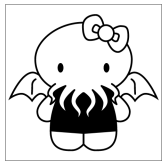
# **sexy.rgtk**: a package for programming **RGtk2** GUI in a user-friendly manner

Damien Leroux[1], Nathalie Villa-Vialaneix[1,2]

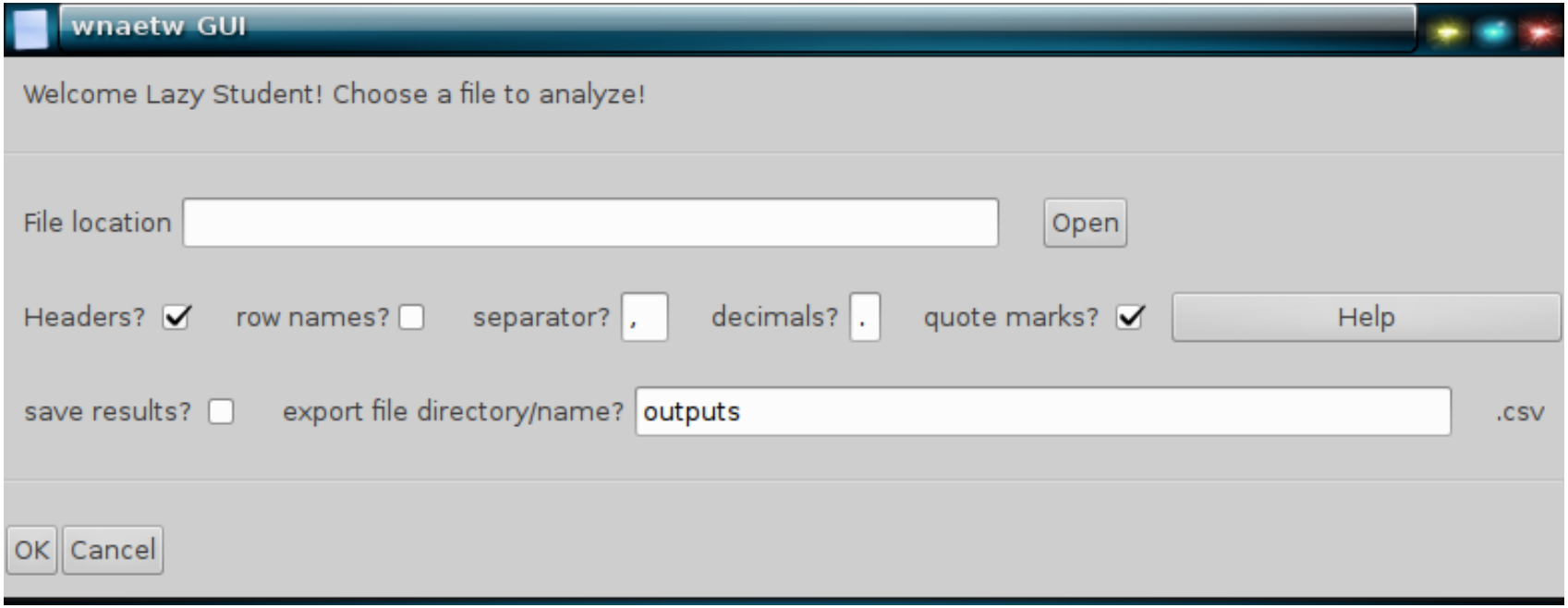
Rencontres R 2013, June 28th

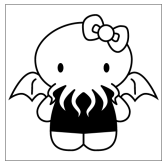
1 

2 

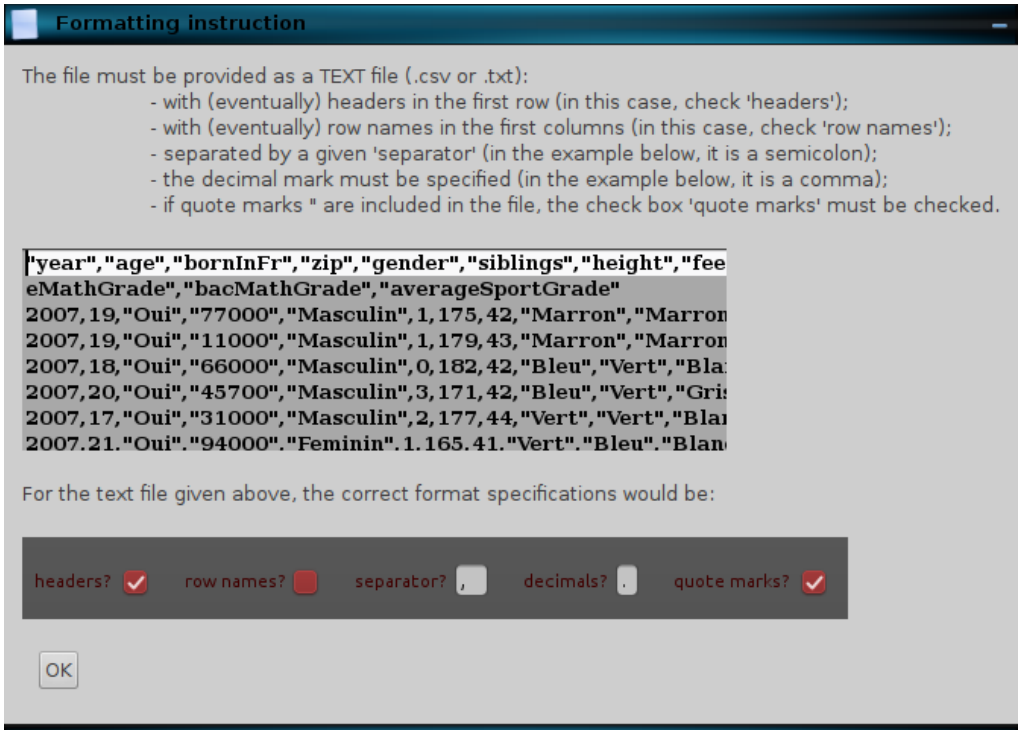



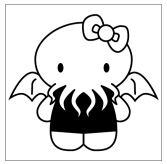
# Typical GUI (demo)



# Typical GUI (demo)



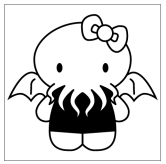



# Typical GUI (demo)

**Here we are, lazy student! Please find below the main statistics:**

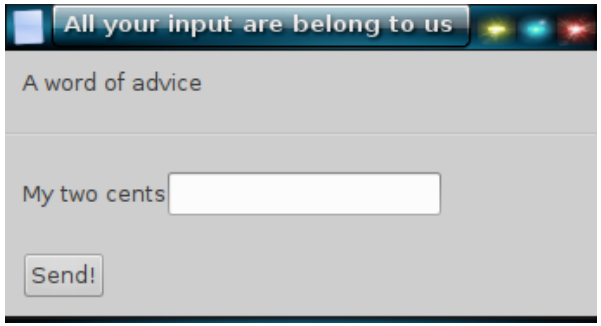
| | year | age | zip | siblings | height | feetSize | dptCode | averageMathGrade | bacMathGrade | averageSportGrade |
|---|---|---|---|---|---|---|---|---|---|---|
