app.R

bingo

2019-04-28

```
library(shiny)
library(shinyjs)
```

```
##
## Attaching package: 'shinyjs'
```

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

```
## 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
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

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

```r
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
```

```r
library(ggplot2)
library(sp)
library(rgeos) #for the revised google places functions
```

```
## rgeos version: 0.4-2, (SVN revision 581)
##  GEOS runtime version: 3.6.1-CAPI-1.10.1
##  Linking to sp version: 1.3-1
##  Polygon checking: TRUE
```

```r
#register google API key
register_google(key="AIzaSyA-YsYmgsuegnG9ZWonhssUq-aQdBr8MHo")
#MAPBOX token
Sys.setenv('MAPBOX_TOKEN' = 'pk.eyJ1IjoibWFzdGVyYmluZ28xIiwiYSI6ImNqdDluOHo2aDAxenQ0OW51dmdkOGIyaDkifQ.
#revised places function
get_poiVER2 <- function(location, radius, type, return_n) {
  key <- "AIzaSyDitfa2CtI_rpIbpJviZRey63DOm7N3ZMA" #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$loca
  results2 <- as.data.frame(results) #turn into dataframe
  colnames(results2) <- c("Name", "Rating", "Latitude", "Longitude") #clean names
  results3 <- results2 %>%
    head(return_n)
  return(results3)
}
#used to pull capital bikeshare data
get_Capital <- function(url) {
  doot <- fromJSON(url)
  doot2 <- doot$data
```

```
  doot3 <- do.call(what = "rbind",
                   args = lapply(doot2, as.data.frame))
  return(doot3)
}
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 get closest for mapping
get_closest <- function(location){
  location <- geocode(location, output = c("latlon"), source = "google")
  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)
}
# Define UI for application that draws a histogram

ui <-  fluidPage(
  shinyjs::useShinyjs(),

  # Application title
  titlePanel("Hippo Hacks"),

  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"), tableOutput("rating")),
        tabPanel("End Point", plotlyOutput("endGraph")),
        tabPanel("Route")
```

```
      )
    )
  )
)
# generate server functions
server <- function(input, output) {
  shinyjs::hide("button2")
  observeEvent(input$button, {

    output$distPlot <- renderPlotly({
      plot_mapbox(mode = "scattermapbox", hoverlabel = list(
        bgcolor = "green", bordercolor = "white"
      )) %>%
        add_markers(data = station_BigData, x = ~ lon, y = ~ lat, fill = "blue",
                    hoverinfo = "text",
                    text = ~paste('Name: ', name,
                                  '\n Available Bikes: ', num_bikes_available,
                                  '\n Available Docks: ', num_docks_available)) %>%
        layout(mapbox = list(zoom = 14,
                             center = list(lat = ~(get_closest(input$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!
    })
    #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, {

    output$choice <- renderText(input$chosen_place)
    output$endGraph <- renderPlotly({
      plot_mapbox(mode = "scattermapbox", hoverlabel = list(
        bgcolor = "green", bordercolor = "white"
      )) %>%
        add_markers(data = station_BigData, x = ~ lon, y = ~ lat, fill = "blue",
                    hoverinfo = "text",
                    text = ~paste('Name: ', name,
                                  '\n Available Bikes: ', num_bikes_available,
                                  '\n Available Docks: ', num_docks_available)) %>%
        layout(mapbox = list(zoom = 14,
                             center = list(lat = ~(get_closest(input$chosen_place)$lat),
                                           lon = ~(get_closest(input$chosen_place)$lon)),
                             style = "streets"),
               showlegend = FALSE) %>%
        add_text(data = get_closest(input$chosen_place), x = ~ lon, y = ~ lat+.0006, text = "Our Sugges
    })
  })

}
# Run the application
if (interactive()){shinyApp(ui = ui, server = server)}
```