

Package ‘jsmodule’

August 20, 2025

Title 'RStudio' Addins and 'Shiny' Modules for Medical Research

Version 1.6.13

Date 2025-08-07

Description

'RStudio' addins and 'Shiny' modules for descriptive statistics, regression and survival analysis.

Depends R (>= 3.4.0)

License Apache License 2.0

Encoding UTF-8

RoxygenNote 7.3.2

Imports data.table, DT, epiDisplay, flextable, forestploter, geepack, GGally, ggplot2, bslib, ggpubr, haven, Hmisc, jskm(>= 0.4.4), jstable(>= 1.3.8), labelled, MatchIt(>= 3.0.0), maxstat, methods, officer, pROC, purrr, readr, readxl, rstudioapi, rvg, scales, see, shiny, shinycustomloader, shinyjs, shinyWidgets, stats, survey, survIDINRI, survival, timeROC, utils, ggrepel, htmltools, riskRegression, R6

URL <https://jinseob2kim.github.io/jsmodule/>,
<https://github.com/jinseob2kim/jsmodule>

BugReports <https://github.com/jinseob2kim/jsmodule/issues>

Suggests testthat, shinytest, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jinseob Kim [aut, cre] (ORCID: <<https://orcid.org/0000-0002-9403-605X>>),
Zarathu [cph, fnd],
Hyunki Lee [aut],
Changwoo Lim [aut],
Jinhwan Kim [aut] (ORCID: <<https://orcid.org/0009-0009-3217-2417>>),
Yoonkyoung Jeon [aut],
Jaewoong Heo [aut],
Youngsun Park [aut] (ORCID: <<https://orcid.org/0009-0009-9336-2281>>),
Hyungwoo Jo [aut],

Jeongmin Seo [aut],
 Hojun LEE [aut],
 Sungho Choi [aut],
 Yeji Kang [aut],
 Mingu Jee [aut]

Maintainer Jinseob Kim <jinseob2kim@gmail.com>

Repository CRAN

Date/Publication 2025-08-20 03:10:02 UTC

Contents

barServer	3
barUI	5
boxServer	6
boxUI	7
coxModule	8
coxUI	10
csvFile	11
csvFileInput	11
FilePs	12
FilePsInput	13
FileRepeated	14
FileRepeatedInput	14
FileSurvey	15
FileSurveyInput	16
forestcoxServer	17
forestcoxUI	19
forestglmServer	21
forestglmUI	22
GEEModuleLinear	24
GEEModuleLogistic	26
GEEModuleUI	27
ggpairsModule	29
ggpairsModule2	30
ggpairsModuleUI1	32
ggpairsModuleUI2	33
ggplotdownUI	34
histogramServer	35
histogramUI	36
jsBasicAddin	37
jsBasicExtAddin	38
jsBasicGadget	39
jsPropensityAddin	40
jsPropensityExtAddin	40
jsPropensityGadget	41
jsRepeatedAddin	42
jsRepeatedExtAddin	42

jsRepeatedGadget	43
jsSurveyAddin	44
jsSurveyExtAddin	44
jsSurveyGadget	45
kaplanModule	46
kaplanUI	48
lineServer	49
lineUI	50
logistic.display2	51
logisticModule2	52
mk.lev2	54
mklist	54
mksetdiff	55
optionUI	55
reclassificationJS	56
regress.display2	58
regressModule2	59
regressModuleUI	60
rocModule	61
rocModule2	63
rocUI	65
ROC_table	67
scatterServer	68
scatterUI	69
survIDINRI_helper	70
tb1module	71
tb1module2	73
tb1moduleUI	74
tb1simple	75
tb1simple2	79
tb1simpleUI	82
templateGenerator	85
timeROChelper	85
timerocModule	86
timerocUI	90
timeROC_table	91
use_jsmodule_style	92

Index	94
--------------	-----------

barServer	<i>barServer: shiny module server for barplot.</i>
-----------	--

Description

Shiny module server for barplot.

Usage

```
barServer(id, data, data_label, data_varStruct = NULL, nfactor.limit = 10)
```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10

Details

Shiny module server for barplot.

Value

Shiny module server for barplot.

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      barUI("bar")
    ),
    mainPanel(
      optionUI("bar"),
      plotOutput("bar_plot"),
      ggplotdownUI("bar")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_bar <- barServer("bar",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$bar_plot <- renderPlot({
    print(out_bar())
  })
}
```

barUI*barUI: shiny module UI for barplot*

Description

Shiny module UI for barplot

Usage

```
barUI(id, label = "barplot")
```

Arguments

id	id
label	label

Details

Shiny module UI for barplot

Value

Shiny module UI for barplot

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      barUI("bar")
    ),
    mainPanel(
      optionUI("bar"),
      plotOutput("bar_plot"),
      ggplotdownUI("bar")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_bar <- barServer("bar",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )
}
```

```

    )

    output$bar_plot <- renderPlot({
      print(out_bar())
    })
  }

```

boxServer

boxServer: shiny module server for boxplot.

Description

Shiny module server for boxplot.

Usage

```
boxServer(id, data, data_label, data_varStruct = NULL, nfactor.limit = 10)
```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10

Details

Shiny module server for boxplot.

Value

Shiny module server for boxplot.

Examples

```

library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      boxUI("box")
    ),
    mainPanel(
      optionUI("box"),
      plotOutput("box_plot"),
      ggplotdownUI("box")
    )
  )

```

```

    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_box <- boxServer("box",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$box_plot <- renderPlot({
    print(out_box())
  })
}

```

boxUI

*boxUI: shiny module UI for boxplot***Description**

Shiny module UI for boxplot

Usage

```
boxUI(id, label = "boxplot")
```

Arguments

id	id
label	label

Details

Shiny module UI for boxplot

Value

Shiny module UI for boxplot

Examples

```

library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(

```

```

    sidebarPanel(
      boxUI("box")
    ),
    mainPanel(
      optionUI("box"),
      plotOutput("box_plot"),
      ggplotdownUI("box")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_box <- boxServer("box",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$box_plot <- renderPlot({
    print(out_box())
  })
}

```

coxModule

coxModule: shiny modulde server for Cox's model.

Description

Shiny modulde server for Cox's model.

Usage

```

coxModule(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  default.unires = T,
  limit.unires = 20,
  id.cluster = NULL,
  ties.coxph = "efron",
  vec.event = NULL,
  vec.time = NULL
)

```

Arguments

input	input
output	output
session	session
data	reactive data
data_label	reactive data label
data_varStruct	reactive list of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	reactive survey data. default: NULL
default.unires	Set default independent variables using univariate analysis.
limit.unires	Change to default.unires = F if number of independent variables > limit.unires, Default: 20
id.cluster	reactive cluster variable if marginal cox model, Default: NULL
ties.coxph	'coxph' ties option, one of 'efron', 'breslow', 'exact', default: 'erfon'
vec.event	event variables as vector for survival analysis, Default: NULL
vec.time	time variables as vector for survival analysis, Default: NULL

Details

Shiny module server for Cox's model.

Value

Shiny module server for Cox's model.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      coxUI("cox")
    ),
    mainPanel(
      DTOutput("coxtable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))
```

```
out_cox <- callModule(coxModule, "cox",
  data = data, data_label = data.label,
  data_varStruct = NULL
)

output$coxtable <- renderDT({
  datatable(out_cox()$table, rownames = T, caption = out_cox()$caption)
})
}
```

coxUI

coxUI: shiny module UI for Cox's model.

Description

Shiny module UI for Cox's model.

Usage

```
coxUI(id)
```

Arguments

id id

Details

Shiny module UI for Cox's model.

Value

coxUI

Examples

```
coxUI(1)
```

csvFile	<i>csvFile: Shiny module Server for file upload.</i>
---------	--

Description

The server-side logic for the ‘csvFileInput’ module. It uses the ‘DataManager’ R6 class to handle all data processing.

Usage

```
csvFile(input, output, session, nfactor.limit = 20)
```

Arguments

input, output, session	Standard Shiny server parameters.
nfactor.limit	An integer, the threshold for unique values to suggest a numeric variable as categorical, Default: 20

Value

A reactive expression that returns a list with two elements: ‘data’ (the processed data.table) and ‘label’ (a data.table with variable label information).

csvFileInput	<i>csvFileInput: Shiny module UI for file upload.</i>
--------------	---

Description

Shiny module UI for file upload supporting csv, xlsx, sav, sas7bdat, and dta formats. It provides UI outputs for various data manipulation options.

Usage

```
csvFileInput(id, label = "Upload data (csv/xlsx/sav/sas7bdat/dta)")
```

Arguments

id	A string, the module id.
label	A string, the label for the file input, Default: ‘Upload data (csv/xlsx/sav/sas7bdat/dta)’

Details

This function only defines the UI. The corresponding server function, ‘csvFile’, handles the logic.

Value

A Shiny UI object.

Examples

```
if (interactive()) {
  library(shiny)
  library(DT)
  library(jstable)

  ui <- fluidPage(
    sidebarLayout(
      sidebarPanel(
        csvFileInput("datafile")
      ),
      mainPanel(
        tabsetPanel(
          type = "pills",
          tabPanel("Data", DTOutput("data")),
          tabPanel("Label", DTOutput("data_label"))
        )
      )
    )
  )

  server <- function(input, output, session) {
    data_info <- callModule(csvFile, "datafile")

    output$data <- renderDT({
      data_info()$data
    })

    output$label <- renderDT({
      data_info()$label
    })
  }
  shinyApp(ui, server)
}
```

FilePs

FilePs: Shiny module Server for propensity score analysis.

Description

Server-side logic for propensity score analysis. It uses ‘DataManager’ for common data tasks and adds specific controls and calculations for propensity score matching.

Usage

```
FilePs(input, output, session, nfactor.limit = 20)
```



```

    )
  )
)

server <- function(input, output, session) {
  mat_info <- callModule(FilePs, "datafile")

  output$data <- renderDT({ mat_info()$data })
  output$matdata <- renderDT({ mat_info()$matdata })
  output$data_label <- renderDT({ mat_info()$label })
}
shinyApp(ui, server)
}

```

FileRepeated

FileRepeated: Server for repeated measures analysis.

Description

Server module for repeated measures analysis. It uses ‘DataManager’ and adds a control for selecting the repeated measures variable.

Usage

