Homework 6

Name: Arthur Viegas Eguia

I worked with: Jacob Aronson (from the stats help lab)

Click the "Knit" button in RStudio to knit this file to a pdf.

Problem 1: Crimes

Scrape the table of data found at https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate and create a plot showing property crime rate (total property crime) vs. violent crime rate (total violent crime). Identify outlier cities by using a plotting command similar to the one below. (Don't blindly use this without thinking about the column names.)

```
> crimes_table <- bow("https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate", user_a
+   scrape() %>%
+   html_table() %>%
+   .[[1]]
>
>
> crimes_table_clean <- crimes_table %>%
+   set_names(.[1,]) %>%
+   janitor::clean_names() %>% # Clean names
+   slice(c(-1, -2)) %>% # Remove the first row
+   mutate(across(everything(), ~na_if(.x, ""))) %>% # Convert empty strings to NA
+   type.convert(as.is = TRUE) %>% # Convert columns to their most appropriate type
+   mutate_at(vars(c(9, 14)), parse_number)
>
> crimes <- data.frame(city = crimes_table_clean[[2]],
+                      violent_crime = crimes_table_clean[[9]],
+                      property_crime = crimes_table_clean[[14]])
```

```
> ggplot(crimes, aes(x = violent_crime, y = property_crime, label = city)) +
+     geom_point() +
+     geom_text(
+       data = filter(crimes, violent_crime > 1500 | property_crime > 6500),
+       check_overlap = TRUE, size = 2.5, nudge_y = 40
+     )
```

Problem 2: Movie scraping

```
> url <- "https://www.boxofficemojo.com/chart/ww_top_lifetime_gross/?offset=0&area=XWW"
```

a.

answer: There is a single table

```
> movies_df <- read_html(url) %>%
+   html_table() #The first item in the list is a tibble, so it is technically a data frame
> class(movies_df[[1]])
[1] "tbl_df"     "tbl"        "data.frame"
```

b.

answer: write your answer here

```
> movies_df_clean <- movies_df[[1]] %>%
+   janitor::clean_names() %>%
+   rename(world_dollars = worldwide_lifetime_gross,
+          domestic_dollars = domestic_lifetime_gross,
+          domestic_percentage = domestic_percent,
+          overseas_dollars = foreign_lifetime_gross,
+          overseas_percentage = foreign_percent) %>%
+   slice(1:100)
>
```

c.

answer:

```
> movie_df_right_types <- movies_df_clean %>%
+   mutate(world_dollars = parse_number(world_dollars),
+          domestic_dollars = parse_number(domestic_dollars),
+          domestic_percentage = parse_number(domestic_percentage),
+          overseas_dollars = parse_number(overseas_dollars),
+          overseas_percentage = parse_number(overseas_percentage))
> movie_df_right_types
# A tibble: 100 x 8
    rank title                world_dollars domestic_dollars domestic_percentage
   <int> <chr>                       <dbl>            <dbl>               <dbl>
 1     1 Avatar                 2923706026        785221649                26.9
 2     2 Avengers: Endgame      2799439100        858373000                30.7
 3     3 Avatar: The Way of ~   2320250281        684075767                29.5
 4     4 Titanic                2264750694        674292608                29.8
 5     5 Star Wars: Episode ~   2071310218        936662225                45.2
 6     6 Avengers: Infinity ~   2052415039        678815482                33.1
 7     7 Spider-Man: No Way ~   1921847111        814115070                42.4
 8     8 Jurassic World         1671537444        653406625                39.1
 9     9 The Lion King          1663079059        543638043                32.7
10    10 The Avengers           1520538536        623357910                41
# i 90 more rows
# i 3 more variables: overseas_dollars <dbl>, overseas_percentage <dbl>,
#   year <int>
```

d.

answer: The titanic position is going to be 26, and the URL is https://www.boxofficemojo.com/title/tt0120338/?ref_=bo_cso_table_4

```
> titanic_position <- read_html(url) %>%
+   html_nodes("a") %>%
+   html_text() %>%
```

```
+   str_to_lower() %>%
+   str_which("titanic")
>
```

```
> titanic_href <- str_c("https://www.boxofficemojo.com",
+                       read_html(url) %>%
+                       html_nodes("a") %>%
+                         html_attr("href")%>%
+                         .[titanic_position])
> titanic_href
[1] "https://www.boxofficemojo.com/title/tt0120338/?ref_=bo_cso_table_4"
```

