Short overview of **Shiny** package: Building an interactive web application straight from R

Rencontres R Toulouse

Marion Aguirrebengoa

May 6th 2019

Outline

Since December 2016

CBI's BioInfomatic Genomic Analysis core facility (big-A)

CBI Toulouse
Centre de Biologie Intégrative

Activities :

- Process NGS raw Data
- Analyze data, Visualization, ...
- Train people and supervise students
- Automatize tasks

Core Facility

Needs:

- Share analysis
- Automatize some tasks (figures, tests, ...)

Core Facility

Needs:

- Share analysis
- Automatize some tasks (figures, tests, ...)

Previous solution ?

Core Facility

Needs:

- Share analysis
- Automatize some tasks (figures, tests, ...)

Previous solution ?

Rmarkdown (Rmd) BUT...

Core Facility

Needs:

- Share analysis
- Automatize some tasks (figures, tests, ...)

Previous solution ?

Rmarkdown (Rmd) BUT...

- Not really interactive
- R knowledge
- OS specificities

Core Facility

Needs:

- Share analysis
- Automatize some tasks (figures, tests, ...)

Previous solution ?

Rmarkdown (Rmd) BUT...

- Not really interactive
- R knowledge
- OS specificities

$\Rightarrow$ **Shiny !**

Core Facility

Needs:

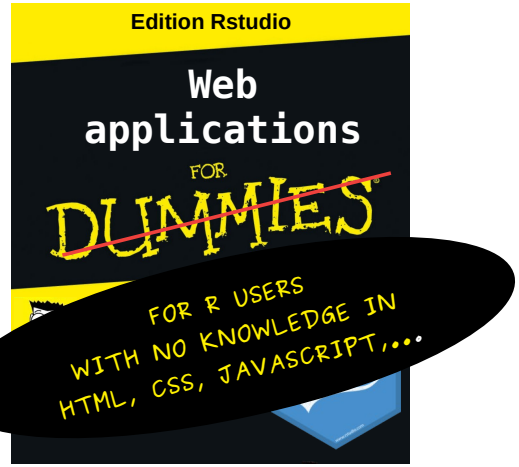
- Share analysis
- Automatize some tasks (figures, tests, ...)

Previous solution ?

Rmarkdown (Rmd) BUT...

- Not really interactive
- R knowledge
- OS specificities

$\Rightarrow$ **Shiny !**



Created in 2012 by R-Studio teams (W.Chang et al.)

Goal

Build interactive web applications or dashboards using R code

Created in 2012 by R-Studio teams (W.Chang et al.)

Goal

Build interactive web applications or dashboards using R code

Advantages :

- Power of R (Advanced statistics capabilities, Visualizations, Reproducibility, Open Source & Free, ...)
- No web development knowledge required

Created in 2012 by R-Studio teams (W.Chang et al.)

Goal

Build interactive web applications or dashboards using R code

Advantages :

- Power of R (Advanced statistics capabilities, Visualizations, Reproducibility, Open Source & Free, ...)
- No web development knowledge required

"Makes it incredibly easy to build interactive web applications with R. [...] make it possible to build beautiful, responsive, and powerful applications with minimal effort."

What does it look like?

Gallery Shiny
http://shiny.rstudio.com/gallery/

Gallery of R Web Apps
http://www.showmeshiny.com/

How it works

```
install.packages("shiny")
library(shiny)
```

How it works

```
install.packages("shiny")
library(shiny)
```

Very easy to start in R-Studio IDE

How it works

Shiny apps composed of 2 parts/files :

- ui : User Interface script for App layout
- server : Server script with R instructions

```
library(shiny)
ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```


shinyApp() function creates a "ShinyApp object" from the pair Ui/Server

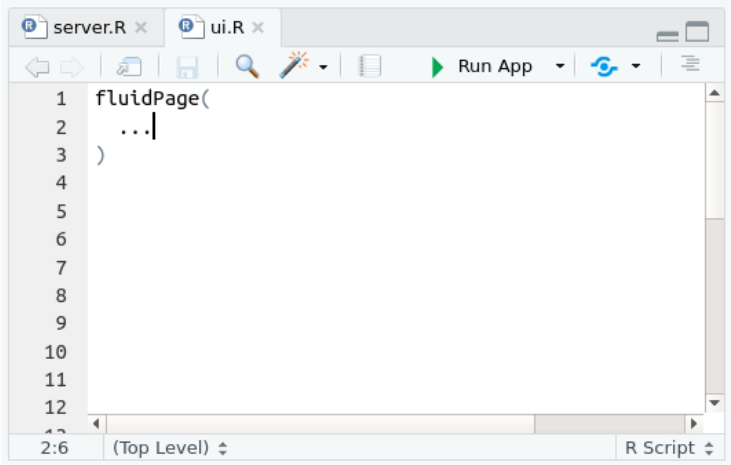
All in 1 "app.R" file

