Package: coreStatsNMR (via r-universe)

September 30, 2024

Title Statistical Functions for Core Analysis Tasks at NMR

Version 1.3.6-7

Description A set of statistical functions for use at NMR Group when completing core analysis tasks: frequency tables, cross-tabs, t-tests, proportion tests, etc.

Depends R (>= 4.2.0)

Imports dplyr, tidyr, purrr, data.table, broom, wrapr, scales, magrittr, weights

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Suggests testthat, knitr, rmarkdown, forcats, ggplot2

VignetteBuilder knitr

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

RemoteUrl https://gitlab.com/NMRgroup/corestatsnmr

RemoteRef HEAD

RemoteSha fd4a8effcf4460036c026fffc6ddb3653f20015a

Contents

aggRval *Area-weighted R-value*

Description

Returns a single area-weighted R-value from the two vectors it accepts as arguments: one vector of R-values, and the other of areas associated with each R-value

Usage

```
aggRval(r_val, area)
```

Arguments

`r_val`	Vector of r-values
`area`	Vector of area associated with each r-value

Value

Single area-weighted r-value

Examples

```
aggRval(c(2,5,20), c(10,10,50))
```

confInterval *Calculate confidence interval (using normal distribution)*

Description

Calculate confidence interval (using normal distribution)

Usage

```
confInterval(x, conf_lvl = 0.9)
```

Arguments

`x`	Numerical vector
`conf_lvl`	A number from 0 to 1 indicating confidence level, defaults to 0.9 or 90%

Value

A dataframe summarizing the sample mean and confidence interval

Examples

```
confInterval(runif(100)); confInterval(runif(1e3))
```

`designEffect` *Calculate the design effect for adjusting cluster sampling sizes*

Description

Calculate the design effect for adjusting cluster sampling sizes

Usage

```
designEffect(n_obs, icc)
```

Arguments

`n_obs`	number. Observations in a cluster (e.g. average lamps in a home)
`icc`	number. Intraclass correlation (similarity of clustered data)

Value

A correction factor for sample sizes drawn from clustered units.

References

http://faculty.smu.edu/slstokes/stat6380/deffdoc.pdf

Examples

```
designEffect(35, 0.75)
```

makeWeights *Generate weights for data from sample and population counts*

Description

Assumes data is provided in columns listing each category that is part of the weighting scheme, then a column for the sample n and a column for the general population.

Usage

```
makeWeights(data, sampleVal, populationVal, digits = 5, checkCols = FALSE)

## S3 method for class 'data.frame'
makeWeights(data, sampleVal, populationVal, digits = 5, checkCols = FALSE)
```

Arguments

`data`	A data.frame (or data.table) to add weights to.
`sampleVal`	A string selecting the column in the data with sample counts
`populationVal`	A string selecting the column in the data with population counts
`digits`	A number of digits to use when rounding proportion weights
`checkCols`	A boolean that toggles whether to calculate checks on proportion and population (included as additional columns)

Value

A dataframe with population and proportion weights, as well as optional intermediate calculations.

Examples

```
myData <- data.frame(HairEyeColor)

myData$Population <- round(runif(nrow(myData),10000,20000),0)

makeWeights(data=myData,sampleVal=”Freq”,populationVal = ”Population”)
```

`mode` *Get the mode of a vector of values*

Description

Get the mode of a vector of values

Usage

```
mode(x, show_all = FALSE)
```

Arguments

`x` A vector of values to calculate the mode from

`show_all` A boolean, if FALSE (default) returns a single mode or NA if there are none/multiple. If TRUE, returns multiple modes, if they exist

Value

The mode(s) of the supplied vector.

Source

https://stackoverflow.com/questions/56552709/r-no-mode-and-exclude-na?noredirect=1#comment99692066_56552709

`pairPropTest` *Pairwise proportion comparisons*

Description

Pairwise proportion comparisons

Usage

```
pairPropTest(
  data,
  indexVar,
  valVar,
  grpVar,
  alpha = 0.1,
  n.min = 10,
  p.adjust.method = p.adjust.methods,
  counts = FALSE
)
```

Arguments

`data` A dataset to calculate proportions for and test for statistically significant differences.

