Package ‘pbi’

March 13, 2024

Type Package

Title Package for the course Practical Bioinformatics at the University of Potsdam

Version 0.1.0

Date 2024-03-12

Description R package for the course Practical Bioinformatics at the University of Potsdam. It contains various implementations of R functions used in the course.

Imports cluster, rpart, tools, tcltk, digest

URL `https://github.com/mittelmark/pbi`

BugReports `https://github.com/mittelmark/pbi/issues`

Suggests knitr, rmarkdown, palmerpenguins, MASS

VignetteBuilder knitr

License MIT + file LICENSE

NeedsCompilation no

Author Detlef Groth [aut, cre] (<`https://orcid.org/0000-0002-9441-3978`>),
Vera Burdova [ctb],
Jillian Denise Hoffmann [ctb],
Lisa Banu Kurt [ctb]

Maintainer Detlef Groth <`dgroth@uni-potsdam.de`>

R topics documented:

---

pbi-package *The pbi R Package for the Course Practical Bioinformatics*

Description

The pbi package contains functions used in the course Practical Bioinformatics at the University of Potsdam.

Details

Some more details: The following object is available:

pbi-class The pbi environment

All functions can be called in a method like style using the pbi environment or in a function style using the `pbi_` - prefix. For example the function `pbi_cohensD` can be called as well as `pbi$cohensD`.

Author(s)

Detlef Groth <dgroth@uni-potsdam.de>

Examples

```
library(pbi)
ls(pbi)
ls('package:pbi')
```

`pbi` *Environment Object with Functions for the Course Practical Bioinformatics*

Description

The functions within the pbi environment are utility functions used in the course Practical Bioinformatics at the University of Potsdam.

Methods

- `pbi$clusterSilhouette` - determine clusterr quality using silhouette index
- `pbi$clusterSimIndex` - determine clusterr quality using silhouette index
- `pbi$cohensD` - effect size between two means
- `pbi$cohensH` - effect size for a 2x2 contingency table
- `pbi$cohensW` - effect size for a 2x2 o larger contingency tables
- `pbi$cv` - coefficient of variation
- `pbi$dassoc` - assocplot with residual shading
- `pbi$dcorr` - pairwise correlations and p-values for data frames and matrices
- `pbi$dcorrplot` - visualize a pairwise correlation matrix
- `pbi$df2md` - convert a data frame or matrix into Markdown code
- `pbi$dist` - extension of the stats::dist function with more measures for binary data and correlation
- `pbi$domainplot` - plot protein domains similar to those on the prosite website
- `pbi$dpairs` - improved pairs plot considering the data types
- `pbi$dpairs.legend` - adding legends to pairs plots
- `pbi$epsilonSquared` - effect size for comparing three or more means of non-normal data
- `pbi$etaSquared` - effect size for comparing three or more means of normal data
- `pbi$file.cat` - show a text file within the R-console

- `pbi$file.head` - show the first lines of a text file
- `pbi$grect` - add colored background and grid lines to an existing plot
- `pbi$impute` - missing value imputation
- `pbi$lmplot` - xyplot with regression line and correlation coefficient
- `pbi$mi` - mutual information
- `pbi$modelQuality` - model quality for classification and regression models
- `pbi$modus` - most often qualitative level in a variable
- `pbi$msa2pwm` - position weight matrix for an alignment
- `pbi$mw` - molecular weight of an protein sequence
- `pbi$package.deps` - display package dependencies
- `pbi$pastel` - create pastel color scales
- `pbi$pca.biplot` - improved biplot for prcomp objects
- `pbi$pca.corplot` - correlation plot for original variables and PC's
- `pbi$pca.pairs` - improved pairs plot for first PC's
- `pbi$pca.plot` - screeplot for prcomp objects
- `pbi$pca.variances` - returns absolute variances for the PC's
- `pbi$pca.varplot` - plot variances for the PC's
- `pbi$toData` - returns the original data for a prcomp object
- `pbi$pcor` - determine partial correlation
- `pbi$pcor.test` - test significance of partial correlations
- `pbi$prosite2regex` - convert a prosite regular expression into a normal regular expression
- `pbi$protscale` - protscale plot for sliding window properties of proteins
- `pbi$readFasta` - read fasta file into a list of sequences
- `pbi$readPepinfo` - read data from the EMBOSS pepinfo tool
- `pbi$report.chisq.test` - formatted text for a Chisq test
- `pbi$report.conf.int` - formatted text to report a confidence interval
- `pbi$report.pval` - formatting a p-value for reports
- `pbi$sem` - standard error of the mean
- `pbi$searchFasta` - search a FASTA file with a regular expression
- `pbi$text2fasta` - convert sequence text into FASTA formatted files
- `pbi$tkregex` - graphical user interface to test regular expressions
- `pbi$wininstall` - install a R script as a executable Batch file on windows
- `pbi$wordFreq` - determine word frequencies in sequences
- `pbi$xyplot` - improved xy-plot with grid and correlation coefficient

Examples

```
set.seed(124)
ls(pbi)
```

`pbi_clusterSilhouette` *Determine Cluster Silhouette Index*

Description

This function can be used to determine the strength of a given clustering. A silhouette index of larger than 0.7 indicates a strong, values between 0.5 and 0.7 a reasonable, between 0.25 and 0.5 a weak and below 0.25 no reasonable structure.

Usage

```
pbi_clusterSilhouette(x,D)
```

Arguments

`x`	cluster ids
`D`	a distance matrix object

Value

return a list with the components avg.width (the silhouette index value for the complete) and clus.avg.widths the indices for the individual clusters.

Examples

```
D=dist(scale(iris[,1:4]))
hcl=hclust(D)
pbi_clusterSilhouette(cutree(hcl,2),D)
pbi_clusterSilhouette(cutree(hcl,3),D)$avg.width
pbi_clusterSilhouette(cutree(hcl,4),D)$avg.width
pbi_clusterSilhouette(cutree(hcl,5),D)$avg.width
# three cluster seems to be the best
```

`pbi_clusterSimIndex` *Compute Similarity Indices of Two Clusterings*

Description

This function computes the similarity indices between two clusterings, such as the Rand, Jaccard and Cohen's Kappa index.

Usage

```
pbi_clusterSimIndex(v1, v2)
```

Arguments

`v1`	vector with cluster ids for first clustering
`v2`	vector with cluster ids for second clustering

Value

return list with the components ...

Examples

```
set.seed(123)
hcl1=hclust(dist(iris[,1:4]))
hcl2=hclust(dist(scale(iris[,1:4])))
round(unlist(pbi_clusterSimIndex(cutree(hcl1,3),
  cutree(hcl2,3))),2)
round(unlist(pbi_clusterSimIndex(cutree(hcl2,3),
  cutree(hcl1,3))),2) # symmetric
round(unlist(pbi_clusterSimIndex(cutree(hcl1,3),
  sample(cutree(hcl2,3)))),2) # random values I
round(unlist(pbi_clusterSimIndex(sample(cutree(hcl1,3)),
  sample(cutree(hcl2,3)))),2) # random values II
```

`pbi_cohensD` *Effect size for the difference between two means*

Description

The function `pbi_cohensD` calculates the effect size for the difference between two means. Due to Cohen's rule of thumb values of 0.2 to 0.5 are considered to stand for small effects, values from 0.5 to 0.8 represent medium effects and values above 0.8 represent large effects.

