Developing A Tool to Uncover The Mysterious New York City Through Taxi Trip Records

A Thesis

Presented to

The Division of

Smith College

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Arts

Wencong (Priscilla) Li

May 2018

Approved for the Division

(Statistical and Data Sciences)

Benjamin Baumer

Jordan Crouser

Acknowledgements

Acknowledgements

I would love to thank my thesis advisor Benjamin Baumer, Assistant Professor of Statistical and Data Sciences at Smith College, for encouraging me to chanllenge myself and and guiding me through this project. I also want to thank all my friedns and my roommates for their love and support.

Table of Contents

List of Tables

List of Figures

Abstract

Abstract

The New York City Taxi Cabs are widely recognized as the icons of New York City. The New York City Taxi & Limousine Commission provide publicly accessible yellow and green taxi trip records for people to do research with. Each taxi trip record is like a little piece of a gigantic puzzle, and all together they draw a picture of what is happening in New York City every day. This thesis presents a more efficient and easy-to-use way for users to retrieve information of both New York City taxi trip record and trip records of other ridesharing services in New York City, such as Uber and Lyft. By focusing on New York City's iconic yellow taxi's trip records, we investigate social and taxi pricing questions. Additionally, this thesis illustrates a way for taxi drivers to better understand where the customers are and where the customers try to go.

Chapter 1

Introduction

1.1 Motivation

Working with medium data in R is not an easy task. Loading medium-sized data into R environment takes a long time and might crush an R session. Creating a user-friendly platform that allows R users to easily work with medium data is my motivation. There are a lot of interesting data that are needed to be explored. In my study, I focus on New York City taxicab data, because there is so much that could be learned from taxicab trip records.

New York City taxi drivers, passengers, and NYC Taxi & Limousine Commission are the three parties who are closely involved in the NYC taxi industry. Each party has its own needs. Better understanding the needs of the three parties and provide solutions to better satisfy their needs are what I am hooping to be the result of this thiesis.

1.2 Background

1.2.1 Yellow Taxi

The Yellow Cabs are widely recognized as the icons of New York City. NYC Taxicabs are operated by private firms and licensed by the New York City Taxi and Limousine Commission (TLC). TLC issues medallions to taxicabs, and every taxicab must have a medallion to operate. There were 13,437 yellow medallion taxicabs licenses in 2014, and taxi patronage has declined since 2011 because of the competition caused by rideshare services.

1.2.2 Green Taxi

The apple green taxicabs in New York City are called Boro taxis and they are allowed to only pick up passengers in outer boroughs and in Manhattan above East 96th and West 110th Streets. Historically, only the yellow medallion taxicabs were allowed to pick up passengers on the street. However, since 95% of yellow taxi pick-ups occurred in Manhattan to the South of 96th Street and at the two airports, Five Borough Taxi Plan was started to allow green taxis to fill in the gap in outer boroughs.

1.2.3 Uber

Uber Technologies Inc., famously known as Uber, is an American technology company that operates private cars worldwide. Uber drivers use their own cars, instead of corporate-owned vehicles, to drive with Uber. In NYC, Uber uses ‘upfront pricing”, meaning that riders are informed about the fares that they will pay before requesting a ride, and gratuity is not required. Riders are given the opportunity to compare different transportation fares before making their decisions on which one to choose.

Uber NYC was launched in May 2011, and it only took 5 years to have its growth to plateau.

1.2.4 Lyft

Similar to Uber, Lyft is also an on-demand transportation company, and it operates the Lyft car transportation mobile app. Lyft is the main competitor of Uber, and it came into market in July 2014 in New York City.

1.3 Literature Review

1.3.1 New York City Taxi

https://www.reuters.com/article/us-driving-roadrage-life/new-york-drivers-named-most-aggressive-angry-in-u-s-idUSTRE55F1J720090616

https://ny.curbed.com/2017/1/17/14296892/yellow-taxi-nyc-uber-lyft-via-numbers

https://nypost.com/2016/12/02/new-york-citys-traffic-is-intentionally-horrible/

http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/

1.3.2 Competition between New York City taxi and e-hail services

http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/

https://www.forbes.com/sites/federicoguerrini/2015/04/09/living-in-new-york-this-app-will-tell-you-which-is-cheaper-uber-or-a-taxi/2/#26bc29904023

https://www.openstreetcab.com

https://ny.curbed.com/2017/1/17/14296892/yellow-taxi-nyc-uber-lyft-via-numbers

https://nycdatascience.com/blog/student-works/finding-fare-uber-recommendation-system/

1.4 Contribution

1.4.1 nyctaxi Package

nyctaxi is an etl-dependent R package that help users to easily get access to New York City Taxi, Uber and Lyft trip data through Extract, Transform, and Load functions (ETL). This package facilitates ETL to deal with medium data that are too big to store on a laptop. Users are given the option to choose specific years and months as the input parameters of the three ETL functions, and a populated SQL database will be returned as the output. Users do not need to learn SQL queries, since all user interaction is in R.

1.4.2 Social Impact of NYC Taxi

New York City taxi drivers, passengers, and NYC Taxi & Limousine Commission are the three parties who are closely involved in the NYC taxi industry. Each party has its own needs: taxi drivers want to maxmize their profit, and in order to do that, they need to maximize the revenue while minimizing the cost. Taxi passengers want the cheapest and most convenient way of transportantion. Since Uber and Lyft launched their services in New York City, many consumers started to demand the cheaper e-hail services. TLC wants to protect both taxi drivers and passengers, and it creates policies to make NYC taxi more accessible to consumers who really need this service. In this section, I think about what each party wants and try to find a way for them to be better-off.

1.4.3 Reproducible Research

Reproducible research and open sources are the very first things that Ben mentioned to me in the beginning of my honors project. As scholars place more emphasis on the reproducibility of research studies, it is essential for me to make my dat and code openly available for people to eith redo my analysis or test my result.

`Knitr` and Github are used in my project to make my study reproducible, ranging from the initial source to raw data to the package I wrote to utlize the raw data to the statistical data analysis. I used an Github Ripository called `thesisdown` to layout the basic structure of my paper, this tool allows students to create reproducible and dynamic techinical report in R Markdown. It also allows users to embed R code and interactive applicationis, and output into PDF, Word, ePub, or gitbook doocuments. `thesisdown` helps users to efficiently put together any paper with similar format.

Github is used to store the scripts for `nyctaxi` and this thesis. `nyctaxi` is available on CRAN for people to download and install, and the source code for data analysis in this thesis is available under the Github account of the author so that scholars can easyil access the information that thery are interested in. In terms of tables, figures, and anything included in the Appendix attached to this thesis, scripts that are used to generate them are included in the Github repository.

Chapter 2

Data and nyctaxi Package

2.1 Data and Storage

The `nyctaxi` R package allows users to download, clean, and load data into SQL databasses. There are four types of data that are available for users to get access to, and they are New York City yellow taxi trip data, NYC green taxi data, NYC uber trip data, and NYC lyft data.

2.1.1 Yellow Taxi

The total size of all yellow taxi trip data `csv` files (from Jan 2010 to Dec 2016) is 191.38 GB, and NYC yellow taxi trip data from Jan 2009 to the most recent month can be found on NYC Taxi & Limousine Commission (TLC). The data were collected and provided to the NYC TLC by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

The yellow taxi trip records include the following fields: pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

2.1.2 Green Taxi

The total size of green taxi trip data `csv` files (from Aug 2013 to Dec 2016) is 7.8 GB, and green taxi trip data from Aug 2013 to the most recent month can be downloaded from NYC Taxi & Limousine Commission (TLC). The data were collected and provided to the NYC TLC by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

The green taxi trip records include the following fields: pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

2.1.3 Uber

The total size of Uber pick-up data (over 4.5 million from Apr to Sep 2014 and 14.3 million from Jan to June 2015) is 4.3 MB, and thanks to FiveThirtyEight who obtained the data from NYC TLC by submitting a Freedom of Information Law request on July 20, 2015, these data are now open to public.

The 2014 Uber data contains four variables: Data/Time (the date and time of the Uber pick-up), Lat (the latitude of the Uber pick-up), Lon (the longitude of the Uber pick-up), and Base (the TLC base company code affiliated with the Uber pickup).

The 2015 Uber data contains four variables: Dispatching_base_num (the TLC base company code of the base that dispatched the Uber), Pickup_date (the date of the Uber pick-up), Affiliated_base_num (the TLC base company code affiliated with the Uber pickup), and locationID (the pick-up location ID affiliated with the Uber pickup).

2.1.4 Lyft

The total size of weely-aggregated Lyft trip data (from Jan 2015 to Dec 2016) is 914.9 MB, and these data are open to public and weekly-aggregated Lyft data from 2015 to the most recent week can be found on NYC OpenData website.

2.1.5 Data Storage

The total size of all `csv` files of the four services is about 200 GB, and a laptop usually has memory less than or equal to 8GB. Limited memory constrains the amount of data that can be loaded by a personal computer. When users load data into R environment, R keeps them in memory; when the amount of data loaded into R environment gets close to the limit of a computer's memory, R becomes unresponsive or force quit the current session. Therefore, better ways to work with data that takes more space than 8 GB is needed. According to Weijia Zhang (2016), comparing to RAM, hard disk is often used to store medium-sized data, because it is affordable and are designed for storing large items permanently. However, retrieving data from hard drives usually takes about 1,000,000 times more time.

2.2 ETL nyctaxi Package

etl is the parent package of nyctaxi. etl package provides a CRAN-friendly framework that allows R users to work with medium data without any knowledge in SQL database. The end result is a populated SQL database, but the user interaction takes place solely within R. It has three operations -extract, transfer, and load- which bring real-time data into local or remote databases. etl-dependent packages make medium data - too big to store in memory on a laptop- more accessible to a wider audience. Additionally, etl-dependent packages use SQL translation supported by dyplr.

nyctaxi was initially designed to work with New York City taxi data, but later on Uber and Lyft data were added and the ETL functions are modified to be specialized in working with these data. This package compiled three major sources of hail service in New York City so that it is convenient for users to compare and contrast the performance of these three services.

This package inherits functions from many packages: etl, dplyr, DBI, rlang, and stringr.

Since SQL databases are good tools for medium data analysis, ETL functions build connection to a SQL database at the back end and convert R code automatically into SQL queries and send them to the SQL database to get data tables containing data of each hail service. Thus, users do not need to have any knowledge of SQL queries and they can draw in any subsets of the data from the SQL database in R.

In general, extract.nyctaxi function download data of the four types of hail service data (yellow taxi, green taxi, uber, and lyft) from the corresponding sources. transform.nyctaxi uses different techniques to clean all four types of data to get then ready for the next step. extract.load loads the data user selected to a SQL database.

nyctaxi lives on the Comprehensive R Archive Network (CRAN), and Packages can be installed with the install.packages() function in R.

```r
# install the package
install.packages("nyctaxi")
```

```r
# load the package
library(nyctaxi)
library(RMySQL)
```

Users need to create an etl object in order to apply the etl operations to it, and only the name of the SQL database, working directory, and type of SQL database need to be specified during initialization. If the type of SQL database is not specified, a local RSQLite database will be generated as default.

```
# initializing an etl object
db <- src_mysql("nyctaxi", user = "urname", host = "host",
    password = "pw")
taxi <- etl("nyctaxi", dir = "~/Desktop/nyctaxi", db)
```

In the example above, a folder called nyctaxi is created on the desktop and a connection to a MySQL database is generated. In the procession of initialization, a local folder contains two subfolders, `raw` and `load`, are also created under the directory the user specifies. `raw` folder stores data downloaded from online open sources, and `load` folder stores cleaned CSV data files that are ready to be loaded into SQL database. The ETL framework keeps data directly scraped from online data sources in their original forms. In this way, the original data is always available to users in case data corruption happens in later stages.

After an etl object is created (nyctaxi is the etl object in this case), four parameters are needed to specify the data that users want: (1) obj: an etl object (2) years: a numeric vector giving the years. The default is the most recent year. (3) months: a numeric vector giving the months. The default is January to December. (4) type: a character variable giving the type of data the user wants to download. There are four types: yellow, green, uber, and lyft. The default is yellow.

2.2.1 Taxi zone shapefile attached to nyctaxi R package

Two datasets are attached to `nyctaxi`. The first one is called `taxi_zones`, and this dataset contains information, such as taxi zone location IDs, location names, and corresponding boroughs for each ID. A shapefile containing the boundaries for the taxi zones, `taxi_zone_lookup`, is also included in the package for users to do spatial analysis.

```r
data("taxi_zones")
# plot(taxi_zones)
```

2.3 Extract-Transform-Load

2.3.1 Extract

`etl_extract.nyctaxi` allows users to download New York City yellow taxi, green taxi, Uber, and Lyft data from the corresponding data sources. It takes the `years`, `months`, and `type` parameters and download the New York City taxi data specified by users. New York City Yellow and Green Taxi data are updated on NYC Taxi & Limousine Commission (TLC) website on a monthly basis.

Uber trip record data is static and small, so we decided to only give users the options to either download all data from April to Sepetember, 2014 or download all Uber trip records from Janaury to June, 2015 at onc. Users do not have the lesuire to download Uber data from a specific month.

Lyft data is updated on NYC Open Data webiste on a weekly basis. Since the weekly-aggregated data is tiny and only data later then 2014 is available, we decided to only allow users to download Lyft data by year.

The default `years` is the current year, and the default `months` are the all twelve months. The default type of transportation is `yellow`. When an invalid month is entered, warning message will suggest users to reconsider their choice and select a new set of month.

2.3.2 Transform

`etl_transform.nyctaxi` allows users to transform New York City yellow taxi, green taxi, Uber, and Lyft data into cleaned formats, and it utlizes different data cleaning techiniques when it transforms data for each transportation type. In general, it cleans the data and creates a new `csv` file in the `load` directory to store the cleaned data. It helps us to retain and protect raw data from being mondified or destroyed. Users are allows to specify the month of interest in order to only transform the data that they are interested in. This functionality helps people to be more efficient with their use of time.

By default, it takes the current year Yellow taxi trip records data files, and save copies of them in the `load` diectory. It skips the cleaning step, because the raw Yellow Taxi data downloaded from TLC is already in a desired format with all variables correctly labelled.

There are a few main transformations that are done by this fucntion:

Green Taxi – Extra Blank Row and Column

Green Taxi monthly data from August 2013 to the most recent month besides 2015 all have a blank second row in the `csv` files. Similar to this problem, Green Taxi data from 2013, 2014, and 2015 all have an extra blank columns attached to the right-most column. These blank row and solumn causes problems in the later stage when users want to load data into SQL database. In order to get Green Taxi data ready for the `load` phase, we used `system()` to invoke the OS command specified to remove the blank rows and columns.

Uber Data – Reconciling Inconsistent Filenames

Uber only released over 4.5 million data records from April to Sepetmber 2014 and 14.3 million records from Janaury to June 2015. Information of different sets of variables are released for 2014 and 2015, and variables have different naming convention. When users want to download data from both years, variables are renamed so that data from both years can be cosolidated into one big dataset with consistent variable names.

Uber Data – Reconciling Inconsistent Data Formats

The data type of `Date/Time` variable in Uber datasets is originally encoded as `character`. In order to enable it to be recognized as `timestamp` by R, we use `ymd_hms` in `lubridate` to transform date time to POSIXct objects.

Optimizing I/O Process

h p://scholarworks.smith.edu/theses/1871 Zhang, Weijia, "Improving access to open-source data about the NYC bike sharing system (Citi Bike)" (2017). eses, Dissertations, and Projects. 1871.

