



PhUSE EU Connect 2019,
Amsterdam

Building interactive web apps – Experiences with R Shiny



DV06

13 November 2019 / Stephanie Fechtner





Agenda

// Motivation

// **R Shiny**

// The user interface

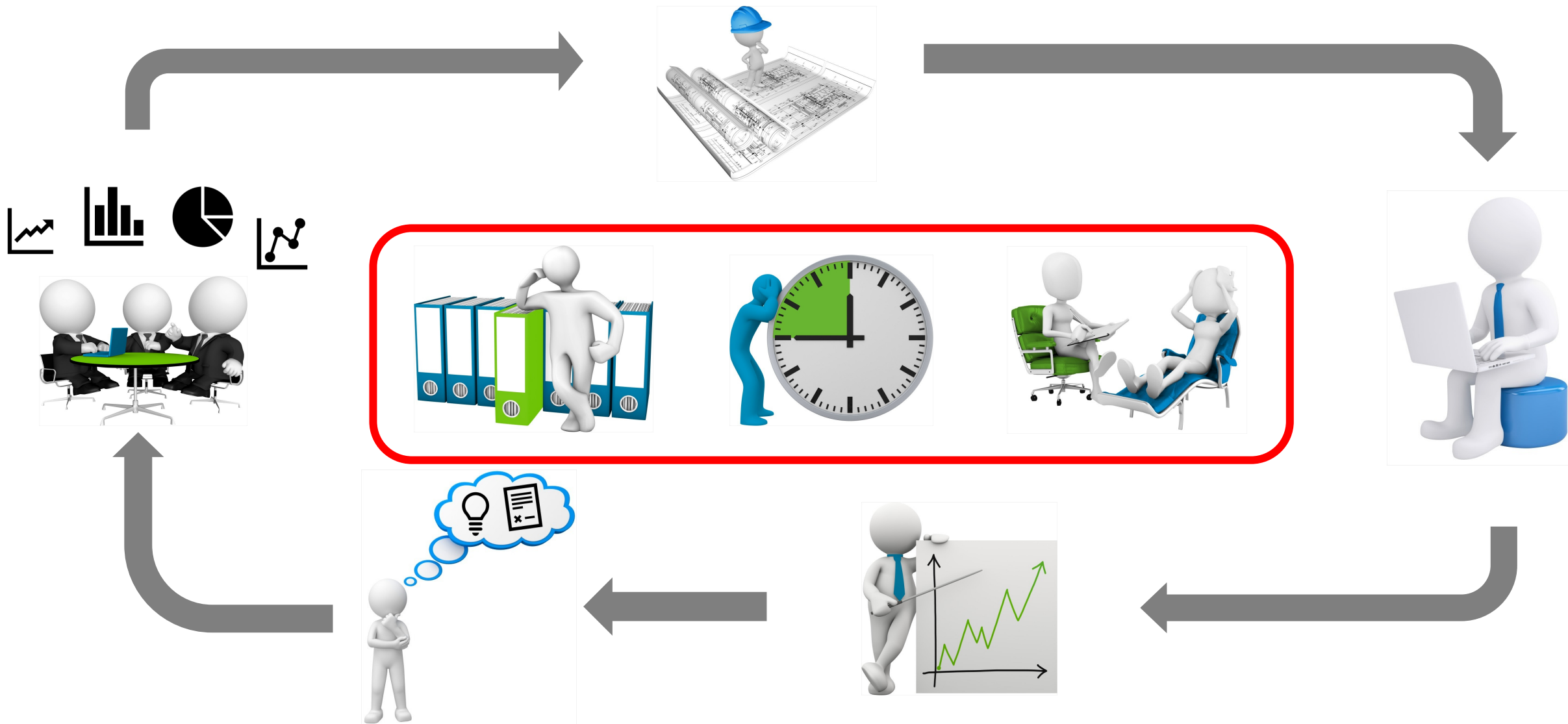
// The server function

// Summary and discussion



Motivation

Creation of figures for a study/project → process



One option for optimization → R Shiny

- // Shiny = R package
- // Creates a web application which can be accessed via the internet
- // Investigators/ medical experts/ physicians can explore the data by themselves
 - // User does not have to know R
- // Apps can reduce the output of unused figures and tables and new data aspects can be easily investigated by medical experts or physicians