Usage

```
pbi_cohensD(x,y,paired=FALSE)
```

Arguments

`x`	vector with numerical values
`y`	vector with two grouping variables, having the same length as num
`paired`	are the data paired, default: FALSE

Value

return Cohen's d value

See Also

`pbi-package`, `pbi-class`, `pbi_cohensW` `pbi_etaSquared`, `pbi_epsilonSquared`

Examples

```
set.seed(125)
data(sleep)
with(sleep,pbi_cohensD(extra,group))
x1=rnorm(100,mean=20,sd=1)
x2=rnorm(100,mean=22,sd=1)
g1=rep('A',100)
```

```
   g2=rep('B',100)
   # difference should be around 2SD
   pbi_cohensD(c(x1,x2),as.factor(c(g1,g2)))
   # biserial correlation coefficient as alternative
   # value is as well large
   cor(c(x1,x2),as.numeric(as.factor(c(g1,g2))))
```

`pbi_cohensH` *Effect size for a 2x2 contingency table*

Description

The function `pbi_cohensH` calculates the effect size for 2x2 contingency tables. Due to Cohen's rule of thumb values of 0.2 to 0.5 are considered to stand for small effects, values from 0.5 to 0.8 represent medium effects and values above 0.8 represent large effects.

Usage

```
pbi_cohensH(tab)
```

Arguments

`tab`	a 2x2 contingency table

Value

return Cohen's h value

See Also

`pbi-package`, `pbi-class`, `pbi_cohensW`

Examples

```
# data from New Eng. J. Med. 329:297-303, 1993
azt=as.table(matrix(c(76,399,129,332), byrow=TRUE,ncol=2))
rownames(azt)=c("AZT","Placebo")
colnames(azt)=c("DiseaseProgress", "NoDiseaseProgress")
pbi_cohensH(azt)
```

`pbi_cohensW` *Effect size for contingency tables*

Description

The function `pbi_cohensW` calculates the effect size for contingency tables. Due to Cohen's rule of thumb values of 0.1 to 0.3 are considered to stand for small effects, values from 0.3 to 0.5 represent medium effects and values above 0.5 represent large effects.

Usage

```
pbi_cohensW(x,p=NULL)
```

Arguments

x	contingency table with counts, usually created using the table command for two variables or vector with counts with observations for a single variable, if x is a vector p must be given
p	expected proportions if x is a vector, in case *x* has length of two, a single value can be given, otherwise p must have same length as x, default: NULL

See Also

pbi-package, pbi-class, pbi_cohensH

Examples

```
data(Titanic)
Titanic[1,1,,]
pbi_cohensW(Titanic[1,1,,])
# Data from New Eng. J. Med. 329:297-303, 1993
azt=as.table(matrix(c(76,399,129,332), byrow=TRUE,ncol=2))
rownames(azt)=c("AZT","Placebo")
colnames(azt)=c("DiseaseProgress", "NoDiseaseProgress")
pbi_cohensW(azt)
# number of boys (25) and girls (15) in a hospital which deviates
# from 50/50 or 51/49 ratios
pbi_cohensW(c(25,15),p=0.5)
pbi_cohensW(c(25,15),p=c(0.51,0.49))
# most extrem case 40 boys and 0 girls
pbi_cohensW(c(40,0),p=0.5)
pbi_cohensW(c(40,0),p=c(0.51,0.49)) # max value here around 2*0.49
```

---

pbi_cv *Coefficient of variation*

Description

The function pbi_cv calculates the coefficient of variation, which is an unit-less measure of data scatter. It is calculated as: *(100*sd(x))/mean(x)*. All data values has to be non-negative.

Usage

```
pbi_cv(x,na.rm=FALSE)
```

Arguments

x	vector with positive numerical values.
na.rm	should NA's be removed, default: FALSE

Value

return numerical value for the coefficient of variation.

See Also

pbi-package, pbi-class, pbi_sem

Examples

```
pbi_cv(rnorm(20,mean=100,sd=4))
pbi_cv(c(1,2,3,4))
pbi_cv(c(1,2,3,4,NA))
pbi_cv(c(1,2,3,4,NA),na.rm=TRUE)
```

`pbi_dassoc` *Assocplots with residual coloring*

Description

This function updates the standard assocplot function from the graphics package with the ability to display residual colors. In blue and red are shown groups with residuals above +4 or below -4 in light colors are shown residuals between 2 and 4 for positive and -4 and -2 for negative residuals.

Usage

```
pbi_dassoc(...,shade=TRUE)
```

Arguments

`...`	arguments delegated to the standard assocplot function.
`shade`	should the residuals been shown, default: TRUE

See Also

`pbi-package`, `pbi-class`

Examples

```
x = margin.table(HairEyeColor, c(1, 2))
pbi_dassoc(x)
```

`pbi_dcorr` *Pairwise correlations and their p-values*

Description

The function is an extension to the standard `stats::cor` function, it calculates as well the p-values for the pairwise associations and returns them in a matrix as well.

Usage

```
pbi_dcorr(data,method='pearson',use='pairwise.complete.ob')
```

Arguments

`data`	matrix or data frame where the variables are in the columns, NA’s are allowed.
`method`	type of correlation to be determined, either ’pearson’, ’spearman’ or ’kendall’, default: ’pearson’
`use`	how to deal with NA’s, default: ’pairwise.complete.obs’

Value

returns list with the following components:

- *estimate* - matrix with correlation values
- *p.value* - matrix with p-values
- *method* - character string with the used correlation method

See Also

pbi-package, pbi-class, pbi_dcorrplot, pbi_mi

Examples

```
data(swiss)
lapply(pbi_dcorr(swiss)[1:2],round,2)
```

---

pbi_dcorrplot *Visualize a correlation matrix*

Description

Visualize a correlation matrix.

Usage

```
pbi_dcorrplot(mt, text.lower=TRUE, text.upper=FALSE, pch=19,
  p.mat=NULL, alpha=0.05, cex.sym=5, cex.r=1, cex.lab=1.4,...)
```

Arguments

mt	matrix with pairwise correlations
text.lower	should in the lower diagonal the correlation coefficient be shown, default: TRUE
text.upper	should in the upper diagonal the correlation coefficient be shown, default: FALSE
pch	the plotting symbol for the correlations, default: 19
p.mat	matrix with p-values to strike out insignificant p-values, default: NULL (not used)
alpha	significance threshold for *p.mat*, default: 0.05
cex.sym	character expansion for the correlation symbols, default: 5
cex.r	character expansion for the r-values if *text.lower* or *text.upper* are set to TRUE, default: 1
cex.lab	character expansion for the variable text labels, default: 1.4
...	arguments delegated to the plot function

See Also

pbi-package, pbi-class, pbi_dcorr

Examples

```
data(swiss)
sw=swiss
colnames(sw)=abbreviate(colnames(swiss),6)
cr=pbi_dcorr(sw,method='spearman')
pbi_dcorrplot(cr$estimate,cex.sym=8,text.lower=TRUE,
   cex.r=1.5,p.mat=cr$p.value)
```

`pbi_df2md` *Convert a data frame or a matrix into a Markdown table*

Description

This function can be used within Rmarkdown documents to display easily a simple Markdown table. For more advance use cases you should other commands such as kable from the knitr package.

Usage

```
pbi_df2md(x,caption='',rownames=TRUE)
```

Arguments

`x`	matrix or data frame
`rownames`	should the rownames be displayed, default: TRUE
`caption`	the caption for the table, it is just displayed below of the table.

Value

return prints to stdout.

See Also

`pbi-package`, `pbi-class`

Examples

```
data(swiss)
pbi_df2md(head(swiss))
```

`pbi_dist` *Distance function which as well supports binary and correlation distances*

Description

This function is an extension to stats::dist function as it supports as well correlation distance and binary distance measures such as Jaccard coefficient and Matching coefficient. The correlation distance is implemented as

$$D_{i,j} = 1 - \frac{r_{i,j} + 1}{2}$$

so negative correlations have low similarities.

Usage

```
pbi_dist(x,method="euclidean",...)
```

Arguments

`x`	data frame or matrix with numerical data, in case of binary data as well boolean and two level nominal data could be supplied.
`method`	the distance measure to be used, one of the measures for *stats::dist* such as "euclidean" or "correlation", alias for "pearson" or "spearman","kendall", for binary data "jc" (Jaccard) and "mc" (Matching coeffient) are supported.
`...`	remaining arguments are forwarded to stats::dist function in case the method is handled by this default method.

