Package ‘EQUALSTATS’

September 23, 2024

Title Algorithm Driven Statistical Analysis for Researchers without Coding Skills

Version 0.5.0

Date 2024-09-11

Author Kurinchi Gurusamy [aut, cre]

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

Depends stringr, shiny, zip

Imports ggplot2, DescTools, cowplot, boot, pwr, ggcorrplot, survival, nnet, MASS, lmtest, pROC, ThresholdROC, ggsurvfit, lme4, mclogit, ordinal, coxme, MuMIn

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 in function.Create_Plots, function.Diagnostic_Accuracy_Tables, and function.Sample_Size_Calculations_Primary, when typos and non-entries caused errors in results.

NeedsCompilation no

Repository CRAN

Date/Publication 2024-09-23 08:30:02 UTC

Contents

`compile_questions` *Compiles Questions for User Interface*

Description

Obtains the questions related to a particular selection and converts them to shiny commands so that the user interface is created.

Usage

```
compile_questions(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

`UI_Initial_texts`	User interface initial texts
`UI_Update_texts`	User interface updated texts
`submit_button_appear_text`	When the submit button should appear in the user interface

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

Examples

```
# Simulate lists provided by EQUAL-STATS ####
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
Predefined_lists <- list(
main_menu = c(
'Calculate summary measures',
'Create plots'
),
menu_short = c(
'Summary_Measures',
'Create_Plots'
),
second_menu_choices = c(
'',
'EQUAL-STATS choice%__%Histogram'
),
label_1 = c(
'Select the variable for which summary measures are required',
'Select the variable%__%Select the variable'
),
label_2 = c(
'Select the variable for which you want separate summary (optional)',
'NULL%__%NULL'
),
label_3 = c(
'Select the summary measures that you want in the report',
'Enter the title for the plot%__%Enter the title for the plot'
),
label_4 = c(
'',
'Select the variable%__%Select the variable'
),
label_5 = c(
'',
''
),
label_6 = c(
  '',
  ''
),
label_7 = c(
  '',
  ''
),
label_8 = c(
  '',
  ''
),
label_9 = c(
'',
''
),
label_10 = c(
'',
''
),
label_11 = c(
'',
''
),
label_12 = c(
'',
''
),
label_13 = c(
'',
''
),
label_14 = c(
'',
''
),
label_15 = c(
'',
''
),
entry_1 = c(
'%_%selectInput%_%rv$import_data$any_type',
'%_%selectInput%_%rv$import_data$any_type'
),
entry_2 = c(
'%_%selectInput%_%c("",setdiff(rv$import_data$categorical, rv$entry[[1]]))',
'NULL%__%NULL'
),
entry_3 = c(
'%_%checkbox%_%rv$summary_measures_choices',
'%_%text%_%"Plot title"%__%%_%text%_%"Plot title"'
),
entry_4 = c(
'',
'%_%selectInput%_%rv$entry[[1]]%__%%_%selectInput%_%rv$entry[[1]]'
),
entry_5 = c(
'',
''
),
entry_6 = c(
'',
''
),
entry_7 = c(
'',
''
),
entry_8 = c(
'',
''
),
entry_9 = c(
'',
''
),
entry_10 = c(
'',
''
),
entry_11 = c(
'',
''
),
entry_12 = c(
'',
''
),
entry_13 = c(
'',
''
),
entry_14 = c(
'',
''
),
entry_15 = c(
'',
''
),
mandatory_1 = c(
'yes',
'yes%__%yes'
),
mandatory_2 = c(
'no',
'NULL%__%NULL'
),
mandatory_3 = c(
'yes',
'no%__%no'
),
mandatory_4 = c(
'',
'no%__%no'
),
mandatory_5 = c(
'',
''
),
mandatory_6 = c(
'',
''
),
mandatory_7 = c(
'',
''
),
mandatory_8 = c(
'',
''
),
mandatory_9 = c(
'',
''
),
mandatory_10 = c(
'',
''
),
mandatory_11 = c(
'',
''
),
mandatory_12 = c(
'',
''
),
mandatory_13 = c(
'',
''
```

