How to create applications using Shiny

ISGlobal

Barcelona, February 13-14th 2018

Part IV: Ways to improve the application

Outline

Part IV: Ways to improve the application

- HTML and CSS[>]
- Pop-ups and Modals[>]
- Collapse panels[>]
- Carrouselles[>]
- Themes (app appearance)[>]
- Sizeable[>]
- Input alerts[>]
- Exercise[>]

HTML and CSS

HTML and CSS

- Using **HTML** and **div** you can insert HTML code to change the form elements appearance, color, etc., insert links,...
- You can specify the element appearance in the header by using HTML function with #age linking to the element whose id is "age".

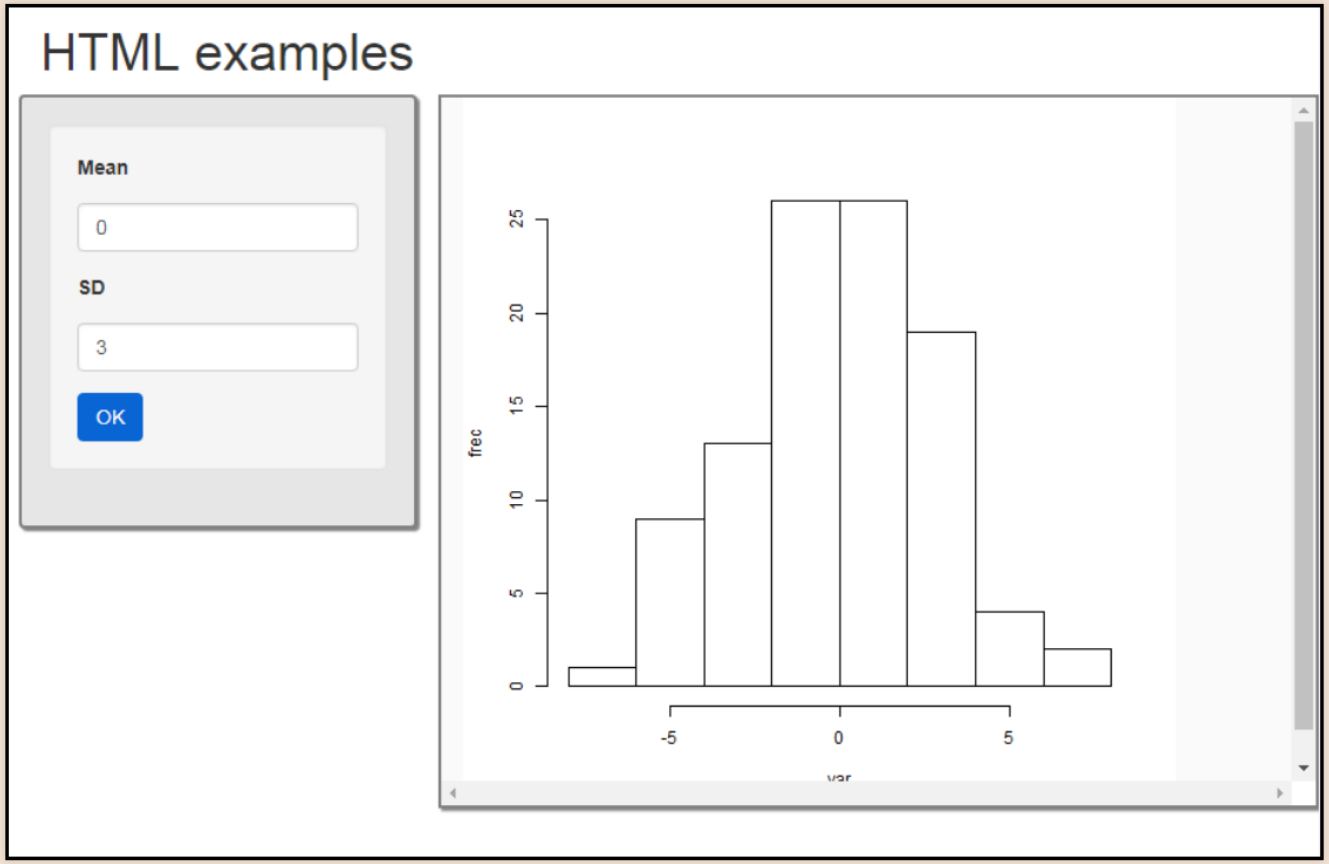
```
HTML("<style type='text/css'> #age{color: red} </style>")
```