Value

return distance matrix

See Also

pbi-package, pbi-class

Examples

```
round(cor(iris[,1:4]),2)
round(as.matrix(pbi_dist(t(iris[,1:4]),method="pearson")),2)
biris=iris[,1:4]
biris=apply(biris,2,function(x) { return(x>median(x)) })
head(biris,3)
head(apply(biris,2,as.numeric),3)
biris=apply(biris,2,as.numeric)
summary(biris)
round(as.matrix(pbi_dist(t(biris),method="mc")),2)
d.can=as.matrix(pbi_dist(t(scale(iris[,1:4])),method="can"))
d.can
d.can=d.can/max(d.can)
round(d.can,2)
```

pbi_domainplot *Protein domain plot*

Description

This function can be used to draw a plot representing protein domains into a standard R plot. The data are send to the website `https://prosite.expasy.org/cgi-bin/prosite/mydomains/`, a URL is created and the image from this url is downloaded to the local file system. The filename is a CRC32 digest of the URL so allowing to cache as well the downloaded results.

Usage

```
pbi_domainplot(domains, ranges, sites, length=1000, hscale=1, cache=TRUE,
    plot=TRUE, cex=0.8, ...)
```

Arguments

`domains`	list where names are the domain names and the values are four integers: start, end, shape (1:6), color (1:4)
`ranges`	list with vectors of three integers: start, end, type (0:1)
`sites`	list with vectors of two integers: position, type (0:1)
`length`	length of the protein, default: 1000
`hscale`	horizontal scaling factor (between 0.1 and 2.0), default: 1.0
`cache`	should the image locally cached, default: TRUE
`plot`	should the image plotted on a R plot devices, default: TRUE
`cex`	character expansion for the domain text and sites if plot is TRUE, default: 0.8
`...`	any argument forwarded to the plot function if plot is TRUE

Value

return a URL or a local filename in case of cache=TRUE, if plot is TRUE NULL is returned invisible

See Also

`pbi-package`, `pbi-class`

Examples

```
url=pbi_domainplot(
    domains=list(MYDOM1=c(100,200,2,1),MYDOM2=c(300,500,3,2)),
    ranges=list(a=c(190,500,1)),
    sites=list(a=c(150,1)),hscale=2.0,plot=FALSE)
print(url)
```

pbi_dpairs *Improved pairs plot considering the data types*

Description

The function `pbi_dpairs` provides an improved pairs plot which accounts for the data type of the actual variables. It will plot in the lower diagonal xy-plots, box-plots or assoc-plots depending on the two data types. In the upper diagonal effect sizes and stars for the p-values for the tests (anova, t.test, chisq.test or cor.test) will be shown. In the diagonal the data distribution will be outlined. This plot is usually an useful visualization for 3-8 variables.

Usage

```
pbi_dpairs(data, col.box='grey80', col.xy="grey60", cex.diag=2.5, order=TRUE,
    pch=19)
```

Arguments

`data`	data frame with columns of class factor, numeric or integer.
`col.box`	colors for the boxplots, either a single value or a vector of colors for each level of a factor variable, default; 'grey80'
`col.xy`	colors for the xy-plots, either a single value of a vector which is as long as the number of data points, default: 'grey60'
`cex.diag`	character expansion for the diagonal texts
`order`	should the variables be ordered by data type and name, this is recommended as it orders the plots, starting with assocplots, then boxplots and finally xyplots, default: TRUE
`pch`	plotting character for xy-plots, default 19 (round circle).

See Also

`pbi-package`, `pbi-class`, `pbi_dpairs.legend`

Examples

```
data(iris)
par(omi = c(0.8, 0.4,0.4,0.4))
pbi_dpairs(iris,col.box=2:4,col.xy=rep(c(2:4),each=50),
   cex.diag=1.6)
pbi_dpairs.legend(levels(iris$Species),col=2:4)

par(omi=c(0.5,0.5,0.8,0.2))
library(MASS)
btwt=birthwt;
for (col in c('low','race','smoke','ptl','ht','ui','ftv')) {
   btwt[,col]=as.factor(btwt[,col])
}
pbi_dpairs(btwt[,2:8],cex.diag=1.6)
    mtext('Birth-Weight data',side=3,outer=TRUE,
    cex=1.5,line=1)
if (require("palmerpenguins")) {
    data(penguins)
    peng=penguins
    colnames(peng)[3]='bill.len'
    colnames(peng)[4]='bill.dep'
    colnames(peng)[5]='flip.len'
    colnames(peng)[6]='mass'
    pbi_dpairs(peng,col.xy=pbi_pastel(3)[as.numeric(peng$species)])
    pbi_dpairs.legend(levels(peng$species),
        col=pbi_pastel(3)[as.numeric(as.factor(levels(peng$species)))])
}
```

`pbi_dpairs.legend` *Adding legend top or bottom to a pbi_dpairs or pairs plot*

Description

The function `pbi_dpairs.legend` allows the user to place a legend outside of a pairs or dpairs plot.

Usage

```
pbi_dpairs.legend(labels, col='grey80', pch=15, cex=1)
```

Arguments

`labels`	txt labels to be plotted
`col`	colors for the plotting characters
`pch`	plotting symbol, default 15
`cex`	the character expansion for text and plotting characters, default: 1

See Also

`pbi-package`, `pbi-class`, `pbi_dpairs`

Examples

```
data(iris)
par(omi = c(0.8, 0.4,0.8,0.4)) # reserve some space top and bottom
pbi_dpairs(iris,col.box=2:4,col.xy=rep(c(2:4),each=50))
pbi_dpairs.legend(levels(iris$Species),col=2:4)
mtext('Iris Data',side=3,outer=TRUE,cex=2,line=1)
```

pbi_epsilonSquared *Effect size epsilon-squared for variables of a kruskal.test*

Description

Cohen's rule of thumb for interpretation is: 0.01-0.09 small, 0.09-0.25 medium and above 0.25 large effect. You can convert epsilon-squared to a Spearman *rho* by using the square root of epsilon-square.

Usage

```
pbi_epsilonSquared(x,y)
```

Arguments

x	vector with numerical values or a linear model or an aov object.
y	vector with factor values or numerical vector of a second class

Value

return numerical value for epsilon-square for given variables.

See Also

pbi-package, pbi-class, pbi_cohensD, pbi_etaSquared

Examples

```
data(iris)
epsilonSquared=pbi_epsilonSquared
epsilonSquared(iris$Sepal.Length,iris$Species)
# two factor example as well for wilcox.test possible
data(ToothGrowth)
epsilonSquared(ToothGrowth$len,as.factor(ToothGrowth$dose))
# close to r-square of spearman!
cor(ToothGrowth$len,ToothGrowth$dose,method="spearman")^2
```

pbi_etaSquared *Effect size eta-squared for an Anova or a linear model object*

Description

Cohen's rule of thumb for interpretation is: 0.01-0.09 small, 0.09-0.25 medium and above 0.25 large effect. You can convert eta-squared to a Pearson *r* by using the square root of eta-squared.

Usage

```
pbi_etaSquared(x,y=NULL)
```

Arguments

`x`	vector with numerical values or a linear model or an aov object.
`y`	either f factor or NULL if x is given as model.

Value

return numerical value for eta-squares for all given variables in the model x or the value for the variable given in y if x is a numerical variable.

See Also

`pbi-package`, `pbi-class`, `pbi_cohensD`, `pbi_epsilonSquared`

Examples

```
data(iris)
etaSquared=pbi_etaSquared
etaSquared(iris$Sepal.Length,iris$Species)
etaSquared(lm(iris$Sepal.Length ~ iris$Species))
etaSquared(aov(iris$Sepal.Length ~ iris$Species))
etaSquared(aov(Sepal.Length ~ Species+Sepal.Width,data=iris))
```

`pbi_file.cat` *Displays a file to the terminal*

Description

Displays the a file to the terminal. Works in a platform independent way.

