Introduction to R Shiny

R Shiny Setup

Make sure that the following libraries are installed:

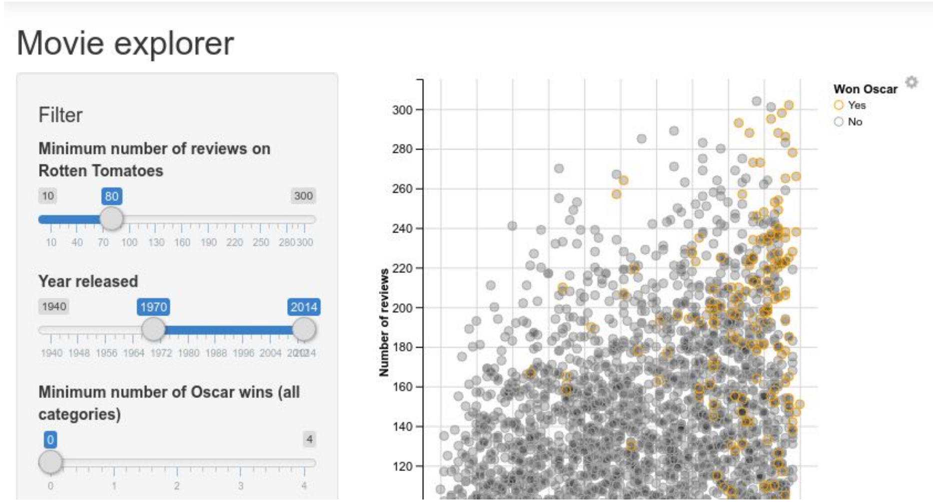
- shiny
- shinydashboard
- DT
- plotly

R Shiny

R Shiny is a packages that facilitates the creation of interactive web apps or dashboards using R.

Only requires R, although some knowledge of HTML can be useful at times.

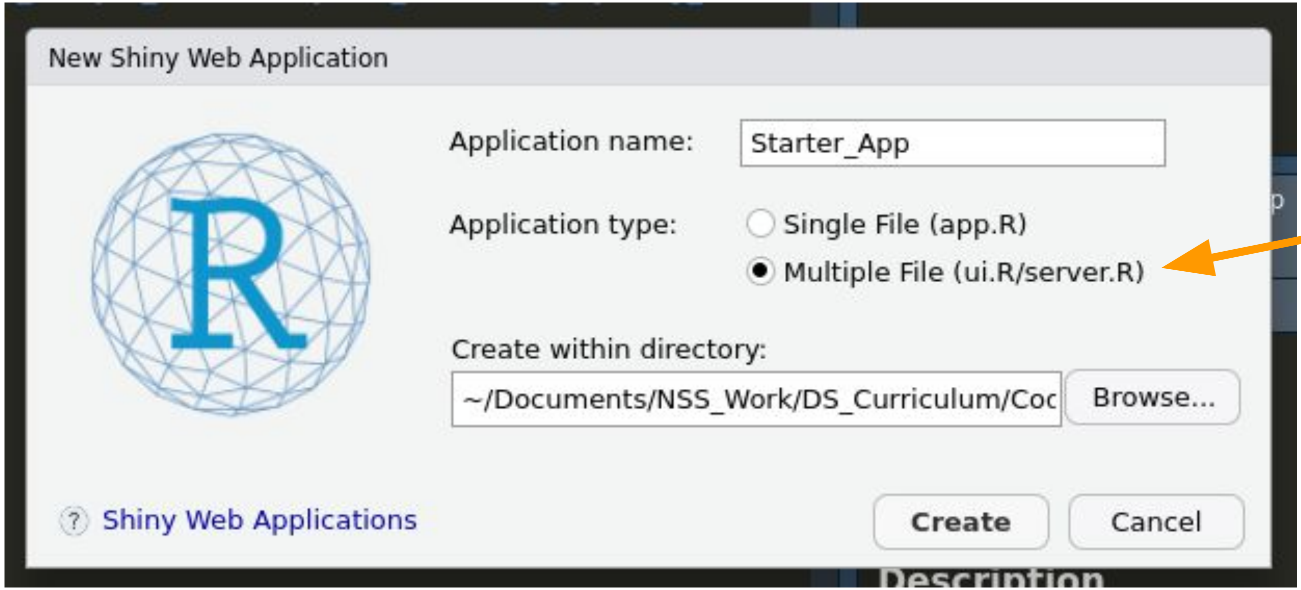
Check out the R Shiny gallery at https://shiny.rstudio.com/gallery/