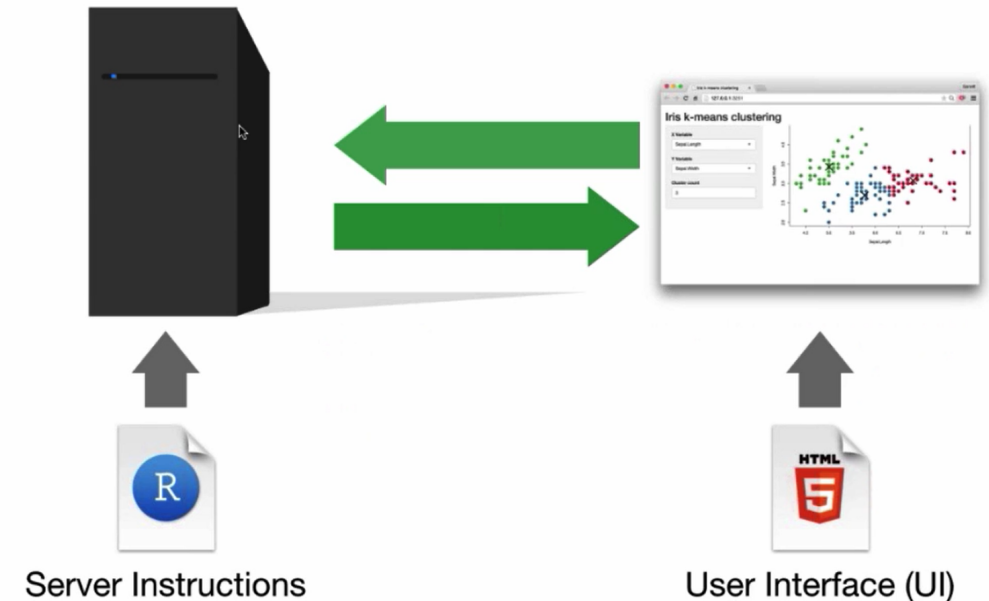




R Shiny

R Shiny

- // Requirements:
 - // R Version 3.0.2 or higher
 - // Integrated development environment RStudio
 - // Installation of R Package shiny
- // Every Shiny app is maintained by a computer running R and consists of two components:
 - // User interface: web page that the user will see; written HTML (generated from R)
 - // Server instructions: set of instructions which tells the server what to do if the app user changes the different input objects within the app





How to build a Shiny app – Step by Step

// Recommended to always start with the same template:

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



User interface

// Input and output objects

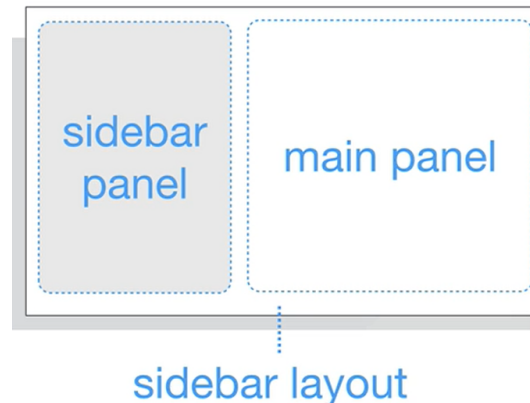
```
ui <- fluidPage(  
  # *Input() functions,  
  # *Output() functions  
)
```

// Inputs: can be modified by the user to provide any kind of value to the app

// Outputs: R objects that the user can see (plots, tables, text,...) and which respond if the user changes the inputs

// Where to display input and output objects? → sidebarLayout()

```
ui <- fluidPage(  
  sidebarLayout(  
    sidebarPanel(),  
    mainPanel()  
  )  
)
```





Input functions

Buttons



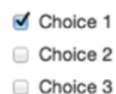
`actionButton()`
`submitButton()`

Single checkbox



`checkboxInput()`

Checkbox group



`checkboxGroupInput()`

Date input



`dateInput()`

Date range



`dateRangeInput()`

File input



`fileInput()`

Numeric input



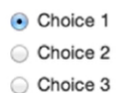
`numericInput()`

Password Input



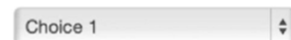
`passwordInput()`

Radio buttons



`radioButtons()`

Select box



`selectInput()`

Sliders



`sliderInput()`

Text input



`textInput()`

// All input functions need specific arguments

// `inputId` (required): give the input function a unique name

// `label` (required): display the label which the user can see

// Specific arguments for each function → R help page

Example – The Framingham Heart Study

- // Aim: Detection of causes and risks of the Coronary heart disease (CHD)
- // “Heart”-dataset of the Sashelp-library
 - // Contains the data of the first 5209 adult patients included in the Framingham Heart Study
 - // 17 variables: Status, Cause of Death, Age CHD Diagnosed, Sex, Age at Start, Height, Weight, Diastolic, Systolic, Metropolitan Relative Weight, Smoking, Age at Death, Cholesterol, Cholesterol Status, Blood Pressure Status, Weight Status, Smoking Status