```
),
mandatory_14 = c(
'',
''
),
mandatory_15 = c(
'',
''
),
numeric_exemptions = c(
'',
''
)
)
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = "Create_Plots",
  second_menu_choice = "EQUAL-STATS choice",
  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()
)
# Functions and packages required to run
library(stringr)
# Final function ####
UI_texts <- compile_questions(Predefined_lists = Predefined_lists, rv = rv)
```

convert_to_user_interface

Converts the Compiled Questions to User Interface

Description

Obtains the compiled questions related to a particular selection and converts each of these questions them to shiny commands so that user interface is created.

Usage

```
convert_to_user_interface(UI_name, label, entry_text, rv)
```

Arguments

UI_name	Text provided by the compile_questions() function
label	Text provided by the compile_questions() function
entry_text	Text provided by the compile_questions() function
rv	A list supplied by 'EQUAL-STATS' application based on user input

Value

output_1	User interface initial text
output_2	User interface updated text

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

Examples

```
# Simulate lists provided by EQUAL-STATS ####
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
UI_name <- "entry_1_UI"
label = "Select the variable for which summary measures are required"
entry_text = "%_%selectInput%_%rv$import_data$any_type"
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = "Summary_Measures",
  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()
)
# Packages required to run
```

```
library(stringr)
# Final function ####
UI_texts <- convert_to_user_interface(UI_name = UI_name,
label = label, entry_text = entry_text, rv = rv)
```

---

function.Check_Distribution   *Check the Distribution of Data*

Description

Uses the functions from **stats** and reports the results of Shapiro-Wilk test for normality along with Kolmogrov Smirnov tests for multiple distribution types. It uses **ggplot2** to create plots and **cowplot** to combine multiple plots.

Usage

```
function.Check_Distribution(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() ggplot2::ggplot() cowplot::plot_grid()

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(ggplot2)
library(cowplot)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Check_Distribution"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Mobility score at 6 months"
# Final function ####
Results <- function.Check_Distribution(Predefined_lists, rv)
```

function.Compare_Groups

Compare Differences Between Groups

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** to compare groups.It uses **ggplot2** to create plots and **cowplot** to combine multiple plots.

Usage

```
function.Compare_Groups(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::fisher.test()` `stats::prop.trend.test()` `stats::chisq.test()` `stats::t.test()` `stats::aov()` `stats::wilcox.test()` `stats::kruskal.test()` `boot::boot()` `ggplot2::ggplot()` `cowplot::plot_grid()`

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 <- "Compare_Groups"
rv$second_menu_choice <- "EQUAL-STATS choice"
rv$entry[[1]] <- "Unable to walk independently at 6 weeks"
rv$entry[[2]] <- "Treatment"
rv$entry[[3]] <- "0.05"
# Final function ####
Results <- function.Compare_Groups(Predefined_lists, rv)
```

`function.Compare_Sample_Pop_Means` *Compare the Sample Mean with Population Mean*

Description

Uses the appropriate tests from **stats** to compare the sample mean with the population mean.

Usage

```
function.Compare_Sample_Pop_Means(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::t.test()` `stats::binom.test()`

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)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Compare_Sample_Pop_Means"
rv$second_menu_choice <- "Categorical variable"
rv$entry[[1]] <- "Obesity status"
rv$entry[[2]] <- 0.4
# Final function ####
Results <- function.Compare_Sample_Pop_Means(Predefined_lists, rv)
```

function.Correlation *Calculate the Correlation between Quantitative Variables*

Description

Calculates the correlation between two quantitative variables using **stats**. It uses **ggcorrplot** to provide a visual representation of the correlation.

Usage

```
function.Correlation(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::cor()` `ggcorrplot::ggcorrplot()`

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 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(ggcorrplot)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Correlation"
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.Correlation(Predefined_lists, rv)
```

`function.Create_Plots` *Create Plots*

Description

Uses **ggplot2** to create various plots.

Usage

```
function.Create_Plots(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 `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(ggplot2)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Create_Plots"
rv$second_menu_choice <- "EQUAL-STATS choice"
rv$entry[[1]] <- "Mobility score at 6 months"
rv$entry[[2]] <- ""
rv$entry[[3]] <- ""
rv$entry[[4]] <- ""
# Final function ####
Results <- function.Create_Plots(Predefined_lists, rv)
```

`function.Diagnostic_Accuracy_Primary`
Calculate the Diagnostic accuracy using Primary Data

Description

Uses **ThresholdROC** and **pROC** to calculate the diagnostic accuracy of a test using a reference standard. This function is used when primary data is available. Use `function.Diagnostic_Accuracy_Tables()` when the data is provided as a 2 x 2 contigency table.

Usage

