Package 'ggplotAssist'

May 8, 2026

Type Package

Title 'RStudio' Addin for Teaching and Learning 'ggplot2'

Version 0.1.3

Imports shiny (>= 0.13), miniUI (>= 0.1.1), rstudioapi (>= 0.5), shinyWidgets, shinyAce, stringr, tidyverse, ggplot2, dplyr, magrittr, tibble, scales, ggthemes, gcookbook, moonBook, editData

Suggests knitr, rmarkdown, markdown

URL https://github.com/cardiomoon/ggplotAssist

BugReports https://github.com/cardiomoon/ggplotAssist/issues

Description An 'RStudio' addin for teaching and learning making plot using the 'ggplot2' package. You can learn each steps of making plot by clicking your mouse without coding. You can get resultant code for the plot.

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Keon-Woong Moon [aut, cre]

Maintainer Keon-Woong Moon <cardiomoon@gmail.com>

Repository CRAN

Date/Publication 2017-11-12 12:24:46 UTC

Contents

`ggplotAssist` *A shiny app for learn ggplot2*

Description

A shiny app for learn ggplot2

Usage

```
ggplotAssist(df = NULL, viewer = "browser")
```

Arguments

`df`	A tibble or a tbl_df or a data.frame to manipulate
`viewer`	Specify where the gadget should be displayed. Possible choices are c("dialog","browser","pane")

Value

An R code for ggplot

Examples

```
library(tidyverse)
library(rstudioapi)
library(miniUI)
library(moonBook)
library(shinyAce)
library(ggthemes)
library(shiny)
library(stringr)
library(editData)
library(shinyWidgets)
library(gcookbook)
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
    result<-ggplotAssist(mtcars)
    result
}
```

selectizeInput3 *side-by-side selectizeInput*

Description

side-by-side selectizeInput

Usage

```
selectizeInput3(..., width = 100)
```

Arguments

...	Further arguments to be passed to selectizeInput
width	Input width in pixel

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
        selectizeInput3("color", "color", choices=colors())
    )
    server <- function(input, output) {

    }
    shinyApp(ui, server)
}
```

splitData *Elongate data.frame with column split by comma*

Description

Elongate data.frame with column split by comma

Usage

```
splitData(df, colname)
```

Arguments

df	a data.frame
colname	column name

Value

An elongated data.frame

`textAreaInput4` *Create side-by side textAreaInput with disabled spell check*

Description

Create side-by side textAreaInput with disabled spell check

Usage

```
textAreaInput4(inputId, label, value = "", bg = NULL, width = "100%",
  ...)
```

Arguments

`inputId`	The input slot that will be used to access the value.
`label`	Display label for the control, or NULL for no label.
`value`	Initial value.
`bg`	backgroung color
`width`	The width of the input in pixel
`...`	arguments to be passed to textInput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
     ui <- fluidPage(
          textAreaInput4("Code","Code","")
     )
     server <- function(input, output) {

     }
     shinyApp(ui, server)
}
```

textFunction *Server function of textFunction shiny module*

Description

Server function of textFunction shiny module

Usage

```
textFunction(input, output, session, argList = reactive(argList),
  editCode = reactive(TRUE), settingData = reactive(NULL))
```

Arguments

input	input
output	output
session	session
argList	A list containing options
editCode	Logical. Wheter or not edit initial R code
settingData	A data.frame contains information about functions

textFunctionInput *UI of textFunction shiny module*

Description

UI of textFunction shiny module

Usage

```
textFunctionInput(id)
```

Arguments

id	A string

Examples

```
library(ggplotAssist)
library(shiny)
# Only run examples in interactive R sessions
if(interactive()){
ui=fluidPage(
   textFunctionInput("text"),
   textOutput("text")
)
server=function(input,output,session){
   rv=reactiveValues()
   rawData=read.csv("data-raw/setting.csv",stringsAsFactors = FALSE)
   settingData=splitData(rawData,"setting")
  rv$argList<-list(label="text",mode="text",value="element_text()",choices=NULL,width=200,
                   bg="lightcyan",placeholder="")
   result=callModule(textFunction,"text",argList=reactive(rv$argList),
                     editCode=reactive(TRUE),settingData=reactive(settingData))
   output$text=renderText({
       result()
   })
}
shinyApp(ui,server)
}
```

`textInput4` *Create side-by side textInput with disabled spell check*

Description

Create side-by side textInput with disabled spell check

Usage

```
textInput4(inputId, label, value = "", width = 100, bg = NULL, ...)
```

Arguments

`inputId`	The input slot that will be used to access the value.
`label`	Display label for the control, or NULL for no label.
`value`	Initial value.
`width`	The width of the input in pixel
`bg`	backgroung color
`...`	arguments to be passed to textInput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
     ui <- fluidPage(
          textInput4("id", "id", ""),
          textInput4("name","name","")
     )
     server <- function(input, output) {

     }
     shinyApp(ui, server)
}
```

`uiOutput3` *Create side-by side uiOutput*

Description

Create side-by side uiOutput

Usage

```
uiOutput3(...)
```

Arguments

`...`	arguments to be passed to uiOutput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
     ui <- fluidPage(
          textInput4("name","name",""),
          uiOutput3("test")
     )
     server <- function(input, output) {

     }
     shinyApp(ui, server)
}
```

Index