International Biometric Conference IBC

Interactive Data Visualization in R with Shiny, Example of *PlantBreedGame*, A serious game to Teach Genomic Selection

Julien DIOT, The University of Tokyo, Lab. of Biometry and Bioinformatics

2024-12-10, IBC 2024, Atlanta USA

About those slides

These slides are build with **Rmarkdown** and **R-shiny** to enable some **interactivity features**.

The **PDF print** of thoses slides **may not be rendered correctly** and the **interactivity features will be not available**.

You can find the **source code** of those slides and how to start the presentation at this GitHub repository: https://github.com/juliendiot42/IBC_2024_presentation (https://github.com/juliendiot42/IBC_2024_presentation)

More information about Rmarkdown and R-shiny presentation:

- https://bookdown.org/yihui/rmarkdown/ioslides-presentation.html#presenter-mode (https://bookdown.org/yihui/rmarkdown/ioslides-presentation.html#presenter-mode)
- https://bookdown.org/yihui/rmarkdown/shiny-documents.html (https://bookdown.org/yihui/rmarkdown/shiny-documents.html)

Overview

The aim of this presentation is to encourage you to try `Shiny` by showing you simple example that you can reproduce and a more complex example (*PlantBreedGame*) to show you `Shiny`'s capabilities.

- What is `Shiny`
- Simple `Shiny` examples (with `R`)
- Shiny application's structure
- Example of *PlantBreedGame*
- R-Shiny's Pro and con
- References

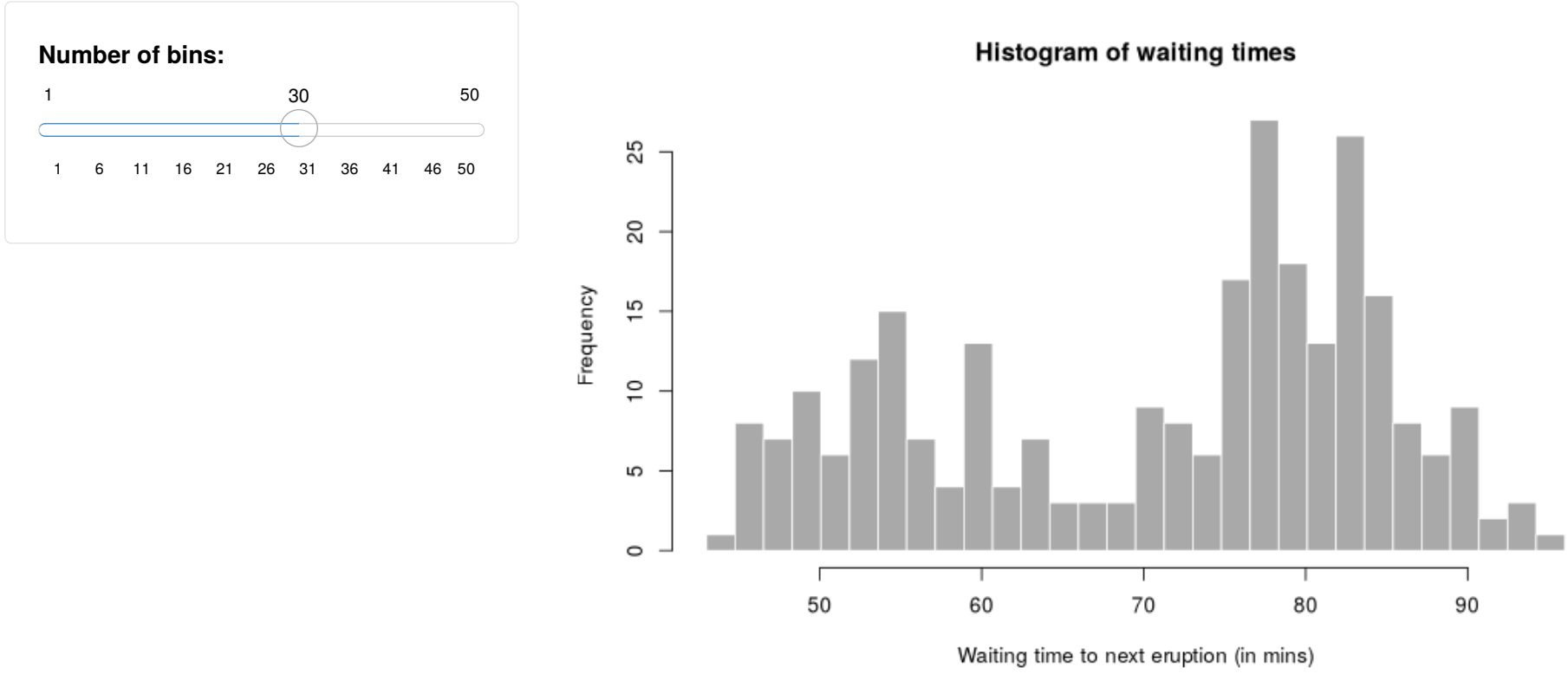
What is Shiny

- Packages for building *"Easy **web apps for data science** without the compromises"* from R and `python`
- Developed by **Posit** (formly R-Studio)
- **R-Package** github.com/rstudio/shiny (https://github.com/rstudio/shiny)
  - Initial commit (https://github.com/rstudio/shiny/tree/e28b3da1badfecb34235e74a43aac4e8da1641bc) on Jun 21, 2012 by Joe Chang @jcheng5 (https://github.com/jcheng5)
  - `v1.0.0` (https://github.com/rstudio/shiny/releases/tag/v1.0.0) published on Jan 14, 2017
  - ~88 contributors, ~5.4K ⭐, CRAN 

