app.R

bingo

2019-04-29

library(rgeos)

## Loading required package: sp

## rgeos version: 0.4-3, (SVN revision 595)  
## GEOS runtime version: 3.6.1-CAPI-1.10.1   
## Linking to sp version: 1.3-1   
## Polygon checking: TRUE

library(shiny)  
library(shinyjs)

##   
## Attaching package: 'shinyjs'

## The following object is masked from 'package:shiny':  
##   
## runExample

## The following object is masked from 'package:rgeos':  
##   
## show

## The following object is masked from 'package:sp':  
##   
## show

## The following objects are masked from 'package:methods':  
##   
## removeClass, show

library(httr)   
library(jsonlite)

##   
## Attaching package: 'jsonlite'

## The following object is masked from 'package:shiny':  
##   
## validate

library(googleway)  
library(plotly)

## Loading required package: ggplot2

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following objects are masked from 'package:googleway':  
##   
## add\_heatmap, add\_markers, add\_polygons

## The following object is masked from 'package:httr':  
##   
## config

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:rgeos':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggmap)

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

##   
## Attaching package: 'ggmap'

## The following object is masked from 'package:plotly':  
##   
## wind

library(ggplot2)  
#register google API key  
Sys.setenv('MAPBOX\_TOKEN' = 'pk.eyJ1IjoibWFzdGVyYmluZ28xIiwiYSI6ImNqdDluOHo2aDAxenQ0OW51dmdkOGIyaDkifQ.KMv1Wkds1VtmtzOOmMWuiw')  
register\_google(key = "AIzaSyDitfa2CtI\_rpIbpJviZRey63D0m7N3ZMA")  
#MAPBOX token  
#revised places function  
get\_poiVER2 <- function(location, radius, type, return\_n) {  
 key <- "AIzaSyDitfa2CtI\_rpIbpJviZRey63D0m7N3ZMA" #set api key for google places  
 doot <- geocode(location, output = c("latlon"), source = "google") #find location  
 testSearch <- google\_places(location = c(doot$lat, doot$lon), #commence search  
 keyword = type,  
 radius = radius\*1609.344,  
 key = key)  
 results <- cbind(testSearch$results$name, testSearch$results$rating, testSearch$results$geometry$location$lat, testSearch$results$geometry$location$lng)   
 results2 <- as.data.frame(results) #turn into dataframe  
 colnames(results2) <- c("Name", "Rating", "Latitude", "Longitude") #clean names  
 results3 <- results2 %>%  
 head(return\_n)  
 return(results3)  
}  
#used to pull capital bikeshare data  
get\_Capital <- function(url) {  
 doot <- fromJSON(url)  
 doot2 <- doot$data  
 doot3 <- do.call(what = "rbind",  
 args = lapply(doot2, as.data.frame))  
 return(doot3)  
}  
#define get closest for mapping  
  
# Define UI for application that draws a histogram  
  
ui <- fluidPage(  
 shinyjs::useShinyjs(),  
 theme = shinythemes::shinytheme("superhero"),  
 # Application title  
 titlePanel("Tour de DC"),  
   
 sidebarLayout(  
   
 # Sidebar with a slider input  
 sidebarPanel(  
 textInput("current", "Address", placeholder = "Enter start address", width = NULL),#text input  
 numericInput("radius", label = h3("Radius (in miles)"), value = 1),  
 radioButtons("poi", label = h3("Places of interest"),  
 choices = list("Food" = 1, "Shopping" = 2, "Museums" = 3, "Leisure" = 4),  
 selected = NULL),  
   
   
 uiOutput("choose"),  
 actionButton("button", "Go"),  
 actionButton("button2", "Submit choice")  
 ),  
   