Meking the file inout and output processes more efficient is an important pasrt of `etl_transform`. According to Zhang's (2017) study, `data.table` only takes half of the time to read from and write into datasets comparing to `readr`. Therefore, `etl_transform` uses `fread()` and `fwrite()` from `data.table` instead of `read_csv` or `write_csv` from `readr` to reduce the data processing time.

2.3.3 Load

`etl_load.nyctaxi` allows users to load New York City yellow taxi, green taxi, Uber, and Lyft data into different data tables in a SQL database. It populates a SQL database with data cleaned by `etl_transform`. In order to reduce the data processing complexity, `init.mysql()` is written under `nyctaxi` to help users to set up five basic table structures for MySQL database.

2.3.4 SQL Database Initialization

`init.mysql()` helps users to set up five basic table structures for MySQL database. `yellow_old` is created for Yellow Taxi data that are prior to August 2016, and `yellow` is created for data later than July 2016. `green`, `uber`, and `lyft` are also initiated for the three transportations.

`etl_init()` can be run after a database connection is built to process to process `init.mysql()` to initialize a MySQL database, and default columns with the correct variable names and typed defined will be automatically generated.

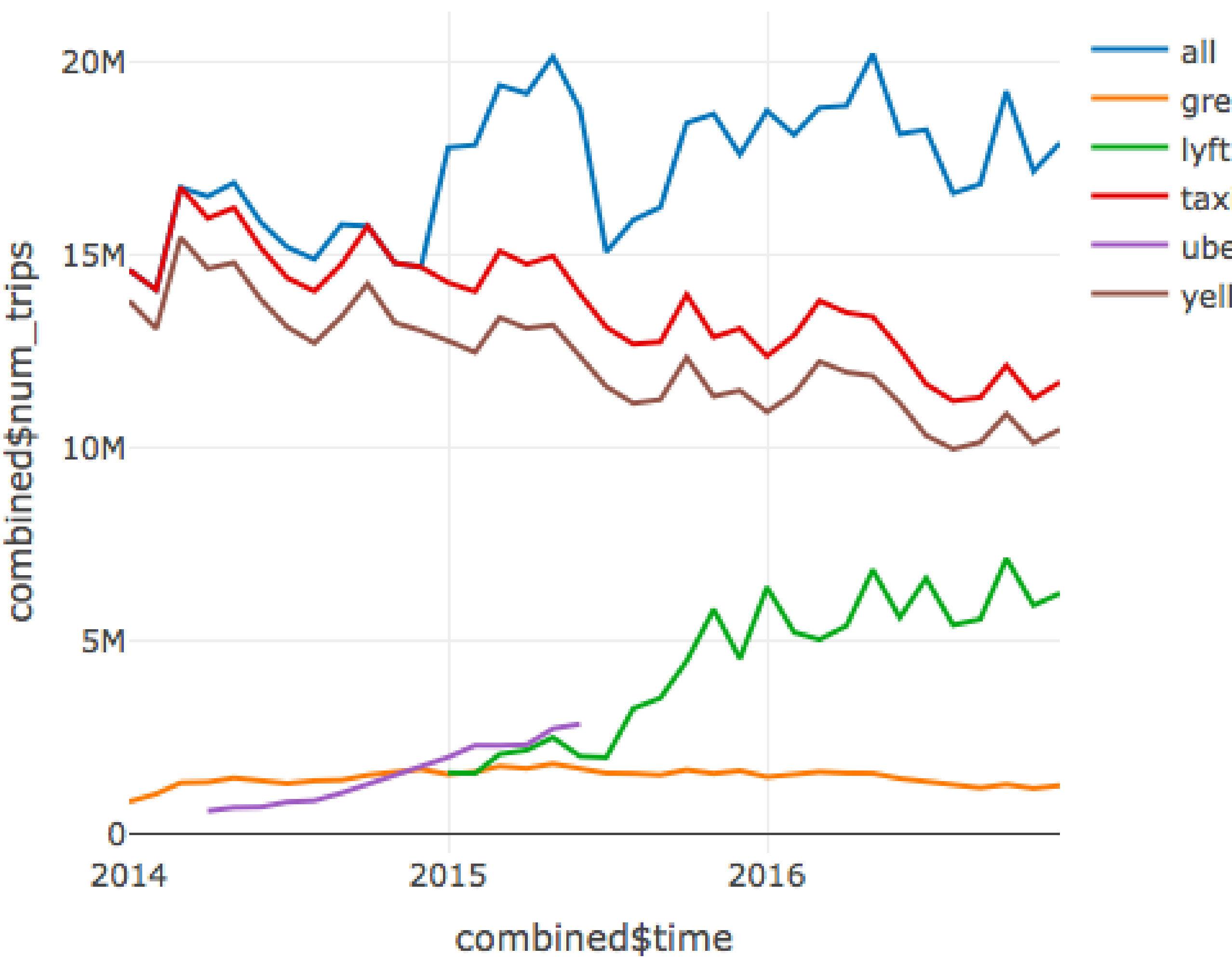
```r
db <- src_mysql("nyctaxi", user = "urname", host = "host",
    password = "pw")
taxi <- etl("nyctaxi", dir = "~/Desktop/nyctaxi", db)
taxi %>% etl_init()
```

In order to increase the query speed at the data analysis stage, `KEY`s are created for multiple variables for each transportation. Since there is no variable containing unique value for each observation, no primary variable is needed. Using `KEY` in data analysis query can spped up the query process.

Due to the large size of Yellow Taxi datasets, `yellow_old` and `yellow` are partitioned into subgroups by `year`. When we need to run query on data from a specific year, having partitions allows MySQL to directly find the data specified without filtering on every single row. It speeds up the query process. A `VIEW` called `yellow_old_sum` is also created to generate a summary table for the number of Yellow Taxi trips in each month.

2.4 New Yoek City Street-hail and E-hail Services Summary

Below is a summary of data that are available to users from 2014 to 2016.



2.5 Source Code

2.5.1 ETL Extract

```
opts_chunk$set(tidy.opts=list(width.cutoff=60))


etl_extract.etl_nyctaxi <-
  function(obj,
years = as.numeric(format(Sys.Date(),'%Y')),
                                    months = 1:12,
                                    type  = "yellow",...) {
  #TAXI YELLOW-------------------------
  taxi_yellow <- function(obj, years, months,...) {
    message("Extracting raw yellow taxi data...")
    remote <- etl::valid_year_month(years, months,
    begin = "2009-01-01") %>%
      mutate_(src =
  ~file.path("https://s3.amazonaws.com/nyc-tlc/trip+data",
                               paste0("yellow", "_tripdata_", year, "-",
                      stringr::str_pad(month, 2, "left", "0"), ".csv")))
    tryCatch(expr = etl::smart_download(obj, remote$src, ...),
             error = function(e){warning(e)},
             finally = warning("Only the following data are availabel on
                                TLC: Yellow taxi data: 2009 Jan -
                               last month"))}
  #TAXI GREEN----------------------
  taxi_green <- function(obj, years, months,...) {
```

```
  message("Extracting raw green taxi data...")
  remote <- etl::valid_year_month(years, months, begin = "2013-08-01") %>%
    mutate_(src =
              ~file.path("https://s3.amazonaws.com/nyc-tlc/trip+data",
                         paste0("green", "_tripdata_", year, "-",
                      stringr::str_pad(month, 2, "left", "0"), ".csv")))
  tryCatch(expr = etl::smart_download(obj, remote$src, ...),
           error = function(e){warning(e)},
           finally = warning("Only the following data are availabel on TLC:
                         Green taxi data: 2013 Aug - last month"))}
#UBER--------------------------------
uber <- function(obj, years, months,...) {
  message("Extracting raw uber data...")
  raw_month_2014 <- etl::valid_year_month(years = 2014, months = 4:9)
  raw_month_2015 <- etl::valid_year_month(years = 2015, months = 1:6)
  raw_month <- bind_rows(raw_month_2014, raw_month_2015)
  path = "https://raw.githubusercontent.com/
  fivethirtyeight/uber-tlc-foil-response/master/uber-trip-data"
  remote <- etl::valid_year_month(years, months)
  remote_small <- intersect(raw_month, remote)
  if (2015 %in% remote_small$year && !(2014 %in% remote_small$year)){
    #download 2015 data
    message("Downloading Uber 2015 data...")
    etl::smart_download(obj, "https://github.com/fivethirtyeight/
                    uber-tlc-foil-response/raw/master/
                uber-trip-data/uber-raw-data-janjune-15.csv.zip",...)}
  else if (2015 %in% remote_small$year && 2014 %in% remote_small$year) {
```

```
    #download 2015 data
    message("Downloading Uber 2015 data...")
    etl::smart_download(obj, "https://github.com/fivethirtyeight/
                  uber-tlc-foil-response/raw/master/uber-trip-data
                    /uber-raw-data-janjune-15.csv.zip",...)
    #download 2014 data
    small <- remote_small %>%
      filter_(~year == 2014) %>%
      mutate_(month_abb = ~tolower(month.abb[month]),
              src = ~file.path(path,
              paste0("uber-raw-data-",month_abb,
              substr(year,3,4),".csv")))
    message("Downloading Uber 2014 data...")
    etl::smart_download(obj, small$src,...)
  } else if (2014 %in% remote_small$year &&
  !(2015 %in% remote_small$year)) {
    message("Downloading Uber 2014 data...")
    #file paths
    small <- remote_small %>%
      mutate_(month_abb =
                ~tolower(month.abb[month]),
              src = ~file.path(path,
              paste0("uber-raw-data-",month_abb,
              substr(year,3,4),".csv")))
    etl::smart_download(obj, small$src,...)}
  else {warning("The Uber data you requested are
                not currently available. Only data
```

```
                from 2014/04-2014/09 and 2015/01-
                2015/06 are available...")}
  }
#LYFT--------------------------------
lyft <- function(obj, years, months,...){
  message("Extracting raw lyft data...")
  #check if the week is valid
  valid_months <- etl::valid_year_month(years, months,
  begin = "2015-01-01")
  base_url = "https://data.cityofnewyork.us/
  resource/edp9-qgv4.csv"
  valid_months <- valid_months %>%
    mutate_(new_filenames =
              ~paste0("lyft-", year, ".csv")) %>%
    mutate_(drop = TRUE)
  #only keep one data set per year
  year <- valid_months[1,1]
  n <- nrow(valid_months)
  for (i in 2:n) {
    if(year == valid_months[i-1,1]) {
      valid_months[i,6] <- FALSE
      year <- valid_months[i+1,1]
    } else {
      valid_months[i,6] <- TRUE
      year <- valid_months[i+1,1]}
    }
  row_to_keep = valid_months$drop
```

```
    valid_months <- valid_months[row_to_keep,]

    #download lyft files, try two different methods
    first_try<-tryCatch(
      download_nyc_data(obj, base_url, valid_months$year,
      n = 50000, names = valid_months$new_filenames),
      error = function(e){warning(e)},
      finally = 'method = "libcurl" fails')
  }

  if (type == "yellow"){taxi_yellow(obj, years, months,...)}
  else if (type == "green"){taxi_green(obj, years, months,...)}
  else if (type == "uber"){uber(obj, years, months,...)}
  else if (type == "lyft"){lyft(obj, years, months,...)}
  else {message("The type you chose does not exit...")}

  invisible(obj)
}
```

