Package 'miniGUI'

October 13, 2022

Version 0.8-1

Date 2018-05-21

Title Tcl/Tk Quick and Simple Function GUI

Author Jorge Luis Ojeda Cabrera <jojeda@unizar.es>

Maintainer Jorge Luis Ojeda Cabrera <jojeda@unizar.es>

Description Quick and simple Tcl/Tk Graphical User Interface
to call functions. Also comprises a very simple experimental
GUI framework.

Depends R (>= 2.5.0), tcltk

License GPL (<= 2.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-22 11:26:51 UTC

R topics documented:

`makeWidgetCmd` *R functions to build a GUI window*

Description

Function that wraps the result of `mapFuncToWidget` as an R\ function that pop us a widget representing the function.

Usage

```
makeWidgetCmd(frmTitle,fun,baseFrame=.TkRoot,STORE=storageName(),
              GRAB=TRUE,SINGLE.EVAL=FALSE)
```

Arguments

frmTitle	title of the GUI window.
fun	function to map.
baseFrame	tcltk parent frame of the GUI window for the function fun.
STORE	A string. Name of the place where to store details needed by the GUI to perform the execution.
GRAB	Logical. When TRUE disable input in any other window.
SINGLE.EVAL	Logical. When set to TRUE it avoids return any value till computation is really finished.

Details

The main use of this function is to obtain a function that called creates a widget that allows the parameter input and execution of function fun. It also adds a **Quit** fun function to close the widget.

Value

This function returns an R\ function.

Author(s)

Jorge Luis Ojeda Cabrera (<jojeda@unizar.es>).

See Also

miniGUI, mapFuncToWidget, tcltk.

Examples

```
require(tcltk)
##
##  a simple example
##
g <- function(a=1,b=rnorm) {cat("--g--");paste("g(a,b)=",a+b(a))}
h <- function(a=1,b=3,c=3) {cat("--h--");paste("h(a,b,c)=",a+b+c)}
## create functions
gg <- makeWidgetCmd("Hay it is g !!",g,GRAB=FALSE)
hh <- makeWidgetCmd("Hay h here !!",h,GRAB=FALSE)
## calling them
gg()
cat("\nClose it before calling hh(), they sharer parameters a and b!!")
hh()
##
##  simple example(continuation)
```

```
##
## to be able to use both at the same time:save info for h in other place
hh <- makeWidgetCmd("Hay h here !!",h,STORE="h")
gg()
hh()
```

`mapFuncToWidget` *Map R functions to a GUI window*

Description

Function map a large class of R functions onto a set of `tcltk` widgets that allows the input of its parameter.

Usage

```
mapFuncToWidget(f,frm,bttLabel="OK",STORE="ff",callSubst="mini GUI call")
```

Arguments

`f`	Function to map.
`frm`	`tcltk` frame to place the GUI window.
`bttLabel`	execution button label.
`STORE`	A string. Name of the place where to store details needed by the GUI to perform the execution.
`callSubst`	string to set `call` attribute/slot in some of the R computations results.

Details

This function returns a frame which contains pairs of `tcltk` labels and text entry (or any other tktcl widget that allows to input values) and a button. In this way, this functions maps an R\ function `f` into a GUI window that allows its computation. Therefore, it provides a map from the a set of R\ function onto some class of GUI windows.

Usually, the way function is executed provides with nasty and long `call` attributes, `call` parameter substitute these allowing a much more comfortable output.

The string provided by `STORE` is used to store the function arguments in the list `miniGUIEnvir$miniGUIData`, enabling in this way the computation of the function.

Value

This function returns a `tcltk` frame (an object created with `tkframe`).

Author(s)

Jorge Luis Ojeda Cabrera (<jojeda@unizar.es>).

See Also

miniGUI, makeWidgetCmd, tcltk.

