Creating Shiny Apps for biostatisticians and bioinformaticians

ISGlobal

21-22 May 2019

Isaac Subirana

Part IV: How to improve the appearance and functionality of the application

Outline

Part IV: Ways to improve the application

- HTML and CSS
- Pop-ups and Modals
- Collapse panels
- Themes (app appearance)
- Sizeable
- Input alerts
- Loader & progress bar
- 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".

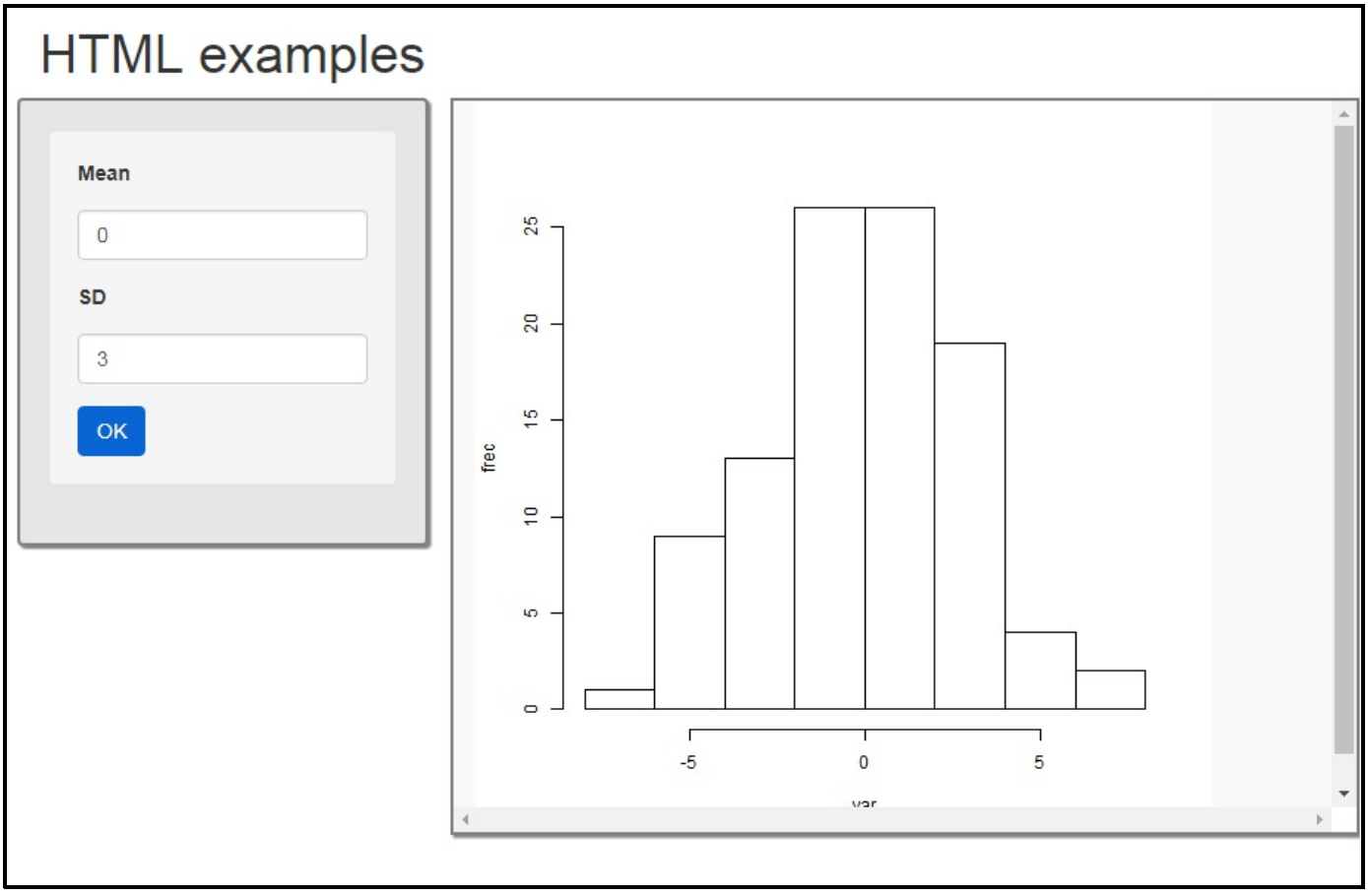
```
HTML("<style type='text/css'> #age{color: red} </style>")
```