- Or just wrapping the element by **div** function in the UI section.

```
div(numericInput("age","Age",30), style="color:red")"
```

- You can also use **HTML** to change the color or insert a link to an input element label.

```
numericInput("age", HTML("<p title='Enter your age'>Age</p>"),30))"
```



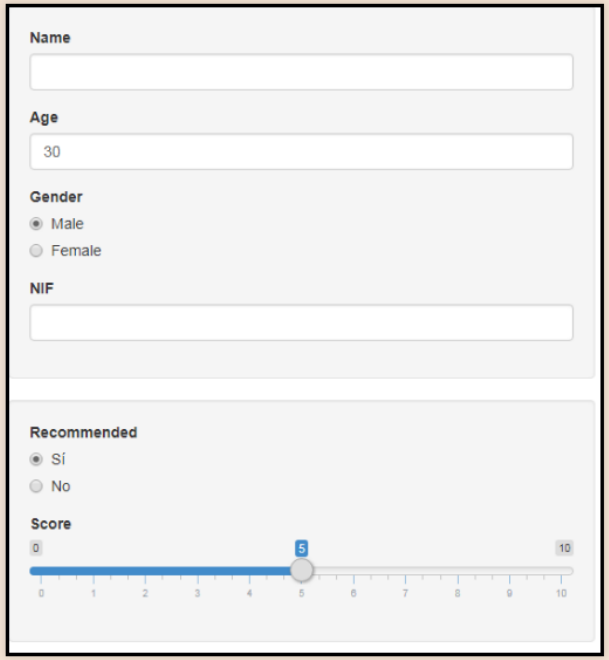
Note what happens when placing the mouse over “mu” or “sd” inputs label.

```
ui <- fluidPage(
  headerPanel("HTML examples"),

  HTML("<style type='text/css'> #inputpanel{background-color: rgb(230,230,230);
       border: 2px solid grey; box-shadow: 2px 2px 1px #888888;} </style>"),
  HTML("<style type='text/css'> #outpanel { background-color: rgb(250,250,250);
       border: 2px solid grey; box-shadow: 2px 2px 1px #888888; overflow:scroll; height:500px}
</style>"),
  HTML("<style type='text/css'> #inputpanel .wellPanel {background-color:rgb(215,215,215);
</style>"),
  HTML("<style type='text/css'> #OK {color:white; background-color:rgb(10,101,212);
       border: solid 1px rgb(10,101,212)} </style>"),

  sidebarLayout(
    sidebarPanel(id="inputpanel",
      wellPanel(
        numericInput("mu", HTML("<p title='Enter the mean'>Mean</p>"), 0),
        numericInput("sd", HTML("<p title='Enter the standard deviation'>SD</p>"),3),
        actionButton("OK", "OK")
      )
    ),
    mainPanel(id="outpanel",
      plotOutput("results")
    )
  )
)

server <- function(input,output){
  output$results <- renderPlot({
    if (input$OK == 0) return(invisible(NULL))
    isolate({
      hist(rnorm(100, input$mu, input$sd), xlab="var", ylab = "frec", main = "")
    })
  }, 500, 500)
}

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

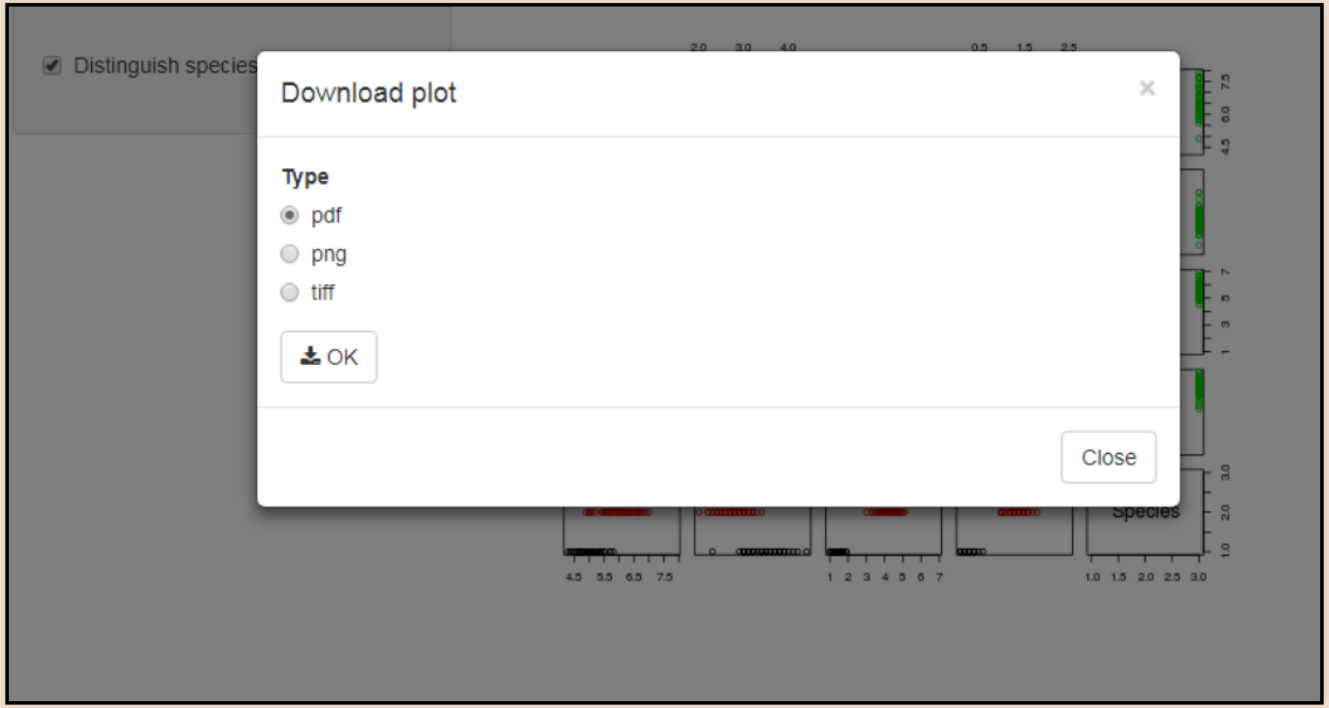
Pop-ups and Modals

Pop-ups

