds 'idx' (), 'var' () and layer (which layer is the principal component layer).

fct "character", a vector naming the functions that will be applied on each layer. "linr" is linear (i.e.) standard matrix products and "tanh" means that the arcus tangens is applied on the result of the matrix product (for non-linearity).

fkt "character", same as fct but the functions used during back propagation.

weightDecay "numeric", the value that is used to incrementally decrease the weight changes to ensure convergence.

featureSorting "logical", indicates if features will be sorted or not. This is used to make the NLPCA assume properties closer to those of standard PCA were the first component is more important for reconstructing the data than the second component.

dataDist "matrix", a matrix of ones and zeroes indicating which values will add to the error.

inverse "logical", network is inverse mode (currently only inverse is supported) or not. Eg. the case when we have truly missing values and wish to impute them.

fCount "integer", Counter for the amount of times features were really sorted.

componentLayer "numeric", the index of 'net' that is the component layer.

ero = "function", the used error function. Currently only one is provided errorHierarchic.

gradient "function", the used gradient function. Currently only one is provided derrorHierarchic.

weights "list", A list holding managements of the weights. The list has two functions, weights$current() and weights$set() which access a matrix in the local environment of this object.

maxIter "integer", the amount of iterations used to train this network.

scalingFactor "numeric", training the network is best made with 'small' values so the original data is scaled down to a suitable range by division with this number.

Methods

vector2matrices Returns the weights in a matrix representation.

Author(s)

Henning Redestig

See Also

nlpca
\textbf{ pcaRes }  
\textit{Class for representing a PCA result}  

\textbf{Description}  
This is a class representation of a PCA result.

\textbf{Details}  
\textbf{Creating Objects}  
\texttt{new("pcaRes",scores=[the scores],loadings=[the loadings],nPcs=[amount of PCs],R2cum=[cumulative R2],nObs=[amount of observations],nVar=[amount of variables],R2=[R2 for each individual PC],sDev=[stdev for each individual PC],centered=[was data centered],center=[original means],varLimit=[what variance limit was exceeded],method=[method used to calculate PCA],missing=[amount of NAs],completeObs=[estimated complete observations])}  

\textbf{Slots}  
- \texttt{scores} "matrix", the calculated scores  
- \texttt{loadings} "matrix", the calculated loadings  
- \texttt{R2cum} "numeric", the cumulative R2 values  
- \texttt{sDev} "numeric", the individual standard deviations of the score vectors  
- \texttt{R2} "numeric", the individual R2 values  
- \texttt{cvstat} "numeric", cross-validation statistics  
- \texttt{nObs} "numeric", number of observations  
- \texttt{nVar} "numeric", number of variables  
- \texttt{centered} "logical", data was centered or not  
- \texttt{center} "numeric", the original variable centers  
- \texttt{scaled} "logical", data was scaled or not  
- \texttt{scl} "numeric", the original variable scales  
- \texttt{varLimit} "numeric", the exceeded variance limit  
- \texttt{nPcs,nP} "numeric", the number of calculated PCs  
- \texttt{method} "character", the method used to perform PCA  
- \texttt{missing} "numeric", the total amount of missing values in original data  
- \texttt{completeObs} "matrix", the estimated complete observations  
- \texttt{network} "nlpcaNet", the network used by non-linear PCA  

\textbf{Methods (not necessarily exhaustive)}  
- \texttt{print} Print function  
- \texttt{summary} Extract information about PC relevance  
- \texttt{screeplot} Plot a barplot of standard deviations for PCs  
- \texttt{slplot} Make a side by side score and loadings plot  
- \texttt{nPcs} Get the number of PCs
plot.pcaRes

- `nObs` Get the number of observations
- `cvstat` Cross-validation statistics
- `nVar` Get the number of variables
- `loadings` Get the loadings
- `scores` Get the scores
- `dim` Get the dimensions (number of observations, number of features)
- `centered` Get a logical indicating if centering was done as part of the model
- `center` Get the averages of the original variables.
- `completeObs` Get the imputed data set
- `method` Get a string naming the used PCA method
- `sDev` Get the standard deviations of the PCs
- `scaled` Get a logical indicating if scaling was done as part of the model
- `scl` Get the scales of the original variables
- `R2cum` Get the cumulative R2

**Author(s)**

Henning Redestig

---

**Description**

Plot the computed diagnostics of PCA model to get an idea of their importance. Note though that the standard screeplot shows the standard deviations for the PCs this method shows the R2 values which empirically shows the importance of the P's and is thus applicable for any PCA method rather than just SVD based PCA.