`indexVar` string. Selects an index column for the dataset

`valVar` string. Selects the column containing counts of successes in data

`grpVar` string. Selects the column containing counts of trials in data

`alpha` number. Significance level (e.g. 0.05 for 95-pct confidence level)

`n.min` number. Minimum counts to consider

`p.adjust.method`
string. Method for adjusting p-values. See ?p.adjust for more details.

`counts` Boolean. Toggles whether function returns significance results or counts (for diagnostic purposes)

Value

A dataframe showing p-values and statistically significant differences for the pairs of variables chosen

`pairTtest` *Pairwise T-test comparisons*

Description

Pairwise T-test comparisons

Usage

```
pairTtest(
  data,
  valVar,
  grpVar,
  alpha = 0.1,
  n.min = 10,
  p.adjust.method = p.adjust.methods
)
```

Arguments

`data` A dataset to calculate difference testing for and test for statistically significant differences.

`valVar` string. Selects the column containing counts of successes in data

`grpVar` string. Selects the column containing counts of trials in data

`alpha` number. Significance level (e.g. 0.05 for 95-pct confidence level)

`n.min` number. Minimum counts to consider

`p.adjust.method`
string. Method for adjusting p-values. See ?p.adjust for more details.

Value

A dataframe showing p-values and statistically significant differences for the pairs of variables chosen

penTable *Generating a penetration table*

Description

Generating a penetration table

Generating a weighted proportion table (2-way)

Usage

```
penTable(
  data,
  index,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  only_ns,
  accuracy,
  normwt = TRUE,
  tot.label = "Total"
)

## S3 method for class 'data.frame'
penTable(
  data,
  index,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  only_ns = FALSE,
  accuracy = 1,
  normwt = TRUE,
  tot.label = "Total"
)
```

Arguments

`data`	A dataset to calculate weighted proportions
`index`	string. Selects an index column for the dataset
`x`	string. Selects the first variable to find proportions for

`y` string. Selects the second variable to find proportions for

`totWeightVar` string. A string selecting the column to weight the population

`inGroupWeightVar`
string. A string selecting the column to use for in-group weights

`only_ns` Boolean. Toggles whether to return penetration table or intermediate table of n's.

`accuracy` number. A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values.

`normwt` Boolean. if TRUE, normalize weights so that the total weighted count is the same as the unweighted one

`tot.label` string. A string label for totals column

Value

A data.frame or data.table showing a penetration table

propTest *Proportion comparisons*

Description

Proportion comparisons

Usage

```
propTest(
  data,
  indexVar,
  valVar,
  grpVar,
  counts = NULL,
  alpha = 0.1,
  n.min = 10,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

`data` A dataset to calculate proportions for and test for statistically significant differences.

`valVar` string. Selects the column containing counts of successes in data

`grpVar` string. Selects the column containing counts of trials in data `counts` vector. Optional vector of strings containing columns counts for successes and trials (otherwise, function calculates counts from valVar and grpVar)

`alpha` number. Significance level (e.g. 0.05 for 95-pct confidence level)

`n.min` number. Minimum counts to consider

`alternative` string. Specifies the alternative hypothesis. See ?prop.test

Value

A dataframe showing p-values and statistically significant differences for the chosen variables

`statsTable` *Generate a statistical summary table, with optional grouping*

Description

Generate a statistical summary table, with optional grouping

Usage

```
statsTable(
  data,
  summVar,
  groupVar = NULL,
  stats,
  accuracy = NULL,
  totCol = TRUE,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  drop0trailing = FALSE,
  colOrder = NULL
)

## S3 method for class 'data.frame'
statsTable(
  data,
  summVar,
  groupVar = NULL,
  stats,
  accuracy = 1,
  totCol = TRUE,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  drop0trailing = FALSE,
  colOrder = NULL
)
```