```
library(shinyBS)

ui <- fluidPage(
  wellPanel(id="person",
    textInput("name", "Name", ""),
    numericInput("age", "Age", 30),
    radioButtons("gender", "Gender", c("Male",
"Female")),
    textInput("dni", "NIF", "")
  ),
  wellPanel(id="product",
    radioButtons("recom", "Recommended", c("Yes", "No")),
    sliderInput("score", "Score", 0, 10, 5)
  ),

  bsTooltip("name", "Write your name and surname"),
  bsTooltip("age", "Enter your age"),
  bsTooltip("dni", "Type your DNI including the letter"),
  bsTooltip("recom", "Would you recommend it?"),
  bsTooltip("score", "Score from 0 to 10", "bottom"),
  bsTooltip("person", "Personal data form", "click"),
  bsTooltip("product", "Product form", "click")

)

server <- function(input, output, session) {}

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



Modals


Distinguish species
Download plot
Type
pdf
png
tiff
OK
Close
Species

UI:

```
library(shinyBS)

ui <- fluidPage(

  sidebarLayout(

    sidebarPanel(
      checkboxInput("groups",
                    "Distinguish species")
    ),

    mainPanel(
      plotOutput("plot"),

      bsModal("modal",
              "Download plot",
              "plot",
        radioButtons("type", "Type",
                     c("pdf", "png", "tiff")),
        downloadButton("down", "OK")
      )

    )
  )
)
```

