Homework 6

Ctrl Alt Elite: Morris Greenberg, Dror Shvadron, Eduardo Schiappa Pietra, Tao Ni

11/05/2019

Setup

```
library(tidyverse)
library(parallel)
library(httr)
library(rjson)
library(knitr)
knitr::opts_chunk$set(echo = TRUE, comment = NA, message = FALSE,
                      warning = FALSE, cache = TRUE, cache.lazy = FALSE)

#devtools::install_github("hrbrmstr/darksky")
library(darksky)
```

Data Collection

Download train data

Here, parallel computing is used to download the files more efficiently.

```
link_base <- "http://www2.stat.duke.edu/~sms185/data/bike/cbs_"
cl <- makeCluster(4, 'PSOCK')
clusterExport(cl, varlist=c("link_base"))
clusterEvalQ(cl, library(tidyverse))
```

```
[[1]]
 [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
 [6] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
[11] "graphics"  "grDevices" "utils"     "datasets"  "methods"
[16] "base"

[[2]]
 [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
 [6] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
[11] "graphics"  "grDevices" "utils"     "datasets"  "methods"
[16] "base"

[[3]]
 [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
 [6] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
[11] "graphics"  "grDevices" "utils"     "datasets"  "methods"
[16] "base"

[[4]]
 [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
 [6] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
[11] "graphics"  "grDevices" "utils"     "datasets"  "methods"
```

```
[16] "base"
```

```
bike_datasets <- parLapply(cl, 2013:2017, function(i){
  dataset <- read_csv(paste0(link_base, i, ".csv"))
  Sys.sleep(abs(rnorm(1)))
  return(dataset)
})
stopCluster(cl)
bike_df <- bind_rows(bike_datasets)
```

Download Station data

```
lat_lon_url <- "https://layer.bicyclesharing.net/map/v1/wdc/map-inventory"
inventory_page <- GET(lat_lon_url)
json <- fromJSON(content(inventory_page, type="text",
                         encoding = "ISO-8859-1"))
geometry <- map(json$features, `[[`, "geometry")
coords <- map(geometry, `[[`, "coordinates")
coords_df <- data.frame(lon = sapply(coords, `[`, 1),
                        lat = sapply(coords, `[`, 2))
properties <- map(json$features, `[[`, "properties")
props_df <- map_df(map(properties, `[[`, "station"), `[`,
                   c("id", "name", "terminal", "installed",
                     "capacity", "renting", "returning"))
inventory_df <- bind_cols(coords_df, props_df) %>%
  mutate(terminal = as.numeric(terminal))
inventory_df$id <- as.integer(inventory_df$id)
saveRDS(inventory_df, file = "data/inventory.Rds")
```

Prep train data

```
bike_df2 <- left_join(bike_df, inventory_df,
                      by=c("Start station number"="terminal"))

bike_df2 <- bike_df2 %>%
  left_join(inventory_df, by=c("End station number"="terminal"),
            suffix=c("_start", "_end"))
```

Save train data

```
dir.create(file.path("data"), showWarnings = FALSE)
saveRDS(bike_df2, file = "data/bike_df2.Rds")
```

Download test data

```
dir.create(file.path("data"), showWarnings = FALSE)
download.file("http://www2.stat.duke.edu/~sms185/data/bike/cbs_test.csv",
              destfile = "data/cbs_test.csv")
test_df <- read_csv("data/cbs_test.csv")
```

Weather data

Weather data is downloaded using the Dark Sky R wrapper package. Once downloaded using the API, it is saved on our local storage.

```
#devtools::install_github("hrbrmstr/darksky")
library(darksky)

DC_lat <- 38.892059 #https://www.latlong.net/
DC_lon <- -77.019913

# historic data NOT RUN
if (FALSE) {
  weather_dates <- seq(as.Date("2013-01-01"), as.Date("2019-01-01"), by="days")

  weather1 <- weather_dates[1:1000]
  weather1_data <- map(.x = weather1,
                       .f = function(x) get_forecast_for(latitude = DC_lat, longitude = DC_lon, x))
  saveRDS(weather1_data, file = "data/weather1.Rds")

  darksky_api_key(force = T)
  weather2 <- weather_dates[1001:2000]
  weather2_data <- map(.x = weather2,
                       .f = function(x) get_forecast_for(latitude = DC_lat, longitude = DC_lon, x))
  saveRDS(weather2_data, file = "data/weather2.Rds")

  darksky_api_key(force = T)
  weather3 <- weather_dates[2001:2192]
  weather3_data <- map(.x = weather3,
                       .f = function(x) get_forecast_for(latitude = DC_lat, longitude = DC_lon, x))
  saveRDS(weather3_data, file = "data/weather3.Rds")

  weather_data <- c(weather1_data, weather2_data, weather3_data)
  saveRDS(weather_data, file = "github_data/weather_data.Rds")
}

# turn to dataframe - hourly
weather_data <- readRDS("github_data/weather_data.Rds")
weather_hr_df <- map_df(weather_data, `[[`, "hourly")
weather_day_df <- map_df(weather_data, `[[`, "daily")
weather_cur_df <- map_df(weather_data, `[[`, "currently")
```

