Package 'finddataanalysis'

July 21, 2023

Title FIND Data Analysis Functions

Version 0.0.0.9000

Description Includes generic functions that are frequently used for data analyses at FIND.

License `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

Imports assertthat,
checkmate,
dplyr,
DT,
ggplot2,
irr,
lubridate,
plotly,
plyr,
PropCIs,
stats

Suggests forestplot,
knitr,
rmarkdown,
testthat

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

R topics documented:

`age_dob_process` *Age and DOB processing*

Description

The function first calculates the age based on the present age variable and if unavailable date of birth and enrollment/consent date, end then runs the age_groups function to divide into different categories.

Usage

```
age_dob_process(
  data.var,
  age.var,
  dob.var,
  date.var,
  lim1,
  lim2,
  lim3 = NULL,
  lim4 = NULL,
  lim5 = NULL
)
```

Arguments

`data.var`	Data table
`age.var`	Variable containing age in years.
`dob.var`	Variable containing date of birth
`date.var`	Variable containing enrolment/consent date
`lim1`	Age limit for categorization
`lim2`	Age limit for categorization
`lim3`	Age limit for categorization
`lim4`	Age limit for categorization
`lim5`	Age limit for categorization

Value

Returns the data table with a calculated age column and categorized age column added.

Examples

```
library(dplyr)
library(lubridate)


my_data <- data.frame(
  ID = c(1, 2, 3, 4, 5),
  age = c(25, NA, 30, NA, NA),
  dob = c("1980-01-15", "1995-03-22", "1975-08-10", "1988-11-05", "2000-09-30"),
 date_enrollment = c("2023-05-10", "2022-12-01", "2022-02-18", "2023-02-05", "2021-07-20")
)

# Display the data
print(my_data)

# Apply the age_dob_process function to the data
age_dob_process(data.var=my_data, age.var = "age", dob.var = "dob", date.var = "date_enrollment", lim1 = 30, li
```

---

age_groups *Age groups*

Description

Creates a new categorical variable based on age.

Usage

```
age_groups(data.var, age, lim1, lim2, lim3 = NULL, lim4 = NULL, lim5 = NULL)
```

Arguments

data.var	Data table
age	Variable containing age in years.
lim1	Age limit for categorization
lim2	Age limit for categorization
lim3	Age limit for categorization
lim4	Age limit for categorization
lim5	Age limit for categorization

Value

Creates a columns with typical ranges.

Examples

```
df = data.frame(ID = c(1:20), age = round(runif(20, 3, 86)))
age_groups(data.var = df, age = "age", lim1=5, lim2=11, lim3=18, lim4=45, lim5=60)
df = data.frame(ID = c(1:20), AGE = round(runif(20, 3, 25)))
age_groups(data.var = df, age = "AGE", lim1=5, lim2=11, lim3=18)
```

---

categorize_result *categorize_result*

Description

Categorize a subject as "Positive" or "Negative", based on a variable of interest and a specified threshold. The values are bigger or equal to the threshold are categorized as "Positive", others "Negative". A list of thresholds can be provided.

Usage

```
categorize_result(dataframe, variable, thresholds)
```

Arguments

dataframe	The data table containing test results
variable	A continuous variable
thresholds	The thresholds to be applied

Value

Data table with the categorization added

Examples

```
df <- data.frame(results = c(sample(1:100, 50, replace=TRUE)))
categorize_result(dataframe = df, variable = "results", thresholds = 50)
```

---

categorize_result_PCR *categorize_result_PCR*

Description

Same as categorize_result but the values are smaller than the threshold are categorized as "Positive", others "Negative".

Usage

```
categorize_result_PCR(dataframe, variable, thresholds)
```

Arguments

dataframe	The data table containing test results
variable	A continuous variable - PCR Ct data
thresholds	The thresholds to be applied

Value

Data table with category added

CI_Calc *Confidence Interval Calculator*

Description

Calculates the confidence interval of a given proportion with normal approximation.

Usage

```
CI_Calc(alpha, proportion, n)
```

Arguments

alpha	significance level
proportion	the proportion
n	sample size

Value

confidence interval lower and upper bounds

Examples

```
CI_Calc(alpha=0.05, proportion=0.62, n= 500)
```

---

cohen_agreement *Cohen agreement*

Description

Calculates Cohen's kappa and generates a table including number of positive and negative agreement, as well as positive, negative, overall percent agreements, and the Kappa value.