```
function.Diagnostic_Accuracy_Primary(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 function.Diagnostic_Accuracy_Tables `pROC::roc()` `pROC::ggroc()` `ThresholdROC::diagnosti`

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(pROC)
library(ThresholdROC)
library(ggplot2)
rv$import_data <- function.read_data(data_file_path)
```

```
# Update choices ####
rv$first_menu_choice <- "Diagnostic_Accuracy_Primary"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Treatment"
rv$entry[[2]] <- "Unable to walk independently at 6 weeks"
# Final function ####
Results <- function.Diagnostic_Accuracy_Primary(Predefined_lists, rv)
```

function.Diagnostic_Accuracy_Tables

Calculate the Diagnostic accuracy using Summary Data

Description

Uses **ThresholdROC** and **pROC** to calculate the diagnostic accuracy of a test using a reference standard. This function is used when the data is provided as a 2 x 2 contigency table. Use `function.Diagnostic_Accuracy_Primary()` when primary data is available.

Usage

```
function.Diagnostic_Accuracy_Tables(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 function.Diagnostic_Accuracy_Primary `pROC::roc()` `pROC::ggroc()` `ThresholdROC::diagnost`

Examples

```
# 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()
)
# Load the necessary packages and functions ####
library(stringr)
library(pROC)
library(ThresholdROC)
library(ggplot2)
rv$first_menu_choice <- "Diagnostic_Accuracy_Tables"
rv$second_menu_choice <- NA
rv$entry[[1]] <- 30
rv$entry[[2]] <- 2
rv$entry[[3]] <- 1
rv$entry[[4]] <- 40
# Final function ####
Results <- function.Diagnostic_Accuracy_Tables(Predefined_lists, rv)
```

function.Generate_Hypothesis

Generate Research Hypothesis

Description

Generates the research hypothesis that can be used in grant applications, study protocols, and scientific reports when information is provided in plain language.

Usage

```
function.Generate_Hypothesis(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

Examples

```
# 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()
)
# Load the necessary packages and functions ####
library(stringr)
rv$first_menu_choice <- "Generate_Hypothesis"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Intensive rehabilitation"
rv$entry[[2]] <- "Standard rehabilitation"
rv$entry[[3]] <- "Intervention is better or worse than comparator"
rv$entry[[4]] <- "Mobility score"
rv$entry[[5]] <- "Higher values of the outcome (or more events) are better for the subject"
rv$entry[[6]] <- 10
rv$entry[[7]] <- ""
# Final function ####
Results <-  function.Generate_Hypothesis(Predefined_lists, rv)
```

`function.Make_Conclusions_Effect_size` *Make Conclusions*

Description

Generates conclusions when summary information, information used for sample size, and the diffences between the groups are provided. The summary information can be generated from function.Summary_Measures and the difference between the groups can be calculated using function.Compare_Groups. It uses the **pwr** to calculate the sample size.

Usage

```
function.Make_Conclusions_Effect_size(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 function.Summary_Measures function.Compare_Groups `pwr::pwr.t.test()` `pwr::pwr.2p.test()` `ggplot2::ggplot()`

Examples

```
# 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()
)
# Load the necessary packages and functions ####
library(stringr)
library(pwr)
library(ggplot2)
rv$first_menu_choice <- "Make_Conclusions_Effect_size"
rv$second_menu_choice <- "Intervention study (binary outcome)"
rv$entry[[1]] <- 0.67
rv$entry[[2]] <- 0.7
rv$entry[[3]] <- -0.12
rv$entry[[4]] <- 0.09
rv$entry[[5]] <- 40
rv$entry[[6]] <- 40
rv$entry[[7]] <- "Intervention is better or worse than comparator"
rv$entry[[8]] <- "Independent samples"
rv$entry[[9]] <- "Higher values of the outcome (or more events) are better for the subject"
rv$entry[[10]] <- 0.05
rv$entry[[11]] <- "0.05"
rv$entry[[12]] <- "0.80"
# Final function ####
Results <- function.Make_Conclusions_Effect_size(Predefined_lists, rv)
```

`function.Mixed_Effects_Regression`

Perform Mixed Effects Regression

Description

Performs mixed effects regression analysis using **lme4** for mixed-effects linear, logistic, and Poisson regression, **mclogit** for mixed-effects mutinomial logistic regression, **ordinal** for mixed-effects ordinal regression, and **coxme** for mixed-effects Cox regression for binary outcomes. It uses **lmerTest** and **MuMIn** for performing stepwise regression. For linear, logistic, and Poisson regression, it chooses the best optimizer using `lme4::allFit()`.

