````
---
title: "R Notebook"
output:
  html_document:
    df_print: paged
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you
execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk
or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```{r}
setwd("~/Documents/Programming/Mother dearest/Postal Code Project")
library(tidyverse)
library(plotly)
library(reticulate)

use_python("/Users/charlengels/anaconda3/bin/python", required = T)


# Check the version of Python.
py_config()

import("googlemaps")
source_python("geocoding.py")

library("ggmap")
ggmap::register_google(key = "AIzaSyCf_dIiMGEZEtw1pjg325Sr8X2Vdg9QDEo")

library(ggplot2)
```

```{r}
members <- read_csv(file = "./new data/address_tf.csv", col_names =
TRUE)
```
```{r}
for (member in members) {

}
# for (state in unique(test$province)) {
#   df_sapo_plot <- test %>%
#     filter(province == state)
#   p <- ggmap(get_googlemap(center = paste(state, " ,South Africa", sep
= ""), size = c(640, 640), zoom = 6, scale = 2, maptype = "roadmap",
color = "color", region = "za"))
#
#   street_plot <- p + geom_point(data = df_sapo_plot , aes(x = Long, y
````
````

```
= Lat, color = province, alpha = 0.5, text = paste(
#     "City / Town: ", suburb, "\n",
#     "Postal Code: ", postcode, "\n",
#     "Province: ", province, "\n",
#     sep = "")))
#   sapo_street_plot <- ggplotly(street_plot, tooltip = "text")
#   htmlwidgets::saveWidget(as_widget(sapo_street_plot),
paste("postal_codes_",state,".html"))
# }
p <- ggmap(get_googlemap(center = "Western Cape, South Africa", size =
c(640, 640), zoom = 6, scale = 2, maptype = "roadmap", color = "color",
region = "za"))
member_plot <- p + geom_point(data = members, aes(x = GeoX, y = GeoX,
color = Province, alpha = 0.5,
                                                  text = paste(
                                                    "Town: ", Town,
"\n",
                                                    "Postal Code: ",
GeoPostalCode, "\n",
                                                    "Province: ",
Province, "\n",
                                                    sep = "")))


member_plotly <- ggplotly(member_plot, tooltip = "text")
htmlwidgets::saveWidget(as_widget(member_plotly),
paste("member_plots.html"))
```


Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

```{r}
luds_file <-read_csv("./new data/Luds_gis.csv")
member_gis <- read_csv("./new data/address_tf.csv", col_types =
"dcccccdcdd")
final_check <- read_csv("./new data/final_checking_rep_has_header.csv")

doctor_file_headers <- c("doctor_no", "scheme", "add1", "add2", "add3",
"town", "postal", "geo_lat", "geo_long", "province")
doctor_file <- read_csv("./new data/address_doctor_nom_tf.csv",
col_names = doctor_file_headers)
```

```
library(tidyverse)
districts <- final_check %>%
  select(Pcode, dis_code)

member_gis <- member_gis %>%
  left_join(districts, by = c("PostalCode" = "Pcode"))

```


```{r}
library(reticulate)
use_python("/Users/charlengels/anaconda3/bin/python", required = T)


# Check the version of Python.
py_config()

import("googlemaps")
source_python("geocoding.py")
```

```{r}
#new geo locations from address

for (i in 1:nrow(member_gis)) {
  lookup <-  paste(member_gis[i,"Line1"], member_gis[i,"line2"],
member_gis[i, "line3"], member_gis[i, "Town"],"South Africa", sep = ",
")


  try (
    expr = {
      co_ords <- get_geocode(lookup)
      member_gis[i, "NewLat"] <- co_ords[[1]]
      member_gis[i, "NewLon"] <- co_ords[[2]]
      print(paste(i, lookup,"Lat", co_ords[[1]],"Long", co_ords[[2]],
sep = " -> "))
      }

  )

### Output below
}
write_csv(member_gis, "./new data/updated_member.csv")
```

```{r}
# dlon = lon2 - lon1
# dlat = lat2 - lat1
# a = (sin(dlat/2))^2 + cos(lat1) * cos(lat2) * (sin(dlon/2))^2
```

```
# c = 2 * atan2( sqrt(a), sqrt(1-a) )
# d = R * c (where R is the radius of the Earth)

#write_csv(member_gis, "./new data/updated_member.csv")

member_gis <- read_csv("./new data/updated_member.csv")

