Exercise IV – R Shiny

Part 3 – Shiny App UI

CEE412 / CET522

TRANSPORTATION DATA MANAGEMENT AND VISUALIZATION

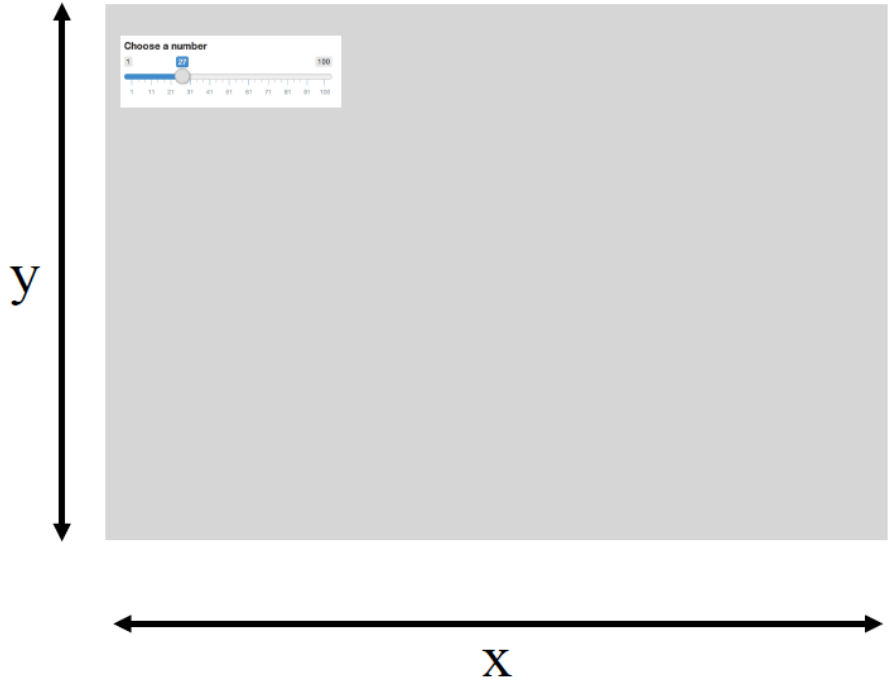
WINTER 2020

Shiny App UI

- In this section, we will introduce
  - How to design the layout of the inputs and outputs
  - How to add image
  - How to use a dashboard as your user interface
- Please note: some of the demos, figures, and screenshot in this slide come from the Shiny tutorial: https://github.com/rstudio-education/shiny.rstudio.com-tutorial

Layout

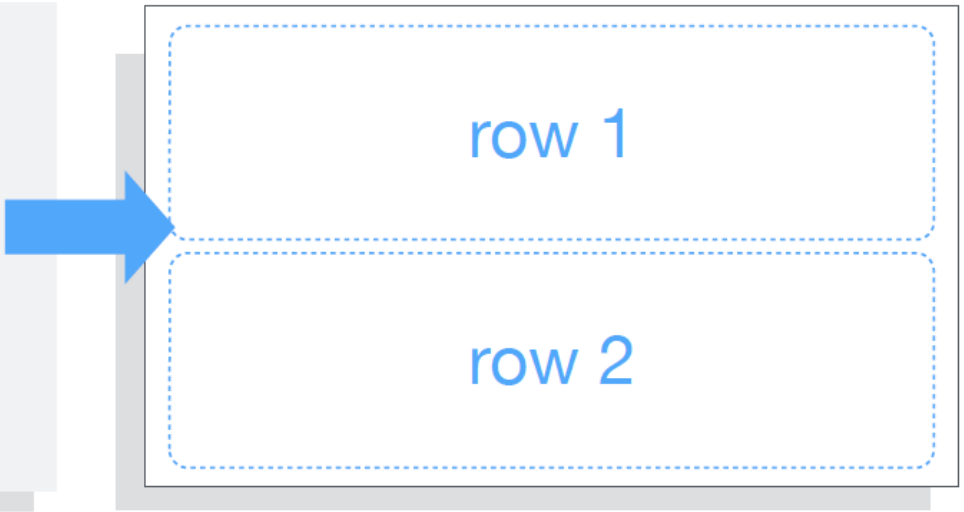
- Use layout functions to position elements within your app
  - From the perspective of y axis, use **fluidRow()**
  - From the perspective of x axis, use **column()**



fluidRow()

- fluidRow() adds rows to the grid. Each new row goes below the previous rows.

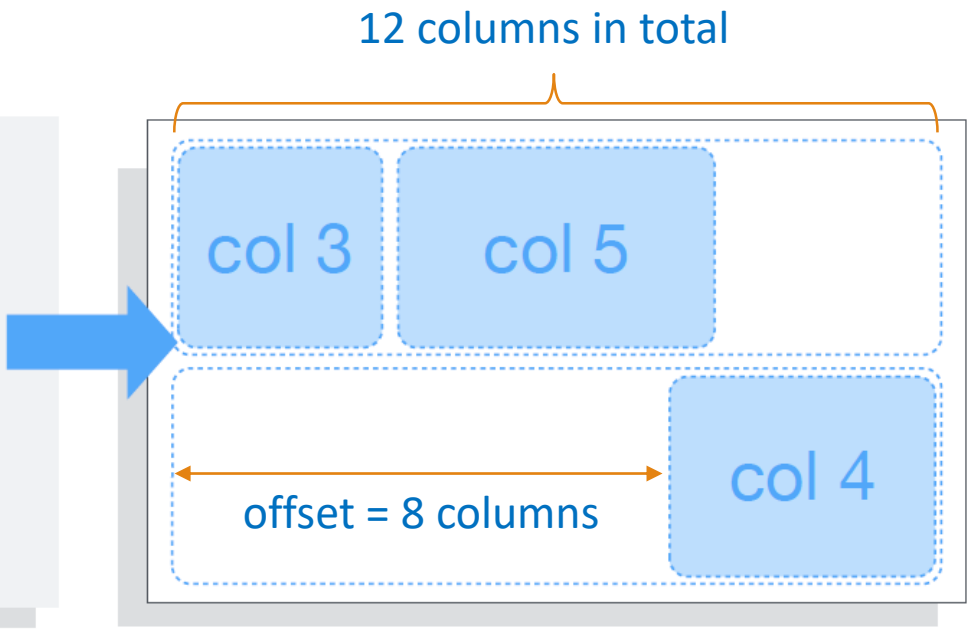
```
ui <- fluidPage(
  fluidRow(),
  fluidRow()
)
```



column()

- column() adds columns within a row. Each new column goes to the left of the previous column.
  - Specify the **width** and **offset** of each column out of 12
    - Each row is separated into 12 columns/parts. column(3) means an elements occupies 3 columns in a row.

