pcaMethods

April 19, 2010

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

```
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
<wolfram.stacklies@gmail.com>

biplot.pcaRes *Plot a overlaid scores and loadings plot*

Description

Visualize two-components simultaneously

Usage

```
biplot.pcaRes(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 λ^{scale} and the observations are scaled by λ^{scale} where `lambda` are the singular values as computed by `princomp`. Normally $0 \le scale \le 1$, 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 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])
biplot(pcIr)
```

bpca *Bayesian PCA Missing Value Estimator*

Description

Implements a Bayesian PCA missing value estimator. The script is a port of the Matlab version provided by Shigeyuki OBA. See also http://hawaii.aist-nara.ac.jp/%7Eshige-o/tools/.
BPCA combines an EM approach for PCA with a Bayesian model. In standard PCA data far from the training set but close to the principal subspace may have the same reconstruction error. BPCA 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.

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, buth 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.

Requires `MASS`.

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

Usage

```
bpca(Matrix, nPcs = 2, completeObs = TRUE, maxSteps = 100,
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 used for re-estimation. Choosing few components may decrease the estimation precision.
`completeObs`	`boolean` Return the complete observations if TRUE. This is the input data with NA values replaced by the estimated values.
`maxSteps`	`numeric` – Maximum number of estimation steps. Default is 100.

verbose	boolean – BPCA prints the number of steps and the increase in precision if set to TRUE. Default is interactive().
...	Reserved for future use. Currently no further parameters are used

Details

Details about the probabilistic model underlying BPCA are found in Oba et. al 2003. The algorithm uses an expectation maximation 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

$$maxSteps \cdot row_{miss} \cdot O(n^3)$$

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

Value

pcaRes	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)

Wolfram Stacklies
Max Planck Institut fuer Molekulare Pflanzenphysiologie, Potsdam, Germany
<wolfram.stacklies@gmail.com>

References

Shigeyuki Oba, Masa-aki Sato, Ichiro Takemasa, Morito Monden, Ken-ichi Matsubara and Shin Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics, 19(16):2088-2096, Nov 2003.*

See Also

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

Examples

```
## Load a sample metabolite dataset with 5% missig values (metaboliteData)
data(metaboliteData)

## Perform Bayesian PCA with 2 components
result <- pca(metaboliteData, method="bpca", nPcs=2, center=FALSE)

## Get the estimated principal axes (loadings)
loadings <- result@loadings
```

```
## Get the estimated scores
scores <- result@scores

## Get the estimated complete observations
cObs <- result@completeObs

## Now make a scores and loadings plot
slplot(result)
```

`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
Max Planck Institut fuer Molekulare Pflanzenphysiologie, Potsdam, Germany
`<wolfram.stacklies@gmail.com>`

pcaNet *Class for representing a neural network for computing Non-linear PCA*

Description

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

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 errror.

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.

error "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.

See Also

`nlpca`

nniRes — *Class for representing a nearest neighbour imputation result*

Description

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

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 befor 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 | 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

pcaRes *Class for representing a PCA result*

Description

This is a class representation of a PCA result

Creating Objects

```
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])
```