Usage

```
cohen_agreement(data_var, var1, var2)
```

Arguments

data_var	data table
var1	The test with "Positive" and "Negative"results.
var2	The comparator with "Positive" and "Negative"results.

Value

Data table

Examples

```
df <- data.frame(Test1 = c(rep("Positive", 20), rep("Negative", 20)), Test2 = c(rep("Positive", 30), rep("Negat
cohen_agreement(data_var = df, var1 = "Test1", var2 = "Test2")
```

---

compare_sen_spe *compare_sen_spe*

Description

Because these are two independent groups, a Z-score is calculated from the two proportions (i.e. sensitivities or specificities) to be compared. By default, the test is two tailed, but lower or upper tail can be specified in parameters.

Usage

```
compare_sen_spe(s1, s2, n1, n2, two.sided = TRUE)
```

Arguments

s1	Sensitivity/specificity of the first group
s2	Sensitivity/specificity of the second group
n1	Total number of confirmed positives/negatives respectively in the first group
n2	Total number of confirmed positives/negatives respectively in the second group
two.sided	If TRUE, equivalence is tested, if FALSE, directional inferences are made

Value

A dataframe containing Z-Score and the P-value

References

Zhou XH, Obuchowski NA and McClish DK. Statistical Methods in Diagnostic Medicine. 2011;2:193-228

Examples

```
result.df <- data.frame(Tests = c("Test1", "Test2"), TP = c(82, 140), FN = c(18, 60))
result.df$N <- result.df$TP +result.df$FN
result.df$sen <- result.df$TP/result.df$N
compare_sen_spe(s1 = result.df$sen[1], s2 = result.df$sen[2], n1 = result.df$N[1], n2 = result.df$N[2],two.sid
```

composite_for_roc *composite_for_roc*

Description

Define multiple composite references to make an analysis similar to a ROC curve but where diagnostic ability of a binary classifier system is assessed as THE REFERENCE discrimination threshold is varied.

Usage

```
composite_for_roc(ref_1, ref_2, test_refs, data_frame)
```

Arguments

ref_1	Reference 1
ref_2	Reference 2
test_refs	Reference categorized based on multiple thresholds
data_frame	Data table

Value

Data table containing all possible composite references based on the thresholds applied

composite_reference *composite_reference*

Description

Generates a composite reference based on 3 tests. When any of the tests is positive, subject is classified as positive. Works with 2 or 3 reference tests.

Usage

```
composite_reference(ref_1, ref_2, ref_3 = NULL, data_frame)
```

Arguments

ref_1	Test1 containing "Positive or "Negative" Results
ref_2	Test2 containing "Positive or "Negative" Results
ref_3	Test3 containing "Positive or "Negative" Results
data_frame	Data table containing the 3 tests

Value

Data table with composite reference added

Examples

```
df <- data.frame(T1 = c(rep("Positive", 10), rep("Negative", 10), rep("Negative", 10)), T2 = c(rep("Positive",
composite_reference(ref_1 = "T1", ref_2 = "T2", ref_3 = NULL, data_frame = df)
```

composite_reference_majority *Composite Reference Majority*

Description

The function reads the results of 3 reference test results and takes the majority as the final result.

Usage

```
composite_reference_majority(ref_1, ref_2, ref_3, data_frame)
```

Arguments

ref_1	Test1 containing "Positive or "Negative" Results
ref_2	Test2 containing "Positive or "Negative" Results
ref_3	Test3 containing "Positive or "Negative" Results
data_frame	Data table containing the 3 tests

Value

Data table with composite reference added

Examples

```
df <- data.frame(T1 = c(rep("Positive", 10), rep("Negative", 10), rep("Negative", 10)), T2 = c(rep("Positive",
composite_reference_majority(ref_1 = "T1", ref_2 = "T2", ref_3 = "T3", data_frame = df)
```

dx_yield *Diagnostic Yield Calculates Diagnostic yield: # Positives per Index Test/ # Positive by Reference Test*

Description

Diagnostic Yield Calculates Diagnostic yield: # Positives per Index Test/ # Positive by Reference Test