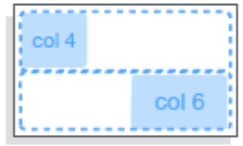
```
ui <- fluidPage(
  fluidRow(
   column(3),
   column(5)),
  fluidRow(
   column(4, offset = 8)
)
```



Grid Layout

Use **fluidRow()** to arrange elements in rows

Use **column()** to arrange elements in columns



Column takes **width** and **offset** arguments

Grid Layout

- Now, try to **write your own code** to place the slider and the histogram into the places as the following figure shows.
  - Refer to the answer in the next page.

```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
```

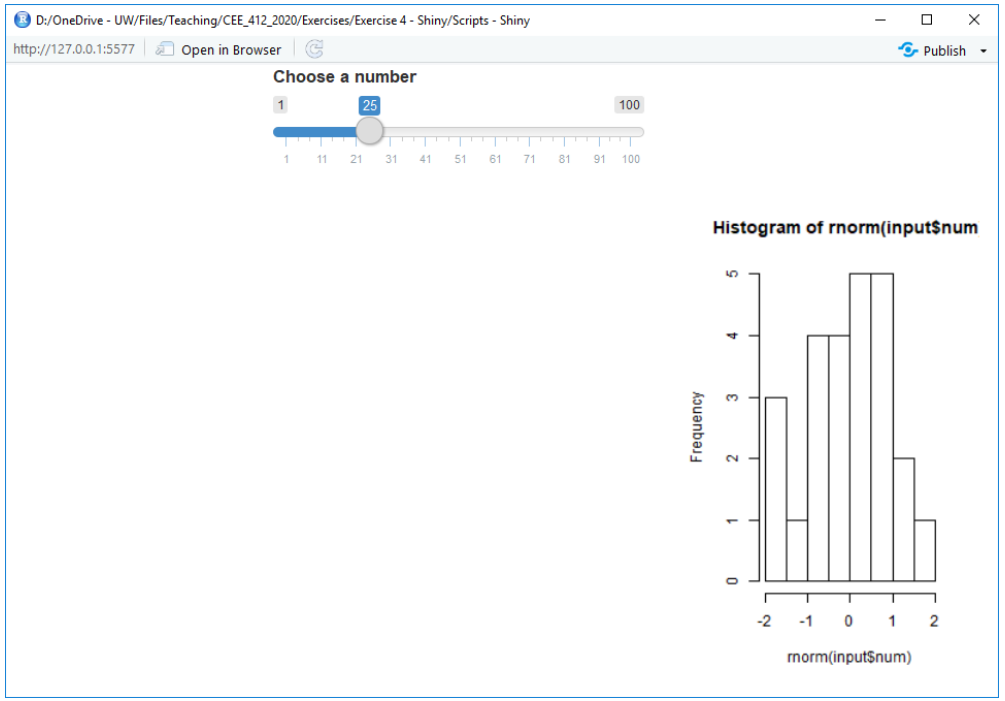
Grid Layout

Demo code file:
Exercises → Exercise 4 → Scripts → part_3_demo_1_grid_layout.R

- Create a new R file and run the following code:

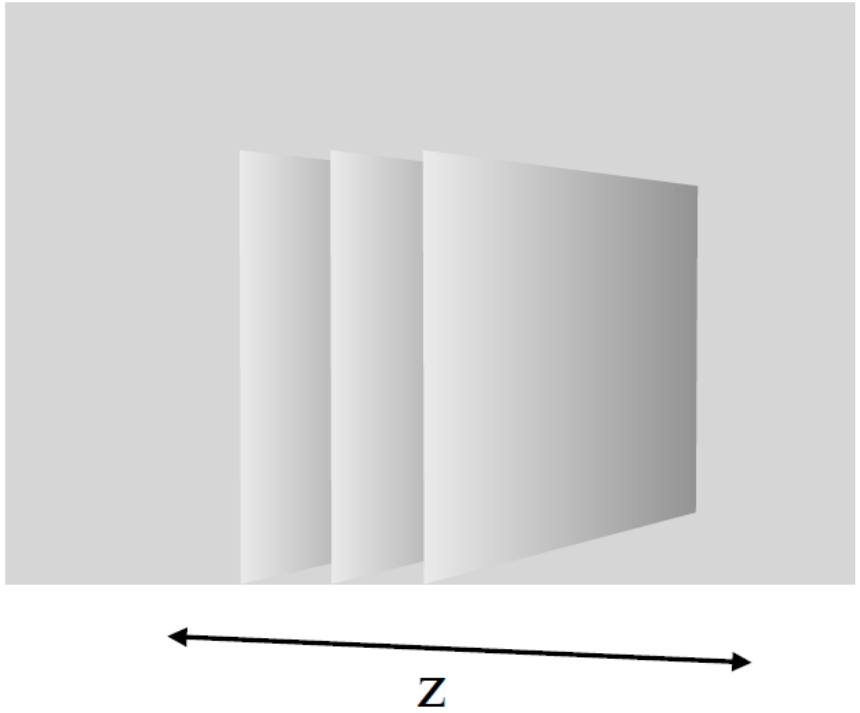
```
library(shiny)
ui <- fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(inputId = "num",
     label = "Choose a number",
     value = 25, min = 1, max = 100))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}
shinyApp(ui = ui, server = server)
```

