

# Package ‘gWidgetsRGtk2’

July 17, 2017

**Version** 0.0-84

**Title** Toolkit Implementation of gWidgets for RGtk2

**Author** Michael Lawrence, John Verzani

**Maintainer** John Verzani <jverzani@gmail.com>

**Depends** methods, grDevices, utils, graphics, RGtk2, gWidgets,  
cairoDevice

**Enhances** RGtk2Extras

**Description** Port of the gWidgets API to the RGtk2 toolkit.

**License** GPL (>= 2)

**LazyLoad** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-07-17 10:51:33 UTC

## R topics documented:

gWidgetsRGtk2-package . . . . .	1
as.gWidgetsRGtk2 . . . . .	4
gdfedit . . . . .	6
gWidgetsRGtk2-misc . . . . .	7
<b>Index</b>	<b>8</b>

---

gWidgetsRGtk2-package *Toolkit implementation of gWidgets for RGtk2*

---

## Description

Port of gWidgets API to RGtk2

## Details

This package allows the gWidgets API to use the RGtk2 package allowing the use of the GTK libraries within R. The documentation for the functions in this package are contained in the gWidgets package.

As gWidgets is meant to be multi-toolkit, this file documents differences from the API, as defined by the man pages of the **gWidgets** package.

### Containers:

If using a ggraphics device, one should call gwindow with the argument `visible=FALSE` then after the device is added call the `visible<-` method to show the window.

To access the underlying gtk container from a gframe object one uses `getToolkitWidget(obj)$getParent()`

The gnotebook changed handler has component `pageno` to indicate the newly selected page number, as the `svalue` method returns the page before the change.

### Widgets:

The gbutton constructor can not be called before **gWidgetsRGtk2** is loaded. This means that an initial call like `gbutton("label", cont=gwindow())` won't work. Instead, either load directly **gWidgetsRGtk2** (not just **gWidgets**) or create another widget, like a top-level window. Something similar is the case for the gdfedit widget.

The gradio widget can now have its items shortened or lengthened via `[<-`.

For the data frame viewer gtable when no filtering is requested – the default – the column headers can be clicked to sort the values. Setting the index to 0 will clear the selection.

In the data frame editor gdf the subset option only works if the column names have not been changed. One can suppress the creation of keyboard navigation and the right click popup on the column headers. The `hiddern` argument `diy` (for do it yourself) if left empty will place in both. A value of `"suppress.key"` or `"suppress.popup"` (or both) will suppress the respective handler.

The gaction constructor produces action objects. The `enabled<-` method can be used to set their sensitivity. The objects can be used with gbutton through the `action` argument, and in the lists defining menubars and toolbars. The `key.accel` argument (for assigning a keyboard accelerator) of the constructor is ignored for now. The `tooltip` is OS sensitive, as it depends on the event loop implementation.

The gtoolbar list can have components that are a) lists with a handler component, b) lists with a separator component, c) gaction instances d) gWidgets, in which case the widget appears in the toolbar. The latter is not portable to other gWidgets implementations.

The gvarbrowser constructor depends on a variable `knownTypes`. A default is provided in the package, but this can be overridden by a) providing a hidden argument `knownTypes` to the constructor or b) setting an option `knownTypes`. In each case this is a named list whose components are character

vectors listing classes of a similar nature. For example, the default value for `knownTypes` included "data sets" = `c("numeric", "logical", "factor", "character", "integer", "data.frame", "matrix", "list", "`

The function used to map a class to an icon is by default `getStockIconFromClass`. This can be changed by assigning a function to the option `gWidgetsStockIconFromClass`. This function should take a class and return a stock icon name. (The class passed is the first value only.)

The `gfile` constructor has the argument `multiple`, which if `TRUE` will allow for multiple selections of files. This feature should be merged into the `gWidgets` API, but for now is passed in via . . .

The `ggraphics` constructor provides a means to embed a graphics window inside a GUI. A right mouse popup allows one to copy the graphic to the clipboard or save it to a file. The different file types are limited by the function `gdkPixbufSave` whose manual page states that `jpg`, `png`, `ico` and `bmp` are permissible.

A few quirks exist.

