Package: EQUALrepeat (via r-universe)

September 8, 2024

Type Package

Title Algorithm Driven Time Series Analysis for Researchers without Coding Skills

Version 0.4.0

Date 2024-09-05

Author Kurinchi Gurusamy [aut, cre]

Maintainer Kurinchi Gurusamy <k.gurusamy@ucl.ac.uk>

Depends stringr, stats, ggplot2, DescTools

Imports zip, cowplot, rstatix, irr, forecast, tseries, urca, vars, viridisLite

Description Support functions for R-based 'EQUAL-STATS' software which automatically classifies the data and performs appropriate statistical tests. 'EQUAL-STATS' software is a shiny application with an user-friendly interface to perform complex statistical analysis. Gurusamy,K (2024)<doi:10.5281/zenodo.13354162>.

License GPL (>= 3)

Encoding UTF-8

URL https://sites.google.com/view/equal-group/home

Note This update fixes bugs noted because of global variables not being recognised as global variables in some environments.

NeedsCompilation no

Repository CRAN

Date/Publication 2024-09-06 16:10:12 UTC

Contents

`function.Measurement_Error`
Calculate Measurement Error

Description

Calculates the intra-rater reliability for categorical data using **irr** and the concordance correlation coefficient and the information required for creating Bland-Altman plots using **DescTools**.

Usage

```
function.Measurement_Error(Predefined_lists, rv)
```

Arguments

`Predefined_lists`	A list supplied by 'EQUAL-STATS' application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input

Value

`analysis_outcome`	Whether the analysis was performed successfullly
`plan`	Plan used for analysis
`code`	Part of code generated for performing the analysis in a standalone version of R
`results`	Analysis results
`results_display`	In order to present a single table, multiple results are combined. This results in some numbers stored as text and can cause very wide tables in the shiny output. This issue is fixed wth some modifications to the results table for display purposes.
`plots_list`	A list of plots generated. Returns "" if no plots are generated.
`plots_list_display`	In the shiny application, only one figure is displayed. Therefore, a composite image is created from the plots for display purposes. Some analysis functions may return `NULL`.
`selections`	Selections made by the user for display.
`display_table`	Whether the results table should be displayed in the shiny app.
`display_plot`	Whether the plot should be displayed in the shiny app.

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

function.submit_choices `irr::kendall()` `irr::kappa2()` `irr::kappam.fleiss()` `DescTools::CCC()`