```
FileRepeated(input, output, session, nfactor.limit = 20)
```

Arguments

input, output, session
Standard Shiny server parameters.

nfactor.limit An integer, the threshold for unique values.

Value

A reactive list with the processed ‘data’, ‘label’, and ‘id.gee’.

FileRepeatedInput

FileRepeatedInput: UI for repeated measures analysis.

Description

File upload UI for repeated measure analysis.

Usage

```
FileRepeatedInput(id, label = "Upload data (csv/xlsx/sav/sas7bdat/dta)")
```

Arguments

`id` A string, the module id.

`label` A string, the label for the file input.

Value

A Shiny UI object.

Examples

```
if (interactive()) {
  library(shiny)
  library(DT)
  library(jstable)

  ui <- fluidPage(
    sidebarLayout(
      sidebarPanel(FileRepeatedInput("datafile")),
      mainPanel(
        tabsetPanel(
          type = "pills",
          tabPanel("Data", DTOutput("data")),
          tabPanel("Label", DTOutput("data_label"))
        )
      )
    )
  )

  server <- function(input, output, session) {
    data_info <- callModule(FileRepeated, "datafile")
    output$data <- renderDT({
      data_info()$data
    })
    output$label <- renderDT({
      data_info()$label
    })
  }
  shinyApp(ui, server)
}
```

Description

Server module for survey data analysis. It uses ‘DataManager’ and adds controls and logic for creating a ‘survey.design’ object.

Usage

```
FileSurvey(input, output, session, nfactor.limit = 20)
```

Arguments

input, output, session
Standard Shiny server parameters.

nfactor.limit An integer, the threshold for unique values.

Value

A reactive list with 'data', 'label', 'naomit', and the 'survey' object.

FileSurveyInput	<i>FileSurveyInput: UI for survey data analysis.</i>
-----------------	--

Description

File upload UI for survey data analysis, with controls for survey design elements.

Usage

```
FileSurveyInput(id, label = "Upload data (csv/xlsx/sav/sas7bdat/dta)")
```

Arguments

id A string, the module id.

label A string, the label for the file input.

Value

A Shiny UI object.

Examples

```
if (interactive()) {
  library(shiny)
  library(DT)
  library(jstable)
  library(survey)

  ui <- fluidPage(
    sidebarLayout(
      sidebarPanel(FileSurveyInput("datafile")),
      mainPanel(
        h4("Survey object details:"),
        verbatimTextOutput("survey_summary"),
        tabsetPanel(
```

```

        type = "pills",
        tabPanel("Data", DTOutput("data")),
        tabPanel("Label", DTOutput("data_label"))
      )
    )
  )
}

server <- function(input, output, session) {
  data_info <- callModule(FileSurvey, "datafile")
  output$data <- renderDT({
    data_info()$data
  })
  output$label <- renderDT({
    data_info()$label
  })
  output$survey_summary <- renderPrint({
    print(data_info()$survey)
  })
}
shinyApp(ui, server)
}

```

forestcoxServer

*forestcoxServer: shiny module server for forestcox***Description**

Shiny module server for forestcox

Usage

```

forestcoxServer(
  id,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  cluster_id = NULL,
  vec.event = NULL,
  vec.time = NULL
)

```

Arguments

id	id
data	Reactive data

<code>data_label</code>	Reactive data label
<code>data_varStruct</code>	Reactive List of variable structure, Default: NULL
<code>nfactor.limit</code>	nlevels limit in factor variable, Default: 10
<code>design.survey</code>	reactive survey data. default: NULL
<code>cluster_id</code>	cluster option variable for marginal cox model
<code>vec.event</code>	event variables as vector for survival analysis, Default: NULL
<code>vec.time</code>	time variables as vector for survival analysis, Default: NULL

Details

Shiny module server for forestcox

Value

Shiny module server for forestcox

See Also

[data.table-package](#), [setDT](#), [setattr](#) [TableSubgroupMultiCox](#) [forest_theme](#), [forest dml read_pptx](#), [add_slide](#), [ph_with](#), [ph_location](#)

Examples

```
library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$disp >= 150))
mtcars$kk1 <- factor(as.integer(mtcars$disp >= 200))
```

```
library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$disp >= 150))
mtcars$kk1 <- factor(as.integer(mtcars$disp >= 200))
```

```
out <- mtcars
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      forestcoxUI("Forest")
    ),
    mainPanel(
      tabsetPanel(
        type = "pills",
        tabPanel(
          title = "Data",
          DTOutput("tablesub"),
        ),
      )
    )
  )
)
```

```

    tabPanel(
      title = "figure",
      plotOutput("forestplot", width = "100%"),
      ggplotdownUI("Forest")
    )
  )
)
)

server <- function(input, output, session) {
  data <- reactive(out)
  label <- reactive(jstable::mk.lev(out))
  outtable <- forestcoxServer("Forest", data = data, data_label = label)
  output$tablesub <- renderDT({
    outtable()[[1]]
  })
  output$forestplot <- renderPlot({
    a
    outtable()[[2]]
  })
}

```

forestcoxUI

forestcoxUI:shiny module UI for forestcox

Description

Shiny module UI for forestcox

Usage

```
forestcoxUI(id, label = "forestplot")
```

Arguments

id	id
label	label, Default: 'forestplot'

Details

Shinymodule UI for forestcox

Value

Shinymodule UI

Examples

```

library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$disp >= 150))
mtcars$kk1 <- factor(as.integer(mtcars$disp >= 200))

library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$disp >= 150))
mtcars$kk1 <- factor(as.integer(mtcars$disp >= 200))

out <- mtcars
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      forestcoxUI("Forest")
    ),
    mainPanel(
      tabsetPanel(
        type = "pills",
        tabPanel(
          title = "Data",
          DTOutput("tablesub")
        ),
        tabPanel(
          title = "figure",
          plotOutput("forestplot", width = "100%"),
          ggplotdownUI("Forest")
        )
      )
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(out)
  label <- reactive(jstable::mk.lev(out))
  outtable <- forestcoxServer("Forest", data = data, data_label = label)
  output$tablesub <- renderDT({
    outtable()[[1]]
  })
  output$forestplot <- renderPlot({
    outtable()[[2]]
  })
}

```

forestglmServer	<i>forestglmServer:shiny module server for forestglm</i>
-----------------	--

Description

Shiny module server for forestglm

Usage

```
forestglmServer(  
  id,  
  data,  
  data_label,  
  family,  
  data_varStruct = NULL,  
  nfactor.limit = 10,  
  design.survey = NULL,  
  repeated_id = NULL  
)
```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
family	family, "gaussian" or "binomial" or 'poisson' or 'quasipoisson'
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	reactive survey data. default: NULL
repeated_id	data when repeated id. default: F

Details

Shiny module server for forestglm

Value

Shiny module server for forestglm

See Also

[TableSubgroupMultiGLM](#), [data.table-package](#), [setDT](#), [setattr](#), [cor](#), [coef](#), [surveysummary](#), [svytable](#), [forest_theme](#), [forest](#), [dml](#), [read_pptx](#), [add_slide](#), [ph_with](#), [ph_location](#)

Examples

```

library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$disp >= 150))
mtcars$kk1 <- factor(as.integer(mtcars$disp >= 200))

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      forestglmUI("Forest")
    ),
    mainPanel(
      tabsetPanel(
        type = "pills",
        tabPanel(
          title = "Data",
          DTOutput("tablesub"),
        ),
        tabPanel(
          title = "figure",
          plotOutput("forestplot", width = "100%"),
          ggplotdownUI("Forest")
        )
      )
    )
  )
)

out <- mtcars

server <- function(input, output, session) {
  data <- reactive(out)
  label <- reactive(jstable::mk.lev(out))
  outtable <- forestglmServer("Forest", data = data, data_label = label, family = "binomial")
  output$tablesub <- renderDT({
    outtable()[[1]]
  })
  output$forestplot <- renderPlot({
    outtable()[[2]]
  })
}

```

forestglmUI

forestglmUI: Shiny module UI for forestglm

Description

Shiny module UI for forestcox

Usage

```
forestglmUI(id, label = "forestplot")
```

Arguments

id	id
label	label, Default: 'forestplot'

Details

Shiny module UI for forestglm

Value

Shiny module UI

Examples

```
library(shiny)
library(DT)
mtcars$vs <- factor(mtcars$vs)
mtcars$am <- factor(mtcars$am)
mtcars$kk <- factor(as.integer(mtcars$displacement) >= 150)
mtcars$kk1 <- factor(as.integer(mtcars$displacement) >= 200)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      forestglmUI("Forest")
    ),
    mainPanel(
      tabsetPanel(
        type = "pills",
        tabPanel(
          title = "Data",
          DTOutput("tablesub"),
        ),
        tabPanel(
          title = "figure",
          plotOutput("forestplot", width = "100%"),
          ggplotdownUI("Forest")
        )
      )
    )
  )
)

out <- mtcars

server <- function(input, output, session) {
  data <- reactive(out)
```

```

label <- reactive(jstable::mk.lev(out))
outtable <- forestglmServer("Forest", data = data, data_label = label, family = "binomial")
output$talesub <- renderDT({
  outtable()[[1]]
})
output$forestplot <- renderPlot({
  outtable()[[2]]
})
}

```

GEEModuleLinear

GEEModuleLinear: shiny modulde server for gaussian generalized estimating equation(GEE) using reactive data.

Description

Shiny modulde server for gaussian generalized estimating equation(GEE) using reactive data.

Usage

```

GEEModuleLinear(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  id.gee,
  vec.event = NULL
)

```

Arguments

input	input
output	output
session	session
data	reactive data, ordered by id.
data_label	reactive data label
data_varStruct	List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
id.gee	reactive repeated measure variable
vec.event	event variables as vector for gaussian generalized estimating equation(GEE), Default: NULL

Details

Shiny module server for gaussian generalized estimating equation(GEE) using reactive data.

Value

Shiny module server for gaussian generalized estimating equation(GEE).

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      GEEModuleUI("linear")
    ),
    mainPanel(
      DTOutput("lineartable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))
  id.gee <- reactive("mpg")

  out_linear <- callModule(GEEModuleLinear, "linear",
    data = data, data_label = data.label,
    data_varStruct = NULL, id.gee = id.gee
  )

  output$lineartable <- renderDT({
    hide <- which(colnames(out_linear())$table) == "sig")
    datatable(out_linear())$table,
    rownames = T, extension = "Buttons", caption = out_linear()$caption,
    options = c(
      opt.tbreg(out_linear()$caption),
      list(columnDefs = list(list(visible = FALSE, targets = hide))),
      list(scrollX = TRUE)
    )
  }) %>% formatStyle("sig", target = "row", backgroundColor = styleEqual("**", "yellow"))
})
}
```

GEEModuleLogistic	<i>GEEModuleLogistic: shiny modulde server for binomial gaussian generalized estimating equation(GEE) using reactive data.</i>
-------------------	--

Description

Shiny modulde server for binomial gaussian generalized estimating equation(GEE) using reactive data.