Slots

scores "matrix", the calculated scores

loadings "matrix", the calculated loadings

R2cum "numeric", the cumulative R2 values

sDev "numeric", the individual standard deviations

R2 "numeric", the individual R2 values

nObs "numeric", amount of observations

nVar "numeric", amount of variables

centered "logical", data was centered or not

center "numeric", the original variable centers

varLimit "numeric", the exceeded variance limit

nPcs "numeric", the amount of calculated PCs

method "character", the method used to perform PCA

missing "numeric", the total amount of missing values in original data

completeObs "matrix", the estimated complete observations

network "nlpcaNet", the network used by non-linear PCA

Methods

print Print function

summary Extract information about PC relevance

screeplot Plot a barplot of standard deviations for PCs

slplot Make a side by side score and loadings plot

nPcs Get the number of PCs

nObs Get the number of observations

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

completeObs Get the imputed data set

method Get a string naming the used PCA method

sDev Get the standard deviations of the PCs

`fitted.pcaRes` *Extract fitted values from PCA.*

Description

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

Usage

```
fitted.pcaRes(object, data=NULL, nPcs=object@nPcs,...)
```

Arguments

`object`	`pcaRes` the `pcaRes` object of interest.
`data`	`matrix` 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. Non-linear PCA is an exception, here if data is NULL then data is set to the completeObs and propagated through the network.
`nPcs`	`numeric` The amount of PC's to consider
`...`	Not passed on anywhere, included for S3 consistency.

Value

A matrix with the fitted values.

Author(s)

Henning Redestig <redestig[at]mpimp-golm.mpg.de>

Examples

```
data(iris)
pcIr <- pca(iris[,1:4])
head(fitted(pcIr, nPcs=1))
```

helix *A helix structured toy data set*

Description

simulated as data set looking like a helix

Usage

```
helix
```

Format

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

Source

Max Planck Institut fuer Molekulare Pflanzenphysiologie, 2005

References

Matthias Scholz, Fatma Kaplan, Charles L. Guy, Joachim Kopka and Joachim Selbig. - Non-linear PCA: a missing data approach. *Bioinformatics 2005 21(20):3887-3895*

KEstimateFast *Estimate best number of Components for missing value estimation*

Description

This is a simple estimator for the optimal number of componets 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.

Usage

```
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` – One of ppca | bpca | svdImpute | nipals
`evalPcs`	`numeric` – The principal components to use for cross validation or cluster sizes if used with llsImpute. Should be an array containing integer values, eg. 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 maybe 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 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
CAS-MPG Partner Institute for Computational Biology, Shanghai, China
`<wolfram.stacklies@gmail.com>`

See Also

`kEstimate`.

Examples

```
## Load a sample metabolite dataset with 5% missing values (metaboliteData)
data(metaboliteData)

# Estimate best number of PCs with ppca for component 2:4
esti <- kEstimateFast(t(metaboliteData), method = "ppca", evalPcs = 2:4, em="nrmsep")

# Plot the result
barplot(drop(esti$eError), xlab = "Components",ylab = "NRMSEP (1 iterations)")

# The best k value is:
print(esti$minNPcs)
```

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. 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 values.

Usage

```
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` – One of ppca | bpca | svdImpute | nipals | nlpca | llsImpute | llsImputeAll. 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

`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 maybe want to do this to compare several methods on a complete data set.

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

`...` Further arguments to `pca()` or `nni()`

Details

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 $(v_{miss} \cdot segs \cdot nruncv \cdot)$ 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 $\sqrt{(\frac{\sum(x-xe)^2}{(n \cdot var(x))})}$, 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

`list` Returns a list with the elements:

- bestNPcs - number of PCs or k for which the minimal average NRMSEP or the maximal Q2 was obtained.
- 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 wich 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
CAS-MPG Partner Institute for Computational Biology, Shanghai, China
`<wolfram.stacklies@gmail.com>`

See Also

`kEstimateFast`, `Q2`, `pca`, `nni`.

Examples

```
## 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(esti$eError), xlab = "Components",ylab = "NRMSEP (1 iterations)")

# The best result was obtained for this number of PCs:
print(esti$bestNPcs)

# Now have a look at the variable wise estimation error
barplot(drop(esti$variableWiseError[, which(esti$evalPcs == esti$bestNPcs)]),
        xlab = "Incomplete variable Index", ylab = "NRMSEP")
```

`leverage` *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

```
leverage(object,...)
```

Arguments

`object`	a pcaRes object
`...`	not used

Details

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

Value

The observation leverages as a numeric vector

Author(s)

Henning Redestig

References

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

Examples

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

llsImpute *LLSimpute algorithm*

Description

Missing value estimation using local least squares (LLS). First, k variables (for Microarrya 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.

