Building our 1st page together

Eva Gfrerer
Statistics, Visualization and More Using "R"
BASIC ANATOMY OF A SHINY APP
Basic anatomy of a Shiny app

Shiny app basically consists of three parts:

- `library(shiny)`
- `#load(data)`
- `ui <- fluidPage()`
- `server <- function(input, output) {}`
- `shinyApp(ui = ui, server = server)`
Basic anatomy of a Shiny app

Shiny app basically consists of three parts:

```r
> library(shiny)
> #load(data)

> ui <- fluidPage()
> server <- function(input, output) {}
> shinyApp(ui = ui, server = server)
```

**ui = user interface**, the setup for the user, where the user can later find the elements. Here the inputs are defined.
Basic anatomy of a Shiny app

Shiny app basically consists of three parts:

> library(shiny)
> #load(data)

> ui <- fluidPage()
> server <- function(input, output) {}
> shinyApp(ui = ui, server = server)

server = a function with instructions on how to build and rebuild the R objects displayed in the UI. Here the outputs are defined (e.g. a ggplot)
Basic anatomy of a Shiny app

Shiny app basically consists of three parts:

> library(shiny)
> #load(data)

> ui <- fluidPage()
> server <- function(input, output) {}
> shinyApp(ui = ui, server = server)

**shinyApp**: combines ui and server into an app.
LET’S START OUR FIRST PAGE!
Let's start our First Page!

- **Load the code:**
  ```r
  ## install shiny & load the libraries
  if (!require("shiny")) install.packages("shiny")
  library(shiny) library(ggplot2)

  ## load data
  A <- read.table("http://www.trutschnig.net/Datensatz.txt", head=TRUE)
  address <- url("http://www.trutschnig.net/RTR2015.RData")
  load(address)
  labels(RTR2015)
  ```

- What’s the first part of our app?
# 1. Define UI for App: start with `fluidPage()` function

```r
ui <- fluidPage(
  # and add a title:
  titlePanel("TextMy 1st page"),
)
```
ui

> # 1. Start with \texttt{fluidPage()} function
> ui <- \texttt{fluidPage(}
>     \texttt{titlePanel("My 1\textsuperscript{st} page"),}
> 
> > # 2. Define layout of our app: We use a sidebar layout
> > \texttt{sidebarLayout(}
> > # This Layout consists of a sidebarPanel and
> > > # a mainPanel:
> > > \texttt{sidebarPanel(),}
> > > \texttt{mainPanel()}
> > )
> )
Extra: There are many different layouts

```r
> sidebarLayout()
> verticalLayout()
> splitLayout()
> flowLayout()
> fluidRow(column(width = 4),
>         column(width = 2, offset = 3)),
>         fluidRow(column(width = 12))
> )
> # ...
```

→ check the Shiny::cheatsheet for more
NOW YOU HAVE A VERY BASIC UI

But the app doesn’t work yet...

→ we still need to define the server
server

> #add after ui
> # 1. basic server function takes two arguments:
> #   input & output
> server <- function(input, output) {}
Now, let’s put it aaall together

> #Combine ui with server to an app
> shinyApp(ui = ui, server = server)

→ Run the App (ui + server + shinyApp)
CONGRATULATIONS,
you have a very basic shinyApp

But it’s still a really boring app
→ we haven’t used any data yet!
Let’s define some inputs:
We need to define input controls for our `sidebarPanel()` and `mainPanel()`

```r
> ui <- fluidPage(
>   titlePanel("My 1st page"),
>   sidebarLayout(
>     # Let's start with the `sidebarPanel()`
>     sidebarPanel(),
>     mainPanel()
>   )
> )
```
We need to define input controls for our `sidebarPanel()` and `mainPanel()`.

```r
ui <- fluidPage(
  titlePanel("My 1st page"),
  sidebarLayout(
    # Let's start with the `sidebarPanel()`
    sidebarPanel(
      # 3. Define an input with `selectInput()` , use our RTR2015 dataset
      selectInput(inputId = "x",
                   label = "X-axis:",
                   choices = c("device", "rtr_speed_dl", "rtr_speed_ul"),
                   selected = "device"),
    ),
    mainPanel()
  )
)
```
Go back to the UI

We need to define input controls for our sidebarPanel() and mainPanel()

```r
ui <- fluidPage(
  titlePanel("My 1st page"),
  sidebarLayout(
    sidebarPanel(
      # 3. Define an input with selectInput(), use our RTR2015 dataset
      selectInput(inputId = "x",
                   label = "X-axis: ",
                   choices = c("device", "rtr_speed_dl", "rtr_speed_ul"),
                   selected = "device"),
      #`We also need inputs for the Y-axis:
      selectInput(inputId = "y",
                   label = "Y-axis: ",
                   choices = c("rtr_ping", "mtime", "rtr_speed_dl", "rtr_speed_ul"),
                   selected = "rtr_ping"),
    ),
    mainPanel()
  )
)
```
Exercise 1

- Change the variable “device” to “op_name”

- It’s important that our app is user friendly. Give our variables readable labels!