Examples

```
data <- cbind.data.frame(
  `Subject ID` = c(
    "S0001", "S0002", "S0003", "S0004", "S0005",
    "S0006", "S0007", "S0008", "S0009", "S0010",
    "S0011", "S0012", "S0013", "S0014", "S0015",
    "S0016", "S0017", "S0018", "S0019", "S0020",
    "S0021", "S0022", "S0023", "S0024", "S0025",
    "S0026", "S0027", "S0028", "S0029", "S0030"),
  `Centre` = c(
    "C_0001", "C_0002", "C_0002", "C_0002", "C_0002",
    "C_0001", "C_0001", "C_0003", "C_0001", "C_0003",
    "C_0001", "C_0002", "C_0002", "C_0001", "C_0003",
    "C_0002", "C_0002", "C_0003", "C_0001", "C_0002",
    "C_0002", "C_0002", "C_0002", "C_0003", "C_0002",
    "C_0001", "C_0003", "C_0001", "C_0001", "C_0001"),
  `Treatment` = c(
    "Intensive rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Standard rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
    "Standard rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation"),
  `Obesity status` = c(
    "Obese", "Non-obese", "Obese", "Non-obese", "Non-obese",
    "Obese", "Obese", "Obese", "Non-obese", "Obese",
    "Non-obese", "Non-obese", "Obese", "Non-obese", "Obese",
    "Obese", "Non-obese", "Obese", "Obese", "Obese",
    "Non-obese", "Non-obese", "Non-obese", "Obese", "Obese",
    "Non-obese", "Obese", "Obese", "Obese", "Obese"),
  `Unable to walk independently at 6 weeks` = c(
```

```
    "unable", "able", "able", "unable", "able",
    "able", "unable", "unable", "unable", "unable",
    "able", "unable", "able", "unable", "unable",
    "able", "unable", "unable", "unable", "unable",
    "able", "able", "able", "able", "unable",
    "able", "able", "unable", "able", "unable"),
  `Mobility score at 6 months` = c(
    86, 65.1, 48, 99.8, 73.4, 70, 74.7, 36.5, 64.6, 85.4,
    41.7, 60.1, 73.3, 42.4, 55.3, 47.3, 85.9, 63, 64.6, 101.8,
    108.1, 72.3, 96.4, 87.5, 66.2, 92.9, 47.7, 55.8, 56.4, 133.8),
  `Pain at 6 weeks` = c(
    "3_severe", "1_mild", "1_mild", "2_moderate", "1_mild",
    "1_mild", "2_moderate", "2_moderate", "1_mild", "3_severe",
    "1_mild", "2_moderate", "1_mild", "3_severe", "3_severe",
    "1_mild", "2_moderate", "3_severe", "2_moderate", "2_moderate",
    "1_mild", "1_mild", "1_mild", "1_mild", "2_moderate",
    "1_mild", "1_mild", "2_moderate", "1_mild", "2_moderate"),
  `Number of falls within 6 months` = c(
    3, 2, 3, 2, 2, 1, 4, 2, 2, 5,
    3, 2, 2, 2, 5, 3, 2, 2, 3, 4,
    3, 1, 2, 2, 2, 7, 2, 1, 1, 8),
  `Mobility score at 12 months` = c(
    90, 69.1, 52, 103.8, 77.4, 74, 78.7, 40.5, 68.6, 89.4,
    45.7, 64.1, 77.3, 46.4, 59.3, 51.3, 89.9, 67, 68.6, 105.8,
    112.1, 76.3, 100.4, 91.5, 70.2, 96.9, 51.7, 59.8, 60.4, 137.8) ,
  `Admission to care home` = c(
    "Not admitted", "Not admitted", "Admitted", "Not admitted", "Admitted",
    "Admitted", "Not admitted", "Admitted", "Admitted", "Not admitted",
    "Admitted", "Admitted", "Not admitted", "Not admitted", "Not admitted",
    "Not admitted", "Admitted", "Admitted", "Not admitted", "Not admitted",
    "Not admitted", "Admitted", "Not admitted", "Admitted", "Admitted",
    "Admitted", "Admitted", "Admitted", "Admitted", "Not admitted"),
  `Follow-up` = c(
    10, 8, 8, 8, 12, 12, 11, 10, 8, 7,
    8, 6, 9, 6, 9, 8, 10, 8, 11, 9,
    6, 9, 12, 9, 8, 11, 12, 9, 10, 11)
)
# Simulate lists provided by EQUAL-STATS
Predefined_lists <- list(
  main_menu = c(
    'Calculate summary measures',
    'Create plots',
    'Check distribution',
    'Compare sample mean versus population mean',
    'Compare groups/variables (independent samples)',
    'Compare groups/variables (paired samples or repeated measures)',
    'Find the correlation (quantitative variables)',
    'Calculate measurement error',
    'Find the diagnostic accuracy (primary data)',
    'Perform sample size and power calculations (primary data)',
    'Perform survival analysis',
    'Perform regression analysis',
    'Analyse time series',
```

```
    'Perform mixed-effects regression',
    'Perform multivariate regression',
    'Generate hypothesis',
    'Perform sample size and power calculations (effect size approach)',
    'Make correct conclusions (effect size approach)',
    'Find the diagnostic accuracy (tabulated data)'
  ),
  menu_short = c(
    'Summary_Measures',
    'Create_Plots',
    'Check_Distribution',
    'Compare_Sample_Pop_Means',
    'Compare_Groups',
    'Repeated_Measures',
    'Correlation',
    'Measurement_Error',
    'Diagnostic_Accuracy_Primary',
    'Sample_Size_Calculations_Primary',
    'Survival_Analysis',
    'Regression_Analysis',
    'Time_Series',
    'Mixed_Effects_Regression',
    'Multivariate_Regression',
    'Generate_Hypothesis',
    'Sample_Size_Calculations_Effect_size',
    'Make_Conclusions_Effect_size',
    'Diagnostic_Accuracy_Tables'
  )
)
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = NA,
  second_menu_choice = NA,
  entry = entry,
  import_data = NULL,
  same_row_different_row = NA,
  submit_button_to_appear = FALSE,
  summary_measures_choices = c("EQUAL-STATS choice", "Total observations",
  "Missing observations", "Available observations"),
  analysis_outcome = list(),
  code = list(),
  plan = list(),
  results = list(),
  plots_list = list(),
  reports = list()
)
# Store the data in the a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions ####
library(stringr)
```

```
library(irr)
library(DescTools)
library(ggplot2)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Measurement_Error"
rv$second_menu_choice <- "EQUAL-STATS choice"
rv$entry[[1]] <- "Mobility score at 6 months"
rv$entry[[2]] <- "Mobility score at 12 months"
# Final function ####
Results <- function.Measurement_Error(Predefined_lists, rv)
```