**Usage**

```r
## S3 method for class 'pcaRes'
plot(x, y = NULL, main = deparse(substitute(object)),
     col = gray(c(0.9, 0.5)), ...)
```

**Arguments**

- `x` pcaRes The pcaRes object.
- `y` not used
- `main` title of the plot
- `col` Colors of the bars
- `...` further arguments to barplot

**Details**

If cross-validation was done for the PCA the plot will also show the CV based statistics. A common rule-of-thumb for determining the optimal number of PCs is the PC where the CV diagnostic is at its maximum but not very far from $\hat{R}^2$. 
Value

None, used for side effect.

Author(s)

Henning Redestig

See Also

screeplot

Examples

data(metaboliteData)
pc <- pca(t(metaboliteData), nPcs=5, cv="q2", scale="uv")
plot(pc)

plotPcs

Plot many side by side scores XOR loadings plots

Description

A function that can be used to visualise many PCs plotted against each other

Usage

plotPcs(object, pcs = 1:nP(object), type = c("scores", "loadings"),
      sl = NULL, hotelling = 0.95, ...)

Arguments

object pcaRes a pcaRes object
pcs numeric which pcs to plot
type character Either "scores" or "loadings" for scores or loadings plot respectively
sl character Text labels to plot instead of a point, if NULL points are plotted
      instead of text
hotelling numeric Significance level for the confidence ellipse. NULL means that no
      ellipse is drawn.
...
Further arguments to pairs on which this function is based.

Details

Uses pairs to provide side-by-side plots. Note that this function only plots scores or loadings but
not both in the same plot.

Value

None, used for side effect.
ppca

Author(s)
Henning Redestig

See Also
prcomp, pca, princomp, slplot

Examples

data(iris)
pcIr <- pca(iris[,1:4], nPcs=3, method="svd")
plotPcs(pcIr, col=as.integer(iris[,4]) + 1)

ppca

Probabilistic PCA

Description
Implementation of probabilistic PCA (PPCA). PPCA allows to perform PCA on incomplete data and may be used for missing value estimation. This script was implemented after the Matlab version provided by Jakob Verbeek (see http://lear.inrialpes.fr/~verbeek/) and the draft “EM Algorithms for PCA and Sensible PCA” written by Sam Roweis.

Usage

ppca(Matrix, nPcs = 2, seed = NA, threshold = 1e-05,
     maxIterations = 1000, ...)

Arguments

Matrix
matrix – Data containing the variables in columns and observations in rows. The data may contain missing values, denoted as NA.

nPcs
numeric – Number of components to estimate. The preciseness of the missing value estimation depends on the number of components, which should resemble the internal structure of the data.

seed
numeric Set the seed for the random number generator. PPCA creates fills the initial loading matrix with random numbers chosen from a normal distribution. Thus results may vary slightly. Set the seed for exact reproduction of your results.

threshold
Convergence threshold.

maxIterations
the maximum number of allowed iterations

... Reserved for future use. Currently no further parameters are used.
Details

Probabilistic PCA combines an EM approach for PCA with a probabilistic model. The EM approach is based on the assumption that the latent variables as well as the noise are normal distributed.

In standard PCA data which is far from the training set but close to the principal subspace may have the same reconstruction error. PPCA defines a likelihood function such that the likelihood for data far from the training set is much lower, even if they are close to the principal subspace. This allows to improve the estimation accuracy.

A method called kEstimate is provided to estimate the optimal number of components via cross validation. In general few components are sufficient for reasonable estimation accuracy. See also the package documentation for further discussion on what kind of data PCA-based missing value estimation is advisable.

Complexity:
Runtime is linear in the number of data, number of data dimensions and number of principal components.

Convergence: The threshold indicating convergence was changed from 1e-3 in 1.2.x to 1e-5 in the current version leading to more stable results. For reproducability you can set the seed (parameter seed) of the random number generator. If used for missing value estimation, results may be checked by simply running the algorithm several times with changing seed, if the estimated values show little variance the algorithm converged well.

Value

Standard PCA result object used by all PCA-based methods of this package. Contains scores, loadings, data mean and more. See pcaRes for details.

Note

Requires MASS. It is not recommended to use this function directly but rather to use the pca() wrapper function.

Author(s)

Wolfram Stacklies

See Also

bpca,svdImpute,prcomp,nipalsPca,pca,pcaRes.