Usage

```
function.Mixed_Effects_Regression(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 function.Regression_Analysis function.Multivariate_Regression `DescTools::BoxCox()` `lme4::lmer()` `lme4::glmer()` `mclogit::mblogit()` `ordinal::clmm()` `coxme::coxme()` `lmerTest::step()` `MuMIn::dredge()` `lme4::allFit()` `ggplot2::ggplot()` `cowplot::plot_grid()` `ggcorrplot::ggcorrplot()` `ggsurvfit::ggsurvfit()`

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(lme4)
library(MuMIn)
library(ggplot2)
library(ggcorrplot)
library(cowplot)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Mixed_Effects_Regression"
rv$second_menu_choice <- "Logistic regression"
rv$entry[[1]] <- "Unable to walk independently at 6 weeks"
rv$entry[[2]] <- ""
rv$entry[[3]] <- "Centre"
rv$entry[[4]] <- "Treatment"
rv$entry[[5]] <- ""
rv$entry[[6]] <- ""
rv$entry[[7]] <- ""
rv$entry[[8]] <- "No"
# Final function ####
Results <- function.Mixed_Effects_Regression(Predefined_lists, rv)
```

`function.Multivariate_Regression` *Perform Multivariate Regression*

Description

Performs multivariate regression analysis using **lme4** for multivariate linear, logistic, and Poisson regression and **stats** for performing stepwise regression. The user interface accepts multinomial and ordinal variables, but the results for regression types other than multivariate linear and logistic regressions are unverified.

Usage

```
function.Multivariate_Regression(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 function.Regression_Analysis function.Mixed_Effects_Regression DescTools::BoxCox() stats::lm() stats::glm() nnet::multinom() MASS::polr() survival::coxph() ggplot2::ggplot() ggcorrplot::ggcorrplot() cowplot::plot_grid()

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(ggcorrplot)
library(cowplot)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Multivariate_Regression"
rv$second_menu_choice <- "Linear regression"
rv$entry[[1]] <- c("Mobility score at 6 months", "Mobility score at 12 months")
rv$entry[[2]] <- ""
rv$entry[[3]] <- "Treatment"
rv$entry[[4]] <- ""
rv$entry[[5]] <- ""
rv$entry[[6]] <- ""
rv$entry[[7]] <- "No"
# Final function ####
Results <- function.Multivariate_Regression(Predefined_lists, rv)
```

`function.plan_upload` *Upload a Plan to Rerun the Analysis*

Description

Once an analysis has been performed, a plan is automatically generated by 'EQUAL-STATS'. This plan can be used to rerun the analysis allowing transparency and reproducibility of analysis. For this function to run successfully, additional information is provided directly by 'EQUAL-STATS' software. For analysis without requiring data upload, function.plan_upload_no_data is used

Usage

```
function.plan_upload(plan_file_path, Predefined_lists, rv)
```

Arguments

`plan_file_path`	The path to the plan file.
`Predefined_lists`	A list supplied by EQUAL-STATS application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input.

Value

Depending upon whether the plan aligned to the data uploaded, either the results of the analysis or message for reason for failure is provided.

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_data() function.plan_upload_no_data()

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)
)
# Create a plan file
# Several additional functions are necessary to execute the plan.
# Therefore, the plan contains wrong field names which are not present in the data
plan <- cbind.data.frame(
"AN0001", "Check_Distribution", "", "Mobility score at 60 months", "", "",
"", "", "", "", "", "", "", "", "", "", "", "", "")
colnames(plan) <- c(
"analysis_number", "first_menu_choice", "second_menu_choice", "entry_1", "entry_2",
"entry_3", "entry_4", "entry_5", "entry_6", "entry_7", "entry_8", "entry_9", "entry_10",
"entry_11", "entry_12", "entry_13", "entry_14", "entry_15", "same_row_different_row")
# 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 and plan in a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
plan_file_path = paste0(tempdir(), "/plan.csv")
write.csv(plan, file = plan_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions
library(stringr)
# Read the data
rv$import_data <- function.read_data(data_file_path)
# Final function ####
```

