GETTING STARTED WITH SHINY - BASICS

Required packages:

- shiny
- shinydashboard
- tidyverse
- palmerpenguins

Some resources:

- This is a great introduction to making a Shiny app: https://deanattali.com/blog/building-shiny-apps-tutorial/
- And here are a bunch of tutorials and examples: https://shiny.rstudio.com/tutorial/
- Here are some cool examples of what you can do with Shiny: https://shiny.rstudio.com/gallery/

EXAMPLE 1: INTRO TO SHINY (UI/SERVER, MEET REACTIVITY)

1. **Get started:**
    1. Create a new project in R
    2. Open a new R script
    3. Add the following at the top of the script:

```
# Attach necessary packages
library(shiny)
library(tidyverse)
library(palmerpenguins)

# Create the user interface:
ui <- fluidPage()

# Create the server function:
server <- function(input, output) {}

# Combine them into an app:
shinyApp(ui = ui, server = server)
```

    4. **Save the script as app.R**
    5. Notice that a 'Run app' option shows up - press it, and see that blank page shows up in the server. Whomp.

2. **Start building the UI**
    a. Here, we'll add a title panel, and a sidebar/main panel layout

```
library(shiny)
library(tidyverse)
library(palmerpenguins)

# Create the user interface:
ui <- fluidPage(
  titlePanel("I am adding a title!"),
  sidebarLayout(
          sidebarPanel("put my widgets here"),
          mainPanel("put my graph here")
          )
)

# Create the server:
server <- function(input, output) {}

# Combine them into an app:
shinyApp(ui = ui, server = server)
```

    b. Save, run app again (you may need to stop "listening" - press stop in Console)

3. Add your first widget in the side panel!

See the Shiny widget gallery: https://shiny.rstudio.com/gallery/widget-gallery.html

1. Within the sidebar panel, add a radioButtons() widget. You need to tell it: the ID of the widget (how will you refer to these widget selections later on in the server?); a label to place at the top of the widget (like a widget title), and the choices of the check boxes. These should **match the inputs of the variable you're going to use it to explore/update based on.** For example, we are going to use the "species" widget to decide which penguin species to show in a graph, and in the 'species' column those are stored as "Adelie", "Gentoo" and "Chinstrap". Those choices should match exactly. *Note: if you want the choice to look different in the widget, but point to entries in a column, you can set "visible_text" = "column_entry" within the 'choices =' argument, as in the "Cool Chinstrap Penguins!" = "Chinstrap" example below. In this case, "Cool Chinstrap

Penguins!" will show up as a choice in the widget, but it will functionally point to the string "Chinstrap"...which will become relevant later on.

```
library(tidyverse)
library(shiny)
library(palmerpenguins)

ui <- fluidPage(
  titlePanel("I am adding a title!"),
  sidebarLayout(
            sidebarPanel("put my widgets here",
                         radioButtons(inputId = "penguin_species",
label = "Choose penguin species", choices =
c("Adelie","Gentoo","Cool Chinstrap Penguins!" = "Chinstrap")
                         )
                         ),
            mainPanel("put my graph here")
  )
)

server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

4. Build a reactive graph in the server, based on selections made in the 'species' widget!

So that widget selection is going to become an input that is used to specify what shows up in our graph. Since we gave that widget the ID of 'species,' then we're going to use input$species when we want to refer to the widget selection. We will also give our OUTPUTS (created in the server) an ID "name" so that we can call them back to the UI.

a. In the server(), create a reactive subset of the 'penguins' df that only includes the species selected by the widget 'species'. Then, create a reactive graph of the flipper length (flipper_length_mm) vs. body mass (body_mass_g) variables in ggplot, and have it show up in our UI (main panel)

```
library(tidyverse)
library(shiny)
library(palmerpenguins)

ui <- fluidPage(
  titlePanel("I am adding a title!"),
```

```
  sidebarLayout(
            sidebarPanel("put my widgets here",
                         radioButtons(inputId =
"penguin_species", label = "Choose penguin species", choices
= c("Adelie","Gentoo","Cool Chinstrap Penguins!" =
"Chinstrap")
                         )
                         ),
            mainPanel("put my graph here")
  )
)


server <- function(input, output) {

            penguin_select <- reactive({
      penguins %>%
                         filter(species ==
input$penguin_species)
                         })

  output$penguin_plot <- renderPlot({

            ggplot(data = penguin_select(), aes(x =
flipper_length_mm, y = body_mass_g)) +
            geom_point()

  })

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

c. Try running the app again...notice that nothing shows up! That's because we haven't added it to our UI yet! We need to call our graph (output$penguin_plot) back in the UI main panel:

```
library(tidyverse)
library(shiny)
library(palmerpenguins)
```

```
ui <- fluidPage(
  titlePanel("I am adding a title!"),
  sidebarLayout(
         sidebarPanel("put my widgets here",
                              radioButtons(inputId =
"penguin_species", label = "Choose penguin species", choices =
c("Adelie","Gentoo","Cool Chinstrap Penguins!" = "Chinstrap")
                     )
                              ),
         mainPanel("put my graph here",
           plotOutput(outputId = "penguin_plot")
           )
  )
)

server <- function(input, output) {

         penguin_select <- reactive({
            penguins %>%
                              filter(species ==
input$penguin_species)
                              })

  output$penguin_plot <- renderPlot({



         ggplot(data = penguin_select(), aes(x =
flipper_length_mm, y = body_mass_g)) +
         geom_point()

  })

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

And now run it! Cool!

5. Let's add another widget to select the color we want to use in our graph.

a. Create a new widget that is a dropdown selection (selectInput) for different colors that a user can choose from for the graph points:

```
library(tidyverse)
library(shiny)
library(palmerpenguins)

# Creating the user interface
ui <- fluidPage(
  titlePanel("I am adding a title!"), # This is the title!
  sidebarLayout( # Adding a sidebar & main panel
            sidebarPanel("put my widgets here",
                         radioButtons(inputId =
"penguin_species", label = "Choose penguin species", choices
= c("Adelie","Gentoo","Cool Chinstrap Penguins!" =
"Chinstrap"), # This is my first widget for penguins species
                         ),
                         selectInput(inputId = "pt_color",
label = "Select point color", choices = c("Awesome red!" =
"red", "Pretty purple" = "purple", "ORAAANGE" = "orange"))
                         ),
            mainPanel("put my graph here", # Adding things to
the main panel
              plotOutput(outputId = "penguin_plot")
              )
  )
)

# Building the server:
server <- function(input, output) {

penguin_select <- reactive({
penguins %>%
            filter(species == input$penguin_species)
            })

  # Create a reactive plot, which depends on 'species' widget
selection:
```

```
output$penguin_plot <- renderPlot({

            ggplot(data = penguin_select(), aes(x = flipper_length_mm, y
            = body_mass_g)) +
                        geom_point()

              })

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

Run the app - and notice that a second widget shows up, but the graph doesn't change. Why not? Have we changed anything in the server that would let it know the color should be reactive to the widget input? Not yet! Let's do that next...

6. Make the graph point color reactive to the widget selection

```
library(tidyverse)
library(shiny)
library(palmerpenguins)

# Creating the user interface
ui <- fluidPage(
  titlePanel("I am adding a title!"), # This is the title!
  sidebarLayout( # Adding a sidebar & main panel
            sidebarPanel("put my widgets here",
                        radioButtons(inputId = "penguin_species", label =
"Choose penguin species", choices = c("Adelie","Gentoo","Cool
Chinstrap Penguins!" = "Chinstrap"), # This is my first widget for
penguins species
                        ),
                        selectInput(inputId = "pt_color", label = "Select
point color", choices = c("Awesome red!" = "red", "Pretty purple" =
"purple", "ORAAANGE" = "orange"))
                        ),
            mainPanel("put my graph here", # Adding things to the main
panel
              plotOutput(outputId = "penguin_plot")
              )
```

```
  )
)

# Building the server:
server <- function(input, output) {

          penguin_select <- reactive({
          penguins %>%
                    filter(species == input$penguin_species)
                    })

  # Create a reactive plot, which depends on 'species' widget
selection:

output$penguin_plot <- renderPlot({

          ggplot(data = penguin_select(), aes(x = flipper_length_mm, y
= body_mass_g)) +
          geom_point(color = input$pt_color)

  })

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

7. Let's add a reactive summary table for male and female penguins of the species selected, which shows up below the graph for that species.

a. First, create a reactive subset of the data (in the server) that you'll use to produce the table, and render the table in the server

```
server <- function(input, output) {

  penguin_select <- reactive({

            penguins %>%
            filter(species == input$species)

  })
```

```
  penguin_table <- reactive({
            penguins %>%
            filter(species == input$penguin_species) %>%
            group_by(sex) %>%
            summarize(
            mean_flip = mean(flipper_length_mm),
            mean_mass = mean(body_mass_g)
            )
  })

  # Create a reactive plot, which depends on 'species'
widget selection:
  output$penguin_plot <- renderPlot({

            ggplot(data = penguin_select(), aes(x =
flipper_length_mm, y = body_mass_g)) +
            geom_point(color = input$pt_color)

  })

  output$penguin_table <- renderTable({

            penguin_table()

  })

}
```

b. Then call it to show up in the UI:

```
ui <- fluidPage(
  titlePanel("I am adding a title!"), # This is the title!
  sidebarLayout( # Adding a sidebar & main panel
            sidebarPanel("put my widgets here",
                         radioButtons(inputId = "species",
label = "Choose penguin species", choices =
c("Adelie","Gentoo","Cool Chinstrap Penguins!" =
"Chinstrap"), # This is my first widget for penguins species
                         ),
```

```
                    selectInput(inputId = "pt_color",
label = "Select point color", choices = c("Awesome red!" =
"red", "Pretty purple" = "purple", "ORAAANGE" = "orange"))
                    ),
           mainPanel("put my graph here", # Adding things to
the main panel
             plotOutput(outputId = "penguin_plot"),
             tableOutput(outputId = "penguin_table")
             )
  )
)
```

c. Run the app, and see that an (unfinished) table shows up! For nicer tables, consider:

Package DT, gt, renderDataTable, reactable