Prepare Data

Calculate Shortest distance between Stations

Geographic distance is calculated between any two stations

```
library(geosphere)

dist_df <- crossing(inventory_df %>%
                      select(start_id = id, start_lat = lat, start_lon = lon),
                    inventory_df %>%
                      select(end_id = id, end_lat = lat, end_lon = lon))
```

```r
dist_df <- dist_df %>%
  rowwise %>%
  mutate(dist = distGeo(c(start_lat, start_lon), c(end_lat, end_lon))) #in meters

saveRDS(dist_df, file = "data/dist_df.Rds")
```

Nearest Neighbor Model

A simple model is used to predict the nearest neighbor for each start station and duration. For each pair of start and end points in the train data, we calculate the mean duration. In the test data, we minimize the difference between the given duration and the predicted mean duration. We use this simple model because we lack the time to incorporate more advanced models.

```r
# load data
bike_df2 <- readRDS(file = "data/bike_df2.Rds")
test_df <- read_csv("data/cbs_test.csv")
inventory_df <- readRDS("data/inventory.Rds")
```

```r
# calc avg duration between stations
NN <- bike_df2 %>%
  group_by(start_station_number=`Start station number`,
           end_station_number = `End station number`) %>%
  summarize(mean_duration = mean(Duration)) %>%
  ungroup

# match NN
predicted <- test_df %>% select(start_station, start_station_number, duration) %>%
  mutate(ride_id = row_number()) %>%
  left_join(NN, by = c()) %>%
  mutate(delta = abs(mean_duration- duration))

# filter on smallest delta
predicted <- predicted %>%
  group_by(ride_id) %>%
  filter(delta == min(delta)) %>%
  sample_n(1) # for cases of equals

# finalize
predicted <- predicted %>%
  select(ride_id, start_station, start_station, start_station_number, end_station_number) %>%
  mutate(probability = 1)
```

Predict Test Data

```r
# pivot to long for easier manipulation
test_df2 <- test_df %>% mutate(ride_id = row_number()) %>%
  pivot_longer(-c(ride_id, start_date, end_date, duration,
                  start_station_number, start_station, bike_number, member_type),
               names_to = "end_station_name",
               values_to = "probability") # DONT MESS WITH THE ORDER OF THIS
```

```
test_df2 <- inventory_df %>%
  select(end_station_name = name, end_station_number = terminal) %>%
  right_join(test_df2)

#adjust probability values
test_df2 <- test_df2 %>%
  select(-probability) %>% # delete old probability
  left_join(predicted, by = c("end_station_number", "ride_id")) %>%
  mutate(probability = replace_na(probability, 0))


# pivot back to wide # VERY SLOW FOR SOME ANNOYING REASON
# test_df3 <- test_df2 %>%
#   pivot_wider(names_from = end_station_name,
#               values_from = probability) %>%
#   select(-ride_id)

# alternative
temp <- matrix(test_df2$probability, ncol = 488, byrow = T)
test_df_out <- test_df
test_df_out[,8:495] <- temp
```

Export prediction

```
dir.create(file.path("out"), showWarnings = FALSE)
write_csv(test_df_out, "out/cbs_ctrl-alt-elite.csv")
```

Analysis of prediction

Most popular start-end pairs in the train and test data

```
pop_train <- bike_df2 %>%
  group_by(`Start station`, `End station`) %>%
  summarize(n=n()) %>%
  arrange(-n) %>%
  ungroup %>%
  slice(1:10)

kable(pop_train)
```

Start station	End station	n
Jefferson Dr & 14th St SW	Jefferson Dr & 14th St SW	3286
Lincoln Memorial	Jefferson Memorial	3258
Jefferson Dr & 14th St SW	Lincoln Memorial	3062
Lincoln Memorial	Jefferson Dr & 14th St SW	2785
Smithsonian-National Mall / Jefferson Dr & 12th St SW	Smithsonian-National Mall / Jefferson Dr & 12th St SW	2548
Lincoln Memorial	Lincoln Memorial	2461
Columbus Circle / Union Station	8th & F St NE	2206
Smithsonian-National Mall / Jefferson Dr & 12th St SW	Lincoln Memorial	1931
8th & F St NE	Columbus Circle / Union Station	1886

Start station	End station	n
Eastern Market Metro / Pennsylvania Ave & 7th St SE	Lincoln Park / 13th & East Capitol St NE	1725

```
predicted <- inventory_df %>%
  select(terminal, name) %>%
  right_join(predicted, by = c("terminal" = "end_station_number")) %>%
  rename(end_station_name = name)

pop_test <- predicted %>%
  group_by(start_station, end_station_name) %>%
  summarize(n=n()) %>%
  arrange(-n) %>%
  ungroup %>%
  slice(1:10)

kable(pop_train)
```

Start station	End station	n
Jefferson Dr & 14th St SW	Jefferson Dr & 14th St SW	3286
Lincoln Memorial	Jefferson Memorial	3258
Jefferson Dr & 14th St SW	Lincoln Memorial	3062
Lincoln Memorial	Jefferson Dr & 14th St SW	2785
Smithsonian-National Mall / Jefferson Dr & 12th St SW	Smithsonian-National Mall / Jefferson Dr & 12th St SW	2548
Lincoln Memorial	Lincoln Memorial	2461
Columbus Circle / Union Station	8th & F St NE	2206
Smithsonian-National Mall / Jefferson Dr & 12th St SW	Lincoln Memorial	1931
8th & F St NE	Columbus Circle / Union Station	1886
Eastern Market Metro / Pennsylvania Ave & 7th St SE	Lincoln Park / 13th & East Capitol St NE	1725