Run



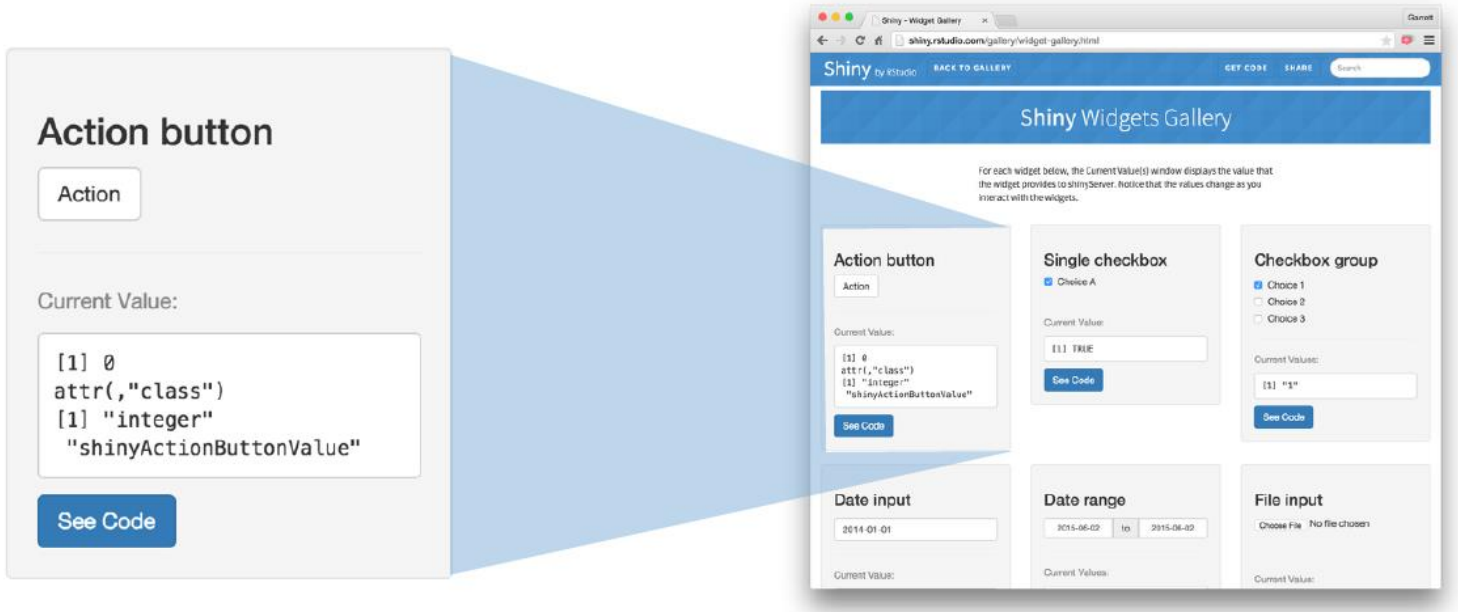
Layout

- Use layout functions to position elements within your app
  - From the perspective of y axis, use Panels.



Panels

- Panels to group multiple elements into a single unit with its own properties.



http://shiny.rstudio.com/gallery/widget-gallery.html

Panels

- Shiny provides many types of panels. We will introduce several types and please refer to the shiny document for more information.

`absolutePanel()`
Panel position set rigidly (absolutely), not fluidly

`conditionalPanel()`
A JavaScript expression determines whether panel is visible or not.

`fixedPanel()`
Panel is fixed to browser window and does not scroll with the page

`headerPanel()`
Panel for the app's title, used with pageWithSidebar()

`inputPanel()`
Panel with grey background, suitable for grouping inputs

`mainPanel()`
Panel for displaying output, used with pageWithSidebar()

`navlistPanel()`
Panel for displaying multiple stacked tabPanels(). Uses sidebar navigation

`sidebarPanel()`
Panel for displaying a sidebar of inputs, used with pageWithSidebar()

`tabPanel()`
Stackable panel. Used with navlistPanel() and tabsetPanel()

`tabsetPanel()`
Panel for displaying multiple stacked tabPanels(). Uses tab navigation

`titlePanel()`
Panel for the app's title, used with pageWithSidebar()

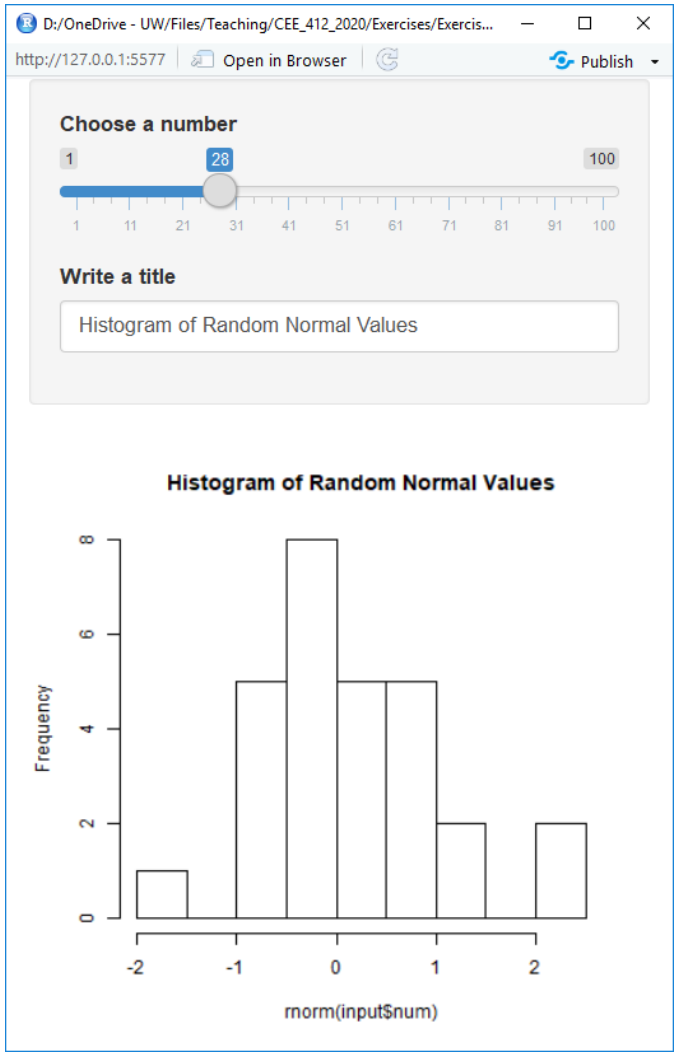
`wellPanel()`
Panel with grey background.

wellPanel()

Run the file:
Exercises → Exercise 4 → Scripts → part_3_demo_2_wellPanel.R

- wellPanel(): group elements into a grey "well"

```
ui <- fluidPage(
        wellPanel(
                sliderInput(inputId = "num",
                        label = "Choose a number",
                         value = 25, min = 1, max = 100),
                textInput(inputId = "title",
                        label = "Write a title",
                        value = "Histogram of Random Normal Values")
        ),
        plotOutput("hist")
)
```



- Try to **write your own code** to test it

tabPanel()