| mean | 2007.94 | 19.09 | 42004.69 | 1.8 | 172.76 | 41.32 | 46.48 | 7.56 | 6.6 | 5.89 |
| median | 2007 | 19 | 33500 | 2 | 174 | 41.5 | 34 | 6 | 7 | 6 |
| min | 2007 | 17 | 11000 | 0 | 158 | 37 | 9 | 2 | 1 | 1 |
| max | 2010 | 26 | 97200 | 4 | 190 | 47 | 97 | 15 | 15 | 11 |
| range | 3 | 9 | 86200 | 4 | 32 | 10 | 88 | 13 | 14 | 10 |
| sd | 1 | 2 | 27836 | 1 | 8 | 2 | 22 | 4 | 4 | 3 |
| kurtosis | -1.36 | 5.37 | -0.99 | -0.17 | -0.58 | -0.67 | -0.47 | -1.34 | -0.91 | -0.77 |
| skewness | 0.8 | 2.09 | 0.53 | 0.69 | 0.02 | 0.35 | 0.65 | 0.38 | 0.32 | 0.24 |
| variation | 0 | 0.09 | 0.66 | 0.6 | 0.05 | 0.06 | 0.47 | 0.57 | 0.6 | 0.47 |
| Q1 | 2007 | 18 | 16477.5 | 1 | 167.25 | 39 | 31 | 4 | 3 | 4 |
| Q3 | 2010 | 20 | 66000 | 2 | 179 | 42.75 | 66 | 12 | 9.5 | 8 |
| gini | 0 | 0.04 | 0.36 | 0.31 | 0.03 | 0.03 | 0.24 | 0.32 | 0.33 | 0.26 |

