Test notes

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

pokemon.data <- read.csv("Pokemon.csv")

### Test 2

# Question 1

# We must give a brief description of a pokemon

# User must type in name of pokemon, and the app will return

# first wikipedia paragraph on that pokemon.

"https://en.wikipedia.org/wiki/Charizard" %>%

read\_html() %>%

html\_nodes(xpath = '//\*[@id="mw-content-text"]/div/p[1]') %>%

html\_text

test <- paste("https://en.wikipedia.org/wiki/","Charizard", sep = "")

test %>%

read\_html() %>%

html\_nodes(xpath = '//\*[@id="mw-content-text"]/div/p[1]') %>%

html\_text

paste("https://en.wikipedia.org/wiki/","Charizard", "Charizard", "\_Pokémon", sep = "")

### Question 2

# First, add ranking columns

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.stats <- pokemon.data.test %>%

filter(Name == "Pikachu")

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

App

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

library(dplyr)

pokemon.data <- read.csv("Pokemon.csv")

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

titlePanel("Pokemon Selection"),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL)

),

# Show a plot of the generated distribution

mainPanel(

textOutput("pokemon.description")

)

)),

tabPanel("Skills Graph",

titlePanel("Pokemon Skills Graph"),

mainPanel(

plotOutput("skills.graph")

))

)

)

# Define server logic

server <- function(input, output) {

# WORK ON getting the output to wait until hit enter

output$pokemon.description <-

renderText({

expr = paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

})

}

# Run the application

shinyApp(ui = ui, server = server)

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(dplyr)

library(rgdal)

library(geojsonio)

library(geojson)

pokemon.data <- read.csv("Pokemon.csv")

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Assigning each pokemon a random number between 1:50 for states

random.numbers <- sample(1:50, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

## Preparing states data for Questions 3 and 4

states.url <- "http://eric.clst.org/assets/wiki/uploads/Stuff/gz\_2010\_us\_040\_00\_5m.json"

states <- geojson\_read(states.url, what = "sp")

states.pokemon <- states

random.numbers <- sample(1:52, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

pokemon.in.state <- NULL

for (i in 1:52) {

pokemon.in.state[i] <- pokemon.data.test %>%

filter(state.number == i) %>%

nrow()

}

states.pokemon <- states

states.list <- c("Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",

"Delaware", "District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho,", "Illinois", "Indiana", "Iowa",

"Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",

"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",

"New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",

"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico", "Rhode Island",

"South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",

"Washington", "West Virginia", "Wisconsin", "Wyoming")

## Adding column to pokemon.data.test for corresponding state

state.number <- seq(1,52,1)

new.states.list <- data.frame(states.list,

state.number)

pokemon.data.test <- left\_join(pokemon.data.test, new.states.list, by = "state.number")

# Now include pokemon density in state table, to be used for color function

state.density.table <- data.frame(new.states.list, pokemon.in.state)

## Making function to color states by density in leaflet

states.pokemon@data <- states@data %>%

mutate(thecolor = state.density.table$pokemon.in.state)

factpal <- colorFactor("Greens", states.pokemon@data$thecolor)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

titlePanel("Pokemon Selection"),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL)

),

# Show a plot of the generated distribution

mainPanel(

textOutput("pokemon.description")

)

)),

tabPanel("Skills Graph",

titlePanel("Pokemon Skills Graph"),

mainPanel(

plotOutput("skills.graph")

)),

tabPanel("Gotta Catch em All",

titlePanel("Pokemon Density in the United States"),

mainPanel(

textOutput("pokemon.description")

))

)

)

# Define server logic

server <- function(input, output) {

# WORK ON getting the output to wait until hit enter

output$pokemon.description <-

renderText({

expr = paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

})

}

# Run the application

shinyApp(ui = ui, server = server)

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

library(dplyr)

pokemon.data <- read.csv("Pokemon.csv")

pokemon.data.test <- pokemon.data

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Assigning each pokemon a random number between 1:50 for states

random.numbers <- sample(1:50, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

## Preparing states data for Questions 3 and 4

states.url <- "http://eric.clst.org/assets/wiki/uploads/Stuff/gz\_2010\_us\_040\_00\_5m.json"

states <- geojson\_read(states.url, what = "sp")

states.pokemon <- states

states.pokemon <- states

states.list <- c("Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",

"Delaware", "District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa",

"Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",

"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",

"New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",

"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico", "Rhode Island",

"South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",

"Washington", "West Virginia", "Wisconsin", "Wyoming")