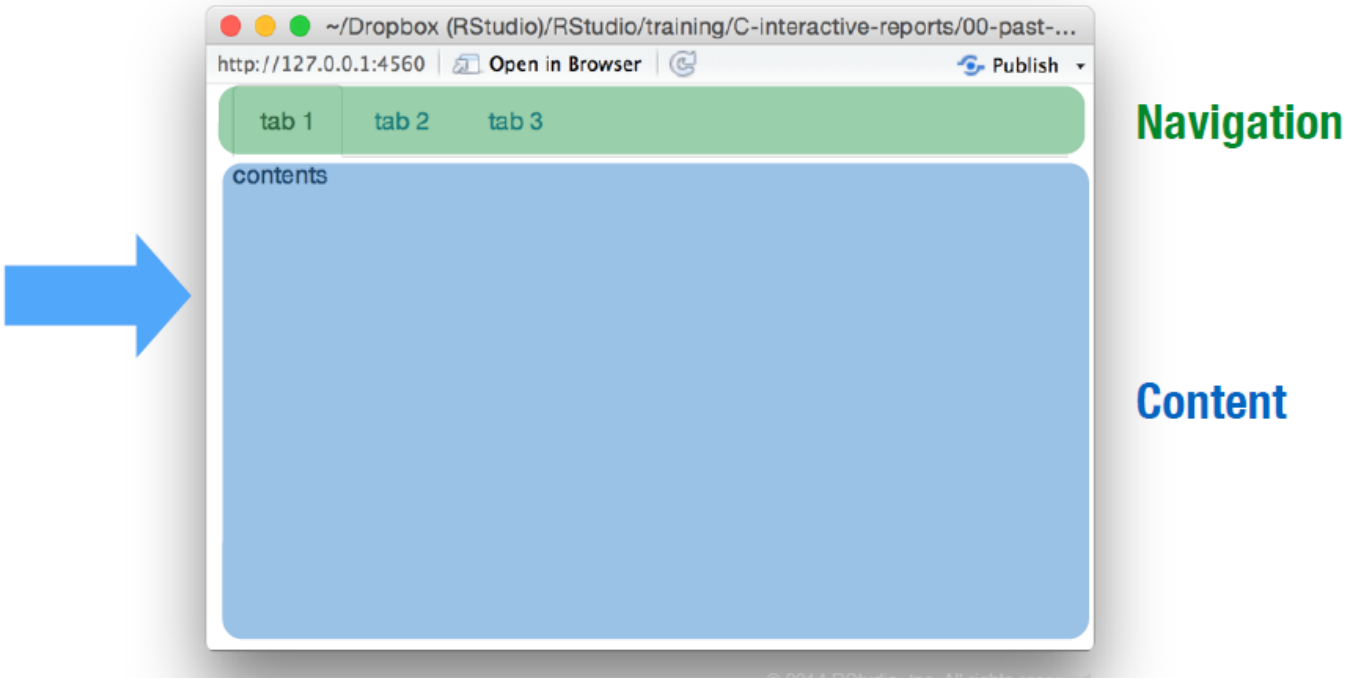
- tabPanel() creates a stackable layer of elements.
- Each tab is like a small UI of its own.
- You need to combine tabPanel()'s with one of:
  - tabsetPanel()
  - navlistPanel()
  - navbarPage()

tabsetPanel()

Run the file:
Exercises → Exercise 4 → Scripts
→part_3_demo_3_tabsetPanel.R

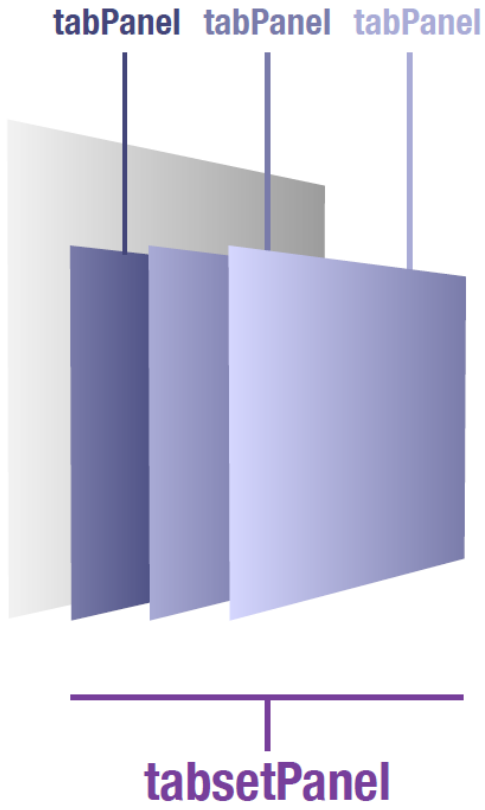
- tabsetPanel() combines tabs into a single *panel*.
- Use *tabs* to navigate between tabs.

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



tabPanel() & tabsetPanel()

- The relationship between tabPanel() and tabsetPanel() can be briefly demonstrated by the following figure:

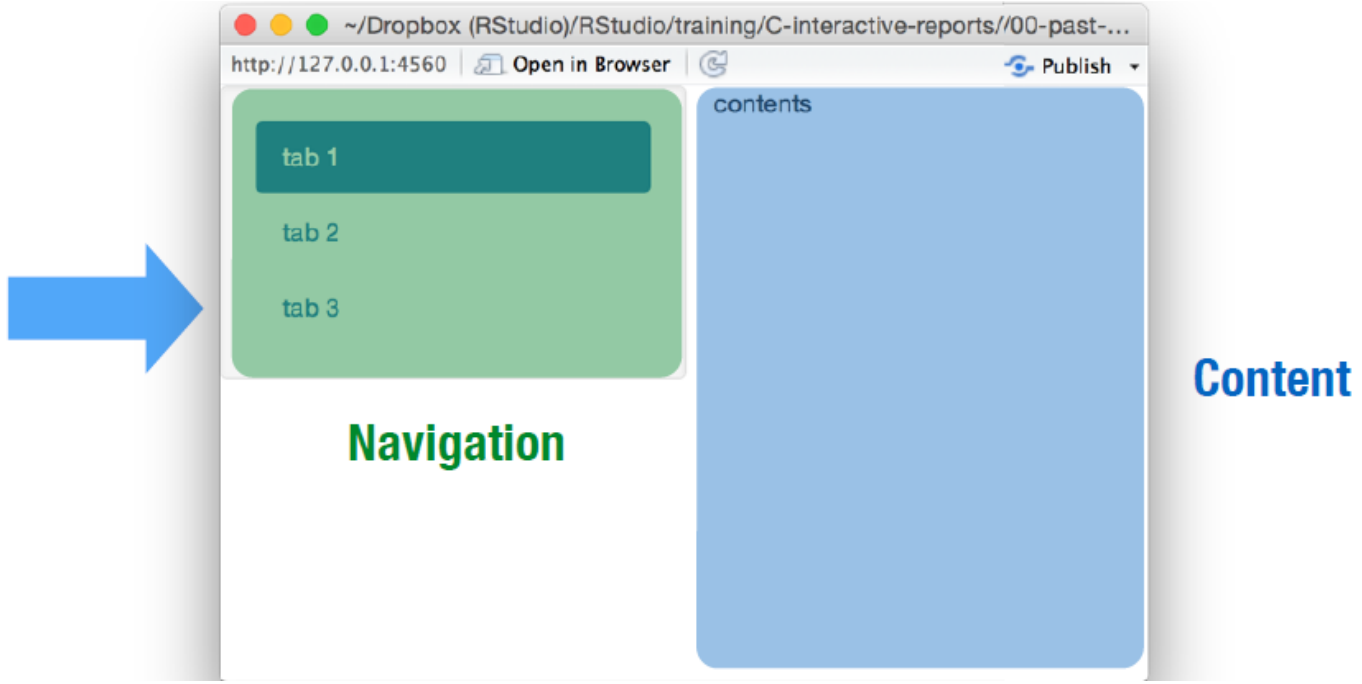


navlistPanel()

Run the file:
Exercises → Exercise 4 → Scripts
→part_3_demo_4_navlistPanel.R

- navlistPanel() combines tabs into a single *panel*.
- Use *links* to navigate between tabs.