Examples

```
require(tcltk)
##
## a window for lm
##
## create some data(in the global environment)
n <- 100
d <- data.frame(x=runif(n))
d$z <- 0.5 * rnorm(n)
d$y <- 2 * d$x + d$z
## create a tcltk frame and give it a title
frm <- tktoplevel()
tkwm.title(frm,"mapFuncToWidget for lm")
## create the GUI window map of lm
mapFuncToWidget(lm,frm)
## ...you may close the window


##
## a window for T tests
##
myTtest <- function(x,y,mu=0) return( t.test(x=x,y=y,mu=mu) )
## create a tcltk frame and give it a title
frm <- tktoplevel()
tkwm.title(frm,"mapFuncToWidget for T tests")
## create the GUI window map of lm
mapFuncToWidget(myTtest,frm)
## ...you may close the window


##
##  a simple example
##
g <- function(a=1,b=rnorm) {cat("--g--");paste("g(a,b)=",a+b(a))}
h <- function(a=1,b=3,c=3) {cat("--h--");paste("h(a,b,c)=",a+b+c)}
## create a tcltk frame and give it a title
frm <- tktoplevel()
tkwm.title(frm,"mapFuncToWidget for g")
## create the GUI window map of g
mapFuncToWidget(g,frm)
## ...you may close the window


##
##  the ue of STORE
##
frm <- tktoplevel()
tkwm.title(frm,"another map")
```

```
mapFuncToWidget(g,frm,bttLabel="press me !!!")
## ... and if you do not close the window
mapFuncToWidget(h,frm,bttLabel="exec h !!!",STORE="fff")
## ...STORE should be added because g and h shares parameter
## names a and b
## now you may close the window


##
## a more involved example(see miniGUI widgets)
##
## some kernels
kernels <- list(
    gaussK=function(x) dnorm(x,0,1),
    SqK=function(x) ifelse( abs(x) <= 1 , 1 , 0 ),
    EpaK=function(x) ifelse( abs(x) <= 1 , 3/(4*sqrt(5))*(1 - x^2/5) , 0 ),
    TrianK=function(x) ifelse( abs(x) <= 1 , (1 - abs(x)) , 0 )
)
## how to compute the density at x
prDensEst <- function(x,dat,h,K) mean( K((x-dat)/h) ) / h
## the fucntion to appear in the frame
prDensCurvEst <- function(datos,
   bandwidth=miniGUIscale(from=.05,to=3,by=.05),
   Kernel=miniGUImenusel(c('"gaussK"','"SqK"','"EpaK"','"TrianK"','"QuartK"'))
)
{
  n <- length(datos)
  Kernel <- kernels[[Kernel]]
  f <-  function(x) sapply(x,function(x) prDensEst(x,datos,bandwidth,Kernel))
  xeval <- seq(min(datos),max(datos),len=100)
  ##plot pts in x axis
  plot(datos,rep(0,n),pch="+",ylim=c(0,1.25*max(f(xeval))),
       ylab="dens.",main="Density")
  curve(f,add=T)
  return(f)
}
formals(prDensCurvEst)$bandwidth <- quote(miniGUIscale(from=.05,to=3,by=.05))
formals(prDensCurvEst)$Kernel <- quote(
      miniGUImenusel(c('"gaussK"','"SqK"','"EpaK"','"TrianK"','"QuartK"'))
    )
frm <- tktoplevel()
tkwm.title(frm,"mapFuncToWidget for f")
aaa <- mapFuncToWidget(prDensCurvEst,frm)
```

`miniGUI` *Simple R GUI*

Description

Function to create a simple Graphical User Interface based on R\ functions based on `tcltk` package.

Usage

```
miniGUI(mainFrameFun=evalPlugin,opFuns=NULL,title="mini GUI",
        init=function(frm) {},WRAPFUN=TRUE)
evalPlugin(ev)
```

Arguments

mainFrameFun	A function to display (params are labels and entry fields) in the main GUI window.
opFuns	Named list of functions to add in the GUI menu **Ops**.
title	Main window GUI title.
init	Function to call before the GUI setup.
WRAPFUN	when TRUE, the default option, an automatic tcltk widget is built for the functions in opFuns.
ev	Expression to evaluate.