```
## S3 method for class 'data.table'
statsTable(
  data,
  summVar,
  groupVar = NULL,
  stats,
  accuracy = 1,
  totCol = TRUE,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  drop0trailing = FALSE,
  colOrder = NULL
)
```

Arguments

`data` A data.frame (or data.table) to use for statistical summary

`summVar` A string selecting the column in ‘data‘ to summarize

`groupVar` A string or list of strings selecting the (optional) columns in ‘data‘ to grop by

`stats` A list of strings selecting summary stats functions (i.e. mean, sd, sum)

`accuracy` A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values.

`totCol` A boolean toggling whether to include a total column

`totWeightVar` A string selecting the column to weight the population

`inGroupWeightVar` A string selecting the column to use for in-group weights

`drop0trailing` A boolean toggling whether to include trailing zeros in the output (converts to strings)

`colOrder` To be deprecated

Value

A data.frame with statistical summary results describing the selected variable.

Examples

```
library(dplyr)

statsTable(iris,
            summVar = "Sepal.Length",
            groupVar = "Species",
            stats = c("n", "min", "max", "weighted.mean", "median", "sd"),
            accuracy = 2)
```

stratRandSample *Conducting stratified random sampling*

Description

Conducting stratified random sampling

Stratified random sampling

Usage

```
stratRandSample(
  data,
  group,
  size,
  select = NULL,
  replace = FALSE,
  bothSets = FALSE,
  keep.rownames = FALSE
)

## S3 method for class 'data.frame'
stratRandSample(
  data,
  group,
  size,
  select = NULL,
  replace = FALSE,
  bothSets = FALSE,
  keep.rownames = NULL
)

## S3 method for class 'data.table'
stratRandSample(
  data,
  group,
  size,
  select = NULL,
  replace = FALSE,
  bothSets = FALSE,
  keep.rownames = FALSE
)
```

Arguments

`data`	A data.frame (or data.table) to use for allocating sample
`group`	string. The column(s) that represent strata

`size` number. If <1, the proportion to take from each stratum. If an integer 1+, the number of samples to take from each stratum. If size is a vector of integers, the number of samples taken for each stratum. Recommended in latter case to use a named vector

`select` list. Named list specifying a subset of strata to use in sampling

`replace` boolean. Toggling whether to sample with replacement

`bothSets` boolean. Toggling whether to return list of sampled and unsampled portions of data

`keep.rownames` For data.tables only. See ?data.table.

Adapted from https://gist.github.com/mrdwab/6424112 and https://gist.github.com/mrdwab/933ffeaa7a1c

Value

A sample of the data passed to the function, optionally accounting for strata.

`tidy.wtd.chi.sq` *Tidy a weighted chi-squared contingency table test*

Description

Tidy a weighted chi-squared contingency table test

Usage

```
## S3 method for class 'wtd.chi.sq'
tidy(x)
```

Arguments

`x` An htest object, such as those created by weights::wtd.chi.sq

Value

A tibble::tibble() with columns for method, coefficients, estimated values, p-value, and other statistics

`tidy.wtd.t.test` *Tidy a weighted t-test object*

Description

Tidy a weighted t-test object

Usage

```
## S3 method for class 'wtd.t.test'
tidy(x)
```

Arguments

x	An htest object, such as those created by weights::wtd.t.test()

Value

A tibble::tibble() with columns for method, coefficients, estimated values, p-value, and other statistics

`wtdFreqTable` *Generating a weighted frequency table (2-way)*

Description

Generating a weighted frequency table (2-way)

Generating a weighted frequency table (2-way)

Usage

```
wtdFreqTable(
  data,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  accuracy = 1,
  normwt = TRUE,
  tot.label = "Statewide",
  colOrder = NULL
)