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



Recap: Panels

Panels group elements into a single unit for aesthetic or functional reasons

Use **tabPanel()** to create a stackable panel

Use **tabsetPanel()** to arrange tab panels into a stack with tab navigation

Use **navlistPanel()** to arrange tab panels into a stack with sidebar navigation

Check the Shiny Layout Guide: http://shiny.rstudio.com/articles/layout-guide.html

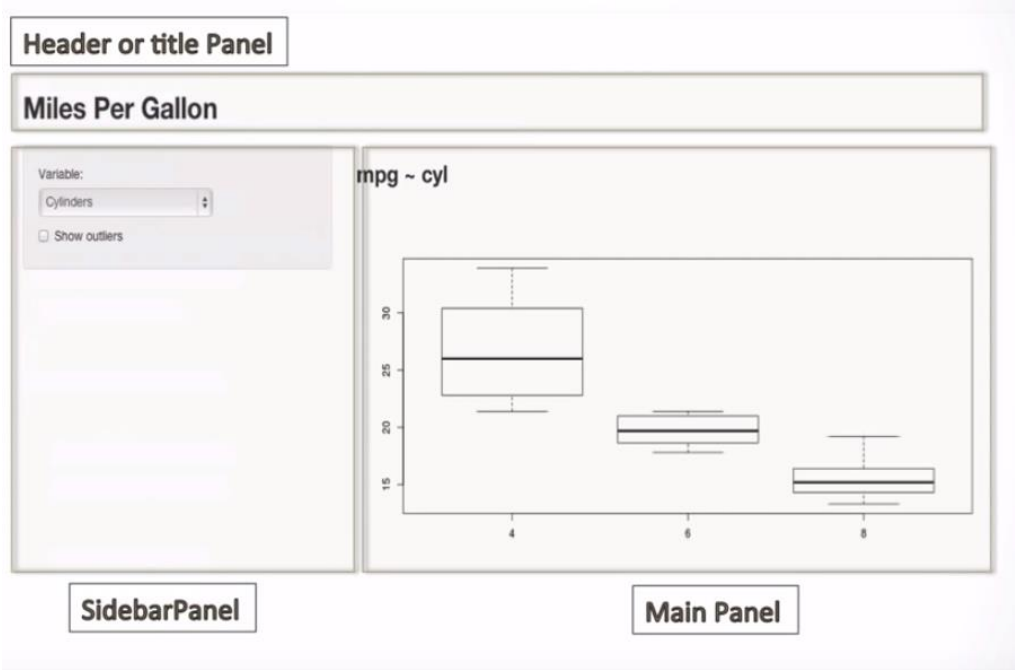
Prepackaged Layout

- sidebarLayout()
- navbarPage()
- navbarMenu()
- dashboardPage()

sidebarLayout()

- Use with sidebarPanel() and mainPanel() to divide app into two sections.
- We have used this panel multiple times. The first demo in Exercise 4 is designed based on the sidebarLayout().

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

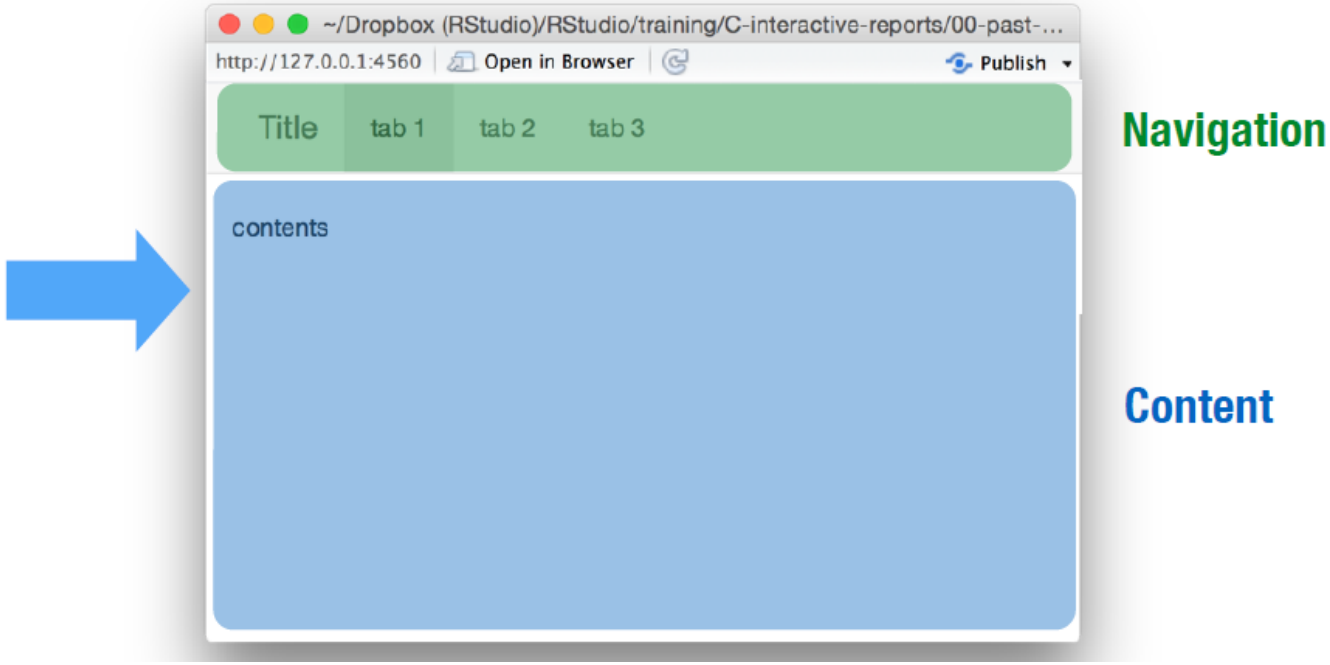


navbarPage()