Usage

```
GEEModuleLogistic(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  id.gee,
  vec.event = NULL
)
```

Arguments

input	input
output	output
session	session
data	reactive data, ordered by id.
data_label	reactive data label
data_varStruct	List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
id.gee	reactive repeated measure variable
vec.event	event variables as vector for binomial gaussian generalized estimating equation(GEE), Default: NULL

Details

Shiny modulde server for binomial gaussian generalized estimating equation(GEE) using reactive data.

Value

Shiny modulde server for binomial gaussian generalized estimating equation(GEE).

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      GEEModuleUI("logistic")
    ),
    mainPanel(
      DTOutput("logistictable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))
  id.gee <- reactive("mpg")

  out_logistic <- callModule(GEEModuleLogistic, "logistic",
    data = data, data_label = data.label,
    data_varStruct = NULL, id.gee = id.gee
  )

  output$logistictable <- renderDT({
    hide <- which(colnames(out_logistic())$table) == "sig")
    datatable(out_logistic())$table,
    rownames = T, extension = "Buttons",
    caption = out_logistic()$caption,
    options = c(
      opt.tbreg(out_logistic())$caption,
      list(columnDefs = list(list(visible = FALSE, targets = hide))),
      list(scrollX = TRUE)
    )
  }) %>% formatStyle("sig", target = "row", backgroundColor = styleEqual("**", "yellow"))
}

```

GEEModuleUI

GEEModuleUI: shiny modulde UI for generalized estimating equation(GEE).

Description

Shiny modulde UI for generalized estimating equation(GEE).

Usage

```
GEEModuleUI(id)
```

Arguments

id id

Details

Shiny module UI for generalized estimating equation(GEE).

Value

Shiny module UI for generalized estimating equation(GEE).

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      GEEModuleUI("linear")
    ),
    mainPanel(
      DTOutput("lineartable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))
  id.gee <- reactive("mpg")

  out_linear <- callModule(GEEModuleLinear, "linear",
    data = data, data_label = data.label,
    data_varStruct = NULL, id.gee = id.gee
  )

  output$lineartable <- renderDT({
    hide <- which(colnames(out_linear())$table) == "sig")
    datatable(out_linear())$table,
    rownames = T, extension = "Buttons", caption = out_linear()$caption,
    options = c(
      opt.tbreg(out_linear()$caption),
      list(columnDefs = list(list(visible = FALSE, targets = hide))),
      list(scrollX = TRUE)
    )
  }) %>% formatStyle("sig", target = "row", backgroundColor = styleEqual("**", "yellow"))
}
```

ggpairsModule	<i>ggpairsModule: shiny module server for basic/scatter plot.</i>
---------------	---

Description

Shiny module server for basic/scatter plot.

Usage

```
ggpairsModule(  
  input,  
  output,  
  session,  
  data,  
  data_label,  
  data_varStruct = NULL,  
  nfactor.limit = 20  
)
```

Arguments

input	input
output	output
session	session
data	data
data_label	data label
data_varStruct	List of variable structure, Default: NULL
nfactor.limit	nlevels limit for categorical variables, Default: 20

Details

Shiny module server for basic/scatter plot.

Value

Shiny module server for basic/scatter plot.

Examples

```
library(shiny)  
library(DT)  
library(data.table)  
library(jstable)  
library(ggplot2)  
library(GGally)
```

```

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      ggpairsModuleUI1("ggpairs")
    ),
    mainPanel(
      plotOutput("ggpairs_plot"),
      ggpairsModuleUI2("ggpairs")
    )
  )
)

server <- function(input, output, session) {
  data <- mtcars
  data.label <- jstable::mk.lev(mtcars)

  out_ggpairs <- callModule(ggpairsModule, "ggpairs",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_ggpairs())
  })
}

```

ggpairsModule2

ggpairsModule2: shiny module server for basic/scatter plot for reactive data.

Description

Shiny module server for basic/scatter plot for reactive data.

Usage

```

ggpairsModule2(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 20
)

```

Arguments

input input

output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	List of variable structure, Default: NULL
nfactor.limit	nlevels limit for categorical variables, Default: 20

Details

Shiny module server for basic/scatter plot for reactive data.

Value

Shiny module server for basic/scatter plot

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(GGally)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      ggpairsModuleUI1("ggpairs")
    ),
    mainPanel(
      plotOutput("ggpairs_plot"),
      ggpairsModuleUI2("ggpairs")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_ggpairs <- callModule(ggpairsModule2, "ggpairs",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_ggpairs())
  })
}
```

ggpairsModuleUI1

ggpairsModuleUI1: Variable selection module UI for ggpairs

Description

Variable selection module UI for ggpairs

Usage

```
ggpairsModuleUI1(id)
```

Arguments

```
id          id
```

Details

Variable selection module UI for ggpairs

Value

Variable selection module UI for ggpairs

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(GGally)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      ggpairsModuleUI1("ggpairs")
    ),
    mainPanel(
      plotOutput("ggpairs_plot"),
      ggpairsModuleUI2("ggpairs")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_ggpairs <- callModule(ggpairsModule2, "ggpairs",
    data = data, data_label = data.label,
```

```

    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_ggpairs())
  })
}

```

ggpairsModuleUI2

ggpairsModuleUI2: Option & download module UI for ggpairs

Description

Option & download module UI for ggpairs

Usage

```
ggpairsModuleUI2(id)
```

Arguments

id id

Details

Option & download module UI for ggpairs

Value

Option & download module UI for ggpairs

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(GGally)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      ggpairsModuleUI1("ggpairs")
    ),
    mainPanel(
      plotOutput("ggpairs_plot"),
      ggpairsModuleUI2("ggpairs")
    )
  )
)

```

```

)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_ggpairs <- callModule(ggpairsModule2, "ggpairs",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_ggpairs())
  })
}

```

ggplotdownUI

*ggplotdownUI: Option & download module UI for ggplot***Description**

Option & download module UI for ggplot

Usage

```
ggplotdownUI(id)
```

Arguments

id id

Details

Option & download module UI for ggplot

Value

Option & download module UI for ggplot

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      kaplanUI("kaplan")
    )
  )
)

```

```

    ),
    mainPanel(
      plotOutput("kaplan_plot"),
      ggplotdownUI("kaplan")
    )
  )
}

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_kaplan <- callModule(kaplanModule, "kaplan",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_kaplan())
  })
}

```

histogramServer

histogramServer: shiny module server for histogram.

Description

Shiny module server for histogram.

Usage

```

histogramServer(
  id,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10
)

```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10

Details

Shiny module server for histogram.

Value

Shiny module server for histogram.

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      histogramUI("histogram")
    ),
    mainPanel(
      plotOutput("histogram"),
      ggplotdownUI("histogram")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_histogram <- histogramServer("histogram",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$histogram <- renderPlot({
    print(out_histogram())
  })
}
```

histogramUI

histogramUI: shiny module UI for histogram

Description

Shiny module UI for histogram

Usage

```
histogramUI(id, label = "histogram")
```

Arguments

id	id
label	label

Details

Shiny module UI for histogram

Value

Shiny module UI for histogram

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      histogramUI("histogram")
    ),
    mainPanel(
      plotOutput("histogram"),
      ggplotdownUI("histogram")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_histogram <- histogramServer("histogram",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$histogram <- renderPlot({
    print(out_histogram())
  })
}
```

jsBasicAddin

jsBasicAddin: Rstudio addin of jsBasicGadget

Description

Rstudio addin of jsBasicGadget

Usage

```
jsBasicAddin()
```

Details

Rstudio addin of jsBasicGadget

Value

Rstudio addin of jsBasicGadget

See Also

[rstudio-editors](#)

Examples

```
if (interactive()) {  
  jsBasicAddin()  
}
```

jsBasicExtAddin

jsBasicExtAddin: RStudio Addin for basic data analysis with external data.

Description

RStudio Addin for basic data analysis with external csv/xlsx/sas7bdat/sav/dta file.

Usage

```
jsBasicExtAddin(nfactor.limit = 20, max.filesize = 2048)
```

Arguments

`nfactor.limit` nlevels limit for categorical variables, Default: 20
`max.filesize` Maximum file size to upload (MB), Default: 2048 (2 GB)

Details

RStudio Addin for basic data analysis with external csv/xlsx/sas7bdat/sav/dta file.

Value

RStudio Addin for basic data analysis with external data.

See Also

[lung fwrite opt.tbreg](#)

Examples

```
if (interactive()) {  
  jsBasicExtAddin()  
}
```

`jsBasicGadget`*jsBasicGadget: Shiny Gadget of Basic Statistics in Medical Research.*

Description

Shiny Gadget including Data, Label info, Table 1, Regression(linear, logistic), Basic plot

Usage

```
jsBasicGadget(data, nfactor.limit = 20)
```

Arguments

<code>data</code>	data
<code>nfactor.limit</code>	nlevels limit for categorical variables

Details

Shiny Gadget including Data, Label info, Table 1, Regression(linear, logistic), Basic plot

Value

Shiny Gadget including Data, Label info, Table 1, Regression(linear, logistic), Basic plot

Examples

```
if (interactive()) {  
  jsBasicGadget(mtcars)  
}
```

jsPropensityAddin	<i>jsPropensityAddin: Rstudio addin of jsPropensityGadget</i>
-------------------	---

Description

Rstudio addin of jsPropensityGadget

Usage

```
jsPropensityAddin()
```

Details

Rstudio addin of jsPropensityGadget

Value

Rstudio addin of jsPropensityGadget

See Also

[rstudio-editors](#)

Examples

```
if (interactive()) {
  jsPropensityAddin()
}
```

jsPropensityExtAddin	<i>jsPropensityExtAddin: RStudio Addin for propensity score analysis with external data.</i>
----------------------	--

Description

RStudio Addin for propensity score analysis with external csv/xlsx/sas7bdat/sav/dta file.