```
plan_outcome <- function.plan_upload(plan_file_path, Predefined_lists, rv)
```

---

function.plan_upload_no_data

Upload a Plan to Rerun the Analysis without Data Upload

Description

Once an analysis has been performed, a plan is automatically generated by 'EQUAL-STATS'. This plan can be used to rerun the analysis allowing transparency and reproducibility of analysis. For this function to run successfully, additional information is provided directly by 'EQUAL-STATS' software. For analysis requiring data upload, `function.plan_upload()` is used

Usage

```
function.plan_upload_no_data(plan_file_path, Predefined_lists, rv, no_data_choices)
```

Arguments

`plan_file_path`	The path to the plan file.
`Predefined_lists`	A list supplied by EQUAL-STATS application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input.
`no_data_choices`	A list of functions that do not require data upload

Value

Depending upon whether the plan aligned to the data uploaded, either the results of the analysis or message for reason for failure is provided.

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.plan_upload()`

Examples

```
# This requires several additional functions to complete successfully.
# Therefore, column names was altered to demonstrate the unsuccessful results.
# Create plan
plan <- cbind.data.frame(
"AN0001", "Generate_Hypothesis", "", "Intensive rehabilitation", "Standard rehabilitation",
"Intervention is better or worse than comparator", "Mobility score",
"Higher values of the outcome (or more events) are better for the subject", "10",
"", "", "", "", "", "", "", "", "", "")
colnames(plan) <- c(
"analysis_number", "menu_choice", "second_menu_choice", "entry_1", "entry_2", "entry_3",
"entry_4", "entry_5", "entry_6", "entry_7", "entry_8", "entry_9", "entry_10",
"entry_11", "entry_12", "entry_13", "entry_14", "entry_15", "same_row_different_row")
# 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()
)
no_data_choices <- c("Generate_Hypothesis", "Sample_Size_Calculations_Effect_size",
"Make_Conclusions_Effect_size", "Diagnostic_Accuracy_Tables")
# Store the plan
plan_file_path = paste0(tempdir(), "/plan.csv")
write.csv(plan, file = plan_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions
library(stringr)
# Final function ####
plan_outcome <- function.plan_upload_no_data(plan_file_path, Predefined_lists, rv, no_data_choices)
```

`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 unusual 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.read_metadata

Change the Data Type and Reference Category

Description

When an user uploads a file in 'EQUAL-STATS' program, the program classify the variable types based on the nature of the data uploaded using the function function.read_data. However, the levels in the categorical data are determined by alphabetical order, which may not be appropriate in many situations, particularly when needs to compare a new treatment to the standard treatment (reference category) and when the event and no event have to be defined correctly. This can be addressed by uploading metadata.

Usage