Missing values are denoted as `NA`

It is not recommended to use this function directely but rather to use the nni() wrapper function.

Usage

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

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.

`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 anyway...

`verbose` `boolean` – Print step number and relative change if TRUE and allVariables = TRUE

`...` Reserved for parameters used in future version of the algorithm

Details

The methods provides 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).

Complexity: 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").

Value

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

Author(s)

Wolfram Stacklies
MPG/CAS Partner Institute for Computational Biology, Shanghai, P.R. China
`<wolfram.stacklies@gmail.com>`

References

Kim, H. and Golub, G.H. and Park, H. - Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics, 2005; 21(2):187-198.*

Troyanskaya O. and Cantor M. and Sherlock G. and Brown P. and Hastie T. and Tibshirani R. and Botstein D. and Altman RB. - Missing value estimation methods for DNA microarrays. *Bioinformatics. 2001 Jun;17(6):520-525.*

See Also

`pca`, `nniRes`, `nni`.

Examples

```
## Load a sample metabolite dataset (metaboliteData) with already 5% of
## data missing
data(metaboliteData)

## Perform llsImpute using k = 10
## Set allVariables TRUE because there are very few complete variables
result <- llsImpute(metaboliteData, k = 10, correlation = "pearson", allVariables = TRUE)

## Get the estimated complete observations
cObs <- result@completeObs
```

metaboliteDataComplete

A complete metabolite data set from an Arabidopsis coldstress experiment

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). The data was created during an in house Arabidopsis coldstress experiment.

Usage

```
metaboliteData
```

Format

A matrix containing 154 observations (rows) and 52 metabolites (columns).

Source

Max Planck Institut fuer Molekulare Pflanzenphysiologie, 2005

References

Matthias Scholz, Fatma Kaplan, Charles L. Guy, Joachim Kopka and Joachim Selbig. - Non-linear PCA: a missing data approach. *Bioinformatics 2005 21(20):3887-3895*

See Also

`metaboliteData`

metaboliteData

An incomplete metabolite data set from an Arabidopsis coldstress experiment

Description

A subset of size 154 x 52 from a larger metabolite data set. The data contains 5% of artificially created uniformly distributed misssing values. The data was created during an in house Arabidopsis coldstress experiment.

Usage

```
metaboliteData
```

Format

A matrix containing 154 observations (rows) and 52 metabolites (columns).

Source

Max Planck Institut fuer Molekulare Pflanzenphysiologie, 2005

References

Matthias Scholz, Fatma Kaplan, Charles L. Guy, Joachim Kopka and Joachim Selbig. - Non-linear PCA: a missing data approach. *Bioinformatics 2005 21(20):3887-3895*

`nipalsPca` *Perform principal component analysis using the Non-linear iterative partial least squares (NIPALS) algorithm.*

Description

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 directely but rather to use the pca() wrapper function.

Usage

```
nipalsPca(Matrix, nPcs=2, center=TRUE, completeObs=TRUE, varLimit=1, maxSteps=50
  threshold=1e-6, verbose=interactive(),...)
```

Arguments

`Matrix` Numerical matrix samples in rows and variables as columns.

`nPcs` Number of components that should be extracted.

`center` Mean center the data column wise if set TRUE

`completeObs` Return the estimated complete observations. This is the input Matrix with NA values replaced by the estimated values.

`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 the 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 $(T_{old} - T)^T (T_{old} - T) >$ `limit`.

`verbose` Show simple progress information.

`...` Only used for passing through arguments.

Details

This method is quite slow what may lead to very long computation times when used on larger matrices. The power in missing value imputation is also quite disputable.

Value

A `pcaRes` object.

Author(s)

Henning Redestig

References

Wold, H. (1966) Estimation of principal components and related models by iterative least squares. In Multivariate Analysis (Ed., P.R. Krishnaiah), Academic Press, NY, 391-420.