`function.rbind_different_column_numbers`
Combine Multiple Dataframes with Different Column Numbers

Description

Base function `rbind.data.frame` requires that the multiple data frames to be combined must have the same column numbers and names. For producing reports for 'EQUAL-STATS', data frames with different column numbers and names are required. This function allows this combination.

Usage

```
function.rbind_different_column_numbers(list, include_columns)
```

Arguments

`list`	A list of data frames to be combined provided as a `list` object.
`include_columns`	Whether the column names of each data frame should be included as the first row indicated as `"Yes"` or `"No"`. Default is `"Yes"`. It is unsual to require `"No"` in this function.

Value

`output`	A data frame with the multiple rows combined.

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

base::rbind.data.frame()

Examples

```
# Create a simulated data frames
df <- cbind.data.frame(
  Age = rnorm(3000, 40, 10),
  Height = rnorm(3000,165,15),
  `Length of hospital stay` = rpois(3000, 8),
  `Number of infections` = rpois(3000, 3)
)
# Calculate means and standard deviations for normally distributed variables
# and median and upper and lower quartiles for non-normally distributed variables
summary_measures <- lapply(1:ncol(df), function(y) {
  if (y<=2) {
    output <- cbind.data.frame(
      Variable = colnames(df)[y],
      Mean = mean(df[,y]),
      `Standard deviation` = sd(df[,y])
    )
  } else {
    output <- cbind.data.frame(
      Variable = colnames(df)[y],
      Median = quantile(df[,y], probs = 0.5),
      `Lower quartile` = quantile(df[,y], probs = 0.25),
      `Upper quartile` = quantile(df[,y], probs = 0.75)
    )
  }
  return(output)
})
# Combine the normally and non-normally distributed variables
normally_distributed_variables <- rbind.data.frame(summary_measures[[1]], summary_measures[[2]])
non_normally_distributed_variables <- rbind.data.frame(summary_measures[[3]], summary_measures[[4]])
# Combining the variables in a single data frame using
# rbind.data.frame causes error
combined_data_frame <- try(rbind.data.frame(normally_distributed_variables,
non_normally_distributed_variables))
combined_data_frame
# Combining the variables in a single data_frame using
# function.rbind_different_column_numbers does not cause error
# Note that data frames must be supplied as a list
# (any number of data frames can be present in the list)
# Final function ####
combined_data_frame_new_function <- function.rbind_different_column_numbers(
list(normally_distributed_variables, non_normally_distributed_variables)
)
combined_data_frame_new_function
```

`function.read_data` *Read a CSV File and Classify Variable Type*

Description

When an user uploads a file in 'EQUAL-STATS' program, the program can automatically classify the variable types based on the nature of the data uploaded. The data and data types are stored in memory. This then determines the options available for questions and the analysis performed. The variable types can be altered using function.read_metadata.

Usage

```
function.read_data(data_file_path)
```

Arguments

`data_file_path`	The path to the data file.

Value

`outcome`	Whether the import was successful.
`message`	The message displayed to the user after the processing. This message also contains the reason for failure if the import was unsuccessful.
`data`	Imported data
`any_type`	All variables in the data
`quantitative`	Quantitative variables in the data
`counts`	Count variables in the data
`categorical`	Categorical variables in the data
`nominal`	Categorical variables without any order in the data
`binary`	Categorical variables with only two possible categories (factors/levels) in the data
`ordinal`	Ordered categorical variables
`date`	Any variables that appear like date
`time`	Any variables that appear like time