1. Drawing a graphic too soon may result in a message about `plot.margins` too small. This comes from trying to draw the first graphic before the window is fully realized.

One workaround is to initially set the window not visible then when the GUI is done, make the window visible. That is, try: `w <- gwindow(visible=FALSE); ggraphics(cont=w); visible(w) <- TRUE; hist(`

2. When there are multiple devices, the standard means of setting a device via `dev.set` are supplemented by mouse handlers. Clicking in the graphics window sets the window as the current device.
3. The handler for `addHandlerClicked` responds to a mouse click. The components `x` and `y` give the coordinates in "usr" coordinates.
4. The handler for `addHandlerChanged` responds to the "rubber-banding" effect that comes from trying to trace out a rectangle in the graphic window. The components `x` and `y` give the coordinates in "usr" coordinates. (These each have two values.) The functions `grconvertX` and `grconvertY` can convert to other coordinate systems for you. See the `ggraphics` help page for an example of how this can be used to update a data frame.

The `gbasicdialog` constructor can be used both ways. The hidden argument `buttons` can take values `ok`, `yes`, `cancel`, `close`, `no`, with a default of `c("ok", "cancel")`.

### Methods:

The `font` method is not implemented.

For widgets which allow markup (`gframe`, `glabel`) PANGO markup is used. This is not HTML, but is similar to basic HTML.

### **gWidgetsRGtk2 and the RGtk2 package:**

The **RGtk2** package is imported only so its namespace, which is large, is not loaded by default. To access its functions, load the package.

The **RGtk2** package and **gWidgetsRGtk2** can be used together in the following ways. First, an **RGtk2** object can be added to a **gWidgetsRGtk2** through the add method of the container. This works for most objects. If you find one that doesn't work, simply place it inside a gtkHBox container, then add that container. Second, a **gWidgetsRGtk2** object can be added to a **RGtk2** container by adding the return value of the getToolkitWidget method of the object. Again, this should work, but if not, the **gWidgetsRGtk2** can be added to a ggroup container first. In either case, the **gWidgetsRGtk2** object should not be previously attached to a container, so in particular the constructor should be called with its container argument as NULL (the default).

### Author(s)

Michael Lawrence, John Verzani

Maintainer: John Verzani <gwidgetsrgtk@gmail.com>

### See Also

gWidgets

---

as.gWidgetsRGtk2

*Coerce an RGtk2 object into a gWidgetsRGtk2 object*

---

### Description

This function coerces an RGtk2 object into a gWidgetsRGtk2 object, thereby allowing most of the methods to work on the new object.

### Usage

```
as.gWidgetsRGtk2(widget, ...)
```

### Arguments

widget	An object of class RGtkObject
...	Ignored here

### Details

Many RGtk2 widgets can be coerced into gWidgetsRGtk2 objects. This allows the method of gWidgets to be called. The example shows how one can use glade to layout a dialog, and use gWidget methods for the handlers.

### Value

Returns a gWidgetsRGtk2 object. (This is not a gWidgets object, so there may be some oddities

**Examples**

```

## Not run:
## This requires glade libraries to be installed before compiling RGtk2
options("guiToolkit"="RGtk2")
library(RGtk2)
library(gWidgets)
library(gWidgetsRGtk2)

gladeFile <- system.file("examples/t.test.glade",package="gWidgetsRGtk2")
GUI <- gladeXMLNew("t.test.glade")

w <- GUI$GetWidget("window1")
w$Show() # show
win <- as.gWidgetsRGtk2(w)

gladeXMLGetWidgetNames <- function(obj) {
  sapply(obj$GetWidgetPrefix(""),gladeGetWidgetName)
}

gladeXMLGetgWidgetsRGtk2 <- function(obj) {
  nms <- obj$GetWidgetNames()
  widgets <- sapply(nms, function(i) obj$GetWidget(i))
  widgets <- sapply(widgets, as.gWidgetsRGtk2)
  return(widgets)
}

l <- GUI$GetgWidgetsRGtk2()

## val names have similar form
valNames <- grep("Val$",GUI$GetWidgetNames())
defHandler <- function(...) {
  lst <- list()
  args <- c("x","y", "mu","alt","var.equal","paired","conf.level")
  for(i in args) {
    key <- paste(i,"Val",sep="")
    widget <- l[[key]]
    val <- svalue(widget)
    if(!is.null(val) && val != "")
      lst[[i]] <- val
  }
  if(!is.null(lst$x)) {
    cmd <- "t.test("
    argList <- c()
    for(i in names(lst)) {
      argList <- c(argList,paste(i,"=",lst[[i]], sep=""))
    }
    cmd <- paste(cmd, paste(argList,collapse=" ",")",sep="")
    print(cmd)
  }
}
## Add handler to each widget
sapply(valNames, function(i) addHandlerChanged(l[[i]],handler=defHandler))