Movie explorer
Filter
Minimum number of reviews on Rotten Tomatoes
10
80
300
Year released
1940
1970
2014
Minimum number of Oscar wins (all categories)
0
4
Number of reviews
Won Oscar
Yes
No

R Shiny

To create a Shiny app from RStudio, choose File > New File > Shiny Web App

Browse to or create a new directory for your app



New Shiny Web Application

Application name: Starter_App

Application type: ○ Single File (app.R) [● Multiple File (ui.R/server.R)]

Create within directory: ~/Documents/NSS_Work/DS_Curriculum/Coc

Choose Multiple File for the Application type.

Components of a Shiny App

Every Shiny App consists of two primary components:

- **ui: “**Front-End”
    - Controls layout and appearance
    - What the user sees
- **server:** “Back-End”
    - Contains instructions for building your app
    - What gets updated/run as the user interacts with the interface

It is often useful to include a third component:

- **global:** Objects visible to both server and UI
    - Useful for loading in datasets and libraries
    - Anything created in global is visible both to ui and to server

ui

Every Shiny app needs a layout - see
https://shiny.rstudio.com/articles/layout-guide.html
Examples:

- *fluidPage*
  - Can add other elements such as a *titlePanel*, *sidbarLayout*, *fluidRow*, or *column*
- *navbarPage* for multiple tabs
- *shinydashboard*

shinydashboard Layout

The shinydashboard layout is recommended for creating your Shiny apps and dashboards.

The shinydashboard layout makes it easy to have tabbed content within your dashboard. See this example: https://rstudio.github.io/shinydashboard/get_started.html

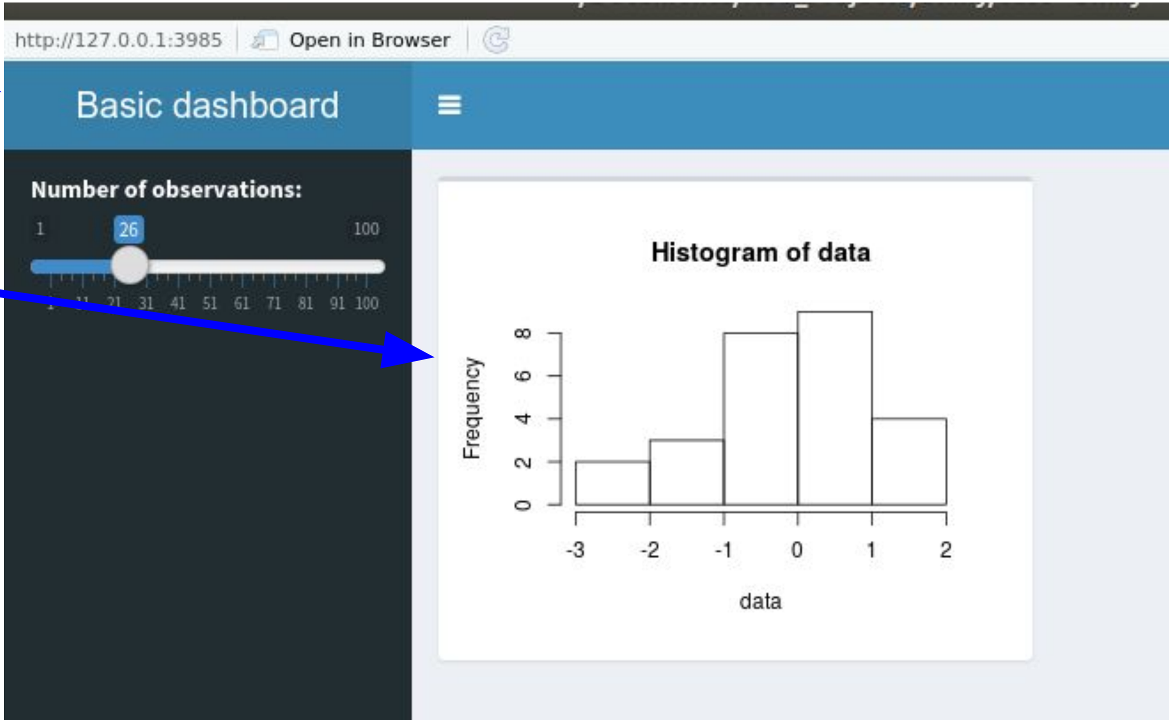
More information about shinydashboard can be found here: https://rstudio.github.io/shinydashboard/ or in this DataCamp course: https://www.datacamp.com/courses/building-dashboards-with-shinydashboard