// Creation of a Shiny app to investigate the heart-dataset including selectable options:

- // Type of plot
- // Variables of the dataset
- // Size and title of the plot

	Status	DeathCause Cause of Death	AgeCHDdiag Age CHD Diagnosed	Sex	AgeAtStart Age at Start	Height	Weight	Diastolic	Systolic	MRW Metropolitan Relative Weight	Smoking	AgeAtDeath Age at Death	Cholesterol	Chol_Status Cholesterol Status	BP_Status Blood Pressure Status	Weight_Status Weight Status	Smoking_Status Smoking Status
1	Dead	Other		Female	29	62.5	140	78	124	121	0	55	.		Normal	Overweight	Non-smoker
2	Dead	Cancer		Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker
3	Alive			Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)
4	Alive			Female	39	65.75	158	80	128	123	0	.	242	High	Normal	Overweight	Non-smoker
5	Alive			Male	42	66	156	76	110	116	20	.	281	High	Optimal	Overweight	Heavy (16-25)
6	Alive			Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker
7	Alive			Female	36	64.75	136	80	112	110	15	.	196	Desirable	Normal	Overweight	Moderate (6-15)
8	Dead	Other		Male	53	65.5	130	80	114	99	0	77	276	High	Normal	Normal	Non-smoker
9	Alive			Male	35	71	194	68	132	124	0	.	211	Borderline	Normal	Overweight	Non-smoker
10	Dead	Cerebral V...		Male	52	62.5	129	78	124	106	5	82	284	High	Normal	Normal	Light (1-5)
11	Alive			Male	39	66.25	179	76	128	133	30	.	225	Borderline	Normal	Overweight	Very Heavy (>...
12	Alive			57 Male	33	64.25	151	68	108	118	0	.	221	Borderline	Optimal	Overweight	Non-smoker
13	Alive			55 Male	33	70	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker
14	Alive			79 Male	57	67.25	165	76	128	118	15	.	.		Normal	Overweight	Moderate (6-15)
15	Alive			66 Male	44	69	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (>...
16	Alive			Female	37	64.5	134	76	120	108	10	.	196	Desirable	Normal	Normal	Moderate (6-15)
17	Alive			Male	40	66.25	151	72	132	112	30	.	192	Desirable	Normal	Overweight	Very Heavy (>...
18	Dead	Cancer		56 Male	56	67.25	122	72	120	87	15	72	194	Desirable	Normal	Underweight	Moderate (6-15)
19	Alive			Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)
20	Dead	Coronary ...		74 Male	46	66.5	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (>...
21	Alive			Female	37	66.25	148	78	110	112	15	.	192	Desirable	Optimal	Overweight	Moderate (6-15)
22	Alive			Female	45	64	147	74	120	119	5	.	209	Borderline	Normal	Overweight	Light (1-5)
23	Alive			Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker
24	Alive			Female	36	63.75	122	84	132	102	0	.	184	Desirable	Normal	Normal	Non-smoker
25	Alive			Female	50	67.5	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)



Example – The Framingham Heart Study

```
ui <- fluidPage(  
  tabsetPanel(  
    tabPanel(titlePanel("Framingham Heart Study"),  
      sidebarLayout(  
        sidebarPanel(width=2,  
          h4("Dataset and title"),  
          fileInput(inputId="file", label="File input:",  
                    accept=c("text/csv", ".csv")),  
          textInput(inputId="title", label="Choose a title  
            for your plot", value="Plot"),  
          h4("Choose the type of plot"),  
          selectInput(inputId="Plot_type",  
            label="Which plot do you like to create?",  
            list('','Histogram','Scatterplot', 'Boxplot'),  
            multiple=FALSE)  
        ),  
        mainPanel()  
      )  
    )  
  )  
  server <- function(input, output) {}  
  shinyApp(ui = ui, server = server)
```

The screenshot shows a web browser window with the URL `http://127.0.0.1:3866` and an 'Open in Browser' button. The application title is 'The Framingham Heart Study'. The interface is divided into a sidebar and a main panel. The sidebar contains three sections: 'Dataset and title' with a 'File input' section showing 'Browse...' and 'No file selected' buttons; 'Choose a title for your plot' with a text input field containing 'Plot'; and 'Choose the type of plot' with a dropdown menu labeled 'Which plot do you like to create?' showing options: 'Histogram', 'Scatterplot', and 'Boxplot'. The main panel is currently empty.

