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 R package to build interactive web applications with R :

- Requires 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 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 ...

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 !

WHAT IS SHINY ?

An R package to build interactive web applications with R :

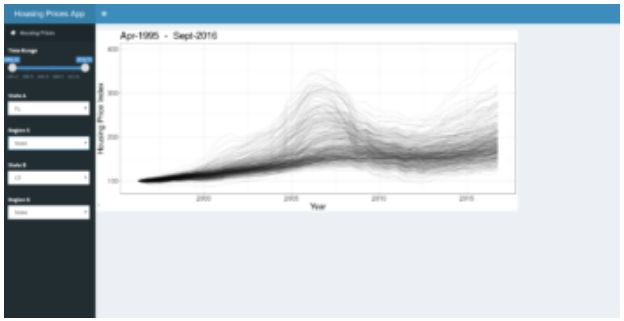
- Requires R (also easier with RStudio)
- Easy to write applications
- No HTML/CSS/JavaScript knowledge required ...
- But ...fully customizable with HTML/CSS/JavaScript though !
- Some examples in a minute

WHAT IS SHINY ?

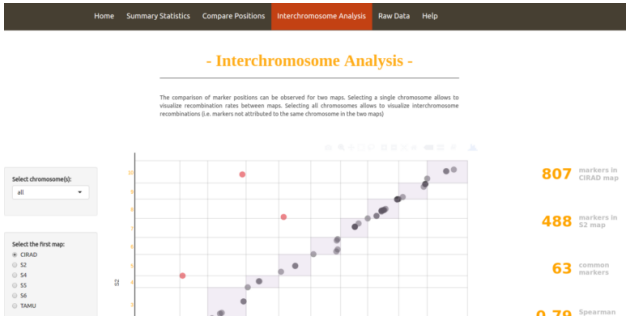
An R package to build interactive web applications with R :

- 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
- `shiny` is **easy**

Examples

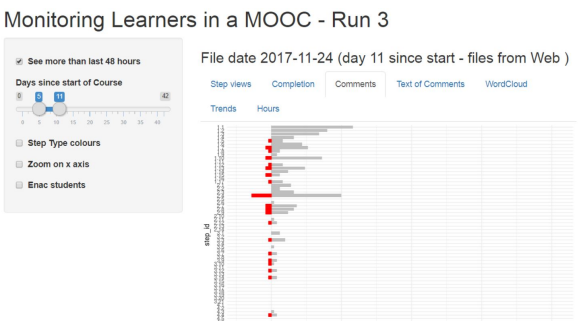


Housing prices by Eric Ray Anderson

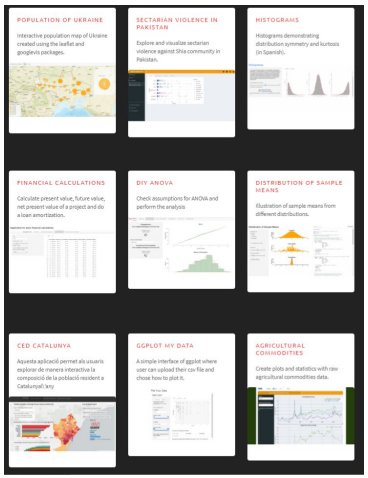


The Genetic Map Comparator by Yan Holtz, Jacques David, Vincent Ranwez

Other examples



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

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

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 ?