2.5.2 ETL Transform

```
opts_chunk$set(tidy.opts=list(width.cutoff=60))
etl_transform.etl_nyctaxi <- function(obj,
                          years = as.numeric(format(Sys.Date(),'%Y')),
                          months = 1:12,
                          type  = "yellow",...) {
  #TAXI YELLOW-----------------------------
```

```
taxi_yellow <- function(obj, years, months) {
  message("Transforming yellow taxi data from raw to
          load directory...")
  #create a df of file path of the files that the user wants to transform
  remote <- etl::valid_year_month(years, months,
  begin = "2009-01-01") %>%
    mutate_(src = ~file.path(attr(obj, "raw_dir"),
    paste0("yellow", "_tripdata_", year, "-",
    stringr::str_pad(month, 2, "left", "0"), ".csv")))
  #create a df of file path of the files that are in the raw directory
  src <- list.files(attr(obj, "raw_dir"), "yellow", full.names = TRUE)
  src_small <- intersect(src, remote$src)
  #Move the files
  in_raw <- basename(src_small)
  in_load <- basename(list.files(attr(obj, "load_dir"), "yellow",
  full.names = TRUE))
  file_remian <- setdiff(in_raw,in_load)
  file.copy(file.path(attr(obj, "raw_dir"),file_remian),
            file.path(attr(obj, "load_dir"),file_remian) )}
#TAXI GREEN-----------------------------
taxi_green <- function(obj, years, months) {
  message("Transforming green taxi data from raw
          to load directory...")
  #create a df of file path of the files that the user wants to transform
  remote <- etl::valid_year_month(years, months,
  begin = "2013-08-01") %>%
    mutate_(src = ~file.path(attr(obj, "raw_dir"),
```

```
    paste0("green", "_tripdata_", year, "-",
    stringr::str_pad(month, 2, "left", "0"), ".csv")))
  #create a df of file path of the files that are in the raw directory
  src <- list.files(attr(obj, "raw_dir"), "green", full.names = TRUE)
  src_small <- intersect(src, remote$src)
  #Clean the green taxi data files
  #get rid of 2nd blank row
  if (length(src_small) == 0){
    message("The files you requested are not available
            in the raw directory.")
  } else{
    #a list of the ones that have a 2nd blank row
    remote_green_1 <- remote %>% filter_(~year != 2015)
    src_small_green_1 <- intersect(src, remote_green_1$src)
    # check that the sys support command line,
    #and then remove the blank 2nd row
    if(length(src_small_green_1) != 0) {
      if (.Platform$OS.type == "unix"){
        cmds_1 <- paste("sed -i -e '2d'", src_small_green_1)
        lapply(cmds_1, system)
      } else {
        message("Windows system does not
        currently support removing the 2nd blank row
        in the green taxi datasets. This might affect
        loading data into SQL...")}
      }else {
        "You did not request for any
```

```
    green taxi data, or all the green
    taxi data you requested are cleaned."}
#fix column number
remote_green_2 <- remote %>%
  filter_(~year %in% c(2013, 2014, 2015)) %>%
  mutate_(keep =
            ~ifelse(year %in% c(2013,2014), 20,21),
          new_file =
            ~paste0("green_tripdata_", year, "_",
                stringr::str_pad(month, 2, "left", "0"),
                             ".csv"))
src_small_green_2 <- intersect(src, remote_green_2$src)
src_small_green_2_df <- data.frame(src_small_green_2)
names(src_small_green_2_df) <- "src"
src_small_green_2_df <- inner_join(src_small_green_2_df,
remote_green_2, by = "src")
src_small_green_2_df <- src_small_green_2_df %>%
  mutate(cmds_2 = paste("cut -d, -f1-", keep," ",src, " > ",
  attr(obj, "raw_dir"),"/green_tripdata_",
  year, "_", stringr::str_pad(month, 2, "left", "0"),".csv",
  sep = ""))
#remove the extra column
if(length(src_small_green_2) != 0) {
  if (.Platform$OS.type == "unix"){
    lapply(src_small_green_2_df$cmds_2, system)}
  else {
    message("Windows system does not currently
```

```
        support removing the 2nd blank row
        in the green taxi datasets. This might
        affect loading data into SQL...")}
      }else {
        "All the green taxi data you
        requested are in cleaned formats."}
    #Find the files paths of the files that need to be transformed
    file.rename(file.path(dirname(src_small_green_2_df$src),
                          src_small_green_2_df$new_file),
                file.path(attr(obj, "load_dir"),
                basename(src_small_green_2_df$src)))
    #Move the files
    in_raw <- basename(src_small)
    in_load <- basename(list.files(attr(obj, "load_dir"),
    "green", full.names = TRUE))
    file_remian <- setdiff(in_raw,in_load)
    file.copy(file.path(attr(obj, "raw_dir"),file_remian),
    file.path(attr(obj, "load_dir"),file_remian) )}}
#UBER--------------------------------
uber <- function(obj) {
  message("Transforming uber data from raw to load directory...")
  #creat a list of 2014 uber data file directory
  uber14_list <- list.files(path = attr(obj, "raw_dir"),
  pattern = "14.csv")
  uber14_list <- data.frame(uber14_list)
  uber14_list <- uber14_list %>% mutate_(file_path =
  ~file.path(attr(obj, "raw_dir"), uber14_list))
```

```
uber14file <- lapply(uber14_list$file_path, readr::read_csv)
n <- length(uber14file)
if (n == 1) {
  uber14 <- data.frame(uber14file[1])
} else if (n == 2) {
  uber14 <- bind_rows(uber14file[1], uber14file[2])
} else if (n > 2) {
  uber14 <- bind_rows(uber14file[1], uber14file[2])
  for (i in 3:n){uber14 <- bind_rows(uber14, uber14file[i])}
}
substrRight <- function(x, n){substr(x, nchar(x)-n+1, nchar(x))}
uber14_datetime <- uber14 %>%
  mutate(date = gsub( " .*$", "", `Date/Time`),
  len_date = nchar(date),
         time = sub('.*\\ ', '', `Date/Time`))
uber14_datetime <- uber14_datetime %>%
  mutate(month =
           substr(`Date/Time`, 1, 1),
         day = ifelse(len_date == 8,
         substr(`Date/Time`, 3,3),substr(`Date/Time`, 3,4)),
         pickup_date =
           lubridate::ymd_hms(paste0("2014-", month, "-",
                                     day, " ", time)))
uber14_df <- uber14_datetime[-c(1,5:9)]

#2015
zipped_uberfileURL <- file.path(attr(obj, "raw_dir"),
```

```
"uber-raw-data-janjune-15.csv.zip")
raw_month_2015 <- etl::valid_year_month(years = 2015, months = 1:6)
remote_2015 <- etl::valid_year_month(years, months)
remote_small_2015 <- inner_join(raw_month_2015, remote_2015)
if(file.exists(zipped_uberfileURL) &&
   nrow(remote_small_2015) != 0){
  utils::unzip(zipfile = zipped_uberfileURL,unzip = "internal",
  exdir = file.path(tempdir(), "uber-raw-data-janjune-15.csv.zip"))
  uber15 <- readr::read_csv(file.path(tempdir(),
  "uber-raw-data-janjune-15.csv.zip",
  "uber-raw-data-janjune-15.csv"))}

names(uber14_df) <- c("lat", "lon", "affiliated_base_num",
"pickup_date")
names(uber15) <- tolower(names(uber15))
uber <- bind_rows(uber14_df, uber15)
utils::write.csv(uber, file.path(tempdir() ,"uber.csv"))
if(nrow(uber) != 0) {
  if (.Platform$OS.type == "unix"){cmds_3 <-
  paste("cut -d, -f2-7",file.path(tempdir(),"uber.csv"), " > ",
  file.path(attr(obj, "load_dir"),"uber.csv"))
    lapply(cmds_3, system)
  } else {
    message("Windows system does not currently
    support removing the 2nd blank row
    in the green taxi datasets. This might
    affect loading data into SQL...")}
```

```
    }else {
      "You did not request for any
      green taxi data, or all the green
      taxi data you requested are cleaned."}
  }
#LYFT--------------------------------
lyft <- function(obj, years, months){
  valid_months <- etl::valid_year_month(years, months = 1,
  begin = "2015-01-01")
  message("Transforming lyft data from raw to load directory...")
  src <- list.files(attr(obj, "raw_dir"), "lyft", full.names = TRUE)
  src_year <- valid_months %>% distinct_(~year)
  remote <- data_frame(src)
  remote <- remote %>%
    mutate_(lcl = ~file.path(attr(obj, "load_dir"),basename(src)),
            basename = ~basename(src), year = ~substr(basename,6,9))
  class(remote$year) <- "numeric"
  remote <- inner_join(remote,src_year, by = "year" )
  for(i in 1:nrow(remote)) {
      datafile <- readr::read_csv(remote$src[i])
      readr::write_delim(datafile, path = remote$lcl[i],
      delim = "|", na = "")}}

#transform the data from raw to load
if (type == "yellow"){taxi_yellow(obj, years, months)}
else if (type == "green"){taxi_green(obj, years, months)}
else if (type == "uber"){uber(obj)}
```

```
  else if (type == "lyft"){lyft(obj, years, months)}
  else {message("The type you chose does not exit...")}

  invisible(obj)
}
```

2.5.3 ETL Load

```
opts_chunk$set(tidy.opts=list(width.cutoff=60))
etl_load.etl_nyctaxi <- function(obj,
 years = as.numeric(format(Sys.Date(),'%Y')),
                                months = 1:12,
                                type  = "yellow", ...) {
  #TAXI YELLOW-----------------------------
  taxi_yellow <- function(obj, years, months,...) {
    #create a df of file path of the files that are in the load directory
    src <- list.files(attr(obj, "load_dir"), "yellow",
    full.names = TRUE)
    src <- data.frame(src)

    #files before 2016-07
    remote_old <- etl::valid_year_month(years, months,
    begin = "2009-01-01", end = "2016-06-30") %>%
      mutate_(src = ~file.path(attr(obj, "load_dir"),
      paste0("yellow", "_tripdata_", year, "-",
      stringr::str_pad(month, 2, "left", "0"), ".csv")))
    src_small_old <- inner_join(remote_old, src, by = "src")
```

```
#files later then 2017-06
remote_new <- etl::valid_year_month(years, months,
begin = "2016-07-01") %>%
  mutate_(src =  ~file.path(attr(obj, "load_dir"),
  paste0("yellow", "_tripdata_", year, "-",
  stringr::str_pad(month, 2, "left", "0"), ".csv")))
src_small_new <- inner_join(remote_new, src, by = "src")
#data earlier than 2016-07
if(nrow(src_small_old) == 0) {
  message("The taxi files (earlier than 2016-07)
          you requested are not available in
          the load directory...")
} else {
  message("Loading taxi data from
          load directory to a sql database...")
  mapply(DBI::dbWriteTable,
         name = "yellow_old", value = src_small_old$src,
         MoreArgs =
           list(conn = obj$con, append = TRUE))}

#data later then 2016-06
if(nrow(src_small_new) == 0) {
  message("The new taxi files (later than 2016-06)
          you requested are not available in the
          load directory...")
} else {
  message("Loading taxi data from load
```

```
            directory to a sql database...")
    mapply(DBI::dbWriteTable,
           name = "yellow", value = src_small_new$src,
           MoreArgs =
             list(conn = obj$con, append = TRUE))}

  }
#TAXI GREEN----------------------------
taxi_green <- function(obj, years, months,...) {
  #create a list of file that the user wants to load
  remote <- etl::valid_year_month(years, months,
  begin = "2013-08-01") %>%
    mutate_(src = ~file.path(attr(obj, "load_dir"),
    paste0("green", "_tripdata_", year, "-",
    stringr::str_pad(month, 2, "left", "0"), ".csv")))
  #create a df of file path of the files that are in the load directory
  src <- list.files(attr(obj, "load_dir"), "tripdata",
  full.names = TRUE)
  src <- data.frame(src)
  #only keep the files thst the user wants to transform
  src_small <- inner_join(remote, src, by = "src")
  if(nrow(src_small) == 0) {
    message("The taxi files you requested
            are not available in the
            load directory...")
  } else {
    message("Loading taxi data from
```

```
            load directory to a sql database...")
    mapply(DBI::dbWriteTable,
           name = "green", value = src_small$src,
           MoreArgs =
             list(conn = obj$con, append = TRUE, ... = ...))}}
#UBER--------------------------------
uber <- function(obj,...) {
  uberfileURL <- file.path(attr(obj, "load_dir"), "uber.csv")
  if(file.exists(uberfileURL)) {
    message("Loading uber data from
            load directory to a sql database...")
    DBI::dbWriteTable(conn = obj$con, name = "uber",
    value = uberfileURL, append = TRUE, ... = ...)
  } else {
    message("There is no uber data
            in the load directory...")}}
#LYFT---------------------------------
lyft <- function(obj, years, months,...){
  message("Loading lyft data from
          load directory to a sql database...")
  #create a list of file that the user wants to load
  valid_months <- etl::valid_year_month(years, months,
  begin = "2015-01-01")
  src <- list.files(attr(obj, "load_dir"), "lyft",
  full.names = TRUE)
  src_year <- valid_months %>% distinct_(~year)
  remote <- data_frame(src)
```

```
    remote <- remote %>% mutate_(tablename = ~"lyft",
    year =~substr(basename(src),6,9))
    class(remote$year) <- "numeric"
    remote <- inner_join(remote,src_year, by = "year" )
    if(nrow(remote) != 0) {
      write_data <- function(...) {
        lapply(remote$src, FUN = DBI::dbWriteTable,
        conn = obj$con, name = "lyft", append = TRUE,
        sep = "|", ... = ...)}
      write_data(...)
    } else {
      message("The lyft files you requested
              are not available in the
              load directory...")}}

  if (type == "yellow"){taxi_yellow(obj, years, months,...)
  }else if (type == "green"){taxi_green(obj, years, months,...)
  }else if (type == "uber"){uber(obj,...)
  }else if (type == "lyft"){lyft(obj, years, months,...)
  }else {message("The type you chose does not exit...")
          }

  invisible(obj)
}
```

2.5.4 utility function

This utility function below was written to shortened the source code in ETL extract.

```
opts_chunk$set(tidy.opts=list(width.cutoff=60))
download_nyc_data <- function(obj, url, years, n, names, ...) {
  url <- paste0(url,"?years=",
              years,"&$limit=", n)
  lcl <- file.path(attr(obj, "raw"), names)
  downloader::download(url, destfile = lcl, ...)
  lcl
}
```

2.5.5 ETL Init

```
DROP TABLE IF EXISTS `yellow_old`;

CREATE TABLE `yellow_old` (
 `VendorID` tinyint DEFAULT NULL,
 `tpep_pickup_datetime` DATETIME NOT NULL,
 `tpep_dropoff_datetime` DATETIME NOT NULL,
 `passenger_count` tinyint DEFAULT NULL,
 `trip_distance` float(10,2) DEFAULT NULL,
 `pickup_longitude` double(7,5) DEFAULT NULL,
 `pickup_latitude` double(7,5) DEFAULT NULL,
 `RatecodeID` tinyint DEFAULT NULL,
 `store_and_fwd_flag` varchar(10) COLLATE latin1_general_ci DEFAULT NULL,
```

```
 `dropoff_longitude` double(7,5) DEFAULT NULL,
 `dropoff_latitude` double(7,5) DEFAULT NULL,
 `payment_type` tinyint DEFAULT NULL,
 `fare_amount` decimal(5,3) DEFAULT NULL,
 `extra` decimal(5,3) DEFAULT NULL,
 `mta_tax` decimal(5,3) DEFAULT NULL,
 `tip_amount` decimal(5,3) DEFAULT NULL,
 `tolls_amount` decimal(5,3) DEFAULT NULL,
 `improvement_surcharge` decimal(5,3) DEFAULT NULL,
 `total_amount` decimal(5,3) DEFAULT NULL,
 KEY `VendorID` (`VendorID`),
 KEY `pickup_datetime` (`tpep_pickup_datetime`),
 KEY `dropoff_datetime` (`tpep_dropoff_datetime`),
 KEY `pickup_longitude` (`pickup_longitude`),
 KEY `pickup_latitude` (`pickup_latitude`),
 KEY `dropoff_longitude` (`dropoff_longitude`),
 KEY `dropoff_latitude` (`dropoff_latitude`)
)
PARTITION BY RANGE( YEAR(tpep_pickup_datetime) ) (
  PARTITION p09 VALUES LESS THAN (2010),
  PARTITION p10 VALUES LESS THAN (2011),
  PARTITION p11 VALUES LESS THAN (2012),
  PARTITION p12 VALUES LESS THAN (2013),
  PARTITION p13 VALUES LESS THAN (2014),
  PARTITION p14 VALUES LESS THAN (2015),
  PARTITION p15 VALUES LESS THAN (2016),
  PARTITION p16 VALUES LESS THAN (2017)
```

```
);

DROP TABLE IF EXISTS `yellow`;

CREATE TABLE `yellow` (
 `VendorID` tinyint DEFAULT NULL,
 `tpep_pickup_datetime` DATETIME NOT NULL,
 `tpep_dropoff_datetime` DATETIME NOT NULL,
 `passenger_count` tinyint DEFAULT NULL,
 `trip_distance` float(10,2) DEFAULT NULL,
 `RatecodeID` tinyint DEFAULT NULL,
 `store_and_fwd_flag` varchar(10) COLLATE latin1_general_ci DEFAULT NULL,
 `PULocationID` tinyint DEFAULT NULL,
 `DOLocationID` tinyint DEFAULT NULL,
 `payment_type` tinyint DEFAULT NULL,
 `fare_amount` decimal(5,3) DEFAULT NULL,
 `extra` decimal(5,3) DEFAULT NULL,
 `mta_tax` decimal(5,3) DEFAULT NULL,
 `tip_amount` decimal(5,3) DEFAULT NULL,
 `tolls_amount` decimal(5,3) DEFAULT NULL,
 `improvement_surcharge` decimal(5,3) DEFAULT NULL,
 `total_amount` decimal(5,3) DEFAULT NULL,
 KEY `VendorID` (`VendorID`),
 KEY `pickup_datetime` (`tpep_pickup_datetime`),
 KEY `dropoff_datetime` (`tpep_dropoff_datetime`),
 KEY `PULocationID` (`PULocationID`),
 KEY `DOLocationID` (`DOLocationID`)
```

```
)
PARTITION BY RANGE( YEAR(tpep_pickup_datetime) ) (
  PARTITION p16 VALUES LESS THAN (2017),
  PARTITION p17 VALUES LESS THAN (2018)
);


DROP TABLE IF EXISTS `green`;

CREATE TABLE `green` (
 `VendorID` tinyint DEFAULT NULL,
 `lpep_pickup_datetime` DATETIME NOT NULL,
 `Lpep_dropoff_datetime` DATETIME NOT NULL,
 `Store_and_fwd_flag` varchar(10) COLLATE latin1_general_ci DEFAULT NULL,
 `RatecodeID` tinyint DEFAULT NULL,
 `Pickup_longitude` double(7,5) DEFAULT NULL,
 `Pickup_latitude` double(7,5) DEFAULT NULL,
 `Dropoff_longitude` double(7,5) DEFAULT NULL,
 `Dropoff_latitude` double(7,5) DEFAULT NULL,
 `Passenger_count` tinyint DEFAULT NULL,
 `Trip_distance` float(10,2) DEFAULT NULL,
 `Fare_amount` decimal(5,3) DEFAULT NULL,
 `Extra` decimal(5,3) DEFAULT NULL,
 `MTA_tax` decimal(5,3) DEFAULT NULL,
 `Tip_amount` decimal(5,3) DEFAULT NULL,
 `Tolls_amount` decimal(5,3) DEFAULT NULL,
 `improvement_surcharge` decimal(5,3) DEFAULT NULL,
```

```
 `Total_amount` decimal(5,3) DEFAULT NULL,
 `Payment_type` tinyint DEFAULT NULL,
 `Trip_type` tinyint DEFAULT NULL,
 KEY `VendorID` (`VendorID`),
 KEY `pickup_datetime` (`lpep_pickup_datetime`),
 KEY `dropoff_datetime` (`Lpep_dropoff_datetime`)
);


DROP TABLE IF EXISTS `lyft`;

CREATE TABLE `lyft` (
 `base_license_number` varchar(15) COLLATE latin1_general_ci DEFAULT NULL,
 `base_name` varchar(40) COLLATE latin1_general_ci DEFAULT NULL,
 `dba` varchar(40) COLLATE latin1_general_ci DEFAULT NULL,
 `pickup_end_date` DATE NOT NULL,
 `pickup_start_date` DATE NOT NULL,
 `total_dispatched_trips` smallint DEFAULT NULL,
 `unique_dispatched_vehicle` smallint DEFAULT NULL,
 `wave_number` tinyint DEFAULT NULL,
 `week_number` tinyint DEFAULT NULL,
 `years` smallint DEFAULT NULL,
 KEY `base_name` (`base_name`),
 KEY `pickup_end_date` (`pickup_end_date`),
 KEY `pickup_start_date` (`pickup_start_date`)
);
```

```
DROP TABLE IF EXISTS `uber`;

CREATE TABLE `uber` (
 `lat` double(7,5) DEFAULT NULL,
 `lon` double(7,5) DEFAULT NULL,
 `dispatching_base_num` varchar(15) COLLATE latin1_general_ci DEFAULT NULL,
 `pickup_date` DATETIME NOT NULL,
 `affiliated_base_num` varchar(15) COLLATE latin1_general_ci DEFAULT NULL,
 `locationid` tinyint DEFAULT NULL,
 KEY `pickup_date` (`pickup_date`),
 KEY `locationid` (`locationid`)
);

CREATE VIEW yellow_old_sum AS SELECT YEAR(tpep_pickup_datetime) as the_year, MONTH(tpep_pi
  FROM yellow_old
  GROUP BY the_year, the_month;
);
```