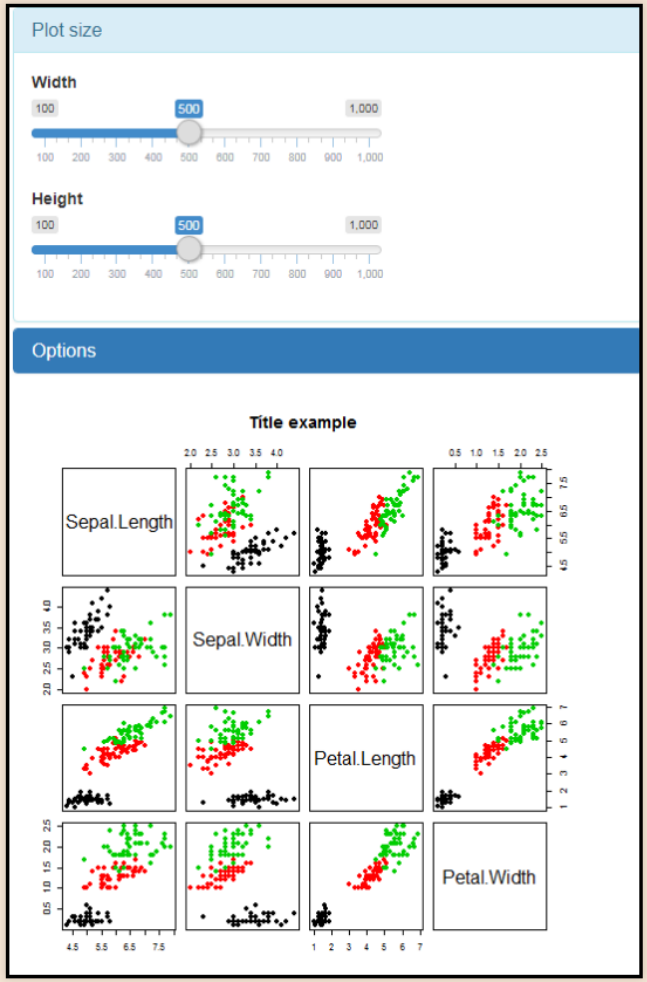
Server:

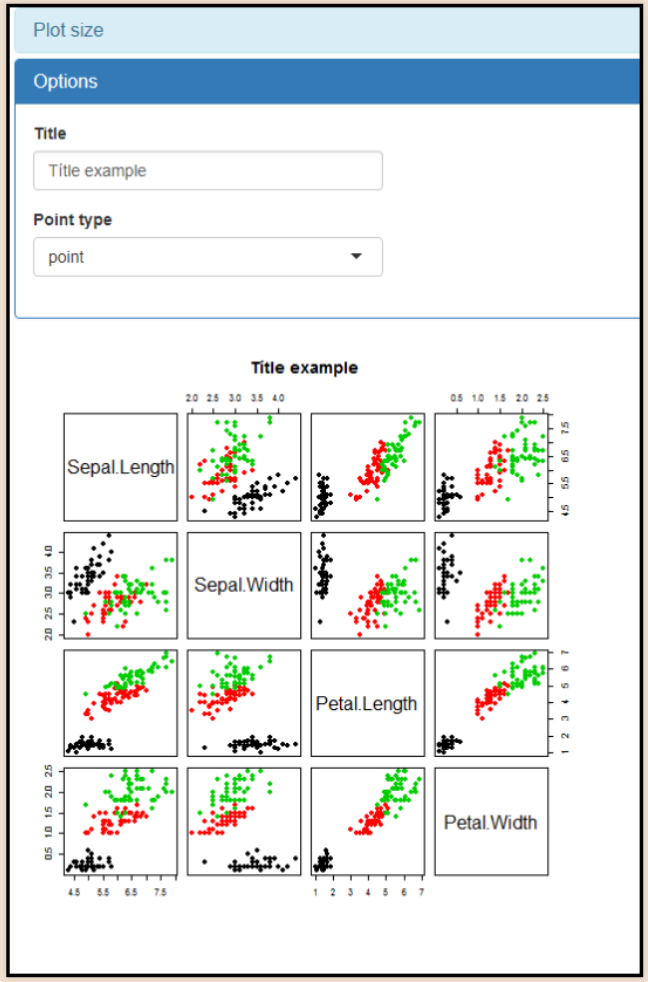
```
server <- function(input, output) {

   output$plot <- renderPlot({
     if (input$groups)
       pairs(iris, col = iris[,5])
     else
       pairs(iris)
   }, width = 500)

   output$down <- downloadHandler(
     filename = function(){
       paste("figure", input$type, sep = ".")
     },
     content = function(ff){
      if (input$type == "pdf") pdf(ff)
      if (input$type == "png") png(ff)
      if (input$type == "tiff") tiff(ff)
      if (input$groups)
        pairs(iris, col = iris[,5])
      else
        pairs(iris)
      dev.off()
     }
   )
}
```