Output functions

Function	inserts
dataTableOutput()	an interactive table
htmlOutput()	raw HTML
imageOutput()	Image
plotOutput()	Plot
tableOutput()	Table
textOutput()	Text
uiOutput()	A Shiny UI element
verbatimTextOutput()	Text

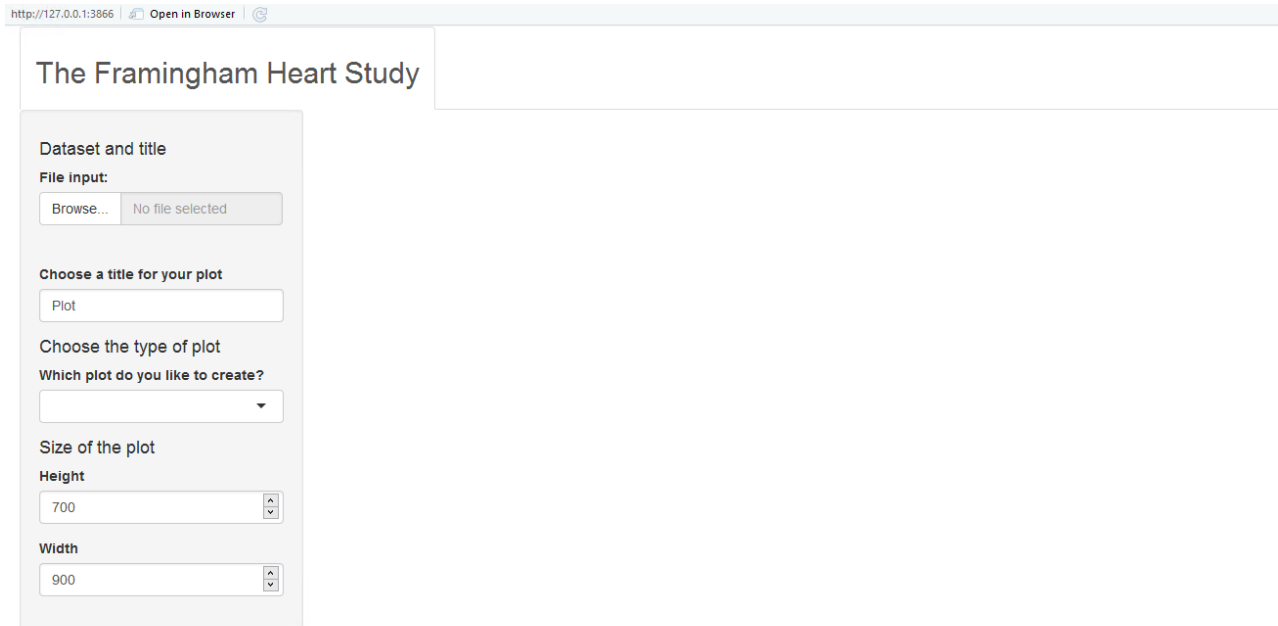
// All output functions need specific arguments

// `outputId` (required): give the output object a unique name

// Specific arguments for each function (optional) → R help page

Example – The Framingham Heart Study

```
ui <- fluidPage(
  tabsetPanel(
    tabPanel(titlePanel("Framingham Heart Study"),
      sidebarLayout(
        sidebarPanel(width=2,
          h4("Dataset and title"),
          fileInput(inputId="file", "label=File input:",
            accept=c("text/csv", ".csv")),
          textInput(inputId="title", label="Choose a title
            for your plot", value="Plot"),
          h4("Choose the type of plot"),
          selectInput(inputId="Plot_type",
            label="Which plot do you like to create?",
            list('','Histogram','Scatterplot', 'Boxplot'),
            multiple=FALSE),
          h4("Size of the plot"),
          numericInput(inputId="hei", label="Height",
            value=700, min=500, max=2000),
          numericInput(inputId="wid", label="width",
            value=900, min=500, max=2000)
        ),
        mainPanel(width=10, plotOutput(outputId = "plot",
          height="auto", width="auto"))
      )
    )
  )
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```



The screenshot shows a web browser window with the URL `http://127.0.0.1:3866` and a tab labeled "Open in Browser". The application title is "The Framingham Heart Study". The interface is divided into a sidebar and a main panel. The sidebar contains the following sections:

- Dataset and title**: A "File input:" section with a "Browse..." button and the text "No file selected". Below it is a "Choose a title for your plot" section with a text input field containing the word "Plot".
- Choose the type of plot**: A section titled "Which plot do you like to create?" with a dropdown menu.
- Size of the plot**: A section with two numeric input fields: "Height" (set to 700) and "Width" (set to 900). Both fields have up and down arrows for adjustment.

The main panel is currently empty, intended for the plot output.



The Server function

// Follow 3 rules to create the server function:

1. Save objects to display to `output$`: `output$object` → `plotOutput("object")` (same name in both components)
2. Build objects to display with `render*()`: creation of the desired type of output

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

3. Use input values with `input$` when the outputs are created: `inputId = "object"` → `input$object`

Example – The Framingham Heart Study