Details

miniGUI pops up a tcltk window widget with a menu bar containing two menus named **Basics** and **Ops** from which different functionality may be addressed during a miniGUI session. The menu **Basics** is used to request general purpose task during the session (like quitting), while **Ops** is usually where more specific tasks, those the GUI is devoted to and that are given in opFuns are grouped. When a menu item from **Ops** is selected a new window widget pops us reflecting all the parameters the function selected has, so that the user can fill text entries or set up the value for such parameters.

{init} can be used to add initialization and checking commands to the GUI. This function is executed before any other command.

When WRAPFUN is FALSE no tcltk widget is created for the functions in opFuns, allowing them to build their own widget. Do not use it unless functions encode its own tcltk, having into account the internals of the package to setup in a proper way the GUI for that specific function.

Value

miniGUI function returns nothing. Nevertheless, the results of the execution of the different functions called during the miniGUI session are available by means of the getMiniGUIans function, and also by means of the **GUI ans.** entry in the menu **Basics**.

Author(s)

Jorge Luis Ojeda Cabrera (<jojeda@unizar.es>).

See Also

miniGUI, makeWidgetCmd, tcltk.

Examples

```
require(tcltk)
##
## a simple example
##
fs <- list(
    f=function(a=1) {cat("--f--");paste("f(a)=",a)},
    g=function(a=1,b=rnorm) {cat("--g--");paste("g(a,b)=",a+b(a))},
    h=function(a=1,b=3,c=3) {cat("--h--");paste("h(a,b,c)=",a+b+c)}
  )
## evalPlugin is provided by the package
miniGUI(evalPlugin,opFuns=fs)

##
## an example with lm and glm functions
##
## create some data(in the global environment)
n <- 100
d <- data.frame(x=runif(n))
d$z <- 0.5 * rnorm(n)
d$y <- 2 * d$x + d$z
## makes a wrapper to access t.test
myTtest <- function(x,y,mu=0) return( t.test(x=x,y=y,mu=mu) )
## call miniGUI with myTtest, lm and glm functions
miniGUI(evalPlugin,opFuns=list("T test"=myTtest,"Lin. Mod."=lm,glm=glm))
## try menu "T test" only setting up x

##
## an example with WRAPFUN set to FALSE
##
gfs <- list()
for(i in names(fs))
{
   ## create GUI for fs[[i]] using miniGUIBase
   gfs[[i]] <- makeWidgetCmd(i,fs[[i]],miniGUIBase)
}
miniGUI(evalPlugin,opFuns=gfs,WRAPFUN=FALSE)
```

`miniGUIhelpers` *Utility functions*

Description

Some utility functions that are not exported

Usage

```
miniGUIgetFormals(f)
miniGUIcallEval(f,p,e)
miniGUIoutput(x,mess="\nminiGUI output: \n")
```

Arguments

f	An R\ function.
p	a list with all the parameters f requires.
e	environment where the parameters p of f are evaluated. By default .GlobalEnv .
x	An R\ object to print.
mess	a string with a brief message that is printed before x.

Details

These functions are internal functions that helps building the GUI map. addMenusCmd adds a menu to the main miniGUI frame. miniGUIgetFormals gets the parameter list of the function f filtering ellipsis. miniGUIcallEval performs the evaluation f when the arguments are set to those of p. miniGUIoutput is used to print out the result of the computation.