Examples

```r
## Load a sample metabolite dataset with 5% missing values (metaboliteData)
data(metaboliteData)
## Perform probabilistic PCA using the 3 largest components
result <- pca(t(metaboliteData), method="ppca", nPcs=3, seed=123)
## Get the estimated complete observations
cObs <- completeObs(result)
## Plot the scores
plotPcs(result, type = "scores")
```
predict-methods  

Predict values from PCA.

Description

Predict data using PCA model

Usage

```r
## S3 method for class 'pcaRes'
predict(object, newdata, pcs = nP(object), pre = TRUE,
         post = TRUE, ...)

## S4 method for signature 'pcaRes'
predict(object, newdata, pcs = nP(object),
         pre = TRUE, post = TRUE, ...)
```

Arguments

- `object`: pcaRes the pcaRes object of interest.
- `newdata`: matrix new data with same number of columns as the used to compute `object`.
- `pcs`: numeric The number of PC’s to consider
- `pre`: pre-process `newdata` based on the pre-processing chosen for the PCA model
- `post`: unpre-process the final data (add the center back etc)
- `...`: Not passed on anywhere, included for S3 consistency.

Details

This function extracts the predict values from a pcaRes object for the PCA methods SVD, Nipals, PPCA and BPCA. Newdata is first centered if the PCA model was and then scores (\(T\)) and data (\(X\)) is ‘predicted’ according to: \(\hat{T} = X_{new} \hat{P} X_{new} = \hat{T} P'\). Missing values are set to zero before matrix multiplication to achieve NIPALS like treatment of missing values.

Value

A list with the following components:

- `scores`: The predicted scores
- `x`: The predicted data

Author(s)

Henning Redestig

Examples

```r
data(iris)
hidden <- sample(nrow(iris), 50)
pcIr <- pca(iris[-hidden,1:4])
pcFull <- pca(iris[,1:4])
irisHat <- predict(pcIr, iris[hidden,1:4])
corr(irisHat$scores[,1], scores(pcFull)[hidden,1])
```
prep  
Pre-process a matrix for PCA

Description
Scaling and centering a matrix.

Usage
prep(object, scale = c("none", "pareto", "vector", "uv"),
    center = TRUE, eps = 1e-12, simple = TRUE, reverse = FALSE, ...)

Arguments
object  Numerical matrix (or an object coercible to such) with samples in rows and variables as columns. Also takes ExpressionSet in which case the transposed expression matrix is used.
scale  One of "UV" (unit variance \(a = a/\sigma_a\)) "vector" (vector normalisation \(b = b/||b||\)), "pareto" (sqrt UV) or "none" to indicate which scaling should be used to scale the matrix with \(a\) variables and \(b\) samples. Can also be a vector of scales which should be used to scale the matrix. NULL value is interpreted as "none".
center  Either a logical which indicates if the matrix should be mean centred or not, or a vector with averages which should be suntracted from the matrix. NULL value is interpreted as FALSE.
eps  Minimum variance, variable with lower variance are not scaled and warning is issued instead.
simple  Logical indicating if only the data should be returned or a list with the pre-processing statistics as well.
reverse  Logical indicating if matrix should be 'post-processed' instead by multiplying each column with its scale and adding the center. In this case, center and scale should be vectors with the statistics (no warning is issued if not, instead output becomes the same as input).
...
Only used for passing through arguments.

Details
Does basically the same as scale but adds some alternative scaling options and functionality for treating pre-processing as part of a model.

Value
A pre-processed matrix or a list with

center  a vector with the estimated centers
scale  a vector with the estimated scales
data  the pre (or post) processed data

Author(s)
Henning Redestig
Examples

```r
object <- matrix(rnorm(50), nrow=10)
res <- prep(object, scale="uv", center=TRUE, simple=FALSE)
obj <- prep(object, scale=res$scale, center=res$center)
## same as original
sum((object - prep(obj, scale=res$scale, center=res$center, rev=TRUE))^2)
```

Cross-validation for PCA

Description

Internal cross-validation can be used for estimating the level of structure in a data set and to optimise the choice of number of principal components.

Usage

```r
Q2(object, originalData = completeObs(object), fold = 5, nruncv = 1,
type = c("krzanowski", "impute"), verbose = interactive(),
variables = 1:nVar(object), ...)
```

Arguments

- **object**: A pcaRes object (result from previous PCA analysis.)
- **originalData**: The matrix (or ExpressionSet) that used to obtain the pcaRes object.
- **fold**: The number of groups to divide the data in.
- **nruncv**: The number of times to repeat the whole cross-validation
- **type**: krzanowski or imputation type cross-validation
- **verbose**: boolean If TRUE Q2 outputs a primitive progress bar.
- **variables**: indices of the variables to use during cross-validation calculation. Other variables are kept as they are and do not contribute to the total sum-of-squares.
- **...**: Further arguments passed to the `pca` function called within Q2.