Chapter 3

New York City Taxi Driver

3.1 Trip-level Tip Inofrmation

The income of Taxi drivers in New York City has two parts: taxi fare and tips. Taxi fare is usually calculated by the meters installed in the taxis, and the rate of fare cannot be changed by taxi drivers. Therefore, in order to make more profit, taxi drivers prefer to pick up passengers who offer big amount of tips. What are the regions that provide the most tips to yellow taxicab drivers?

In the following analysis, I will focus on trip data collected in August 2016. Taxi drivers usually does not correctly record the amount of tips paid by cash or check. Therefore, in order to find out the regions that offer the most tips, we need to filter out the trips that are not paid by credit or debit card.

```
yellow_2016.08_tip <- yellow_2016.08_cleaned %>% filter(fare_amount >
    0) %>% filter(tip_amount > 0) %>% filter(payment_type ==
    1) %>% filter(tip_amount < fare_amount)
```

Instead of the absolute amount of tips, we want to focus on the percentage of tips that passengers pay in addition to the total fare amount. Therefore, we use tip amount over fare amount to calculate the percent tip.

```
yellow_2016.08_tip <- yellow_2016.08_tip %>% mutate(tip_perct = tip_amount/fare_amount)
```

Let's visualize the distribution of percent tip of all trips occurred in August 2016:

```
library(ggplot2)
tip_individual <- ggplot(data = yellow_2016.08_tip,
    aes(x = tip_perct)) + xlab("Tips, percent") + geom_histogram(binwidth = 0.005) +
    geom_vline(xintercept = c(0.2), col = "red", linetype = "longdash") +
    geom_vline(xintercept = c(0.25), col = "green",
        linetype = "longdash") + geom_vline(xintercept = c(0.28),
    col = "yellow", linetype = "longdash")
tip_individual
```

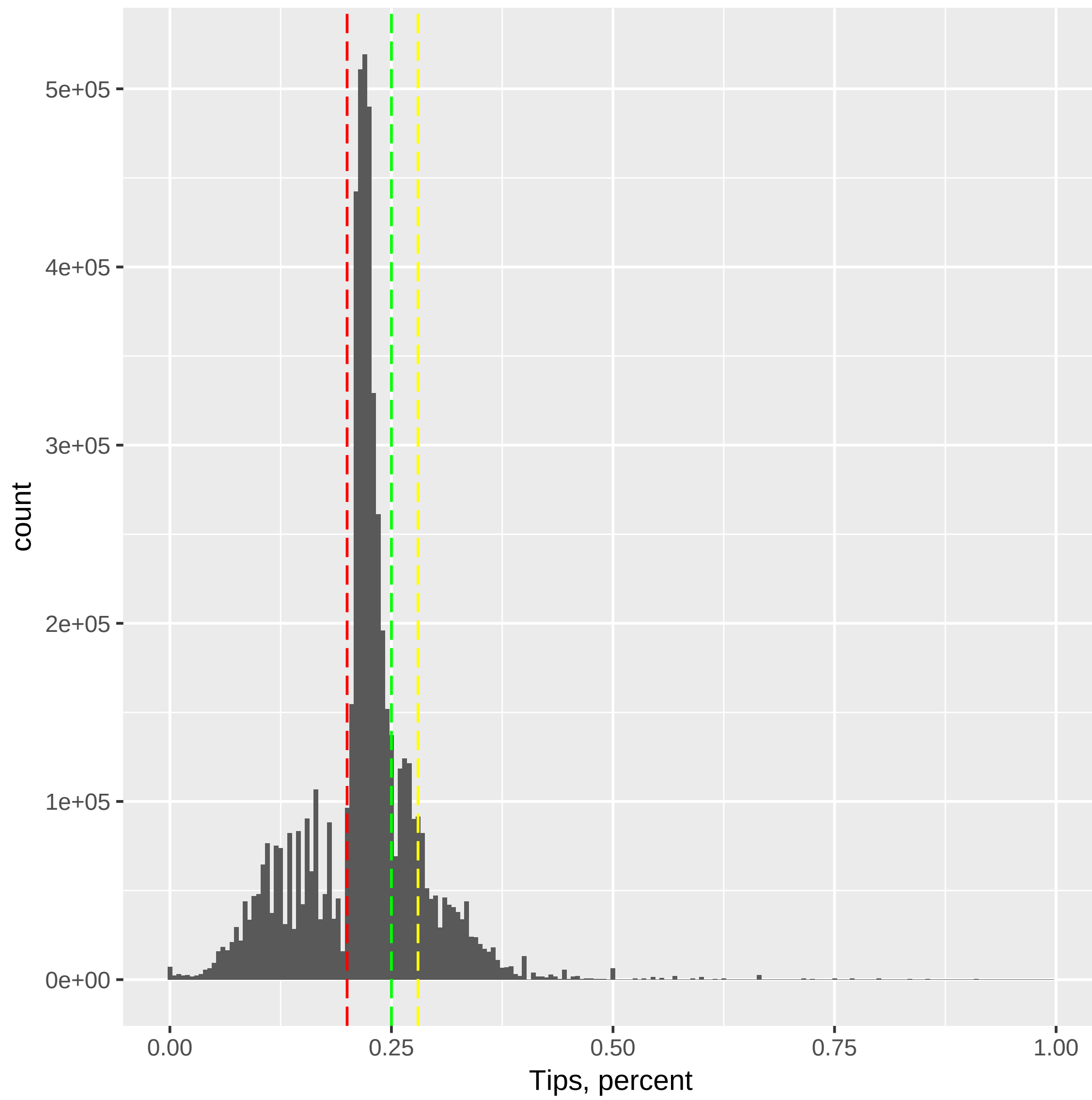


Figure 3.1: Percent Tip Paid by Passengers on Individual Yellow Taxi Trip in NYC

3.2 Aggregated Zone-level Tip Information

Instead of studying factors that affect individual trips' percent tip, it is more useful to study the aggregated effect of each zone on percent tip.

Table 3.1: Ten taxi pick-up zones with the highest average tip without any threshold

LocationID	avg_tip	num_trips	avg_dis	Borough	Zone
46	0.6000000	1	0.000000	Bronx	City Island
15	0.5052363	10	5.584000	Queens	Bay Terrace/Fort Totten
175	0.5020756	16	4.246875	Queens	Oakland Gardens
98	0.3541095	29	5.887241	Queens	Fresh Meadows
21	0.3270126	28	6.307857	Brooklyn	Bensonhurst East
135	0.3269817	77	5.126753	Queens	Kew Gardens Hills
11	0.3211306	14	4.470000	Brooklyn	Bath Beach
121	0.3040954	24	4.494167	Queens	Hillcrest/Pomonok
210	0.2997459	36	3.799444	Brooklyn	Sheepshead Bay
150	0.2987986	12	13.161667	Brooklyn	Manhattan Beach

```
data(taxi_zone_lookup)
```

Taxi drivers are required to be indifferent to where passengers are going. Therefore, it makes sense to investigate the average amount of tips paid for each pick-up zone. What are the taxi pick-up zones that have the highest tip percents?

We first calculate the average percent tip paid for each pick-up zone. Here is a list of pick-up zones with their average percent tip:

```
tip_pickup <- yellow_2016.08_tip %>% group_by(PULocationID) %>%
    summarise(avg_tip = mean(tip_perct), num_trips = n(),
        avg_dis = mean(trip_distance)) %>% rename(LocationID = PULocationID) %>%
    left_join(taxi_zone_lookup, by = "LocationID") %>%
    arrange(desc(avg_tip)) %>% filter(Zone != "Unknown")


library(knitr)
kable(tip_pickup[1:10, ], caption = "Ten taxi pick-up zones with the highest average tip
```

Below is a histogram of average percent tips paid for all pick-up zones. As show on the plot, the first peak is around 20%, which is the cheapest default option on the touch panel for passengers to chose.

```
pickup_vis <- ggplot(data = tip_pickup, aes(x = avg_tip)) +
    xlab("Tips, percent") + geom_histogram(binwidth = 0.005) +
    geom_vline(xintercept = c(0.2), col = "red", linetype = "longdash") +
    geom_vline(xintercept = c(0.25), col = "green",
        linetype = "longdash") + geom_vline(xintercept = c(0.28),
    col = "yellow", linetype = "longdash") + scale_x_continuous(limits = c(0,
    0.5))
pickup_vis
```

```
Warning: Removed 3 rows containing non-finite values (stat_bin).
```

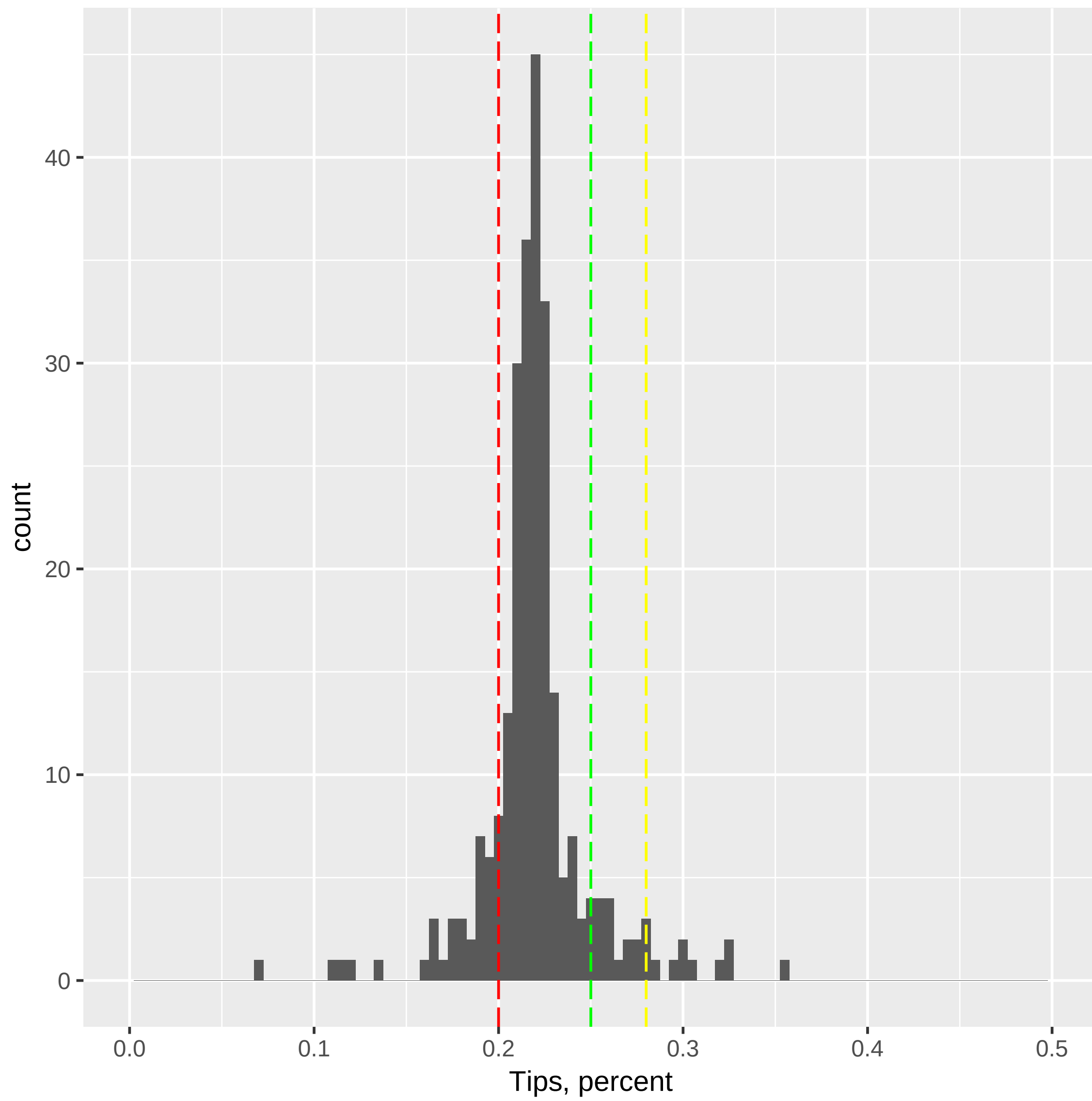
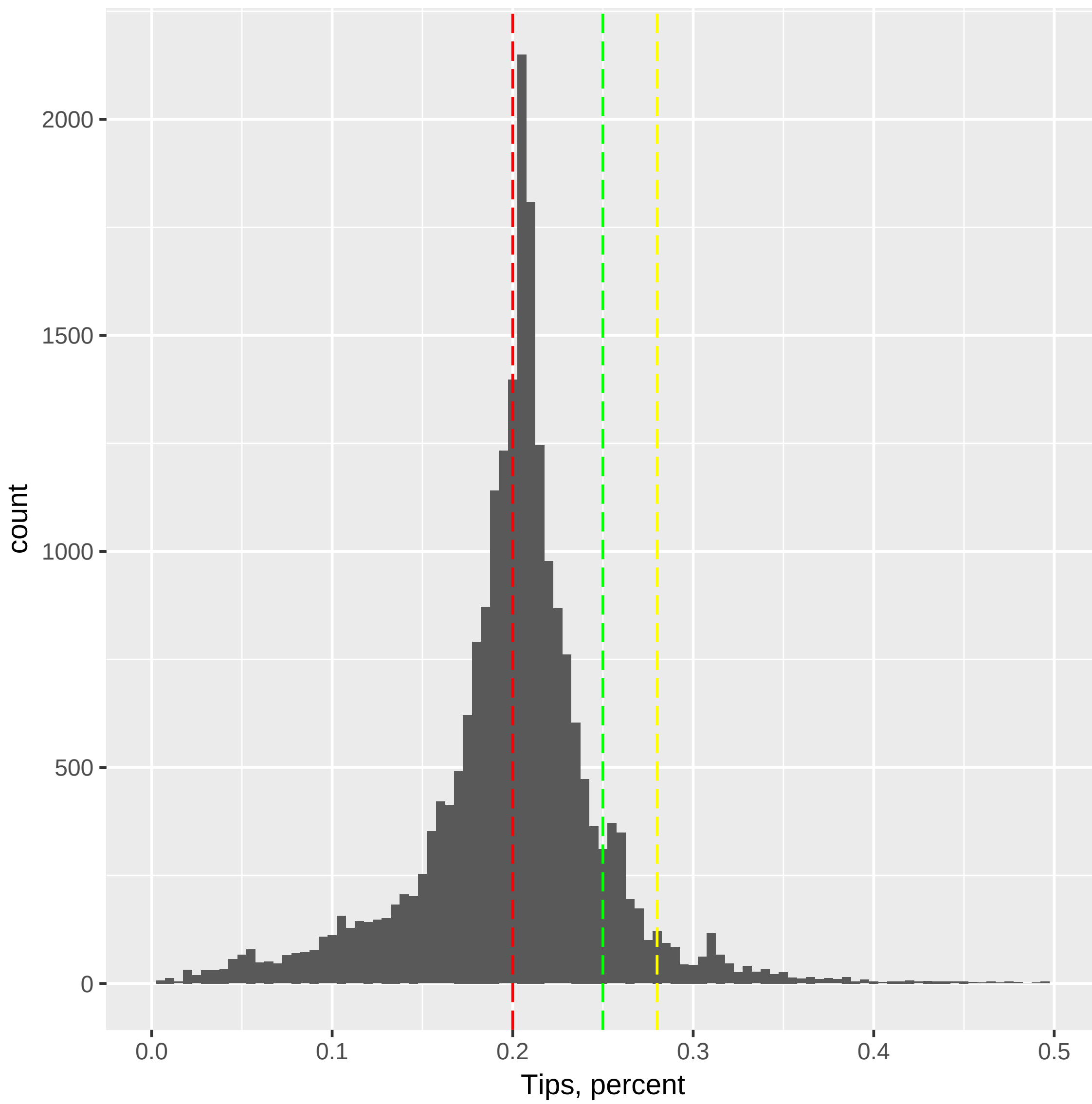


