Practice Set 3

Kauila Tengan

Due by 10pm ET on Friday

Practice Set Information

During the week, you will get further practice with the material by working through the Practice Set, a set of problems designed to give you practice beyond the examples produced in the text.

You may work through these problems with peers, but all work must be completed by you (see the Honor Code in the syllabus) and you must indicate who you worked with below.

Even then, the best approach here is to try the problems on your own before discussing them with peers, and then write your final solutions yourself.

GitHub Workflow

1. Before editing this file, verify you are working on the copy saved in *your* repo for the course (check the filepath and the project name in the top right corner).
2. Before editing this file, make an initial commit of the file to your repo to add your copy of the problem set.
3. Change your name at the top of the file and get started!
4. You should *save, knit, and commit* the .Rmd file each time you've finished a question, if not more often. You should also *push* your commits back onto GitHub occasionally (you can do this after each commit).
5. When you think you are done with the assignment, save the pdf as "*Name_thisfilename_date*.pdf" before committing and pushing (this is generally good practice but also helps me in those times where I need to download all student homework files).

Gradescope Upload

For each question (e.g., 3.1), allocate all pages associated with the specific question. If your work for a question runs onto a page that you did not select, you may not get credit for the work. If you do not allocate *any* pages when you upload your pdf, you may get a zero for the assignment.

You can resubmit your work as many times as you want before the deadline, so you should not wait until the last minute to submit some version of your work. Unexpected delays/crises that occur on the day the assignment is due do not warrant extensions (please submit whatever you have done to receive partial credit).

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (notes, textbook, etc) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

-

Problem 1 **Shiny app: Skateboard** Complete the app updates you started on Tuesday! Open the *lab07b-shiny-skateboards* folder and the corresponding *app.R* file. Choose one of the options below to update your app. When your updates are complete, copy your final code into the chunk below and upload the pdf to Gradescope. Then, publish your app reply to the appropriate thread on Campuswire with (1) the URL to your published Shiny app; and (2) a paragraph explaining what story your Shiny app is telling, and how the interactivity you created enhances the telling of that story.

Option 1: Update the template to still explore the skateboards dataset, but with different app functionality. Update (1) at least 2 different widgets and (2) either the layout (e.g. not in tabs or different page layout) or the theme (check out the **shinythemes** package). Like a challenge? Incorporate one of the click, hover, or brush features (read about plot interactions).