Exercise: Create a pop-up when placing the mouse over the plot saying “save it”.

Collapse panels

Collapse panels





```r
library(shinyBS)
ui <- fluidPage(

  bsCollapse(id = "collapseExample", open = "Plot size",
    bsCollapsePanel("Plot size",
      sliderInput("width", "Width", 100, 1000, 500, 50),
      sliderInput("height", "Height", 100, 1000, 500, 50)
    , style = "info"),
    bsCollapsePanel("Options",
      textInput("main", "Title", "title"),
      selectInput("pch", "Point type",
                  c("point"=19, "squared"=22, "diamond"=23))
    , style = "primary")
  ),

  uiOutput("result")
)

server <- function(input, output) {
  output$plot <- renderPlot({
    pairs(iris[,-5], col=iris[,5],
          main = input$main,
          pch = as.double(input$pch))
  })
  output$result <- renderUI({
    plotOutput("plot", width=input$width, height=input$height)
  })
}

runApp(list(ui=ui,server=server))
```

Exercise: change the `multiple` argument from `bsCollapse` function to `TRUE`.

Carrouselles

Carrouselles

- The **bsplus** packages allows to create carrouselles, among other features.
- This package is available on CRAN repository:

```
install.packages(bsplus)
```

- Or install the last vresion from Github repository

```
install.packages("devtools")
devtools::install_github("ijlyttle/bsplus")
```

