Honors Project Paper

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)

_____

Advisor Benjamine Baumer

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class (for LaTeX) and the R bookdown package, in general.

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

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.

Dedication

You can have a dedication here if you wish.

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 trips.

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 Yellow Taxi

1.3.2 Green Taxi

1.3.3 Uber

1.3.4 Lyft

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

1.4.3 Modeling of NYC Taxi Fare

1.4.4 Reproducible Research

Chapter 2

Data and nyctaxi Package

2.1 Data and Storage

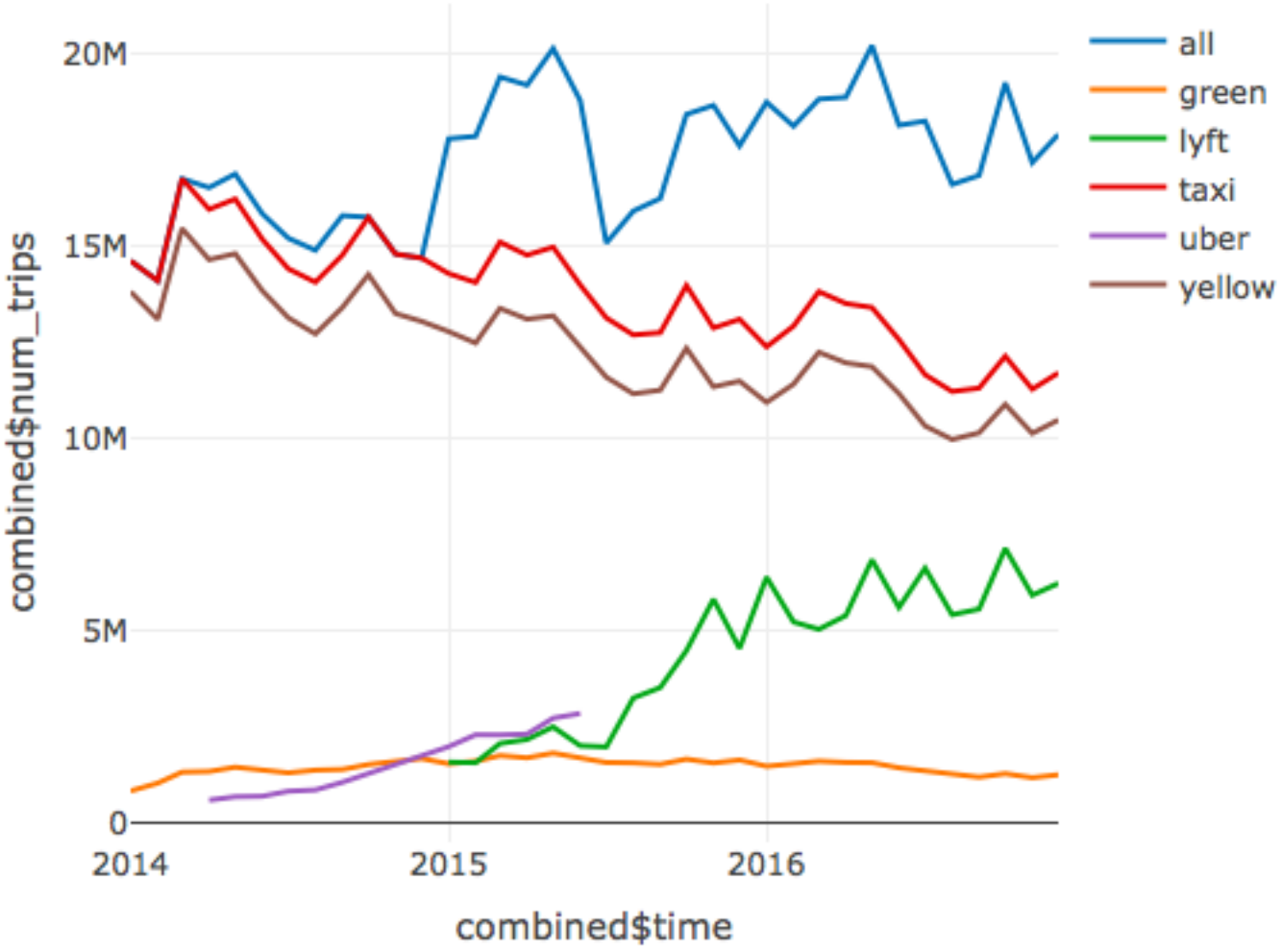


Figure 2.1

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 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.