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

function.read_metadata

Examples

```
# Create simulated data ####
data <- cbind.data.frame(
  `Subject ID` = c(
    "S0001", "S0002", "S0003", "S0004", "S0005",
    "S0006", "S0007", "S0008", "S0009", "S0010",
    "S0011", "S0012", "S0013", "S0014", "S0015",
    "S0016", "S0017", "S0018", "S0019", "S0020",
    "S0021", "S0022", "S0023", "S0024", "S0025",
    "S0026", "S0027", "S0028", "S0029", "S0030"),
  `Centre` = c(
    "C_0001", "C_0002", "C_0002", "C_0002", "C_0002",
    "C_0001", "C_0001", "C_0003", "C_0001", "C_0003",
    "C_0001", "C_0002", "C_0002", "C_0001", "C_0003",
    "C_0002", "C_0002", "C_0003", "C_0001", "C_0002",
    "C_0002", "C_0002", "C_0002", "C_0003", "C_0002",
    "C_0001", "C_0003", "C_0001", "C_0001", "C_0001"),
  `Treatment` = c(
    "Intensive rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Standard rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
    "Standard rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
    "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation"),
  `Obesity status` = c(
    "Obese", "Non-obese", "Obese", "Non-obese", "Non-obese",
    "Obese", "Obese", "Obese", "Non-obese", "Obese",
    "Non-obese", "Non-obese", "Obese", "Non-obese", "Obese",
    "Obese", "Non-obese", "Obese", "Obese", "Obese",
    "Non-obese", "Non-obese", "Non-obese", "Obese", "Obese",
    "Non-obese", "Obese", "Obese", "Obese", "Obese"),
  `Unable to walk independently at 6 weeks` = c(
    "unable", "able", "able", "unable", "able",
    "able", "unable", "unable", "unable", "unable",
    "able", "unable", "able", "unable", "unable",
    "able", "unable", "unable", "unable", "unable",
    "able", "able", "able", "able", "unable",
    "able", "able", "unable", "able", "unable"),
  `Mobility score at 6 months` = c(
    86, 65.1, 48, 99.8, 73.4, 70, 74.7, 36.5, 64.6, 85.4,
```

```
    41.7, 60.1, 73.3, 42.4, 55.3, 47.3, 85.9, 63, 64.6, 101.8,
    108.1, 72.3, 96.4, 87.5, 66.2, 92.9, 47.7, 55.8, 56.4, 133.8),
  `Pain at 6 weeks` = c(
    "3_severe", "1_mild", "1_mild", "2_moderate", "1_mild",
    "1_mild", "2_moderate", "2_moderate", "1_mild", "3_severe",
    "1_mild", "2_moderate", "1_mild", "3_severe", "3_severe",
    "1_mild", "2_moderate", "3_severe", "2_moderate", "2_moderate",
    "1_mild", "1_mild", "1_mild", "1_mild", "2_moderate",
    "1_mild", "1_mild", "2_moderate", "1_mild", "2_moderate"),
  `Number of falls within 6 months` = c(
    3, 2, 3, 2, 2, 1, 4, 2, 2, 5,
    3, 2, 2, 2, 5, 3, 2, 2, 3, 4,
    3, 1, 2, 2, 2, 7, 2, 1, 1, 8),
  `Mobility score at 12 months` = c(
    90, 69.1, 52, 103.8, 77.4, 74, 78.7, 40.5, 68.6, 89.4,
    45.7, 64.1, 77.3, 46.4, 59.3, 51.3, 89.9, 67, 68.6, 105.8,
    112.1, 76.3, 100.4, 91.5, 70.2, 96.9, 51.7, 59.8, 60.4, 137.8)
)
# Store this in a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
# Load the necessary packages
library(stringr)
# Final function ####
imported_data_types <- function.read_data(data_file_path)
```

`function.Repeated_Measures` *Compare Differences between Repeated Measurements*

Description

Calculates the skewness and kurtosis and results of Shapiro-Wilk test and Kolmogrov-Smirnov tests using **DescTools** and **stats** to determine normality. It uses the the appropriate tests from **stats**, **DescTools**, and **rstatix** to compare differences over time. For quantitative data, it is also possible to perform ANCOVA, i.e., compare the differences in change in measurements over time between groups. It uses **ggplot2** to create plots and **cowplot** to combine multiple plots.

Usage

```
function.Repeated_Measures(Predefined_lists, rv)
```

Arguments

`Predefined_lists`	A list supplied by 'EQUAL-STATS' application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input

Value

`analysis_outcome`
: Whether the analysis was performed successfullly

`plan`
: Plan used for analysis

`code`
: Part of code generated for performing the analysis in a standalone version of R

`results`
: Analysis results

`results_display`
: In order to present a single table, multiple results are combined. This results in some numbers stored as text and can cause very wide tables in the shiny output. This issue is fixed wth some modifications to the results table for display purposes.

