Package ‘tkRplotR’

June 20, 2020

Type Package

Title Display Resizable Plots

Version 0.1.3

Author Filipe Campelo

Maintainer Filipe Campelo <`fcampelo@ci.uc.pt`>

Description Display a plot in a Tk canvas.

License GPL (>= 2)

Depends R (>= 3.5), tcltk, grDevices

SystemRequirements Tcl/Tk (>= 8.6)

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-20 19:30:03 UTC

R topics documented:

addTkBind *Add Tk Binds*

Description

Add binds to previous defined bindings

Usage

```
addTkBind(win, event, fun = NULL)
```

Arguments

`win`	window
`event`	event
`fun`	a function

Details

This function adds a new bind while keeping the previous defined binds.

Examples

```
## Not run:

tt <- tktoplevel()
tt <- tkRplot(tt, function () plot(1:10))
FUN <- local({
  canPos <-.Tcl(paste(tt$env$canvas, "create text 0 0 "))
    function (x, y) {
        x <- as.numeric(x)
        y <- as.numeric(y)
       tkdelete(tt$env$canvas, tclvalue(canPos))
           xy <- formatC(tk2usr(x, y),
                   digits = 2,
                     format = "f",
                       width = 5)
    canPos <<- .Tcl(
     paste(tt$env$canvas, "create text 40 10 -fill blue -justify left -text { ",
            xy[1], " ", xy[2],
            "} -font {Helvetica -10}"))
  }})

tkbind(tt$env$canvas, "<Motion>", FUN)
tkbind(tt$env$canvas, "<Motion>") #to give current bidings
FUN1 <- function (x,y) print(tk2usr(x,y))
addTkBind(tt$env$canvas, "<Motion>", FUN1)
tkbind(tt$env$canvas, "<Motion>") #to give current bidings
```

```
## End(Not run)
```

---

setCoef — *Functions to Convert Tk and User Coordinates*

Description

Convert Tk coordinates from/to user coordinates.

Usage

```
setCoef(W, width, height)
getCoef(W)
tk2usr(W, x = NULL, y = NULL)
usr2tk(W, x = NULL, y = NULL)
```

Arguments

`W`	the window (toplevel). If W is missing the getCoef function returns the coefficients for the last toplevel visited.
`width`	width of the canvas (image)
`height`	height of the canvas (image)
`x`	x position.
`y`	y position.

Examples

```
## Not run:

bb <- 1
tt <- tktoplevel()
tt <- tkRplot(tt, function() {
 x <- 1:20 / 20
   plot(
   x,
   x ^ bb,
   col = "#0000ff50",
       xlab = "x",
           ylab = paste0("x^", bb),
               type = "l",
                   axes = FALSE,
                       lwd = 4)
   title(main = bb)
     points(x,
      x ^ bb,
      col = "#ff000050",
      pch = 19,
      cex = 2)
        axis(1)
        axis(2)
          box()
          })

 getCoef()

 tkbind(tt$env$canvas, "<Button-1>", function(x, y)
 print(tk2usr(x, y)))

 # A more complex example
 local({
 canPos <-.Tcl(paste(tt$env$canvas, "create text 0 0 "))
 canPosX <-.Tcl(paste(tt$env$canvas, "create text 0 0 "))
 canPosY <-.Tcl(paste(tt$env$canvas, "create text 0 0 "))
 lineVertical <- .Tcl(paste(tt$env$canvas, "create line 0 0 0 0"))
 lineHorizontal<-.Tcl(paste(tt$env$canvas, "create line 0 0 0 0"))
 tkbind(tt, "<Motion>", function (x, y) {
   x <- as.numeric(x)
     y <- as.numeric(y)
       for (i in c(canPos, lineVertical, lineHorizontal,canPosX,canPosY))
       tkdelete(tt$env$canvas, tclvalue(i))

            xy <- formatC(tk2usr(x, y),
                             digits = 2,
                              format = "f",
                               width = 5)

xRange <- tt$env$plt[1:2] * tt$env$width
   yRange <- (1 - tt$env$plt[4:3]) * tt$env$height
     canPos <<- .Tcl(
 paste(tt$env$canvas, "create text 40 10 -fill blue -justify left -text { ",
     xy[1], " ", xy[2],
     "} -font {Helvetica -10}"))
      if (x < xRange[1] | x > xRange[2])
         return()
         if (y < yRange[1] | y > yRange[2])
           return()
canPosX <<- .Tcl(paste(tt$env$canvas, "create text ", x, yRange[1]-10,
           " -fill blue -justify center -text { ",xy[1],
           "} -font {Helvetica -10}"))
canPosY <<- .Tcl(paste(tt$env$canvas, "create text ",xRange[2]+10, y,
   " -fill blue -justify center -text { ",xy[2], "} -font {Helvetica -10}"))
           lineVertical <<- .Tcl(paste(tt$env$canvas, "create line ",
                 x,      yRange[1],      x,      yRange[2],
                    "-fill blue -dash 4"))
           lineHorizontal <<- .Tcl(paste(tt$env$canvas,
                 "create line ",
                  xRange[1],  y, xRange[2], y,
                       "-fill blue -dash 4"))})
     tkbind(tt$env$canvas, "<Leave>", function (x, y)
     {tkdelete(tt$env$canvas, tclvalue(canPos))}})
     } )


## End(Not run)
```

