Code Appendix - cleaning and index

Mark Brackenrig

29 April 2018

Code Appendix

Index

The index file runs vital loading and cleaning scripts. The purpose of this script is to allow everyone to be working from a consistent code base and data set, while not rewriting data directly.

We decided that unless there was cost requirements or some other restriction (like API request numbers), we would not overwrite data.

```
#Index File

setwd("C:/Users/mbrackenrig/Documents/Assignment-2")

#Pull in data
source("Preprocessing/ReadAirbnb.r")

setwd("C:/Users/mbrackenrig/Documents/Assignment-2")

#Clean data - Put it in correct formate, etc..
source("Preprocessing/Cleaning.r")
```

Read AirBnB file

Read airbnb reads all of the data in our shared drive. It only looks for CSV files currently, but we may extend this to files that are of a different file type.

```
### Ingest Airbnb Dataset ###

setwd("G:/Team Drives/STDS - Assignment 2 - 3MDL/Dataset/AirBnB")

#Read all file names in the AirBnB Dataset file
files <- list.files()

library(readr)

#Run through each file and
for(i in 1:length(files)){
#Read file
  if(grepl(pattern = ".csv",x = files[i])==T){
temp <- read_csv(files[i])
#call the file by the name of the csv
assign(substr(files[i], start = 0, stop = nchar(files[i])-4),temp)
#remove the temp file name
rm(temp)  }
}
```

Cleaning

Finally, the cleaning script aims to clean the data set so that:

- All of the data is in the correct format
- Unnecessary columns are removed from the data set
- Remove records that appear to be scraped incorrectly or appear inconsistent

Note: I have removed much of the actual code, this code is just example snippets of the full script

```
## Snippets only!

#### Price Fields ####
    #clean Price fields
    priceclean <- function(x){
      x <- gsub(pattern = "\\$", replacement = "", x)
      x <- gsub(pattern = "\\,", replacement = "", x)
      x <- as.numeric(x)
    }

    #Check price fields
    #subset(colnames(listings),grepl("price",tolower(colnames(listings)))==T)

    #other price fields clean
    listings$price <- priceclean(listings$price)
    listings$weekly_price <- priceclean(listings$weekly_price)
    listings$monthly_price <- priceclean(listings$monthly_price)

#Other price fields
    listings$security_deposit <- priceclean(listings$security_deposit)
    listings$cleaning_fee <- priceclean(listings$cleaning_fee)
    listings$extra_people <- priceclean(listings$extra_people)


#### Remove URLs ###
listings <- subset(listings, select = c( subset(colnames(listings),grepl("url",tolower(colnam
es(listings)))==F)))


### coerce to correct format ###

listings$host_response_rate <- gsub(pattern = "\\%", replacement = "", listings$host_response
_rate)

#Coerce the text null values to actual null values
#this is cosmetic code, its just so that R doesnt throw a warning
listings$host_response_rate[listings$host_response_rate=="N/A"] <- NA

#Convert to correct format
listings$host_response_rate <- as.numeric(listings$host_response_rate)
listings$host_is_superhost <- as.factor(listings$host_is_superhost)

#### Convert nested string format to tidy format ####

## Amenities

Amenities <- subset(listings, select = c("id", "amenities"))
Amenities$amenities <- gsub(pattern = "\\{", replacement = "",Amenities$amenities)
Amenities$amenities <- gsub(pattern = "\\}", replacement = "",Amenities$amenities)
Amenities$amenities <- gsub(pattern = '\"',fixed = T, replacement = "",Amenities$amenities)

split <- strsplit(Amenities$amenities,split = ",")
split <- unlist(split)
split <- unique(split)
```

```
for(i in 1:length(split)){
  Amenities$temp <- grepl(pattern = split[i] ,x = Amenities$amenities)
  colnames(Amenities)[colnames(Amenities)=="temp"] <- paste0("Amenities_",split[i])
  temp <- NULL
}
Amenities$amenities <- NULL

keep <- c()
for(i in 2:ncol(Amenities)){
  #Choose 650, i.e. at least approximately 2% of properties should have this feature
  if(sum(Amenities[,i]==T)>650){
keep <- c(keep, i)

  }
}

#this vector is an index vector of all of the columns I suggest we include in the analysis fo
r amenities
keep <- c(1,keep)
```

Google Maps Distance Matrix Data Collection

This script was used to collect data from the google maps distance matrix. Currently we have only run this script for the sydney opera house, but will expand this to other areas.

```
#install.packages("gmapsdistance")

library(gmapsdistance)

#The gmaps function is as follows
#gmapsdistance(origin, destination, combinations, mode, key,
#shape, avoid, departure, dep_date, dep_time,
#traffic_model, arrival, arr_date, arr_time)

#Register do parallel so it is faster
library(doParallel)

registerDoParallel(cores = detectCores()-1)

#get the unique listing (note they are all unique)
Timedata1 <- unique(subset(listings, select = c("latitude", "longitude","id")))


#Test
#gmapsdistance(origin =paste0("-33.8","+,+","151"),destination = 'Sydney+Opera+House', depart
ure = as.numeric(as.POSIXct("2018-04-21 12:00:00")),combinations = "all",mode = "transit", sh
ape = "wide")



####Public Transport to Opera House####
Opera_Public_Data <- foreach(i=1:nrow(Timedata)) %dopar% {
  library(gmapsdistance)
  #Dont steal my API
  set.api.key('YOU MUST SEt THE API KEY!!!')

  OperaTimePublic <- gmapsdistance(origin =paste0(Timedata$latitude[i],"+",Timedata$longitude
[i]),destination = 'Sydney+Opera+House', departure = as.numeric(as.POSIXct("2018-04-21 12:00:
00")),combinations = "all",mode = "transit", shape = "wide")

}

#Write to DF
Opera_Public <- as.data.frame(matrix(unlist(Opera_Public_Data), ncol = 3,byrow = TRUE))
Opera_Public <- cbind(Opera_Public, listings$id)

#Rename for Cleaning
colnames(Opera_Public) <- c("Time", "Distance", "Status", "id")

#Change columns to correct format
#Time in Seconds
Opera_Public$Time <- as.numeric(as.character(Opera_Public$Time))

#Distance in Meters
Opera_Public$Distance <- as.numeric(as.character(Opera_Public$Distance))


write_csv(Opera_Public, "G:/Team Drives/STDS - Assignment 2 - 3MDL/Dataset/AirBnB/Opera House
by Public Transport.csv")
```