Details

This method calculates $Q^2$ for a PCA model. This is the cross-validated version of $R^2$ and can be interpreted as the ratio of variance that can be predicted independently by the PCA model. Poor (low) $Q^2$ indicates that the PCA model only describes noise and that the model is unrelated to the true data structure. The definition of $Q^2$ is:

$$Q^2 = 1 - \frac{\sum_i^k \sum_j^n (x - \hat{x})^2}{\sum_i^k \sum_j^n x^2}$$

for the matrix $x$ which has $n$ rows and $k$ columns. For a given number of PC’s $x$ is estimated as $\hat{x} = TP'$ (T are scores and P are loadings). Although this defines the leave-one-out cross-validation this is not what is performed if fold is less than the number of rows and/or columns. In 'impute' type CV, diagonal rows of elements in the matrix are deleted and the re-estimated. In 'krzanowski' type CV, rows are sequentially left out to build fold PCA models which give the loadings. Then, columns are sequentially left out to build fold models for scores. By combining
scores and loadings from different models, we can estimate completely left out values. The two types may seem similar but can give very different results, Krzanowski typically yields more stable and reliable result for estimating data structure whereas impute is better for evaluating missing value imputation performance. Note that since Krzanowski CV operates on a reduced matrix, it is not possible estimate $Q^2$ for all components and the result vector may therefore be shorter than `nPcs(object)`.

**Value**

A matrix or vector with $Q^2$ estimates.

**Author(s)**

Henning Redestig, Ondrej Mikula

**References**


**Examples**

```r
data(iris)
x <- iris[,1:4]
pcIr <- pca(x, nPcs=3)
q2 <- Q2(pcIr, x)
barplot(q2, main="Krzanowski CV", xlab="Number of PCs", ylab=expression(Q^2))
## q2 for a single variable
q2(pcIr, x, variables=2)
pcIr <- pca(x, nPcs=3, method="nipals")
q2 <- Q2(pcIr, x, type="impute")
barplot(q2, main="Imputation CV", xlab="Number of PCs", ylab=expression(Q^2))
```

---

**R2cum, pcaRes-method**

*Cumulative $R^2$ is the total ratio of variance that is being explained by the model*

**Description**

Cumulative $R^2$ is the total ratio of variance that is being explained by the model.

**Usage**

```r
## S4 method for signature 'pcaRes'
R2cum(object, ...)
```

**Arguments**

- `object` a `pcaRes` model
- `...` Not used

**Value**

Get the cumulative $R^2$
**Author(s)**

Henning Redestig

**Description**

Flexible calculation of R2 goodness of fit.

**Usage**

```r
## S4 method for signature 'pcaRes'
R2VX(object, direction = c("variables", "observations", "complete"), data = completeObs(object), pcs = nP(object))
```

**Arguments**

- `object`: a PCA model object
- `direction`: choose between calculating R2 per variable, per observation or for the entire data with 'variables', 'observations' or 'complete'.
- `data`: the data used to fit the model
- `pcs`: the number of PCs to use to calculate R2

**Value**

A vector with R2 values

**Author(s)**

Henning Redestig

**Examples**

```r
R2VX(pca(iris))
```
**repmat**

Replicate and tile an array.

Description

Creates a large matrix B consisting of an M-by-N tiling of copies of A

Usage

```r
repmat(mat, M, N)
```
RnipalsPca

Arguments

mat numeric matrix
M number of copies in vertical direction
N number of copies in horizontal direction

Value

Matrix consisting of M-by-N tiling copies of input matrix

Author(s)

Wolfram Stacklies

RnipalsPca NIPALS PCA implemented in R

Description

PCA by non-linear iterative partial least squares

Usage

RnipalsPca(Matrix, nPcs = 2, varLimit = 1, maxSteps = 5000, threshold = 1e-06, verbose = interactive(), ...)

Arguments

Matrix Pre-processed (centered, scaled) numerical matrix samples in rows and variables as columns.
nPcs Number of components that should be extracted.
varLimit Optionally the ratio of variance that should be explained. nPcs is ignored if varLimit < 1
maxSteps Defines how many iterations can be done before algorithm should abort (happens almost exclusively when there were some wrong in the input data).
threshold The limit condition for judging if the algorithm has converged or not, specifically if a new iteration is done if \((T_{old} - T) \times (T_{old} - T)^T > \text{limit}\).
verbose Show simple progress information.
... Only used for passing through arguments.