Usage

```
jsPropensityExtAddin(nfactor.limit = 20, max.filesize = 2048)
```

Arguments

nfactor.limit	nlevels limit for categorical variables, Default: 20
max.filesize	Maximum file size to upload (MB), Default: 2048 (2 GB)

Details

RStudio Addin for propensity score analysis with external csv/xlsx/sas7bdat/sav/dta file.

Value

RStudio Addin for propensity score analysis with external data.

See Also

[pbc.fwrite, data.table, svydesign, opt.tbreg](#)

Examples

```
if (interactive()) {
  jsPropensityExtAddin()
}
```

jsPropensityGadget	<i>jsPropensityGadget: Shiny Gadget for propensity score analysis.</i>
--------------------	--

Description

Shiny Gadget including original/matching/IPTW data, Label info, Table 1, Cox model, Basic/kaplan-meier plot.

Usage

```
jsPropensityGadget(data, nfactor.limit = 20)
```

Arguments

data	data
nfactor.limit	nlevels limit for categorical variables, Default: 20

Details

Shiny Gadget including original/matching/IPTW data, Label info, Table 1, Cox model, Basic/kaplan-meier plot.

Value

Shiny Gadget including original/matching/IPTW data, Label info, Table 1, Cox model, Basic/kaplan-meier plot.

See Also

[data.table, matchit, match.data, cox2.display, svycox.display, survfit, coxph, Surv, jskm, svyjskm, ggsave, svykm](#)

Examples

```
if (interactive()) {
  jsPropensityGadget(mtcars)
}
```

jsRepeatedAddin

jsRepeatedAddin: Rstudio addin of jsRepeatedGadget

Description

Rstudio addin of jsRepeatedGadget

Usage

```
jsRepeatedAddin()
```

Details

Rstudio addin of jsRepeatedGadget

Value

Rstudio addin of jsRepeatedGadget

See Also

[rstudio-editors](#)

Examples

```
if (interactive()) {
  jsRepeatedAddin()
}
```

jsRepeatedExtAddin

jsRepeatedExtAddin: RStudio Addin for repeated measure analysis with external data.

Description

RStudio Addin for repeated measure analysis with external csv/xlsx/sas7bdat/sav/dta file.

Usage

```
jsRepeatedExtAddin(nfactor.limit = 20, max.filesize = 2048)
```

Arguments

`nfactor.limit` nlevels limit for categorical variables, Default: 20
`max.filesize` Maximum file size to upload (MB), Default: 2048 (2 GB)

Details

RStudio Addin for repeated measure analysis with external csv/xlsx/sas7bdat/sav/dta file.

Value

RStudio Addin for repeated measure analysis with external data.

See Also

[fwrite colon opt.tbreg](#)

Examples

```
if (interactive()) {  
  jsRepeatedExtAddin()  
}
```

jsRepeatedGadget	<i>jsRepeatedGadget: Shiny Gadget of Repeated measure analysis.</i>
------------------	---

Description

Shiny Gadget including Data, Label info, Table 1, GEE(linear, logistic), Basic plot

Usage

```
jsRepeatedGadget(data, nfactor.limit = 20)
```

Arguments

`data` data
`nfactor.limit` nlevels limit for categorical variables

Details

Shiny Gadget including Data, Label info, Table 1, GEE(linear, logistic), Basic plot

Value

Shiny Gadget including Data, Label info, Table 1, GEE(linear, logistic), Basic plot

Examples

```
if (interactive()) {
  jsRepeatedGadget(mtcars)
}
```

jsSurveyAddin

jsSurveyAddin: Rstudio addin of jsSurveyGadget

Description

Rstudio addin of jsSurveyGadget

Usage

```
jsSurveyAddin()
```

Details

Rstudio addin of jsSurveyGadget

Value

Rstudio addin of jsSurveyGadget

See Also

[rstudio-editors](#)

Examples

```
if (interactive()) {
  jsSurveydAddin()
}
```

jsSurveyExtAddin

jsSurveyExtAddin: RStudio Addin for survey data analysis with external data.

Description

RStudio Addin for survey data analysis with external csv/xlsx/sas7bdat/sav/dta file.

Usage

```
jsSurveyExtAddin(nfactor.limit = 20, max.filesize = 2048)
```

Arguments

`nfactor.limit` nlevels limit for categorical variables, Default: 20
`max.filesize` Maximum file size to upload (MB), Default: 2048 (2 GB)

Details

RStudio Addin for survey data analysis with external csv/xlsx/sas7bdat/sav/dta file.

Value

RStudio Addin for survey data analysis with external data.

See Also

[fwrite opt.tb1,opt.tbreg](#)

Examples

```
if (interactive()) {
  jsSurveyExtAddin()
}
```

jsSurveyGadget	<i>jsSurveyGadget: Shiny Gadget of survey data analysis.</i>
----------------	--

Description

Shiny Gadget including Data, Label info, Table 1, svyglm, Basic plot

Usage

```
jsSurveyGadget(data, nfactor.limit = 20)
```

Arguments

`data` data
`nfactor.limit` nlevels limit for categorical variables

Details

Shiny Gadget including Data, Label info, Table 1, svyglm, Basic plot

Value

Shiny Gadget including Data, Label info, Table 1, svyglm, Basic plot

Examples

```
if (interactive()) {
  jsSurveyGadget(mtcars)
}
```

kaplanModule

kaplanModule: shiny module server for kaplan-meier plot.

Description

Shiny module server for kaplan-meier plot.

Usage

```
kaplanModule(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  id.cluster = NULL,
  timeby = NULL,
  range.x = NULL,
  range.y = NULL,
  vec.event = NULL,
  vec.time = NULL
)
```

Arguments

input	input
output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	Reactive survey data. default: NULL
id.cluster	Reactive cluster variable if marginal model, Default: NULL
timeby	timeby, Default: NULL
range.x	range of x axis, Default: NULL

<code>range.y</code>	range of y axis, Default: NULL
<code>vec.event</code>	event variables as vector for survival analysis, Default: NULL
<code>vec.time</code>	time variables as vector for survival analysis, Default: NULL

Details

Shiny module server for kaplan-meier plot.

Value

Shiny module server for kaplan-meier plot.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      kaplanUI("kaplan")
    ),
    mainPanel(
      plotOutput("kaplan_plot"),
      ggplotdownUI("kaplan")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_kaplan <- callModule(kaplanModule, "kaplan",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_kaplan())
  })
}
```

`kaplanUI`*kaplanUI: shiny module UI for kaplan-meier plot*

Description

Shiny module UI for kaplan-meier plot

Usage

```
kaplanUI(id)
```

Arguments

<code>id</code>	<code>id</code>
-----------------	-----------------

Details

Shiny module UI for kaplan-meier plot

Value

Shiny module UI for kaplan-meier plot

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      kaplanUI("kaplan")
    ),
    mainPanel(
      plotOutput("kaplan_plot"),
      ggplotdownUI("kaplan")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_kaplan <- callModule(kaplanModule, "kaplan",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )
}
```

```

    output$kaplan_plot <- renderPlot({
      print(out_kaplan())
    })
  }

```

lineServer

lineServer: shiny module server for lineplot.

Description

Shiny module server for lineplot.

Usage

```
lineServer(id, data, data_label, data_varStruct = NULL, nfactor.limit = 10)
```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10

Details

Shiny module server for lineplot.

Value

Shiny module server for lineplot.

Examples

```

library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      lineUI("line")
    ),
    mainPanel(
      optionUI("line"),
      plotOutput("line_plot"),
      ggplotdownUI("line")
    )
  )

```

```

    )
  )

  server <- function(input, output, session) {
    data <- reactive(mtcars)
    data.label <- reactive(jstable::mk.lev(mtcars))

    out_line <- lineServer("line",
      data = data, data_label = data.label,
      data_varStruct = NULL
    )

    output$line_plot <- renderPlot({
      print(out_line())
    })
  }
}
```

lineUI	<i>lineUI: shiny module UI for lineplot</i>
--------	---

Description

Shiny module UI for lineplot

Usage

```
lineUI(id, label = "lineplot")
```

Arguments

id	id
label	label

Details

Shiny module UI for lineplot

Value

Shiny module UI for lineplot

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
```

```

        lineUI("line")
      ),
      mainPanel(
        optionUI("line"),
        plotOutput("line_plot"),
        ggplotdownUI("line")
      )
    )
  )
}

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_line <- lineServer("line",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$line_plot <- renderPlot({
    print(out_line())
  })
}

```

logistic.display2

logistic.display2: Modified epiDisplay's logistic.display function.

Description

Modified epiDisplay's logistic.display function for reactive data.

Usage

```

logistic.display2(
  logistic.model,
  alpha = 0.05,
  crude = TRUE,
  crude.p.value = FALSE,
  decimal = 2,
  simplified = FALSE
)

```

Arguments

logistic.model	glm object(binomial)
alpha	alpha, Default: 0.05
crude	crude, Default: TRUE
crude.p.value	crude.p.value, Default: FALSE

decimal	decimal, Default: 2
simplified	simplified, Default: FALSE

Details

Modified epiDisplay's logistic.display function for reactive data.

Value

logistic table

Examples

```
model1 <- glm(am ~ cyl + disp, data = mtcars, family = binomial)
logistic.display2(model1, crude = TRUE, crude.p.value = TRUE, decimal = 3)
```

logisticModule2	<i>logisticModule2: Shiny module server for logistic regression for reactive data.</i>
-----------------	--

Description

Shiny module server for logistic regression for reactive data.

Usage

```
logisticModule2(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  default.unires = T,
  limit.unires = 20,
  vec.event = NULL
)
```

Arguments

input	input
output	output
session	session
data	reactive data
data_label	reactive data label

<code>data_varStruct</code>	List of variable structure, Default: NULL
<code>nfactor.limit</code>	nlevels limit in factor variable, Default: 10
<code>design.survey</code>	reactive survey data. default: NULL
<code>default.unires</code>	Set default independent variables using univariate analysis, Default: T
<code>limit.unires</code>	Change to <code>default.unires = F</code> if number of independent variables > <code>limit.unires</code> , Default: 20
<code>vec.event</code>	event variables as vector for logistic regression, Default: NULL

Details

Shiny modulde server for logistic regression.

Value

Shiny modulde server for logistic regression.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      regressModuleUI("logistic")
    ),
    mainPanel(
      DTOutput("logistictable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_logistic <- callModule(logisticModule2, "logistic",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$logistictable <- renderDT({
    datatable(out_logistic()$table, rownames = T, caption = out_logistic()$caption)
  })
}
```

mk.lev2	<i>mk.lev2: level generating function</i>
---------	---

Description

make level for sav files with labels pre defined from SPSS

Usage

```
mk.lev2(out.old, out.label)
```

Arguments

out.old	raw data
out.label	pre-defined label data

Value

out.label data labels updated

mklist	<i>mklist: function to make variable list Including specific variables.</i>
--------	---

Description

Function to make variable list Including specific variables.