See Also

prcomp, princomp, pca

Examples

```
data(iris)
pcIr <- nipalsPca(iris[,1:4], nPcs=2)
```

---

nlpca *Non-linear PCA*

Description

Neural network based non-linear PCA

Usage

```
nlpca(Matrix, nPcs=2, center=TRUE, completeObs=TRUE, maxSteps=2*prod(dim(Matrix)
```

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 thenumber of components, which should resemble the internal structure of the data.

center boolean Mean center the data if TRUE

completeObs boolean Return the complete observations if TRUE. This is the original data with NA values filled with the estimated values.

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 2feature units (principal components), one hidden layer fornon-linearity and three output units (original amount ofvariables).

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

`pcaRes` Standard PCA result object used by all PCA-basedmethods of this package. Contains scores, loadings, data meanand more. See `pcaRes` for details.

Author(s)

Based on a matlab script by Matthias Scholz <matthias.scholz[at]uni-greifswald.de> and ported to R by HenningRedestig <redestig[at]mpimp-golm.mpg.de>

References

Matthias Scholz, Fatma Kaplan, Charles L Guy, Joachim Kopkaand Joachim Selbig. Non-linear PCA: a missing dataapproach. *Bioinformatics, 21(20):3887-3895, Oct 2005*

Examples

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

`nni` *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.

`subset` For convenience one can pass a large matrix but only use the variable specified as subset. Can be colnames or indices.

`method` Currently "llsImpute" only.

`...` Further arguments to the chosen method.

Details

This method is wrapper function to llsImpute, See documentation for `link{llsImpute}` Extra arguments usually given to this function include:

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)
```

`pca` *Perform principal component analysis*

Description

Can be used for computing PCA on a numeric matrix for visualisation, information extraction and missing value imputation.

Usage

```
pca(object, method=c("svd", "nipals", "bpca", "ppca",
"svdImpute", "nlpca", "robustPca"), 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.

`subset` For convenience one can pass a large matrix but only use the variable specified as subset. Can be colnames or indices.

`method` One of "svd", "nipals", "bpca", "nlpca" or "ppca".

`...` Further arguments to the chosen pca method.

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`.

bpca: An iterative method using a Bayesian model to handle missing values. See documentation for `bpca`.

ppca: An iterative method using a probabilistic model to handle missing values. See documentation for `ppca`.

svdImpute: Uses expectation maximation to perform SVD PCA on incomplete data. See documentation for `svdImpute`.

Extra arguments usually given to this function include:

nPcs: The amount of principal components to extract

Value

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

Author(s)

Wolfram Stacklies, Henning Redestig

References

Wold, H. (1966) Estimation of principal components and related models by iterative least squares. In Multivariate Analysis (Ed., P.R. Krishnaiah), Academic Press, NY, 391-420.

Shigeyuki Oba, Masa-aki Sato, Ichiro Takemasa, Morito Monden, Ken-ichi Matsubara and Shin Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics, 19(16):2088-2096, Nov 2003.*

Troyanskaya O. and Cantor M. and Sherlock G. and Brown P. and Hastie T. and Tibshirani R. and Botstein D. and Altman RB. - Missing value estimation methods for DNA microarrays. *Bioinformatics. 2001 Jun;17(6):520-5.*

See Also

`prcomp`, `princomp`, `nipalsPca`, `svdPca`

Examples

```
data(iris)
## Usually some kind of scaling is appropriate
pcIr <- pca(iris[,1:4], nPcs = 2, method="nipals")
pcIr <- pca(iris[,1:4], nPcs = 2, method="svd")
## Get a short summary on the calculated model
summary(pcIr)
## Scores and loadings plot
slplot(pcIr, sl=as.character(iris[,5]))
```

`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:object@nPcs, 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.