Run the file:
Exercises → Exercise 4 → Scripts
→part_3_demo_5_navbarPage.R

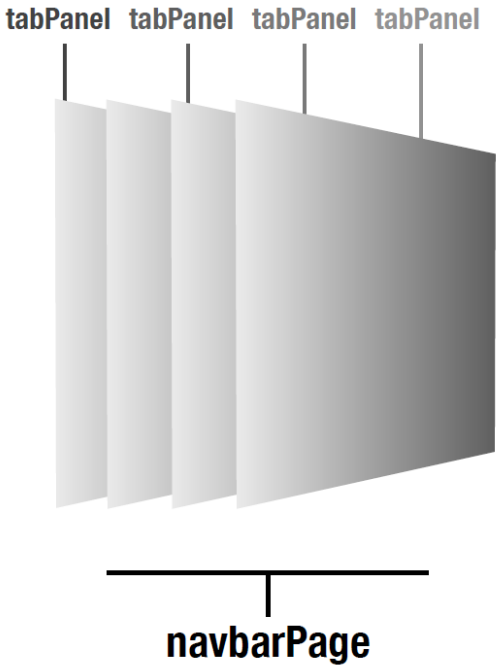
- navbarPage() combines tabs into a single *page*. *navbarPage() replaces fluidPage(). Requires title.*

```
navbarPage(title = "Title",
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   tabPanel("tab 3", "contents")
)
```



tabPanel() & navbarPage()

- The relationship between tabPanel() and navbarPage() can be briefly demonstrated by the following figure:



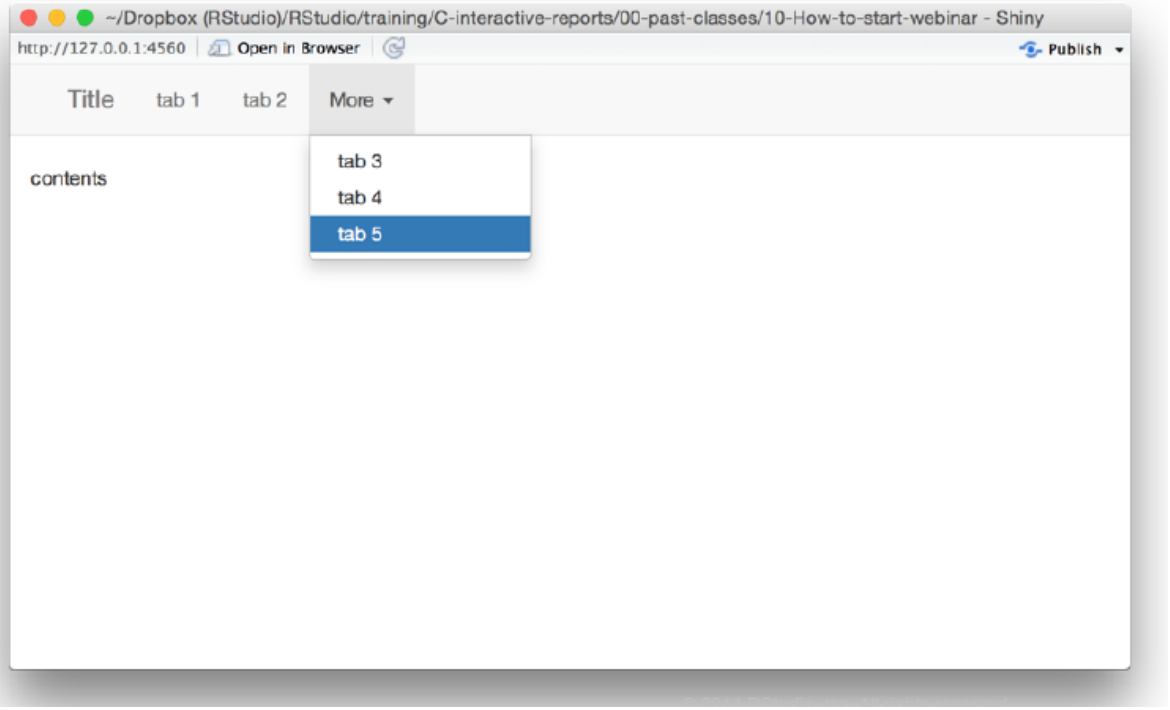
- Check the difference between this figure and the one shown in page 15.

navbarMenu()

Run the file:
Exercises → Exercise 4 → Scripts
→part_3_demo_6_navbarMenu.R

- navbarMenu() combines tab links into a dropdown menu for navbarPage()

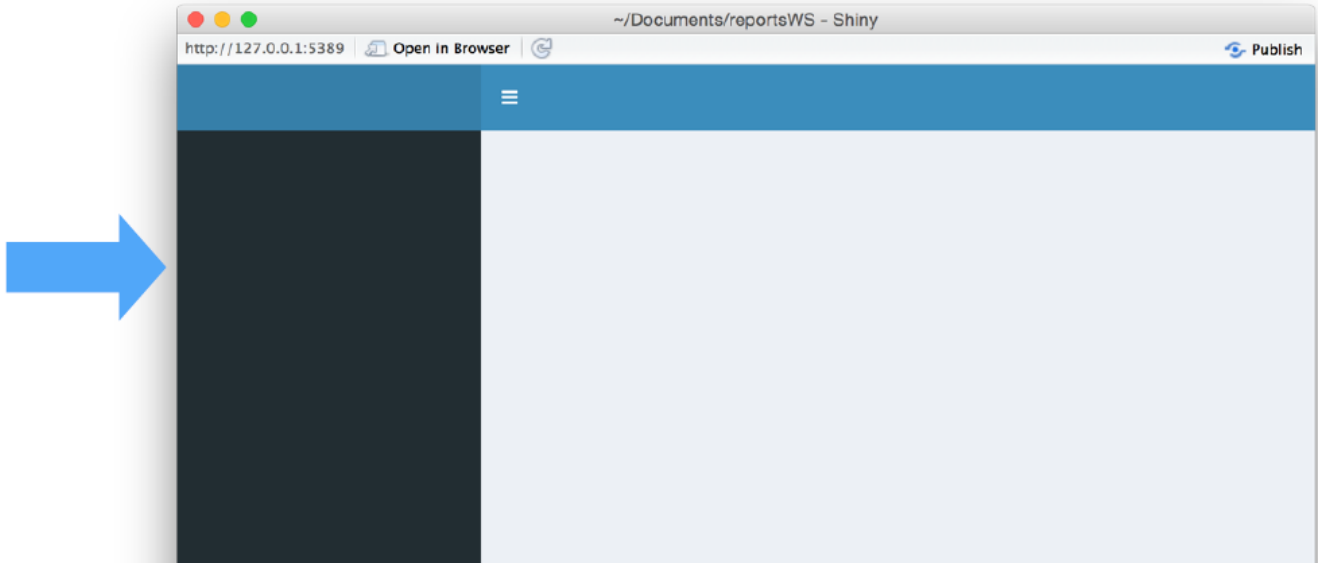
```
navbarPage(title = "Title",
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   navbarMenu(title = "More",
     tabPanel("tab 3", "contents"),
     tabPanel("tab 4", "contents"),
     tabPanel("tab 5", "contents")
  )
)
```



dashboardPage()