Usage

```
mklist(varlist, vars)
```

Arguments

varlist	Original variable list.
vars	variable to include.

Details

Internal function

Value

variable list Including specific variables.

Examples

```
data_varStruct <- list(variable = names(mtcars))
mklist(data_varStruct, names(mtcars))
```

mksetdiff	<i>mksetdiff: function to make variable list excluding specific variables.</i>
-----------	--

Description

Function to make variable list excluding specific variables.

Usage

```
mksetdiff(varlist, vars)
```

Arguments

varlist	Original variable list
vars	variable to exclude.

Details

Internal function

Value

variable list excluding specific variables.

Examples

```
data_varStruct <- list(variable = names(mtcars))
mksetdiff(data_varStruct, "mpg")
```

optionUI	<i>optionUI: Option UI with icon</i>
----------	--------------------------------------

Description

Option UI with icon

Usage

```
optionUI(id)
```

Arguments

id	id
----	----

Details

Option UI with icon

Value

Option UI with icon

See Also

[dropdownButton](#), [tooltipOptions](#)

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      kaplanUI("kaplan")
    ),
    mainPanel(
      optionUI("kaplan"),
      plotOutput("kaplan_plot"),
      ggplotdownUI("kaplan")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_kaplan <- callModule(kaplanModule, "kaplan",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$kaplan_plot <- renderPlot({
    print(out_kaplan())
  })
}
```

reclassificationJS	<i>reclassificationJS: Function for reclassification table and statistics</i>
--------------------	---

Description

Modified function of PredictABEL::reclassification: return output table

Usage

```
reclassificationJS(
  data,
  cOutcome,
  predrisk1,
  predrisk2,
  cutoff,
  dec.value = 3,
  dec.p = 3
)
```

Arguments

data	Data frame or matrix that includes the outcome and predictors variables.
cOutcome	Column number of the outcome variable.
predrisk1	Vector of predicted risks of all individuals using initial model.
predrisk2	Vector of predicted risks of all individuals using updated model.
cutoff	Cutoff values for risk categories. Define the cut-off values. Ex: c(0,.20,.30,1)
dec.value	digits of value, Default: 4
dec.p	digits of p, Default: 3

Details

Modified function of PredictABEL::reclassification

Value

Table including NRI(categorical), NRI(continuous), IDI with 95

See Also

[rcorrp.cens](#)

Examples

```
m1 <- glm(vs ~ am + gear, data = mtcars, family = binomial)
m2 <- glm(vs ~ am + gear + wt, data = mtcars, family = binomial)
reclassificationJS(
  data = mtcars, cOutcome = 8,
  predrisk1 = predict(m1, type = "response"),
  predrisk2 = predict(m2, type = "response"), cutoff = c(0, .20, .40, 1)
)
```

regress.display2	<i>regress.display2: modified epiDisplay's regress.display function</i>
------------------	---

Description

regress.display function for reactive data

Usage

```
regress.display2(  
  regress.model,  
  alpha = 0.05,  
  crude = FALSE,  
  crude.p.value = FALSE,  
  decimal = 2,  
  simplified = FALSE  
)
```

Arguments

regress.model	lm object
alpha	alpha, Default: 0.05
crude	crude, Default: FALSE
crude.p.value	crude.p.value, Default: FALSE
decimal	decimal, Default: 2
simplified	simplified, Default: FALSE

Details

regress.display function for reactive data

Value

regress table

Examples

```
model1 <- glm(mpg ~ cyl + disp + vs, data = mtcars)  
regress.display2(model1, crude = TRUE, crude.p.value = TRUE, decimal = 3)
```

regressModule2	<i>regressModule2: Shiny modulde server for linear regression for reactive data.</i>
----------------	--

Description

Shiny modulde server for linear regression for reactive data.

Usage

```
regressModule2(  
  input,  
  output,  
  session,  
  data,  
  data_label,  
  data_varStruct = NULL,  
  nfactor.limit = 10,  
  design.survey = NULL,  
  default.unires = T,  
  limit.unires = 20,  
  vec.event = NULL  
)
```

Arguments

input	input
output	output
session	session
data	reactive data
data_label	reactive data label
data_varStruct	List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	reactive survey data. default: NULL
default.unires	Set default independent variables using univariate analysis, Default: T
limit.unires	Change to default.unires = F if number of independent variables > limit.unires, Default: 20
vec.event	event variables as vector for linear regression, Default: NULL

Details

Shiny modulde server for linear regression.

Value

Shiny module server for linear regression.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      regressModuleUI("linear")
    ),
    mainPanel(
      DTOutput("lineartable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_linear <- callModule(regressModule2, "linear",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$lineartable <- renderDT({
    datatable(out_linear()$table, rownames = T, caption = out_linear()$caption)
  })
}
```

regressModuleUI

regressModuleUI: shiny module UI for linear regression.

Description

Shiny module UI for linear regression.

Usage

```
regressModuleUI(id)
```

Arguments

id id

Details

Shiny module UI for linear regression.

Value

Shiny module UI for linear regression.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      regressModuleUI("linear")
    ),
    mainPanel(
      DTOutput("lineartable")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_linear <- callModule(regressModule2, "linear",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$lineartable <- renderDT({
    datatable(out_linear()$table, rownames = T, caption = out_linear()$caption)
  })
}
```

rocModule*rocModule: shiny module server for roc analysis*

Description

shiny module server for roc analysis

Usage

```
rocModule(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  id.cluster = NULL
)
```

Arguments

input	input
output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	Reactive survey data. default: NULL
id.cluster	Reactive cluster variable if marginal model, Default: NULL

Details

shiny module server for roc analysis

Value

shiny module server for roc analysis

See Also

[quantile](#) [setkey](#) [ggroc](#) [geeglm](#) [svyglm](#) [theme_modern](#)

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(pROC)
ui <- fluidPage(
  sidebarLayout(
```

```

    sidebarPanel(
      rocUI("roc")
    ),
    mainPanel(
      plotOutput("plot_roc"),
      tableOutput("cut_roc"),
      ggplotdownUI("roc"),
      DTOutput("table_roc")
    )
  )
}

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(data1))

  out_roc <- callModule(rocModule, "roc",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$plot_roc <- renderPlot({
    print(out_roc())$plot
  })

  output$cut_roc <- renderTable({
    if (is.null(out_roc())$cut)) return(NULL)
    print(out_roc())$cut
  })

  output$table_roc <- renderDT({
    datatable(out_roc())$tb,
    rownames = F, editable = F, extensions = "Buttons",
    caption = "ROC results",
    options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
  })
}

```

rocModule2

rocModule2: shiny module server for roc analysis- input number of model as integer

Description

shiny module server for roc analysis- input number of model as integer

Usage

```
rocModule2(
```

```

    input,
    output,
    session,
    data,
    data_label,
    data_varStruct = NULL,
    nfactor.limit = 10,
    design.survey = NULL,
    id.cluster = NULL
  )

```

Arguments

input	input
output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	Reactive survey data. default: NULL
id.cluster	Reactive cluster variable if marginal model, Default: NULL

Details

shiny module server for roc analysis- input number of model as integer

Value

shiny module server for roc analysis- input number of model as integer

See Also

[quantile](#) [setkey](#) [ggroc](#) [geeglm](#) [svyglm](#) [theme_modern](#)

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(pROC)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      rocUI("roc")
    ),
  ),

```

```

    mainPanel(
      plotOutput("plot_roc"),
      tableOutput("cut_roc"),
      ggplotdownUI("roc"),
      DTOutput("table_roc")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(data1))

  out_roc <- callModule(rocModule2, "roc",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$plot_roc <- renderPlot({
    print(out_roc()$plot)
  })

  output$cut_roc <- renderTable({
    print(out_roc()$cut)
  })

  output$table_roc <- renderDT({
    datatable(out_roc()$tb,
      rownames = F, editable = F, extensions = "Buttons",
      caption = "ROC results",
      options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
    )
  })
}

```

rocUI

rocUI: shiny module UI for roc analysis

Description

Shiny module UI for roc analysis

Usage

```
rocUI(id)
```

Arguments

id	id
----	----

Details

Shiny module UI for roc analysis

Value

Shiny module UI for roc analysis

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(pROC)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      rocUI("roc")
    ),
    mainPanel(
      plotOutput("plot_roc"),
      tableOutput("cut_roc"),
      ggplotdownUI("roc"),
      DTOutput("table_roc")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(data1))

  out_roc <- callModule(rocModule, "roc",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$plot_roc <- renderPlot({
    print(out_roc())$plot
  })

  output$cut_roc <- renderTable({
    if (is.null(out_roc())$cut) return(NULL)
    print(out_roc())$cut
  })

  output$table_roc <- renderDT({
    datatable(out_roc())$tb,
    rownames = F, editable = F, extensions = "Buttons",
    caption = "ROC results",
    options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
  })
}
```

```
    )
  })
}
```

ROC_table	<i>ROC_table: extract AUC, NRI and IDI information from list of roc object in pROC packages.</i>
-----------	--

Description

extract AUC, NRI and IDI information from list of roc in pROC packages

Usage

```
ROC_table(ListModel, dec.auc = 3, dec.p = 3)
```

Arguments

ListModel	list of roc object
dec.auc	digits for AUC, Default: 3
dec.p	digits for p value, Default: 3

Details

extract AUC, NRI and IDI information from list of roc object in pROC packages.

Value

table of AUC, NRI and IDI information

See Also

[ci.auc](#), [roc.test.data.table](#), [rbindlist](#)

Examples

```
library(pROC)
m1 <- glm(vs ~ am + gear, data = mtcars, family = binomial)
m2 <- glm(vs ~ am + gear + wt, data = mtcars, family = binomial)
m3 <- glm(vs ~ am + gear + wt + mpg, data = mtcars, family = binomial)
roc1 <- roc(m1$y, predict(m1, type = "response"))
roc2 <- roc(m2$y, predict(m2, type = "response"))
roc3 <- roc(m3$y, predict(m3, type = "response"))
list.roc <- list(roc1, roc2, roc3)
ROC_table(list.roc)
```

scatterServer	<i>scatterServer: shiny module server for scatterplot.</i>
---------------	--

Description

Shiny module server for scatterplot.

