Shiny is good for you!

Christophe Bontemps
*Toulouse School of Economics, INRA*

@Xtophe_Bontemps
ABOUT ME & MY JOB

▶ Econometrician @ Toulouse School of Economics
ABOUT ME & MY JOB

- Econometrician @ Toulouse School of Economics
- R useR! (among others)
ABOUT ME & MY JOB

- Econometrician @ *Toulouse School of Economics*
- R useR! (among others)
- Teach Data Visualisation (among others)
ABOUT ME & MY JOB

➤ Econometrician @ Toulouse School of Economics
➤ R useR! (among others)
➤ Teach Data Visualisation (among others)
ABOUT ME & MY JOB

- Econometrician @ Toulouse School of Economics
- R user! (among others)
- Teach Data Visualisation (among others)

- Co-organiser of the Toulouse Dataviz Meetup
WHAT IS SHINY?

An \texttt{R} package to build interactive web applications with R:

- Requires \texttt{R} (also easier with RStudio)
WHAT IS SHINY?

An R package to build interactive web applications with R:

- Requires R (also easier with RStudio)
- Easy to write applications
WHAT IS SHINY?

An \texttt{R} package to build interactive web applications with R:

- Requires \texttt{R} (also easier with RStudio)
- Easy to write applications
- No HTML/CSS/JavaScript knowledge required...
WHAT IS SHINY?

An \texttt{R} package to build interactive web applications with R:

- Requires \texttt{R} (also easier with RStudio)
- Easy to write applications
- No HTML/CSS/JavaScript knowledge required ...
- But ...fully customizable with HTML/CSS/JavaScript though!
WHAT IS SHINY?

An R package to build interactive web applications with R:

▸ Requires R (also easier with RStudio)
▸ Easy to write applications
▸ No HTML/CSS/JavaScript knowledge required...
▸ But ...fully customizable with HTML/CSS/JavaScript though!
▸ Some examples in a minute
What is shiny?

An \texttt{R} package to build interactive web applications with R:

- Requires \texttt{R} (also easier with RStudio)
- Easy to write applications
- No HTML/CSS/JavaScript knowledge required ...
- But ...fully customizable with HTML/CSS/JavaScript though!
- Some examples in a minute
- \texttt{shiny} is easy
**Examples**

**Housing prices by Eric Ray Anderson**

**The Genetic Map Comparator by Yan Holtz, Jacques David, Vincent Ranwez**
OTHER EXAMPLES

Monitoring Learners in a MOOC - Run 3

A MOOC monitor (C. Bontemps, DEE 2017)

Show me shiny (Fully reusable applications)
Why interactions in Dataviz?

For Unwin et al. (2006), interactions have only 3 components:

- Querying
Why interactions in Dataviz?

For Unwin et al. (2006), interactions have only 3 components:

- Querying
- Selection and linking
**Why interactions in Dataviz?**

For Unwin et al. (2006), interactions have only 3 components:

- **Querying**
- **Selection and linking**
- **Varying plot characteristics**
INTERACTIONS?

▶ Querying:
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs
INTERACTIONS?

► Querying:
  ▶ Adding informing on the fly (e.g. What is the value of that outlier?)
  ▶ Adding dimensions to 2D-graphs

► Selection and linking:
INTERACTIONS?

▶ Querying :
  ▶ Adding informing on the fly (e.g. What is the value of that outlier?)
  ▶ Adding dimensions to 2D-graphs
▶ Selection and linking :
  ▶ Choosing variables of interest, displays of interest
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- Selection and linking:
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- Selection and linking:
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers

- Varying plot characteristics:
INteractions?

- **Querying:**
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- **Selection and linking:**
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers

- **Varying plot characteristics:**
  - Rescaling (zoom & pan), resizing, zooming, reordering,...
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- Selection and linking:
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers

- Varying plot characteristics:
  - Rescaling (zoom & pan), resizing, zooming, reordering,
  - Scale, colour (colour blind option), legend
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- Selection and linking:
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers

- Varying plot characteristics:
  - Rescaling (zoom & pan), resizing, zooming, reordering...
  - Scale, colour (colour blind option), legend
  - Time varying animations
INTERACTIONS?