- Visit **bsplus** webpage here

--------Summary descriptives table by 'Intervention group'---------

	Control N=2042	MedDiet + Nuts N=2100	MedDiet + VOO N=2182	p-value
Sex:				<0.001
Male	39.8%	46.1%	41.2%	
Female	60.2%	53.9%	58.8%	
Age	67.3±6.28	66.7±6.02	67.0±6.21	0.003
Smoking:				0.444
Never	62.8%	60.0%	61.9%	
Current	13.2%	14.1%	13.4%	
Former	24.0%	26.0%	24.7%	
Body mass index	30.3±3.96	29.7±3.77	29.9±3.71	<0.001
Waist circumference	101±10.8	100±10.6	100±10.4	0.045
Waist-to-height ratio	0.63±0.07	0.62±0.06	0.63±0.06	<0.001
Hypertension	83.8%	82.8%	81.9%	0.249
Type-2 diabetes	47.5%	45.2%	49.6%	0.017
Dyslipidemia	72.4%	73.3%	71.5%	0.423
Family history of premature CHD	22.6%	21.9%	23.2%	0.581
Hormone-replacement therapy	1.68%	1.61%	1.84%	0.850
MeDiet Adherence score	8.44±1.94	8.81±1.90	8.77±1.97	<0.001
follow-up to main event (years)	4.16 [2.72;5.62]	4.72 [2.80;5.76]	5.02 [3.41;5.88]	<0.001
AMI, stroke, or CV Death	4.75%	3.33%	3.90%	0.064

```r
library(bsplus); library(magrittr)

ui <- fluidPage(
  titlePanel("Carrouselles example"),
  HTML("<style type='text/css'> #carousel{height:700px;} </style>"),
  bs_carousel(id="carousel", use_indicators=TRUE, use_controls=TRUE)%>%
  bs_append(content=bs_carousel_image(src="example1.png", width="50%"))%>%
  bs_append(content=bs_carousel_image(src="example2.png", width="40%"))%>%
  bs_append(content=bs_carousel_image(src="example3.png", width="40%"))%>%
  bs_append(content=bs_carousel_image(src="example4.png", width="40%"))
)

server <- function(input, output) {}

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

Exercise: Reprodue this example with four images files stored inside www folder

Themes

Themes

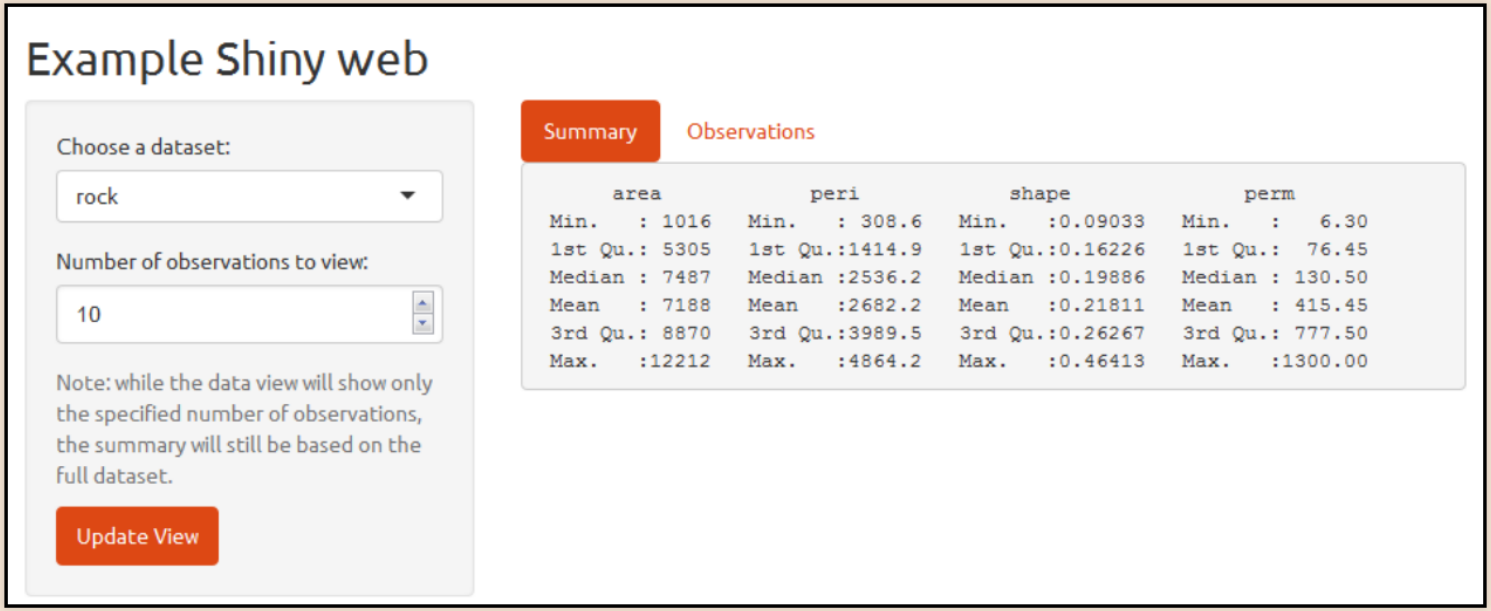
- Using the **`shinythemes`** package you can change the app appearance very easily.
- This package contains a collection of CSS themes.
- The theme is chosen thru **`shinytheme`** function in `themes` argument of `fluidPage` function.
- The available themes are: “cerulean”, “cosmo”, “flatly”, “journal”, “readable”, “spacelab”, “united”
- Alternatively to **shinythemes**, you can costumize the app appearance by writting a CSS file which must be stored inside `www` folder. Then you specify the CSS file name to `theme` argument of `fluidPage` function.
- See all themes in this website

UI

```
library(shinythemes)

ui <- fluidPage(

  # specify the CSS theme or CSS file
  theme = shinytheme("united"),

  titlePanel("Example Shiny web"),
  sidebarLayout(
    sidebarPanel(
      selectInput("dataset", "Choose a dataset:",
                  c("rock", "pressure", "cars")),
      numericInput("obs",
              "Number of observations to view:",
10),
      helpText("Note: while the data view will
               show only the specified",
               "number of observations,
               the summary will still be based",
               "on the full dataset."),
      submitButton("Update View")
    ),
    mainPanel(
      tabsetPanel(type="pills",
        tabPanel("Summary",
          verbatimTextOutput("summary")
        ),
        tabPanel("Observations",
          tableOutput("view")
        )
      )
    )
  )
)
```

Server