```
function.read_metadata(rv, metadata_file_path)
```

Arguments

`rv`	A list supplied by 'EQUAL-STATS' application based on user input.
`metadata_file_path`	The path to the metadata file.

Value

`outcome`	Whether the data types and the order of the categories (levels/factors) were modified successfully.
`message`	The message displayed to the user after the processing. This message also contains the reason for failure if the modification was unsuccessful.
`data`	Imported data (with the structure modified as per the metadata)
`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 values (factors/levels) in the data
`ordinal`	Ordered categorical variables
`date`	Any variables that were declared as date and could be convered to date
`date`	Any variables that were declared as time and could be convered to 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_data()`

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)
)
# Create a meta-data file
# The first row indicates data type and the second row indicates
# the reference category (for all types of categorical variables)
metadata_first_two_rows <- cbind.data.frame(
c("nominal","S0001"), c("nominal", "C_0001"), c("binary", "Standard rehabilitation"),
c("binary", "Non-obese"), c("binary","able"), c("quantitative", NA), c("ordinal", "1_mild"),
c("counts", NA), c("quantitative", NA))
colnames(metadata_first_two_rows) <- colnames(data)
# The subsequent rows indicate the levels in the user-defined order
# for all categorical variables
# For this simulation, we will change the reference category
# but retain the remaining default order of variable levels
# First find the maximum number of levels
maximum_number_of_levels <- max(as.numeric(sapply(1:ncol(data), function(y) {
  if ((metadata_first_two_rows[1,y] == "quantitative") |
  (metadata_first_two_rows[1,y] == "counts")) {
    NA
  } else {
    nlevels(factor(data[,y]))
  }
})), na.rm = TRUE)
metadata_subsequent_rows <- lapply(1:ncol(data), function(y) {
  if ((metadata_first_two_rows[1,y] == "quantitative") |
  (metadata_first_two_rows[1,y] == "counts")) {
    output <- rep(NA,(maximum_number_of_levels-1))
  } else {
    categories <- sort(unique(data[,y]))
    categories <- categories[categories != metadata_first_two_rows[2,y]]
    output <- c(categories, rep(NA,(maximum_number_of_levels-1-length(categories))))
  }
}
```

```
)
metadata_subsequent_rows <- do.call(cbind.data.frame, metadata_subsequent_rows)
colnames(metadata_subsequent_rows) <- colnames(data)
metadata <- rbind.data.frame(metadata_first_two_rows, metadata_subsequent_rows)
# Simulate lists provided by EQUAL-STATS ####
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = "Summary_Measures",
  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 and metadata in a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
metadata_file_path = paste0(tempdir(), "/metadata.csv")
write.csv(metadata, file = metadata_file_path, row.names = FALSE, na = "")
# Load the necessary packages and functions
library(stringr)
# Read the data
rv$import_data <- function.read_data(data_file_path)
# Final function ####
meta_data_implemented_data_types <- function.read_metadata(rv, metadata_file_path)
```

`function.Regression_Analysis`
Perform Regression Analysis

Description

Performs regression analysis without mixed-effects for a single reponse using **stats** for linear, logistic, and Poisson regression, **nnet** for mutinomial logistic regression, **MASS** for ordinal regression, and **survival** for Cox regression for binary outcomes. It uses **stats** for performing stepwise regression.

Usage

```
function.Regression_Analysis(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() function.Mixed_Effects_Regression() function.Multivariate_Regression() `DescTools::BoxCox()` `stats::lm()` `stats::glm()` `nnet::multinom()` `MASS::polr()` `survival::coxph()` `stats::step()` `ggplot2::ggplot()` `ggcorrplot::ggcorrplot()` `cowplot::plot_grid()`

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 a folder
data_file_path = paste0(tempdir(), "/data.csv")
write.csv(data, file = data_file_path, row.names = FALSE, na = "")
rv$import_data <- function.read_data(data_file_path)
# Load the necessary packages and functions ####
library(stringr)
library(DescTools)
library(ggplot2)
library(ggcorrplot)
library(cowplot)
# Update choices ####
rv$first_menu_choice <- "Regression_Analysis"
rv$second_menu_choice <- "EQUAL-STATS choice"
rv$entry[[1]] <- "Unable to walk independently at 6 weeks"
rv$entry[[2]] <- ""
rv$entry[[3]] <- "Treatment"
rv$entry[[4]] <- "Obesity status"
rv$entry[[5]] <- ""
rv$entry[[6]] <- ""
rv$entry[[7]] <- "Yes"
# Final function ####
Results <- function.Regression_Analysis(Predefined_lists, rv)
```

`function.Sample_Size_Calculations_Effect_size` *Perform Sample Size Calculations using Effect Size Approach*

Description

Performs the power and sample size calculations using **pwr** to calculate the power and sample size for binary and continuous outcomes. It uses **ggplot2** to create plots. This functions takes summary information as input. If you want to calculate the sample size based on primary data, use function.Sample_Size_Calculations_Primary.