- Querying:
  - Adding informing on the fly (e.g. What is the value of that outlier?)
  - Adding dimensions to 2D-graphs

- Selection and linking:
  - Choosing variables of interest, displays of interest
  - Selecting sub-samples, groups of interest, outliers

- Varying plot characteristics:
  - Rescaling (zoom & pan), resizing, zooming, reordering...
  - Scale, colour (colour blind option), legend
  - Time varying animations
  - Adding interaction between graphs (panels, tabs)
**WHY SHINY?**

With **shiny** we use mostly the last 2 features: "*selection and linking*" & "*changing the plot characteristics*". But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
**Why shiny?**

With **shiny** we use mostly the last 2 features: “*selection and linking*” & “*changing the plot characteristics*”. But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny** is easy
**WHY SHINY?**

With **shiny** we use mostly the last 2 features: "*selection and linking*" & "*changing the plot characteristics*". But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny** is easy
- **shiny** is not the only one!
**WHY SHINY?**

With **shiny** we use mostly the last 2 features: **"selection and linking" & "changing the plot characteristics"**. But:

- **shiny is not only for dataviz, also for easy web sharing applications.**
- **shiny is easy**
- **shiny is not the only one!**
  - **Tableau**
**Why shiny?**

With shiny we use mostly the last 2 features: “selection and linking” & “changing the plot characteristics”. But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny** is easy
- **shiny** is not the only one!
  - Tableau
  - D3.js
**WHY SHINY?**

With **shiny** we use mostly the last 2 features: “ *selection and linking*” & “ *changing the plot characteristics*”. But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny** is easy
- **shiny** is not the only one!
  - Tableau
  - D3.js

Specificity of shiny
**WHY SHINY?**

With shiny we use mostly the last 2 features: "selection and linking" & "changing the plot characteristics". But:

- **shiny is not only for dataviz, also for easy web sharing applications.**
- **shiny is easy**
- **shiny is not the only one!**
  - Tableau
  - D3.js

**Specificity of shiny**

- Simple, open source, based on major statistical software
**Why shiny?**

*With shiny we use mostly the last 2 features: “selection and linking” & “changing the plot characteristics”. But:*

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny** is easy
- **shiny** is not the only one!
  - Tableau
  - D3.js

**Specificity of shiny**

- Simple, open source, based on major statistical software
- Everything you do in R can be integrated in shiny!
**WHY SHINY?**

With **shiny** we use mostly the last 2 features: “*selection and linking*” & “*changing the plot characteristics*”. But:

- **shiny** is not only for dataviz, also for easy web sharing applications.
- **shiny is easy**
- **shiny is not the only one!**
  - Tableau
  - D3.js

**Specificity of shiny**

- Simple, open source, based on major statistical software
- Everything you do in R can be integrated in shiny!
- Huge community, lots of developments
**WHY SHINY?**

With shiny we use mostly the last 2 features: "selection and linking" & "changing the plot characteristics". But:

- shiny is not only for dataviz, also for easy web sharing applications.
- shiny is easy
- shiny is not the only one!
  - Tableau
  - D3.js

Specificity of shiny

- Simple, open source, based on major statistical software
- Everything you do in R can be integrated in shiny!
- Huge community, lots of developments
- Lots of re-usable examples
HOW WORKS SHINY?

There are basically 2 files

- The user interface file (ui.R)
How works Shiny?

There are basically 2 files

▶ The user interface file (ui.R)
▶ The R code server (server.R)
HOW WORKS SHINY?

There are basically 2 files

▶ The user interface file (ui.R)

▶ The R code server (server.R)

▶ Eventually, a global file with initial treatments (global.R)
HOW WORKS SHINY?

There are basically 2 files

- The user interface file (ui.R)
- The R code server (server.R)
- Eventually, a global file with initial treatments (global.R)
- Other things I don’t want to talk now!
HOW WORKS SHINY?

The server.R computes (in R) the elements that the ui.R request and displays

- In the ui.R, we find functions that are simply HTML wrappers
The server.R computes (in R) the elements that the ui.R request and displays

- In the ui.R, we find functions that are simply HTML wrappers
- The server.R computes elements requested
HOW WORKS SHINY?

The server.R computes (in R) the elements that the ui.R request and displays

- In the ui.R, we find functions that are simply HTML wrappers
- The server.R computes elements requested

The two files are very different
HOW WORKS SHINY?

The server.R computes (in R) the elements that the ui.R request and displays

- In the ui.R, we find functions that are simply HTML wrappers
- The server.R computes elements requested

The two files are very different

- Code in the ui.R file is shiny code (+ html)
HOW WORKS SHINY?

The server.R computes (in R) the elements that the ui.R request and displays

- In the ui.R, we find functions that are simply HTML wrappers
- The server.R computes elements requested

The two files are very different
- Code in the ui.R file is shiny code (+ html)
- Code in the server.R is R code
STRUCTURE OF A SHINY APP

The basic structure is simple, `ui.R` & `server.R` should be in the same directory.

From Iowa State university
HOW WORKS SHINY?