e.

```
> temp_url <- "https://www.boxofficemojo.com/chart/ww_top_lifetime_gross/?offset=#&area=XWW"
```

answer: Indeed,"https://www.boxofficemojo.com/chart/ww_top_lifetime_gross/?offset=200&area=XWW" (which is just adding offset = 200) takes me to the page with movies starting at 201.

```
> pull_table <- function(url_input){
+   modified_df <- read_html(url_input) %>%
+   html_table() %>%
+     .[[1]] %>%
+     janitor::clean_names() %>%
+     rename(world_dollars = worldwide_lifetime_gross,
+            domestic_dollars = domestic_lifetime_gross,
+            domestic_percentage = domestic_percent,
+            overseas_dollars = foreign_lifetime_gross,
+            overseas_percentage = foreign_percent) %>%
+            mutate(across(c(3, 4, 5, 6, 7), parse_number))
+   ranks <- c()
+   for(i in modified_df$rank){
+     if(is.character(i)){
+       ranks <- c(ranks, parse_number(i))
+     }
+     else{
+       ranks <- c(ranks, as.numeric(i))
+     }
+   }
+   modified_df$rank <- ranks
+   return(modified_df)
+ }
>
> test_url <- "https://www.boxofficemojo.com/chart/ww_top_lifetime_gross/?offset=800#&area=XWW"
>
> pull_table(test_url)
# A tibble: 200 x 8
    rank title                world_dollars domestic_dollars domestic_percentage
   <dbl> <chr>                        <dbl>            <dbl>               <dbl>
 1   801 The Prince of Egypt      218613188        101413188                46.4
 2   802 Jack Reacher             218340595         80070736                36.7
 3   803 Kingdom of Heaven        218237071         47398413                21.7
 4   804 Smallfoot                218015531         83315531                38.2
 5   805 The Emoji Movie          217776646         86089513                39.5
```

```
 6   806 Smile                    217408513        105935048                48.7
 7   807 Too Cool to Kill         217254604           185882                 0.1
 8   808 Dracula Untold           217124280         56280355                25.9
 9   809 Central Intelligence     216940871        127440871                58.7
10   810 Million Dollar Baby      216763646        100492203                46.4
# i 190 more rows
# i 3 more variables: overseas_dollars <dbl>, overseas_percentage <dbl>,
#   year <int>
```

f.

answer:

```
> all_url <- "https://www.boxofficemojo.com/chart/ww_top_lifetime_gross/?offset="
> url_last_part <- "#&area=XWW"
> idx <- seq(0, 800, by = 200)
> movie_df <- map_df(idx, ~{
+   new_url <- str_glue("{all_url}{.x}{url_last_part}")
+   pull_table(new_url)
+ })
> glimpse(movie_df)
Rows: 1,000
Columns: 8
$ rank               <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,~
$ title              <chr> "Avatar", "Avengers: Endgame", "Avatar: The Way of~
$ world_dollars      <dbl> 2923706026, 2799439100, 2320250281, 2264750694, 20~
$ domestic_dollars   <dbl> 785221649, 858373000, 684075767, 674292608, 936662~
$ domestic_percentage <dbl> 26.9, 30.7, 29.5, 29.8, 45.2, 33.1, 42.4, 39.1, 32~
$ overseas_dollars   <dbl> 2138484377, 1941066100, 1636174514, 1590458086, 11~
$ overseas_percentage <dbl> 73.1, 69.3, 70.5, 70.2, 54.8, 66.9, 57.6, 60.9, 67~
$ year               <int> 2009, 2019, 2022, 1997, 2015, 2018, 2021, 2015, 20~
> head(movie_df)
# A tibble: 6 x 8
   rank title                world_dollars domestic_dollars domestic_percentage
  <dbl> <chr>                        <dbl>            <dbl>               <dbl>
1     1 Avatar                  2923706026        785221649                26.9
2     2 Avengers: Endgame       2799439100        858373000                30.7
3     3 Avatar: The Way of W~   2320250281        684075767                29.5
4     4 Titanic                 2264750694        674292608                29.8
5     5 Star Wars: Episode V~   2071310218        936662225                45.2
6     6 Avengers: Infinity W~   2052415039        678815482                33.1
# i 3 more variables: overseas_dollars <dbl>, overseas_percentage <dbl>,
#   year <int>
```

Problem 3: Penguins

```
> ui <- fluidPage(
+   plotOutput("plot", height = 500)
+ )
>
> server <- function(input, output){
+   output$plot <- renderPlot({
+     g <- ggplot(penguins, aes(x = bill_length_mm, y = body_mass_g))
```

```
+     g + geom_point()
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

a.

answer:

```
> ui <- fluidPage(
+   checkboxInput("check", "Include color?", value = FALSE),
+   plotOutput("plot", height = 500)
+ )
>
> server <- function(input, output){
+   output$plot <- renderPlot({
+     if(input$check){
+       g <- ggplot(penguins, aes(x = bill_length_mm, y = body_mass_g, color = species))
+     }
+     else{
+        g <- ggplot(penguins, aes(x = bill_length_mm, y = body_mass_g))
+     }
+     g + geom_point()
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

b.

answer:

```
> ui <- fluidPage(
+   checkboxInput("check", "Include color?", value = FALSE),
+   varSelectInput("selectInputX", "X value", selected = "bill_length_mm", data = penguins %>% select(3
+   varSelectInput("selectInputY", "Y value", selected = "body_mass_g", data = penguins %>% select(3:6)
+   plotOutput("plot", height = 500)
+ )
>
> server <- function(input, output){
+   output$plot <- renderPlot({
+     if(input$check){
+       g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY, color = species))
+     }
+     else{
+        g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY))
+     }
+     g + geom_point()
+   })
+ }
>
```

```
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

c.

answer:

```
> ui <- fluidPage(
+   checkboxInput("check", "Include color?", value = FALSE),
+   varSelectInput("selectInputX", "X value", selected = "bill_length_mm", data = penguins %>% select(3
+   varSelectInput("selectInputY", "Y value", selected = "body_mass_g", data = penguins %>% select(3:6)
+   plotOutput("plot", height = 500, click = "my_click"),
+   dataTableOutput("my_table")
+ )
>
> server <- function(input, output){
+   output$plot <- renderPlot({
+     if(input$check){
+       g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY, color = species))
+     }
+     else{
+        g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY))
+     }
+     g + geom_point()
+   })
+   output$my_table <- renderDataTable({
+     nearPoints(penguins, input$my_click)
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

d.

answer:

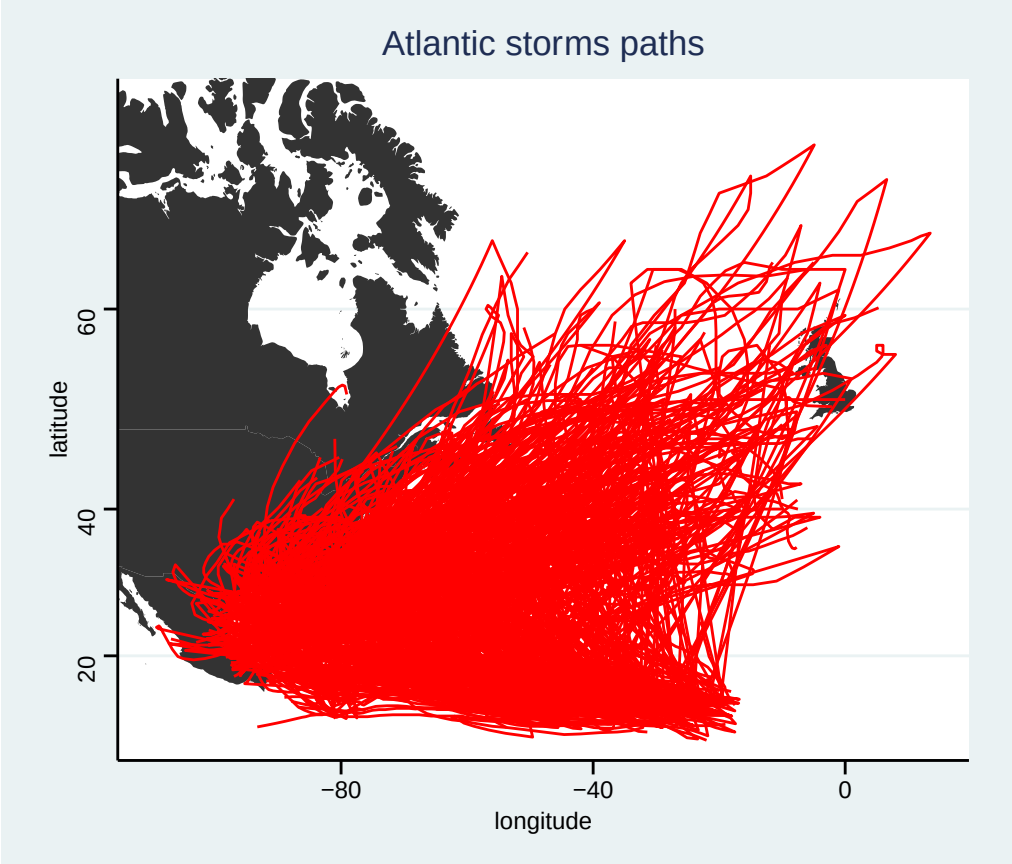
```
> ui <- fluidPage(
+   checkboxInput("check", "Include color?", value = FALSE),
+   varSelectInput("selectInputX", "X value", selected = "bill_length_mm", data = penguins %>% select(3
+   varSelectInput("selectInputY", "Y value", selected = "body_mass_g", data = penguins %>% select(3:6)
+   plotOutput("plot", height = 500, brush = "my_brush"),
+   dataTableOutput("my_table")
+ )
>
> server <- function(input, output){
+   output$plot <- renderPlot({
+     if(input$check){
+       g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY, color = species))
+     }
+     else{
+        g <- ggplot(penguins, aes(x = !!input$selectInputX, y = !!input$selectInputY))
+     }
+     g + geom_point()
+   })
```

```
+   output$my_table <- renderDataTable({
+     brushedPoints(penguins, input$my_brush)
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

Problem 4: Storm paths

Special thanks to Jacob Aaronson