shinydashboard Layout

A shinydashboard consists of 3 parts:

- dashboardHeader()
- dashboardSidebar()
- dashboardBody()

dashboardPage()



ui Organization

There are a number of elements that can be used to keep the contents of your dashboardBody organized.

fluidRow() - Elements within a fluidRow will appear on the same line

column() - Determines how much horizontal space an element will take up (between 1 and 12)

box() - A basic container to hold content.

ui - Widgets

Widgets can be used to allow the user to interact with your app.

There are a variety of types of widgets:

- *actionButton*
- *selectInput*
- *sliderInput*
- *textInput*

See the Shiny Widgets Gallery for more information:
https://shiny.rstudio.com/gallery/widget-gallery.html

ui - Widgets

All widgets require a **name** (used to access its value) and a **label** (which the user sees).

```
sliderInput("bins","Number of bins:", min = 1, max = 50, value = 30)
```

name

label

In the **server** portion of our app, we will be able to access the value of this widget by using *input$bins.*

server - Reactive Elements

Reactive elements automatically respond when the user modifies a widget value.

Can be used to update a plot, reslice a dataframe, call a function, or other tasks.

Shiny can determine what the output from a reactive expression depends on and only rerun it if the input changes.

If you modularize your code into a series of reactive function calls, you can make your app more efficient.

See https://shiny.rstudio.com/tutorial/written-tutorial/lesson6/

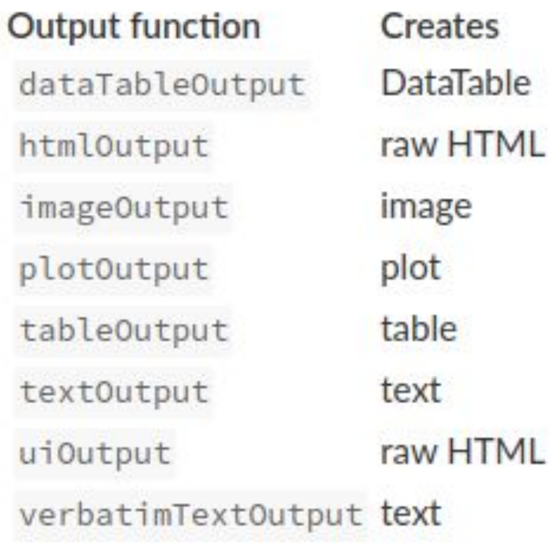
server - Reactive Elements

Two steps to add a reactive element:

1. Add an R object to your user interface
2. Tell Shiny how to build the object in the server function. The object will be reactive if the code that builds it uses a widget value.

ui - Reactive Elements

There are a number of different types of objects you can add to your interface.
The function you used depends on the type of object you are adding.
These functions are called in the **ui** side.