- 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))
```



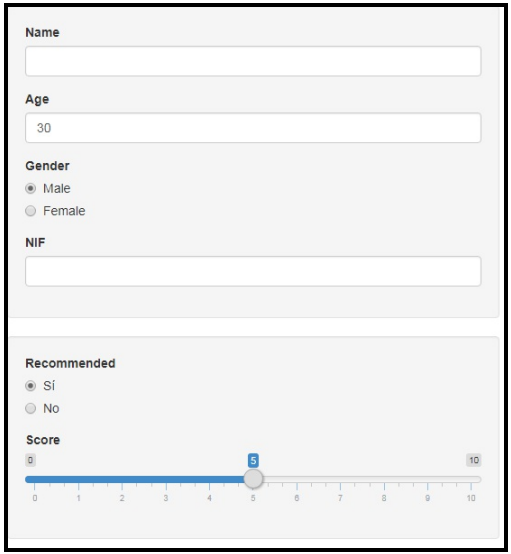
Note what happens when placing the mouse over "mu" or "sd" inputs label.

```
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)
```

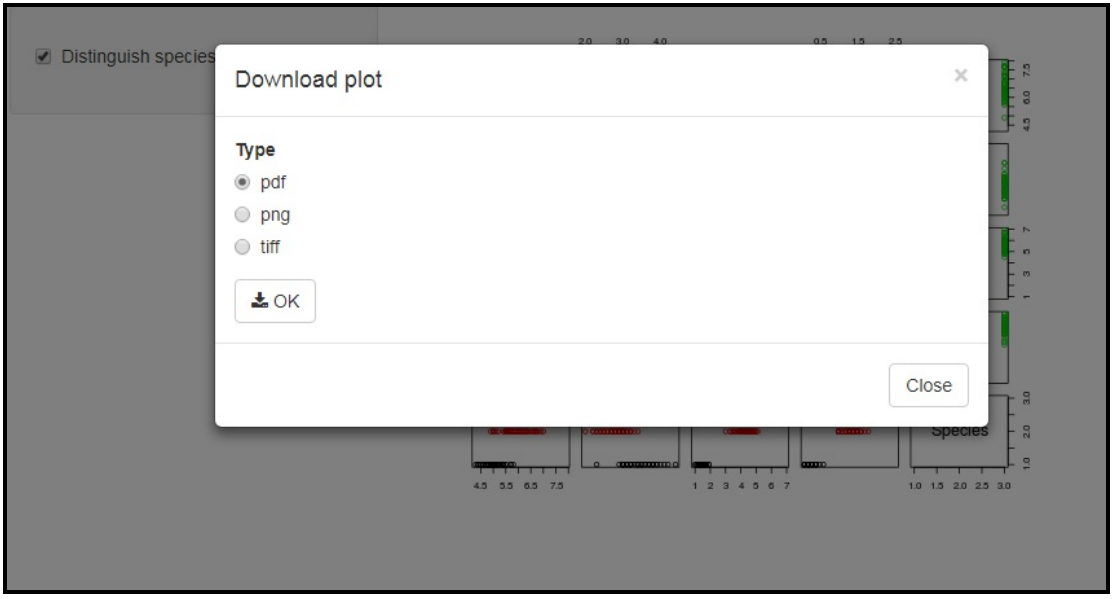
Pop-ups and Modals

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:

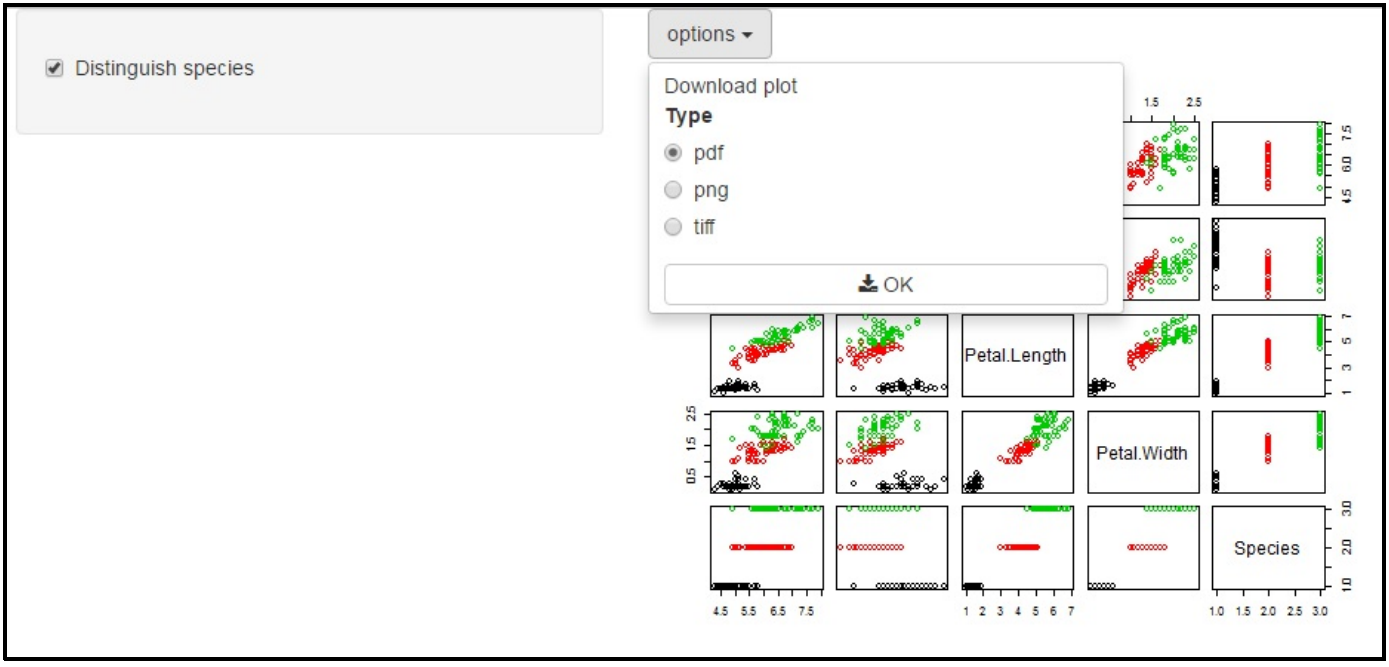
```
library(shinyBS)