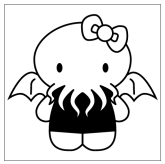
OK



# Conceptual GUI structure

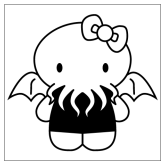
- Window — title:"All your input are belong to us"
  - Frame
    - Vertical container
      - Label — text:"A word of advice"
      - Separator
      - Horizontal container
        - Label — text:"My two cents"
        - Text input
      - Button — label:"Send!" — click:sendData()



# **RGtk2** Actual code (simple version)

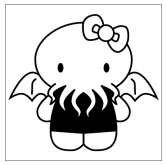
```
library(RGtk2)

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,
    "All your input are belong to us")
w$add(vc)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

gSignalConnect(but, "clicked", sendData)
```



# **RGtk2** Actual code (simple version)

```
library(RGtk2)

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,
    "All your input are belong to us")
w$add(vc)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

gSignalConnect(but, "clicked", sendData)
```

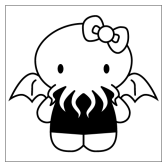
Widget construction



# **RGtk2** Actual code (simple version)

```
library(RGtk2)

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,    "All your input are belong to us")
w$add(vc)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

gSignalConnect(but, "clicked", sendData)
```

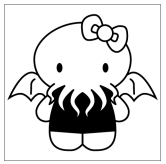
Widget configuration



# **RGtk2** Actual code (simple version)

```
library(RGtk2)

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,
    "All your input are belong to us")
w$add(vc)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

gSignalConnect(but, "clicked", sendData)
```

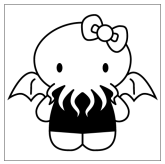
Widget tree creation



# **RGtk2** Actual code (simple version)

```
library(RGtk2)

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,
    "All your input are belong to us")
w$add(vc)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

gSignalConnect(but, "clicked", sendData)
```

Assign behaviour to events



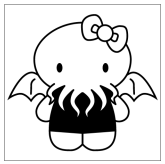
# **sexy.rgtk** manifesto

## **sexy.rgtk** aims at

- Make the widget tree creation more natural
- Make the code more readable
- Make the basic and common operations easier and more concise

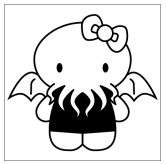
## While retaining the full power of **RGtk2**

- **sexy.rgtk** is a layer on top of **RGtk2**
- plain **RGtk2** code can be mixed with **sexy.rgtk** code



# **sexy.rgtk** equivalent code

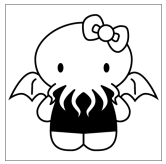
```
library(sexy.rgtk)

sendData <- function(...) {
    print(ws$entry$text)
}

ws <- Window(
    title="All your input are belong to us",
    contents=
        VBox(contents=list(
            Label("A word of advice"),
            HSeparator(),
            HBox(contents=list(
                Label("My two cents"),
                Entry(use.name="entry"))),
            Button(label="Send!",
                   on("clicked", sendData)))))
```