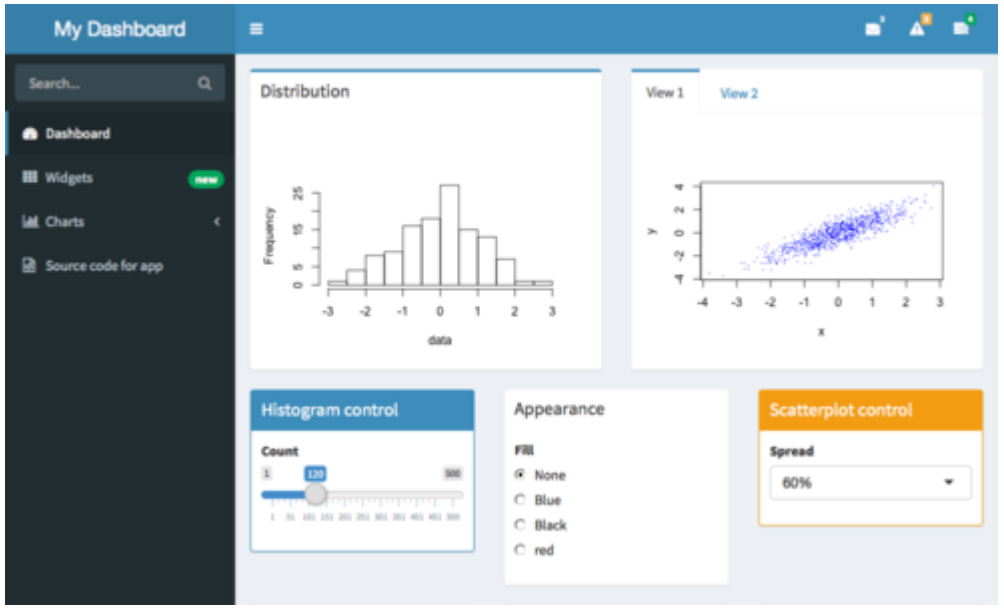
- dashboardPage() comes in the shinydashboard package

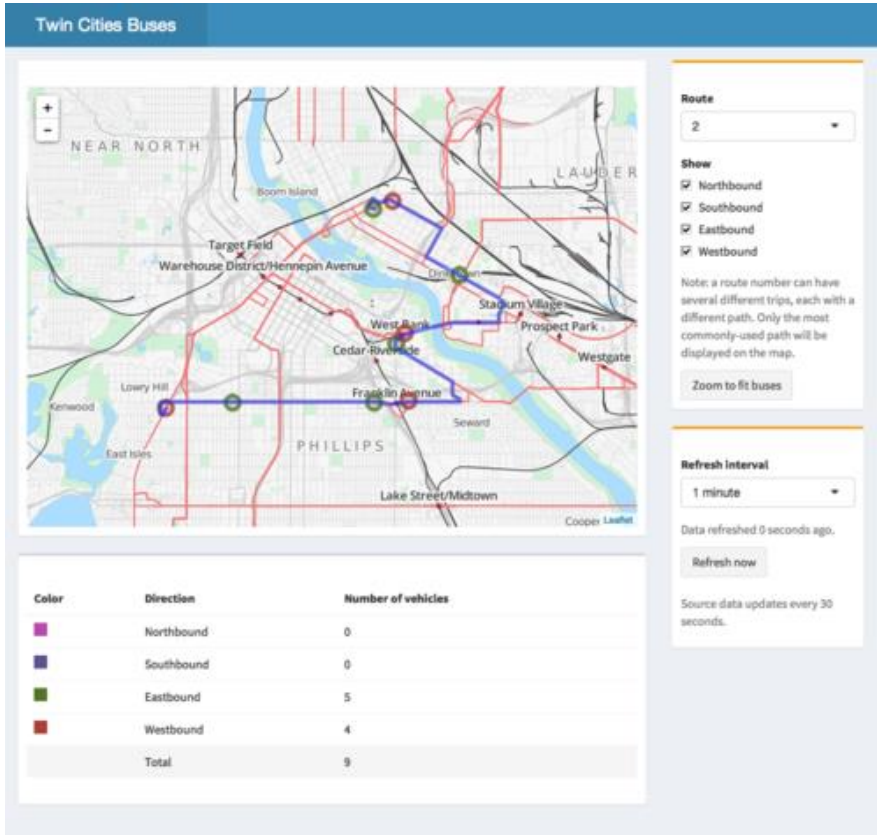
```
library(shinydashboard)
ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
```



Shiny Dashboard

- http://rstudio.github.io/shinydashboard/
  - A package of layout functions for building administrative dashboards with Shiny





Shiny Dashboard

- Installation

```
install.packages("shinydashboard")
```

  - Basic Example
    - Create a new R file and run the following code:

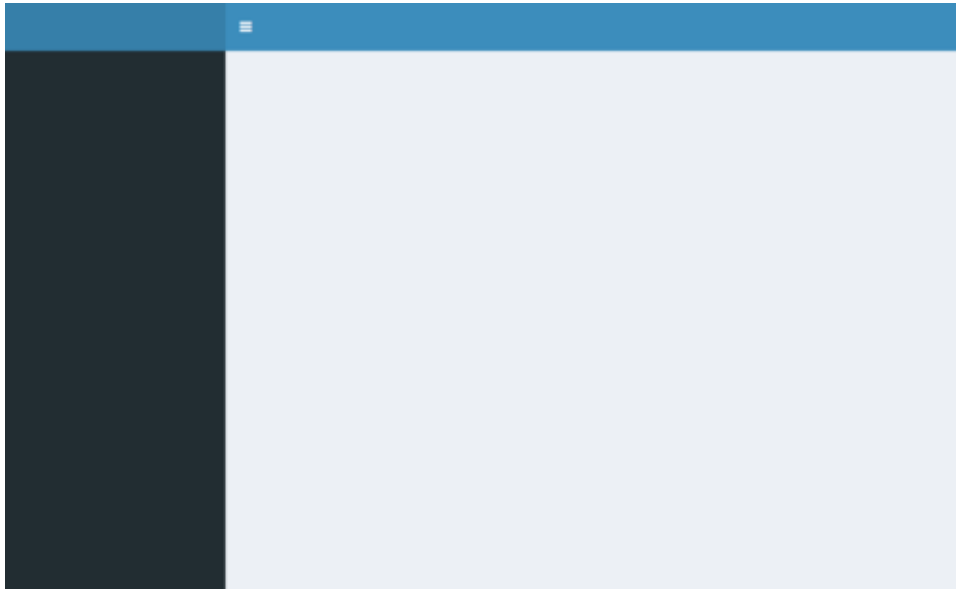
```
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
          dashboardHeader(),
          dashboardSidebar(),
          dashboardBody()
)

server <- function(input, output) { }

shinyApp(ui, server)
```