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".

Modals (shinyWidgets)



UI:

```
library(shinyWidgets)

ui <- fluidPage(

  sidebarLayout(

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

    mainPanel(
      dropdownButton(circle=FALSE,
        label="options",
        "Download plot",
        radioButtons("type", "Type",
                     c("pdf", "png", "tiff")),
        downloadButton("down", "OK")
      ),
      plotOutput("plot")
    )
  )
)
```

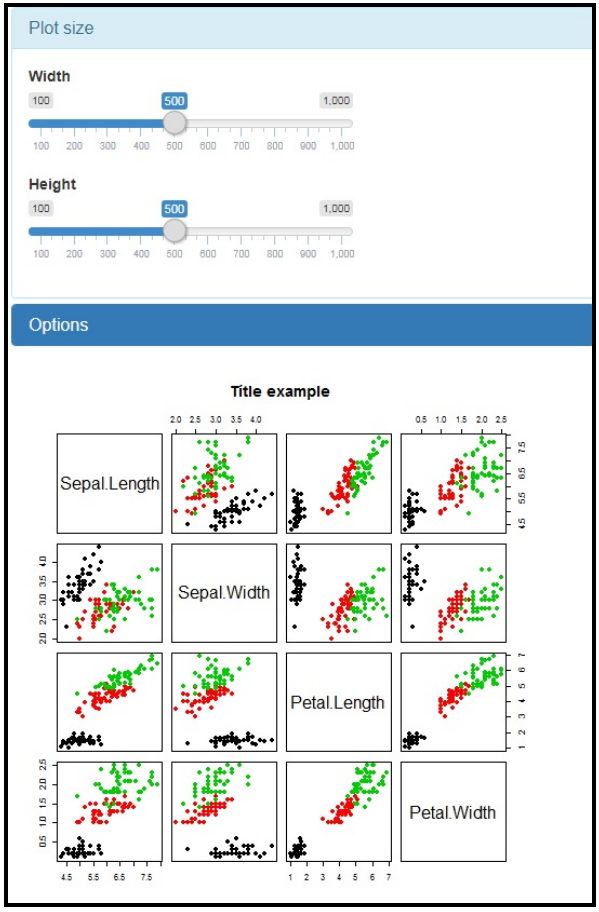
Server:

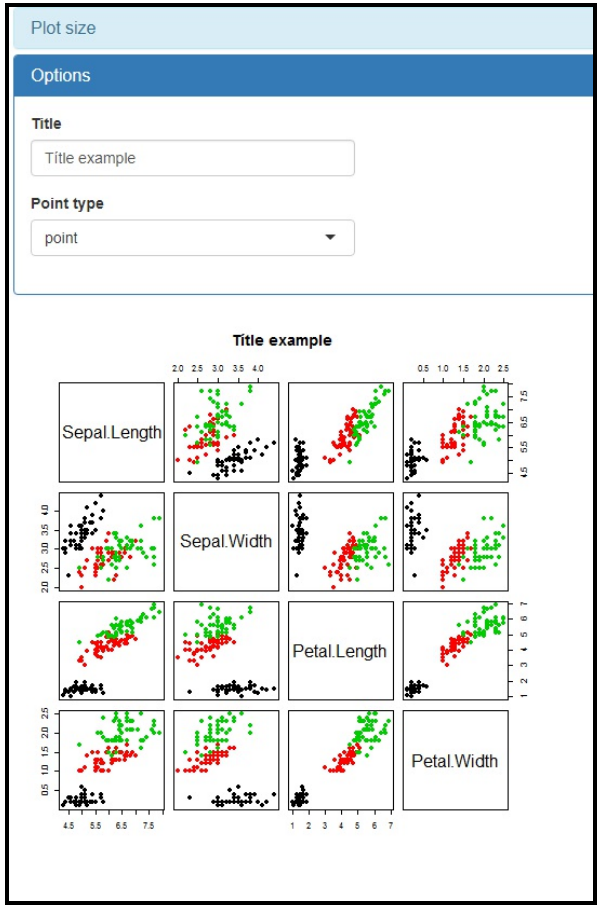
```
server <- function(input, output) {

   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()
     }
   )
}
```

Note that it creates a button. When clicking on it a panel is dropped down.