(https://r-pkg.org/pkg/shiny)
- **Python package** github.com/posit-dev/py-shiny (https://github.com/posit-dev/py-shiny)
  - first commit (https://github.com/posit-dev/py-shiny/tree/5f6905833822301a59757247140dce3f4d6cb339) on Jul 27, 2021 by Winston Chang @wch (https://github.com/wch)
  - `v1.0.0` (https://github.com/posit-dev/py-shiny/releases/tag/v1.0.0)

Shiny example

- Basic example app from R-Studio: `File → New File → Shiny Web App ...`



Shiny example, code

- Shiny app are structured around 2 components **`ui` and `server`**:

`ui`

- defines **the visual** aspect of the application:
  - The general layout
  - The content
  - The inputs
  - The output types and location

`server`

- defines **the logic to generate the outpus** from the inputs

- `shiny` handles the communication between `ui` and `server`
- The usage of inputs/outputs **"ids"** ensures **the links** between `ui` and `server`

Shiny example, code

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Then we can run start the app with `shinyApp(ui = ui, server = server)`

Shiny example, code

We define the page structure: 1 page (`fluidPage`) with a title (`titlePanel`), and a “sidebar” and “main area” layout (`sidebarLayout`, `sidebarPanel`, `mainPanel`)

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Shiny example, code

We define the inputs: here **a slider** with `sliderInput` with **id `bins`**

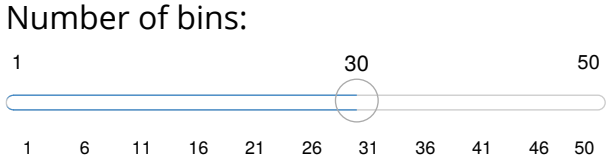
```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins

  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```



Shiny example, code

We define the outputs: here **a basic R plot** with `plotOutput` with **id `distPlot`**

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Shiny example, code

We specify the logic for the outputs `distPlot`.

- `plotOutput` works in combination with `renderPlot`

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Shiny example, code

We access the value of the input with `input$<id>` here `input$bins`

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(

      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
                length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Shiny example, code

Inside the “render functions” it is standanrd R code that should return the correct object

```
ui ← fluidPage(
  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(

      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
)
```

```
server ← function(input, output) {
  output$distPlot ← renderPlot({
    # generate bins based on input$bins from ui.R
    x    ← faithful[, 2]
    bins ← seq(min(x), max(x),
               length.out = input$bins + 1)

    # draw the histogram with the specified
    # number of bins
    hist(x, breaks = bins, col = 'darkgray',
         border = 'white',
         xlab = 'Waiting time to next eruption (in mins)',
         main = 'Histogram of waiting times')
  })
}
```

Another simple example with `plotly`

`Shiny` works very well with other packages oriented on interactivity like `plotly` (https://plotly.com/r/), `DT` (https://rstudio.github.io/DT/), `leaflet` (https://rstudio.github.io/leaflet/)...

X variable

Sepal.Length

Y variable

Sepal.Width

PlantBreedGame

- https://github.com/timflutre/PlantBreedGame (https://github.com/timflutre/PlantBreedGame)
- Serious game to teach the principles of selective breeding build with `R-Shiny`
- Players are plant breeder and must carry out a successful selection campaign
- They upload “request files” and the app simulate the requested data: phenotypes, haplotypes genotypes
- App structured have been developed during a Master 2 student project in 2 months (some upates have been made after)

Shiny Pros & Cons

Pros

- **No webfront skills** required (but HTML/CSS/JS knowledge is helpfull)
- **Pure R** (or Python), any R-Package can be integrated
- Rapid development
- No web-server required

Cons

- Performances:
  - limited by **R's performances**
  - single treaded: **Calculations "block" the app**, for all other users
- **Basic design** (without HTML/CSS/JS)

Summary:

- Great **for "statisticians"**
- Great for **internal communication** (eg. Shiny powered slides like those ones)
- Good for **Prototypes** / **Proof of Concept**, or **"Inernal App"**

Some References

Shiny:

- `Shiny`'s official website (https://shiny.posit.co/)
- Source code for `R` (https://github.com/rstudio/shiny) and for `python` (https://github.com/posit-dev/py-shiny)

Some usefull Shiny tools:

- `Shiny`'s gallery (several app examples) (https://shiny.posit.co/r/gallery/)
- List of basic UI inputs (https://gallery.shinyapps.io/081-widgets-gallery/)
- Some `Shiny` extentions (for `R`) (https://github.com/nanxstats/awesome-shiny-extensions)

PlantBreedGame:

- Source code on GitHub (https://github.com/timflutre/PlantBreedGame)
- *Crop Science*, Letter to the Editor: *PlantBreedGame: A Serious Game that Puts Students in the Breeder's Seat*, by Timothée Flutre, Julien Diot, Jacques David, https://doi.org/10.2135/cropsci2019.03.0183le (https://doi.org/10.2135/cropsci2019.03.0183le)

"Shiny-Slides":

- Create slides with `R-markdown` (https://bookdown.org/yihui/rmarkdown/ioslides-presentation.html#presenter-mode)
- Integrate `Shiny` in `Rmd` document (https://bookdown.org/yihui/rmarkdown/shiny-documents.html)

Thank you very much !

Source code of the presentation is available on GitHub: https://github.com/juliendiot42/IBC_2024_presentation (https://github.com/juliendiot42/IBC_2024_presentation)