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

# load the package
library(nyctaxi)
```

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

2.3 Extract

etl_extract.nyctaxi allows users to download New York City yellow taxi, green taxi, Uber, and Lyft data that are specific to their month of interest.

2.3.1 Yellow Taxi

2.3.2 Green Taxi

2.3.3 Uber

2.3.4 Lyft

2.4 Transform

etl_extract.nyctaxi allows users to transform New York City yellow taxi, green taxi, Uber, and Lyft data into forms that are meaningful to users.

2.4.1 Yellow Taxi

2.4.2 Green Taxi

2.4.3 Uber

2.4.4 Lyft

2.5 Load

etl_extract.nyctaxi allows users to load New York City yellow taxi, green taxi, Uber, and Lyft data into different data tables in a SQL database.

2.5.1 Yellow Taxi

2.5.2 Green Taxi

2.5.3 Uber

2.5.4 Lyft

2.6 SQL Database Initialization

2.6.1 Yellow Taxi

2.6.2 Green Taxi

2.6.3 Uber

2.6.4 Lyft

2.7 New Yoek City Hail Service Summary

2.8 Source Code

2.8.1 ETL Extract

```
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"), ".c
    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$src, ...),
           error = function(e){warning(e)},
           finally = warning("Only the following data are availabel on TLC:
                             Green taxi data: 2013 Aug - last month"))}
#UBER-------------------------------------------------------------------
uber <- function(obj, years, months,...) {
  message("Extracting raw uber data...")
  raw_month_2014 <- etl::valid_year_month(years = 2014, months = 4:9)
  raw_month_2015 <- etl::valid_year_month(years = 2015, months = 1:6)
  raw_month <- bind_rows(raw_month_2014, raw_month_2015)
  path = "https://raw.githubusercontent.com/fivethirtyeight/uber-tlc-foil-response/mas
  remote <- etl::valid_year_month(years, months)
  remote_small <- intersect(raw_month, remote)
  if (2015 %in% remote_small$year && !(2014 %in% remote_small$year)){
    #download 2015 data
    message("Downloading Uber 2015 data...")
    etl::smart_download(obj, "https://github.com/fivethirtyeight/uber-tlc-foil-respons
  else if (2015 %in% remote_small$year && 2014 %in% remote_small$year) {
    #download 2015 data
    message("Downloading Uber 2015 data...")
    etl::smart_download(obj, "https://github.com/fivethirtyeight/uber-tlc-foil-respons
    #download 2014 data
    small <- remote_small %>%
      filter_(~year == 2014) %>%
      mutate_(month_abb = ~tolower(month.abb[month]),
              src = ~file.path(path, paste0("uber-raw-data-",month_abb,substr(year,3,4
    message("Downloading Uber 2014 data...")
    etl::smart_download(obj, small$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
    etl::smart_download(obj, small$src,...)}
  else {warning("The Uber data you requested are not currently available. Only data fr
  }
#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)
}
```

2.8.2 ETL Transform

```
etl_transform.etl_nyctaxi <- function(obj, years = as.numeric(format(Sys.Date(),'
                                      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
    #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$src)
    src_small_green_2_df <- data.frame(src_small_green_2)
    names(src_small_green_2_df) <- "src"
    src_small_green_2_df <- inner_join(src_small_green_2_df, remote_green_2, by
    src_small_green_2_df <- src_small_green_2_df %>%
      mutate(cmds_2 = paste("cut -d, -f1-", keep," ",src, " > ",attr(obj, "raw_d
                            year, "_", stringr::str_pad(month, 2, "left", "0"),"
    #remove the extra column
    if(length(src_small_green_2) != 0) {
      if (.Platform$OS.type == "unix"){
        lapply(src_small_green_2_df$cmds_2, system)}
      else {
        message("Windows system does not currently support removing the 2nd blan
                in the green taxi datasets. This might affect loading data into
      }else {
        "All the green taxi data you requested are in cleaned formats."}
    #Find the files paths of the files that need to be transformed------------
    file.rename(file.path(dirname(src_small_green_2_df$src),
                          src_small_green_2_df$new_file),
                file.path(attr(obj, "load_dir"), basename(src_small_green_2_df$s
    #Move the files
    in_raw <- basename(src_small)
    in_load <- basename(list.files(attr(obj, "load_dir"), "green", full.names =
    file_remian <- setdiff(in_raw,in_load)
    file.copy(file.path(attr(obj, "raw_dir"),file_remian), file.path(attr(obj,
#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_d
  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,
           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.
  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"

  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.pa
      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 req
  }
#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)
}
```

2.8.3 ETL Load

```
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 =
      mutate_(src = ~file.path(attr(obj, "load_dir"),
                               paste0("yellow", "_tripdata_", year, "-",
                                      stringr::str_pad(month, 2, "left", "0"), ".c
    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
  } 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 i
  } 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$con, name = "uber",
                        value = uberfileURL, append = TRUE, ... = ...)
    } else {
      message("There is no uber data in the load directory...")}}
  #LYFT-----------------------------------------------------------------
  lyft <- function(obj, years, months,...){
    message("Loading lyft data from load directory to a sql database...")
    #create a list of file that the user wants to load
    valid_months <- etl::valid_year_month(years, months, begin = "2015-01-01")
    src <- list.files(attr(obj, "load_dir"), "lyft", full.names = TRUE)
    src_year <- valid_months %>% distinct_(~year)
    remote <- data_frame(src)
    remote <- remote %>% mutate_(tablename = ~"lyft", year = ~substr(basename(src)
    class(remote$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 director

  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)
}
```

2.8.4 utils

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

```
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
}
```

2.8.5 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
  FROM yellow_old
  GROUP BY the_year, the_month;
);
```

Chapter 3

Using New York City Yellow Taxi Data to Answer Real Life Problems

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.

