Package 'utilities'

January 14, 2021

Type Package

Title Data Utility Functions

Version 0.1.0

Date 2021-01-07

Author Ben O'Neill [aut, cre]

Maintainer Ben O'Neill <ben.oneill@hotmail.com>

Description Data utility functions for use in probability and statistics. Includes functions for computing higher-moments for samples and their decompositions.
Also includes utilities to examine functional mappings between factor variables and other variables in a data set.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports stats

Suggests ggplot2, ggdag

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-14 08:30:02 UTC

R topics documented:

kurtosis *Sample Kurtosis*

Description

kurtosis returns the sample kurtosis of a data vector/matrix

Usage

```
kurtosis(x, kurt.type = NULL, kurt.excess = FALSE, na.rm = FALSE)
```

Arguments

x	A data vector/matrix
kurt.type	The type of kurtosis statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
kurt.excess	Logical value; if TRUE the function gives the excess kurtosis (instead of raw kurtosis)
na.rm	Logical value; if TRUE the function removes NA values

Details

This function computes the sample kurtosis for a data vector or matrix. For a vector input the function returns a single value for the sample kurtosis of the data. For a matrix input the function treats each column as a data vector and returns a vector of values for the sample kurtosis of each of these datasets. The function can compute different types of kurtosis statistics using the kurt.type input.

Value

The sample kurtosis of the data vector/matrix

Examples

```
kurtosis(rnorm(1000))
kurtosis(rexp(1000))
```

log *Logarithm Function*

Description

log returns the logarithm of the input

Usage

```
log(x, base = exp(1))
```

Arguments

x	An input value (numeric/complex scalar or vector)
base	The base for the logarithm (a positive scalar value)

Details

This version of the logarithm function allows both numeric and complex inputs, including negative numeric values. If the output of the logarithm has no complex part then the output is given as a numeric value.

Value

The logarithm of the input

Examples

```
log(1)
log(-1)
```

mappings *Examine mappings between factor variables in a data-frame*

Description

mappings determines the mappings between factor variables in a data-frame

Usage

```
mappings(data, na.rm = TRUE, all.vars = FALSE, plot = TRUE)
```

Arguments

data	A data-frame (or an object coercible to a data-frame)
na.rm	Logical value; if TRUE the function removes NA values from consideration
all.vars	Logical value; if TRUE the function only examines factor variables in the data-frame; if FALSE the function examines all variables in the data-frame (caution is required in interpretation of output)
plot	Logical value; if TRUE the function plots the DAG for the mappings (requires ggplot2 and ggdag to work)

Details

In preliminary data analysis prior to statistical modelling, it is often useful to investigate whether there are mappings between factor variables in a data-frame in order to see if any of these factor variables are redundant (i.e., fully determined by other factor variables). This function takes an input data-frame data and examines whether there are any mappings between the factor variables. (Note that the function will interpret all character variables as factors but will not interpret numeric or logical variables as factors.) The output is a list showing the uniqueness of the binary relations between the factor variables (a logical matrix showing left-uniqueness in the binary relations), the mappings between factor variables, the redundant and non-redundant factor variables, and the directed acyclic graph (DAG) of these mappings (the last element requires the user to have the ggdag package installed; it is omitted if the package is not installed). If plot = TRUE the function also returns a plot of the DAG (if ggdag and ggplot2 packages are installed).

Note that the function also allows the user to examine mappings between all variables in the data-frame (i.e., not just the factor variables) by setting all.vars = TRUE. The output from this analysis should be interpreted with caution; one-to-one mappings between non-factor variables are common (e.g., when two variables are continuous it is almost certain that they will be in a one-to-one mapping), and so the existence of a mapping may not be indicative of variable redundancy.

Note on operation: If na.rm = FALSE then the function analyses the mappings between the factors/variables without removing NA values. In this case an NA value is treated as a missing value that could be any outcome. Consequently, for purposes of determining whether there is a mapping between the variables, an NA value is treated as if it were every possible value. The mapping is falsified if there are at least two identical values in the domain (which may include one or more NA values) that map to different values in the codomain (which may include one or more NA values).

Value

A list object of class 'mappings' giving information on the mappings between the variables

Examples

```
DATA <- data.frame(
  VAR1 = c(0,1,2,2,0,1,2,0,0,1),
  VAR2 = c('A','B','B','B','A','B','B','A','A','B'),
  VAR3 = 1:10,
  VAR4 = c('A','B','C','D','A','B','D','A','A','B'),
  VAR5 = c(1:5,1:5)
)
```

```
# Apply mappings
mappings(DATA, all.vars = TRUE, plot = FALSE)
```

`moments` *Sample Moments*

Description

`moments` returns the sample moments of a data vector/matrix

Usage

```
moments(
  x,
  skew.type = NULL,
  kurt.type = NULL,
  kurt.excess = FALSE,
  na.rm = TRUE,
  include.sd = FALSE
)
```

Arguments

`x`	A data vector/matrix/list
`skew.type`	The type of kurtosis statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
`kurt.type`	The type of kurtosis statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
`kurt.excess`	Logical value; if `TRUE` the function gives the excess kurtosis (instead of raw kurtosis)
`na.rm`	Logical value; if `TRUE` the function removes `NA` values
`include.sd`	Logical value; if `TRUE` the output includes a column for the sample standard deviation (if needed)

Details

This function computes the sample moments for a data vector, matrix or list (sample mean, sample variance, sample skewness and sample kurtosis). For a vector input the function returns a single value for each sample moment of the data. For a matrix or list input the function treats each column/element as a data vector and returns a matrix of values for the sample moments of each of these datasets. The function can compute different types of skewness and kurtosis statistics using the `skew.type`, `kurt.type` and `kurt.excess` inputs. (For details on the different types of skewness and kurtosis statistics, see Joanes and Gill 1998.)