Usage

```
scatterServer(id, data, data_label, data_varStruct = NULL, nfactor.limit = 10)
```

Arguments

id	id
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10

Details

Shiny module server for scatterplot.

Value

Shiny module server for scatterplot.

Examples

```
library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      scatterUI("scatter")
    ),
    mainPanel(
      plotOutput("scatter_plot"),
      ggplotdownUI("scatter")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))
```

```

    out_scatter <- scatterServer("scatter",
      data = data, data_label = data.label,
      data_varStruct = NULL
    )

    output$scatter_plot <- renderPlot({
      print(out_scatter())
    })
  }
}

```

scatterUI

scatterUI: shiny module UI for scatterplot

Description

Shiny module UI for scatterplot

Usage

```
scatterUI(id, label = "scatterplot")
```

Arguments

id	id
label	label

Details

Shiny module UI for scatterplot

Value

Shiny module UI for scatterplot

Examples

```

library(shiny)
library(ggplot2)
library(ggpubr)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      scatterUI("scatter")
    ),
    mainPanel(
      plotOutput("scatter_plot"),
      ggplotdownUI("scatter")
    )
  )
)

```

```

)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_scatter <- scatterServer("scatter",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$scatter_plot <- renderPlot({
    print(out_scatter())
  })
}

```

survIDINRI_helper	<i>survIDINRI_helper: Helper function for IDI.INF.OUT in survIDINRI packages</i>
-------------------	--

Description

Helper function for IDI.INF.OUT in survIDINRI packages

Usage

```

survIDINRI_helper(
  var.event,
  var.time,
  list.vars.ind,
  t,
  data,
  dec.auc = 3,
  dec.p = 3,
  id.cluster = NULL
)

```

Arguments

var.event	event
var.time	time
list.vars.ind	list of independent variable
t	time
data	data
dec.auc	digits for AUC, Default: 3
dec.p	digits for p value, Default: 3
id.cluster	cluster variable if marginal model, Default: NULL

Details

Helper function for IDI.INF.OUT in survIDINRI packages

Value

IDI, NRI

See Also

[data.table](#) [model.matrix](#) [coxph](#) [Surv](#) [IDI.INF.OUT](#) [IDI.INF](#)

Examples

```
# library(survival)
# survIDINRI_helper("status", "time", list.vars.ind = list("age", c("age", "sex")),
#                   t = 365, data = lung)
```

tb1module	<i>tb1module: table 1 shiny module server.</i>
-----------	--

Description

Table 1 shiny module server for descriptive statistics.

Usage

```
tb1module(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
  showAllLevels = T,
  argsExact = list(workspace = 2 * 10^7, simulate.p.value = T)
)
```

Arguments

input	input
output	output
session	session
data	Data
data_label	Data label

data_varStruct Variable structure list of data, Default: NULL
 nfactor.limit maximum factor levels to include, Default: 10
 design.survey survey data of survey package. default: NULL
 showAllLevels Show All label information with 2 categorical variables, Default: T
 argsExact Option for Fisher exact test memory limit.

Details

Table 1 shiny module server for descriptive statistics.

Value

Table 1 shiny module server for descriptive statistics.

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      tb1moduleUI("tb1")
    ),
    mainPanel(
      DTOutput("table1")
    )
  )
)

server <- function(input, output, session) {
  data <- mtcars
  data.label <- jstable::mk.lev(mtcars)

  out_tb1 <- callModule(tb1module, "tb1",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$table1 <- renderDT({
    tb <- out_tb1()$table
    cap <- out_tb1()$caption
    out.tb1 <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
    return(out.tb1)
  })
}

```

tb1module2*tb1module2: table 1 shiny module server for reactive data.*

Description

Table 1 shiny module server for descriptive statistics for reactive data.

Usage

```
tb1module2(  
  input,  
  output,  
  session,  
  data,  
  data_label,  
  data_varStruct = NULL,  
  nfactor.limit = 10,  
  design.survey = NULL,  
  showAllLevels = T,  
  argsExact = list(workspace = 2 * 10^7, simulate.p.value = T)  
)
```

Arguments

input	input
output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	Variable structure list of data, Default: NULL
nfactor.limit	maximum factor levels to include, Default: 10
design.survey	Reactive survey data of survey package. Default: NULL
showAllLevels	Show All label information with 2 categorical variables, Default: T
argsExact	Option for Fisher exact test memory limit.

Details

Table 1 shiny module server for descriptive statistics.

Value

Table 1 shiny module server for descriptive statistics.

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      tb1moduleUI("tb1")
    ),
    mainPanel(
      DTOutput("table1")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_tb1 <- callModule(tb1module2, "tb1",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$table1 <- renderDT({
    tb <- out_tb1()$table
    cap <- out_tb1()$caption
    out.tb1 <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
    return(out.tb1)
  })
}

```

tb1moduleUI

tb1moduleUI: table 1 module UI.

Description

Table 1 shiny module UI for descriptive statistics.

Usage

```
tb1moduleUI(id)
```

Arguments

id id

Details

Table 1 shiny module UI for descriptive statistics.

Value

Table 1 module UI.

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      tb1moduleUI("tb1")
    ),
    mainPanel(
      DTOutput("table1")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- reactive(jstable::mk.lev(mtcars))

  out_tb1 <- callModule(tb1module2, "tb1",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$table1 <- renderDT({
    tb <- out_tb1()$table
    cap <- out_tb1()$caption
    out.tb1 <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
    return(out.tb1)
  })
}
```

tb1simple

tb1simple: tb1 module server for propensity score analysis

Description

Table 1 module server for propensity score analysis

Usage

```
tblsimple(
  input,
  output,
  session,
  data,
  matdata,
  data_label,
  data_varStruct = NULL,
  group_var,
  showAllLevels = T
)
```

Arguments

input	input
output	output
session	session
data	Original data with propensity score
matdata	Matching data
data_label	Data label
data_varStruct	List of variable structure, Default: NULL
group_var	Group variable to run propensity score analysis.
showAllLevels	Show All label information with 2 categorical variables, Default: T

Details

Table 1 module server for propensity score analysis

Value

Table 1 with original data/matching data/IPTW data

See Also

[var_label CreateTableOneJS svydesign](#)

Examples

```
library(shiny)
library(DT)
library(data.table)
library(readxl)
library(jstable)
library(haven)
library(survey)
ui <- fluidPage(
```

```

    sidebarLayout(
      sidebarPanel(
        FilePsInput("datafile"),
        tb1simpleUI("tb1")
      ),
      mainPanel(
        DTOutput("table1_original"),
        DTOutput("table1_ps"),
        DTOutput("table1_iptw")
      )
    )
  )
)

server <- function(input, output, session) {
  mat.info <- callModule(FilePs, "datafile")

  data <- reactive(mat.info()$data)
  matdata <- reactive(mat.info()$matdata)
  data.label <- reactive(mat.info()$data.label)

  vlist <- eventReactive(mat.info(), {
    mklist <- function(varlist, vars) {
      lapply(
        varlist,
        function(x) {
          inter <- intersect(x, vars)
          if (length(inter) == 1) {
            inter <- c(inter, "")
          }
          return(inter)
        }
      )
    }
  })
  factor_vars <- names(data())[data()[, lapply(.SD, class) %in% c("factor", "character")]]
  factor_list <- mklist(data_varStruct(), factor_vars)
  conti_vars <- setdiff(names(data()), c(factor_vars, "pscore", "iptw"))
  conti_list <- mklist(data_varStruct(), conti_vars)
  nclass_factor <- unlist(data()[, lapply(.SD, function(x) {
    length(unique(x)[!is.na(unique(x))])
  })],
  .SDcols = factor_vars
  ])
  class01_factor <- unlist(data()[, lapply(.SD, function(x) {
    identical(levels(x), c("0", "1"))
  })],
  .SDcols = factor_vars
  ])
  validate(
    need(!is.null(class01_factor), "No categorical variables coded as 0, 1 in data")
  )
  factor_01vars <- factor_vars[class01_factor]
  factor_01_list <- mklist(data_varStruct(), factor_01vars)

```

```

group_vars <- factor_vars[nclass_factor >= 2 & nclass_factor <= 10 &
  nclass_factor < nrow(data())]]
group_list <- mklist(data_varStruct(), group_vars)
except_vars <- factor_vars[nclass_factor > 10 | nclass_factor == 1 |
  nclass_factor == nrow(data())]]

## non-normal: shapiro test
f <- function(x) {
  if (diff(range(x, na.rm = T)) == 0) {
    return(F)
  } else {
    return(shapiro.test(x)$p.value <= 0.05)
  }
}

non_normal <- ifelse(nrow(data()) <= 3 | nrow(data()) >= 5000,
  rep(F, length(conti_vars)),
  sapply(conti_vars, function(x) {
    f(data()[[x]])
  })
)
return(list(
  factor_vars = factor_vars, factor_list = factor_list, conti_vars = conti_vars,
  conti_list = conti_list, factor_01vars = factor_01vars,
  factor_01_list = factor_01_list, group_list = group_list,
  except_vars = except_vars, non_normal = non_normal
))
})

out.tb1 <- callModule(tb1simple2, "tb1",
  data = data, matdata = matdata, data_label = data.label,
  data_varStruct = NULL, vlist = vlist,
  group_var = reactive(mat.info())$group_var
)

output$table1_original <- renderDT({
  tb <- out.tb1()$original$table
  cap <- out.tb1()$original$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
  return(out)
})

output$table1_ps <- renderDT({
  tb <- out.tb1()$ps$table
  cap <- out.tb1()$ps$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
  return(out)
})

output$table1_iptw <- renderDT({
  tb <- out.tb1()$iptw$table
  cap <- out.tb1()$iptw$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)

```

```
      return(out)
    })
  }
```

tb1simple2	<i>tb1simple2: tb1 module for propensity score analysis for reactive data</i>
------------	---

Description

tb1 module for propensity score analysis for reactive data

Usage