Details

Can be used for computing PCA on a numeric matrix using either the NIPALS algorithm which is an iterative approach for estimating the principal components extracting them one at a time. NIPALS can handle a small amount of missing values. It is not recommended to use this function directly but rather to use the pca() wrapper function. There is a C++ implementation given as nipalsPca which is faster.

Value

A pcaRes object.
robustPca

PCA implementation based on robustSvd

Description

This is a PCA implementation robust to outliers in a data set. It can also handle missing values, it is however NOT intended to be used for missing value estimation. As it is based on robustSVD we will get an accurate estimation for the loadings also for incomplete data or for data with outliers. The returned scores are, however, affected by the outliers as they are calculated inputData X loadings. This also implies that you should look at the returned R2/R2cum values with caution. If the data show missing values, scores are calculated by just setting all NA - values to zero. This is not expected to produce accurate results. Please have also a look at the manual page for robustSvd. Thus this method should mainly be seen as an attempt to integrate robustSvd() into the framework of this package. Use one of the other methods coming with this package (like PPCA or BPCA) if you want to do missing value estimation. It is not recommended to use this function directly but rather to use the pca() wrapper function.

Usage

robustPca(Matrix, nPcs = 2, verbose = interactive(), ...)

Arguments

Matrix

matrix – Data containing the variables in columns and observations in rows. The data may contain missing values, denoted as NA.

nPcs

numeric – Number of components to estimate. The preciseness of the missing value estimation depends on the number of components, which should resemble the internal structure of the data.

verbose

boolean Print some output to the command line if TRUE

... 

Reserved for future use. Currently no further parameters are used
robustSvd

Details
The method is very similar to the standard prcomp() function. The main difference is that robustSvd() is used instead of the conventional svd() method.

Value
Standard PCA result object used by all PCA-based methods of this package. Contains scores, loadings, data mean and more. See pcaRes for details. are used.

Author(s)
Wolfram Stacklies

See Also
robustSvd, svd, prcomp, pcaRes.

Examples
```r
## Load a complete sample metabolite data set and mean center the data
data(metaboliteDataComplete)
mdc <- scale(metaboliteDataComplete, center=TRUE, scale=FALSE)
## Now create 5% of outliers.
cond <- runif(length(mdc)) < 0.05;
mdcOut <- mdc
mdcOut[cond] <- 10
## Now we do a conventional PCA and robustPca on the original and the data
## with outliers.
## We use center=FALSE here because the large artificial outliers would
## affect the means and not allow to objectively compare the results.
resSvd <- pca(mdc, method="svd", nPcs=10, center=FALSE)
resSvdOut <- pca(mdcOut, method="svd", nPcs=10, center=FALSE)
resRobPca <- pca(mdcOut, method="robustPca", nPcs=10, center=FALSE)
## Now we plot the results for the original data against those with outliers
## We can see that robustPca is hardly affected by the outliers.
plot(loadings(resSvd)[,1], loadings(resSvdOut)[,1])
plot(loadings(resSvd)[,1], loadings(resRobPca)[,1])
```

robustSvd

Alternating L1 Singular Value Decomposition

Description
A robust approximation to the singular value decomposition of a rectangular matrix is computed using an alternating L1 norm (instead of the more usual least squares L2 norm). As the SVD is a least-squares procedure, it is highly susceptible to outliers and in the extreme case, an individual cell (if sufficiently outlying) can draw even the leading principal component toward itself.

Usage
```r
robustSvd(x)
```
Arguments

x  A matrix whose SVD decomposition is to be computed. Missing values are allowed.

Details

See Hawkins et al (2001) for details on the robust SVD algorithm. Briefly, the idea is to sequentially estimate the left and right eigenvectors using an L1 (absolute value) norm minimization.

Note that the robust SVD is able to accommodate missing values in the matrix x, unlike the usual svd function.

Also note that the eigenvectors returned by the robust SVD algorithm are NOT (in general) orthogonal and the eigenvalues need not be descending in order.

Value

The robust SVD of the matrix is x = u d v'.

d  A vector containing the singular values of x.

u  A matrix whose columns are the left singular vectors of x.

v  A matrix whose columns are the right singular vectors of x.