Usage

```
function.Sample_Size_Calculations_Effect_size(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 function.Sample_Size_Calculations_Primary `pwr::pwr.t.test()` `pwr::pwr.anova.test()` `pwr::pwr.2p.test()` `ggplot2::ggplot()`

Examples

```
# 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()
)
# Load the necessary packages and functions ####
library(stringr)
library(pwr)
library(ggplot2)
rv$first_menu_choice <- "Sample_Size_Calculations_Effect_size"
rv$second_menu_choice <- "Intervention study (binary outcome)"
rv$entry[[1]] <- 0.5
rv$entry[[2]] <- 0.4
rv$entry[[3]] <- "Intervention is better or worse than comparator"
rv$entry[[4]] <- "Independent samples"
# Final function ####
Results <-  function.Sample_Size_Calculations_Effect_size(Predefined_lists, rv)
```

function.Sample_Size_Calculations_Primary

Perform Sample Size Calculations from Primary Data

Description

Performs the power and sample size calculations using **pwr** to calculate the power and sample size for binary and continuous outcomes. It uses **ggplot2** to create plots. This functions takes primary data as input. If you want to calculate the sample size based on summary information, use function.Sample_Size_Calculations_Effect_size.

Usage

```
function.Sample_Size_Calculations_Primary(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 function.Sample_Size_Calculations_Effect_size `pwr::pwr.t.test()` `pwr::pwr.anova.test()` `pwr::pwr.2p.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(pwr)
library(ggplot2)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Sample_Size_Calculations_Primary"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Mobility score at 6 months"
rv$entry[[2]] <- "Treatment"
rv$entry[[3]] <- 10
# Final function ####
Results <- function.Sample_Size_Calculations_Primary(Predefined_lists, rv)
```

`function.submit_choices` *Wrapper Function That Performs the Analysis and Generates the Standalone Codes*

Description

This function is a wrapper function that chooses the correct analysis function to perform, saves the results in a zip folder, and generates the code for the standalone analysis.

Usage

```
function.submit_choices(Predefined_lists, rv, code_prefix, no_data_choices)
```

Arguments

`Predefined_lists`	A list supplied by 'EQUAL-STATS' application
`rv`	A list supplied by 'EQUAL-STATS' application based on user input
`code_prefix`	Code that forms the start of the standalone code
`no_data_choices`	A list of functions that do not require data upload

Value

`Analysis_results`	Analysis results includes the results of the last called analysis, a plan which includes all previous analysis performed in the session with the data set, and a standard alone R code which allows reproducibility of results.

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.Diagnostic_Accuracy_Tables

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
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
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'
  )
)
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = NA,
  second_menu_choice = NA,
  entry = entry,
  import_data = NULL,
  same_row_different_row = NA
)
no_data_choices <- c("Generate_Hypothesis", "Sample_Size_Calculations_Effect_size",
"Make_Conclusions_Effect_size", "Diagnostic_Accuracy_Tables")
code_prefix <- ""
# 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(zip)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Summary_Measures"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Obesity status"
rv$entry[[2]] <- "Treatment"
rv$entry[[3]] <- "EQUAL-STATS choice"
Results <- function.submit_choices(Predefined_lists, rv, code_prefix, no_data_choices)
```

`function.Summary_Measures` *Calculate Summary Measures*

Description

Calculates the summary (descriptive) measures of data, such as proportions and the confidence intervals for categorical data and mean, standard deviation, confidence intervals, median, quartiles, skewness and kurtosis. It uses **base** and **DescTools** to calculate these measures.

Usage

```
function.Summary_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 `DescTools::MeanCI()` `DescTools::MedianCI()` `DescTools::Kurt()` `DescTools::Skew()` `DescTools::BinomCI()` `DescTools::MultinomCI()`

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)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Summary_Measures"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Obesity status"
rv$entry[[2]] <- "Treatment"
rv$entry[[3]] <- "EQUAL-STATS choice"
# Final function ####
Results <-  function.Summary_Measures(Predefined_lists, rv)
```

`function.Survival_Analysis` *Perform Survival Analysis*

Description

Performs analysis of survival data (time-to-event data). It uses the **survival** to calculate the survival tables and **survfit** to generate the Kaplan-Meier plot.

Usage

```
function.Survival_Analysis(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 `survival::coxph()` `ggsurvfit::ggsurvfit()`

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(survival)
library(ggsurvfit)
rv$import_data <- function.read_data(data_file_path)
# Update choices ####
rv$first_menu_choice <- "Survival_Analysis"
rv$second_menu_choice <- NA
rv$entry[[1]] <- "Admission to care home"
rv$entry[[2]] <- "Follow-up"
rv$entry[[3]] <- "Treatment"
# Final function ####
Results <- function.Survival_Analysis(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)
```

`text_to_evaluate_first_menu_selection` *Converts the List Provided to Create User Interface*

Description

Obtains the processed list related to a particular selection and converts them to questions, if there are no second level selections so that the shiny user interface is created. If there are second level selections, it creates the text for second menu.