Usage

```
pbi_file.cat(filename)
```

Arguments

`filename`	filename of a text file

See Also

`pbi-package`, `pbi-class`, `pbi_file.head`

Examples

```
head(pbi_file.cat(system.file(
"files/pepinfo-spike-sars2.txt",package="pbi")),n=10)
```

`pbi_file.head` *Displays the first n lines of a file to the terminal*

Description

Displays the first *n* lines of a file to the terminal.

Usage

```
pbi_file.head(filename,n=6)
```

Arguments

`filename`	filename of a text file
`n`	number of first lines to display, default: 6

See Also

`pbi-package`, `pbi-class`, `pbi_file.cat`

Examples

```
pbi_file.head(system.file(
"files/pepinfo-spike-sars2.txt",package="pbi"))
```

`pbi_grect` *Colored background and grid lines for a plot*

Description

The function can be used to add a background color to existing plots. In case the plotting was already done you should add a very transparent color, using for instance 33 as the last two digits for a RGB color.

Usage

```
pbi_grect(col='#c0c0c033',grid=FALSE)
```

Arguments

`col`	- background color for the plot, default: '#c0c0c033'
`grid`	- should a grid been drawn, default: FALSE

See Also

`pbi-package`, `pbi-class`

Examples

```
par(mfrow=c(1,2),mai=rep(0.8,4))
  data(iris)
  pbi_xyplot(iris[,1:2],col=as.numeric(iris$Species)+1)
  pbi_grect()
  plot(iris[,1:2],type="n")
  pbi_grect(col="#ffe0e0",grid=TRUE)
  points(iris[,1:2],col=as.numeric(iris$Species)+1,pch=19)
```

`pbi_impute` *Missing value imputation using mean, rpart or knn methods*

Description

Replaces missing values with a reasonable guess by different imputation methods such as the simple and not recommended methods mean and median, where NA's are replaced with the mean or median for this variable or the more recommended methods using rpart decision trees or knn using a correlation distance based k-nearest neighbor approach. The rpart method can be as well used to replace missing values for categorical variables. In case of median and mean imputations for categorical variables the modus is used, so missing values are replaced with the most often category. This is rarely reasonable.

Usage

```
pbi_impute(x,method="mean", k=5, cor.method="spearman")
```

Arguments

`x`	either a matrix or data frame
`method`	the method used for replacing missing values, either mean, median, rpart or knn, default mean (not recommended).
`k`	in case of knn imputation the number of neighbors, default: 5
`cor.method`	in case of distance determination the correlation method, default: 'spearman'

Value

return depending on the input either a data frame or matrix with NA's replaced by imputed values.

See Also

`pbi-package`, `pbi-class`

Examples

```
  data(iris)
  ir=as.matrix(iris[,1:4])
  idx=sample(1:length(ir),50)
  ir[idx]=NA
  summary(ir)
  irc=as.matrix(ir)
  ir=pbi_impute(ir)
  summary(ir)
```

```
ir=iris
ir[1,3]=NA; ir[2,4]=NA; ir[c(1,3),5]=NA
head(ir,3)
head(pbi_impute(ir,method="rpart"),3)
irmea=pbi_impute(irc,method="mean")
irknn=pbi_impute(irc,method="knn")
irrpt=pbi_impute(irc,method="rpart")
cor(as.matrix(iris[,1:4])[is.na(irc)],irmea[is.na(irc)])
cor(as.matrix(iris[,1:4])[is.na(irc)],irknn[is.na(irc)])
cor(as.matrix(iris[,1:4])[is.na(irc)],irrpt[is.na(irc)])
```

pbi_lmplot *XY-plot with linear model fits and confidence lines*

Description

The plot visualizes the linear fit description between numerical variables, together with the confidence limits for the predictions as well as for the linear model. The code is based on a tutorial by Conrad Halling (2006), see http://sphaerula.com/legacy/R/linearRegression.html

Usage

```
pbi_lmplot(x, y, col="blue", pch=19, col.lm="red", grid=TRUE, ...)
```

Arguments

`x`	vector with numerical variables
`y`	vector with numerical variables
`col`	color for points and confidence lines for predictions, default: blue
`pch`	plotting character, default: 19
`col.lm`	color for the linear model and the confidence lines, default: red
`grid`	should a plot grid be drawn, default: TRUE
`...`	other arguments delegated to the standard plot function

See Also

`pbi-package`, `pbi-class`, `pbi_modelQuality`

Examples

```
c20.22=c(
17.9, 18.3, 18.3, 18.4, 18.4, 20.2, 20.3, 21.8, 21.9,
22.1, 23.1, 24.2, 24.4)
ins.sens=c(
250, 220, 145, 115, 230, 200, 330, 400, 370, 260, 270,
 530, 375)

pbi_lmplot(x=c20.22,y=ins.sens,
   xlab='%C20-22 Fatty Acids',ylim=c(0,600),
   xlim=c(17,25),main='best fit',
   ylab='Insuline Sensitivity Index (mg/m^2/min)')
```

```
legend('bottomright',c('best fit','fit 95% CI',
       'prediction 95% CI'),lty=c(1,2,2),
       col=c('red','red','blue'))
```

---

pbi_mi *Mutual information for two vectors or a binned table*

Description

Returns: mutual information value, or matrix of all pairwise values in case input is matrix or data frame

Usage

```
pbi_mi(x,y=NULL, breaks=4)
```

Arguments

x	either a binned table, a numerical vector, a data frame or matrix
y	a numerical vector if x is not a binned table, matrix or data frame
breaks	number of breaks to create a binned table if x and y are numerical vectors, default: 4

Value

return mutual information value or matrix of all pairwise values in case input is matrix or data frame.

See Also

pbi-package, pbi-class, pbi_dcorr

Examples

```
rn1=rnorm(100,mean=10,sd=1);
rn2=rn1+0.5*rnorm(100)
cor(rn1,rn2) # high
cor(rn1,sample(rn2)) #random
pbi_mi(rn1,rn2) # high
pbi_mi(rn1,sample(rn2)) #random
tab=table(cut(rn1,breaks=4),cut(rn2,breaks=4))
pbi_mi(tab)
pbi_mi(rn1,rn2,breaks=4)
pbi_mi(rn1,rn2,breaks=7)
data(iris)
round(pbi_mi(iris[,1:4]),2)
mii=pbi_mi(iris[,1:4])
round(mii/diag(mii),2)
round(cor(iris[,1:4]),2)
```

pbi_modelQuality *Quality measures for regression and classification models*

Description

The functions returns a few basic model quality measures such as RMSE, MSE and MAE in case the data in x and y are numeric, or Accuracy, Sensitivity, Specificity and Balanced Classification rate (BCR) in case the data are factors.

Usage

```
pbi_modelQuality(x,y)
```

Arguments

x	numercial values or factor values representing the true values.
y	numerical values or factor values representing the predicted values.

Value

return list with the components MAE, MSE, RMSE for numerical values or ACC, SENS, SPEC and BCR for factors (classes).