Note

Two differences from the usual SVD may be noted. One relates to orthogonality. In the conventional SVD, all the eigenvectors are orthogonal even if not explicitly imposed. Those returned by the AL1 algorithm (used here) are (in general) not orthogonal. Another difference is that, in the L2 analysis of the conventional SVD, the successive eigen triples (eigenvalue, left eigenvector, right eigenvector) are found in descending order of eigenvalue. This is not necessarily the case with the AL1 algorithm. Hawkins et al (2001) note that a larger eigen value may follow a smaller one.

Author(s)

Kevin Wright, modifications by Wolfram Stacklies

References


See Also

svd, nipals for an alternating L2 norm method that also accommodates missing data.

Examples

```r
## Load a complete sample metabolite data set and mean center the data
data(metaboliteDataComplete)
mdc <- prep(metaboliteDataComplete, center=TRUE, scale="none")
## Now create 5% of outliers.
cond <- runif(length(mdc)) < 0.05;
mdcOut <- mdc
mdcOut[cond] <- 10
```
Now we do a conventional SVD and a robustSvd on both, the original and the data with outliers.
```
resSvd <- svd(mdc)
resSvdOut <- svd(mdcOut)
resRobSvd <- robustSvd(mdc)
resRobSvdOut <- robustSvd(mdcOut)
```
Now we plot the results for the original data against those with outliers.
```
plot(resSvd$v[,1], resSvdOut$v[,1])
plot(resRobSvd$v[,1], resRobSvdOut$v[,1])
```

### scaled, pcaRes-method

**Check if scaling was part of the PCA model**

**Description**

Check if scaling was part of the PCA model

**Usage**

```
scaled(object, ...)
```

**Arguments**

- `object`: pcaRes object
- `...`: Not used

**Value**

TRUE if scaling was part of the PCA model

**Author(s)**

Henning Redestig

### scl, pcaRes-method

**Get the scales (e.g. standard deviations) of the original variables**

**Description**

Get the scales (e.g. standard deviations) of the original variables

**Usage**

```
scl(object, ...)
```

**Arguments**

- `object`: pcaRes object
- `...`: Not used
Value

Vector with the scales

Author(s)

Henning Redestig

See Also

prep

scores,pcaRes-method

Get scores from a pcaRes object

Description

Get scores from a pcaRes object

Usage

## S4 method for signature 'pcaRes'
scores(object, ...)

Arguments

object a pcaRes object

... not used

Value

The scores as a matrix

Author(s)

Henning Redestig

See Also

scores.pcaRes
scores.pcaRes  Get scores from a pcaRes object

Description
Get scores from a pcaRes object

Usage
## S3 method for class 'pcaRes'
scores(object, ...)

Arguments
object a pcaRes object
... not used

Value
The scores as a matrix

Author(s)
Henning Redestig

sDev,pcaRes-method  Get the standard deviations of the scores (indicates their relevance)

Description
Get the standard deviations of the scores (indicates their relevance)

Usage
sDev(object, ...)

Arguments
object pcaRes object
... Not used

Value
Standard deviations of the scores

Author(s)
Henning Redestig
showNniRes

Description
Print a nniRes model

Usage
showNniRes(x, ...)

Arguments
x An nniRes object
... Not used

Value
Nothing, used for side-effect
simpleEllipse

Author(s)

Henning Redestig

Description

Get a confidence ellipse for uncorrelated bivariate data

Usage

simpleEllipse(x, y, alfa = 0.95, len = 200)

Arguments

x  first variable
y  second variable
alfa  confidence level of the circle
len  Number of points in the circle

Details

As described in 'Introduction to multi and megavariate data analysis using PCA and PLS' by Eriks-son et al. This produces very similar ellipse as compared to the ellipse function the ellipse package except that this function assumes that and y are uncorrelated (which they of are if they are scores or loadings from a PCA).

Value

A matrix with X and Y coordinates for the circle

Author(s)

Henning Redestig

See Also

eellipse
Description

A common way of visualizing two principal components

Usage

```
slplot(object, pcs=c(1,2), scoresLoadings=c(TRUE, TRUE),
       sl="def", ll="def", hotelling=0.95, rug=TRUE, sub=NULL,...)
```

Arguments

- `object`: a pcaRes object
- `pcs`: which two pcs to plot
- `scoresLoadings`: Which should be shown scores and or loadings
- `sl`: labels to plot in the scores plot
- `ll`: labels to plot in the loadings plot
- `hotelling`: confidence interval for ellipse in the score plot
- `rug`: logical, rug x axis in score plot or not
- `sub`: Subtitle, defaults to annotate with amount of explained variance.
- `...`: Further arguments to plot functions. Prefix arguments to `par()` with 's' for the scores plot and 'l' for the loadings plot. I.e. `cex` become `scex` for setting character expansion in the score plot and `lcex` for the loadings plot.