## Adding column to pokemon.data.test for corresponding state

sample(states.list, 800, replace = TRUE)

pokemon.data.test$state <- sample(states.list, 800, replace = TRUE)

new.states.list <- data.frame(c(1:52), states.list)

pokemon.data.test <- left\_join(pokemon.data.test, new.states.list, by = c("state" = "states.list"))

pokemon.data.test$state.number <- pokemon.data.test$c.1.52.

pokemon.data.test$c.1.52. <- NULL

pokemon.in.state <- NULL

for (i in 1:52) {

pokemon.in.state[i] <- pokemon.data.test %>%

filter(state.number == i) %>%

nrow()

}

# Now include pokemon density in state table, to be used for color function

state.density.table <- data.frame(new.states.list, pokemon.in.state)

## Making function to color states by density in leaflet

states.pokemon@data <- states@data %>%

mutate(thecolor = state.density.table$pokemon.in.state)

factpal <- colorFactor("Greens", states.pokemon@data$thecolor)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

titlePanel("Pokemon Selection"),

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL)),

mainPanel(

textOutput("pokemon.description")

)

)),

tabPanel("Skills Graph",

titlePanel("Pokemon Skills Graph"),

mainPanel(

plotOutput("skills.graph")

)),

tabPanel("Map Page",

titlePanel("Map stuff"),

sidebarLayout(

sidebarPanel(

verbatimTextOutput("text1"),

actionButton(inputId = "button1",

label = "Catch em!")),

mainPanel(leafletOutput("map"),

verbatimTextOutput("pokemon.catch")))

)

)

)

# Define server logic

server <- function(input, output) {

# WORK ON getting the output to wait until hit enter

output$pokemon.description <-

renderText({

expr = paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- NULL

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

})

output$map <- renderLeaflet({

leaflet(states.pokemon) %>%

setView(-96, 37.8, 4) %>%

addPolygons(stroke = FALSE, smoothFactor = 0.2, fillOpacity = 1,

color = ~factpal(thecolor),

weight = 2,

opacity = 1,

dashArray = "1",

highlight = highlightOptions(

weight = 5,

color = "#666",

dashArray = "",

fillOpacity = 0.7,

bringToFront = FALSE),

layerId = states.pokemon@data$NAME)

})

output$text1 <- renderPrint({

if(is.null(input$map\_shape\_click$id)){

print("Select a state")}

else{

pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)}

})

output$pokemon.catch <- renderPrint({

})

}

# Run the application

shinyApp(ui = ui, server = server)

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

library(dplyr)

pokemon.data <- read.csv("Pokemon.csv")

pokemon.data.test <- pokemon.data

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Assigning each pokemon a random number between 1:50 for states

random.numbers <- sample(1:50, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

## Preparing states data for Questions 3 and 4

states.url <- "http://eric.clst.org/assets/wiki/uploads/Stuff/gz\_2010\_us\_040\_00\_5m.json"

states <- geojson\_read(states.url, what = "sp")

states.pokemon <- states

states.pokemon <- states

states.list <- c("Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",

"Delaware", "District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa",

"Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",

"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",

"New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",

"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico", "Rhode Island",

"South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",

"Washington", "West Virginia", "Wisconsin", "Wyoming")

## Adding column to pokemon.data.test for corresponding state

sample(states.list, 800, replace = TRUE)

pokemon.data.test$state <- sample(states.list, 800, replace = TRUE)

new.states.list <- data.frame(c(1:52), states.list)

pokemon.data.test <- left\_join(pokemon.data.test, new.states.list, by = c("state" = "states.list"))

pokemon.data.test$state.number <- pokemon.data.test$c.1.52.

pokemon.data.test$c.1.52. <- NULL

pokemon.in.state <- NULL

for (i in 1:52) {

pokemon.in.state[i] <- pokemon.data.test %>%

filter(state.number == i) %>%

nrow()

}

# Now include pokemon density in state table, to be used for color function

state.density.table <- data.frame(new.states.list, pokemon.in.state)

## Making function to color states by density in leaflet

states.pokemon@data <- states@data %>%

mutate(thecolor = state.density.table$pokemon.in.state)

factpal <- colorFactor("Greens", states.pokemon@data$thecolor)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

titlePanel("Pokemon Selection"),

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL)),

mainPanel(

textOutput("pokemon.description")