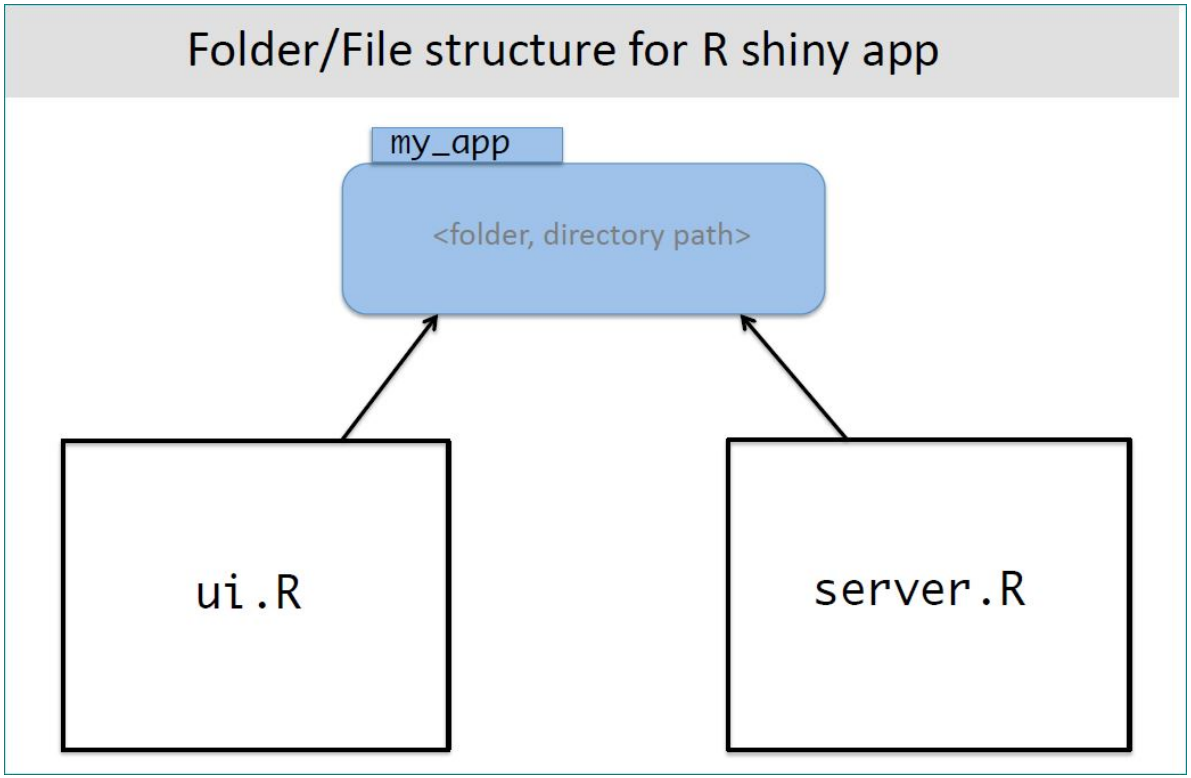
The `server.R` computes (in R) the elements that the `ui.R` request and displays

- 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)

```
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)

```
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)

```
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)

```
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`

```
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`

```
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`

```
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`

```
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`

```
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')

  })

})
```

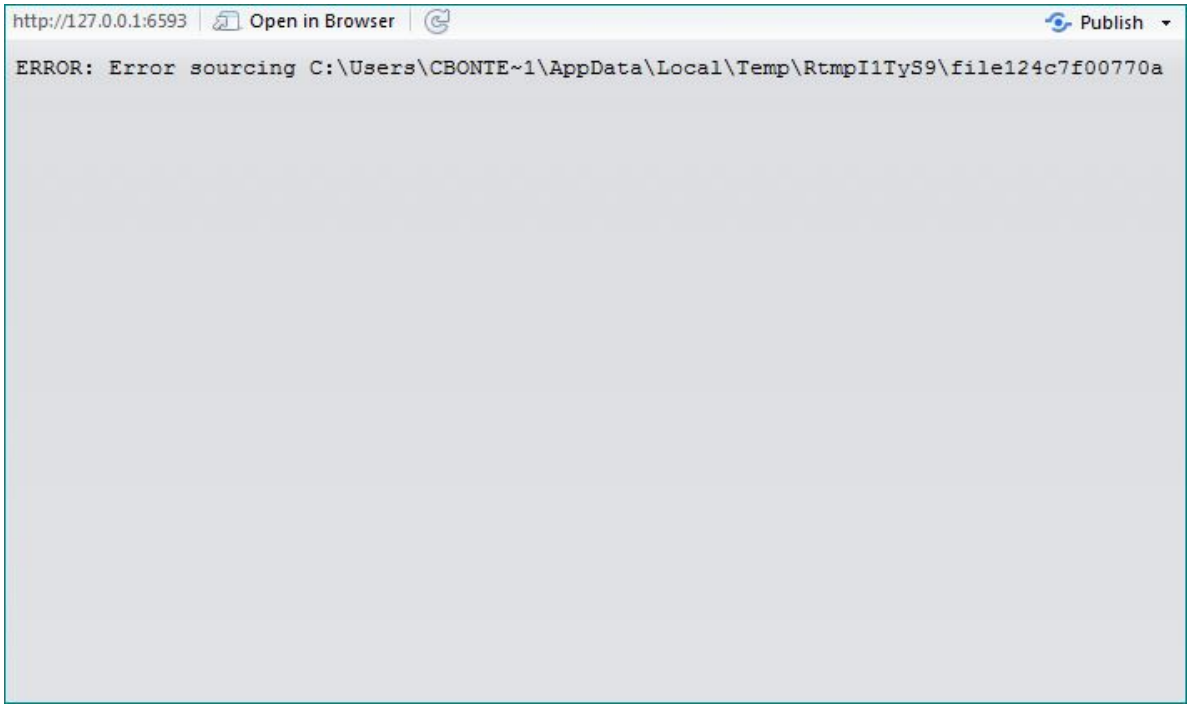
WHAT IS UI.R DOING ?