Output function	Creates
`dataTableOutput`	DataTable
`htmlOutput`	raw HTML
`imageOutput`	image
`plotOutput`	plot
`tableOutput`	table
`textOutput`	text
`uiOutput`	raw HTML
`verbatimTextOutput`	text

ui - Reactive Elements

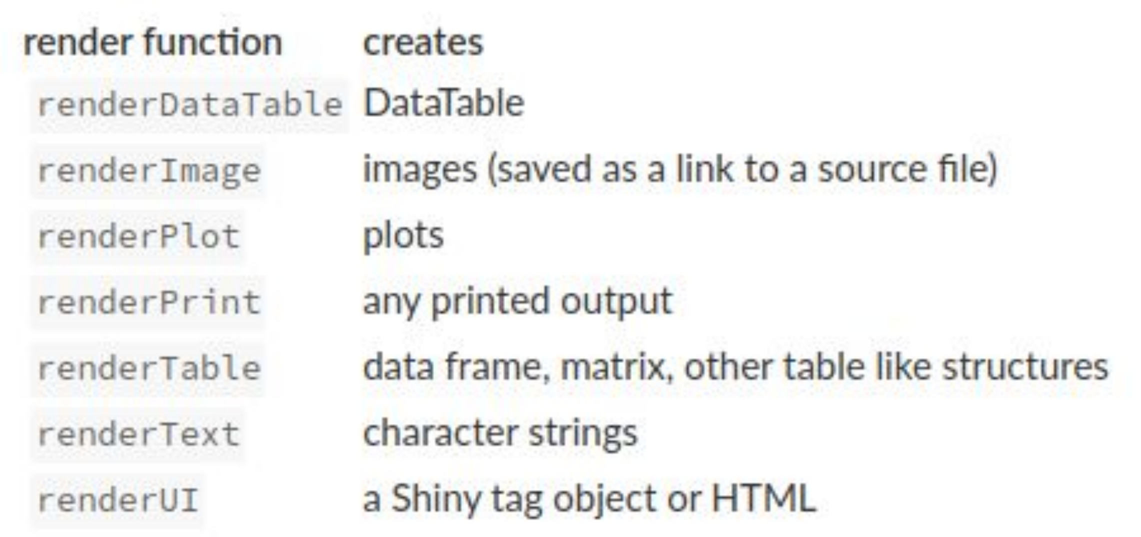
```
plotOutput("distPlot")
```

display a plot

specifically, display output$distPlot (generated on the server side)

server - Reactive Elements

For every output function you call, there should be a corresponding **render*** call on the **server** side:



render function	creates
`renderDataTable`	DataTable
`renderImage`	images (saved as a link to a source file)
`renderPlot`	plots
`renderPrint`	any printed output
`renderTable`	data frame, matrix, other table like structures
`renderText`	character strings
`renderUI`	a Shiny tag object or HTML

ui - Reactive Elements

how to reference this plot in ui

This must contain an expression that generates a plot, contained inside the braces {}

```
output$distPlot <- renderPlot({
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
    })
```