```
ui <- fluidPage(
  tabsetPanel(
    tabPanel(titlePanel("Framingham Heart Study"),
      sidebarLayout(
        sidebarPanel(width=2,
          h4("Dataset and title"),
          fileInput(inputId="file", label="File input:",
            accept=c("text/csv", ".csv")),
          textInput(inputId="title", label="Choose a title
            for your plot", value="Plot"),
          h4("Choose the type of plot"),
          selectInput(inputId="Plot_type",
            label="which plot do you like to create?",
            list('', 'Histogram', 'Scatterplot', 'Boxplot'),
            multiple=FALSE),
          h4("Choose corresponding variables"),
          uiOutput("x"),
          uiOutput("y"),
          h4("Size of the plot"),
          numericInput(inputId="hei", label="Height",
            value=700, min=500, max=2000),
          numericInput(inputId="wid", label="Width",
            value=900, min=500, max=2000)
        ),
        mainPanel(width=10, plotOutput(outputId = "plot",
          height="auto", width="auto"))
      )
    )
  )
)
```

```
server <- function(input, output) {
  data1 <- reactive({
    if (is.null(input$file)){
      return(NULL)
    }
    read.csv(input$file$datapath, header=T, sep=",")
  })
  output$x <- renderUI({
    if (is.null(data1())) return()
    tagList(
      selectInput(inputId="x", label="Choose a variable for the
        x-axis", choices=names(data1()), multiple=FALSE)
    )
  })
  output$y <- renderUI({
    if (is.null(data1()) || input$Plot_type == "Histogram")
      return()
    tagList(
      selectInput(inputId="y", label="Choose a variable for the
        y-axis", choices=names(data1()), multiple=FALSE)
    )
  })
}
```

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



Dataset and title

Browse...

Upload complete

900

Example – The Framingham Heart Study

```
ui <- fluidPage(
  tabsetPanel(
    tabPanel(titlePanel("Framingham Heart Study"),
      sidebarLayout(
        sidebarPanel(width=2,
          h4("Dataset and title"),
          fileInput(inputId="file", label="File input:",
            accept=c("text/csv", ".csv")),
          textInput(inputId="title", label="Choose a title
            for your plot", value="Plot"),
          h4("Choose the type of plot"),
          selectInput(inputId="Plot_type",
            label="which plot do you like to create?",
            list('', 'Histogram', 'Scatterplot', 'Boxplot'),
            multiple=FALSE),
          h4("Choose corresponding variables"),
          uiOutput("x"),
          uiOutput("y"),
          h4("Size of the plot"),
          numericInput(inputId="hei", label="Height",
            value=700, min=500, max=2000),
          numericInput(inputId="wid", label="width",
            value=900, min=500, max=2000)
        ),
        mainPanel(width=10, plotOutput(outputId = "plot",
          height="auto", width="auto"))
      )
    )
  )
)
```

```
server <- function(input, output) {
  ...
  output$plot <- renderPlot({
    if (is.null(data1())) return()