Usage

```
dx_yield(data_arg, ref_arg, index_arg, conf.level = 0.95)
```

Arguments

data_arg	The data frame
ref_arg	Defined reference column
index_arg	The index test results column
conf.level	Level of confidence

Value

Diagnostic yield with confidence intervals

Examples

```
df_ex <- data.frame(REF = c(rep('Positive', 50), rep('Negative', 50)), INDEX = c(rep('Positive', 3), rep('Negat

dx_yield(data_arg = df_ex, ref_arg = "REF", index_arg = "INDEX", conf.level = 0.95)
```

`format_colnames` *format_colnames*

Description

Edits the column names of the table generated by cohen_agreement, to make it more explicit.

Usage

```
format_colnames(data_frame, test1, test2)
```

Arguments

`data_frame`	the data table generated by cohen_agreement
`test1`	Names of the test
`test2`	Name of the comparator

Value

Returns nicely formatted colnames

Examples

```
df <- data.frame(Test1 = c(rep("Positive", 20), rep("Negative", 20)), Test2 = c(rep("Positive", 30), rep("Negat
results <- cohen_agreement(data_var = df, var1 = "Test1", var2 = "Test2")
print(results)
format_colnames(data_frame = results, test1 = "This Test", test2 = "The Other Test")
```

group_means_concordence *Group Means - Supplement for Concordance Analysis*

Description

Calculate group means of a continuous variable of interest in all possible combinations if agreements between three reference columns

Usage

```
group_means_concordence(data_var, mean_var, var1, var2, var3)
```

Arguments

data_var	The data table
mean_var	The continuous variable from which the mean will be calculated
var1	A bifactor variable with "Positive" and "Negative" results.
var2	A bifactor variable with "Positive" and "Negative" results.
var3	A bifactor variable with "Positive" and "Negative" results.

Value

Mean of a numeric column based on groups defined by combination of three reference columns

Examples

```
df <- data.frame(Test1 = c(rep("Positive", 20), rep("Negative", 20)), Test2 = c(rep("Negative", 5), rep("Posit

group_means_concordence(data_var = df, mean_var = "MyVariable", var1 = "Test1", var2 = "Test2", var3 = "Test3")
```

multi_sen_spe *multi_sen_spe for Table Display*

Description

Calculates and displays sensitivity and specificity with confidence intervals from a data frame for multiple index tests and the reference test. To be used for displaying the result in a table.

Usage

```
multi_sen_spe(data_var, list_index, ref, conf.level = 0.95, index_names = NULL)
```

Arguments

data_var	Data table containing tests results
list_index	A list of index tests with "Negative" and "Positive" results
ref	The reference test with "Negative" and "Positive" results
conf.level	The confidence level, 1-alpha – default 95%
index_names	An optional list containing names of the index tests

Value

A data table with TP/FP/TN/FN, sensitivity, specificity and confidence intervals for all index tests specified in the list

Examples

```
library(finddataanalysis)
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega

multi_sen_spe(data_var = df, list_index = c("Index1", "Index2", "Index3"), ref = "Reference", conf.level = 0.95
```

---

multi_sen_spe_dt_out *multi_sen_spe with DT table output*

Description

Executes multi_sen_spe function and creates a nicely formatted table output with download options using DT package

Usage

```
multi_sen_spe_dt_out(
  data_var,
  list_index,
  ref,
  conf.level = 0.95,
  index_names = NULL,
  file_name = "performance_eval"
)
```

Arguments

data_var	Data table containing tests results
list_index	A list of index tests with "Negative" and "Positive" results
ref	The reference test with "Negative" and "Positive" results
conf.level	The confidence level, 1-alpha – default 95%
index_names	An optional list containing names of the index tests
file_name	An optional text containing the name of the file that will be downloaded

Value

A formatted data table displaying performance characteristics

Examples

```
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega

multi_sen_spe_dt_out(data_var = df, list_index = c("Index1", "Index2", "Index3"), ref = "Reference", conf.leve
```

`multi_sen_spe_forest` *multi_sen_spe_forest for forest plots*

Description

Calculates and displays sensitivity and specificity with confidence intervals from a data frame for multiple index tests and the reference test. To be used for displaying the result in a table.