```
library(shiny)
ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

All in 1 "app.R" file

```
library(shiny)
ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

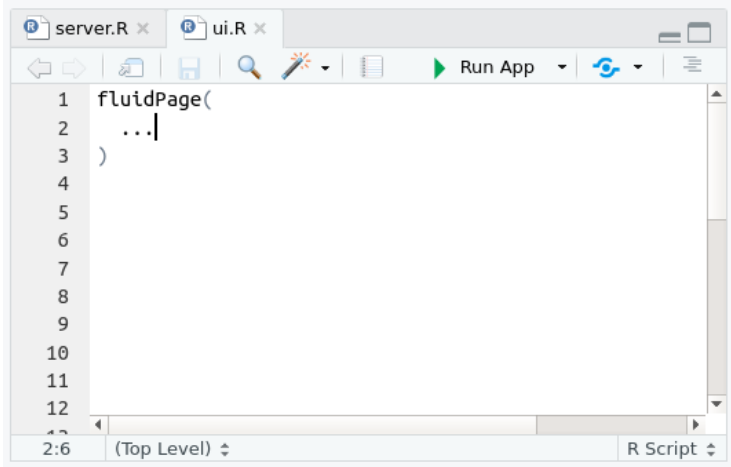

2 file in the same directory newdir/ : "ui.R" & "server.R"



All in 1 "app.R" file

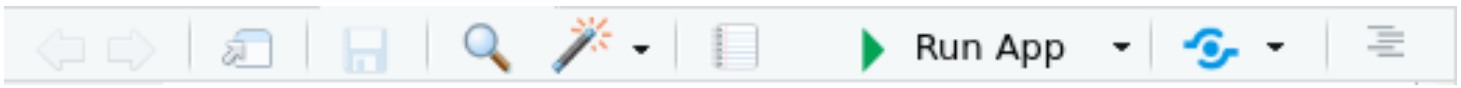
```
library(shiny)
ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```


2 file in the same directory newdir/ : "ui.R" & "server.R"



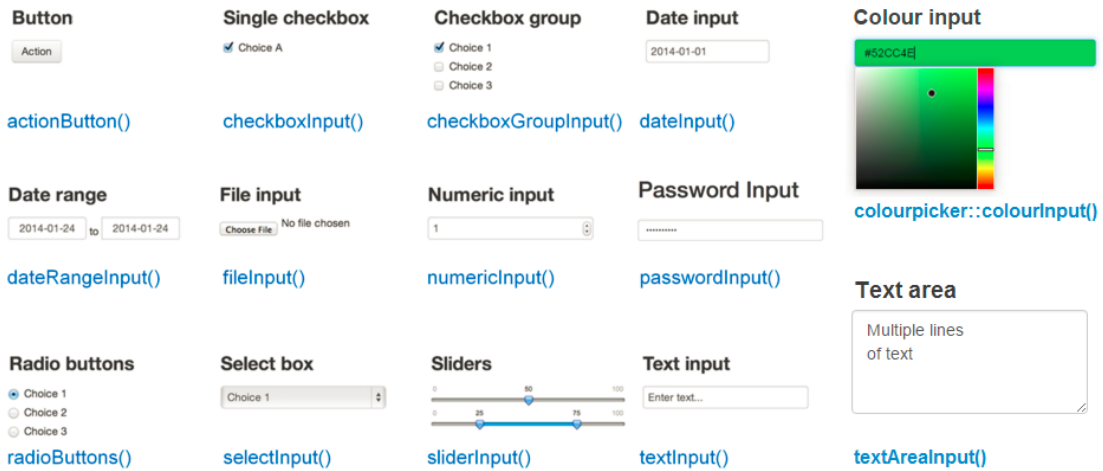
app <- shinyApp(ui, server)
runApp(app)

runApp("newdir")