 # Show a plot of the generated distribution  
 mainPanel(  
 tabsetPanel(  
 tabPanel("Starting Point",plotlyOutput("distPlot"), h1("Introduction"), "Thank you for coming to see Tour de DC. This App was built durring HippoHacks 2019 at GWU. Our goal   
 was to make it easy for people to see things to do around the Capital Bikeshare system. You can get started by  
 putting any address in the box below, then selecting a area of interest and we will generate a list of close place of interests  
 and the closest bikeshare to you. Select something you like and check the second panel for a route to a bikeshare close  
 to that place of interest. Please enjoy playing around with the map as it is interactive and has some nice hover info"),  
 tabPanel("End Point", plotlyOutput("endGraph")),  
 tabPanel("Route", "Currently in development, check back later!")  
 )  
 )  
 )  
)  
# generate server functions  
server <- function(input, output) {  
 shinyjs::hide("button2") #hide submit button  
 observeEvent(input$button, {  
 #get initial data whenever button is pressed  
 station\_DataDF <- get\_Capital("https://gbfs.capitalbikeshare.com/gbfs/en/station\_information.json")  
 station\_statusDF <- get\_Capital("https://gbfs.capitalbikeshare.com/gbfs/en/station\_status.json")  
 station\_BigData <- station\_DataDF %>%  
 left\_join(station\_statusDF, by = "station\_id")  
 #Define for closest Location  
 get\_closest <- function(location){  
 location <- geocode(location, output = c("latlon"), source = "google")  
 station\_BigData <- station\_BigData %>% filter(num\_bikes\_available > 0)  
 set1sp <- SpatialPoints(location) #define location points  
 station\_sub <- station\_BigData[, 6:5]#subset for lon lat  
 station\_subSP <- SpatialPoints(station\_sub) #match it back to a spatial data frame  
 blep <- gDistance(set1sp, station\_subSP, byid = TRUE)  
 n <- station\_BigData %>% #cbind back into larger data frame  
 cbind(blep)  
 dock <- n %>%  
 arrange(`1`) %>%  
 head(1L) %>%  
 select(name, lat, lon)  
 return(dock)  
 }  
 #route to the bike rqack  
 current\_loc <- get\_closest(input$current)  
 revised\_loc <- revgeocode(c(current\_loc$lon, current\_loc$lat))  
 walking <- route(from = input$current, to = revised\_loc, mode = "walking",structure = "route", output = "simple")  
 output$distPlot <- renderPlotly({  
 plot\_mapbox(mode = "scattermapbox" #hoverlabel = list(  
 #bgcolor = "green", bordercolor = "white"  
 ) %>%#)   
 add\_markers(data = station\_BigData, x = ~ lon, y = ~ lat, fill = "blue",  
 hoverinfo = "text",  
 text = ~paste('Name: ', name,  
 '\n Available Bikes: ', num\_bikes\_available,  
 '\n Available Docks: ', num\_docks\_available)) %>%  
 layout(mapbox = list(zoom = 15,  
 center = list(lat = ~(get\_closest(input$current)$lat),  
 lon = ~(get\_closest(input$current)$lon)),  
 style = "streets"),  
 showlegend = FALSE) %>%  
 add\_text(data = get\_closest(input$current), x = ~ lon, y = ~ lat+.0004, text = "Our Suggestion!") %>%  
 add\_paths(data = walking, x = ~lon, y = ~lat, size = I(3), alpha = 0.7,  
 hoverinfo = "text", text = ~paste("Take this path to the dock!")) %>%  
 add\_markers(data = geocode(input$current, output = c("latlon"), source = "google"), x = ~lon, y = ~lat,  
 hoverinfo = "text",  
 text = ~paste("Your Location!"))  
 })  
 #printing out user input  
 #defining type based on POI info from user  
 if (input$poi == 1)  
 {type =c("restaurant")  
 n\_type=1  
 return\_n=c(5)  
 }  
 else if (input$poi == 2)  
 {type =c("shopping\_mall","clothing\_store")  
 n\_type=2  
 return\_n=c(2,3)  
 }  
 else if (input$poi == 3)  
 {type= c("museum")  
 n\_type=1  
 return\_n=c(5)  
 }  
 else if (input$poi == 4)  
 {type= c("zoo","amusement\_park","aquarium","bowling\_alley","movie\_theater")  
 n\_type=5  
 return\_n=c(1,1,1,1,1)  
 }  
  
 #new method of POI   
 ratings <- get\_poiVER2(location = input$current, radius = input$radius, type = type, return\_n = 20)  
   
 output$rating <- renderTable(ratings)  
 list\_of\_places <- head(ratings, 5L) %>% select(Name)  
 list\_of\_places$Name <- as.character(list\_of\_places$Name)  
 list\_of\_places <- list\_of\_places$Name  
   
   
 if (input$poi == 1)  
 {header="Food"  
 }  
 else if (input$poi == 2)  
 {  
 header="Shopping"  
 }  
 else if (input$poi == 3)  
 {header="Museums"  
 }  
 else if (input$poi == 4)  
 {  
 header="Leisure"  
 }  
 output$choose <- renderUI({  
 radioButtons("chosen\_place", label = h3(header),  
 choices = list\_of\_places  
 )  
 })  
 shinyjs::show("button2")   
 })#end of observe for button  
   