See Also

pbi-package, pbi-class, pbi_lmplot

Examples

```
library(MASS)
data(swiss)
# regression measures
mod=lm(bwt~lwt,data=birthwt)
summary(mod)$r.squared
pred=predict(mod,newdata=birthwt)
unlist(pbi_modelQuality(birthwt$bwt,pred))
mod=lm(bwt~lwt+age,data=birthwt)
summary(mod)$r.squared
unlist(pbi_modelQuality(birthwt$bwt,predict(mod,newdata=birthwt)))

# classification measures
library(rpart)
# simple two class problem
sex=as.factor(c(rep("M",50),rep("F",50)))
height=c(rnorm(50,mean=180,sd=7),rnorm(50,mean=178,sd=6))
mod=rpart(sex~height)
table(sex,pred=predict(mod,newdata=data.frame(height=height),type="class"))
unlist(pbi_modelQuality(sex,
       predict(mod,newdata=data.frame(height=height),type="class")))
# simple two class problem but unbalanced
sex=as.factor(c(rep("M",30),rep("F",70))) # rare males
height=c(rnorm(30,mean=180,sd=7),rnorm(70,mean=178,sd=6))
mod=rpart(sex~height)
```

```
table(sex,pred=predict(mod,newdata=data.frame(height=height),type="class"))
unlist(pbi_modelQuality(sex,
       predict(mod,newdata=data.frame(height=height),type="class")))
# difficult three class problem
table(iris$Species)
mod=rpart(Species~.,data=iris)
pred=predict(mod,newdata=iris,type="class")
table(pred,iris$Species)
# convert to two classes (setosa)
table(c('s','v','v')[as.numeric(iris$Species)])
unlist(pbi_modelQuality(
   as.factor(c('s','v','v')[as.numeric(iris$Species)]),
   as.factor(c("s","v","v")[as.numeric(pred)])))
# convert to two classes (versicolor)
unlist(pbi_modelQuality(
   as.factor(c('x','v','x')[as.numeric(iris$Species)]),
   as.factor(c("x","v","x")[as.numeric(pred)])))
# convert to two classes (virginica)
unlist(pbi_modelQuality(
   as.factor(c('x','x','v')[as.numeric(iris$Species)]),
   as.factor(c("x","x","v")[as.numeric(pred)])))
```

`pbi_modus` *Most often level in a categorical variable*

Description

Simple implementation of returning the most most often appearinglevel(s) of a categorical variable.

Usage

```
pbi_modus(x)
```

Arguments

`x`	a vector with elements of class factor

Value

return most often apparent level in the categorical variable

See Also

`pbi-package`, `pbi-class`

Examples

```
pbi_modus(c('A','A','B','C'))
pbi_modus(c('A','A','B','B','C'))
```

pbi_msa2pwm *PFM, PPM and PWM for a alignment file*

Description

Implementation is partially derived from Dave Tang, see: https://davetang.org/muse/2013/10/01/position-weight-matrix/

Usage

```
pbi_msa2pwm(filename)
```

Arguments

filename	filename of a file wither with two columns (second column with aligned sequences, or a single column file with the first column having the sequences.

Value

return list with the following components:

- PFM - Position frequency matrix
- PPM - Position probability matrix
- PWM - Position weight matrix
- PWMPC - Position weight matrix with added pseudocounts (untested)

See Also

pbi-package, pbi-class, pbi_mw, pbi_readFasta

Examples

```
# data example: https://en.wikipedia.org/wiki/Position_weight_matrix
  cat("GAGGTAAAC
TCCGTAAGT
CAGGTTGGA
ACAGTCAGT
TAGGTCATT
TAGGTACTG
ATGGTAACT
CAGGTATAC
TGTGTGAGT
AAGGTAAGT
",file="test.clu")
lapply(pbi_msa2pwm("test.clu"),round,2)
```

`pbi_mw` *Molecular weight for a given sequence*

Description

Calculates and returns the molecular weight for a given amino acid sequence.

Usage

```
pbi_mw(seq)
```

Arguments

`seq`	an amino acid sequence string

Value

return the protein molecular weight.

See Also

`pbi-package`, `pbi-class`, `pbi_readFasta`

Examples

```
pbi_mw("AACTLIS")
pbi_mw("A")
```

`pbi_package.deps` *Package(s) which are required by the given package name*

Description

The function helps in identifying the nested package dependencies for a given package.

Usage

```
pbi_package.deps(pkgName,mode='all')
```

Arguments

`pkgName`	an package name given as text string.
`mode`	which package names to return, the following modes are available: 'all' - all required packages, 'install' - not yet installed packages, 'nonbase' - packages not in the standard R installation

Value

return list of required packages.

See Also

pbi-package, pbi-class

Examples

```
   ## Not run:
     pbi_package.deps('igraph',mode='nonbase')
     pbi_package.deps('igraph',mode='all')

## End(Not run)
```

---

pbi_pastel *Up to 20 pastel colors*

Description

This is an alternative color creation function for R versions before 3.6 where the function ‘hcl.colors‘ is not available.

Usage

```
pbi_pastel(n)
```

Arguments

n	number of colors requested, must be within 2 and 20

Value

return vector of colors in RGB codes of requested length ’n’.

See Also

pbi-package, pbi-class

Examples

```
pbi_pastel(4)
par(mai=c(0.2,0.2,0.2,0.1))
plot(1:20,col=pbi_pastel(20),cex=3,pch=15)
```

pbi_pca.biplot *Improved biplot for prcomp objects*

Description

The function pca.biplot provides an improved biblot for visualizing the pairwise scores of individual principal components of an object created using the function _prcomp_. In contrast to the default biplot function this plot visualizes the data as points and not row numbers, it allows to display groups using color codes and distribution ellipses.

Usage

```
pbi_pca.biplot(pca,pcs=c("PC1","PC2"), pch=19, col='black', text=NULL,
    arrows=TRUE, arrow.fac=1, arrow.n=-1, ellipse=FALSE, ell.fill=FALSE,
    xlab=NULL, ylab=NULL, grid=TRUE, scale=NULL, ...)
```

Arguments

pca	pca object of class _prcomp_, created using the function _prcomp_.
pcs	the components to plot, default: c('PC1','PC2')
pch	plotting character, default: 19
col	plotting color, default: black
text	optional text labels, if not given pch is used, default: NULL
arrows	should loading arrows be displayed, default: TRUE
arrow.fac	scaling factor for arrow length, default: 1
arrow.n	how many arows per PC to display, default: -1 (all)
ellipse	should 85 and 95 confidence intervals for the chisq distribution be shown. If this is shown colors for each group using the col argument must be given, default: FALSE
ell.fill	should a filled 85 percent confidence interval be shown, colors will be used from the plotting color with opacity, default: FALSE
xlab	custom xlab, if not given the PC name with variance in percent is shown, default: NULL
ylab	custom ylab, if not given the PC name with variance in percent is shown, default: NULL
grid	should a plot grid be drawn, default: TRUE
scale	function to scale to coordinate values sich as asinh, default: NULL
...	additional arguments delegated to the standard plot function

See Also

pbi-package, pbi-class, pbi_pca.corplot, pbi_pca.pairs, pbi_pca.plot, pbi_pca.toData, pbi_pca.variances, pbi_pca.varplot

Examples

```
par(mfrow=c(1,2),mai=c(0.8,0.8,0.6,0.2))
data(iris)
pci=prcomp(iris[,1:4],scale=TRUE)
pbi_pca.biplot(pci,col=rep(2:4,each=50),ellipse=TRUE,ell.fill=TRUE,
    arrow.fac=2.3,arrows=TRUE)
legend('topright',pch=19,col=2:4,levels(iris$Species))
# just a score plot
pbi_pca.biplot(pci,col=rep(2:4,each=50),ellipse=TRUE,ell.fill=TRUE,
    arrow.fac=2.3,arrows=FALSE)
pbi_pca.biplot(pci,col=rep(2:4,each=50),ellipse=FALSE,arrow.fac=2.3,arrows=FALSE)
pbi_pca.biplot(pci,pcs=c('PC1','PC3'),col=rep(2:4,each=50),
       ellipse=FALSE,arrow.fac=2.3,arrows=FALSE)
data(swiss)
col=c(2,4)[as.numeric(cut(swiss$Catholic,breaks=c(0,20,100)))]
pcs=prcomp(swiss,scale=TRUE)
pbi_pca.biplot(pcs,arrow.fac=2,grid=FALSE,
      col=col,ellipse=TRUE,text=substr(rownames(swiss),1,3))
pbi_pca.biplot(pcs,arrow.fac=2,grid=TRUE,
     col=col,text=substr(rownames(swiss),1,3),scale=asinh,arrows=FALSE)
```

---

pbi_pca.corplot *PCA correlation plot*

Description

The function provides a PCA correlation plot to show associations between PCs and variables. The closer a variable to the PC coordinate the higher the correlation, the more away from the center of the coordinate system, the higher the impact of the variable on this PC. You can think about the corplot as a biplot ommiting the samples and the arrows.