#find missing long geocodes
fix_missing_geo <- member_gis %>%
  filter(is.na(member_gis$NewLat))
write_csv(fix_missing_geo, "./new data/missing geo codes.csv")

for (i in 1:nrow(member_gis)) {
  if (is.na(member_gis[i,"NewLat"])) {
    member_gis[i,"NewLat"] <- member_gis[i,"GeoX"]
    member_gis[i,"NewLon"] <- member_gis[i,"GeoY"]
    member_gis[i,"Err"] <- "Y"
    member_gis[i,"Err_reason"] <- "no google lookup"
  }
  if (member_gis[i,"NewLat"] >-21) {
    member_gis[i,"NewLat"] <- member_gis[i,"GeoX"]
    member_gis[i,"NewLon"] <- member_gis[i,"GeoY"]
    member_gis[i,"Err"] <- "Y"
    member_gis[i,"Err_reason"] <- "Bad Lat"
  }
  if (member_gis[i,"NewLon"] < 15) {
    member_gis[i,"NewLat"] <- member_gis[i,"GeoX"]
    member_gis[i,"NewLon"] <- member_gis[i,"GeoY"]
    member_gis[i,"Err"] <- "Y"
    member_gis[i,"Err_reason"] <- "Bad Lon"
  }
  if (member_gis[i,"NewLon"] > 34) {
    member_gis[i,"NewLat"] <- member_gis[i,"GeoX"]
    member_gis[i,"NewLon"] <- member_gis[i,"GeoY"]
    member_gis[i,"Err"] <- "Y"
    member_gis[i,"Err_reason"] <- "Bad Lon"
  }

}

summary(member_gis)

erroneous <- member_gis %>%
  filter(Err == "Y" )

hist(erroneous$NewLon)
summary(erroneous)

write_csv(member_gis, "./new data/updated_member_geo_mod.csv")
```
```{r}
```

```
gcd.hf <- function(long1, lat1, long2, lat2) {
  R <- 6371 # Earth mean radius [km]
  delta.long <- (long2 - long1)
  delta.lat <- (lat2 - lat1)
  a <- sin(delta.lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta.long/2)^2
  c <- 2 * asin(min(1,sqrt(a)))
  d = R * c
  return(d) # Distance in km
}

for (i in 1:nrow(member_geo_dist)) {
  member_geo_dist[i, "err_dist"] <- gcd.hf(member_geo_dist[i, "GeoY"],
member_geo_dist[i, "GeoX"],member_geo_dist[i, "NewLon"],
member_geo_dist[i, "NewLat"])
}

# member_geo_dist <- member_gis %>%
#   mutate(err_dist = gcd.hf(GeoY, GeoX,NewLon, NewLat))

# -33.91
# 18.50
# CPT
# -33.91156
# 18.513970

print(gcd.hf(18.50, -33.91, 18.513970, -33.91156))

```


```{r}
# -33.61223
# 26.88717
j <- ggmap(get_googlemap(center = c(mean(luds_file$y_dd, na.rm = TRUE),
mean(luds_file$x_dd, na.rm = TRUE)), zoom = 9, size = c(640, 640), scale
= 2, format = "png8", maptype = "hybrid", color = "color"), extent =
"device")

mem_lon <- mean(member_gis$GeoY, na.rm = TRUE)
mem_lat <- mean(member_gis$GeoX, na.rm = TRUE)
mem <- ggmap(get_googlemap(center = c(mem_lon, mem_lat), zoom = 10, size
= c(640, 640), scale = 2, format = "png8", maptype = "roadmap", color =
"color"), extent = "device")

mem_plot <- mem + geom_point(data = member_gis, aes( x = GeoY, y = GeoX,
alpha = 0.1), color = "blue")

lud_plot <- j + geom_point(data = luds_file, aes(x = y_dd, y = x_dd,
alpha = 0.5), color = "red")
```

```
lud_plotly <- ggplotly(lud_plot)

mem_plotly <- ggplotly(mem_plot)


htmlwidgets::saveWidget(as_widget(lud_plotly), paste("lud_plot.html"))
htmlwidgets::saveWidget(as_widget(mem_plotly), paste("mem_plot.html"))

```


```{r}
#### Plotting the various provincial maps
for (state in unique(member_gis$Province)) {
  member_gis_plot <- member_gis %>%
    filter(Province == state)
  doctor_file_temp <- doctor_file %>%
    filter(province == state)
  blank <- ggmap(get_googlemap(center = paste("Cape Town", state, "South
Africa", sep = ", "),zoom = 11, size = c(640, 640), scale = 2, format =
"png8", maptype = "roadmap", color = "bw"), extent = "device")
  temp_plot <- blank + geom_point(data = member_gis_plot, aes(x = GeoY,
y = GeoX), colour = "purple") + geom_point(data = member_gis_plot, aes(x
= NewLon, y = NewLat, alpha = 0.25, colour = dis_code)) +
    geom_point(data = doctor_file_temp, aes(x = geo_long, y = geo_lat,
shape = scheme)) +
    geom_point(data = new_hospital, aes(x = x_dd, y = y_dd, shape =
type, alpha = 0.25), colour = "yellow")


  temp_plotly <- ggplotly(temp_plot)
  htmlwidgets::saveWidget(as_widget(temp_plotly), paste(state, ".html",
sep = ""))
}
# getwd()

```

```{r}
new_hospital <- read_csv("./new data/hospitals.csv")
```