Figure 3.2: Percent Tip Paid by Passengers on Each Pick-up Taxi Zone in NYC

```
tip_region <- yellow_2016.08_tip %>% group_by(PULocationID,
    DOLocationID) %>% summarise(avg_tip = mean(tip_perct),
    trips = n(), avg_dis = mean(trip_distance)) %>%
    # filter(trips > 10) %>%
arrange(desc(avg_tip)) %>% rename(LocationID = PULocationID) %>%
    left_join(taxi_zone_lookup, by = "LocationID")
```

```
# zone

region_vis <- pickup_vis %+% tip_region

region_vis
```

```
Warning: Removed 95 rows containing non-finite values (stat_bin).
```



The 20% peak is more clearly shown when we calculate the percent tips for each pick-up and drop-off locatins pair instead of pick-up location only.

Does trip distance increase the percent tips paid? One of the questions that I always wonder is whether longer trips result in higher tip percent. It takes taxi drivers more time to complete longer trips, so passengers might want to compensate taxi drivers more. I personally pay higher percent of tips for longer rides, so I believe trip distance has an impact on percentage of tips paid.

```
tip_distance <- lm(avg_tip ~ avg_dis + LocationID +
    DOLocationID, data = tip_region)
summary(tip_distance)$coef[1:2, ]
```

```
                 Estimate   Std. Error     t value  Pr(>|t|)
(Intercept)  2.025175e-01 1.140007e-03 177.6457606 0.0000000
avg_dis     -3.083442e-07 1.565416e-06  -0.1969727 0.8438507
```

Acoording to the simple linear regression result, trip distance does not have significant impact on the percent of tips paid, controlling for both pick-up and drop-off locations.

3.3 Which zones have the highest percent tip?

Let's fist take a look at which pick-up zones have the highest number of pickups.

```
data("taxi_zones")
library(sp)
```

```
Warning: package 'sp' was built under R version 3.4.3
```

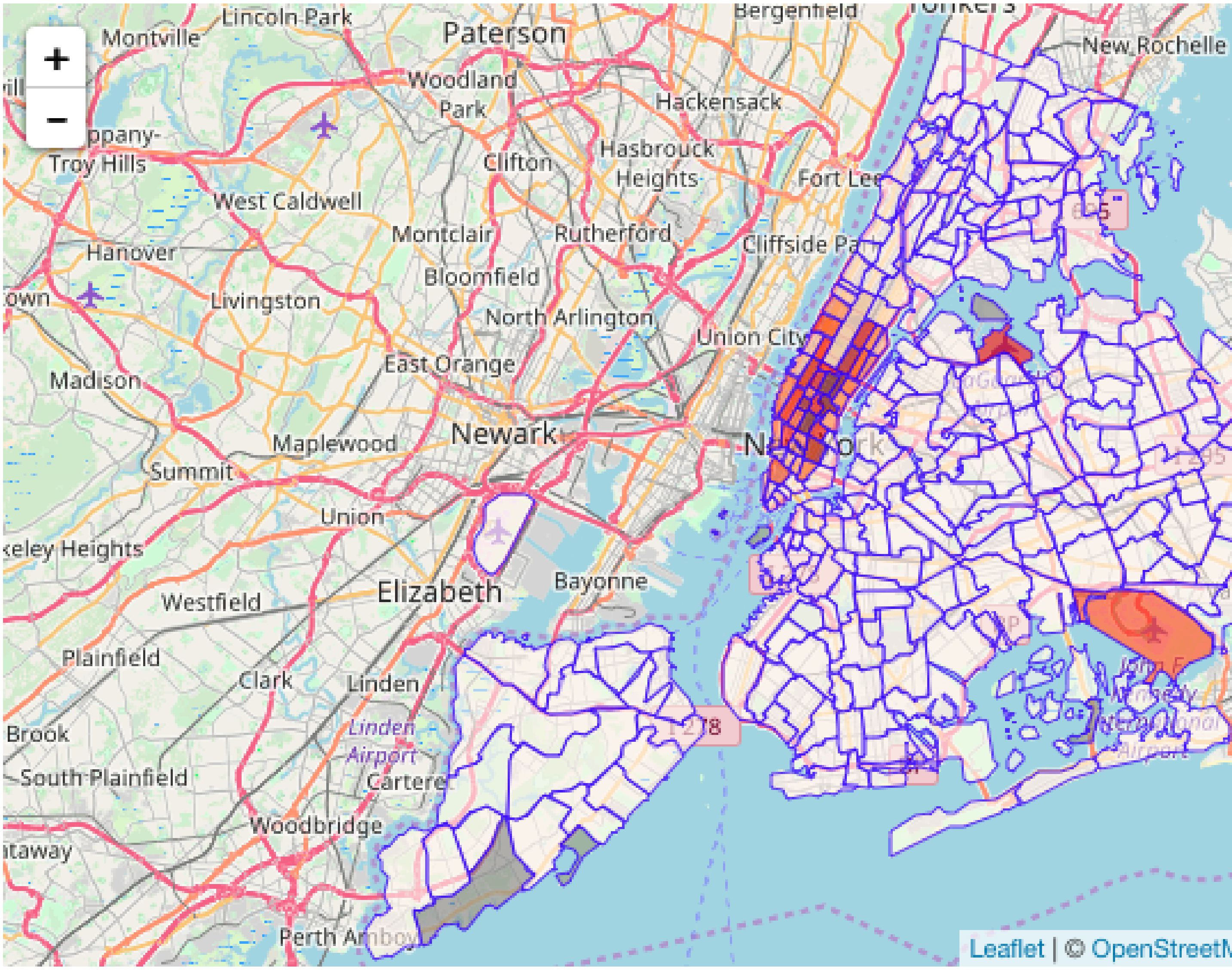
```
pick_up_zones <- merge(taxi_zones, tip_pickup, by.x = "LocationID",
    by.y = "LocationID")
```

We can create a heat map to visulizae the number of trip for each pick-up zones on a map of New York City Taxi Zones.

```r
library(leaflet)

library(webshot)
```

```
Warning: package 'webshot' was built under R version 3.4.3
```

```r
reds = colorNumeric("Reds", domain = NULL)

# create visulization leaflet(data = pick_up_zones)

# %>% addTiles() %>% addPolygons(fillColor =

# ~reds(num_trips), fillOpacity = 0.6, weight = 1,

# opacity = 0.8) %>% setView(lat = 40.7128, lng =

# -74.0060, zoom = 10) library(htmlwidgets)

# saveWidget(m, file =

# '/Users/priscilla/Desktop/Honors

# Thesis/writing/figures/temp.html') URL <-

# '/Users/priscilla/Desktop/Honors

# Thesis/writing/figures/temp.html' webshot(URL,

# cliprect = 'viewport')
```



It's obivous that Manhattan, La Guardia Airport, and JKF Airport have the most number of pick-ups.

Most yellow cab pick-ups occur in Manhattan. If we focus on the pick-up zones that have more than 900 trips per month or 30 trips per day, then we observe that many pick-up zones that have the highest percent tips are in Brooklyn.

```r
# pick a threshold for the cutoff number of trips
pickup_zone_900 <- tip_pickup %>% filter(num_trips >=
    900) %>% arrange(desc(avg_tip))
```

Table 3.2: Ten taxi pick-up zones with the highest percent tip with threshold equals to 900

LocationID	avg_tip	num_trips	avg_dis	Borough	Zone
106	0.2343283	1088	3.600248	Brooklyn	Gowanus
223	0.2335789	3014	4.212296	Queens	Steinway
37	0.2318139	2309	3.251091	Brooklyn	Bushwick South
80	0.2290748	4547	3.212947	Brooklyn	East Williamsburg
189	0.2286701	1842	3.361070	Brooklyn	Prospect Heights
112	0.2278641	3709	3.335044	Brooklyn	Greenpoint
7	0.2277946	7277	3.397217	Queens	Astoria
40	0.2275125	3537	4.186876	Brooklyn	Carroll Gardens
36	0.2271561	1334	3.571627	Brooklyn	Bushwick North
230	0.2267445	171017	3.044212	Manhattan	Times Sq/Theatre District

```
# pickup_zone_900 %>% head(10)
kable(pickup_zone_900[1:10, ], caption = "Ten taxi pick-up zones with the highest
```

People might think it is more reasonable to ses a list that is populated with Zones in Manhattan, since that's where all the wealthy people live. However, it turns out that passengers who get on taxis in Brooklyn pays more tips.

If we focus on the pick-up zones that have more than 90000 trips per month or 3000 trips per day, then we observe that all pick-up zones that have the highest percent tips are in Manhattan besides La Guardia Airport.

```
# pick a threshold for the cutoff of number of
# trips
pickup_zone_90000 <- tip_pickup %>% filter(num_trips >=
    90000) %>% arrange(desc(avg_tip))
# pickup_zone_90000 %>% head(10)
kable(pickup_zone_90000[1:10, ], caption = "Ten taxi pick-up zones with the highes
```

There are more than 100 times more yellow cab pick-ups that happen in Manhattan

Table 3.3: Ten taxi pick-up zones with the highest percent tip with threshold equals to 90000

LocationID	avg_tip	num_trips	avg_dis	Borough	Zone
230	0.2267445	171017	3.044212	Manhattan	Times Sq/Theatre District
186	0.2249650	213759	2.399181	Manhattan	Penn Station/Madison Sq West
138	0.2249391	177262	10.084311	Queens	LaGuardia Airport
161	0.2245180	230968	2.533839	Manhattan	Midtown Center
100	0.2245116	115242	2.467806	Manhattan	Garment District
162	0.2237261	224543	2.578228	Manhattan	Midtown East
237	0.2226464	193035	1.942023	Manhattan	Upper East Side South
48	0.2226313	180209	2.576908	Manhattan	Clinton East
239	0.2224459	134925	2.454595	Manhattan	Upper West Side South
163	0.2218928	152459	2.556783	Manhattan	Midtown North

everyday than in Brooklyn, and that is why there are many dense red-shade polygons in the visulization above.

3.4 Do taxi drivers tend to go to zones that offer high tips?

So far, we have learned what pick-up zones offer the highest percent tip. Now, we want to dig into the relationships between percent tip and taxi-zone-specific variables.

It is not easy to find an available taxi on the street on New York City, because the demand for taxi trips is much higher than the supply. Does paying more tips help customers to more easily get taxis? If customers from certain regions keep paying higher tips, taxi drivers might be able to learn from their experiences in those regions, and be more willing to wonder around those regions more often and pick up passengers. Pick-up zones with higher tips should attract more taxi drivers with the control of taxi zones. Let's test it out and see whether it is true:

```
tip_region$LocationID <- as.character(tip_region$LocationID)
tip_pickup$LocationID <- as.character(tip_pickup$LocationID)
tip_and_trip_1 <- lm(trips ~ avg_tip + LocationID,
    data = tip_region)
# summary(tip_and_trip_1)
summary(tip_and_trip_1)$coef[1:2, ]
```

```
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -188.5729   598.5170 -0.3150669 7.527139e-01
avg_tip     1447.2714   116.1562 12.4596964 1.629761e-35
```

Each one percent increase in average tips in pick-up zones is associated with 1447.2714 increase in the number of trips per month, controlling the pick-up zone.

```
9942263/31
```

```
[1] 320718.2
```

```
1447.2714/31
```

```
[1] 46.68617
```

In August 2016, yellow cabs made an average of 320,718 daily trips. Additionally, each one percent increase in average tips in pick-up zones is associated with 47 increase in the number of trips per day in a specific pick-up zone.

3.4.1 Which pick-up zone has the highest price per minute?

New York City Taxi Fare & Limousine Commission has information on how New York City taxi fare amount is calculated on their official website.

Metered Fare Information Onscreen rate is 'Rate #01 – Standard City Rate.' *The initial charge is $2.50.* Plus 50 cents per 1/5 mile or 50 cents per 60 seconds in slow traffic or when the vehicle is stopped. *In moving traffic on Manhattan streets, the meter should "click" approximately every four downtown blocks, or one block going cross-town (East-West).* There is a 50-cent MTA State Surcharge for all trips that end in New York City or Nassau, Suffolk, Westchester, Rockland, Dutchess, Orange or Putnam Counties. *There is a 30-cent Improvement Surcharge.* There is a daily 50-cent surcharge from 8pm to 6am. *There is a $1 surcharge from 4pm to 8pm on weekdays, excluding holidays.* Passengers must pay all bridge and tunnel tolls. *Your receipt will show your total fare including tolls. Please take your receipt.* The driver is not required to accept bills over $20. *Please tip your driver for safety and good service.* There are no charges for extra passengers or bags.

In taxi fare calculation, the only unknown variable is slow-trafice time, and all other variables were collected by the meters installed on each medallion taxi for each trip. It is reasonable to assume that for trips with the same pick-up and drop-off locations, the longer the total slow traffic time is, the longer the trip would take. Taxi drivers are compensated for both the normal-speed trip distance and the time spent in slow-traffice. According to the fare calculation algorithm, in moving traffic on Manhattan streets, the meter should "click" approximately every four downtown blocks, or one block going cross-town (East-West); in slow traffic, the meter should "click" every 60 seconds. Therefore, slow traffic reduces the fare per minute ratio.