Let’s built our first shiny application with RStudio
WHAT IS UI.R DOING?

ui.R is collecting actions (inputs) and displaying elements (outputs)

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
    )
  )
))
```
**WHAT IS **\texttt{ui.R} **DOING?**

\texttt{ui.R} is collecting actions (inputs) and displaying elements (outputs)

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
    )
  )
))
```
What is UI.R doing?

UI.R is collecting actions (inputs) and displaying elements (outputs)

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
    )
  )
))
```
WHAT IS \texttt{ui.R} DOING?

\texttt{ui.R} is collecting actions (inputs) and displaying elements (outputs)

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
        )
    )

))
```
**WHAT IS SERVER.R DOING?**

server.R is receiving actions (inputs) and computing elements (outputs) to be displayed by ui.R

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')

  })

})
```


**WHAT IS SERVER.R DOING?**

`server.R` is receiving actions (inputs) and computing elements (outputs) to be displayed by `ui.R`

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')

  })

})
```
**What is server.R doing?**

`server.R` is receiving actions (inputs) and computing elements (outputs) to be displayed by `ui.R`.

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')
  )
})
```
**WHAT IS SERVER.R DOING?**

**server.R** is receiving actions (inputs) and computing elements (outputs) to be displayed by **ui.R**.

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')

  })

})
```

What is server.R doing?

server.R is receiving actions (inputs) and computing elements (outputs) to be displayed by ui.R

```r
library(shiny)

# Define server logic required to draw a histogram
shinyServer(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')

  })
})
```
WHAT IS UI.R DOING?

ui.R is collecting actions (inputs) and displaying elements (outputs)

```r
library(shiny)

# Define UI for application that draws a histogram
shinyUI(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")
    )
  )
))
```
LIVE DEMO - PREPARED WITH E. MAIGNÉ (INRA)

Let us modify `ui.R` and `server.R`
A NOTE ON SCOPE

What is done once vs what is done every time the function is called?

```r
server
  > library(shiny)
  > Some R code # Will load once, when Shiny starts, and will be available to each session
  > server <- function(input, output) {
  >     Some R code # Objects here are defined in each session
  >     output$xxxx <- renderPlot({
  >       # Objects here are defined each time this function is called
  >       some R code using input$zzz
  >     })
  > }
```

See scoping in shiny
**WHATEVER YOU DO IN R, CAN BE DONE IN SHINY!**

server.R basically receives parameters (inputs) and computes! So whatever you do in R can be an output for shiny:

▶ **Text (summaries, estimation results, raw numbers, ..)**
WHATEVER YOU DO IN R, CAN BE DONE IN SHINY!

server.R basically receives parameters (inputs) and computes! So whatever you do in R can be an output for shiny:

- Text (summaries, estimation results, raw numbers, ..)
- Plot (Statistical, images, interactive plots? ...)

---

**Whatever you do in R, can be done in Shiny!**

*server.R* basically receives parameters (inputs) and computes! So whatever you do in R can be an output for Shiny:

- Text (summaries, estimation results, raw numbers, ..)
- Plot (Statistical, images, interactive plots? ...)
- Table (Standard, table widget, customized, ...)

Whatever you do in R, can be done in Shiny!
Many options for the interface: Highly customizable!

ui.R has a huge (and increasing) collection of Inputs:

- `Button`
- `Single checkbox`
- `Checkbox group`
- `Date input`
- `Colour input`
- `Date range`
- `File input`
- `Numeric input`
- `Password Input`
- `Radio buttons`
- `Select box`
- `Sliders`
- `Text input`
- `TextAreaInput()`
STRUCTURE OF A MORE COMPLEX SHINY APP

For more complex structures `global.R` can complement `ui.R` & `server.R` (in the same directory)

Folder/File structure for R shiny app if you have a data set to read-in and/or manipulate prior to use.

From Iowa State university
SHINY?

An R package to build interactive web applications with R:
- Requires R (also easier with RStudio)
SHINY?

An R package to build interactive web applications with R:

- Requires R (also easier with RStudio)
- It is very easy to write applications
**SHINY**?

An R package to build interactive web applications with R:

- Requires R (also easier with RStudio)
- It is **very easy** to write applications
- No HTML/CSS/JavaScript knowledge required...
SHINY?

An \texttt{R} package to build interactive web applications with R:

- Requires \texttt{R} (also easier with RStudio)
- It is \texttt{very easy} to write applications
- No HTML/CSS/JavaScript knowledge required...
- Publication tool embedded
SHINY?

An R package to build interactive web applications with R:

➤ Requires R (also easier with RStudio)
➤ It is very easy to write applications
➤ No HTML/CSS/JavaScript knowledge required ...
➤ Publication tool embedded

see ShinyApps.io
REFERENCES