# **sexy.rgtk** equivalent code (comparison)

```
library(sexy.rgtk)

sendData <- function(...) {
    print(ws$entry$text)
}

ws <- Window(
    title="All your input are belong to us",
    contents=
        VBox(contents=list(
            Label("A word of advice"),
            HSeparator(),
            HBox(contents=list(
                Label("My two cents"),
                Entry(use.name="entry"))),
            Button(label="Send!",
                   on("clicked", sendData)))))
```

```
library(RGtk2)

sendData <- function(...) {
    print(gtkEntryGetText(entry))
}

vc <- gtkVBoxNew()
vc$packStart(gtkLabelNew("A word of advice"))
vc$packStart(gtkHSeparatorNew())
hc <- gtkHBoxNew()
hc$packStart(gtkLabelNew("My two cents"))
entry <- gtkEntryNew()
hc$packStart(entry)
vc$packStart(hc)
but <- gtkButtonNew()
gtkButtonSetLabel(but, "Send!")
vc$packStart(but)
w <- gtkWindowNew()
gtkWindowSetTitle(w,
    "All your input are belong to us")
w$add(vc)

gSignalConnect(but, "clicked", sendData)
```



# How does it work ?

## Using **RGtk2** (almost consistent) semantics

- Constructors

  gtkLabelNew

  *section* *class* *action (simple constructor)*

  gtkLabelNewWithMnemonic

  *section* *class* *action (fancy constructor)*

- Modificators

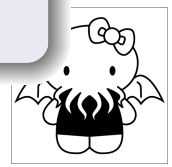
  gtkLabelSetText

  *section* *class* *!* *property*

- Accessors

  gtkLabelGetText

  *section* *class* *!* *property*



# How does it work ?

## **sexy.rgtk** covers all classes, getters, and setters in the **gtk** section

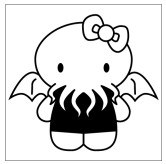
```
library(RGtk2)
l <- gtkLabelNew()
gtkLabelSetText(l, "pouet")
print(gtkLabelGetText(l))
```

```
library(sexy.rgtk)
l <- Label()
l$text <- "pouet"
print(l$text)
```

## **sexy.rgtk** combines the construction and configuration steps

```
library(RGtk2)
l <- gtkLabelNew()
gtkLabelSetText(l, "pouet")
gtkLabelSetAngle(l, 23)
```

```
library(sexy.rgtk)


l <- Label(text="pouet", angle=23)
```



# How does it work ?

## **sexy.rgtk** makes the widget hierarchy more explicit

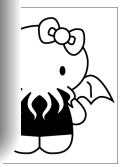
```
library(RGtk2)
l <- gtkLabelNew()
w <- gtkWindowNew()
gtkLabelSetText(l, "Foobar")
gtkWindowSetTitle(w, "Wibble")
w$add(l)
```

```
library(sexy.rgtk)


w <- Window(
        title="Wibble",
        contents=Label(text="Foobar"))
```

## Signal connections are also more better *(sic)* with **sexy.rgtk**

```
library(RGtk2)

callback <- function(widget, data) {
    cat("Pouet.\n")
}

b <- gtkButtonNew()
gtkButtonSetLabel(b, "Fire")
gSignalConnect(b, "clicked", callback)
# or
connectSignal(b, "clicked", callback)
# Oh wait, this breaks the naming rules
```

```
library(sexy.rgtk)

callback <- function(widget, data) {
    cat("Pouet.\n")
}

b <- Button(label="Fire",
            on("clicked",
               run=cat("Pouet.\n")))
# or
b <- Button(label="Fire",
            on("clicked", callback))
```



# Feature summary

## All the gtk* functions are wrapped in an easier interface.

```
library(RGtk2)
```

```
library(sexy.rgtk)
```

- Unified constructors

```
bu <- gtkButtonNew()
bu <- gtkButtonNewFromStock(...)
```