 observeEvent(input$button2, {   
   
 #get bikeshare data in here  
 station\_DataDF <- get\_Capital("https://gbfs.capitalbikeshare.com/gbfs/en/station\_information.json")  
 station\_statusDF <- get\_Capital("https://gbfs.capitalbikeshare.com/gbfs/en/station\_status.json")  
 station\_BigData <- station\_DataDF %>%  
 left\_join(station\_statusDF, by = "station\_id")  
 #define function for finding closest location with docks to park  
 get\_closest\_end <- function(location){  
 location <- geocode(location, output = c("latlon"), source = "google")  
 station\_BigData <- station\_BigData %>% filter(num\_docks\_available > 0)  
 set1sp <- SpatialPoints(location) #define location points  
 station\_sub <- station\_BigData[, 6:5]#subset for lon lat  
 station\_subSP <- SpatialPoints(station\_sub) #match it back to a spatial data frame  
 blep <- gDistance(set1sp, station\_subSP, byid = TRUE)  
 n <- station\_BigData %>% #cbind back into larger data frame  
 cbind(blep)  
 dock <- n %>%  
 arrange(`1`) %>%  
 head(1L) %>%  
 select(name, lat, lon)  
 return(dock)  
 }  
 #the normal get closest function  
 get\_closest <- function(location){  
 location <- geocode(location, output = c("latlon"), source = "google")  
 station\_BigData <- station\_BigData %>% filter(num\_bikes\_available > 0)  
 set1sp <- SpatialPoints(location) #define location points  
 station\_sub <- station\_BigData[, 6:5]#subset for lon lat  
 station\_subSP <- SpatialPoints(station\_sub) #match it back to a spatial data frame  
 blep <- gDistance(set1sp, station\_subSP, byid = TRUE)  
 n <- station\_BigData %>% #cbind back into larger data frame  
 cbind(blep)  
 dock <- n %>%  
 arrange(`1`) %>%  
 head(1L) %>%  
 select(name, lat, lon)  
 return(dock)  
 }  
 if (input$poi == 1)  
 {type =c("restaurant")  
 n\_type=1  
 return\_n=c(5)  
 }  
 else if (input$poi == 2)  
 {type =c("shopping\_mall","clothing\_store")  
 n\_type=2  
 return\_n=c(2,3)  
 }  
 else if (input$poi == 3)  
 {type= c("museum")  
 n\_type=1  
 return\_n=c(5)  
 }  
 else if (input$poi == 4)  
 {type= c("zoo","amusement\_park","aquarium","bowling\_alley","movie\_theater")  
 n\_type=5  
 return\_n=c(1,1,1,1,1)  
 }  
   
 #new method of POI   
 ratings <- get\_poiVER2(location = input$current, radius = input$radius, type = type, return\_n = 20)  
 #update location details  
 current\_loc <- get\_closest(input$current)  
 revised\_loc <- revgeocode(c(current\_loc$lon, current\_loc$lat))  
 #end bike location + reverse geocode  
 needed <- get\_closest\_end(input$chosen\_place)  
 needed\_rev <- revgeocode(c(needed$lon, needed$lat))  
 #calculate route from one dock to another dock  
 biking <- route(from = revised\_loc, to = needed\_rev, mode = "bicycling", structure = "route", output = "simple")  
 #output graph  
 output$choice <- renderText(input$chosen\_place)  
 output$endGraph <- renderPlotly({  
 plot\_mapbox(mode = "scattermapbox"#, hoverlabel = list(  
 #bgcolor = "green", bordercolor = "white"  
 ) %>% #)  
 add\_markers(data = station\_BigData, x = ~ lon, y = ~ lat, fill = "blue",  
 hoverinfo = "text",  
 text = ~paste('Name: ', name,  
 '\n Available Bikes: ', num\_bikes\_available,  
 '\n Available Docks: ', num\_docks\_available)) %>%  
 layout(mapbox = list(zoom = 14,  
 center = list(lat = ~(needed$lat),  
 lon = ~(needed$lon)),  
 style = "streets"),  
 showlegend = FALSE) %>%  
 add\_text(data = needed, x = ~ lon, y = ~ lat+.0006, text = "Our Suggestion!") %>%  
 add\_paths(data = biking, x = ~lon, y = ~lat, size = I(3), alpha = 0.7,  
 hoverinfo = "text", text = ~paste("Take this path to the dock!"))  
 })  
 })  
   
}   
# Run the application   
if (interactive()){shinyApp(ui = ui, server = server)}