)

)),

tabPanel("Skills Graph",

titlePanel("Pokemon Skills Graph"),

mainPanel(

plotOutput("skills.graph")

)),

tabPanel("Map Page",

titlePanel("Map stuff"),

sidebarLayout(

sidebarPanel(

verbatimTextOutput("text1"),

actionButton(inputId = "button1",

label = "Catch em!")),

mainPanel(leafletOutput("map"),

verbatimTextOutput("pokemon.catch")

))

))

)

# Define server logic

server <- function(input, output) {

# WORK ON getting the output to wait until hit enter

output$pokemon.description <-

renderText({

expr = paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- NULL

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

})

output$map <- renderLeaflet({

leaflet(states.pokemon) %>%

setView(-96, 37.8, 4) %>%

addPolygons(stroke = FALSE, smoothFactor = 0.2, fillOpacity = 1,

color = ~factpal(thecolor),

weight = 2,

opacity = 1,

dashArray = "1",

highlight = highlightOptions(

weight = 5,

color = "#666",

dashArray = "",

fillOpacity = 0.7,

bringToFront = FALSE),

layerId = states.pokemon@data$NAME)

})

output$text1 <- renderPrint({

if(is.null(input$map\_shape\_click$id)){

print("Select a state")}

else{

pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)}

})

observeEvent(input$button1, {

pokemons <- pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)

output$pokemon.catch <- renderPrint(

paste("Congrats! You caught a",as.character(pokemons[sample(1:nrow(pokemons),1),]))

)

})

}

# Run the application

shinyApp(ui = ui, server = server)

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

library(dplyr)

pokemon.data <- read.csv("Pokemon.csv")

pokemon.data.test <- pokemon.data

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Assigning each pokemon a random number between 1:50 for states

random.numbers <- sample(1:50, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

## Preparing states data for Questions 3 and 4

states.url <- "http://eric.clst.org/assets/wiki/uploads/Stuff/gz\_2010\_us\_040\_00\_5m.json"

states <- geojson\_read(states.url, what = "sp")

states.pokemon <- states

states.pokemon <- states

states.list <- c("Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",

"Delaware", "District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa",

"Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",

"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",

"New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",

"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico", "Rhode Island",

"South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",

"Washington", "West Virginia", "Wisconsin", "Wyoming")

## Adding column to pokemon.data.test for corresponding state

sample(states.list, 800, replace = TRUE)

pokemon.data.test$state <- sample(states.list, 800, replace = TRUE)

new.states.list <- data.frame(c(1:52), states.list)

pokemon.data.test <- left\_join(pokemon.data.test, new.states.list, by = c("state" = "states.list"))

pokemon.data.test$state.number <- pokemon.data.test$c.1.52.

pokemon.data.test$c.1.52. <- NULL

pokemon.in.state <- NULL

for (i in 1:52) {

pokemon.in.state[i] <- pokemon.data.test %>%

filter(state.number == i) %>%

nrow()

}

# Now include pokemon density in state table, to be used for color function

state.density.table <- data.frame(new.states.list, pokemon.in.state)

## Making function to color states by density in leaflet

states.pokemon@data <- states@data %>%

mutate(thecolor = state.density.table$pokemon.in.state)

factpal <- colorFactor("Greens", states.pokemon@data$thecolor)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

titlePanel("Pokemon Selection"),

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL),

actionButton(inputId = "button2",

label = "Search your Pokemon")),

mainPanel(

textOutput("pokemon.description")

)

)),

tabPanel("Skills Graph",

titlePanel("Pokemon Skills Graph"),

mainPanel(

plotOutput("skills.graph")

)),

tabPanel("Map Page",

titlePanel("Map stuff"),

sidebarLayout(

sidebarPanel(

verbatimTextOutput("text2"),

verbatimTextOutput("text1"),

actionButton(inputId = "button1",

label = "Catch em!")),

mainPanel(leafletOutput("map"),

verbatimTextOutput("pokemon.catch")

))

))

)

# Define server logic

server <- function(input, output) {

observeEvent(input$button2, {

search.poke <- paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

output$pokemon.description <- renderText({

search.poke

})

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- NULL

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none")