Value

A data frame containing the sample moments of the data vector/matrix

Examples

```
#Create some subgroups of mock data and a pooled dataset
set.seed(1)
N    <- c(28, 44, 51)
SUB1 <- rnorm(N[1])
SUB2 <- rnorm(N[2])
SUB3 <- rnorm(N[3])
DATA <- list(Subgroup1 = SUB1, Subgroup2 = SUB2, Subgroup3 = SUB3)
POOL <- c(SUB1, SUB2, SUB3)

#Compute sample moments for subgroups and pooled data
MOMENTS <- moments(DATA)
POOLMOM <- moments(POOL)

#Compute pooled moments via sample decomposition
sample.decomp(moments = MOMENTS)
```

---

plot.data.mappings *Plot components from data mapping*

Description

This needs ggplot2 and ggdag to function correctly.

Usage

```
## S3 method for class 'data.mappings'
plot(x, node.size = 1, text.size = 1, line.width = 1, ...)
```

Arguments

x	a data mapping
node.size	node size
text.size	label size for a node
line.width	line width
...	not used

Value

nothing

sample.all *All Sampling Variations/Permutations*

Description

`sample.all` returns a matrix of all sampling variations/permutations from a set of integers

Usage

```
sample.all(n, size = n, replace = FALSE, prob = NULL)
```

Arguments

`n`	Number of integers to sample from
`size`	Length of the sample vectors
`replace`	Logical value; if `FALSE` the sampling is without replacement; if `TRUE` the sampling is with replacement
`prob`	Probability vector giving the sampling probability for each element (must be a probability vector with length `n`)

Details

This function computes all sample vectors of size `size` composed of the elements `1,...,n`, either with or without replacement of elements. If `size = n` and `replace = TRUE` then the list of all sample vectors corresponds to a list of all permutations of the integers `1,...,n`.

Value

A matrix of all permutations of the elements `1,...,n` (rows of the matrix give the permutations)

Examples

```
sample.all(n = 4, replace = FALSE)
```

sample.decomp *Sample decomposition*

Description

`sample.decomp` returns the data-frame of sample statistics for sample groups and their pooled sample

Usage

```
sample.decomp(
  moments = NULL,
  n = NULL,
  sample.mean = NULL,
  sample.sd = NULL,
  sample.var = NULL,
  sample.skew = NULL,
  sample.kurt = NULL,
  names = NULL,
  pooled = NULL,
  skew.type = NULL,
  kurt.type = NULL,
  kurt.excess = NULL,
  include.sd = FALSE
)
```

Arguments

moments	A data-frame of moments (an object of class 'moments')
n	A vector of sample sizes
sample.mean	A vector of sample means
sample.sd	A vector of sample standard deviations
sample.var	A vector of sample variances
sample.skew	A vector of sample skewness
sample.kurt	A vector of sample kurotsis
names	A vector of names for the sample groups
pooled	The number of the pooled group (if the pooled group is already present)
skew.type	The type of skewness statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
kurt.type	The type of kurtosis statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
kurt.excess	Logical value; if TRUE the sample kurtosis is the excess kurtosis (instead of the raw kurtosis)
include.sd	Logical value; if TRUE the output includes a column for the sample standard deviation (if needed)

Details

It is often useful to take a set of sample groups with known sample statistics and aggregate these into a single pooled sample and find the sample statistics of the pooled sample. Likewise, it is sometimes useful to take a set of sample groups and a pooled group with known sample statistics and determine the statistics of the other group required to complete the pooled sample. Both of these tasks can be accomplished using decomposition formulae for the sample size, sample mean and sample variance (or sample standard deviation). This function implements either of these two

decomposition methods to find the sample statistics of the pooled sample or the other group remaining to obtain the pooled sample. The user inputs vectors for the sample size, sample mean and sample variance (or sample standard deviation). By default the groups are taken to be separate groups and the function computes the sample statistics for the pooled sample However, the user can input the number pooled sample as the input `pooled`; in this case that group is treated as the pooled sample and the function computes the other sample group required to obtain this pooled sample. The function returns a data-frame showing the sample statistics for all the groups including the pooled sample.

Value

A data-frame of all groups showing their sample sizes and sample moments

See Also

`moments`

`skewness` *Sample Skewness*

Description

`skewness` returns the sample skewness of a data vector/matrix

Usage

```
skewness(x, skew.type = NULL, na.rm = FALSE)
```

Arguments

`x`	A data vector/matrix
`skew.type`	The type of skewness statistic used ('Moment', 'Fisher Pearson' or 'Adjusted Fisher Pearson')
`na.rm`	Logical value; if `TRUE` the function removes `NA` values

Details

This function computes the sample skewness for a data vector or matrix. For a vector input the function returns a single value for the sample skewness of the data. For a matrix input the function treats each column as a data vector and returns a vector of values for the sample skewness of each of these datasets. The function can compute different types of skewness statistics using the `skew.type` input.

Value

The sample skewness of the data vector/matrix

Examples

```
skewness(rnorm(1000))
skewness(rexp(1000))
```

Index