`plots_list`
: A list of plots generated. Returns "" if no plots are generated.

`plots_list_display`
: In the shiny application, only one figure is displayed. Therefore, a composite image is created from the plots for display purposes. Some analysis functions may return `NULL`.

`selections`
: Selections made by the user for display.

`display_table`
: Whether the results table should be displayed in the shiny app.

`display_plot`
: Whether the plot should be displayed in the shiny app.

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

function.submit_choices `stats::shapiro.test()` `stats::ks.test()` `DescTools::Kurt()` `DescTools::Skew()` `stats::mcnemar.test()` `DescTools::CochranQTest()` `stats::t.test()` `rstatix::get_anova_table()` `stats::wilcox.test()` `stats::friedman.test()` `ggplot2::ggplot()`

Examples

```
# Create simulated data ####
data <- cbind.data.frame(
  `Subject ID` = c(
    "S0001", "S0002", "S0003", "S0004", "S0005",
    "S0006", "S0007", "S0008", "S0009", "S0010",
    "S0011", "S0012", "S0013", "S0014", "S0015",
```

```
  "S0016", "S0017", "S0018", "S0019", "S0020",
  "S0021", "S0022", "S0023", "S0024", "S0025",
  "S0026", "S0027", "S0028", "S0029", "S0030"),
`Centre` = c(
  "C_0001", "C_0002", "C_0002", "C_0002", "C_0002",
  "C_0001", "C_0001", "C_0003", "C_0001", "C_0003",
  "C_0001", "C_0002", "C_0002", "C_0001", "C_0003",
  "C_0002", "C_0002", "C_0003", "C_0001", "C_0002",
  "C_0002", "C_0002", "C_0002", "C_0003", "C_0002",
  "C_0001", "C_0003", "C_0001", "C_0001", "C_0001"),
`Treatment` = c(
  "Intensive rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
  "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
  "Intensive rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
  "Standard rehabilitation", "Intensive rehabilitation", "Standard rehabilitation",
  "Standard rehabilitation", "Intensive rehabilitation", "Intensive rehabilitation",
  "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
  "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
  "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation",
  "Intensive rehabilitation", "Standard rehabilitation", "Standard rehabilitation",
  "Intensive rehabilitation", "Standard rehabilitation", "Intensive rehabilitation"),
`Obesity status` = c(
  "Obese", "Non-obese", "Obese", "Non-obese", "Non-obese",
  "Obese", "Obese", "Obese", "Non-obese", "Obese",
  "Non-obese", "Non-obese", "Obese", "Non-obese", "Obese",
  "Obese", "Non-obese", "Obese", "Obese", "Obese",
  "Non-obese", "Non-obese", "Non-obese", "Obese", "Obese",
  "Non-obese", "Obese", "Obese", "Obese", "Obese"),
`Unable to walk independently at 6 weeks` = c(
  "unable", "able", "able", "unable", "able",
  "able", "unable", "unable", "unable", "unable",
  "able", "unable", "able", "unable", "unable",
  "able", "unable", "unable", "unable", "unable",
  "able", "able", "able", "able", "unable",
  "able", "able", "unable", "able", "unable"),
`Mobility score at 6 months` = c(
  86, 65.1, 48, 99.8, 73.4, 70, 74.7, 36.5, 64.6, 85.4,
  41.7, 60.1, 73.3, 42.4, 55.3, 47.3, 85.9, 63, 64.6, 101.8,
  108.1, 72.3, 96.4, 87.5, 66.2, 92.9, 47.7, 55.8, 56.4, 133.8),
`Pain at 6 weeks` = c(
  "3_severe", "1_mild", "1_mild", "2_moderate", "1_mild",
  "1_mild", "2_moderate", "2_moderate", "1_mild", "3_severe",
  "1_mild", "2_moderate", "1_mild", "3_severe", "3_severe",
  "1_mild", "2_moderate", "3_severe", "2_moderate", "2_moderate",
  "1_mild", "1_mild", "1_mild", "1_mild", "2_moderate",
  "1_mild", "1_mild", "2_moderate", "1_mild", "2_moderate"),
`Number of falls within 6 months` = c(
  3, 2, 3, 2, 2, 1, 4, 2, 2, 5,
  3, 2, 2, 2, 5, 3, 2, 2, 3, 4,
  3, 1, 2, 2, 2, 7, 2, 1, 1, 8),