Usage

```
pbi_pca.corplot(pca,pcs=c("PC1","PC2"), main="Correlation plot",cex=NULL,nvar=64,...)
```

Arguments

pca	pca object which was created using the function prcomp.
pcs	vector of two PCs to be plotted against each other, default: c('PC1','PC2')
main	title of the plot, default: 'Correlation plot'
cex	character expansion for the samples, default: NULL (automatic calculation)
nvar	number of variables which will be displayed, for both components the variables which contributes mostly to the variances will be used, default: 64.
...	remaining arguments are delegated to the standard plot function

See Also

pbi-package, pbi-class, pbi_pca.biplot, pbi_pca.pairs, pbi_pca.plot, pbi_pca.toData, pbi_pca.variances, pbi_pca.varplot

Examples

```
par(mfrow=c(1,2),mai=c(0.7,0.7,0.5,0.1))
library(cluster)
data(votes.repub)
pca=prcomp(t(na.omit(votes.repub)))
pbi_pca.corplot(pca)
data(swiss)
pca=prcomp(swiss,scale.=TRUE)
pbi_pca.corplot(pca)
```

pbi_pca.pairs *Improved pairs plot for prcomp objects*

Description

The function `pbi_pca.pairs` provides an improved pairs plot for visualizing the pairwise scores of the individual components of an analyses using the function `prcomp`. In contrast to the default pairs function this plot visualizes in the diagonal as well the variances and a density line for the component scores.

Usage

```
pbi_pca.pairs(pca, n=10, groups=NULL, col='black', pch=19, legend=FALSE, ...)
```

Arguments

`pca`	pca object which was created using the function `prcomp`.
`n`	maximal number of components to visualize, default: 10
`groups`	vector with classes having the same length than the input matrix for prcomp has rows, default: NULL
`col`	colors for the plotting, character, default: 'black'
`pch`	plotting, symbol, default: 19
`legend`	should the legend be displayed on top, default: FALSE
`...`	additional arguments delegated to the standard `pairs` function

See Also

`pbi-package`, `pbi-class`, `pbi_pca.biplot`, `pbi_pca.corplot`, `pbi_pca.plot`, `pbi_pca.toData`, `pbi_pca.variances`, `pbi_pca.varplot`

Examples

```
data(iris)
pci=prcomp(iris[,1:4],scale=TRUE)
pbi_pca.pairs(pci,pch=15,groups=iris[,5],
   legend=TRUE,oma=c(5,4,4,4),col=as.numeric(iris[,5])+1)
```

pbi_pca.plot *Improved bar or screeplot for prcomp objects*

Description

The function `pbi_pca.plot` provides an improved bar- or screeplot for visualizing the variances of the individual components of an analyses using the function `prcomp`. In contrast to the default plot function this plot visualize cumulative and individual variances in percent.

Usage

```
pbi_pca.plot(pca, n=10, type="bar", cex=1.5, legend=TRUE,
    xlab="Components", ylab="Variance (%)", pc.col=c("light blue","grey"), ...)
```

Arguments

`pca`	pca object which was created using the function `prcomp`
`n`	maximal number of components to visualize, default: 10
`type`	plotting type either "bar" or "scree", default: "bar"
`cex`	character expansion for the legend and the screeplot plotting characters, default: 1.5
`legend`	should the legend be displayed on top, default: TRUE
`xlab`	label given to the x-axis, default: 'Components'
`ylab`	label given to the y-axis, default: 'Variance (%)'
`pc.col`	colors for the PC variances, first individual, second color for the cumulative variance, default: c("light blue","grey")
`...`	additional arguments delegated to the standard plot function

See Also

`pbi-package`, `pbi-class`, `pbi_pca.biplot`, `pbi_pca.corplot`, `pbi_pca.pairs`, `pbi_pca.toData`, `pbi_pca.variances`, `pbi_pca.varplot`

Examples

```
data(iris)
par(mfrow=c(1,2))
pcai=prcomp(iris[,1:4],scale=TRUE)
pbi_pca.plot(pcai)
pbi_pca.plot(pcai,type="scree",legend=FALSE)
```

`pbi_pca.toData` *Transform prcomp PCA objects back to data*

Description

The method allows you transform PCA data back to original data. This can be as well used to eliminate some components and then create data by removing the effect of these components.

Usage

```
pbi_pca.toData(pca)
```

Arguments

`pca`	pca object of class prcomp

Value

return data matrix with the original data.

See Also

`pbi-package`, `pbi-class`, `pbi_pca.biplot`, `pbi_pca.corplot`, `pbi_pca.pairs`, `pbi_pca.plot`, `pbi_pca.variances`, `pbi_pca.varplot`

Examples

```
pca=prcomp(iris[,1:4])
head(iris[,1:4])
head(pbi_pca.toData(pca))
# remove effect of first component
pca$rotation[,1]=0
head(pbi_pca.toData(pca))
```

`pbi_pca.variances` *Absolute variance contributions of variables to PC's*

Description

The function returns the absolute variance contributions for each variable to each component. Every squared loading value for each component and variable is multiplied with the component importance. The sum of the returned matrix is therefor 1.

Usage

```
pbi_pca.variances(pca)
```

Arguments

`pca`	a PCA object created with 'prcomp'.

Value

return matrix with absolute variances for each component and variable.

See Also

pbi-package, pbi-class, pbi_pca.biplot, pbi_pca.corplot, pbi_pca.pairs, pbi_pca.plot, pbi_pca.toData, pbi_pca.varplot

Examples

```
pca=prcomp(USArrests,scale=TRUE)
round(pbi_pca.variances(pca),2)
sum(pbi_pca.variances(pca))
```

---

pbi_pca.varplot *PCA variance plot*

Description

The function provides a PCA matrix plot to show associations between PCs and variables. Shown are the squared values, but retaining the original sign of the the variances. So the abolute sum of all values should be one.

Usage

```
pbi_pca.varplot(pca, pcs=10, main="Variance plot", cex.lab=1.5, cex.sym=8,
         cex.var=1, pch=16, ...)
```

Arguments

`pca`	pca object which was created using the function `prcomp`
`pcs`	number of the first PC's or vector of PC names to be plotted, default: 10 (or max number of PC's)
`main`	title of the plot, default: 'Variance plot'
`cex.lab`	character expansion for column and rownames, default: 1.5
`cex.sym`	character expansion for the plotting symbols, default: 8
`cex.var`	character expansion for the variance values, default: 1
`pch`	plotting character for the variances, default: 16 (filled circle)
`...`	remaining arguments are delegated to the standard plot function

See Also

pbi-package, pbi-class, pbi_pca.biplot, pbi_pca.corplot, pbi_pca.pairs, pbi_pca.plot, pbi_pca.toData, pbi_pca.variances, pbi_pca.varplot

Examples

```
data(USArrests)
pbi_pca.varplot(prcomp(USArrests,scale=TRUE),cex.sym=8)
data(swiss)
pca=prcomp(swiss,scale=TRUE)
round(pbi_pca.variances(pca),3)
pbi_pca.varplot(pca,cex.sym=5,cex.var=0.7,cex.lab=1)
```

`pbi_pcor` *Partial correlation test for two variables*

Description

Calculate partial correlation coefficient of either parametric ("Pearson") or non-parametric ("Spearman") statistics corrected for one or more other variables.