```
server <- function(input, output) {
  datasetInput <- reactive({
    switch(input$dataset,
           "rock"=rock,
           "pressure"=pressure,
           "cars"=cars)
  })
  output$summary <- renderPrint({
    dataset <- datasetInput()
    summary(dataset)
  })
  output$view <- renderTable({
    head(datasetInput(), n = input$obs)
  })
}
```



Exercises:

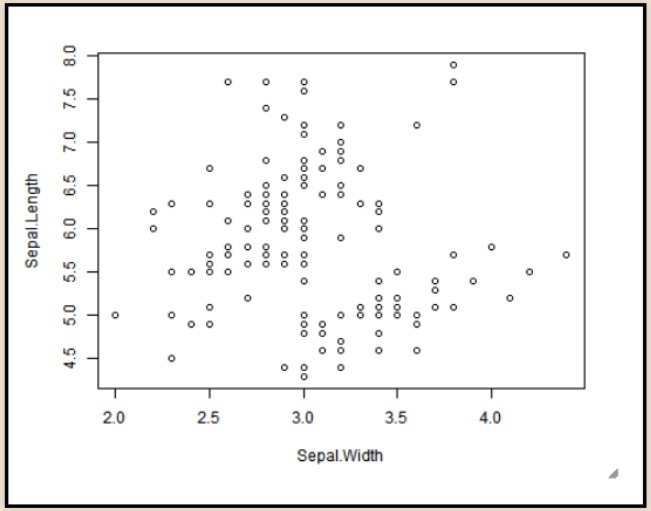
- Change the argument `theme` to other theme.
- Specify a CSS file from http://bootswatch.com/

Sizeable

Sizeable

- Using the **jqui_resizabled** function from the **shinyjqui** package, you can resize interactively any widget (like a plot, input element or panel).
- This package is available on CRAN and it also contains functions to drag or sort widgets.

```
library(shinyjqui)

ui <- fluidPage(

  jqui_resizabled(
    plotOutput('plot', '200px', '200px')
  )

)

server <- function(input, output) {
  output$plot <- renderPlot({
    plot(Sepal.Length ~ Sepal.Width, iris)
  })
}

shinyApp(ui, server)
```



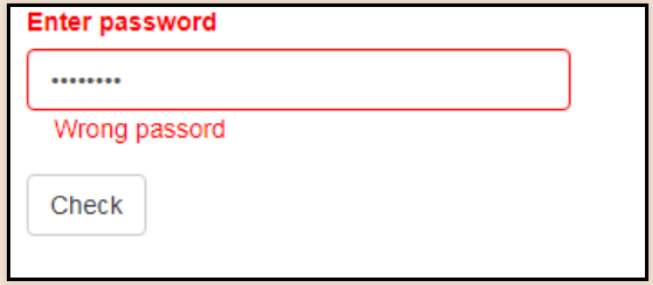
Exercise: Place a download button to save the plot.

Input alerts

Input alerts

- Using the **feedback** from the **shinyFeedback** package you can create alerts (with a message and colour) beside an input widget.
- Its arguments are the contition, text and color.

```
require(shinyFeedback)

ui <- fluidPage(
  useShinyFeedback(),
  passwordInput("pass","Enter password"),
  actionButton("check","Check")
)

server <- function(input, output) {
 observeEvent(input$check, {
   feedback("pass",
            condition=input$pass!="123",
            text="Wrong passord",
            color="red")
  })
}