Usage

```
multi_sen_spe_forest(
  data_var,
  list_index,
  ref,
  conf.level = 0.95,
  index_names = NULL
)
```

Arguments

`data_var`	Data table containing test results
`list_index`	A list of index test with "Negative" and "Positive" results
`ref`	The reference test with "Negative" and "Positive" results
`conf.level`	The confidence level, 1-alpha – default 95%
`index_names`	An optional list containing names of the index tests

Value

A data table with TP/FP/TN/FN, sensitivity, specificity and confidence intervals, to be fed into forestplot function.

Examples

```
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega

multi_sen_spe_forest(data_var = df, list_index = c("Index1", "Index2", "Index3"), ref = "Reference", conf.leve
```

multi_sen_spe_out_forest *multi_sen_spe with forest plot output*

Description

Executes multi_sen_spe function and creates forest plots automatically.

Usage

```
multi_sen_spe_out_forest(
  data_var,
  list_index,
  ref,
  conf.level = 0.95,
  index_names = NULL,
  labels = "Tests"
)
```

Arguments

data_var	Data table containing tests results
list_index	A list of index tests with "Negative" and "Positive" results
ref	The reference test with "Negative" and "Positive" results
conf.level	The confidence level, 1-alpha – default 95%
index_names	An optional list containing names of the index tests
labels	An optional text for the axis label

Value

Interactive forest plots in a list

Examples

```
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega
plots <- multi_sen_spe_out_forest(data_var = df, list_index = c("Index1", "Index2", "Index3"), ref = "Reference
plots[[1]]
plots[[2]]
```

nice_table *nice_table*

Description

To create a well formatted table from the basic data frame version of the performance characteristic table generated using dplyr for subgroup analyses, where the data is not formatted in the long format necessarily.

Usage

```
nice_table(data_var, group_name)
```

Arguments

data_var	data frame
group_name	Name of the grouping variable used in subgroups

Value

A nicely formatted DT table

Examples

```
library(dplyr)
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega

sex_ana <- df %>% group_by(Sex) %>% group_modify(~ performance_eval_auto(
  data_var = (.x),
  list_index =c("Index1", "Index2"),
  ref = "Reference",
  conf.level = 0.95,
  index_names = NULL,
  labels = "Test",
  forest_plot = FALSE, #!!!
  table_output = FALSE, #!!!
  file_name = "performance_eval",
  data_long = FALSE, #!!!
  group_var = NULL
))
nice_table(sex_ana, "Sex")
```

---

performance_eval_auto *performance_eval_auto*

Description

Calculates diagnostic accuracy characteristics and generates report quality tables and figures. Executes multi_sen_spe_out_forest, multi_sen_spe_dt_out, and returns a list object with two forest plots and a DT data table for display.The function works with wide and long formats.

Usage

```
performance_eval_auto(
  data_var,
  list_index,
  ref,
  conf.level = 0.95,
  index_names = NULL,
  labels = "Test",
  forest_plot = TRUE,
  table_output = TRUE,
```

```
  file_name = "performance_eval",
  data_long = FALSE,
  group_var = NULL
)
```

Arguments

data_var	Data table containing tests results
list_index	A list of index tests with "Negative" and "Positive" results. If the data is in long format, the name of the column where the results are stored.
ref	The reference test with "Negative" and "Positive" results
conf.level	The confidence level, 1-alpha – default 95%
index_names	An optional list containing names of the index tests. Not used with long data format.
labels	An optional text for the axis labels in forest plots.
forest_plot	Should the forest plots be generated? Default TRUE
table_output	Should a table output be generated? Default TRUE
file_name	An optional text for the file name that will be available for download when the table output is generated
data_long	Is the data in the long format? Default FALSE
group_var	The variable name where the groups are defined. Enter without quotation marks (e.g. ColName instead of "ColName"). Not used with wide data format.