Collapse panels

Collapse panels





```
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`.

Themes

Themes

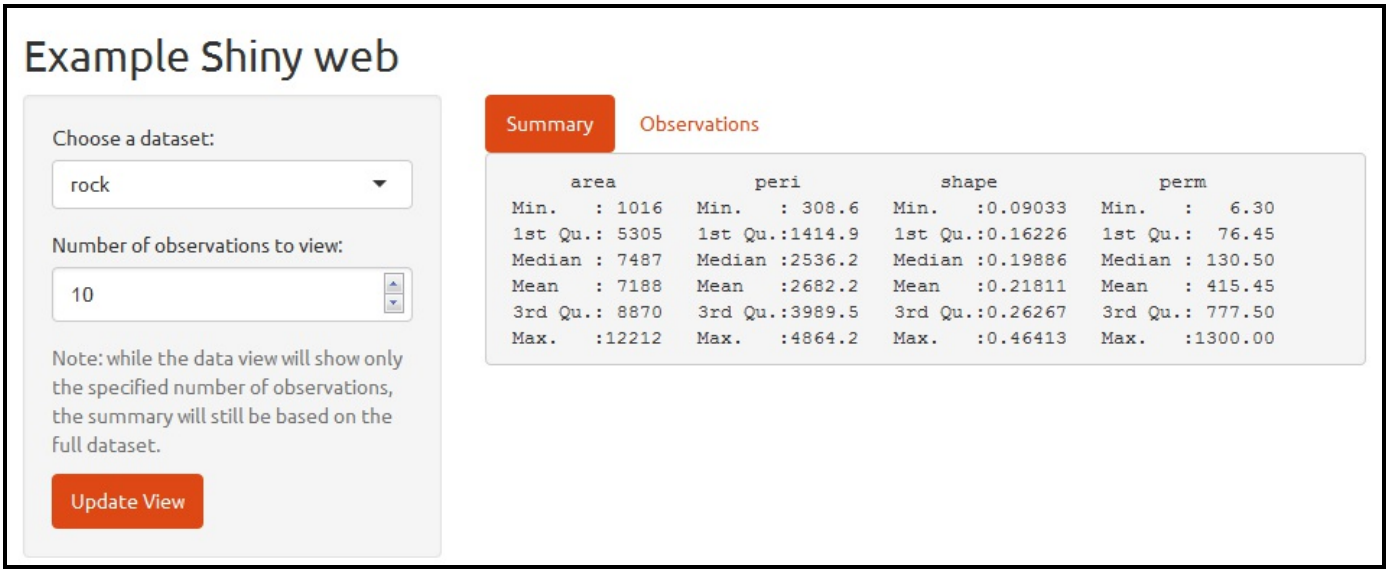
- Using the **`shinythemes`** package you can change the app appearance very easily.
- This package contains a collection of CSS themes.
- The theme is chosen thru **`shinytheme`** function in **`themes`** argument of **`fluidPage`** function.
- The available themes are: "cerulean", "cosmo", "flatly", "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(
  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/

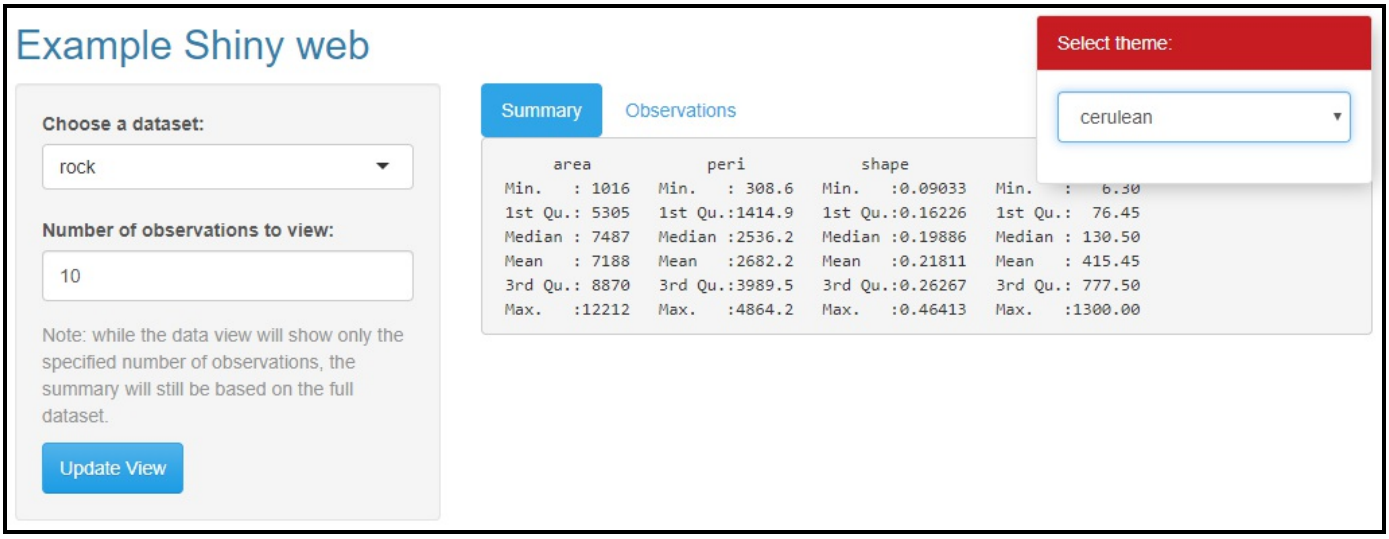
It is also possible to choose theme interactively with **`themeSelector()`**.

UI