Option 2: use this as a template to create a Shiny app for a one of the following datasets from the **fivethirtyeight** package: `candy_rankings` (candy characteristics and popularity); `hate_crimes` (hate crimes in US states, 2010-2015); `mad_men` (tv performers and their post-show career), `ncaa_w_bball_tourney` (women's NCAA div 1 basketball tournament, 1982-2018); or `nfl_suspensions` (NFL suspensions, 1946-2014). The data dictionaries are included in pdfs on Moodle.

```
# Load necessary packages
library(shiny)
library(shinythemes)
library(tidyverse)
library(DT)
library(ggrepel)
library(shinyWidgets)
#library(fivethirtyeight)

# Import data
skateboards <- read_csv("electric_skateboards.txt")

############################################################
# Define choice values and labels for widgets (user inputs) #
# - Define vectors for choice values and labels             #
# - Can then refer to them in server                        #
############################################################

# For TAB 1 HISTOGRAM widgets:

## For selectInput, 'choices' object should be a NAMED LIST
hist_choice_values <- c("price", "range", "top_speed", "weight", "battery")
hist_choice_names <- c("Price", "Range", "Top Speed", "Weight", "Battery")
names(hist_choice_values) <- hist_choice_names

## For checkboxGroupInput
drv_choices <-  unique(skateboards$drive)


# For TAB 2 SCATTERPLOT widgets:

## For radio button
size_choice_values <- c("price", "weight", "battery")
```

```
size_choice_names <- c("Price", "Weight", "Battery")
names(size_choice_values) <- size_choice_names

## For selectizeInput choices for skateboard name, pull directly from data
name_choices <- unique(skateboards$board)


# For TAB 3 TABLE widgets:

## For selectizeInput choices for company name, pull directly from data
cmpy_choices <- unique(skateboards$company)

#############
#    ui     #
#############
ui <- navbarPage(
  title = "Electric Skateboards",
  theme = shinytheme("superhero"),


  # Tab 1: Histogram
  tabPanel(
    title = "Histogram",

    sidebarLayout(
      sidebarPanel(

        selectInput(inputId = "histvar",
                    label = "Choose a variable of interest to plot:",
                    choices = hist_choice_values,
                    selected = "price"),

        sliderInput(inputId = "binsize",
                    label = "Slide to Change the Histogram Binwidth",
                    min = 20,
                    max = 100,
                    value = 30),

        checkboxGroupInput(inputId = "drv",
                           label = "Include drive types:",
                           choices = drv_choices,
                           selected = drv_choices,
                           inline = TRUE),

        # prettyCheckbox(
        #   inputId = "drv", label = "Include drive types:", icon = icon("thumbs-up"),
        #   status = "default", shape = "curve", animation = "pulse",
        #   choices = drv_choices, selected = drv_choices, inline = TRUE
        # ),

        textInput(inputId = "title",
                  label = "Write a title",
                  value = "Histogram")
```

```
    ),

    mainPanel(plotOutput(outputId = "hist"))
  )
),

# Tab 2: Scatterplot
tabPanel(
  title = "Scatterplot",

  sidebarLayout(

    sidebarPanel(
      radioButtons(inputId = "pt_size",
                   label = "Size points by:",
                   choices = size_choice_values,
                   selected = "weight"),

      selectizeInput(inputId = "id_name",
                     label = "Identify skateboard(s) in the scatterplot:",
                     choices = name_choices,
                     selected = NULL,
                     multiple = TRUE)
    ),

    mainPanel(plotOutput(outputId = "scatter"))
  )
),

# Tab 3: Table
tabPanel(
  title = "Table",

  sidebarLayout(
    sidebarPanel(
      selectizeInput(inputId = "cmpy",
                     label = "Choose one or more companies:",
                     choices = cmpy_choices,
                     selected = "DIYElectric",
                     multiple = TRUE)
    ),

    mainPanel(DT::dataTableOutput(outputId = "table"))
  )
),

# Tab 4: Original graph
tabPanel(
  title = "Original Graph",

  sidebarLayout(
    sidebarPanel(
      tags$div(
```

```
          HTML(paste("Original figure was presented by ",
                     tags$a(href="https://www.electricskateboardhq.com/boards-comparison/",
                            "HQ Skateboard"),
                     sep = "")
          )
        )
      ),

      mainPanel(h3("Information overload!"),
                plotOutput(outputId = "original")
      )
    )
  )

)


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

  # TAB 1: HISTOGRAM
  data_for_hist <- reactive({
    data <- filter(skateboards, drive %in% input$drv)
  })

  output$hist <- renderPlot({
    ggplot(data = data_for_hist(), aes_string(x = input$histvar)) +
      geom_histogram(color = "#2c7fb8", fill = "#7fcdbb", alpha = 0.7, binwidth = input$binsize) +
      labs( title = input$title,
            x = hist_choice_names[hist_choice_values == input$histvar],
            y = "Number of Skateboards")
  })

  # TAB 2: INTERACTIVE SCATTERPLOT
  output$scatter <- renderPlot({
    skateboards %>%
      filter(drive != "Direct") %>%
    ggplot(aes_string(x = "range", y = "top_speed", size = input$pt_size)) +
      geom_point(color = "#2c7fb8") +
      labs(title = "Electric Skateboards",
           subtitle = "August 2018",
           x = "Range (miles)",
           y = "Top Speed (mph)",
           size = size_choice_names[size_choice_values == input$pt_size]) +
      geom_label_repel(data = filter(skateboards, board %in% input$id_name),
                       aes(label = board), show.legend = FALSE) +
      facet_grid(~drive)
  })

  # TAB 3: TABLE
  data_for_table <- reactive({
```

```
    data <- filter(skateboards, company %in% input$cmpy)
  })

  output$table <- DT::renderDataTable({
    data_for_table()
  })

  # TAB 4: RE-CREATION OF ORIGINAL FIGURE (STATIC)
  output$original <- renderPlot({
    ggplot(data = skateboards, aes(x = range, y = top_speed,
                                   color = company,
                                   shape = drive,
                                   size = weight)) +
      geom_point() +
      geom_text(aes(label = board), hjust = 0, nudge_x = 0.05, size = 3) +
      labs(title = "Electric Skateboards",
           subtitle = "August 2018",
           x = "Range (miles)",
           y = "Top Speed (mph)",
           shape = "Drive type",
           size = "Weight of board") +
      guides(color = FALSE)
  })
}

####################
# call to shinyApp #
####################
shinyApp(ui = ui, server = server)
```