

# Data Visualization with R Shiny tutorial

Ariane Ducellier

University of Washington - Fall 2023

# Improving the UI - Using shiny themes

```
library(shinythemes)
fluidpage(theme=shinytheme("darkly"),
...)
```

If you want the user to be able to change the theme:

```
library(shinythemes)
fluidpage(theme=shinytheme("darkly"),
          themeSelector(),
...)
```

See a list of themes here: <http://rstudio.github.io/shinythemes/>

# Improving the UI - Using the grid layout

```
fluidPage(title="...",  
  fluidRow(  
    column(6,  
      wellPanel(  
        sliderInput( ... )))  
    column(6, ... )  
  ),  
  ...  
)
```

The sum of the widths of the columns must be 12. `wellPanel` creates a panel around the slider. `hr()` creates a horizontal rule to break the screen.

# Downloading plots

In the file `ui.R`:

```
downloadButton("downloadPlot",  
               label = "Download plot")
```

In the file `server.R`:

```
thePlot <- reactive( ... code to make plot ... )  
output$downloadPlot <- downloadHandler(  
  filename <- function() {"filename"},  
  content <- function(file){  
    png(file, width=980, height=400, ... )  
    iris.plot <- thePlot()  
    print(iris.plot)  
    dev.off()  
  },  
  contentType = "image/png"  
)
```

# Downloading data

In the file `ui.R`:

```
downloadButton("downloadData",  
               label = "Download data")
```

In the file `server.R`:

```
theData <- reactive( ... code to produce data ... )  
output$downloadData <- downloadHandler(  
  filename = function(){ "iris.csv" },  
  content <- function(file){  
    write.csv(theData(), file)  
  },  
  contentType = "text/csv"  
)
```

# Interactive plots - Click points

In the file `ui.R`:

```
plotOutput("plot", click = "plot_click"),  
            tableOutput("plot_clickedpoints")
```

In the file `server.R`:

```
output$plot_clickedpoints <- renderTable({  
  res <- nearPoints(iris,  
                    input$plot_click,  
                    "Sepal.Length",  
                    "Sepal.Width")  
  if (nrow(res) == 0)  
    return()  
  res  
})
```

# Interactive plots - Hover over plot

In the file `ui.R`:

```
plotOutput("plot",  
           hover = hoverOpts(id = "plot_hover",  
                             delayType = "throttle")  
)
```

In the file `server.R`:

```
output$plot_hoverinfo <- renderPrint({  
  cat("Hover (throttled):\n")  
  str(input$plot_hover)  
})
```