Usage

```
pbi_pcor(x,y,z,method='pearson')
```

Arguments

`x`	numeric vector, missing values are allowed
`y`	numeric vector, missing values are allowed
`z`	numeric vector, matrix or data frame, missing values are allowed
`method`	character string indicating which partial correlation coefficient is to be computed, either "pearson" (default), or "spearman"

Value

return partial correlation coefficient between x and y given z.

See Also

`pbi-package`, `pbi-class`, `pbi_pcor.test`

Examples

```
  y.data <- data.frame(
    hl=c(7,15,19,15,21,22,57,15,20,18),
    disp=c(0.000,0.964,0.000,0.000,0.921,0.000,0.000,1.006,0.000,1.011),
    deg=c(9,2,3,4,1,3,1,3,6,1),
     BC=c(1.78e-02,1.05e-06,1.37e-05,7.18e-03,0.00e+00,0.00e+00,0.00e+00,
          4.48e-03,2.10e-06,0.00e+00)
  )
  # partial correlation between "hl" and "disp" given "deg" and "BC"
  pbi_pcor(y.data$hl,y.data$disp,y.data[,c("deg","BC")])
```

`pbi_pcor.test` *Partial correlation test for two variables*

Description

Calculate partial correlation coefficient and parametric ("Pearson") or non-parametric ("Spearman") test statistics for two variables corrected for one or more other variables.

Usage

```
pbi_pcor.test(x,y,z,method='pearson')
```

Arguments

`x`	numeric vector, missing values are allowed
`y`	numeric vector, missing values are allowed
`z`	numeric vector, matrix or data frame, missing values are allowed
`method`	character string indicating which partial correlation coefficient is to be computed, either "pearson" (default), or "spearman"

Value

return list with the following components:

> - _estimate_ - gives the partial correlation coefficient between x and y given z - _p.value_ - gives the p-value of the test - _statistics_ - gives the value of the test statistics - _n_ - gives the number of samples after deleting all the missing samples - _gn_ - gives the number of given variables - _method_ - gives the correlation method used

See Also

`pbi-package`, `pbi-class`, `pbi_pcor`

Examples

```
y.data = data.frame(
 hl=c(7,15,19,15,21,22,57,15,20,18),
 disp=c(0.000,0.964,0.000,0.000,0.921,0.000,0.000,1.006,0.000,1.011),
 deg=c(9,2,3,4,1,3,1,3,6,1),
  BC=c(1.78e-02,1.05e-06,1.37e-05,7.18e-03,0.00e+00,0.00e+00,0.00e+00,
       4.48e-03,2.10e-06,0.00e+00)
)
# partial correlation between "hl" and "disp" given "deg" and "BC"
pbi_pcor.test(y.data$hl,y.data$disp,y.data[,c("deg","BC")])
```

`pbi_prosite2regex` *Convert PROSITE patterns to regular expressions*

Description

This little function converts a PROSITE pattern to a normal regular expression. Please note, that for searching in FASTA files those regular expressions did work correctly if the sequence pattern in the FASTA file is splitted over two files, the sequences should be before fused to single line sequences. The function [pbi_searchFasta](#searchFasta) does this.

Usage

```
pbi_prosite2regex(pattern)
```

Arguments

`pattern`	a PROSITE pattern.

Value

return a "normal" regular expression.

See Also

`pbi-package`, `pbi-class`

Examples

```
pbi_prosite2regex("A-T-x(0,3)-{ALV}-A")
```

`pbi_protscale` *Calculate and plot protscale moving averages*

Description

This functions takes a given input sequence and calculates the hydophobicity averages over a slising window of length 9 using the Kyte-Doolitle scale. The averaged scores can be as well plotted.

Usage

```
pbi_protscale(sequence,plot=FALSE,col='orange')
```

Arguments

`sequence`	either a sequence in FASTA format or a simple sequence or text string.
`plot`	should the biophysical properties be plotted, default: FALSE
`col`	color for the plot

Value

return vector of window averages (invisible).

See Also

pbi-package, pbi-class, pbi_readFasta

Examples

```
fasta="
>sp|P04156.1|PRIO_HUMAN
MANLGCWMLVLFVATWSDLGLCKKRPKPGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHG
GGWGQPHGGGWGQPHGGGWGQGGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIH
FGSDYEDRYYRENMHRYPNQVYYRPMDEYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVV
EQMCITQYERESQAYYQRGSSMVLFSSPPVILLISFLIFLIVG"
pbi_protscale(fasta,plot=TRUE,col="light blue")
```

---

pbi_readFasta *Read in a (small) fasta file into a list*

Description

This function can be used to read in a small FASTA file into a listg where the keys are the ids and the values are are the sequences without line breaks. This code is very slow for large files.

Usage

```
pbi_readFasta(filename)
```

Arguments

filename	a sequence file in FASTA format

Value

return a list with the sequence ids as keys and the sequences as values.

See Also

pbi-package, pbi-class, pbi_searchFasta

Examples

```
fout = file('minions.fasta','w')
cat(">Minion1\nSTSTTS\n>Minion2\nTTTTTT\n>Minion3\nSTSTTT\n",file=fout)
cat(">Minion4\nSTTTTT\n>Minion5\nSSTTTT\n>Minion6\nSSSTST\n",file=fout)
cat("\n>Minion7\nSSSSTT\n", file=fout)
close(fout)
seq=pbi_readFasta("minions.fasta")
seq[7]
M=adist(unlist(seq))
M
plot(hclust(as.dist(M)))
```

pbi_readPepinfo — *Read data from the EMBOSS pepinfo tool*

Description

This is a function to visualize the biophysical properties of a protein using the output of the EMBOSS pepinfo tool which can be accessed online at https://www.ebi.ac.uk/jdispatcher/seqstats/emboss_pepinfo After submitting your sequence you have to use the file at "Result Files->Tool Output" the file ending with ".output".

Usage

```
pbi_readPepinfo(file,region=NULL)
```

Arguments

`file`	the result file from the EMBOSS pepinfo file.
`region`	string matching a certain region such as "Doolittle", if not given all available regions will be shown, default: NULL

Value

return dataframe with the columns: Position, Aminoacid, Result.

See Also

`pbi-package`, `pbi-class`, `pbi_readFasta`

Examples

```
pepfile=system.file("files/pepinfo-spike-sars2.txt",package="pbi")
if (file.exists(pepfile)) {
  pbi_readPepinfo(pepfile)
  res=pbi_readPepinfo(pepfile,region="Doolittle")
  plot(res$Result ~ res$Position,type="h",col="orange")
}
```

pbi_report.chisq.test — *Return a formatted text string for reporting a chisq.test*

Description

Return a formatted text string for reporting a chisq.test.

Usage

```
pbi_report.chisq.test(tab)
```

Arguments

`tab`	a contigency table

Value

return formatted text string for reporting a chisq.test in a LaTeX/Sweave document

See Also

pbi-package, pbi-class, pbi_report.conf.int, pbi_report.pval

Examples

```
azt=as.table(matrix(c(76,399,129,332), byrow=TRUE,ncol=2))
rownames(azt)=c("AZT","Placebo")
colnames(azt)=c("DiseaseProgress", "NoDiseaseProgress")
pbi_report.chisq.test(azt)
```

pbi_report.conf.int *Formatted text strings for reporting a confidence interval*

Description

Return a formatted text string for reporting a confidence interval.

Usage

```
pbi_report.conf.int(ci,round=2)
```

Arguments

ci	a confidence interval consisting of two numerical values
round	rounding digits, default: 2

Value

return a formatted text string for reporting a confidence interval to be included in a LaTeX/Sweave document.