> #example code for readable labels:
> ...
> choices = c("Readable Label 1" = "variable_name1",
> "Readable Label 2" = "variable_name1"),
> selected = "variable_name1")
Exercise 1: Solution

```r
ui <- fluidPage(
  titlePanel("My 1st page"),
  sidebarLayout(
    sidebarPanel(
      # 3. Define our sidebar panel containing input controls
      selectInput(inputId = "x",
                  label = "X-axis",
                  choices = c("Operator" = "op_name",
                                "Download Speed" = "rtr_speed_dl",
                                "Upload Speed" = "rtr_speed_ul"),
                  selected = "op_name"),

      selectInput(inputId = "y",
                  label = "Y-axis:",
                  choices = c("rtr_ping", "mtime",
                                "rtr_speed_dl", "rtr_speed_ul"),
                  selected = "rtr_ping"),
    ),
    mainPanel()
  )
)
```
Complete UI

```r
> ui <- fluidPage(
>   sidebarLayout(
>     sidebarPanel(
>       # 3. Define our sidebar panel containing input controls
>       selectInput(inputId = "x",
>           label = "X-axis",
>           choices = c(
>             "Operator" = "op_name",
>             "Download Speed" = "rtr_speed_dl",
>             "Upload Speed" = "rtr_speed_ul"),
>           selected = "op_name"),
>       selectInput(inputId = "y",
>           label = "Y-axis:",
>           choices = c("rtr_ping", "mtime",
>                         "rtr_speed_dl", "rtr_speed_ul"),
>           selected = "rtr_ping"),
>     ),
>     # 4. Let’s complete the UI: Define output of our sidebar
>     mainPanel(plotOutput(outputId = "boxplot"))
>   )
> )
```
server

> # We just defined a plot in the input. BUT we also need to
> # make it clear what the output of this plot should be!
> server <- function(input, output) {

> # The renderPlot({}) function in the output works together
> # with the plotOutput() function in the input.
> # It specifies **how** the plot output should be updated.
> output$boxplot <- renderPlot({
>   ggplot(data = RTR2015, aes_string(x =
>   input$x, y = input$y)) +
>   geom_boxplot()
> })
> }

> 
Extra:
render*() and *Output() functions work together

DT::render**DataTable**(expr, options, callback, escape, env, quoted)

render**Image**(expr, env, quoted, deleteFile)

render**Plot**(expr, width, height, res, ..., env, quoted, func)

render**Table**(expr, ..., env, quoted, func)

render**Text**(expr, env, quoted, func)

render**UI**(expr, env, quoted, func)

data**Table**Output(outputId, icon, ...)

image**Output**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)

plot**Output**(outputId, width, height, ...)

table**Output**(outputId)

text**Output**(outputId, container, inline)

ui**Output**(outputId, inline, container, ...)
Exercise 2

Change the “boxplot” to a “scatterplot” as it might show our data better...
Exercise 2: Solution

```r
debugging
>
> #UI
> ui <- fluidPage(
>   ...,
>   ...,
>   ...,
>   mainPanel(plotOutput(outputId = "scatterplot"))
> )
> #Server
> server <- function(input, output) {
>   output$scatterplot <- renderPlot({
>     ggplot(data = RTR2015, aes_string(x =
>       input$x, y = input$y)) +
>     geom_point()
>   })
> }
> #complete the shinyApp
> shinyApp(ui = ui, server = server)
```
Exercise 3: Fun time

Choose:

• Add a z-axis as a variable for colour!
• Make own app with the bankomat data
Where to find more about shinyApp

- **In the next session of this course!**
- **Shiny Homepage** <https://shiny.rstudio.com/tutorial>
- **Shiny::cheat sheet**
  <https://shiny.rstudio.com/articles/cheatsheet.html>
The End