Details

This method is meant to be used as a quick way to visualize results, if you want a more specific plot you probably want to get the scores, loadings with `scores(object)`, `loadings(object)` and then design your own plotting method.

Value

None, used for side effect.

Note

Uses layout instead of par to provide side-by-side so it works with Sweave (but can not be combined with `par(mfrow=..)`)

Author(s)

Henning Redestig

See Also

`pca`, `biplot`
sortFeatures

Examples

data(iris)
pcIr <- pca(iris[,1:4], scale="uv")
slplot(pcIr, sl=NULL, spch=5)
slplot(pcIr, sl=NULL, lcex=1.3, scol=as.integer(iris[,5]))

sortFeatures  Sort the features of NLPCA object

Description
Sort the features of NLPCA object

Usage

sortFeatures(nlnet, trainIn, trainOut)

Arguments

nlnet  The nlnet
trainIn  Training data in
trainOut  Training data after it passed through the net

Value

...

Author(s)
Henning Redestig

summary  Summary of PCA model

Description
Print a brief description of the PCA model

Usage

## S3 method for class 'pcaRes'
summary(object, ...)

Arguments

object  a pcaRes object
...  Not used
svdImpute

Description
This implements the SVDimpute algorithm as proposed by Troyanskaya et al, 2001. The idea behind the algorithm is to estimate the missing values as a linear combination of the k most significant eigengenes.

Usage
svdImpute(Matrix, nPcs = 2, threshold = 0.01, maxSteps = 100, verbose = interactive(), ...)

Arguments
- **Matrix**
  Pre-processed (centered, scaled) data with variables in columns and observations in rows. The data may contain missing values, denoted as NA.
- **nPcs**
  numeric – Number of components to estimate. The preciseness of the missing value estimation depends on the number of components, which should resemble the internal structure of the data.
- **threshold**
  The iteration stops if the change in the matrix falls below this threshold.
- **maxSteps**
  Maximum number of iteration steps.
- **verbose**
  Print some output if TRUE.
- **...**
  Reserved for parameters used in future version of the algorithm

Details
Missing values are denoted as NA. It is not recommended to use this function directly but rather to use the pca() wrapper function.

As SVD can only be performed on complete matrices, all missing values are initially replaced by 0 (what is in fact the mean on centred data). The algorithm works iteratively until the change in the estimated solution falls below a certain threshold. Each step the eigengenes of the current estimate are calculated and used to determine a new estimate. Eigengenes denote the loadings if pca is performed considering variable (for Microarray data genes) as observations.

An optimal linear combination is found by regressing the incomplete variable against the k most significant eigengenes. If the value at position j is missing, the j'th value of the eigengenes is not used when determining the regression coefficients.

Value
Standard PCA result object used by all PCA-based methods of this package. Contains scores, loadings, data mean and more. See pcaRes for details.
Note

Each iteration, standard PCA (prcomp) needs to be done for each incomplete variable to get the eigengenes. This is usually fast for small data sets, but complexity may rise if the data sets become very large.

Author(s)

Wolfram Stacklies

References


Examples

```r
## Load a sample metabolite dataset with 5% missing values
data(metaboliteData)
## Perform svdImpute using the 3 largest components
result <- pca(metaboliteData, method="svdImpute", nPcs=3, center = TRUE)
## Get the estimated complete observations
cObs <- completeObs(result)
## Now plot the scores
plotPcs(result, type = "scores")
```

svdPca

Perform principal component analysis using singular value decomposition

Description

A wrapper function for prcomp to deliver the result as a pcaRes method. Supplied for compatibility with the rest of the pcaMethods package. It is not recommended to use this function directly but rather to use the pca() wrapper function.

Usage

```r
svdPca(Matrix, nPcs = 2, varLimit = 1, verbose = interactive(), ...)
```

Arguments

- **Matrix**: Pre-processed (centered and possibly scaled) numerical matrix samples in rows and variables as columns. No missing values allowed.
- **nPcs**: Number of components that should be extracted.
- **varLimit**: Optionally the ratio of variance that should be explained. nPcs is ignored if varLimit < 1
- **verbose**: Verbose complaints to matrix structure
- **...**: Only used for passing through arguments.
Value

A pcaRes object.