Value

performance evaluation results as plots, formatted table, or simple data frame

Examples

```
df <- data.frame(Index1 = c(rep("Positive", 20), rep("Negative", 20)), Index2 = c(rep("Positive", 5), rep("Nega

# All outputs
eval_output <- performance_eval_auto(data_var = df, list_index = c("Index1", "Index2", "Index3"), ref = "Refere
eval_output$sen_plot
eval_output$spe_plot
eval_output$table

# Forest plot outputs
eval_output_only_forest <- performance_eval_auto(data_var = df, list_index = c("Index1", "Index2", "Index3"),
eval_output_only_forest$sen_plot
eval_output_only_forest$spe_plot
eval_output_only_forest$table #NULL

# A simple data frame output where the table is not formatted. This form is computer friendly.
eval_output_simple_df <- performance_eval_auto(data_var = df, list_index = c("Index1", "Index2", "Index3"), re

eval_output_simple_df


# Performance Evaluation by Groups
data(my_dataset)
head(my_dataset)
```

```
eval_output <- performance_eval_auto(data_var = my_dataset, list_index = "Result", ref = "RefTest", conf.level
```

`ROC_calculate` *Calculate ROC-like characteristics (i.e. based on varying REFRENCE thresholds)*

Description

Calculates sensitivity and false positivity rate based on the multiple reference columns defined by the thresholds FOR an analysis similar to a ROC curve but where diagnostic ability of a binary classifier system is assessed as THE REFERENCE discrimination threshold is varied.

Usage

```
ROC_calculate(data_var, index, ref, conf.level = 0.95, var_name_suffix_cut)
```

Arguments

`data_var`	data table
`index`	index test column containing results: "Positive" or "Negative"
`ref`	a list containing names of the reference test columns
`conf.level`	confidence level
`var_name_suffix_cut`	the suffix based on which the reference test column name to be cut. This will result in suffix + threshold

Value

sensitivity and specificity based on differnt thresholds to classify a reference positive or negative

---

`sens_spe` *Sensitivity and specificity (Table Display)*

Description

Calculate sensitivity and specificity with confidence intervals from a data frame that contains the results of the index and the reference test. To be used for displaying the result in a table.

Usage

```
sens_spe(data_var, index, ref, conf.level = 0.95)
```

Arguments

`data_var`	Data table containing test results
`index`	The index test with "Negative" and "Positive" results
`ref`	The reference test with "Negative" and "Positive" results
`conf.level`	The confidence level, 1-alpha – default 95%

Value

A data table with TP/FP/TN/FN, sensitivity, specificity and confidence intervals

Examples

```
df <- data.frame(Index = c(rep("Positive", 5), rep("Negative", 30), rep("Positive", 5)), Reference = c(rep("Pos
sens_spe(data_var = df, index = "Index", ref = "Reference", conf.level = 0.95)
```

---

sens_spe_for_forest *Sensitivity and specificity (for forest plots)*

Description

Calculate sensitivity and specificity with confidence intervals from a data frame that contains the results of the index and the reference test. To be used for displaying the result in a forest plot.

Usage

```
sens_spe_for_forest(data_var, index, ref, conf.level = 0.95)
```

Arguments

data_var	Data table containing test results
index	The index test with "Negative" and "Positive" results
ref	The reference test with "Negative" and "Positive" results
conf.level	The confidence level, 1-alpha – default 95%

Value

A data table with TP/FP/TN/FN, sensitivity, specificity and confidence intervals

Examples

```
library(forestplot)
library(dplyr)
df <- data.frame(Index = c(rep("Positive", 20), rep("Negative", 20)), Reference = c(rep("Positive", 30), rep("N
forest_data <- cbind(Test = "Index", sens_spe_for_forest(data_var = df, index = "Index", ref = "Reference", con
forest_data %>% forestplot(mean = Sensitivity,
                           lower = SensLower,
                           upper = SensUpper,
                           labeltext = Test,
                           title = "Sensitivity",
                           zero = NA,
                           xticks = c( 0, 20, 40, 60, 80, 100),
                           txt_gp = fpTxtGp(ticks=gpar(cex=1)),
                           boxsize = .1
                           )
```

`temperature_groups` *Regroup axillary temperature values*

Description

Creates a new categorical variable based on the body temperature.

Usage

```
temperature_groups(data.var, temperature)
```

Arguments

`data.var`	Data table
`temperature`	Variable containing body temperature in Celsius

Value

Creates a new column frame with body temperature ranges

Examples

```
df = data.frame(ID = c(1:20), Temperature = runif(20, 36, 42))
temperature_groups(data.var = df, temperature = "Temperature")
```