```
tb1simple2(
  input,
  output,
  session,
  data,
  matdata,
  data_label,
  data_varStruct = NULL,
  vlist,
  group_var,
  showAllLevels = T
)
```

Arguments

input	input
output	output
session	session
data	Original reactive data with propensity score
matdata	Matching reactive data
data_label	Reactive data label
data_varStruct	List of variable structure, Default: NULL
vlist	List including factor/continuous/binary/except/non-normal variables
group_var	Group variable to run propensity score analysis.
showAllLevels	Show All label information with 2 categorical variables, Default: T

Details

Table 1 module server for propensity score analysis

Value

Table 1 with original data/matching data/IPTW data

See Also

[CreateTableOneJS svydesign](#)

Examples

```
library(shiny)
library(DT)
library(data.table)
library(readxl)
library(jstable)
library(haven)
library(survey)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      FilePsInput("datafile"),
      tb1simpleUI("tb1")
    ),
    mainPanel(
      DTOutput("table1_original"),
      DTOutput("table1_ps"),
      DTOutput("table1_iptw")
    )
  )
)

server <- function(input, output, session) {
  mat.info <- callModule(FilePs, "datafile")

  data <- reactive(mat.info()$data)
  matdata <- reactive(mat.info()$matdata)
  data.label <- reactive(mat.info()$data.label)

  vlist <- eventReactive(mat.info(), {
    mklist <- function(varlist, vars) {
      lapply(
        varlist,
        function(x) {
          inter <- intersect(x, vars)
          if (length(inter) == 1) {
            inter <- c(inter, "")
          }
          return(inter)
        }
      )
    }
  })

  factor_vars <- names(data())[data()[, lapply(.SD, class) %in% c("factor", "character")]]
  factor_list <- mklist(data_varStruct(), factor_vars)
  conti_vars <- setdiff(names(data()), c(factor_vars, "pscore", "iptw"))
  conti_list <- mklist(data_varStruct(), conti_vars)
  nclass_factor <- unlist(data()[, lapply(.SD, function(x) {
```

```

      length(unique(x)[!is.na(unique(x))])
    }},
    .SDcols = factor_vars
  ])
  class01_factor <- unlist(data()[, lapply(.SD, function(x) {
    identical(levels(x), c("0", "1"))
  })],
    .SDcols = factor_vars
  ])
  validate(
    need(!is.null(class01_factor), "No categorical variables coded as 0, 1 in data")
  )
  factor_01vars <- factor_vars[class01_factor]
  factor_01_list <- mklist(data_varStruct(), factor_01vars)
  group_vars <- factor_vars[nclass_factor >= 2 & nclass_factor <= 10 &
    nclass_factor < nrow(data())]
  group_list <- mklist(data_varStruct(), group_vars)
  except_vars <- factor_vars[nclass_factor > 10 | nclass_factor == 1 |
    nclass_factor == nrow(data())]

  ## non-normal: shapiro test
  f <- function(x) {
    if (diff(range(x, na.rm = T)) == 0) {
      return(F)
    } else {
      return(shapiro.test(x)$p.value <= 0.05)
    }
  }

  non_normal <- ifelse(nrow(data()) <= 3 | nrow(data()) >= 5000,
    rep(F, length(conti_vars)),
    sapply(conti_vars, function(x) {
      f(data()[[x]])
    })
  )
  return(list(
    factor_vars = factor_vars, factor_list = factor_list, conti_vars = conti_vars,
    conti_list = conti_list, factor_01vars = factor_01vars,
    factor_01_list = factor_01_list, group_list = group_list,
    except_vars = except_vars, non_normal = non_normal
  ))
})

out.tb1 <- callModule(tb1simple2, "tb1",
  data = data, matdata = matdata, data_label = data.label,
  data_varStruct = NULL, vlist = vlist,
  group_var = reactive(mat.info())$group_var
)

output$table1_original <- renderDT({
  tb <- out.tb1()$original$table
  cap <- out.tb1()$original$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
})

```

```
      return(out)
    })

    output$table1_ps <- renderDT({
      tb <- out.tb1()$ps$table
      cap <- out.tb1()$ps$caption
      out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
      return(out)
    })

    output$table1_iptw <- renderDT({
      tb <- out.tb1()$iptw$table
      cap <- out.tb1()$iptw$caption
      out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
      return(out)
    })
  }
}
```

tb1simpleUI	<i>tb1simpleUI: tb1 module UI for propensity score analysis</i>
-------------	---

Description

Table 1 module UI for propensity score analysis.

Usage

```
tb1simpleUI(id)
```

Arguments

id id

Details

tb1 module UI for propensity score analysis

Value

Table 1 UI for propensity score analysis

Examples

```
library(shiny)
library(DT)
library(data.table)
library(readxl)
library(jstable)
library(haven)
library(survey)
```

```

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      FilePsInput("datafile"),
      tb1simpleUI("tb1")
    ),
    mainPanel(
      DTOutput("table1_original"),
      DTOutput("table1_ps"),
      DTOutput("table1_iptw")
    )
  )
)

server <- function(input, output, session) {
  mat.info <- callModule(FilePs, "datafile")

  data <- reactive(mat.info()$data)
  matdata <- reactive(mat.info()$matdata)
  data.label <- reactive(mat.info()$data.label)

  vlist <- eventReactive(mat.info(), {
    mklist <- function(varlist, vars) {
      lapply(
        varlist,
        function(x) {
          inter <- intersect(x, vars)
          if (length(inter) == 1) {
            inter <- c(inter, "")
          }
          return(inter)
        }
      )
    }
  })
  factor_vars <- names(data())[data()[, lapply(.SD, class) %in% c("factor", "character")]]
  factor_list <- mklist(data_varStruct(), factor_vars)
  conti_vars <- setdiff(names(data()), c(factor_vars, "pscore", "iptw"))
  conti_list <- mklist(data_varStruct(), conti_vars)
  nclass_factor <- unlist(data()[, lapply(.SD, function(x) {
    length(unique(x)[!is.na(unique(x))])
  })],
  .SDcols = factor_vars
  ])
  class01_factor <- unlist(data()[, lapply(.SD, function(x) {
    identical(levels(x), c("0", "1"))
  })],
  .SDcols = factor_vars
  ])
  validate(
    need(!is.null(class01_factor), "No categorical variables coded as 0, 1 in data")
  )
  factor_01vars <- factor_vars[class01_factor]

```

```

factor_01_list <- mklist(data_varStruct(), factor_01vars)
group_vars <- factor_vars[nclass_factor >= 2 & nclass_factor <= 10 &
  nclass_factor < nrow(data())]
group_list <- mklist(data_varStruct(), group_vars)
except_vars <- factor_vars[nclass_factor > 10 | nclass_factor == 1 |
  nclass_factor == nrow(data())]

## non-normal: shapiro test
f <- function(x) {
  if (diff(range(x, na.rm = T)) == 0) {
    return(F)
  } else {
    return(shapiro.test(x)$p.value <= 0.05)
  }
}

non_normal <- ifelse(nrow(data()) <= 3 | nrow(data()) >= 5000,
  rep(F, length(conti_vars)),
  sapply(conti_vars, function(x) {
    f(data()[[x]])
  })
)
return(list(
  factor_vars = factor_vars, factor_list = factor_list,
  conti_vars = conti_vars, conti_list = conti_list, factor_01vars = factor_01vars,
  factor_01_list = factor_01_list, group_list = group_list,
  except_vars = except_vars, non_normal = non_normal
))
})

out.tb1 <- callModule(tb1simple2, "tb1",
  data = data, matdata = matdata, data_label = data.label,
  data_varStruct = NULL, vlist = vlist,
  group_var = reactive(mat.info())$group_var
)

output$table1_original <- renderDT({
  tb <- out.tb1()$original$table
  cap <- out.tb1()$original$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
  return(out)
})

output$table1_ps <- renderDT({
  tb <- out.tb1()$ps$table
  cap <- out.tb1()$ps$caption
  out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
  return(out)
})

output$table1_iptw <- renderDT({
  tb <- out.tb1()$iptw$table
  cap <- out.tb1()$iptw$caption

```

```

      out <- datatable(tb, rownames = T, extension = "Buttons", caption = cap)
      return(out)
    })
  }

```

templateGenerator	<i>templateGenerator: Shiny Gadget for global/app.R template.</i>
-------------------	---

Description

Opens a Shiny app that allows users to generate a Shiny project template.

Usage

```
templateGenerator()
```

Details

Shiny Gadget including Data, Label info, Table 1, Regression(linear, logistic), Basic plots

Value

Shiny Gadget including Data, Label info, Table 1, Regression(linear, logistic), Basic plots

Examples

```

if (interactive()) {
  templateGenerator()
}

```

timeROChelper	<i>timeROChelper: Helper function for timerocModule</i>
---------------	---

Description

Helper function for timerocModule

Usage

```

timeROChelper(
  var.event,
  var.time,
  vars.ind,
  t,
  data,
  design.survey = NULL,
  id.cluster = NULL
)

```

Arguments

<code>var.event</code>	event
<code>var.time</code>	time
<code>vars.ind</code>	independent variable
<code>t</code>	time
<code>data</code>	data
<code>design.survey</code>	survey data, Default: NULL
<code>id.cluster</code>	cluster variable if marginal model, Default: NULL

Details

Helper function for timerocModule

Value

timeROC and coxph object

See Also

[coxph](#) [svycoxph](#) [predict](#) [timeROC](#)

Examples

```
# library(survival)
# timeROChelper("status", "time", c("age", "sex"), t = 365, data = lung)
```

timerocModule	<i>timerocModule: shiny module server for time-dependent roc analysis</i>
---------------	---

Description

shiny module server for time-dependent roc analysis

shiny module server for time-dependent roc analysis- input number of model as integer

Usage

```
timerocModule(
  input,
  output,
  session,
  data,
  data_label,
  data_varStruct = NULL,
  nfactor.limit = 10,
  design.survey = NULL,
```