shinyApp(ui, server)
```



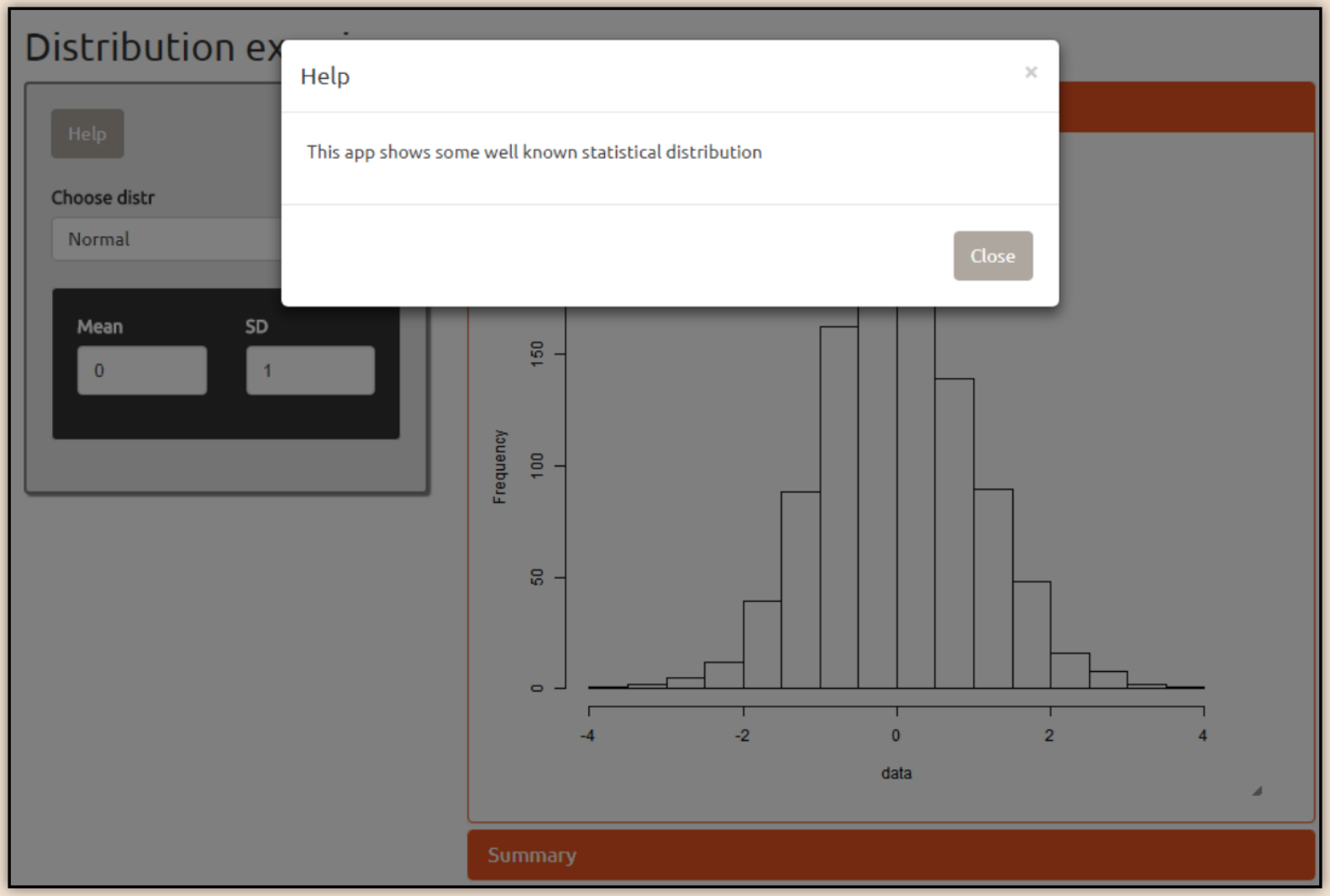
Exercise: Repeat this example for a `numericInput` like age, with 0-100 as range, and introducing the proper alert texts. Note that you can place more than one `feedback` (one for each condition).

Exercise

Exercise

From the app created in part II:

- Add pop-ups.
- Modify some input element or panel appearance.
- Create collapse panels.
- Change the app appearance using `shinythemes`.
- Make the outputs (plot, summaries) resizable.
- Create an alert for parameters numericInputs.


Distribution ex
Help
Choose distr
Normal
Mean
0
SD
1
Help
This app shows some well known statistical distribution
Close
Frequency
0
50
100
150
-4
-2
0
2
4
data
Summary