# HCAP FYSP Shiny PDF

Brian Kim

17/07/2020

*Portions of this curriculum are adopted from Preceptor David Kane's GOV50 Class at Harvard University.*

## Shiny Instructions

## We are going to make a Shiny app. To do that, we need you to do several things.

- Sign up for a Shiny account at https://www.shinyapps.io/.
- Install the Shiny package with: `install.packages("shiny")` (you can do this on an Rmd file)

## Let's connect Shiny and our Rstudio

For that, we are going to check the following website: https://shiny.rstudio.com/articles/shinyapps.html

**It is imperative that you follow the steps of that page until you get to "A Demo" section.

Once that happens, let's make a Shiny app.

- Create a Shiny app from RStudio with: File > New File > Shiny Web App...

In the popup, choose Single File (app.R)

You should now have a functioning example Shiny App - click Run App or use the keyboard shortcut Cmd/Ctrl + Shift + Enter to see this app in action! You can view it in a new window, or in an external web browser (see the dropdown arrow next to Run App for the options). Notice that the example includes a slider that allows the viewer to change the number of the bins that the histogram has.

## You have a Shiny app running. What now?

Now that you have your functioning Shiny App up and running, let's take a closer look at our files and directories. Notice that the app.R file has been created within a directory. It is important to remember that only the files within this directory will be accessible to the app when it is online. Taking a look at the app.R file, we can see that there are four necessary elements in this file that create the working app:

First, it calls library(shiny) Then, it defines a user interface with ui <- fluidPage(...) It also defines server logic, which takes in an input from the UI, and produces an output based on that input, as defined by server <- function(input, output) {...} Finally, it calls shinyApp(ui, server) to run the app.

## First Setup

Let's copy and paste this to our app.R file.

```
ui <- navbarPage(
  "Some Random Title",
  tabPanel("Models",
           fluidPage(
              titlePanel("Model Title"),
              sidebarLayout(
                  sidebarPanel(
                      selectInput(
                          "plot_type",
                          "Plot Type",
                          c("Option A" = "a", "Option B" = "b")
                      )),
                  mainPanel(plotOutput("line_plot")))
            )),
  tabPanel("About",
           titlePanel("About"),
           h3("Project Background and Motivations"),
           p("Hello, this is where I talk about my project."),
           h3("About Me"),
           p("My name is ______ and I study ______.
              You can reach me at ______@college.harvard.edu.")))
```

This sets our page to be two tabs, one being about the graph and one introducing ourselves.This is the user interface.

Let's set up the server too, which allows us to put in these different content that we want in our pages.

```
server <- function(input, output) {
    output$line_plot <- renderPlot({
        # Generate type based on input$plot_type from ui

        ifelse(
            input$plot_type == "a",

            # If input$plot_type is "a", plot histogram of "waiting" column
            # from the faithful dataframe

            x   <- faithful[, 2],

            # If input$plot_type is "b", plot histogram of "eruptions" column
            # from the faithful dataframe

            x   <- faithful[, 1]
        )

        # Draw the histogram with the specified number of bins

        hist(x, col = 'darkgray', border = 'white')
    })
}
```

At the very end, you should have this following code or else it won't run.

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

Great! We have our basic page set up. What we could do now is press "Run App" to see how it shows.

## Now we're going to input our graphs from the other exercise.

But how can we do this?

The easiest way is to make a completely new R script, input all the same code (except for the installations). It's also helpful to have the data that we used in the shiny folder itself. This isn't necessary, but if we were to ignore, we would have to change directories and figure out the local source - which warrants a new conversation altogether.

That's what we're going to do.

Once everything is copied and pasted, we are going to go back to the app.R file and "source" the saved, new file. I titled this "modelforproject.R".

You have to run this code in the very beginning of app.R to make sure we have the graphs inputted correctly.

```
source("modelforproject.R")
```

Let's customize by changing the initial two graphs to the ones that we made.

Looking at server, that's where I know I want to put the graphs in because that's how it'll be loaded.

Good thing though, we did all our calculations in advance on modelforproject.R and we have our names; sampleplot and sampleplot2.

If it returns "a", which at the moment titles as "Option A" in the ui, we can then put `return(sampleplot)` to return that graph.

If it returns "b", titled as "Option B", we can then put `return(sampleplot2)` afterwards.

Other things we can customize right now (there's so much you can do with Shiny, but these are like the REALLY Basics and there's a lot more to this) –

why not change theme?

We can use the code `install.packages("shinythemes") library(shinythemes)`

and then search up which theme we like on Google.

I personally did cerulean, which you can do in the very beginning:

```
ui <- navbarPage(
  theme = shinytheme("cerulean"),

  # insert title of page

  title = "HCAP FYSP Project"
```

## Let's publish our webpage.

Well, let's check Run and see if we like what we're seeing.

Do we like it? Let's publish it.

Next to the Run App button, there should be a blue button and an arrow. Click on the arrow, and click publish.

> Important. The file that you're publishing HAS TO BE app.R, or you will encounter errors.

Click publish, click publish, etc.

Might take some time.

And once you publish it, but you realize that you made some errors and you need to republish, no problem! Just fix your thing and publish again. The page will update.

Once the page is published, you will see that the graph is very awkwardly sized. This can be fixed with additional instructions.

# CHALLENGE:

1) Figuring out how to resize the graphs so it looks nice (.5 points) and
2) Add a hyperlink to your email address on the about page and hyperlinks to other websites (.5 points).

If you are able to do both, you can proceed to do the following challenges: *for these challenges, you can choose one to do – or do both!*

1) Add a profile picture of yourself to the "About" page. (1pt)
2) Add another tab page and put another graph with explanation. (1pt)

If you are able to recreate any of the shiny pages **here** without directly looking at the app.R code (or the code that runs the Shiny UI and system interface), you may even be able to get an early promotion to Board as Foreign Liaison Chair and replace Brian.

## Inspirations:

*For inspirations on Shiny projects, feel free to look at this website:* https://www.davidkane.info/files/final_projects.html *and this resource (which, thanks to Preceptor David Kane, I was able to do a lot):* https://davidkane9.github.io/PPBDS/shiny.html