Run



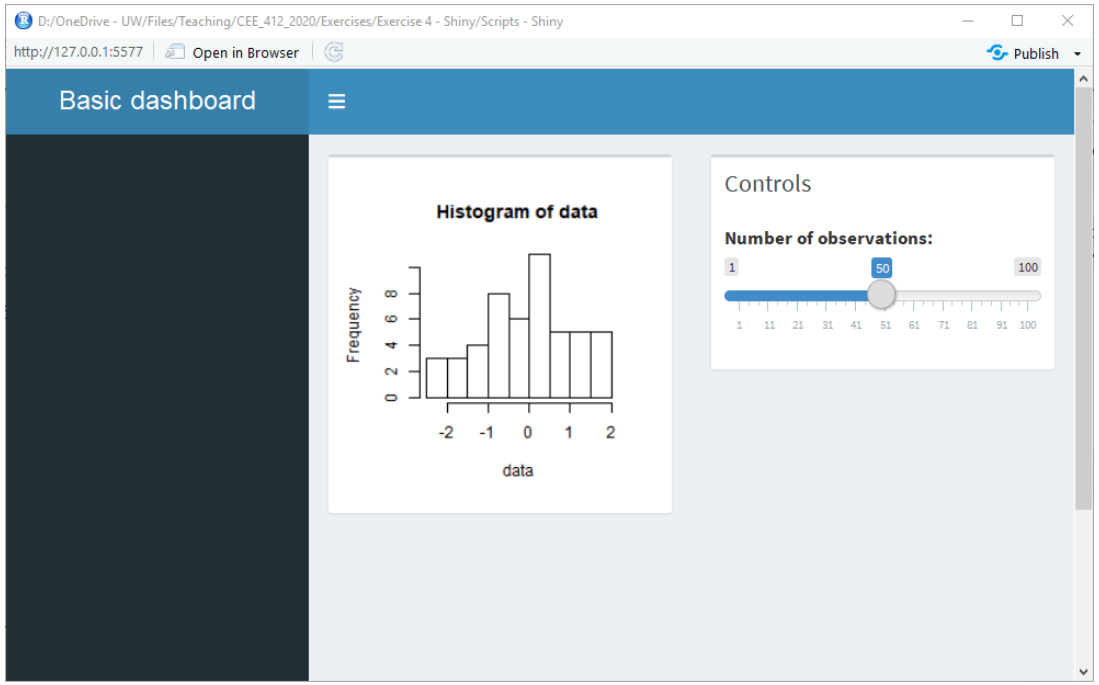
Combine Hist with Dashboard

Demo code file:
Exercises → Exercise 4 → Scripts → part_3_demo_7_dashboard.R

- Create a new R file and run the following code:

```
library(shinydashboard)
ui <- dashboardPage(
        dashboardHeader(title = "Basic dashboard"),
        dashboardSidebar(),
        dashboardBody(
                # Boxes need to be put in a row (or column)
                fluidRow(
                        box(plotOutput("plot1", height = 250)),
                        box(
                                title = "Controls",
                                sliderInput("slider", "Number of observations:", 1, 100, 50)
                         )
                )
        )
)
server <- function(input, output) {
        histdata <- rnorm(500)
        output$plot1 <- renderPlot({
                data <- histdata[seq_len(input$slider)]
                 hist(data)
         })
}
shinyApp(ui, server)
```

Run



Dashboard

- To customize your dashboard's components, such as header, sidebar, body, and layout, please refer to this link: http://rstudio.github.io/shinydashboard/get_started.html
  - The document in this link is pretty simple and intuitive.

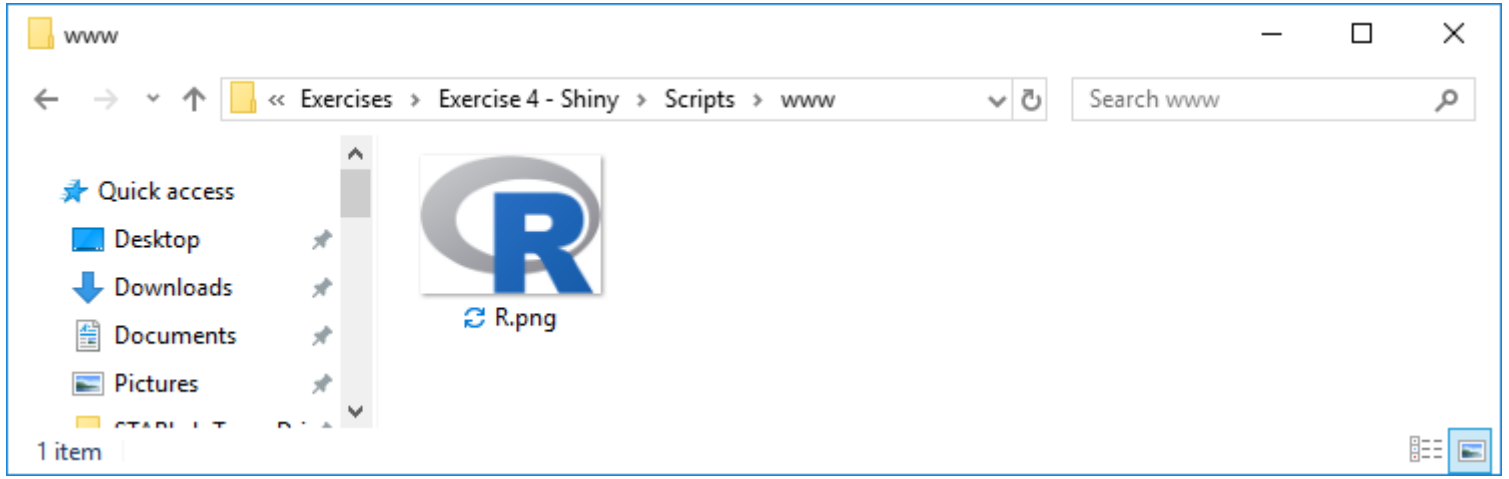
- You can find more dashboard examples in the Shiny User Show case.

Add Image

- Add an img from a file in www, using the following code.

```
ui <- fluidPage(
  tags$img(height = 100,
           width = 100,
           src = "R.png")
)
```

- To add an image from a file, first save the file in a subdirectory named **www**



Add Image

Demo code file:
Exercises → Exercise 4 → Scripts → part_3_demo_8_add_image.R

- Create a new R file and run the following code:

```
library(shiny)

ui <- fluidPage(
  tags$img(height = 100,
           width = 100,
           src = "R.png"),
  titlePanel("Hello Shiny!")
)

server <- function(input, output) {}

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

Run