

Package ‘tkRplotR’

July 22, 2025

Version 0.1.7

Title Display Resizable Plots

Author Filipe Campelo <fcampelo@ci.uc.pt>

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

ByteCompile true

LazyLoad no

LazyData no

Repository CRAN

RoxygenNote 7.1.1

NeedsCompilation no

Date/Publication 2021-06-27 09:40:02 UTC

Contents

tkRplotR-package	2
addTkBind	2
setCoef	3
setVariable	5
tkBinds	6
tkLocator	7
tkRplot	8

Index	12
--------------	-----------

tkRplotR-package	<i>Display Resizable Plots</i>
------------------	--------------------------------

Description

This package contains functions for plotting in a Tk canvas.

Details

Package:	tkRplotR
Type:	Package
License:	GPL (>= 2)

Main Functions

`tkRplot` display a plot in a Tk toplevel window

`tkRreplot` refresh the plot created by `tkRplot`

Author(s)

Filipe Campelo <fcampelo@ci.uc.pt>

addTkBind	<i>Add Tk Binds</i>
-----------	---------------------

Description

Add binds to previous defined bindings

Usage

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

Arguments

<code>win</code>	window
<code>event</code>	event
<code>fun</code>	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, Get, and Remove Variables***Description**

Define, get, and remove variables

Usage

```

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

```

Arguments

name	name of the variable
value	the value of the variable

Examples

```

setVariable("var1", 1)
exists("var1")
getVariable("var1")
rmVariable("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:
tkbb <- tclVar(1)
tt <- tkToplevel()
tt <- tkRplot(tt, function() {
b <- .tcl2num(tkbb)
x <- 1:20 / 20
plot(
  x,
  x ^ b,
  col = "#0000ff50",
  xlab = "x",
  ylab = paste0("x^", b),
  type = "l",
  axes = FALSE,
  lwd = 4)
title(main = b)

```

```

points(x,
  x ^ b,
  col = "#ff000050",
  pch = 19,
  cex = 2)
  axis(1)
  axis(2)
  box()
})

s <-
tkscale(
  tt,
  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

Displays 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:
#Example 1 without using tkReplot function (tkRplotR version > 0.1.6)
tk_b <- tclVar(1)
tk_x <- tclVar(10)
tk_main <- tclVar('...')

tt0 <- tkToplevel()
tt0 <- tkRplot(tt0, function(...) {
# get values of tclvariables
  x <- .tcl2num(tk_x)
  x <- 1:x
  b <- .tcl2num(tk_b)
  main <- .tcl2String(tk_main)

  plot(
    x,
    x ^ b ,
    col = "#0000ff50",
    xlab = "x",
    ylab = expression(x^b),
    type = "l",
    axes = FALSE,
    lwd = 4)
  title(main = main)
  points(x,
    x ^ b,
    col = "#ff000050",
    pch = 19,
    cex = 2)

  axis(1)
  axis(2)
  box()
})

s01 <- tkScale(
  tt0,
  #command = function(...) .tkRreplot(tt0),
  from = 10,
  to = 60,
  label = 'x',
  variable = tk_x,
```

```

    showvalue = TRUE,
    resolution = 1,
    repeatdelay = 200,
    repeatinterval = 100,
    orient = "hor"
)

s02 <- tkscale(
  tt0,
  #command = function(...) .tkRreplot(tt0),
  from = 0.05,
  to = 2.00,
  label = 'b',
  variable = tk_b,
  showvalue = TRUE,
  resolution = 0.01,
  repeatdelay = 200,
  repeatinterval = 100,
  orient = "ver"
)

e01 <- tkentry(tt0,
               textvariable = tk_main,
               validate = 'all', validatecommand="")
tkpack(s02,
       side = "left",
       expand = FALSE,
       #'anchor = "c",
       before = tt0$env$canvas,
       fill = "both")

tkpack(s01,
       side = "bottom",
       expand = FALSE,
       #'anchor = "c",
       before = tt0$env$canvas,
       fill = "both")

tkpack(e01,
       side = "top",
       expand = FALSE,
       #'anchor = "c",
       before = tt0$env$canvas,
       fill = "both")

#Example 2 using tkReplot function (tkRplotR version < 0.1.7)
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 <- tk scale(
  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

* package

tkRplotR-package, [2](#)

.tkRreplot (tkRplot), [8](#)

addTkBind, [2](#)

getCoef (setCoef), [3](#)

getVariable (setVariable), [5](#)

rmVariable (setVariable), [5](#)

setCoef, [3](#)

setVariable, [5](#)

tk2usr (setCoef), [3](#)

tkBinds, [6](#)

tkLocator, [7](#)

tkRplot, [2](#), [8](#)

tkRplotR (tkRplotR-package), [2](#)

tkRplotR-package, [2](#)

tkRreplot, [2](#)

tkRreplot (tkRplot), [8](#)

usr2tk (setCoef), [3](#)