    ### Histogram ###
    if (input$Plot_type == "Histogram") {
      histogram( ~ data1()[,which(names(data1())==input$x)],
        data=data1(), xlab=input$x, main=input$title)
    }
    ### Scatterplot ###
    else if (input$Plot_type == "Scatterplot") {
      xyplot(data1()[,which(names(data1())==input$y)] ~
        data1()[,which(names(data1())==input$x)], data=data1(),
        xlab=input$x, ylab=input$y, main=input$title)
    }
    ### Boxplot ###
    else if (input$Plot_type == "Boxplot") {
      bwplot(data1()[,which(names(data1())==input$y)] ~
        data1()[,which(names(data1())==input$x)], data=data1(),
        xlab=input$x, ylab=input$y, main=input$title)
    }
  }, height=function(x) input$hei, width=function(x) input$wid)
}
```

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

The Framingham Heart Study

Dataset and title

File input:

Browse...

heart.csv

Upload complete

Choose a title for your plot

Boxplots

Choose the type of plot

Which plot do you like to create?

Boxplot

Choose corresponding variables

Choose a variable for the x-axis

Smoking_Status

Choose a variable for the y-axis

Cholesterol

Size of the plot

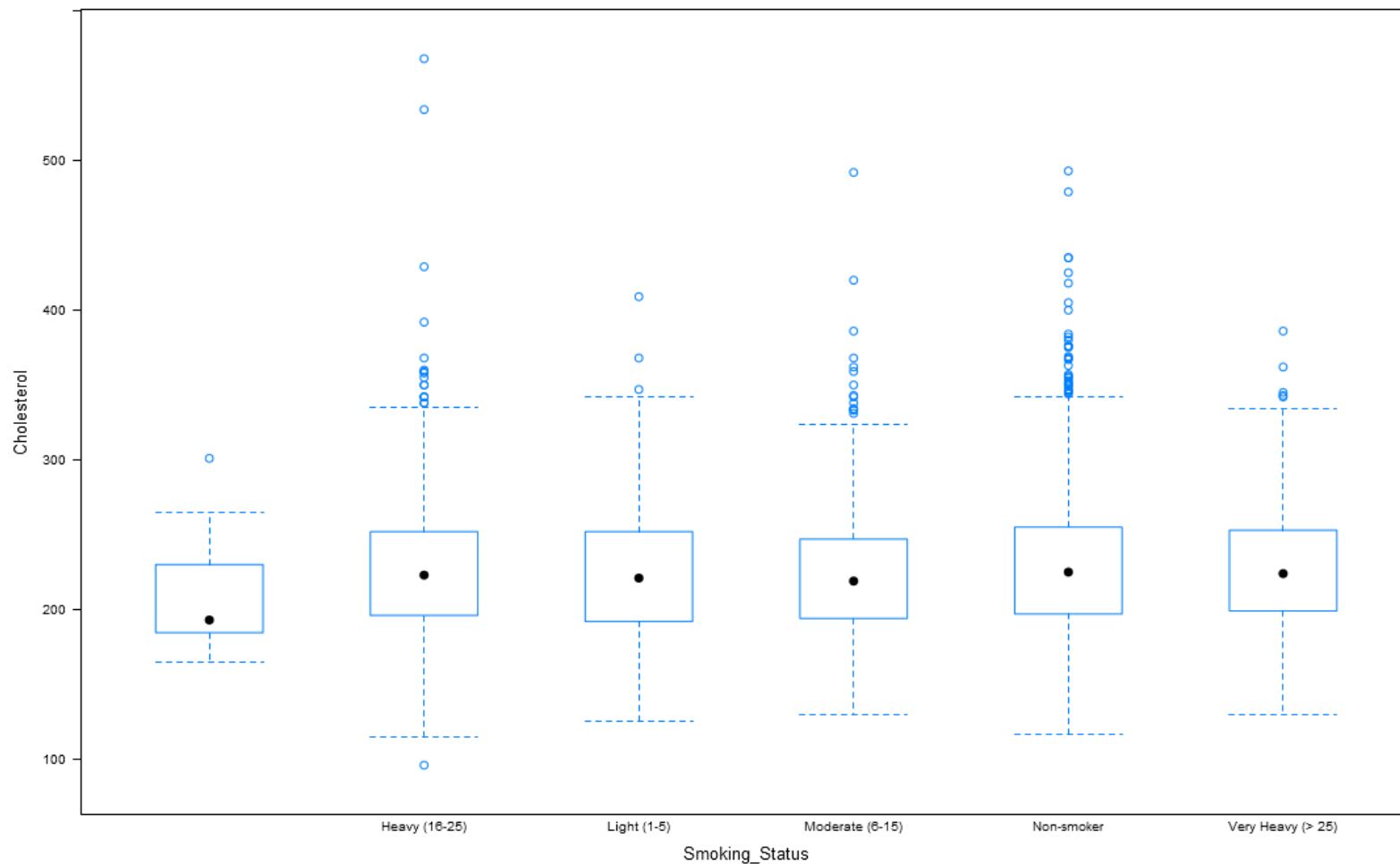
Height

700

Width

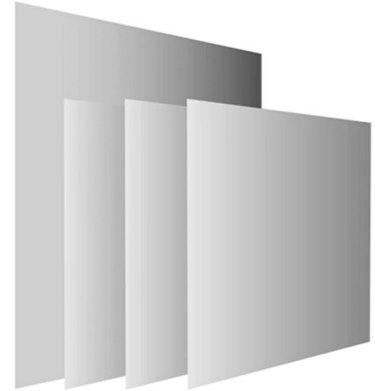
1100

Boxplots

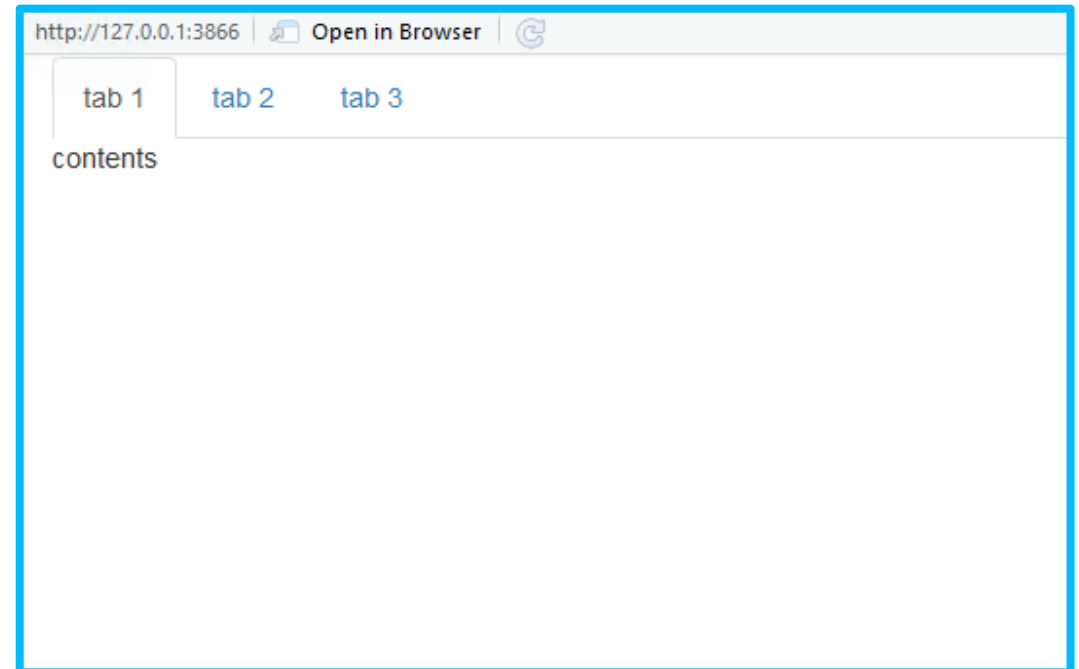


Further functions/ Appearance of the app

- // Use of panels: addition of more plots or additional information
- // Groups elements into a single unit for aesthetic or functional reasons
- // `tabPanel()` creates a stackable panel
- // `tabsetPanel()` arranges tab panels into a stack with tab navigation



