Package: Dtableone (via r-universe)

August 23, 2024

Type Package

Title Tabular Comparison of Paired Diagnostic Tests

Version 1.1.0

Description Offers statistical methods to compare diagnostic performance between two binary diagnostic tests on the same subject in clinical studies. Includes functions for generating formatted tables to display diagnostic outcomes, facilitating a clear and comprehensive comparison directly through the R console. Inspired by and extending the functionalities of the 'DTComPair', 'tableone', and 'gtsummary' packages.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 4.3.0)

Imports dplyr, epiR (>= 2.0.61), irr (>= 0.84.1), pROC (>= 1.18.5)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Youngmi Park [aut, cre], Soyeon Ahn [aut], Seong Jun Byun [aut]

Maintainer Youngmi Park <02141@snubh.org>

Repository CRAN

Date/Publication 2024-03-25 20:20:02 UTC

Contents

`ci95` *Format Confidence Intervals*

Description

This function formats confidence intervals with specified precision.

Usage

```
ci95(sumtable, my.digit = 1, table.aspercent = FALSE, print.aspercent = TRUE)
```

Arguments

`sumtable`	A numeric vector of length 3 containing the estimate and its lower and upper confidence limits.
`my.digit`	Integer; the number of decimal places to use (default is 1).
`table.aspercent`	Logical; if TRUE, treats ‘sumtable‘ values as percentages (default is FALSE).
`print.aspercent`	Logical; if TRUE, prints output as percentages (default is TRUE).

Value

A character string representing the formatted confidence interval. The returned string is in the format of "estimate (lower limit-upper limit)". For example, a return value of "50.0 (45.0-55.0)" indicates an estimate of 50.0 with a 95 and ‘print.aspercent‘ are both set to TRUE, the estimate and confidence limits are expressed as percentages, facilitating easy interpretation of the interval as a percentage range.

Examples

```
ci95(c(50, 45, 55))
ci95(c(0.5, 0.45, 0.55), my.digit = 2, table.aspercent = TRUE, print.aspercent = TRUE)
```

`CreateTableD2` *Compare Two Binary Diagnostic Tests in Clinical Studies*

Description

This function compares the diagnostic performance between two modalities on the same subject within clinical studies. It generates formatted tables displaying diagnostic outcomes for sensitivity, specificity, accuracy, positive predictive value (PPV), negative predictive value (NPV), and area under the curve (AUC), providing a clear and comprehensive comparison directly through the R console.

Usage

```
CreateTableD2(x, my.printlayout = TRUE)
```

Arguments

`x`	A data frame containing the diagnostic test outcomes and the true disease status.
`my.printlayout`	Logical; if TRUE, prints the result table to the console and possibly saves it to a file.

Value

A list containing three data frames: 'Diseased', 'Non-diseased', and 'Comparison'. - 'Diseased': A data frame showing the contingency table for diseased cases based on the two diagnostic tests. It contains the counts of true positives, false negatives, false positives, and true negatives for the first diagnostic test compared to the second. - 'Non-diseased': A data frame showing the contingency table for non-diseased cases based on the two diagnostic tests. Similar to 'Diseased', it contains counts of true negatives, false positives, false negatives, and true positives. - 'Comparison': A data frame summarizing the diagnostic performance metrics (sensitivity, specificity, accuracy, PPV, NPV, and AUC) for each modality, along with the p-values from statistical tests comparing the two modalities. Each row represents a different metric, with columns for the estimated value of the first modality, the estimated value of the second modality, and the p-value assessing the difference between the two. This structure allows for a comprehensive overview of the comparative diagnostic performance of the two tests, facilitating easy interpretation and analysis.

Examples

```
# Assuming that data1, data2, data3, and data4 are available
# and contain columns `y1`, `y2`, and `d`
# where `y1` and `y2` are the outcomes of the two diagnostic tests,
# and `d` is the true disease status.

data(data1)
data(data2)
data(data3)
data(data4)
```

```
# Checking the structure of one of the datasets
str(data1)

# Creating tables using CreateTableD2 function for each dataset
CreateTableD2(data1)
CreateTableD2(data2)
CreateTableD2(data3)
CreateTableD2(data4)
```

`data1` *data1*

Description

The data1 to run the examples.

`data2` *data2*

Description

The data2 to run the examples.

`data3` *data3*

Description

The data3 to run the examples.

`data4` *data4*

Description

The data4 to run the examples.

loadPackage *Install and Load Required Packages*

Description

This function checks if a package is installed. If not, it stops and suggests the user to install the package manually. Once installed, the package is loaded into the R session.

Usage

```
loadPackage(pkg)
```

Arguments

pkg	A character string naming the package to be loaded.

Value

No return value. This function is called for its side effect of loading a package into the R session. It does not attempt to install the package automatically, relying instead on the user's action based on the provided message.

Examples

```
## Not run:
  # To use this function, make sure the required package is already installed.
  # For example, to load the ggplot2 package, first ensure it's installed:
  # install.packages("ggplot2")
  loadPackage("ggplot2")

## End(Not run)
```

printp *Format P-Values*

Description

This function formats p-values with specified precision. P-values smaller than 0.001 are formatted as "<0.001".

Usage

```
printp(pv, my.digit = 3)
```

Arguments

pv	A numeric value representing a p-value.
my.digit	Integer; the number of decimal places to use (default is 3).

Value

A character string representing the formatted p-value. The function ensures that the returned string is formatted according to the specified number of decimal places (‘my.digit‘). For p-values smaller than 0.001, the function returns "<0.001" to indicate statistical significance at a high level. This formatting helps in distinguishing between different levels of statistical significance and can be particularly useful in reporting the results of statistical tests. The format "0.000" is avoided to provide a clear indication of very small p-values, enhancing the interpretability of statistical outputs. For example, a return value of "<0.001" suggests a very strong evidence against the null hypothesis, whereas a value of "0.045" formatted with ‘my.digit=3‘ suggests a weaker, but still significant, evidence at the 0.05 level.

Examples

```
printp(0.000234)
printp(0.00456, my.digit = 2)
```

Index