refers to the value from the sliderInput

Resources for Learning Shiny

RStudio’s official tutorial (2.5 hours): https://shiny.rstudio.com/tutorial/

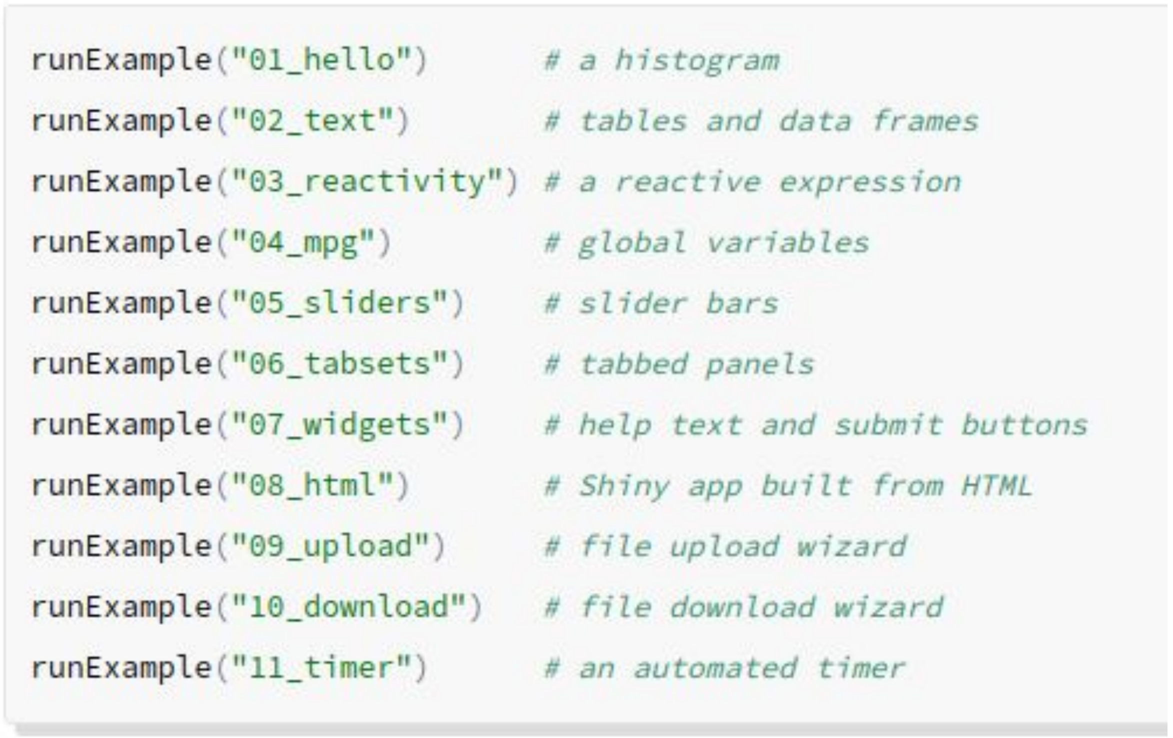
RStudio’s written tutorial: https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/

DataCamp Courses:
https://www.datacamp.com/courses/building-dashboards-with-shinydashboard
https://www.datacamp.com/courses/building-web-applications-in-r-with-shiny-case-studies

The Gallery (A lot of apps will let you see the code):
https://shiny.rstudio.com/gallery/

Resources for Learning Shiny

Shiny comes with some built-in examples as well:



```
runExample("01_hello")       # a histogram
runExample("02_text")        # tables and data frames
runExample("03_reactivity") # a reactive expression
runExample("04_mpg")         # global variables
runExample("05_sliders")     # slider bars
runExample("06_tabsets")     # tabbed panels
runExample("07_widgets")     # help text and submit buttons
runExample("08_html")        # Shiny app built from HTML
runExample("09_upload")      # file upload wizard
runExample("10_download")    # file download wizard
runExample("11_timer")       # an automated timer
```

Minimum Requirements for Midcourse Projects

- Clean and explore your chosen dataset(s) and perform analysis working towards answering the question posed
- Create an **interactive** Shiny app to present your findings. This app should facilitate further exploration and discovery by the user.
- Put together an 8-12 minute accompanying presentation with slides and a demo of the app.
- Work should be in a GitHub Repository with a detailed README.
- Deadlines - Internal presentations: **January 16**

  Presentations: **January 18**

Before Leaving for the Break after Saturday’s Class:

1. Get final approval for your Midcourse project
2. Have a check-in with me
3. Create a Shiny app with at minimum one interactive visualization, using data from Data Question 5

On the day we get back, January 2, the building will not be open until 5:00!