New York CIty has the worst traffic jam, and it has overtaken Miami to be voted the U.S. city with the angriest and most aggressive drivers in 2009, according to a survey on road rage released on Tuesday. Bad traffic also cause slow-traffic, and taxi drivers tend to suck in traffic during rush hours. Does fare per minute ratio have an impact on the percent tip that passengers pay? Do passengers compensate taxi drivers more during rush hours? Are passengers sympathetic to taxi drivers for the time they spend in slow traffic?

```
library(lubridate)
yellow_2016.08_time <- yellow_2016.08_tip %>% mutate(duration = round((tpep_dropof
    tpep_pickup_datetime)/60, 2)) %>% mutate(duration = as.numeric(duration)) %>%
    filter(duration > 0) %>% mutate(fare_per_min = fare_amount/duration)

# summary(yellow_2016.08_time$fare_per_min)
fare_min_ratio <- lm(tip_perct ~ fare_per_min, data = yellow_2016.08_time)
summary(fare_min_ratio)
```

```

Call:
lm(formula = tip_perct ~ fare_per_min, data = yellow_2016.08_time)

Residuals:
     Min       1Q   Median       3Q      Max
-0.21882 -0.01891  0.00242  0.02685  0.77849

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.189e-01  2.815e-05 7776.48   <2e-16 ***
fare_per_min -1.785e-05  6.984e-07  -25.56   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06937 on 6088066 degrees of freedom
```

```
Multiple R-squared:  0.0001073, Adjusted R-squared:  0.0001071
F-statistic: 653.2 on 1 and 6088066 DF,  p-value: < 2.2e-16
```

As shown in the regression result, fare per minute ratio has a significant negative impact on percent tip. Since having more slow traffic time spent on the road reduces the fare per minute ratio, slow traffice does increase the impact on percent tip. Passengers do pay more tips to taxi drivers during rush hours.

Chapter 4

New York City Taxi Consumer

4.1 How long does it take for passengers to get to JFK, La Guardia, and Newark Airports? When is the best time to travel?

We want to calculate the average number of minutes it takes to go to all three airport from a specific taxi zone at every hour. First, we want to focus on trips going to any of the three airports, JFK, LaGuardia, or Newark Airport.

```
library(dplyr)
library(readr)
library(lubridate)

to_jfk_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    132) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
```

```
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime))
```

```
Warning in as.POSIXlt.POSIXct(x): unknown timezone 'zone/tz/2018c.1.0/
zoneinfo/America/New_York'
```

```
to_lg_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    138) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime))

to_newark_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    1) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime))
```

Now we need to calculate the average amount of time it take for each zone at during each hour.

```
to_jfk_zone <- to_jfk_trip %>% group_by(PULocationID,
    hour) %>% summarise(jfk_avg_min = mean(min)) %>%
    arrange(PULocationID)

to_lg_zone <- to_lg_trip %>% group_by(PULocationID,
    hour) %>% summarise(lg_avg_min = mean(min)) %>%
```

```
    arrange(PULocationID)


to_newark_zone <- to_newark_trip %>% group_by(PULocationID,
    hour) %>% summarise(newark_avg_min = mean(min)) %>%
    arrange(PULocationID)
```

So far, we have created three tables summarsing the average number of minutes it takes to go to all three airports for every hour from different taxi zones.

```
head(to_jfk_zone)
```

```
# A tibble: 6 x 3
# Groups:   PULocationID [3]
  PULocationID  hour jfk_avg_min
         <int> <int>       <dbl>
1            1    14       59.32
2            1    19       68.58
3            3    14       54.15
4            3    19       37.82
5            4     2       28.52
6            4     3       25.98
```

It would be easier if we combine all three tables and put all information related to trip duration to airports into one table.

```
two_transp <- full_join(to_jfk_zone, to_lg_zone, by = c("PULocationID",
    "hour"))
three_transp <- full_join(two_transp, to_newark_zone,
    by = c("PULocationID", "hour"))
```

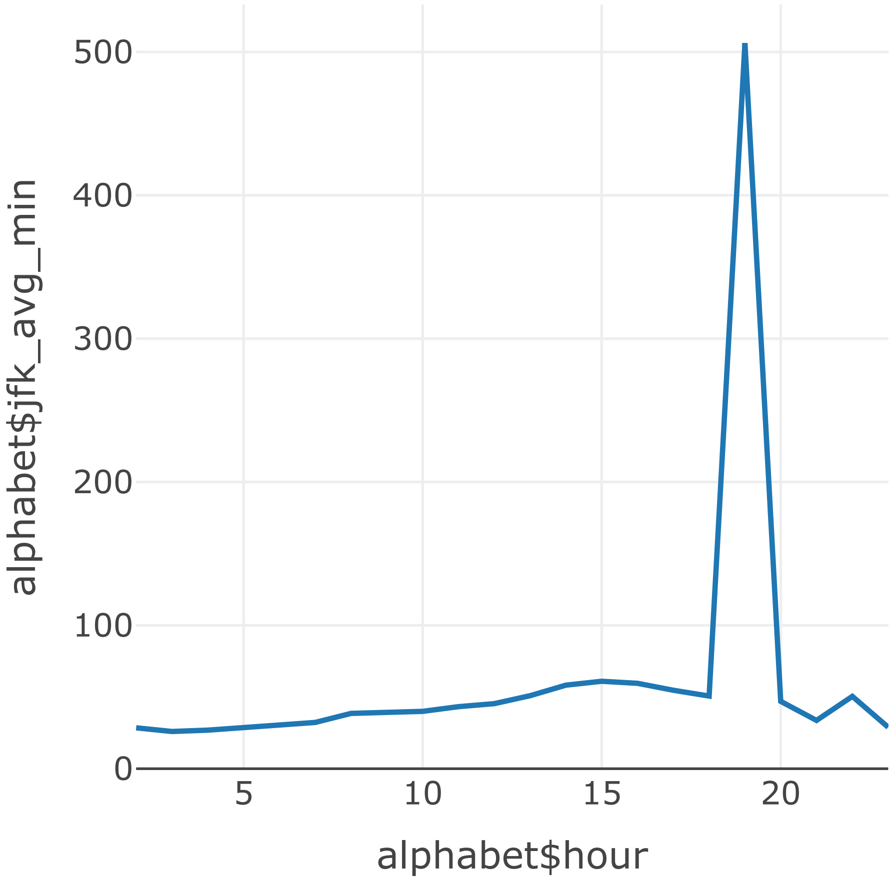
```
three_transp %>% filter(PULocationID == 4) %>% head(5)
```

```
# A tibble: 5 x 5
# Groups:   PULocationID [1]
  PULocationID  hour jfk_avg_min lg_avg_min newark_avg_min
         <int> <int>       <dbl>      <dbl>          <dbl>
1            4     2    28.52000   16.89000         32.830
2            4     3    25.98000   19.40667         44.830
3            4     4    26.90800   23.36909         32.450
4            4     5    28.95812   18.01286         28.238
5            4     6    29.88083   21.97208         27.415
```

From Alphabet City, Manhattan to all three airport Alphabet City, Manhattan has pick-up zone ID number 4. Let's take a look at how much time is needed to travel to all three airports from taxi zone No.4.

```
alphabet <- three_transp %>% filter(PULocationID ==
    4)
```

```
library(plotly)
jfk <- plot_ly(x = ~alphabet$hour, y = ~alphabet$jfk_avg_min,
    mode = "lines")
jfk
```



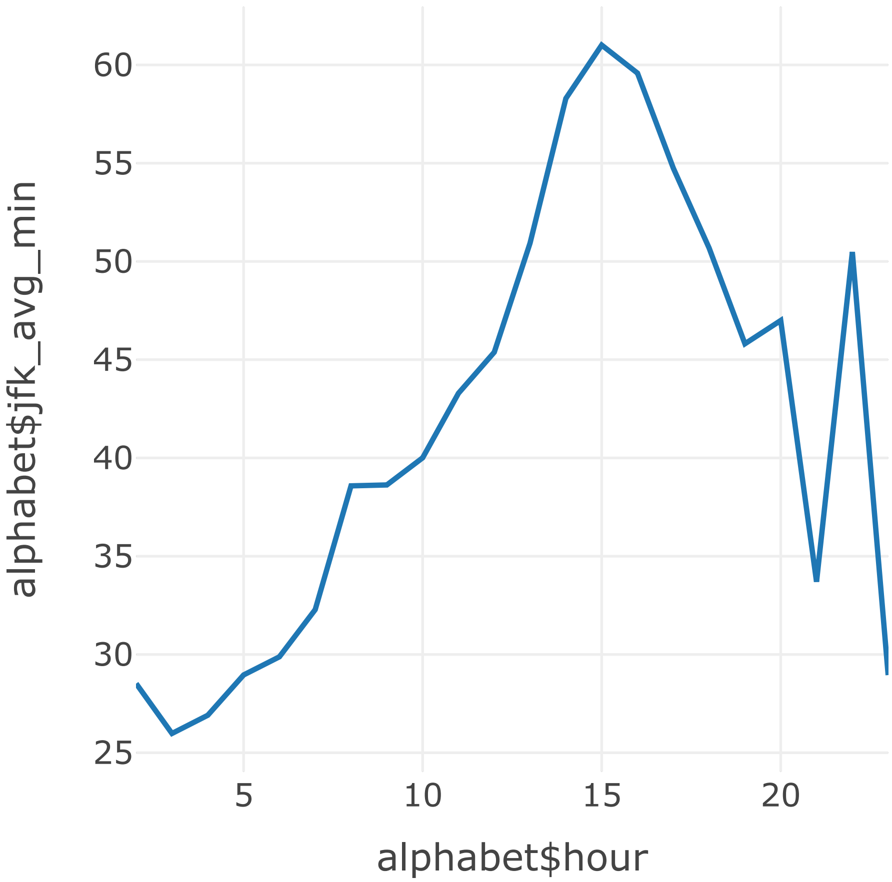
As shown in the graph, there is a huge peak at 7pm, and it takes on average 506 minutes to go to JFK from Alphabet City if you depart from 6-7pm. This is caused by a huge outlier in the original dataset.

Let's filter out trips that took less than one minute and trips that took more than 5 hours out of the dataset, and then regenerate the visulization.

```
to_jfk_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    132) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime)) %>%
    filter(min <= 300) %>% filter(min > 1)
```

```
to_lg_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    138) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime)) %>%
    filter(min <= 300) %>% filter(min > 1)

to_newark_trip <- yellow_2016.08_cleaned %>% filter(DOLocationID ==
    1) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(duration = tpep_dropoff_datetime -
    tpep_pickup_datetime) %>% mutate(min = round(as.numeric(duration)/60,
    2)) %>% mutate(hour = hour(tpep_pickup_datetime)) %>%
    filter(min <= 300) %>% filter(min > 1)
```

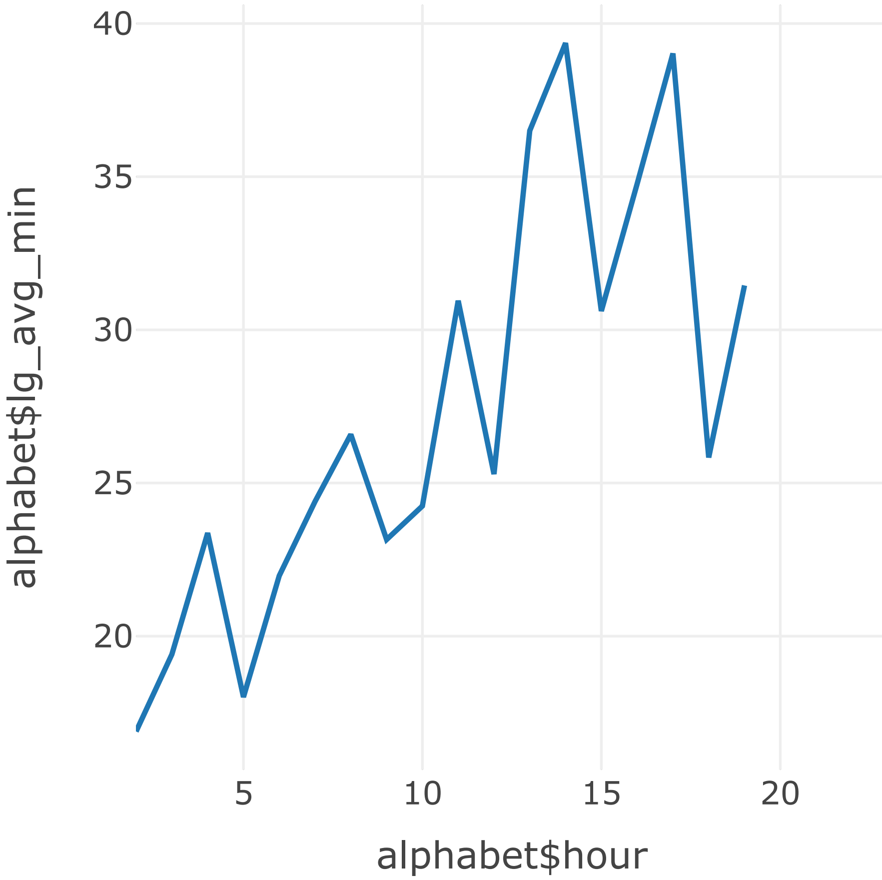
```
jfk <- plot_ly(x = ~alphabet$hour, y = ~alphabet$jfk_avg_min,
    mode = "lines")
jfk
```



According to the plot above, it takes the least time to travel to JFK Airport around 2am in the morning, and it takes the most time around 3pm in the afternoon.

```
lg <- plot_ly(x = ~alphabet$hour, y = ~alphabet$lg_avg_min,
    mode = "lines")
lg
```

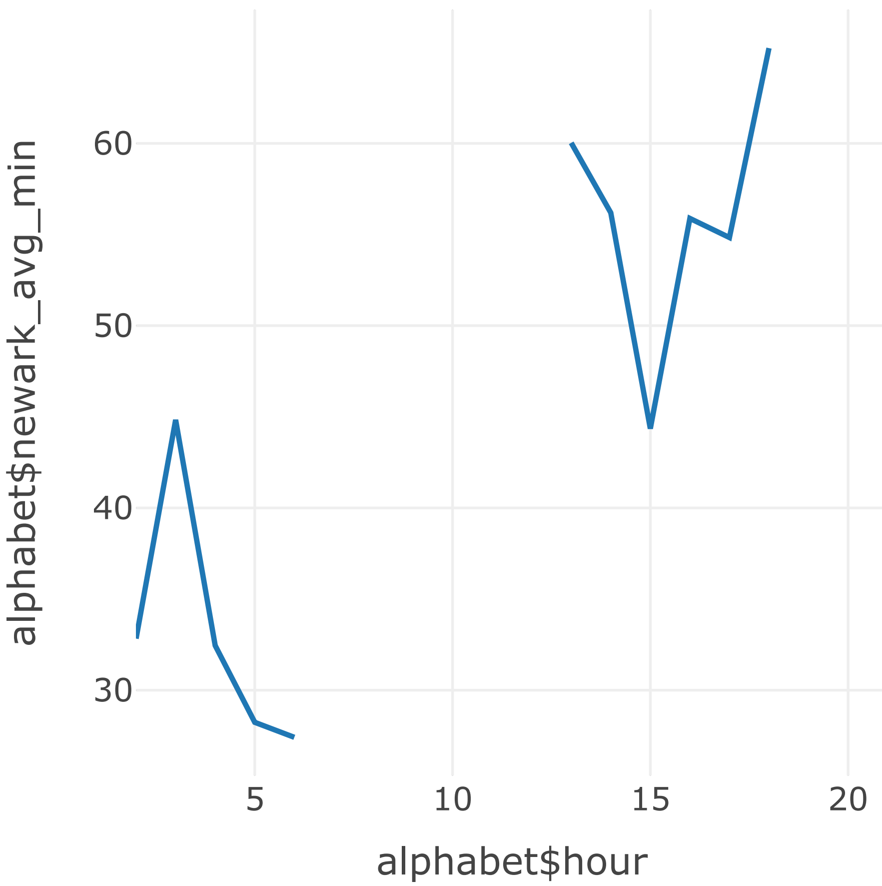
```
No trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#scatter
```



According to the plot above, it takes the least time to travel to La Guardia Airport around 1am in the morning, and it takes the most time around 2pm in the afternoon.

```
newark <- plot_ly(x = ~alphabet$hour, y = ~alphabet$newark_avg_min,
    mode = "lines")
newark
```

```
No trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#scatter
```



As shown in the plot, not a lot of passengers take yellow taxi to travel to Newark Airport. Therefore, we are only able to capture the average amount of time it takes to travel to Newark for certain hours of departure. It takes the least time to travel to Newark at 6am in the morning, and it takes the most time around 6pm in the evening.