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)
```

`plotR2` *R2 plot (screeplot) for PCA*

Description

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

Usage

```
plotR2(object, nPcs=object@nPcs, type = c("barplot", "lines"), main = deparse(su
```

Arguments

`object`	`pcaRes` The pcaRes object.
`nPcs`	`numeric` The amount of PC's to consider.
`type`	`character` Barplot or line plot
`main`	`character` The main label of the plot
`...`	Passed on to `screeplot`

Value

None, used for side effect.

Author(s)

Henning Redestig <redestig[at]mpimp-golm.mpg.de

See Also

screeplot

ppca *Probabilistic PCA Missing Value Estimator*

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. Thanks a lot!

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.

Requires `MASS`

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

Usage

```
ppca(Matrix, nPcs = 2, center = TRUE, completeObs = TRUE, seed = NA, ...)
```

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.
`center`	`boolean` Mean center the data if TRUE
`completeObs`	`boolean` Return the complete observations if TRUE. This is the original data with NA values filled with the estimated values.
`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.
`...`	Reserved for future use. Currently no further parameters are used.

Details

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 what leads to much 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. This should, however not be necessary with the lowered threshold.

Value

`pcaRes`	Standart 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)

Wolfram Stacklies
Max Planck Institut fuer Molekulare Pflanzenphysiologie, Potsdam, Germany
`<wolfram.stacklies@gmail.com>`

See Also

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

Examples

```
## Load a sample metabolite dataset with 5% missing values (metaboliteData)
data(metaboliteData)

## Perform probabilistic PCA using the 3 largest components
result <- pca(metaboliteData, method="ppca", nPcs=3, center=TRUE)

## Get the estimated principal axes (loadings)
loadings <- result@loadings

## Get the estimated scores
scores <- result@scores

## Get the estimated complete observations
cObs <- result@completeObs

## Now plot the scores
plotPcs(result, type = "scores")
```

predict.pcaRes *Predict values from PCA.*

Description

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 :

$\hat{T} = X_{new}P$

$\hat{X}_{new} = \hat{T}P'$

Missing values are set to zero before matrix multiplication to achieve NIPALS like treatment of missing values.

Usage

```
predict.pcaRes(object, newdata, pcs=nPcs(object), ...)
```

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
`...`	Not passed on anywhere, included for S3 consistency.

Value

A list with the following components:

`scores`	The predicted scores
`x`	The predicted data

Author(s)

Henning Redestig <henning[at]psc.riken.jp>

Examples

```
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])
cor(irisHat$scores[,1], scores(pcFull)[hidden,1])
```

prep *Preprocess a matrix for PCA*

Description

Implements simple preprocessing alternatives for scaling a matrix.

Usage

```
prep(object, scale=c("none", "pareto", "vector", "UV"), center=TRUE, ...)
```

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.
`center`	Indicates if the matrix should be mean centred or not.
`scale`	One of "UV" (unit variance $a = a/\sigma_a$) "vector" (vector normalisation $b = b/\|\|b\|\|$), "pareto" or "none" to indicate which scaling should be used to scale the matrix with a variables and b samples.
`...`	Only used for passing through arguments.

Details

Does basically the same as `scale` but adds some alternative scaling options.

Value

A matrix with attribute "scaled:center" if centring was done.

Author(s)

Wolfram Stacklies, Henning Redestig

See Also

`scale`

Examples

```
object <- matrix(rnorm(50), nrow=10)
object <- prep(object, scale="vector", center=TRUE)
```

Q2 *Perform internal 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