setVariable *Set and Get Variables*

Description

Define and get variables

Usage

```
setVariable(name, value = NULL)
getVariable(name)
```

Arguments

`name`	name of the variable
`value`	the value of the variable

Examples

```
setVariable("var1", 1)
exists("var1")
getVariable("var1")

getVariable("tkRplotR_pngType")
```

tkBinds *Define Tk Binds To Allow Automatic Resizing*

Description

Add binds to automatically resize the graph

Usage

```
tkBinds(parent, expose = TRUE, configure = TRUE)
```

Arguments

`parent`	parent Tk toplevel window
`expose`	if TRUE update graph when the window is expose
`configure`	if TRUE update the graph when the window is update

Details

This function adds the binds needed to automatically resize the graph

Examples

```
## Not run:
bb <- 1
tkbb <- tclVar(1)
tt <- tktoplevel()
tt <- tkRplot(tt, function() {
 x <- 1:20 / 20
   plot(
   x,
   x ^ bb,
   col = "#0000ff50",
       xlab = "x",
           ylab = paste0("x^", bb),
               type = "l",
                   axes = FALSE,
                       lwd = 4)
   title(main = bb)
     points(x,
      x ^ bb,
      col = "#ff000050",
      pch = 19,
      cex = 2)
        axis(1)
        axis(2)
          box()
          })

 f <- function(...) {
 b <- as.numeric(tclvalue(tkbb))
   if (b != bb) {
       bb <<- b
           tkRreplot(tt)
             }
     }

     s <-
       tkscale(
       tt,
       command = f,
       from = 0.05,
       to = 2.00,
       variable = tkbb,
       showvalue = FALSE,
       resolution = 0.05,
       orient = "horiz"
       )
```

```
        tkpack(s,
        side = "bottom",
        before = tt$env$canvas,
        expand = FALSE,
        fill = "both")

 # to disable the automatic resizing of the graph
   tkBinds(parent = tt, expose = FALSE, configure = FALSE)

 # to enable again the automatic resising
 # tkBinds(parent = tt, expose = TRUE, configure = TRUE)

## End(Not run)
```

`tkLocator` *Gives the Position*

Description

Gives the position when the left mouse button is pressed + "Ctrl" button.

Usage

```
tkLocator(parent, n = 1)
```

Arguments

`parent`	Tk toplevel window
`n`	the number of points to locate

Value

A list with x and y components which are the coordinates of the identified points.

Examples

```
## Not run:
bb <- 1
tt <- tktoplevel()
tt <- tkRplot(tt, function() {
 x <- 1:20 / 20
   plot(
   x,
   x ^ bb,
   col = "#0000ff50",
       xlab = "x",
           ylab = paste0("x^", bb),
               type = "l",
                   axes = FALSE,
                       lwd = 4)
```

```
    title(main = bb)
      points(x,
       x ^ bb,
       col = "#ff000050",
       pch = 19,
       cex = 2)
         axis(1)
         axis(2)
           box()
           })
 tkLocator(tt, 2)

## End(Not run)
```

---

tkRplot *Tk Rplot With Resizing*

Description

Dispaly a plot in a Tk toplevel window.

Usage

```
tkRplot(W, fun, width = 490, height = 490, ...)
tkRreplot(W, fun, width, height,  ...)
.tkRreplot(W)
```

Arguments

`W`	Tk toplevel window
`fun`	function to produce the plot
`width`	image width
`height`	image height
`...`	additional arguments

Examples

```
## Not run:
bb <- 1
tkbb <- tclVar(1)
tt <- tktoplevel()
f <- function(...) {
 b <- as.numeric(tclvalue(tkbb))
   if (b != bb) {
       bb <<- b
           tkRreplot(tt)
```

```
                }
        }

    tt <- tkRplot(tt, function() {
     x <- 1:20 / 20
       plot(
       x,
       x ^ bb,
       col = "#0000ff50",
           xlab = "x",
               ylab = paste0("x^", bb),
                   type = "l",
                       axes = FALSE,
                           lwd = 4)
       title(main = bb)
         points(x,
          x ^ bb,
          col = "#ff000050",
          pch = 19,
          cex = 2)
            axis(1)
            axis(2)
              box()
              })

      s <- tkscale(
           tt,
           command = f,
           from = 0.05,
           to = 2.00,
           variable = tkbb,
           showvalue = TRUE,
           resolution = 0.01,
           repeatdelay = 50,
           repeatinterval = 100,
           orient = "horiz"
           )

           tkpack(s,
           side = "bottom",
           expand = FALSE,
           before = tt$env$canvas,
           fill = "both")


    ## End(Not run)
```

Index