## S3 method for class 'data.frame'
wtdFreqTable(
  data,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  accuracy = 1,
  normwt = TRUE,
  tot.label = "Total",
  colOrder = NULL
)
```

Arguments

`data` A dataset to calculate weighted frequencies. Only operational for data.frame for now

`x` string. Selects the first variable to find frequencies for

`y` string. Selects the second variable to find frequencies for

`totWeightVar` string. A string selecting the column to weight the population

`inGroupWeightVar` string. A string selecting the column to use for in-group weights

`accuracy` number. A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values.

`normwt` Boolean. if TRUE, normalize weights so that the total weighted count is the same as the unweighted one

`tot.label` string. Label for totals column

`colOrder` vector. Vector of strings to set the order for the colum given by variable x

Value

A data.frame showing a two-way weighted frequency table

`wtdPairPropTest` *Weighted pairwise proportion comparisons*

Description

Weighted pairwise proportion comparisons

Usage

```
wtdPairPropTest(
  data,
  indexVar,
  valVar,
  grpVar,
  weightVar,
  alpha = 0.1,
  n.min = 10,
  p.adjust.method = p.adjust.methods,
  counts = FALSE
)
```

Arguments

`data` A dataset to calculate proportions for and test for statistically significant differences.

`indexVar` string. Selects an index column for the dataset

`valVar` string. Selects the column containing counts of successes in data

`grpVar` string. Selects the column containing counts of trials in data

`weightVar` string. Selects the column containing weights in the data

`alpha` number. Significance level (e.g. 0.05 for 95-pct confidence level)

`n.min` number. Minimum counts to consider

`p.adjust.method`
string. Method for adjusting p-values. See ?p.adjust for more details.

`counts` Boolean. Toggles whether function returns significance results or counts (for diagnostic purposes)

Value

A dataframe showing p-values and statistically significant differences for the pairs of variables chosen

---

`wtdPairTtest` *Pairwise Weighted T-Test comparisons*

Description

Pairwise Weighted T-Test comparisons

Usage

```
wtdPairTtest(
  data,
  valVar,
  grpVar,
  weightVar,
  alpha = 0.1,
  n.min = 10,
  p.adjust.method = p.adjust.methods
)
```

Arguments

`data`	A dataset to calculate difference testing for and test for statistically significant differences.
`valVar`	string. Selects the column containing counts of successes in data
`grpVar`	string. Selects the column containing counts of trials in data
`weightVar`	string. Selects the column containing weights in the data
`alpha`	number. Significance level (e.g. 0.05 for 95-pct confidence level)
`n.min`	number. Minimum counts to consider
`p.adjust.method`	string. Method for adjusting p-values. See ?p.adjust for more details.

Value

A dataframe showing p-values and statistically significant differences for the pairs of variables chosen

`wtdPropTable` *Generating a weighted proportion table (2-way)*

Description

Generating a weighted proportion table (2-way)

Generating a weighted proportion table (2-way)

Usage

```
wtdPropTable(
  data,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  pct_format = TRUE,
  accuracy = 0.1,
  normwt = TRUE,
  tot.label = "Total",
  colOrder = NULL
)

## S3 method for class 'data.frame'
wtdPropTable(
  data,
  x,
  y,
  totWeightVar = NULL,
  inGroupWeightVar = NULL,
  pct_format = TRUE,
  accuracy = 0.1,
  normwt = TRUE,
  tot.label = "Total",
  colOrder = NULL
)
```

Arguments

`data` A dataset to calculate weighted proportions

`x` string. Selects the first variable to find proportions for

`y` string. Selects the second variable to find proportions for

`totWeightVar` string. A string selecting the column to weight the population

`inGroupWeightVar` string. A string selecting the column to use for in-group weights

`pct_format` boolean. Toggles whether proportions are given as decimals or percents (converts to strings)

`accuracy` number. A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values.

`normwt` Boolean. if TRUE, normalize weights so that the total weighted count is the same as the unweighted one

`tot.label` string. A string label for totals column

`colOrder` vector. A vector of strings to set the order for the column given by variable x

Value

A data.frame or data.table showing a two-way weighted proportion table

Index