Author(s)

Henning Redestig

See Also

prcomp, princomp, pca

Examples

data(metaboliteDataComplete)
mat <- prep(t(metaboliteDataComplete))
pc <- svdPca(mat, nPcs=2)
## better use pca()
pc <- pca(t(metaboliteDataComplete), method="svd", nPcs=2)

tempFixNas <- Temporary fix for missing values

Description

Simply replace completely missing rows or cols with zeroes.

Usage

tempFixNas(mat)

Arguments

mat a matrix

Value

The original matrix with completely missing rows/cols filled with zeroes.

Author(s)

Henning Redestig
Description

Tranform the vectors of weights to matrix structure

Usage

```
## S4 method for signature 'matrix'
vector2matrices(object, net)
```

Arguments

- `object`: an `nlpcaNet`
- `net`: the neural network

Value

weights in matrix structure

Author(s)

Henning Redestig

---

Description

Tranform the vectors of weights to matrix structure

Usage

```
## S4 method for signature 'nlpcaNet'
vector2matrices(object)
```

Arguments

- `object`: an `nlpcaNet`

Value

weights in matrix structure

Author(s)

Henning Redestig
wasna.pcaRes-method  

Get a matrix with indicating the elements that were missing in the input data. Convenient for estimating imputation performance.

Description

Get a matrix with indicating the elements that were missing in the input data. Convenient for estimating imputation performance.

Usage

wasna(object, ...)

Arguments

object pcaRes object

... Not used

Value

A matrix with logicals

Author(s)

Henning Redestig

Examples

data(metaboliteData)
data(metaboliteDataComplete)
result <- pca(metaboliteData, nPcs=2)
plot(completeObs(result)[wasna(result)], metaboliteDataComplete[wasna(result)])

weightsAccount  

Create an object that holds the weights for nlpcaNet. Holds and sets weights in using an environment object.

Description

Create an object that holds the weights for nlpcaNet. Holds and sets weights in using an environment object.

Usage

weightsAccount(w)

Arguments

w matrix – New weights
weightsAccount

Value

A weightsAccount with set and current functions.

Author(s)

Henning Redestig
Index

*Topic **algebra**
  robustSvd, 55

*Topic **classes**
  nniRes, 34
  pcaNet, 40
  pcaRes, 42

*Topic **datasets**
  helix, 18
  metaboliteData, 28
  metaboliteDataComplete, 28

*Topic **multivariate**
  asExprSet, 3
  biplot-methods, 4
  bPCA, 5
  checkData, 10
  fitted-methods, 16
  kEstimate, 18
  kEstimateFast, 20
  leverage, pcaRes-method, 22
  llsImpute, 24
  nipalsPca, 29
  nni, 33
  pca, 38
  plotPcs, 44
  ppca, 45
  predict-methods, 47
  Q2, 49
  residuals-methods, 52
  RnipalsPca, 53
  robustPca, 54
  slplot, pcaRes-method, 62
  svdImpute, 64
  svdPca, 65
  asExprSet, 3
  biplot, 62
  biplot, pcaRes-method (biplot-methods), 4
  biplot-methods, 4
  biplot, pcaRes (biplot-methods), 4
  bPCA, 5, 38, 46
  BPCA_dostep, 7
  BPCA_initmodel, 8
  center (center, pcaRes-method), 9
  center, pcaRes-method, 9
  centered (centered, pcaRes-method), 9
  centered, pcaRes-method, 9
  checkData, 10
  completeObs (completeObs, nniRes-method), 11
  completeObs, nniRes-method, 11
  completeObs, pcaRes-method (completeObs, nniRes-method), 11
  cvseg, 11
  cvstat (cvstat, pcaRes-method), 12
  cvstat, pcaRes-method, 12
  deletediagonals, 13
  derrorHierarchic, 13
  dim.pcaRes, 14
  DModX (DModX, pcaRes-method), 14
  DModX, pcaRes-method, 14
  errorHierarchic, 15
  fitted, pcaRes-method (fitted-methods), 16
  fitted-methods, 16
  fitted, pcaRes (fitted-methods), 16
  forkNlpcaNet, 17
  getHierarchicIdx, 17
  helix, 18
  kEstimate, 7, 18, 21
  kEstimateFast, 20, 20
  leverage (leverage, pcaRes-method), 22
  leverage, pcaRes-method, 22
  lineSearch, 23
  llnr, 23
  listPcaMethods, 24
  llsImpute, 24, 33
  loadings (loadings, ANY-method), 26
  loadings, ANY-method, 26
  loadings, pcaRes-method, 26
  loadings.pcaRes, 27, 27