```

    id.cluster = NULL,
    iid = TRUE,
    NRIIDI = TRUE
  )

  timerocModule2(
    input,
    output,
    session,
    data,
    data_label,
    data_varStruct = NULL,
    nfactor.limit = 10,
    design.survey = NULL,
    id.cluster = NULL,
    iid = T,
    NRIIDI = T
  )

```

Arguments

input	input
output	output
session	session
data	Reactive data
data_label	Reactive data label
data_varStruct	Reactive List of variable structure, Default: NULL
nfactor.limit	nlevels limit in factor variable, Default: 10
design.survey	Reactive survey data. default: NULL
id.cluster	Reactive cluster variable if marginal model, Default: NULL
iid	logical, get CI of AUC, Default: T
NRIIDI	logical, get NRI & IDI, Default: T

Details

shiny module server for time-dependent roc analysis

shiny module server for time dependent roc analysis- input number of model as integer

Value

shiny module server for time-dependent roc analysis

shiny module server for time dependent roc analysis- input number of model as integer

See Also

[quantile setkey data.table rbindlist](#)

[quantile setkey data.table rbindlist](#)

Examples

```

library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(timeROC)
library(survIDINRI)

ui <- fluidPage(sidebarLayout(
  sidebarPanel(timerocUI("timeroc")),
  mainPanel(
    plotOutput("plot_timeroc"),
    ggplotdownUI("timeroc"),
    DTOutput("table_timeroc")
  )
))

server <- function(input, output, session) {
  data <- reactive({
    dt_data <- as.data.table(pbc)

    factor_vars <- names(dt_data)[sapply(dt_data, function(x){length(table(x))}) <= 6]
    dt_data[, (factor_vars) := lapply(.SD, factor), .SDcols = factor_vars]

    return(dt_data)
  })

  data.label <- reactive({
    jstable::mk.lev(data())
  })

  out_timeroc <- callModule(
    timerocModule,
    "timeroc",
    data = data,
    data_label = data.label,
    data_varStruct = NULL
  )

  observe({
    tb <- tryCatch(out_timeroc())$tb, error = function(e) NULL)
    print(tb)
  })

  output$plot_timeroc <- renderPlot({
    {
      print(out_timeroc())$plot)
    }
  })
}

```

```

output$table_timeroc <- renderDT({
  datatable(
    out_timeroc()$tb,
    rownames = F,
    editable = F,
    extensions = "Buttons",
    caption = "ROC results",
    options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
  )
})

}

library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(timeROC)
library(survIDINRI)

ui <- fluidPage(sidebarLayout(
  sidebarPanel(timerocUI("timeroc")),
  mainPanel(
    plotOutput("plot_timeroc"),
    ggplotdownUI("timeroc"),
    DTOutput("table_timeroc")
  )
))

server <- function(input, output, session) {
  data <- reactive({
    dt_data <- as.data.table(pbc)

    factor_vars <- names(dt_data)[sapply(dt_data, function(x){length(table(x))}) <= 6]
    dt_data[, (factor_vars) := lapply(.SD, factor), .SDcols = factor_vars]

    return(dt_data)
  })

  data.label <- reactive({
    jstable::mk.lev(data())
  })

  out_timeroc <- callModule(
    timerocModule2,
    "timeroc",
    data = data,
    data_label = data.label,

```

```

      data_varStruct = NULL
    )

    observe({
      tb <- tryCatch(out_timeroc())$tb, error = function(e) NULL)
      print(tb)
    })

    output$plot_timeroc <- renderPlot({
      {
        print(out_timeroc())$plot)
      }
    })

    output$table_timeroc <- renderDT({
      datatable(
        out_timeroc()$tb,
        rownames = F,
        editable = F,
        extensions = "Buttons",
        caption = "ROC results",
        options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
      )
    })

  }

```

timerocUI

timerocUI: shiny module UI for time-dependent roc analysis

Description

Shiny module UI for time-dependent roc analysis

Usage

```
timerocUI(id)
```

Arguments

id id

Details

Shiny module UI for time-dependent roc analysis

Value

Shiny module UI for time-dependent roc analysis

Examples

```
library(shiny)
library(DT)
library(data.table)
library(jstable)
library(ggplot2)
library(timeROC)
library(survIDINRI)
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      timerocUI("timeroc")
    ),
    mainPanel(
      plotOutput("plot_timeroc"),
      ggplotdownUI("timeroc"),
      DTOutput("table_timeroc")
    )
  )
)

server <- function(input, output, session) {
  data <- reactive(mtcars)
  data.label <- jstable::mk.lev(mtcars)

  out_timeroc <- callModule(timerocModule, "timeroc",
    data = data, data_label = data.label,
    data_varStruct = NULL
  )

  output$plot_timeroc <- renderPlot({
    print(out_timeroc()$plot)
  })

  output$table_timeroc <- renderDT({
    datatable(out_timeroc()$tb,
      rownames = F, editable = F, extensions = "Buttons",
      caption = "ROC results",
      options = c(jstable::opt.tbreg("roctable"), list(scrollX = TRUE))
    )
  })
}
```

timeROC_table

timeROC_table: extract AUC information from list of timeROChelper object.

Description

extract AUC information from list of timeROChelper object.

Usage

```
timeROC_table(ListModel, dec.auc = 3, dec.p = 3)
```

Arguments

ListModel	list of timeROChelper object
dec.auc	digits for AUC, Default: 3
dec.p	digits for p value, Default: 3

Details

extract AUC information from list of timeROChelper object.

Value

table of AUC information

See Also

[confint.data.table](#)

Examples

```
# library(survival)
# list.timeROC <- lapply(list("age", c("age", "sex")),
#                         function(x){
#                           timeROChelper("status", "time", x, t = 365, data = lung)
#                         })
# timeROC_table(list.timeROC)
```

use_jsmodule_style	<i>Include jsmodule CSS styling</i>
--------------------	-------------------------------------

Description

Adds the custom 'style.css' file bundled with the jsmodule package to a Shiny UI. This allows consistent styling (e.g., bold navbar title, font tweaks, spacing) across all Shiny applications using this package.

Usage

```
use_jsmodule_style()
```

Details

This function is meant to be used inside the UI of a Shiny app. It automatically locates and includes the ‘style.css’ file found in ‘inst/assets/’ of the jsmodule package installation.

Value

An HTML ‘<link>’ tag that loads the CSS into a Shiny UI

See Also

[include](#)

Examples

```
## Not run:  
use_jsmodule_style()  
  
## End(Not run)
```

Index

add_slide, [18](#), [21](#)

barServer, [3](#)
barUI, [5](#)
boxServer, [6](#)
boxUI, [7](#)

ci.auc, [67](#)
coef, [21](#)
colon, [43](#)
confint, [92](#)
cor, [21](#)
cox2.display, [41](#)
coxModule, [8](#)
coxph, [41](#), [71](#), [86](#)
coxUI, [10](#)
CreateTableOneJS, [76](#), [80](#)
csvFile, [11](#)
csvFileInput, [11](#)

data.table, [41](#), [67](#), [71](#), [87](#), [92](#)
dml, [18](#), [21](#)
dropdownButton, [56](#)

FilePs, [12](#)
FilePsInput, [13](#)
FileRepeated, [14](#)
FileRepeatedInput, [14](#)
FileSurvey, [15](#)
FileSurveyInput, [16](#)
forest, [18](#), [21](#)
forest_theme, [18](#), [21](#)
forestcoxServer, [17](#)
forestcoxUI, [19](#)
forestglmServer, [21](#)
forestglmUI, [22](#)
fwrite, [38](#), [41](#), [43](#), [45](#)

geeglm, [62](#), [64](#)
GEEModuleLinear, [24](#)
GEEModuleLogistic, [26](#)
GEEModuleUI, [27](#)
ggpairsModule, [29](#)
ggpairsModule2, [30](#)
ggpairsModuleUI1, [32](#)
ggpairsModuleUI2, [33](#)
ggplotdownUI, [34](#)
ggroc, [62](#), [64](#)
ggsave, [41](#)

histogramServer, [35](#)
histogramUI, [36](#)

IDI.INF, [71](#)
IDI.INF.OUT, [71](#)
include, [93](#)

jsBasicAddin, [37](#)
jsBasicExtAddin, [38](#)
jsBasicGadget, [39](#)
jskm, [41](#)
jsPropensityAddin, [40](#)
jsPropensityExtAddin, [40](#)
jsPropensityGadget, [41](#)
jsRepeatedAddin, [42](#)
jsRepeatedExtAddin, [42](#)
jsRepeatedGadget, [43](#)
jsSurveyAddin, [44](#)
jsSurveyExtAddin, [44](#)
jsSurveyGadget, [45](#)

kaplanModule, [46](#)
kaplanUI, [48](#)

lineServer, [49](#)
lineUI, [50](#)
logistic.display2, [51](#)
logisticModule2, [52](#)
lung, [38](#)

match.data, [41](#)
matchit, [41](#)

mk.lev2, [54](#)
mklist, [54](#)
mksetdiff, [55](#)
model.matrix, [71](#)

opt.tb1, [45](#)
opt.tbreg, [38](#), [41](#), [43](#), [45](#)
optionUI, [55](#)

pbc, [41](#)
ph_location, [18](#), [21](#)
ph_with, [18](#), [21](#)
predict, [86](#)

quantile, [62](#), [64](#), [87](#)

rbindlist, [67](#), [87](#)
rcorrp.cens, [57](#)
read_pptx, [18](#), [21](#)
reclassificationJS, [56](#)
regress.display2, [58](#)
regressModule2, [59](#)
regressModuleUI, [60](#)
roc.test, [67](#)
ROC_table, [67](#)
rocModule, [61](#)
rocModule2, [63](#)
rocUI, [65](#)

scatterServer, [68](#)
scatterUI, [69](#)
setattr, [18](#), [21](#)
setDT, [18](#), [21](#)
setkey, [62](#), [64](#), [87](#)
Surv, [41](#), [71](#)
surveysummary, [21](#)
survfit, [41](#)
survIDINRI_helper, [70](#)
svycox.display, [41](#)
svycoxph, [86](#)
svydesign, [41](#), [76](#), [80](#)
svyglm, [62](#), [64](#)
svyjkm, [41](#)
svykm, [41](#)
svytable, [21](#)

TableSubgroupMultiCox, [18](#)
TableSubgroupMultiGLM, [21](#)
tb1module, [71](#)
tb1module2, [73](#)

tb1moduleUI, [74](#)
tb1simple, [75](#)
tb1simple2, [79](#)
tb1simpleUI, [82](#)
templateGenerator, [85](#)
theme_modern, [62](#), [64](#)
timeROC, [86](#)
timeROC_table, [91](#)
timeROChelper, [85](#)
timerocModule, [86](#)
timerocModule2(timerocModule), [86](#)
timerocUI, [90](#)
tooltipOptions, [56](#)

use_jsmodule_style, [92](#)

var_label, [76](#)