})

output$map <- renderLeaflet({

leaflet(states.pokemon) %>%

setView(-96, 37.8, 4) %>%

addPolygons(stroke = FALSE, smoothFactor = 0.2, fillOpacity = 1,

color = ~factpal(thecolor),

weight = 2,

opacity = 1,

dashArray = "1",

highlight = highlightOptions(

weight = 5,

color = "#666",

dashArray = "",

fillOpacity = 0.7,

bringToFront = FALSE),

layerId = states.pokemon@data$NAME)

})

### Intro text

observeEvent(input$map\_shape\_click$id, {

correct.state <- input$map\_shape\_click$id

output$text2 <- renderPrint(

paste("The pokemon currently residing in", as.character(input$map\_shape\_click$id), "are")

)

})

### The original

output$text1 <- renderPrint({

if(is.null(input$map\_shape\_click$id)){

print("Select a state")}

else{

pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)}

})

observeEvent(input$button1, {

pokemons <- pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)

output$pokemon.catch <- renderPrint(

paste("Congrats! You caught a",as.character(pokemons[sample(1:nrow(pokemons),1),]))

)

})

}

# Run the application

shinyApp(ui = ui, server = server)

library(data.table)

library(ggplot2)

library(stringr)

library(tidyverse)

library(rvest)

library(readxl)

library(httr)

library(shiny)

library(plotly)

library(leaflet)

library(geojsonio)

library(dplyr)

library(shinyWidgets)

library(png)

pokemon.data <- read.csv("Pokemon.csv")

pokemon.data.test <- pokemon.data

# Adding ranking columns corresponding to variables HP, Attack, Defence, Sp. Atk,

# Sp. Def, and Speed

pokemon.data.test$HPRank <- NULL

order.scores<-order(pokemon.data.test$HP, pokemon.data.test$Name)

pokemon.data.test$HPRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$AttackRank <- NULL

order.scores <- order(pokemon.data.test$Attack, pokemon.data.test$Name)

pokemon.data.test$AttackRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$DefenseRank <- NULL

order.scores<-order(pokemon.data.test$Defense, pokemon.data.test$Name)

pokemon.data.test$DefenseRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..AtkRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Atk, pokemon.data.test$Name)

pokemon.data.test$Sp..AtkRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$Sp..DefRank <- NULL

order.scores<-order(pokemon.data.test$Sp..Def, pokemon.data.test$Name)

pokemon.data.test$Sp..DefRank[order.scores] <- 1:nrow(pokemon.data.test)

pokemon.data.test$SpeedRank <- NULL

order.scores<-order(pokemon.data.test$Speed, pokemon.data.test$Name)

pokemon.data.test$SpeedRank[order.scores] <- 1:nrow(pokemon.data.test)

# Assigning each pokemon a random number between 1:50 for states

random.numbers <- sample(1:50, 800, replace = TRUE)

pokemon.data.test$state.number <- random.numbers

## Preparing states data for Questions 3 and 4

states.url <- "http://eric.clst.org/assets/wiki/uploads/Stuff/gz\_2010\_us\_040\_00\_5m.json"

states <- geojson\_read(states.url, what = "sp")

states.pokemon <- states

states.pokemon <- states

states.list <- c("Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",

"Delaware", "District of Columbia", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa",

"Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",

"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada",

"New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",

"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Puerto Rico", "Rhode Island",

"South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",

"Washington", "West Virginia", "Wisconsin", "Wyoming")

## Adding column to pokemon.data.test for corresponding state

sample(states.list, 800, replace = TRUE)

pokemon.data.test$state <- sample(states.list, 800, replace = TRUE)

new.states.list <- data.frame(c(1:52), states.list)

pokemon.data.test <- left\_join(pokemon.data.test, new.states.list, by = c("state" = "states.list"))

pokemon.data.test$state.number <- pokemon.data.test$c.1.52.

pokemon.data.test$c.1.52. <- NULL

pokemon.in.state <- NULL

for (i in 1:52) {

pokemon.in.state[i] <- pokemon.data.test %>%

filter(state.number == i) %>%

nrow()

}

# Now include pokemon density in state table, to be used for color function

state.density.table <- data.frame(new.states.list, pokemon.in.state)

## Making function to color states by density in leaflet

states.pokemon@data <- states@data %>%

mutate(thecolor = state.density.table$pokemon.in.state)

factpal <- colorFactor("Greens", states.pokemon@data$thecolor)