```
ui <- fluidPage(
  shinythemes::themeSelector(),
  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)
  })
}
```

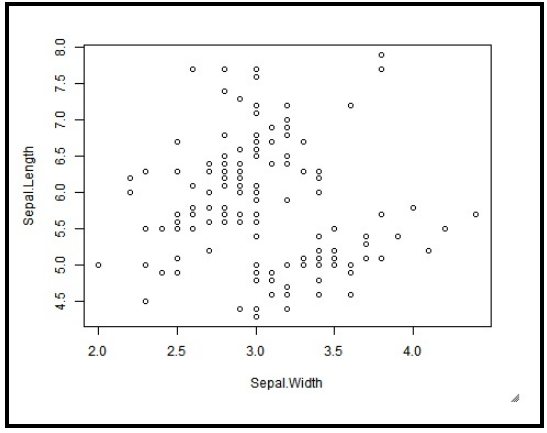

Example Shiny web
Select theme:
cerulean
Choose a dataset:
rock
Number of observations to view:
10
Note: while the data view will show only the specified number of observations, the summary will still be based on the full dataset.
Update View
Summary
Observations
area peri shape
Min. : 1016 Min. : 308.6 Min. :0.09033 Min. : 6.30
1st Qu.: 5305 1st Qu.:1414.9 1st Qu.:0.16226 1st Qu.: 76.45
Median : 7487 Median :2536.2 Median :0.19886 Median : 130.50
Mean : 7188 Mean :2682.2 Mean :0.21811 Mean : 415.45
3rd Qu.: 8870 3rd Qu.:3989.5 3rd Qu.:0.26267 3rd Qu.: 777.50
Max. :12212 Max. :4864.2 Max. :0.46413 Max. :1300.00

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)
```



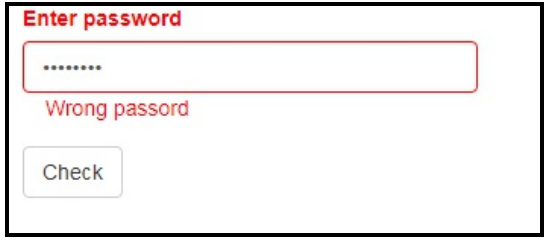
Exercise: Place a download button to save the plot.

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)
```

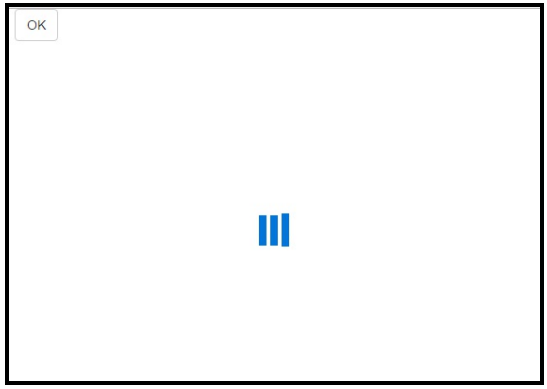


Loader & progress bar

Loader

- Use the **`withSpinner`** from the **`shinycssloaders`** package to show loader when computing time consuming processes when rendering a plot or a table, etc.

```
library(devtools)
install_github('andrewsali/shinycssloaders')
library(shinycssloaders)

ui <- fluidPage(
  actionButton("ok","OK"),
  withSpinner(plotOutput("plot"), type=1)
)

server <- function(input, output){
  output$plot <- renderPlot({
    if (input$ok==0) return(invisible(NULL))
    isolate({
      Sys.sleep(10)
      plot(1)
    })
  })
}

shinyApp(ui, server)
```

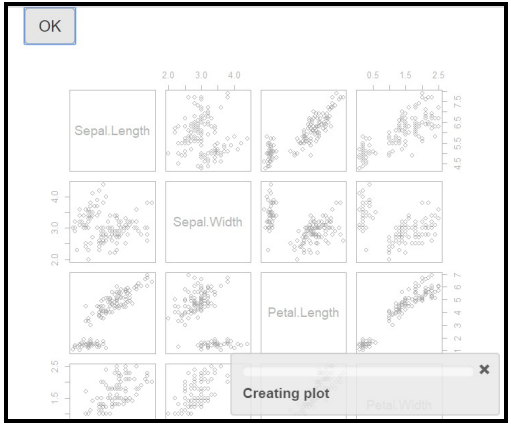


Exercise: Change **`type`** argument.

Progress bar

- Use **`withProgress`** to create a progress bar.
- It is specified in **server** section, when rendering or creating a reactive object or in any process, that takes a while.