At the present moment, miniGUIeval is the same as the function miniGUIcallEval, while miniGUIEnvir is used to store miniGUI internal data. In particular, miniGUIEnvir$miniGUIans stores the result of the last computation made by a call to any of the miniGUI menu functions or any function widget created with by the functions makeWidgetCmd or mapFuncToWidget. On the other hand miniGUIEnvir$miniGUIData stores information and parameters required to compute function widgets. In ordet to do so, function storageName is used to avoid name collisons. The functions setMiniGUIData setMiniGUIans, getMiniGUIData, getMiniGUIans are used to set and get data from miniGUIEnvir$miniGUIData and miniGUIEnvir$miniGUIans resp..

Author(s)

Jorge Luis Ojeda Cabrera (<jojeda@unizar.es>).

See Also

miniGUI, makeWidgetCmd, mapFuncToWidget, tcltk.

---

miniGUIinputWidget *Entry widgets*

Description

Function that builds different input methods.

Usage

```
miniGUIentry(x,...)
miniGUIscale(from,to,by,...)
miniGUImenusel(xx,...)
```

Arguments

x	An R\ symbol, or numerical or character value. It can also be any R\ expression.
from,to,by	three numerical values.
xx	Any vector of mode numeric or character.
...	Any other sort of present or future parameters.

Details

These functions implements different input methods. In order to work these should appear as the default values of parameters in the definition of the function whose widget is to be built. In this way, the specification of the GUI input method for all the parameters can be done in a simple way by means of the definition if the function. It is worth mentioning that functions defined in this way can use parameters in the ordinary way if a value is provided for them. See the examples below.

In order to map a function onto a widget, `mapFuncToWidget` uses a `tkentry` that contains the character conversion of the default value for that parameter if there exist such a value, or that contains nothing there is no such a default value.

`...` stands for any other useful or future parameter. Currently you may use `NAME` to specify the parameter label in the input widgets

`miniGUIdefaultEntry` is the default input widget, at the moment a simple `tkentry`.

`miniGUIentry(x)` makes the `tkentry` related to the parameter to contain `x`. This widget is included as an example of the way widget can be added.

`miniGUIscale(from,to,by)` uses `tkscale` to show a slider that allows to input numerical values in the range `from`, `to` with an increment of `by`.

`miniGUImenusel(xx)` uses `ttkcombobox` (needs Tcl version 8.5 or later) to show a menu with entries `xx`, a character or numerical vector.

These functions and their implementation show how new input widget can be added in a simple way.

Value

All these functions returns an object `miniGUIwidget`, that is a list with at least the entry `widget` that should be a function and any other detail.

The function `widget` builds an entry widget using `tcltk` functions and should return it. This function should be defined having three parameters: `FRAME`, `STORE`, `VAR`. In short, the first one is used by the internal code to provide a `tcltk` parent frame, the second to provide a place where to save the value of the parameter and the third one is used to save the parameter name.

The implementation details may change in the future.

Author(s)

Jorge Luis Ojeda Cabrera (<jojeda@unizar.es>).

See Also

`miniGUI`, `makeWidgetCmd`, `tcltk`.

Examples

```
  require(tcltk)
  ##
  ##  simple example
  ##
  #   ...define a function
  h <- function(a=miniGUImenusel(c(1,5,10)),
b=miniGUIscale(from=5,to=10,by=2),
c=miniGUIentry(4),
d=miniGUImenusel(c("T","F")),
               e
)
  {
    cat("--h--");paste("h(a,b,c)=",d*(a+b+c))
  }
  ## building it
  hmm <- makeWidgetCmd("Hay !!",h)
  hmm()
  ##
  ##  another example
  ##
  ## create some data(in the global environment)
  n <- 100
  d <- data.frame(x=runif(n))
  d$z <- 0.5 * rnorm(n)
  d$y <- 2 * d$x + d$z
  ## def mylm method
  mylm <- lm
  formals(mylm)$method <- quote( miniGUImenusel(c('"qr"','"model.frame"')) )
  formals(mylm)$x <- quote( miniGUImenusel(c("FALSE","TRUE")) )
  ## add this stuff
  miniGUI(evalPlugin,opFuns=list(mylm=mylm,lm=lm))
```

Index