Data Visualization with R
Shiny tutorial

Ariane Ducellier

University of Washington - Fall 2025

Progress bar

If some computation in `server.R` can take a long time, it is useful to wrap the corresponding code inside the Shiny `withProgress()` function.

In `server.R`

```
withProgress(message = ... ,
  detail = ' ... ', value = 0,
  ... function code ...
  incProgress(1/3)
  ... function code ...
  incProgress(1/3)
  ... function code ...
  incProgress(1/3)
  ... function code ...
})
```

R Flexdashboard

```
Go to File >
      New File >
      R Markdown >
      From Template >
      Flex Dashboard
```

Click on `Knit` to see the empty dashboard.

R Flexdashboard

In the first R block, load the libraries and the data:

```
library(flexdashboard)
library(tidyverse)
library(leaflet)
load("geocodedData.Rdata")
```

Change the names of the R Markdown headers and fill the R blocks with the code from `dashboard1.Rmd`.

Click on `Knit` to see the final dashboard.

Adding shiny to the flexdashboard

Modify the header by adding shiny and using a `rows` orientation:

```
title: "Flexdashboard 2"
runtime: shiny
```

We will add one sidebar column:

```
Column {.sidebar}
```

We fill the R block with R shiny code to create a slider and a checkbox as done previously in `ui.R`.

Adding shiny to the flexdashboard

Create a simple row and a row with several tabs:

```
Row
```

```
Row {.tabset}
```

We fill the R block with R shiny code to create plots as done previously in `server.R`.

In this case, the filtering is done for every block of R code. We cannot define a reactive object to filter the years.

Shiny dashboards

```
library(shinydashboard)
header <- dashboardHeader( )
sidebar <- dashboardSidebar()
body <- dashboardBody()
dashboardPage(header, sidebar, body,
  title = NULL,
  skin = c("blue", "black", "purple", "green", "red",
           "yellow"))
```

Adding a menu to the sidebar

```
sidebarMenu(id = NULL,
  menuItem("Name",
    icon = ... ,
    tabName = ... ,
    badgeLabel = ... ,
    badgeColor = ... ,
    ...
  ),
  sliderInput( ... )
)
```

`tabName` will be referred to in the dashboard body to create the corresponding graph.

Improving the UI - Adding a menu to the sidebar

```
tabItems(
  tabItem(tabName = ... ,
    fluidRow(
      box(width = 10,
          plotOutput("trend"),
          checkboxInput( ... )),
          box(width = 2, ... )
    ),
  )
)
```

`tabName` corresponds to the value given in `menuItem` in the sidebar.

Improving the UI - Adding info boxes

In the file `ui.R`:

```
infoBoxOutput(width = 3, "infoYears")
```

In the file `server.R`:

```
output$infoYears = renderInfoBox({
  infoBox(title,
    value = NULL,
    icon = ... ,
    color = ... ,
    fill = ...
  )
)
```

Improving the UI - Adding icons

```
tabPanel("Trend",
         plotOutput("trend"),
         icon = icon("calendar"))
```

```
tabPanel("Summary",
        textOutput("summary"),
        icon = icon("user", lib = "glyphicon"))
```

See a list of icons here:

- https://fontawesome.com/icons
- https://icons.getbootstrap.com/

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, ... ))
  hr(),
  ...
)
```

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)
})
```

Sharing with Gist

Go to https://gist.github.com/.

If you have a GitHub account, you should have on account on Gist too.

Create a project with a description, an `ui.R` and a `server.R` files.

Get the URL of your project.

In RStudio, run:

```
library(shiny)
runGist("https://gist.github.com/MyName/identifier")
```

Sharing with GitHub

On GitHub, create a repository with your dataset, and the `ui.R` and `server.R` files.

In rStudio, run:

```
library(shiny)
runGitHub("repository_name", "user_name")
```

Sharing through Shinyapps.io

Create a free account on https://www.shinyapps.io/.

In RStudio, install the package `rsconnect`.

When creating your account, `Shinyapps.io` will ask you to set your `Shinyapps.io` account information on RStudio by running:

```
rsconnect::setAccountInfo(name='yourname',
token='some_token', secret='some_secret')
```

To deploy your application on `Shinyapps.io`, run on RStudio:

```
library(rsconnect)
rsconnect::deployApp("/path/your_path_to_your_app")
```

Check on `Shinyapps.io` the URL of your application:
https://arianeducellier.shinyapps.io/magnitudes/