```
ui <- fluidPage(
  actionButton("ok","OK"),
  plotOutput("plot")
)

server <- function(input, output){
  output$plot <- renderPlot({
    if (input$ok==0) return(invisible(NULL))
    isolate({
      withProgress(message='Creating plot',value=0,{
        Sys.sleep(5)
        pairs(iris[,1:4])
        incProgress(amount=1)
      })
    })
  })
}

shinyApp(ui, server)
```

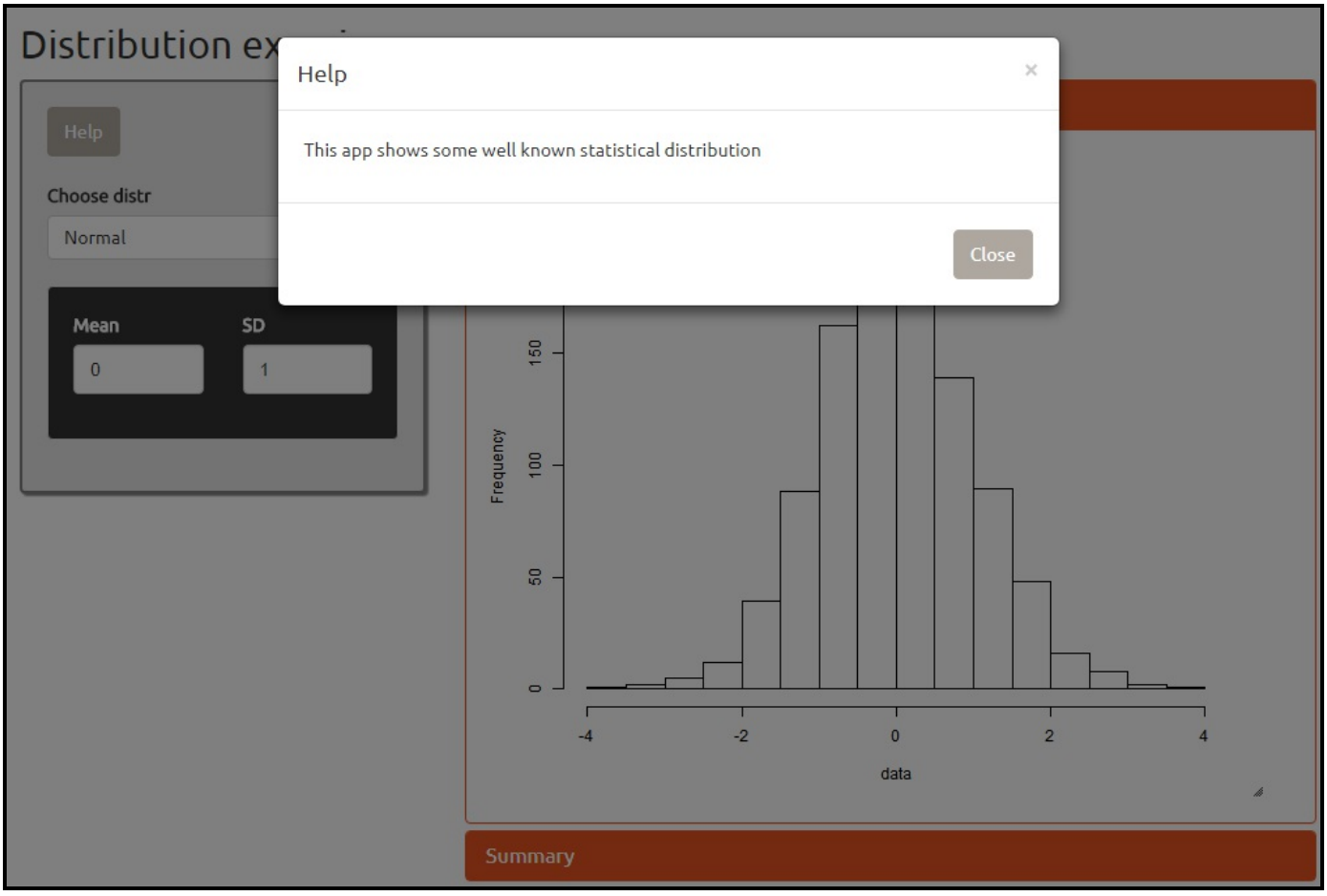


Exercise

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