# Define UI for application

ui <- fluidPage(

tabsetPanel(

tabPanel("Pokemon Description",

setBackgroundImage("http://images6.fanpop.com/image/photos/39400000/1st-generation-pokemon-39423803-4724-2835.jpg"),

titlePanel("Pokemon Selection"),

sidebarLayout(

sidebarPanel(

textInput(inputId = "selection",

label = "Enter a Pokemon",

value = NULL),

actionButton(inputId = "button2",

label = "Search your Pokemon"),

textOutput("pokemon.description")),

mainPanel(textOutput('text3'),

tags$head(tags$style("#text3{color: black;

font-size: 30px;

font-style: oblique;

}"

)

))

)),

tabPanel("Skills Data ",

titlePanel(textOutput("text4")),

mainPanel(

plotOutput("skills.graph")

)),

tabPanel("Pokemon Map",

titlePanel("Pokemon in the United States"),

sidebarLayout(

sidebarPanel(

textOutput("text5"),

textOutput("text2"),

verbatimTextOutput("text1"),

actionButton(inputId = "button1",

label = "Catch em!"),

textOutput("pokemon.catch")),

mainPanel(textOutput("text6"), tags$head(tags$style("#text6{color: black;

font-size: 18px;

font-style: normal;

}"

)

),

leafletOutput("map")

))

))

)

# Define server logic

server <- function(input, output) {

observeEvent(input$button2, {

search.poke <- paste("https://pokemondb.net/pokedex/", input$selection, sep = "") %>%

read\_html() %>%

html\_nodes(xpath = '/html/body/main/div[1]/div[1]/p') %>%

html\_text

output$pokemon.description <- renderText({

search.poke

})

})

output$text3 <- renderText({

print("Go catch em all!")

})

observeEvent(input$selection, {

tab2.title <- paste("Skill data for", input$selection)

output$text4 <- renderText(

tab2.title

)

})

output$skills.graph <-

renderPlot({

pokemon.stats <- pokemon.data.test %>%

filter(Name == input$selection)

best.rank <- NULL

best.rank <- which.max(pokemon.stats[,14:19])

best.rank <- names(best.rank)

skill.frame <- data.frame(Skill = c("HPRank", "AttackRank", "DefenceRank", "Sp..AtkRank",

"Sp..DefRank", "SpeedRank"),

Rank = c(pokemon.stats$HPRank, pokemon.stats$AttackRank, pokemon.stats$DefenseRank,

pokemon.stats$Sp..AtkRank, pokemon.stats$Sp..DefRank, pokemon.stats$SpeedRank))

skill.frame %>%

ggplot(aes(x = Skill,

y = Rank,

fill = factor(ifelse(Skill == best.rank,

"Highlighted",

"Normal")))) +

geom\_bar(stat = "identity") +

theme(legend.position="none") +

xlab("Skills") +

ylab("Rank (out of all 800 pokemon)") +

ylim(0, 800)

})

observeEvent(input$selection, {

which.state <- paste0(input$selection, " is in ", pokemon.data.test %>%

filter(Name == input$selection) %>%

select(state),".")

output$text5 <- renderText(

which.state

)

})

output$text6 <- renderText({

"The density of pokemon living in each state, with darker hues representing increasing density"

})

output$map <- renderLeaflet({

leaflet(states.pokemon) %>%

setView(-96, 37.8, 4) %>%

addPolygons(stroke = FALSE, smoothFactor = 0.2, fillOpacity = 1,

color = ~factpal(thecolor),

weight = 2,

opacity = 1,

dashArray = "1",

highlight = highlightOptions(

weight = 5,

color = "#666",

dashArray = "",

fillOpacity = 0.7,

bringToFront = FALSE),

layerId = states.pokemon@data$NAME)

})

### Intro text

observeEvent(input$map\_shape\_click$id, {

correct.state <- input$map\_shape\_click$id

output$text2 <- renderText(

paste("The pokemon currently residing in", input$map\_shape\_click$id, "are:")

)

})

### The original

output$text1 <- renderPrint({

if(is.null(input$map\_shape\_click$id)){

print("Select a state")}

else{

pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)}

})

observeEvent(input$button1, {

pokemons <- pokemon.data.test %>%

filter(state == input$map\_shape\_click$id) %>%

select(Name)

output$pokemon.catch <- renderText(

paste("Congrats! You caught a",as.character(pokemons[sample(1:nrow(pokemons),1),]))

)

})

}

# Run the application

shinyApp(ui = ui, server = server)