```
Q2(object, originalData, nPcs=object@nPcs, fold=5, nruncv=10,
segments=NULL, verbose=interactive(), ...)
```

Arguments

`object`	A `pcaRes` object (result from previous PCA analysis.)
`originalData`	The matrix (or ExpressionSet) that used to obtain the pcaRes object. Must not contain any missing values.
`nPcs`	The amount of principal components to estimate Q2 for.
`fold`	The amount of groups to divide the data in.
`nruncv`	The amount of times to repeat the whole cross-validation
`segments`	`list` A predefined list where each element is the set of indices to leave out. Note that if this is provided, Q2 becomes deterministic (if the PCA is deterministic of course).
`verbose`	`boolean` If TRUE Q2 outputs a primitive progress bar.
`...`	Further arguments passed to the pca() function called within Q2

Details

This method calculates Q^2 for a PCA model. This is the predictory version of R^2 and can be interpreted as the ratio of variance in a left out data chunk that can be estimated by the PCA model. Poor (low) Q^2 means 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 amount of PC's x is estimated as $\hat{x} = TP'$ (T are scores and P are loadings). Though this defines the leave-one-out cross-validation this is not what is performed if fold is less than the amount of rows and/or columns.

Diagonal rows of elements in the matrix are deleted and the re-estimated. You can choose your own segmentation as well make sure no complete row or column is lost.

Value

A matrix with Q^2 estimates.

Author(s)

Wolfram Stacklies, Henning Redestig

References

Wold, H. (1966) Estimation of principal components and related models by iterative least squares. In Multivariate Analysis (Ed., P.R. Krishnaiah), Academic Press, NY, 391-420.

See Also

`pca`

Examples

```
data(iris)
pcIr <- pca(iris[,1:4], nPcs=2, method="ppca")
#can only get Q2 estimats for the two first PC's
q2 <- Q2(pcIr, iris[,1:4], nruncv=2)
#Typically Q2 increases only very slowly after the optimal amount of PC's
boxplot(q2~row(q2), xlab="Amount of PC's", ylab=expression(Q^2))
```

`residuals.pcaRes` *Residuals values from a PCA model.*

Description

This function extracts the residuals values from a pcaRes object for the PCA methods SVD, Nipals, PPCA and BPCA

Usage

```
residuals.pcaRes(object, data, nPcs=object@nPcs, ...)
```

Arguments

`object` `pcaRes` the `pcaRes` object of interest.

`data` `matrix` The data that was used to calculate the PCA model (or a different dataset to e.g. adress its proximity to the model).

`nPcs` `numeric` The amount of PC's to consider

`...` Not passed on anywhere, included for S3 consistency.

Value

A `matrix` with the residuals

Author(s)

Henning Redestig <redestig[at]psc.riken.jp>

Examples

```
data(iris)
pcIr <- pca(iris[,1:4])
head(residuals(pcIr, iris[,1:4]))
```

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 caluclated 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 directely but rather to use the pca() wrapper function.

Usage

```
robustPca(Matrix, nPcs = 2, center = TRUE, completeObs = FALSE, verbose = inte
```

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.
`center`	`boolean` Mean center the data if TRUE
`completeObs`	`boolean` Return the complete observations if TRUE. This is the original data with NA values filled with the estimated values. Please note that robustPca was NOT designed for missing value estimation. Use one of the other pca methods, like e.g. BPCA, for missing value estimation!
`verbose`	`boolean` Print some output to the command line if TRUE
`...`	Reserved for future use. Currently no further parameters are used.

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

`pcaRes`	Standart 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)

Wolfram Stacklies
CAS-MPG Partner Institute for Computational Biology, Shanghai, China.
<wolfram.stacklies@gmail.com>

See Also

robustSvd, svd, prcomp, pcaRes.

Examples

```
## 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 effected by the outliers.
plot(resSvd@loadings[,1], resSvdOut@loadings[,1])
plot(resSvd@loadings[,1], resRobPca@loadings[,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).

Usage

```
robustSvd(x)
```

Arguments

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

Details

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.

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 accomodate 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`.