Usage

```
text_to_evaluate_first_menu_selection(Predefined_lists, rv)
```

Arguments

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

Value

`output`	Either a list of texts to create the shiny interface for questions when there are no second menu choices or character text to create the second menu.

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

Examples

```
# Simulate lists provided by EQUAL-STATS ####
entry <- list()
entry <- lapply(1:15, function(x) entry[[x]] <- '')
Predefined_lists <- list(
main_menu = c(
'Calculate summary measures',
'Create plots'
),
menu_short = c(
'Summary_Measures',
'Create_Plots'
),
second_menu_choices = c(
'',
'EQUAL-STATS choice%__%Histogram'
),
label_1 = c(
'Select the variable for which summary measures are required',
'Select the variable%__%Select the variable'
),
label_2 = c(
'Select the variable for which you want separate summary (optional)',
'NULL%__%NULL'
),
label_3 = c(
'Select the summary measures that you want in the report',
'Enter the title for the plot%__%Enter the title for the plot'
),
label_4 = c(
'',
'Select the variable%__%Select the variable'
),
label_5 = c(
'',
''
),
label_6 = c(
  '',
  ''
),
label_7 = c(
  '',
  ''
),
label_8 = c(
  '',
  ''
),
label_9 = c(
'',
''
),
label_10 = c(
'',
''
),
label_11 = c(
'',
''
),
label_12 = c(
'',
''
),
label_13 = c(
'',
''
),
label_14 = c(
'',
''
),
label_15 = c(
'',
''
),
entry_1 = c(
'%_%selectInput%_%rv$import_data$any_type',
'%_%selectInput%_%rv$import_data$any_type'
),
entry_2 = c(
'%_%selectInput%_%c("",setdiff(rv$import_data$categorical, rv$entry[[1]]))',
'NULL%__%NULL'
),
entry_3 = c(
'%_%checkbox%_%rv$summary_measures_choices',
'%_%text%_%"Plot title"%__%%_%text%_%"Plot title"'
),
entry_4 = c(
'',
'%_%selectInput%_%rv$entry[[1]]%__%%_%selectInput%_%rv$entry[[1]]'
),
entry_5 = c(
'',
''
),
entry_6 = c(
'',
''
),
entry_7 = c(
'',
''
),
entry_8 = c(
'',
''
),
entry_9 = c(
'',
''
),
entry_10 = c(
'',
''
),
entry_11 = c(
'',
''
),
entry_12 = c(
'',
''
),
entry_13 = c(
'',
''
),
entry_14 = c(
'',
''
),
entry_15 = c(
'',
''
),
mandatory_1 = c(
'yes',
'yes%__%yes'
),
mandatory_2 = c(
'no',
'NULL%__%NULL'
),
mandatory_3 = c(
'yes',
'no%__%no'
),
mandatory_4 = c(
'',
'no%__%no'
),
mandatory_5 = c(
'',
''
),
mandatory_6 = c(
'',
''
),
mandatory_7 = c(
'',
''
),
mandatory_8 = c(
'',
''
```

```
),
mandatory_9 = c(
'',
''
),
mandatory_10 = c(
'',
''
),
mandatory_11 = c(
'',
''
),
mandatory_12 = c(
'',
''
),
mandatory_13 = c(
'',
''
),
mandatory_14 = c(
'',
''
),
mandatory_15 = c(
'',
''
),
numeric_exemptions = c(
'',
''
)
)
rv <- list(
  StorageFolder = tempdir(),
  first_menu_choice = "Create_Plots",
  second_menu_choice = "EQUAL-STATS choice",
  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()
)
# Functions and packages required to run
library(stringr)
```

```
# Final function ####
# When there is no second menu
rv$first_menu_choice <- "Summary_Measures"
rv$second_menu_choice <- NA
first_menu_questions <- text_to_evaluate_first_menu_selection(
Predefined_lists = Predefined_lists, rv = rv)
# When there is a second menu
rv$first_menu_choice <- "Create_Plots"
rv$second_menu_choice <- "EQUAL-STATS choice"
second_menu_text <- text_to_evaluate_first_menu_selection(
Predefined_lists = Predefined_lists, rv = rv)
```

Index