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

```
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 ?

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$xxx<- renderPlot({
              # Objects here are defined each time this function is called
              some R code using input$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 ? ...)

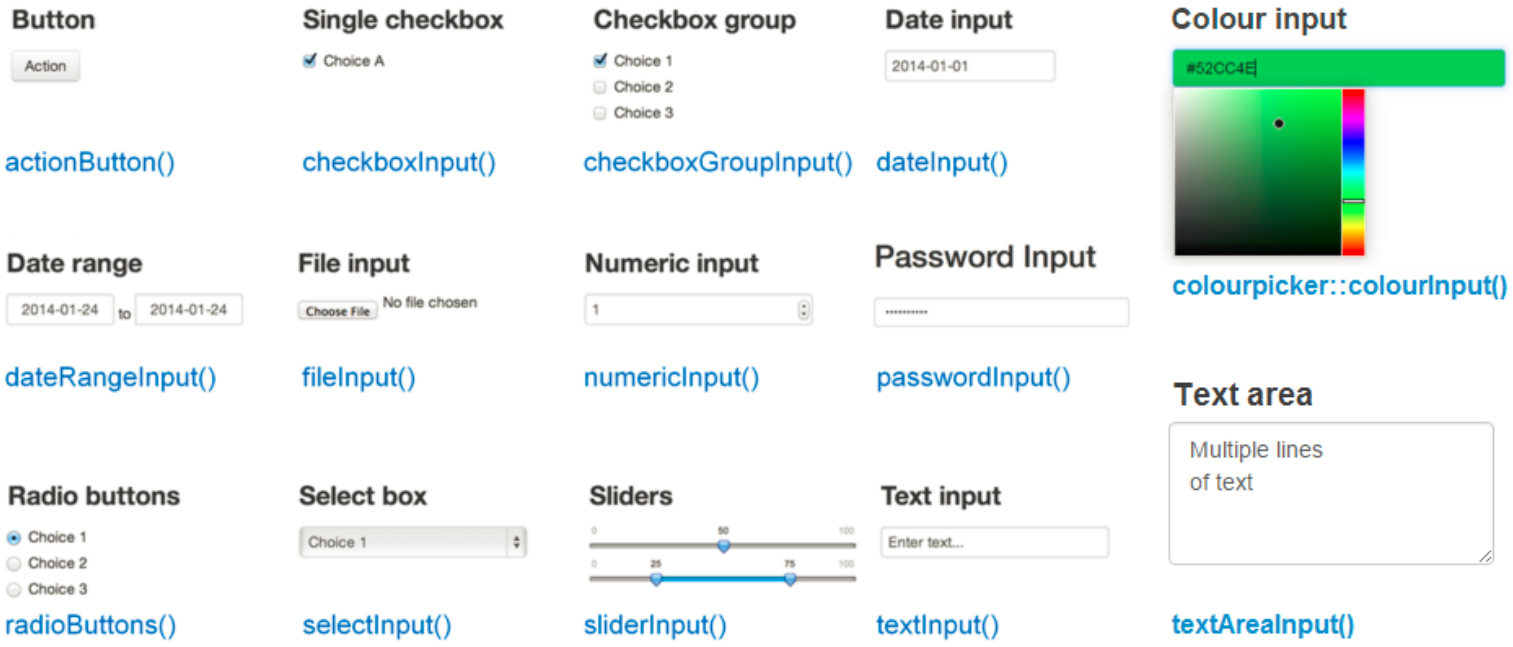
WHATEVER YOU DO IN R, CAN BE DONE IN SHINY !

`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, ...)

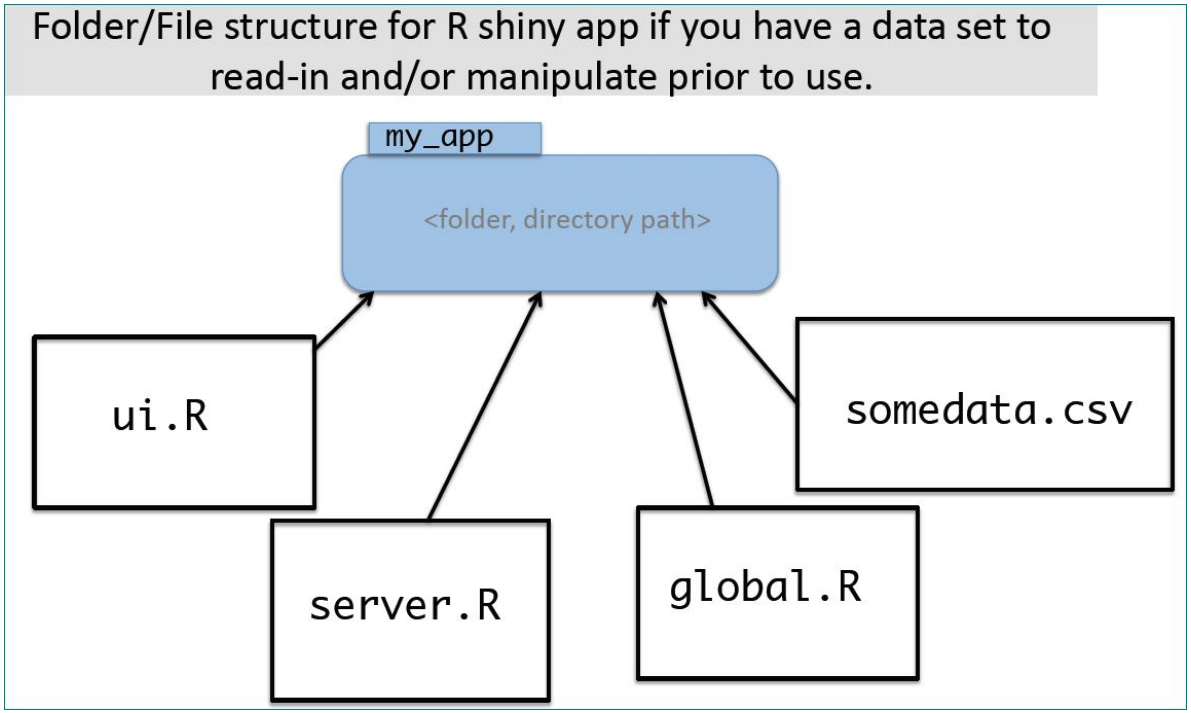
Many options for the interface : Highly customable !

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



Structure of a more complex shiny app

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



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 ?

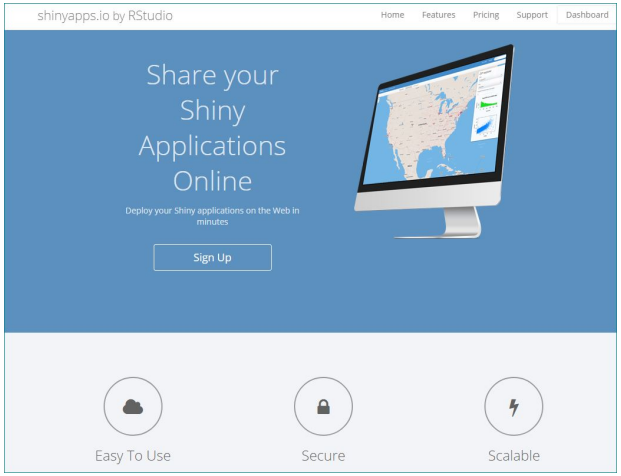
An R package to build interactive web applications with R :

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

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 I

Unwin, A., Theus, M., and Hofmann, H. (2006). *Graphics of large datasets : visualizing a million*. Springer Science & Business Media.