Ui.R - Client file

1. Layout : plain and formatted text, HTML elements, title, theme layout, ...
2. Inputs (widgets)



3. Placeholders for outputs

Remark : All UI functions are simply HTML wrappers

Server.R

1. Acquire input
2. Run all analysis
3. Create outputs

Server.R

1. Acquire input
2. Run all analysis
3. Create outputs

function	creates
`renderDataTable()`	An interactive table (from a data frame, matrix, or other table-like structure)
`renderImage()`	An image (saved as a link to a source file)
`renderPlot()`	A plot
`renderPrint()`	A code block of printed output
`renderTable()`	A table (from a data frame, matrix, or other table-like structure)
`renderText()`	A character string
`renderUI()`	a Shiny UI element

Rules :

1) Call outputs and inputs
2) Use $ to call input
3) Create output with *Render()* function (to generate HTML tags) and use $ to call it

Shiny Reactivity

Concept of reactive programming

Change input => run R code => Change output

3 types of reactive :

- reactive source : choose by user via interface
- reactive endpoint : display plots in user navigator
- reactive conductor : elements in between

Application deployment

1. shinyapps.io platform, Cloud hosted by R-Studio :
   - + : Free, Easy to use, Secure & Scalable
   - - : Max 5 apps & 25 hours/month or less

Application deployment

1. shinyapps.io platform, Cloud hosted by R-Studio :
   - + : Free, Easy to use, Secure & Scalable
   - - : Max 5 apps & 25 hours/month or less
2. Configure you own Shiny server
   - + :
     - Unlimited number of apps
     - Provide a unique URL for each application
     - Automatically start and stop applications
   - - :
     - Specific skills to maintain shiny server
     - Free only with few users
     - Restrict access not available with free version

To go further : Combining Shiny & R Markdown

RMarkdown interactive document

- add "runtime: shiny" to the documents YAML header
- add Shiny widgets and render functions to the R code chunks

Rmarkdown compile the document into a reactive Shiny app (html doc with reactive components)

```
1 ---
2 title: "shiny-Rmarkdown"
3 author: "M. Aguirrebengoa"
4 date: "6 mai 2019"
5 output: html_document
6 runtime: shiny
7 ---
```

"runtime:shiny" to notify RStudio, change "Knit" to "Run Document"

To go further : Some nice functions & packages

- Some functions to improve your app :
  - *DataTable()* : Javascript table with option
  - *conditionalPanel()* [ui.R] : display elements within condition
  - *reactiveValues()* [server.R] : store value in a list
  - *updateXXXInput()* [server.R] : Change input values
  - ...
- *rCharts* : Packages for dynamics JavaScript plots in R
- *shinythemes* : Predefine shiny themes
- *shinydashboard* : Package dashbord design
- Improve Shiny apps with html widgets programmed in R, or directly HTML, CSS ou JavaScript widgets
- ...

References I

W.Chang, J.Cheng, JJ.Allaire, Y.Xie & J.McPherson
Shiny: Web Application Framework for R.
R package v.1.2.0, 2018.
CRAN.R-project.org/package=shiny

Colin from Data Scientist & R Hacker
A la dcouverte de Shiny.
ThinkR, 2018.
thinkr.fr/a-decouverte-de-shiny/

D.Attali
Building Shiny apps an interactive tutorial.
R-bloggers, 2015.
www.r-bloggers.com/building-shiny-apps-an-interactive-tutorial/

A. Deschamps
R pour le web : Shiny 101.
DACTA, 2016.
www.dacta.fr/blog/r-shiny.html

References II

RStudio Community
Reactivity - An overview.
Shiny from R Studio, 2017.
shiny.rstudio.com/articles/reactivity-overview.html

G. Grolemund
Introduction to R Markdown.
Shiny from R Studio, 2014.
shiny.rstudio.com/articles/rmarkdown.html

M. Edmondson & T. Wilson
RMarkdown and Shiny.
Digital Analysts: R and staTISTICS (DARTISTICS), 2017.
www.dartistics.com/rmarkdown-shiny.html

Conclusion

Shiny

- To share your analysis in an interactive way
- Easy to use in R-Studio IDE
- R power, free and open-source
- Support concepts of reproducibility

Could easily replace classical R scripts or Rmd Reports

Conclusion

Shiny

- To share your analysis in an interactive way
- Easy to use in R-Studio IDE
- R power, free and open-source
- Support concepts of reproducibility

Could easily replace classical R scripts or Rmd Reports

Want to try it ?

Great tutorial done by R-Studio
https://shiny.rstudio.com/tutorial/