```
> ctry <- map_data("world",
+                  region = c(
+                    "usa",
+                    "mexico",
+                    "canada",
+                    "uk"
+                  ))
> base_map <- ggplot(ctry) +
+   geom_polygon(aes(x = long,  y = lat, group = group)) +
+   labs(
+     x = "longitude",
+     y = "latitude",
+     title = "Atlantic storms paths"
+     )
>
> base_map +
+   geom_path(data = storms, aes(x = long, y = lat, group = name), color = "red") +
+   coord_map(xlim  = c(min(storms$long), max(storms$long)),
+             ylim  = c(min(storms$lat), max(storms$lat)))
```




```
> head(storms)
# A tibble: 6 x 13
  name   year month   day  hour   lat  long status       category  wind pressure
  <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>           <dbl> <int>    <int>
1 Amy    1975     6    27     0  27.5 -79   tropical de~       NA    25     1013
2 Amy    1975     6    27     6  28.5 -79   tropical de~       NA    25     1013
3 Amy    1975     6    27    12  29.5 -79   tropical de~       NA    25     1013
4 Amy    1975     6    27    18  30.5 -79   tropical de~       NA    25     1013
5 Amy    1975     6    28     0  31.5 -78.8 tropical de~       NA    25     1012
6 Amy    1975     6    28     6  32.4 -78.7 tropical de~       NA    25     1012
# i 2 more variables: tropicalstorm_force_diameter <int>,
#   hurricane_force_diameter <int>
```

a.

answer:

```
> Individualstorm <- storms %>%
+       filter(name == "Amy") %>%
+   group_by(name) %>%
+   mutate(date = make_datetime(year = year,
+                               month = month,
+                               day = day,
+                               hour = hour)) %>%
+   mutate(ellapsed_time = difftime(date, min(date), units = "hours")) %>%
+   mutate(ellapsed_time = as.numeric(ellapsed_time))
>
```

```
> ui <- fluidPage(
+   sliderInput("mySlider", "Storm Time", min = 0, max = 180, value = 90, step = 6),
+   plotOutput("plot", height = 500)
+
+ )
>
> server <- function(input, output){
+
+   ellapsedTime <- reactive({filter(Individualstorm, ellapsed_time <= input$mySlider)})
+   output$plot <- renderPlot({
+     base_map +  geom_path(data = ellapsedTime(), aes(x = long, y = lat, group = name), color = "red")
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

b.

answer:

```
> ui <- fluidPage(
+   selectInput("selectStorm", "Storm Name", selected = "Amy", choices = unique(storms$name)),
+   plotOutput("plot", height = 500)
+ )
>
> server <- function(input, output){
+
+   stormOfInterest <- reactive({filter(storms, name == input$selectStorm)})
+   output$plot <- renderPlot({
+     base_map + geom_path(data = stormOfInterest(), aes(x = long, y = lat), color = "red")
+   })
+ }
>
> # you can modify the height to avoid scrolling
> shinyApp(ui, server, options = list(height = 600))
```

```
> #  coord_map(xlim  = c(min(long), max(long)),
> #            ylim  = c(min(lat), max(lat)), data = stormOfInterest)
```