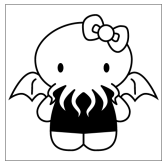
```
bu <- Button()
bu <- Button(from.stock=list(...))
```

- Natural getters and setters

```
gtkButtonSetFocusOnClick(bu, T)
gtkButtonGetLabel(bu, "...")
```

```
bu$focus.on.click <- T
bu$label
```

- Coherent use of **names()** to include getters and setters



# Feature summary

## All the gtk* functions are wrapped in an easier interface.

```
library(RGtk2)
```

```
library(sexy.rgtk)
```

- Everything can be actually done via the constructors

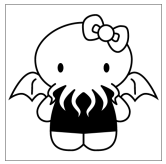
```
w <- gtkWindowNew()

l <- gtkLabelNew()
gtkLabelSetText(l, "1234567890")
gtkLabelSelectRegion(l, 2, 5)

w$add(l)
```

```
w <- Window(contents=
        Label(use.name="lab",
            text="1234567890",
            apply=list(
                SelectRegion=
                    list(2, 5)
            ))
```

- Widget access through the hierarchy with **use.name=**

```
gtkLabelSetText(l, "toto pouet")
```

```
w$lab <- "toto pouet"
```



# There's more!

## **sexy.rgtk** also tries to address some common patterns

- Associate a label to an input widget

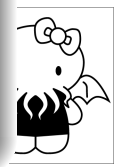
```
library(RGtk2)
l <- gtkLabelNew()
gtkLabelSetText(l, "Input")
i <- gtkEntryNew()
hbox <- gtkHBoxNew()
hbox$packStart(l)
hbox$packStart(i)
# to use a mnemonic:
l <- gtkLabelNewWithMnemonic()
gtkLabelSetText(l, "_Input")
gtkLabelSetMnemonicWidget(l, i)
```

```
library(sexy.rgtk)

# also supports vertical alignment
# and label on either side of the
# widget
x <- LabeledWidget("Input",
                   Entry())
# to use a mnemonic:
x <- LabeledWidget("_Input",
                   Entry(),
                   mnemonic=T)
```

- Rows or columns of widgets (VBoxes inside HBox or vice versa)

```
library(RGtk2)
vb <- gtkVBoxNew()
hb <- gtkHBoxNew()
hb$packStart(gtkLabelNew())
hb$packStart(gtkLabelNew())
vb$packStart(hb)
hb <- gtkHBoxNew()
hb$packStart(gtkEntryNew())
vb$packStart(hb)
```

```
library(sexy.rgtk)

# of course it can also become
# very verbose when you actually
# configure the widgets


vb <- Rows(Label(), Label(), br,
           Entry())
```

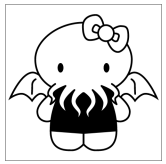


# There's more!

## Last but not least: displaying a data.frame

```
library(RGtk2)
# I'm sorry this is way too gory.
# In short:
# - create an rGtkDataFrame(
#                     my.data.frame)
#   (this is the datastore)
# - create renderer function(s)
#   function(col, colrenderer,
#            model, iter, data)
# - create a GtkTreeView
# - associate the datastore with the
#   tree view
# - declare each column to display
#   and assign a renderer to each
#   of them
```

```
library(sexy.rgtk)

# Supports specific renderers only if
# required, with a default to
# plaintext as seen in the demo.
# By default, all columns are to be
# displayed.
# The interface to the renderers is
# simplified.

# Also supports row names.

df.widget <- DataFrame(my.data.frame)

# et voila!
```



# References

- Build a simple GUI with RGtk2
  http://tuxette.nathalievilla.org/?p=866&lang=en
- RGtk2: R bindings for Gtk 2.8.0 and above
  http://cran.r-project.org/web/packages/RGtk2/index.html
- Mulcyber: sexy.rgtk: Project home https://mulcyber.toulouse.inra.fr/projects/sexy-rgtk/