`Mobility score at 12 months` = c(
  90, 69.1, 52, 103.8, 77.4, 74, 78.7, 40.5, 68.6, 89.4,
  45.7, 64.1, 77.3, 46.4, 59.3, 51.3, 89.9, 67, 68.6, 105.8,
```

```
    112.1, 76.3, 100.4, 91.5, 70.2, 96.9, 51.7, 59.8, 60.4, 137.8) ,
  `Admission to care home` = c(
    "Not admitted", "Not admitted", "Admitted", "Not admitted", "Admitted",
    "Admitted", "Not admitted", "Admitted", "Admitted", "Not admitted",
    "Admitted", "Admitted", "Not admitted", "Not admitted", "Not admitted",
    "Not admitted", "Admitted", "Admitted", "Not admitted", "Not admitted",
    "Not admitted", "Admitted", "Not admitted", "Admitted", "Admitted",
    "Admitted", "Admitted", "Admitted", "Admitted", "Not admitted"),
  `Follow-up` = c(
    10, 8, 8, 8, 12, 12, 11, 10, 8, 7,
    8, 6, 9, 6, 9, 8, 10, 8, 11, 9,
    6, 9, 12, 9, 8, 11, 12, 9, 10, 11)
)
# Simulate lists provided by EQUAL-STATS
Predefined_lists <- list(
  main_menu = c(
    'Calculate summary measures',
    'Create plots',
    'Check distribution',
    'Compare sample mean versus population mean',
    'Compare groups/variables (independent samples)',
    'Compare groups/variables (paired samples or repeated measures)',
    'Find the correlation (quantitative variables)',
    'Calculate measurement error',
    'Find the diagnostic accuracy (primary data)',
    'Perform sample size and power calculations (primary data)',
    'Perform survival analysis',
    'Perform regression analysis',
    'Analyse time series',
    'Perform mixed-effects regression',
    'Perform multivariate regression',
    'Generate hypothesis',
    'Perform sample size and power calculations (effect size approach)',
    'Make correct conclusions (effect size approach)',
    'Find the diagnostic accuracy (tabulated data)'
  ),
  menu_short = c(
    'Summary_Measures',
    'Create_Plots',
    'Check_Distribution',
    'Compare_Sample_Pop_Means',
    'Compare_Groups',
    'Repeated_Measures',
    'Correlation',
    'Measurement_Error',
    'Diagnostic_Accuracy_Primary',
    'Sample_Size_Calculations_Primary',
    'Survival_Analysis',
    'Regression_Analysis',
    'Time_Series',
    'Mixed_Effects_Regression',
    'Multivariate_Regression',
    'Generate_Hypothesis',
```

```
    'Sample_Size_Calculations_Effect_size',
    'Make_Conclusions_Effect_size',
    'Diagnostic_Accuracy_Tables'
  )
)
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = NA,
  second_menu_choice = NA,
  entry = entry,
  import_data = NULL,
  same_row_different_row = NA,
  submit_button_to_appear = FALSE,
  summary_measures_choices = c("EQUAL-STATS choice", "Total observations",
  "Missing observations", "Available observations"),
  analysis_outcome = list(),
  code = list(),
  plan = list(),
  results = list(),
  plots_list = list(),
  reports = list()
)
# Store the data in a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions ####
library(stringr)
library(DescTools)
library(ggplot2)
library(cowplot)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Repeated_Measures"
rv$second_menu_choice <- "EQUAL-STATS choice"
rv$entry[[1]] <- "Subject ID"
rv$entry[[2]] <- "Mobility score at 6 months"
rv$entry[[3]] <- "Mobility score at 12 months"
rv$entry[[4]] <- ""
rv$entry[[5]] <- "0.05"
rv$same_row_different_row <- "Same row"
# Final function ####
Results <- function.Repeated_Measures(Predefined_lists, rv)
```

---

function.Time_Series *Analysis Time Series Data*

Description

Performs analysis of time series data and predict the variables over time. It takes into account the seasonality and trend of data and uses **forecast**, **vars**, **urca** to make the predictions. It uses **ggplot2** to create plots.