See Also

pbi-package, pbi-class, pbi_report.chisq.test, pbi_report.pval

Examples

```
azt=as.table(matrix(c(76,399,129,332), byrow=TRUE,ncol=2))
rownames(azt)=c("AZT","Placebo")
colnames(azt)=c("DiseaseProgress", "NoDiseaseProgress")
pbi_report.conf.int(prop.test(azt)$conf.int)
```

pbi_report.pval — *Return a p-value for reporting*

Description

Return a p-value for reporting, either giving the three alpha thresholds, <0.05, <0.01, or <0.001 or using the star syntax.

Usage

```
pbi_report.pval(p.val,star=FALSE)
```

Arguments

`p.val`	a numerical p-value.
`star`	boolean, should the one-three star syntax be used, default: FALSE.

Value

return pvalue in shown in alpha-threshold or star syntax. If the p-value is not significant, either the value or any empty string is returned if the star syntax is used.

See Also

`pbi-package`, `pbi-class`, `pbi_report.chisq.test`, `pbi_report.conf.int`

Examples

```
report.pval = pbi_report.pval
report.pval(1/10000)
report.pval(1/10000,star=TRUE)
report.pval(0.02,star=TRUE)
report.pval(0.12,star=TRUE)
report.pval(c(0.001,0.01,0.3,0.02))
```

pbi_searchFasta — *Search a FASTA file with a regular expression*

Description

This function searches FASTA files by removing line breaks within the sequence belonging to the same ID.

Usage

```
pbi_searchFasta(filename,pattern)
```

Arguments

`filename`	a sequence file in FASTA format
`pattern`	a standard regular expression which will be applied on the sequence for the ID

Value

return the matching ID's

See Also

pbi-package, pbi-class, pbi_readFasta

Examples

```
# the pattern splits over two lines for the first sequence
pbi_searchFasta(filename=system.file("files/human-tRNAs.fasta",package="pbi"),
   pattern="GACGC{5}AT{2}CTCT")
```

---

pbi_sem *standard error of the mean*

Description

Calculates the standard error of the mean for a given numerical vector.

Usage

```
pbi_sem(x,na.rm=FALSE)
```

Arguments

x	a numerical vector.
na.rm	logical vecor indicating if missing values should be removed, default: FALSE

Value

return computed standard error of the mean.

See Also

pbi-package, pbi-class, pbi_cv

Examples

```
pbi_sem(rnorm(50,mean=10,sd=3))
pbi_sem(rnorm(1000,mean=10,sd=3))
```

pbi_text2fasta *Convert any Text file to a FASTA file using only Aminoacid letters*

Description

This function loops over the given directory and converts text files to a FASTA file format. Only uppercase letters are used for the output file.

Usage

```
pbi_text2fasta(dir,pattern="*",outfile="stdout")
```

Arguments

`dir`	a directory
`pattern`	regular expression for the files which should be analysed
`outfile`	the output filename, default: 'stdout'

Value

return the number of sequences

See Also

`pbi-package`, `pbi-class`, `pbi_readFasta`

Examples

```
pbi_text2fasta(dir=".",pattern='*.R$',outfile="test.fasta")
pbi_file.head("test.fasta",n=3)
```

pbi_tkregex *Graphical user interface for testing regular expressions*

Description

This function provides a graphical user interface to execute regular expressions either as PROSITE patterns or a standard regular expressions on the text entered or pasted into the text area. Just copy and paste your text into the text area below of the two entry fields. In the entry fields you can write your regular expression either as normal regular expression in the bottom field or as PROSITE pattern in the top entry. Thereafter press ENTER, your found patterns will be highlighted in red and the number of found patterns will be shown as well.

Usage

```
pbi_tkregex()
```

See Also

`pbi-package`, `pbi-class`

Examples

```
  ## Not run:
    pbi_tkregex()

## End(Not run)
```

---

pbi_wininstall — *Create an executable Batch script on Windows for R applications within the users PATH*

Description

The function two files in the users PATH for executables on Windows, a BATCH file and a script file both in the same folder and with the same file prefix. This allows the user to directly execute a Rscript by pressing Win-R as a shortcut and the then entering the application name assume you have a file: _hw.R_

Usage

```
pbi_wininstall(filename="")
```

Arguments

`filename`	optional filename which should be installed directly without asking for the filename, default: ""

See Also

`pbi-package`, `pbi-class`

Examples

```
  ## Not run:
  pbi_wininstall("hw.R")

## End(Not run)
```

---

pbi_wordFreq — *Number of words with a given length in a sequence*

Description

The function creates a sliding window of length ‘wlength‘ over the given text or sequence string creating words of length ‘wlength‘. The number how often a certain word appears is counted. So - ‘AABAAC‘ - contains the words: ‘AA‘, ‘AB‘, ‘BA‘, ‘AA‘ and ‘AC‘.

Usage

```
pbi_wordFreq(seq,wlength=2)
```

Arguments

`seq`	sequence as text string or vector of text strings
`wlength`	word size, default: 2

Value

return list object with words as keys and counts as values or data frame in case input is a vector

See Also

pbi-package, pbi-class

Examples

```
  seq="AAABBBCCCDEFAABBCCDD"
  unlist(pbi_wordFreq(seq))
  unlist(pbi_wordFreq(seq,wlength=4))
  minions=read.table(text='
  Minion1      STSTTS
  Minion2      TTTTTT
  Minion3      STSTTT
  Minion4      STTTTT
  Minion5      SSTTTT
  Minion6      SSSTST
  Minion7      SSSSTT
',row.names=1)
  min=pbi_wordFreq(minions[,1],wlength=2)
  rownames(min)=rownames(minions)
  min
```

---

pbi_xyplot — *Improved XY-plot which as well displays a grid and the correlation coefficient*

Description

This is just a simple illustrative function to demonstrate how a standard R function can be modified and extended using different type of arguments overwriting default settings and using the ellipsis and delegating remaining arguments to the default function

Usage

```
pbi_xyplot(x, y, col="blue", pch=19, grid=TRUE, xlab=NULL, ylab=NULL,
                  ellipse=FALSE,ell.fill=FALSE,show.r=TRUE,...)
```

Arguments

`x`	vector with numerical values, or a matrix or data frame
`y`	vector with numerical values, can be NULL if x is matrix or data frame, default: NULL
`col`	color for the plotting symbols

`pch`	plotting character, default: 19 (filled circle)
`grid`	should be a grid drawn, default: TRUE
`xlab`	xlabel, default: NULL (x)
`ylab`	ylabel, default: NULL (y)
`ellipse`	draw an ellipse for 85
`ell.fill`	should the ellipse be filled for the 85
`show.r`	should the Pearson correlation been shown on top, default: TRUE
`...`	remaining arguments delegated to the standard plot function

See Also

`pbi-package`, `pbi-class`, `pbi_lmplot`

Examples

```
data(iris)
par(mfrow=c(2,2),mai=c(0.8,0.8,0.8,0.1))
pbi_xyplot(iris$Sepal.Width,iris$Sepal.Length,
    xlab="Sepal.Width",ylab="Sepal.Height",
    col=as.numeric(iris$Species)+1)
pbi_xyplot(iris$Petal.Width,iris$Petal.Length,
    xlab="Petal.Width",ylab="Petal.Height",
    col=as.numeric(iris$Species)+1)
legend("bottomright",fill=2:4,legend=levels(iris$Species))
pbi_xyplot(iris$Sepal.Width,iris$Sepal.Length,
    xlab="Sepal.Width",ylab="Sepal.Height",
    col=as.numeric(iris$Species)+1,ellipse=TRUE)
pbi_xyplot(iris$Sepal.Width,iris$Sepal.Length,
    xlab="Sepal.Width",ylab="Sepal.Height",
    col=as.numeric(iris$Species)+1,ellipse=TRUE,ell.fill=TRUE)
```

Index