Warning

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

Hawkins, Douglas M, Li Liu, and S Stanley Young (2001) Robust Singular Value Decomposition, National Institute of Statistical Sciences, Technical Report Number 122. http://www.niss.org/technicalreports/tr122.pdf

See Also

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

Examples

```
## 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 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
## We can see that robustSvd is hardly effected by the outliers.
plot(resSvd$v[,1], resSvdOut$v[,1])
plot(resRobSvd$v[,1], resRobSvdOut$v[,1])
```

`slplot` *Plot a side by side scores and loadings plot*

Description

A common way of representing PCA result for two component

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
`rug`	logical, rug x axis or not
`sub`	Subtitle, defaults to annotate with amount of explained variance.
`...`	Further arguments to plot functions

Details

Uses layout instead of par to provide side-by-side so it works with Sweave.

Value

None, used for side effect.

Author(s)

Henning Redestig

See Also

`prcomp`, `pca`, `princomp`

Examples

```
data(iris)
pcIr <- pca(iris[,1:4], scale="UV", method="svd")
slplot(pcIr, sl=NULL, pch=5, col=as.integer(iris[,5]))
```

`svdImpute` *SVDimpute algorithm*

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.

Missing values are denoted as `NA`

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

Usage

```
svdImpute(Matrix, nPcs = 2, center=TRUE, completeObs=TRUE, threshold = 0.01,
  maxSteps = 100, 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.
`center`	Mean center the data if TRUE
`completeObs`	Return the estimated complete observations if TRUE. This is the input data with NA values replaced by the estimated values.
`threshold`	The iteration stops if the change in the matrix falls below this threshold, the default is 0.01. (0.01 was empirically determined by Troyanskaya et. al)
`maxSteps`	Maximum number of iteration steps. Default is 100.
`verbose`	Print some output if TRUE. Default is interactive()
`...`	Reserved for parameters used in future version of the algorithm

Details

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.

Complexity: 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.

Value

`pcaRes`	Standart 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)

Wolfram Stacklies
Max Planck Institut fuer Molekulare Pflanzenphysiologie, Potsdam, Germany
`<wolfram.stacklies@gmail.com>`

References

Troyanskaya O. and Cantor M. and Sherlock G. and Brown P. and Hastie T. and Tibshirani R. and Botstein D. and Altman RB. - Missing value estimation methods for DNA microarrays. *Bioinformatics. 2001 Jun;17(6):520-5.*

See Also

`bpca`, `ppca`, `prcomp`, `nipalsPca`, `pca`, `pcaRes`.

Examples

```
## Load a sample metabolite dataset with 5% missing values (metaboliteData)
data(metaboliteData)

## Perform svdImpute using the 3 largest components
result <- pca(metaboliteData, method="svdImpute", nPcs=3, center = TRUE)

## Get the estimated principal axes (loadings)
loadings <- result@loadings

## Get the estimated scores
scores <- result@scores

## Get the estimated complete observations
cObs <- result@completeObs

## Now plot the scores
plotPcs(result, type = "scores")
```

`svdPca` *Perform principal component analysis using singular value decomposition*

Description

A wrapper function for R's standard function `prcomp`. Delivers the result as a `pcaRes` method for compatibility with the rest of the pcaMethods package.

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

Usage

```
svdPca(Matrix, nPcs=2, center=TRUE, completeObs=FALSE,
  varLimit=1, verbose=interactive(), ...)
```

Arguments

`Matrix`	Numerical matrix samples in rows and variables as columns.
`nPcs`	Number of components that should be extracted.
`center`	Center the data column wise if TRUE
`completeObs`	Return the complete observations. This exisits for compatibility only, as svdPca cannot missing values. If set TRUE the input matrix will be returned in the `completeObs` field.
`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.

Details

svdPca can preferrably be called using `pca(object, method="svd")`.

Value

A `pcaRes` object.

Author(s)

Henning Redestig

See Also

`prcomp`, `princomp`, `pca`

Examples

```
data(iris)
pcIr <- svdPca(iris[,1:4], nPcs=2)
```

Index