Usage

```
function.Time_Series(Predefined_lists, rv)
```

Arguments

`Predefined_lists`	A list supplied by 'EQUAL-STATS' application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input

Value

`analysis_outcome`	Whether the analysis was performed successfullly
`plan`	Plan used for analysis
`code`	Part of code generated for performing the analysis in a standalone version of R
`results`	Analysis results
`results_display`	In order to present a single table, multiple results are combined. This results in some numbers stored as text and can cause very wide tables in the shiny output. This issue is fixed wth some modifications to the results table for display purposes.
`plots_list`	A list of plots generated. Returns "" if no plots are generated.
`plots_list_display`	In the shiny application, only one figure is displayed. Therefore, a composite image is created from the plots for display purposes. Some analysis functions may return `NULL`.
`selections`	Selections made by the user for display.
`display_table`	Whether the results table should be displayed in the shiny app.
`display_plot`	Whether the plot should be displayed in the shiny app.

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

function.submit_choices stats::ts() forecast::auto.arima() forecast::forecast.fracdiff() forecast::findfrequency() forecast::ndiffs() tseries::adf.test() urca::ca.jo() vars::vec2var() vars::VAR() viridisLite::viridis() ggplot2::ggplot() cowplot::plot_grid()

Examples

```
# Create simulated data ####
data <- cbind.data.frame(
  Date = as.Date("2023-04-01") + 0:99,
  `Currency exchange rate`= rnorm(100, mean = 95, sd = 5)
)
# Simulate lists provided by EQUAL-STATS
Predefined_lists <- list(
  main_menu = c(
    'Calculate summary measures',
    'Create plots',
    'Check distribution',
    'Compare sample mean versus population mean',
    'Compare groups/variables (independent samples)',
    'Compare groups/variables (paired samples or repeated measures)',
    'Find the correlation (quantitative variables)',
    'Calculate measurement error',
    'Find the diagnostic accuracy (primary data)',
    'Perform sample size and power calculations (primary data)',
    'Perform survival analysis',
    'Perform regression analysis',
    'Analyse time series',
    'Perform mixed-effects regression',
    'Perform multivariate regression',
    'Generate hypothesis',
    'Perform sample size and power calculations (effect size approach)',
    'Make correct conclusions (effect size approach)',
    'Find the diagnostic accuracy (tabulated data)'
  ),
  menu_short = c(
    'Summary_Measures',
    'Create_Plots',
    'Check_Distribution',
    'Compare_Sample_Pop_Means',
    'Compare_Groups',
    'Repeated_Measures',
    'Correlation',
    'Measurement_Error',
    'Diagnostic_Accuracy_Primary',
    'Sample_Size_Calculations_Primary',
    'Survival_Analysis',
    'Regression_Analysis',
```

```
    'Time_Series',
    'Mixed_Effects_Regression',
    'Multivariate_Regression',
    'Generate_Hypothesis',
    'Sample_Size_Calculations_Effect_size',
    'Make_Conclusions_Effect_size',
    'Diagnostic_Accuracy_Tables'
  )
)
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = NA,
  second_menu_choice = NA,
  entry = entry,
  import_data = NULL,
  same_row_different_row = NA,
  submit_button_to_appear = FALSE,
  summary_measures_choices = c("EQUAL-STATS choice", "Total observations",
  "Missing observations", "Available observations"),
  analysis_outcome = list(),
  code = list(),
  plan = list(),
  results = list(),
  plots_list = list(),
  reports = list()
)
# Store the data in a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions ####
library(stringr)
library(forecast)
library(viridisLite)
library(cowplot)
library(ggplot2)
rv$import_data <- function.read_data(data_file_path)
rv$first_menu_choice <- "Time_Series"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Currency exchange rate"
rv$entry[[2]] <- "Date"
rv$entry[[3]] <- ""
rv$entry[[4]] <- ""
rv$entry[[5]] <- "Univariate"
rv$entry[[6]] <- 10
# Final function ####
Results <- function.Time_Series(Predefined_lists, rv)
```

---

round_near *Rounds a Variable to the Nearest Pretty Number*

Description

For some graphs, **Base** `pretty` function may not provide the correct rounding. This is a different algorithm suitable for the graphs produced in 'EQUAL-STATS' software.

Usage

```
round_near(x)
```

Arguments

`x`	A numeric variable.

Value

A "pretty number" suitable for use in graphs.

Note

This is part of a suite of functions required to allow 'EQUAL-STATS' program to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`base::pretty()`

Examples

```
x <- 7
round_near(x)

x <- 754
round_near(x)
```

Index