Package: preputils (via r-universe)

August 26, 2024

Type Package

Title Utilities for Preparation of Data Analysis

Version 1.0.3

Date 2020-05-18

Author Josef Frank

Maintainer Josef Frank <josef.frank@gmx.ch>

Description Miscellaneous small utilities are provided to mitigate issues with messy, inconsistent or high dimensional data and help for preprocessing and preparing analyses.

Imports data.table

License GPL-3

Suggests knitr, rmarkdown

NeedsCompilation no

LazyData true

Date/Publication 2020-05-18 22:10:02 UTC

Repository https://joseffrank.r-universe.dev

RemoteUrl https://github.com/cran/preputils

RemoteRef HEAD

RemoteSha 910fdd8b3429f98357167762a7f517b0b4ddf716

Contents

beta2m 2
coalesce 3
fillmap 3
filterpca 5
fixlimits 6
incons 7
m2beta 7
normalize 8

pcv 9
rmbat 10
vifx 11
write.data 12

Index **13**

`beta2m` *Convert proportional data to M-Values*

Description

Proportional data are commonly modelled using a glm approach with logit link function. When performing the logit transformation in advance separately, simple OLS methods can be applied.

Usage

```
beta2m(b)
```

Arguments

`b`	vector or matrix holding the original data

Details

Data are transformed according to $M = log[2](\frac{b}{1-b})$. The input data are assumed to have the range 0<b<1. Data outside this range will lead to missing values. Corner cases (data of b=0 or b=1) can be handled by use of fixlimits().

Value

A named vector/matrix with same dimensions as b and $log[2]$ transformed values

Examples

```
a <- sapply(c(0.01,0.05,0.5,0.8,0.9),function(x) rbinom(30,100,x)/100)
matplot(a,pch=20)
matplot(beta2m(a),pch=20)
matplot(a,beta2m(a),pch=20)
```

`coalesce` *select 1st existing value of several columns*

Description

Inspired by the respectively named sql function. In a list of vectors with equal length the function for each of the observations selects the first value that is not NA, in the order provided

Usage

```
coalesce(...)
```

Arguments

`...`	vectors holding the values, sepearated by "," (commonly columns of data.frame)

Details

Data are transformed according to

$$b = \frac{2^M}{2^M + 1}$$

Value

A named vector holding the supplemented values

Examples

```
a1 = c(1,NA,NA,NA)
a2 = c(2,2,NA,NA)
a3 = c(NA,3,3,NA)
cbind(a1,a2,a3,suppl=coalesce(a1,a2,a3))
```

`fillmap` *Supplement missing values in mapping of data*

Description

In properly normalized data bases, 1:1 mapping should be complete and unique. In real world data however ID mappings or data base key candidates are repeated over and over across observations, partly with missing data in case of merged data set. fillmap supplements NAs in mapping variables as far as possible

Usage

```
fillmap(x, y, what = "xy", rmdup=FALSE, rmmiss=FALSE,
    printori=FALSE)
```

Arguments

`x, y`	vectors of equal length, holding the mapping values, sepearated by ","
`what`	data to be returned, either 1st ("x") or 2nd argument ("y") or a data,table, cointaining both ("xy")
`rmdup`	remove duplicates from mapping (TRUE) or return all rows in original order (FALSE)
`rmmiss`	remove rows, where not mapping could be found (TRUE) or return all rows (FALSE)
`printori`	print original variables side by side

Details

incons assumes a 1:1 mapping between provided variables, as is commonly the case for example in ID translation steps.

For all cases where a proper unambiguous 1:1 matching exists. The missings values are filled in

Value

Vector or data.table with original mapping data, where NAs are filled in whith supplemented data where possible

Examples

```
library(data.table)
pheno1 <- data.frame(id1=c(1,2,3,4),id2=c(11,22,NA,NA),phenodat=c(NA,NA,NA,"d"))
pheno2 <- data.frame(id1=c(NA,NA,NA),id2=c(11,22,33),phenodat=c("a","b","c"))
pheno3 <- data.frame(id1=c(4,3),id2=c(44,33),phenodat=c(NA,NA))
phenoges <- rbind(rbind(pheno1,pheno2),pheno3)
with(phenoges,fillmap(id1,phenodat))
with(phenoges,fillmap(id1,phenodat,rmdup=TRUE))
with(phenoges,fillmap(id1,phenodat,rmmiss=TRUE))
with(phenoges,fillmap(id1,phenodat,rmdup=TRUE,rmmiss=TRUE))
with(phenoges,fillmap(id2,phenodat))
with(phenoges,fillmap(id2,phenodat,rmdup=TRUE))
with(phenoges,fillmap(id2,phenodat,rmmiss=TRUE))
with(phenoges,fillmap(id2,phenodat,rmdup=TRUE,rmmiss=TRUE))
phenosupp <- with(phenoges,fillmap(id1,id2))
names(phenosupp) <- c("id1","id2")
phenosupp$phenodat <- fillmap(phenosupp$id1,phenoges$phenodat,what="y")
unique(phenosupp)
```