A Shiny app will be added here. This app will allow users to choose a pick up zone of their interest, and output the best time to travel from that zone to all three airports in New York.

4.2 How does weather affect the number of taxi and Lyft trips?

According to Schneider (2015), there are lots of confounding variables between weather and taxi rides, including seasonal trends, annual growth due to boro taxis, and whether weather events happen to fall on weekdays or weekends, but it would appear that snowfall has a significant negative impact on daily taxi ridership. In this section, we use New York City Yellow Taxi and Lyft data to calculate the number of trips occurred on and before snow days.

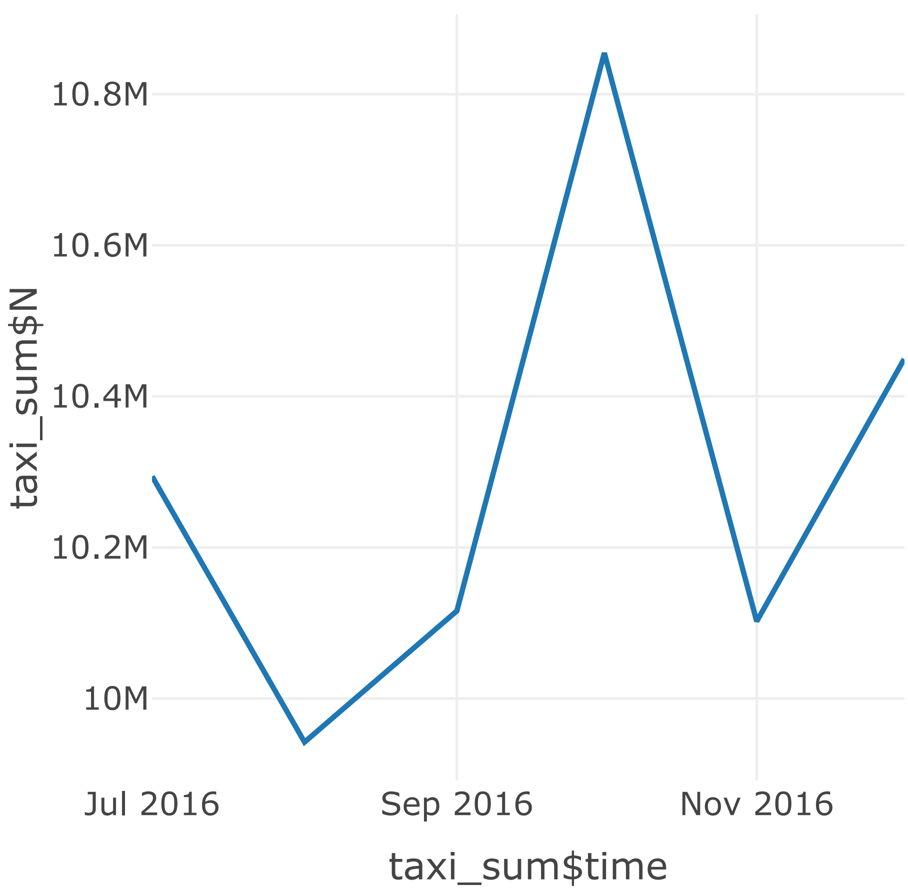
```
taxi_summary_2017 <- taxi %>% tbl("yellow") %>% mutate(year = year(tpep_pickup_datetime),
    month = month(tpep_pickup_datetime)) %>% group_by(year,
    month) %>% summarise(N = n()) %>% collect()
```

```
Warning in .local(conn, statement, ...): Decimal MySQL column 10 imported
as numeric

Warning in .local(conn, statement, ...): Decimal MySQL column 11 imported
as numeric

Warning in .local(conn, statement, ...): Decimal MySQL column 12 imported
as numeric

Warning in .local(conn, statement, ...): Decimal MySQL column 13 imported
as numeric

Warning in .local(conn, statement, ...): Decimal MySQL column 14 imported
as numeric

Warning in .local(conn, statement, ...): Decimal MySQL column 15 imported
as numeric
```

```
Warning in .local(conn, statement, ...): Decimal MySQL column 16 imported
as numeric
```

```
taxi_sum <- taxi_summary_2017 %>% mutate(year_month = paste0(year,
    "-", stringr::str_pad(month, 2, "left", "0"), "-01")) %>%
    mutate(time = as.POSIXct(year_month))

library(plotly)
p <- plot_ly(x = ~taxi_sum$time, y = ~taxi_sum$N, mode = "lines")
p
```

```
No trace type specified:
  Based on info supplied, a 'scatter' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#scatter
```



I downloaded daily Central Park weather data from the National Climatic Data Center, and joined it to the taxi data to see if we could learn anything else about the relationship between weather and taxi rides.

More analysis needed here

Chapter 5

New York City Taxi Fare & Limousine Commission

5.1 Should there be a flat rate between Manhattan and John F. Kennedy International Airport?

Why is there a flat rate to and from JFK airport and any location in Manhattan? Why is the flat rate $52? Does TLC make profit from the $52 flat rate? Does $52 reduce the cogestion on the road to JFK airport and make taking a train a more preferable choice? The New York City taxi trip records can reveal the answers to these questions.

Imagine it's your first time travelling to New York City, and you decided to live in a hotel in Manhattan Since you do not know much about the city, the $52 flat rate is nice for you, and it incentivizes you to take taxi to the JFK Airport. If there is no flat rate, there is uncertainty in how much someone needs to pay to take a taxi to

JFK, and tourists might instead choose to take the train, even though taking a train would cost them more time and inconvenience.

Additionally, people who are native to Manhattan would have paid more than \$52 to take a taxi to go to the JFK Airport. The higher the taxi fare is, the less the demand for taxi will be. Therefore, having a flat rate,helps taxi drivers to get more trips from Manhattan to JFK Airport.

5.1.1 People in Manhattan benefit from the \$52 flat rate.

If there is no flat rate between JFK and Manhattan, how much would passengers pay for the distance they travelled between JFK Airport AND Manhattan? And how much more or less should they have paid comparing to the \$52 flat rate?

In this study, we are only interested in yellow taxi trip between Manhattan and JFK Airport.

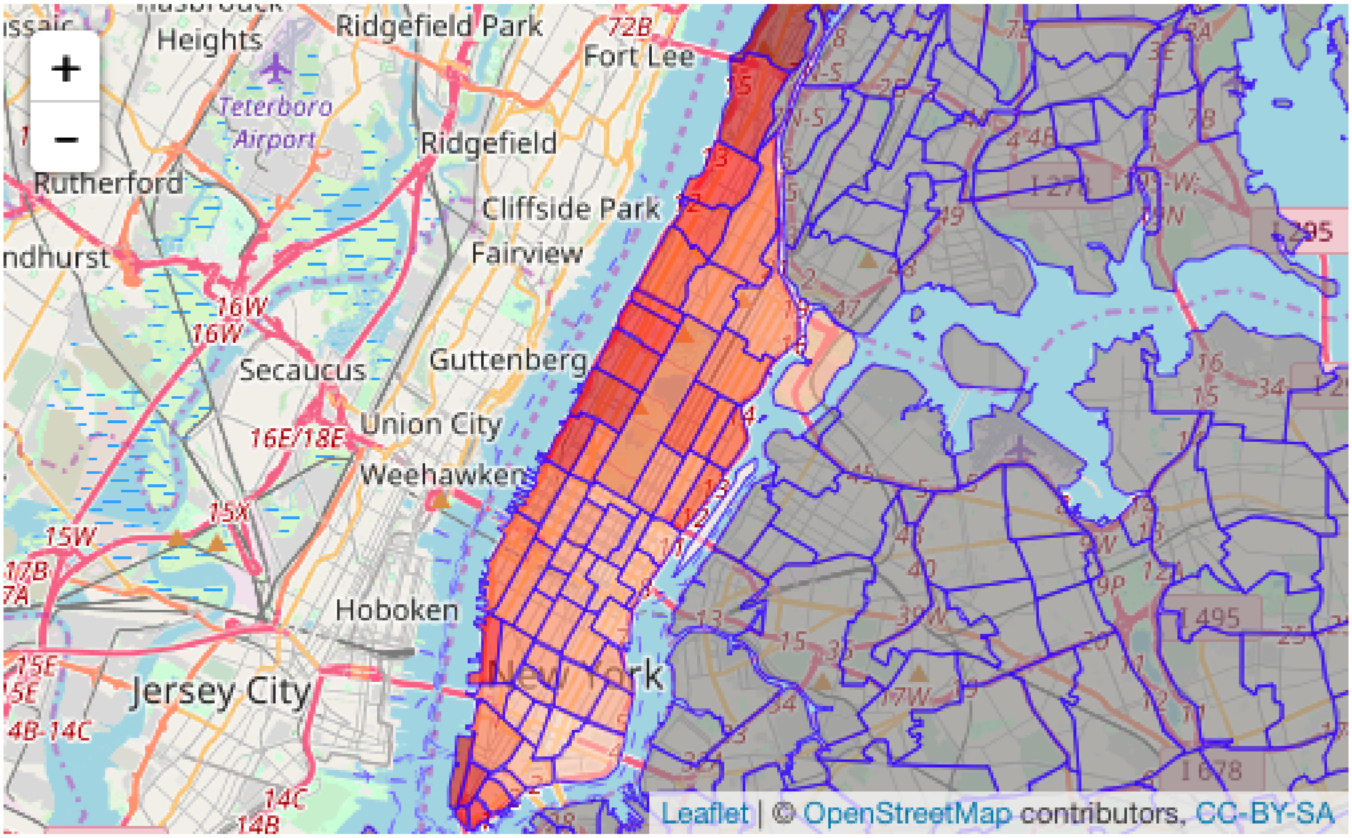
```
# jfk_trip <- taxi %>% tbl('yellow') %>%
# filter(RatecodeID == 2) %>% collect()


jfk_trip <- yellow_2016.08_cleaned %>% filter(RatecodeID ==
    2) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID)
```

Trips from Manhattan to JFK Airport We first focus on all the trips that departed in Manhattan and went to JFK Airport, and then we calculate the estmated fare amount that the passengers should have paid based on the distance travelled from each pick-up point to JFK Airport based on the fare rate suggested by TLC for each pick-up zone.

```r
data("taxi_zone_lookup")

to_jfk <- jfk_trip %>% filter(DOLocationID == 132) %>%
    mutate(est_fare = 2.5 + 0.5 * trip_distance * 5 +
        extra + improvement_surcharge + mta_tax + tolls_amount) %>%
    mutate(est_diff = est_fare - fare_amount) %>% rename(LocationID = PULocationID
    left_join(taxi_zone_lookup, by = "LocationID") %>%
    filter(Borough == "Manhattan")

to_jkf_zone <- to_jfk %>% group_by(LocationID) %>%
    summarise(num_trips = n(), avg_est_fare = mean(est_fare),
        avg_est_diff = mean(est_diff)) %>% left_join(taxi_zone_lookup,
    by = "LocationID")
```

Here is a map of estmated fare amount calculated by taking the average of all estimated fare amounts from the same pick-up zone to JFK Airport based on the fare rate suggested by TLC for each pick-up zone.

```r
to_jkf_fare <- merge(taxi_zones, to_jkf_zone, by.x = "LocationID",
    by.y = "LocationID")

# leaflet(data = to_jkf_fare) %>% addTiles() %>%
# addPolygons(fillColor = ~reds(avg_est_fare),
# fillOpacity = 0.6, weight = 1, opacity = 0.8) %>%
# setView(lat = 40.7128, lng = -74.0060, zoom = 10)
```



According to the map, trips from Midtown on average cost less than trips from other taxi zones in Manhattan.

```
to_jkf_zone_above <- to_jkf_zone %>% filter(avg_est_fare >
    52) %>% arrange(desc(avg_est_fare))


kable(to_jkf_zone_above[1:10, ], caption = "Ten pick-up zones with the highest avergae f
```

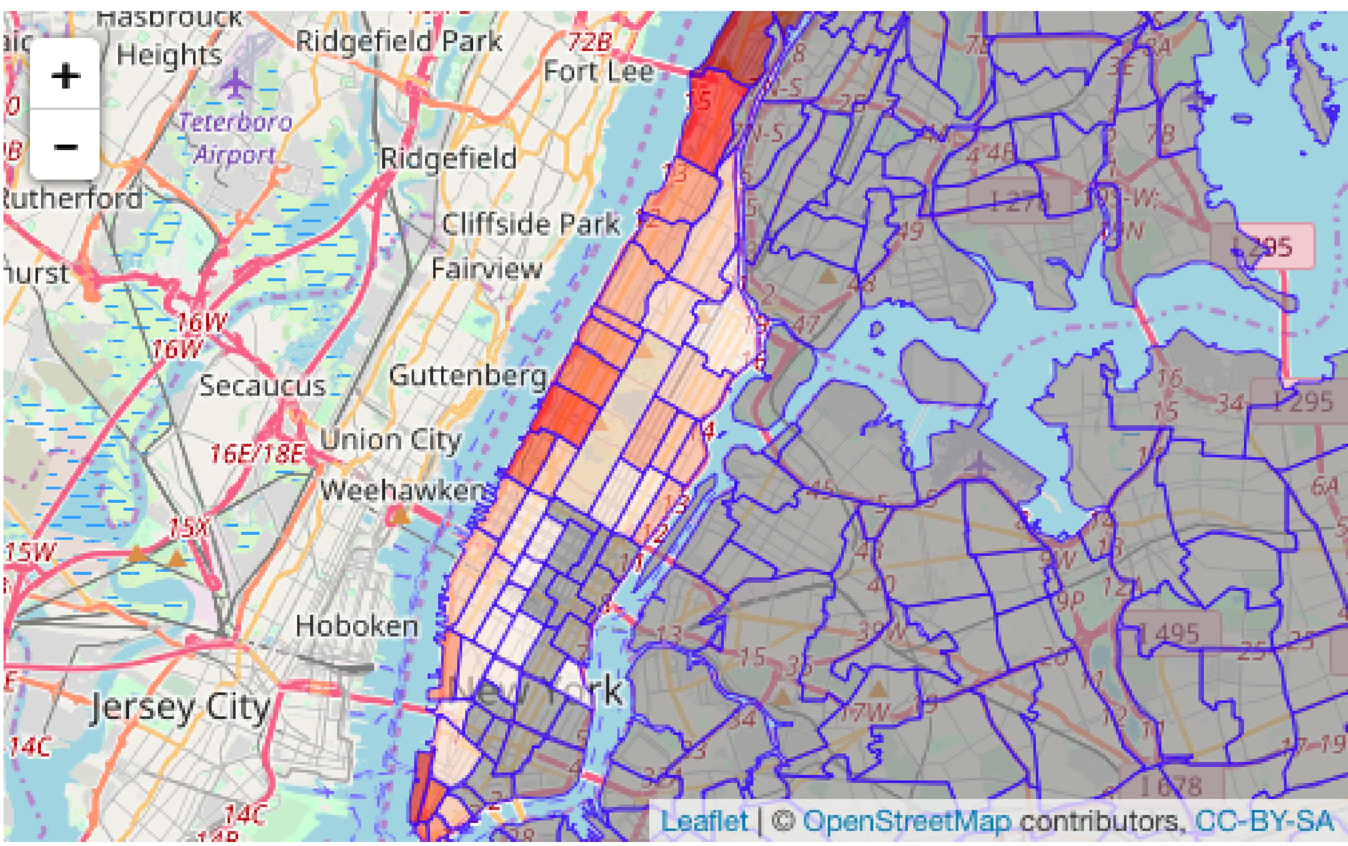
Let's visualize the taxi zones that would have costed more than the $52 flat rate.

```
to_jkf_fare_above <- merge(taxi_zones, to_jkf_zone_above,
    by.x = "LocationID", by.y = "LocationID")


# leaflet(data = to_jkf_fare_above) %>% addTiles()
# %>% addPolygons(fillColor = ~reds(avg_est_fare),
```