```
fluidPage(
  tabsetPanel(
    tabPanel("tab 1", "contents"),
    tabPanel("tab 2", "contents"),
    tabPanel("tab 3", "contents")
  )
)
```



Example – The Framingham Heart Study

```
ui <- fluidPage(
  ...

  mainPanel(width=10,
    tabsetPanel(type = "tabs",
      tabPanel("Plot",
        plotOutput(outputId = "plot",
          height="auto", width="auto")),
      tabPanel("Dataset",
        tableOutput("dataset"))
    )
  )
)
```

```
server <- function(input, output) {
  ...

  output$dataset <- renderTable({
    data1()
  })
}
```

http://127.0.0.1:3866 Open in Browser

The Framingham Heart Study

Dataset and title

File input:

Browse... heart.csv Upload complete

Choose a title for your plot

Plot

Choose the type of plot

Which plot do you like to create?

Choose corresponding variables

Choose a variable for the x-axis

Status

Choose a variable for the y-axis

Status

Size of the plot

Height

700

Width

900

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Stat
Dead	Other	NA	Female	29	62.50	140	78	124	121	0	55	NA		Normal
Dead	Cancer	NA	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High
Alive		NA	Female	57	62.25	132	90	170	114	10	NA	250	High	High
Alive		NA	Female	39	65.75	158	80	128	123	0	NA	242	High	Normal
Alive		NA	Male	42	66.00	156	76	110	116	20	NA	281	High	Optimal
Alive		NA	Female	58	61.75	131	92	176	117	0	NA	196	Desirable	High
Alive		NA	Female	36	64.75	136	80	112	110	15	NA	196	Desirable	Normal
Dead	Other	NA	Male	53	65.50	130	80	114	99	0	77	276	High	Normal
Alive		NA	Male	35	71.00	194	68	132	124	0	NA	211	Borderline	Normal
Dead	Cerebral Vascular Disease	NA	Male	52	62.50	129	78	124	106	5	82	284	High	Normal
Alive		NA	Male	39	66.25	179	76	128	133	30	NA	225	Borderline	Normal
Alive		57	Male	33	64.25	151	68	108	118	0	NA	221	Borderline	Optimal
Alive		55	Male	33	70.00	174	90	142	114	0	NA	188	Desirable	High
Alive		79	Male	57	67.25	165	76	128	118	15	NA	NA		Normal
Alive		66	Male	44	69.00	155	90	130	105	30	NA	292	High	High
Alive		NA	Female	37	64.50	134	76	120	108	10	NA	196	Desirable	Normal
Alive		NA	Male	40	66.25	151	72	132	112	30	NA	192	Desirable	Normal
Dead	Cancer	56	Male	56	67.25	122	72	120	87	15	72	194	Desirable	Normal
Alive		NA	Female	42	67.75	162	96	138	119	1	NA	200	Borderline	High
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High



Overall appearance of the app - shinythemes

// R package shinythemes can be used to change the overall appearance of the app

// Add the selected theme at the beginning of the program for the user interface:

```
library(shinythemes)
```

```
fluidPage(theme = shinytheme("THEME"),...)
```

THEME = SUPERHERO:

// Different themes available

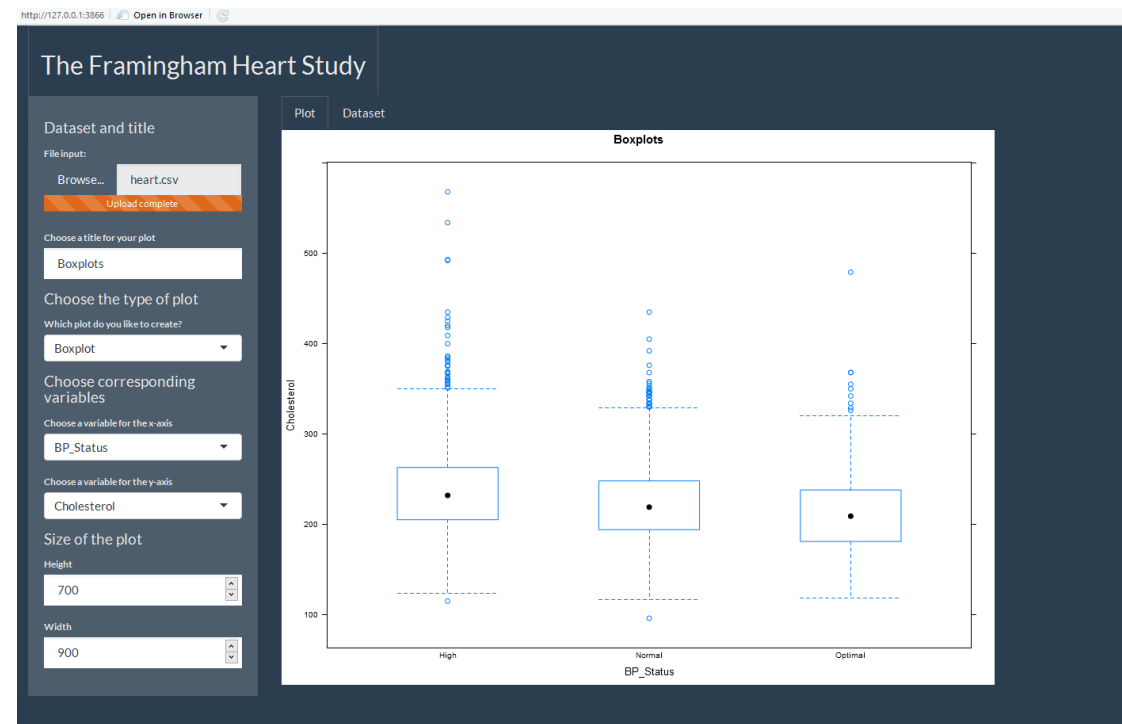
(<https://rstudio.github.io/shinythemes/>):

// CERULEAN

// COSMO

// CYBORG

// ...





Summary and discussion



Summary and discussion – R Shiny

- // Efficient tool to create interactive web applications without knowing any HTML, but just the programming language R
- // Two components of a Shiny app: the user interface and the server function
- // Web applications can be helpful to support non-statisticians and non-programmers to explore the data on their own without knowing R
- // Flexibility and the layout of the app can be chosen by the programmer
- // The more flexible the app shall be, the more complex the R code becomes and the transparency of the app can decrease
 - // Worth to draw up easy-handling requirements for the app → App still useful for the investigator and programmable for the developer



References

- // RStudio (2017). *Gallery*. <https://shiny.rstudio.com/gallery/> (from 04.10.2019)
- // RStudio (2017). *Learn Shiny*. <https://shiny.rstudio.com/tutorial/> (from 30.09.2019)
- // RStudio (2014). *Shiny Themes*. <https://rstudio.github.io/shinythemes/> (from 04.10.2019)
- // SAS Institute Inc. (2017). *Sashelp Data Sets*. Cary, NC, USA.



Thank you!



Questions?