`filterpca` *Filter data set using PCA*

Description

Noise removal in data set by means of using principal component analysis. Optionally calculate distances (reconstruction error and Mahalanobis distance

Usage

```
filterpca(x,npc=NULL,pcs=NULL,scale.=F,
 method=c("k","t"),resulttype=c("p","d","b"),lambda=NULL)
```

Arguments

`x`	data set
`npc`	Number of leading principal components to be used for reconstruction of data set after filtering (positive integer) or number of last components to be skipped (negative integer).
`pcs`	Vector of integers providing column numbers of components to be included for reconstruction (positive numbers) or components to be skipped (negative numbers). In case of mixed signs negative numbers are ignored.
`scale.`	should values be scaled to unit variance before PCA?
`method`	One of either "k" or "t", with following meaning: "k": No further filtering except from ignoring some components when projecting back into original space; "t": Additionally threshold data by setting all value with absolute value below lambda to 0
`resulttype`	Type of resulting value, either matrix of projected values (p), distances (d) or a list containing both (b)
`lambda`	cutoff to be used for thresholding data. Lambda = NULL instructs to use a predefined value of 5% of the mean deviation

Details

The function performs PCA on provided data set. Noise is removed by reconstructing original values either on only a subset of extracted PCs, thresholding PC-scores (setting all values with absolute value below provided cutoff to 0) or a combination of both.

Value

Depending on requested resulttype:

`p`	Matrix with original observations projected back onto original attribute space after filtering
`d`	Data frame with Mahalanobis distance of observations calculated only on subset of requested PCs and with reconstruction error
`b`	List containing both values mentioned above

Examples

```
a = iris[-5]
b0 = filterpca(a,npc=4,res="b")
b1 = filterpca(a,npc=3,res="b")
b2 = filterpca(a,npc=2,res="b")
pairs(b0,pch=20,col=iris$Species)
pairs(b1,pch=20,col=iris$Species)
pairs(b2,pch=20,col=iris$Species)
```

`fixlimits` *Fix extremes for logit transformation*

Description

Change extreme values in proprtional data prior to logit transformation

Usage

```
fixlimits(x)
```

Arguments

`x`	name of vector to adjust

Details

The function assumes a data range of 0<=x<=1. Data outside this range are regarded as measurement errors and recoded to NA. In order to avoid generating missings during logit transformation values >=1 and <=0 respectively are shifted to lie within the range (0,1) excluding the borders themselves by recoding them to the mean of the respective border and and the most extreme nearest neighbour.

Value

vector of same length as x with adjusted values

Examples

```
fixlimits(0:5/5)
```

`incons` *Detect inconsistencies in 1:1 mapping*

Description

In properly normalized data bases, no inconsistencies should be present. In real world data however ID mappings or data base key candidates are repeated over and over across observations, especially in mult centric studies with basic research data. incons tries to detect and flag these mapping discrepanices

Usage

```
incons(x, y, printproblems=FALSE)
```

Arguments

`x, y`	vectors of equal length, holding the mapping values, sepearated by ","
`printproblems`	Should a table of found problems be printed in addition to the returned flag?

Details

incons assumes a 1:1 mapping between provided variables, as is commonly the case for example in ID translation steps

Value

A named vector indicating whether ambiguous mapping does occur (TRUE) or mapping is clean (FALSE)

Examples

```
id1 = c(1,2,2,3,4)
id2 = c("a","b","c","d","d")
ambiguous <- incons(id1,id2,print=TRUE)
data.frame(id1,id2,ambiguous)
```

`m2beta` *Convert logit transformed M-Values of proportional data back to original 0/1 range*

Description

Despite conducting analysis of proportional data in M space, for publication figures the estimated values are commonly shown in the original space (range between 0 and 1). This function provides backscaling of the M values to original space by inverting the logit transformation done by beta2m()

Usage

```
m2beta(M)
```

Arguments

`M`	vector or matrix holding the original data

Details

Data are transformed according to

$$b = \frac{2^M}{2^M + 1}$$

Value

A named vector/matrix with same dimensions as M and transformed values

Examples

```
b = 1:99 / 100
M = beta2m(b)
plot(b,m2beta(M))
print(all.equal(b, m2beta(M)))
```

`normalize` *Normalize numeric variable to range(0,1)*

Description

Changes range of numeric variables to have min=0 and max=1

Usage

```
normalize(x)
```

Arguments

`x`	name of object to normalize

Details

The function changes the range of the named numeric vector to finally have min(x)=0 and max(x)=1.