```

```
## put handler on dismiss button
addHandlerChanged(l[['dismiss']], handler = function(h,...) dispose(win))

## End(Not run)
```

---

gdfedit

*gWidgets interface for RGtk2Extras data editor widget*


---

## Description

An alternative widget for editing a data frame using the RGtk2DfEdit package

## Usage

```
gdfedit(items = NULL,
        name = paste(deparse(substitute(items)), "1", sep="."),
        container = NULL, ..., toolkit = guiToolkit())
```

## Arguments

items	data frame to be edited
name	Name of data frame to save value to
container	An optional container to attach widget to
...	Can be used to override default colors.
toolkit	Which GUI toolkit to use

## Details

The gdf widget is used for editing data frames, but does not have the most natural keyboard handling. The **RGtk2Extras** package by Tom Taverner provides a more powerful and easier to use interface for editing data frame, and this wraps that widget into gWidgetsRGtk2.

The addHandlerColumnClicked function can be used to add a handler to the event when a column header is clicked. The component of the first argument column.no contains the column number.

This widget is a bit different from the others as it is not imported from gWidgets. As such, it won't exist until this **gWidgetsRGtk2** package is loaded. In practical terms, you need to realize a widget before this one can be realized.

## See Also

gtable

**Examples**

```
## Not run:
w<-gwindow()
g<-gggroup(cont=w)
df<-gdfedit(iris, cont=g)

## check names
names(df)
names(df)[1]<-"new"
rownames(df)
colnames(df)

## check [
df[,]
df[1,]
df[,1]

## no [<- function
## check dim stuff
dim(df)
length(df)

## handler

addHandlerColumnClicked(df, handler<-function(h,...) {
  print(h$column.no)
})

## End(Not run)
```

---

gWidgetsRGtk2-misc      *Miscellaneous functions in gWidgetsRGtk*

---

**Description**

These functions are hardly worth documenting. They are used by pmg, but are not part of the gWidgets API, nor meant for general consumption.

# Index

## \*Topic **interface**

- as.gWidgetsRGtk2, 4
- gdfedit, 6
- gWidgetsRGtk2-misc, 7

## \*Topic **package**

- gWidgetsRGtk2-package, 1
- .gdfedit (gdfedit), 6
- .stockIconFromClass,guiWidgetsToolkitRGtk2-method  
(gWidgetsRGtk2-misc), 7
- .stockIconFromClass-methods  
(gWidgetsRGtk2-misc), 7
- .stockIconFromObject,guiWidgetsToolkitRGtk2-method  
(gWidgetsRGtk2-misc), 7
- .stockIconFromObject-methods  
(gWidgetsRGtk2-misc), 7

as.gWidgetsRGtk2, 4

gdfedit, 6

gWidgetsRGtk2 (gWidgetsRGtk2-package), 1

gWidgetsRGtk2-misc, 7

gWidgetsRGtk2-package, 1

Paste (gWidgetsRGtk2-misc), 7

rpel (gWidgetsRGtk2-misc), 7

stockIconFromClass  
(gWidgetsRGtk2-misc), 7

stockIconFromObject  
(gWidgetsRGtk2-misc), 7

str1 (gWidgetsRGtk2-misc), 7

str2 (gWidgetsRGtk2-misc), 7

stripWhiteSpace (gWidgetsRGtk2-misc), 7

Timestamp (gWidgetsRGtk2-misc), 7

Timestamp<- (gWidgetsRGtk2-misc), 7

untaintName (gWidgetsRGtk2-misc), 7