Table 5.1: Ten pick-up zones with the highest avergae fare from Manhattan to JKF Airport

LocationID	num_trips	avg_est_fare	avg_est_diff	Borough	Zone
127	6	65.06250	13.062500	Manhattan	Inwood
243	8	63.82125	11.821250	Manhattan	Washington Heights North
13	1266	62.44055	10.440545	Manhattan	Battery Park City
244	74	60.27892	8.278919	Manhattan	Washington Heights South
239	1685	60.11053	8.110525	Manhattan	Upper West Side South
261	723	59.59012	7.590118	Manhattan	World Trade Center
143	655	59.26524	7.265244	Manhattan	Lincoln Square West
238	1233	59.02019	7.020191	Manhattan	Upper West Side North
12	41	58.44146	6.441463	Manhattan	Battery Park
88	362	57.79963	5.799627	Manhattan	Financial District South

```
# fillOpacity = 0.6, weight = 1, opacity = 0.8) %>%

# setView(lat = 40.7128, lng = -74.0060, zoom = 10)



# add t test to test average fare vs 52
```



Therefore, passengers from places in Manhattan besides Midtown, East Village, and some parts of Lower Manhattan benefit from the $52 flat rate. However, people living in Midtown, East Village, and some parts of Lower Manhattan might be relatively more indifferent to the price of taxi. Instead, they probably put more emphasis on convenience and time.

```
mean(to_jkf_zone$avg_est_diff)
```

```
[1] 2.138185
```

On average people travel from Manhattan pay $2.14 less with the $52 flat rate policy.

5.2 However, are taxi drivers happy when their passengers are going to JFK Airport from Manhattan?

Everytime I travel to New York City, I always take Yellow cabs to go around the city. It seemed to me that the cab drivers were always happy whenever they heard me telling them that I need to go to the JFK Airport from Manhattan. Are taxi drivers happy when their passengers are going to JFK Airport from Manhattan? How much on average would taxi driver make on their way bavk to the city from Manhattan?

```
from_jfk <- yellow_2016.08_cleaned %>% filter(PULocationID ==
    132) %>% filter(payment_type != 3) %>% filter(trip_distance >
    0) %>% filter(fare_amount > 0) %>% filter(PULocationID !=
    DOLocationID) %>% mutate(est_fare = 2.5 + 0.5 *
```

```
    trip_distance * 5 + extra + improvement_surcharge +
    mta_tax + tolls_amount) %>% mutate(est_diff = est_fare -
    fare_amount)
```

Since a taxi driver coming from Manhattan to JFK Airport could be directed back to anywhere in the city. We can calculate the average taxi fare amount that the taxi drivers would get paid for a random trip from JFK Airport to any part of the city.

```
mean(from_jfk$est_fare)
```

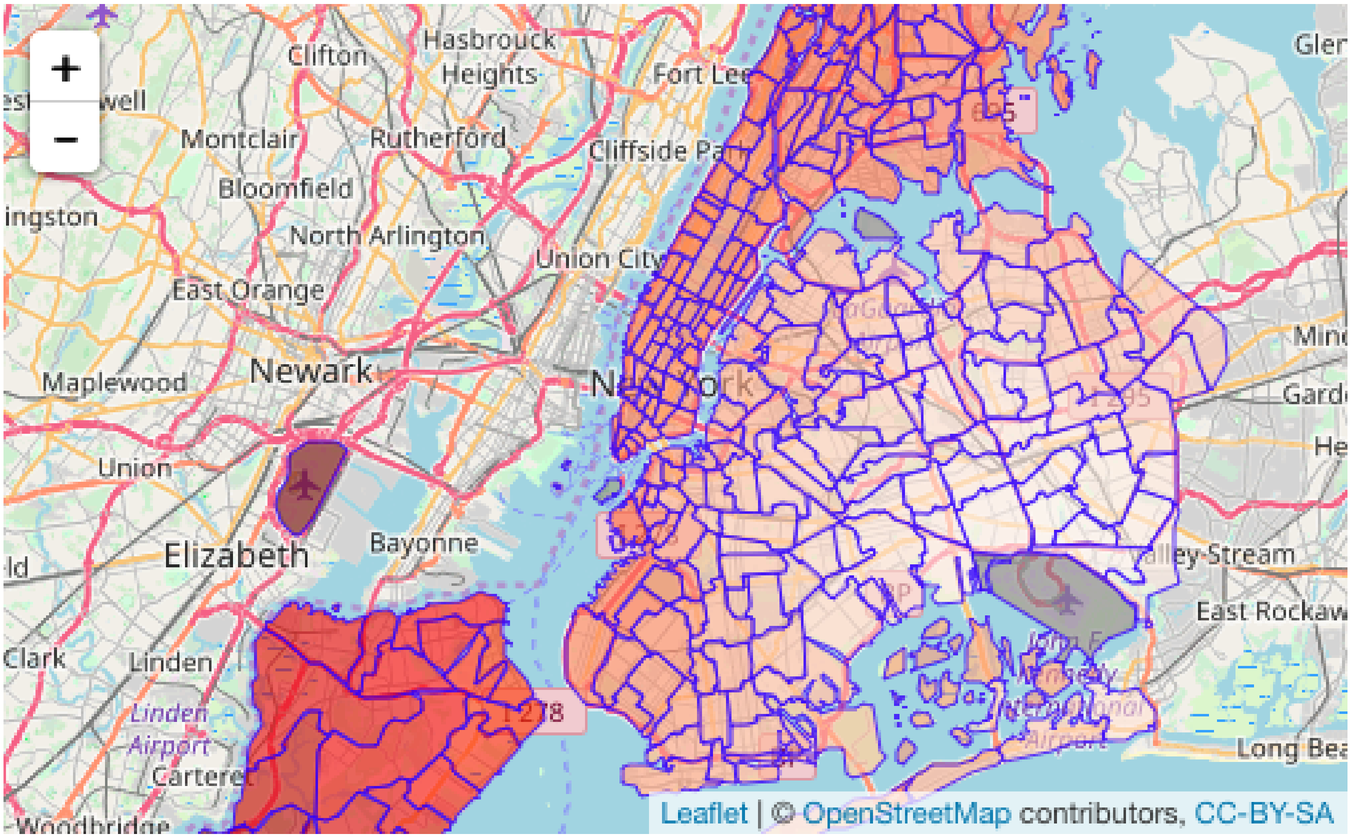
```
[1] 46.89639
```

On average, taxi drivers would be paid for $46.90 for a trip from the JFK Airport to any taxi zone. What are the most popular drop-off zones for yellow taxis from JFK Airport?

```
from_jkf_zone <- from_jfk %>% group_by(DOLocationID) %>%
    summarise(num_trips = n(), avg_est_fare = mean(est_fare),
        avg_est_diff = mean(est_diff)) %>% rename(LocationID = DOLocationID) %>%
    left_join(taxi_zone_lookup, by = "LocationID")
```

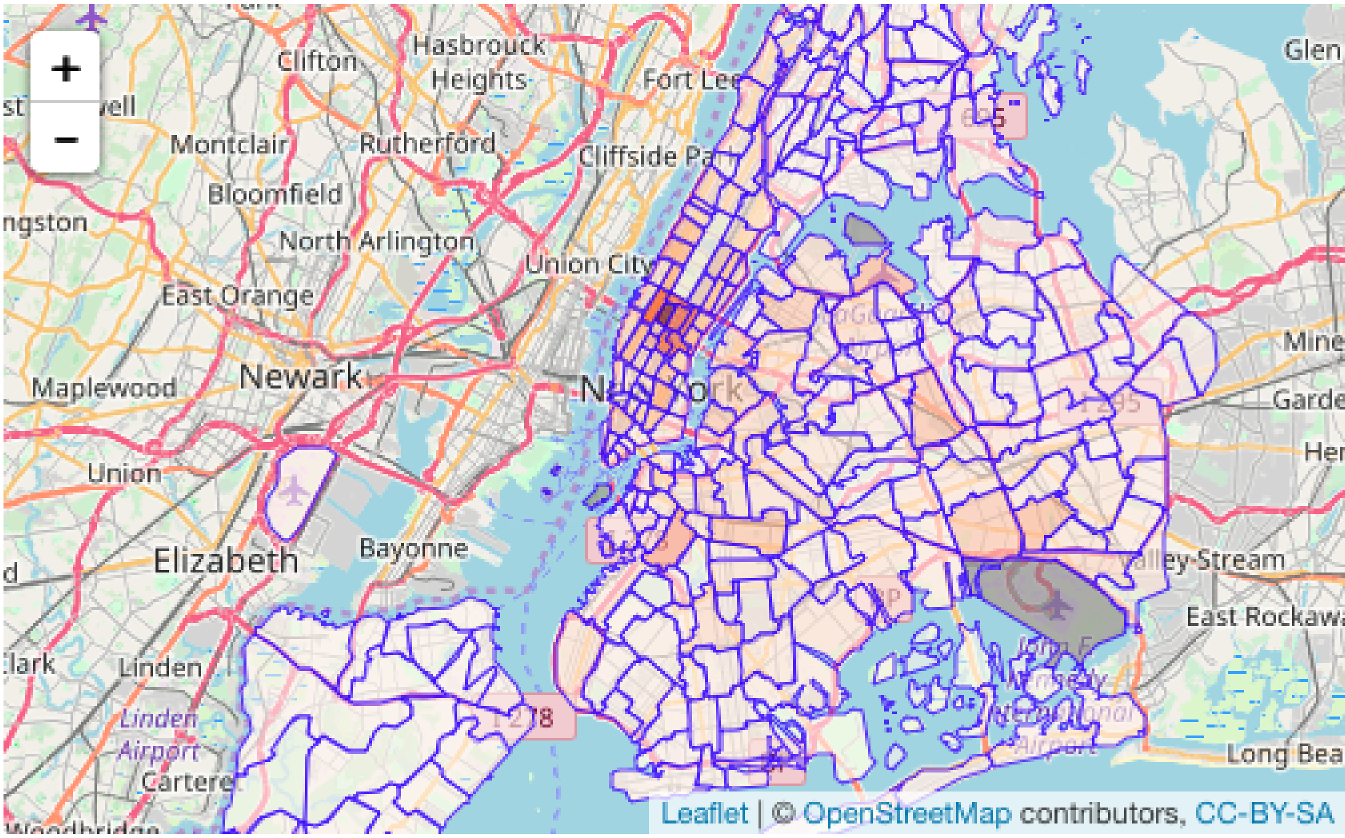
What's the average fare to each dropp-off zone from JFK Airport?

```
from_jkf_fare <- merge(taxi_zones, from_jkf_zone, by.x = "LocationID",
    by.y = "LocationID")


# leaflet(data = from_jkf_fare) %>% addTiles() %>%
# addPolygons(fillColor = ~reds(avg_est_fare),
# fillOpacity = 0.6, weight = 1, opacity = 0.8) %>%
# setView(lat = 40.7128, lng = -74.0060, zoom = 10)
```



As we expected, the red shades are smoothly distributed, since taxi zones that are futher away should cost more to get there.

```
# leaflet(data = from_jkf_fare) %>% addTiles() %>%
# addPolygons(fillColor = ~reds(num_trips),
# fillOpacity = 0.6, weight = 1, opacity = 0.8) %>%
# setView(lat = 40.7128, lng = -74.0060, zoom = 10)
```



According to the map above, Manhattan is still the most popular destination for passengers depart from the JFK Airport.

```
from_jkf_zone %>% mutate(Manhattan = ifelse(Borough ==
    "Manhattan", 1, 0)) %>% group_by(Manhattan) %>%
    summarise(all_trips = sum(num_trips))
```

```
# A tibble: 2 x 2
  Manhattan all_trips
      <dbl>     <int>
1         0    116248
2         1    131231
```

According to the summary, the total amount of trips from JFK Airport to Manhattan is 10% more than the total number of trips travelling from JFK Airport to all other

Borough. Therefore, it is very likely for taxi drivers to get passengers who want to go to Manhattan with a flat rate of $52. In this case, a round trip to and from JFK Airport is worthy. Therefore, taxi drivers should be pretty happy when their passengers are going to JFK Airport from Manhattan.

Chapter 6

Conclusion

6.1 Future Research

For future study, I would love to investigate the sharp decline in the consumption of NYC yellow cab after e-hail services were introduced into the NYC ride-hail market. I also want to study what the impact of introducing new GPS and entertainment system is on the number of rides. The global product and marketing at Verifone, Jason Gross, said that, “I like to say that we provide what Uber says it provides.” With the raised expectation among rides caused by Uber and Lyft, yellow taxi industry need to respond quickly. How does the market react to the newly installed entertainment system? Has the market share of yellow cab rebounded since 2016? By looking into the patterns in market shares, it might be possible for me to predict the future market share distribution and find out what features of ride-hail transportation are the ones that affect market share distribution the most.

Appendix A

The First Appendix

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the `include = FALSE` chunk tag) to help with readibility and/or setup.

In the main Rmd file

```
# This chunk ensures that the thesisdown package is
# installed and loaded. This thesisdown package
# includes the template files for the thesis.
if (!require(devtools)) install.packages("devtools",
    repos = "http://cran.rstudio.com")
if (!require(thesisdown)) devtools::install_github("ismayc/thesisdown")
library(thesisdown)
```

In Chapter ??:

Appendix B

The Second Appendix, for Fun

References

Angel, E. (2000). *Interactive computer graphics : A top-down approach with opengl.* Boston, MA: Addison Wesley Longman.

Angel, E. (2001a). *Batch-file computer graphics : A bottom-up approach with quicktime.* Boston, MA: Wesley Addison Longman.

Angel, E. (2001b). *Test second book by angel.* Boston, MA: Wesley Addison Longman.