Value

vector of same length as x with normalized values

Examples

```
normalize(1:5)
```

pcv — *PCA on automatically selected attributes in high dimensional data*

Description

Conduct PCA on variables with biggest variance in high dimensional data matrix

Usage

```
pcv(x, cols=5, sites=5000)
```

Arguments

`x`	name of data matrix
`cols`	number of principal components to extract
`sites`	number of attributes to consider

Details

pcv assumes data in a numeric matrix and variable major format, i.e. every line corresponds to to a variable, while the columns correspond to the individual observations. This is commonly the case for data in high throughput experiments where the number of data points per individuals is high (> 10,000), while the size of batches is comparably small (dozens to hundreds). Variables with missing values are disregarded for the selection.

Use t() to transpose individual major data sets beforehand.

pcv selects the attributes with the highest variance up to the numbers provided, but takes considerations to limit these to the actual size of the present data set.

This is often used as first step in high throughput measurements to detect global effects of known batch variables.

Value

matrix with rows corresponding to observations and columns to extracted components. Values denote the scores on the extracted components for the respective observations.

Examples

```
pcs <- pcv(t(iris[1:4]),cols=2)
cor(pcs,iris[-5])
```

rmbat *batch effect removal by mean centering and shifting*

Description

Remove known categorical batch effects from high dimensional data sets

Usage

```
rmbat(x,batches)
```

Arguments

`x`	name of object to be processed. This is a matrix in atribute major format (rows correspond to variables, columns to observations)
`batches`	Vector with batch identifiers for each of the columns in x

Details

For each variable the mean values of all batches are shifted to the grand mean of the total sample. On case of several independent bacth effects being present in th data set, thes can either be combined in one batch variable, or the batches can be removed one at a time by chaining the processing and caling the cuntiong with each of the batch variables in turn

Value

matrix with same dimensions as x and batch effects removed

Note

This function is intended for use with methods that do not inherently allow inclusion of covariates in the analysis itself, e.g. pca or heatmap. If methods are used that allow inclusion of batches in analysis like linear models, that is preferred, as the method above can otherwise greatly reduce power if batches are correlated with the effect variable

Examples

```
# create data set
n_obs = 8
n_var = 10
predictor <- rep(0:1,n_obs*0.5)
pure_effect <- outer(rnorm(n_var),predictor)
error <- matrix(rnorm(n_var*n_obs),n_var,n_obs)
batch1 <- rep(1:2,each=n_obs*0.5)
batch2 <- rep(c(1,2,1,2),each=n_obs*0.25)
batch_effect1 <- outer(rnorm(n_var)*2,scale(batch1))[,,1]
batch_effect2 <- outer(rnorm(n_var)*4,scale(batch2))[,,1]
batch_effect <- batch_effect1 + batch_effect2
```

```
data_measured <- pure_effect + batch_effect + error

zero = outer(rep(0,n_var),rep(0,n_obs))
b1 <- rmbat(batch_effect1,batch1)
b2 <- rmbat(batch_effect2,batch2)
b12a <- rmbat(batch_effect1,paste(batch1,batch2))
b12b <- batch_effect
all.equal(b1,zero)
all.equal(b2,zero)
all.equal(b12a,zero)
all.equal(b12b,zero)
```

vifx *Compute Variance inflation factor*

Description

Calculate variance inflation factors (VIF) for all numeric variables contained in data set

Usage

```
vifx(x)
```

Arguments

x	name of data frame for which the VIFs should be computed

Details

The function reads in the object named in x, builds a linear model for each of the contained variables in the data set, regressing the selected variable on all other numeric variables contained in the data set.

The multiple R-squared is computed and transformed to VIF using following formula: $VIF_i = \frac{1}{1-R_i^2}$

Value

A named vector with names given by the contained numeric variables and values by the computed respective VIFs

Examples

```
vifx(iris)
```

write.data *Create text data files using convenient defaults*

Description

Several presets are provided for creating text data files. Functions are based on write.table, with some predefined extras to save time when writing data sets to clear text files

Usage

```
write.tab(data, filename, sep="\t", quote=FALSE, row.names=FALSE, ...)
```

Arguments

`data`	name of object to be saved
`filename`	File name
`sep`	Column separator, see details
`quote`	Should text data be quoted? Default FALSE
`row.names`	Whether rownames whould be included in output, default=FALSE
`...`	Further arguments passed on to write.table()

Details

Both of the named functions just use the filename as 2nd positional argument and call write.table(). Difference between both is that write.tab has predefined the column separator as "\t", while write.space uses the write.table default " ".

Examples

```
    ## Not run:
        write.tab(iris,"~/iris_tab.txt")
        write.space(iris,"~/iris_space.txt")

## End(Not run)
```

Index

beta2m, 2

coalesce, 3

fillmap, 3
filterpca, 5
fixlimits, 6

incons, 7

m2beta, 7

normalize, 8

pcv, 9

rmbat, 10

vifx, 11

write.data, 12
write.space *(*write.data*)*, 12
write.tab *(*write.data*)*, 12