3.1 New York City Taxi Driver

3.1.1 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.

```
library(dplyr)
library(readr)
dataset <- read_csv("~/Desktop/Honors Thesis/thesis/index/data/yellow_2016.08_cleaned.cs
```

```
Parsed with column specification:
cols(
  VendorID = col_integer(),
  tpep_pickup_datetime = col_datetime(format = ""),
  tpep_dropoff_datetime = col_datetime(format = ""),
  passenger_count = col_integer(),
  trip_distance = col_double(),
  RatecodeID = col_integer(),
  store_and_fwd_flag = col_character(),
  PULocationID = col_integer(),
  DOLocationID = col_integer(),
  payment_type = col_integer(),
  fare_amount = col_double(),
  extra = col_double(),
  mta_tax = col_double(),
  tip_amount = col_double(),
  tolls_amount = col_double(),
  improvement_surcharge = col_double(),
  total_amount = col_double()
)
```

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

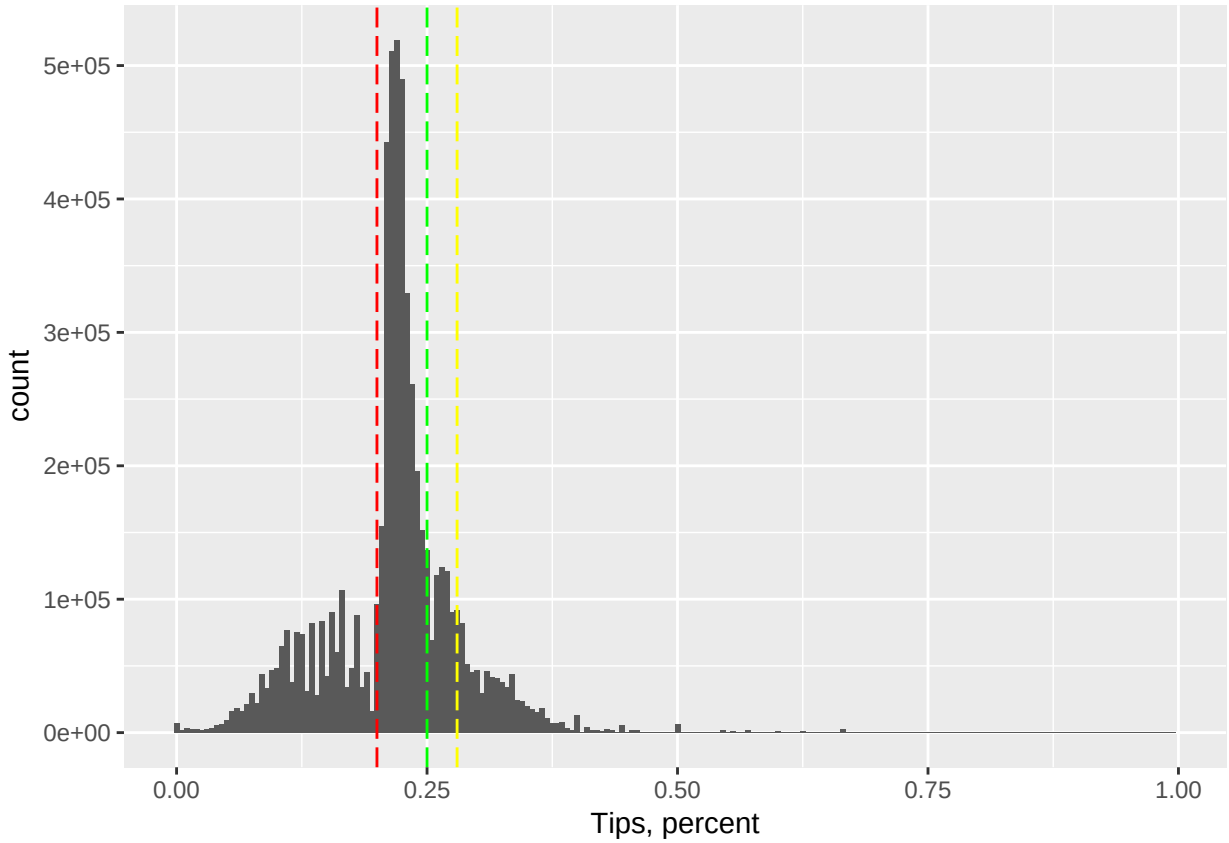
Instead of the absolute amount of tips, we want to focus on the percentage of tips that passengers pay in addition to the total fare amount.

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

Let's visualize the distribution of tip percentage:

```
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(0.20), 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
```



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(tip_perct ~ trip_distance, data = yellow_2016.08_tip)
summary(tip_distance)
```

```
Call:
lm(formula = tip_perct ~ trip_distance, data = yellow_2016.08_tip)

Residuals:
     Min       1Q   Median       3Q      Max
-0.21887 -0.01889  0.00244  0.02682  0.77850

Coefficients:
                Estimate Std. Error  t value Pr(>|t|)
(Intercept)    2.189e-01  2.812e-05 7785.080   <2e-16 ***
trip_distance -4.146e-09  8.729e-09   -0.475    0.635
---
```

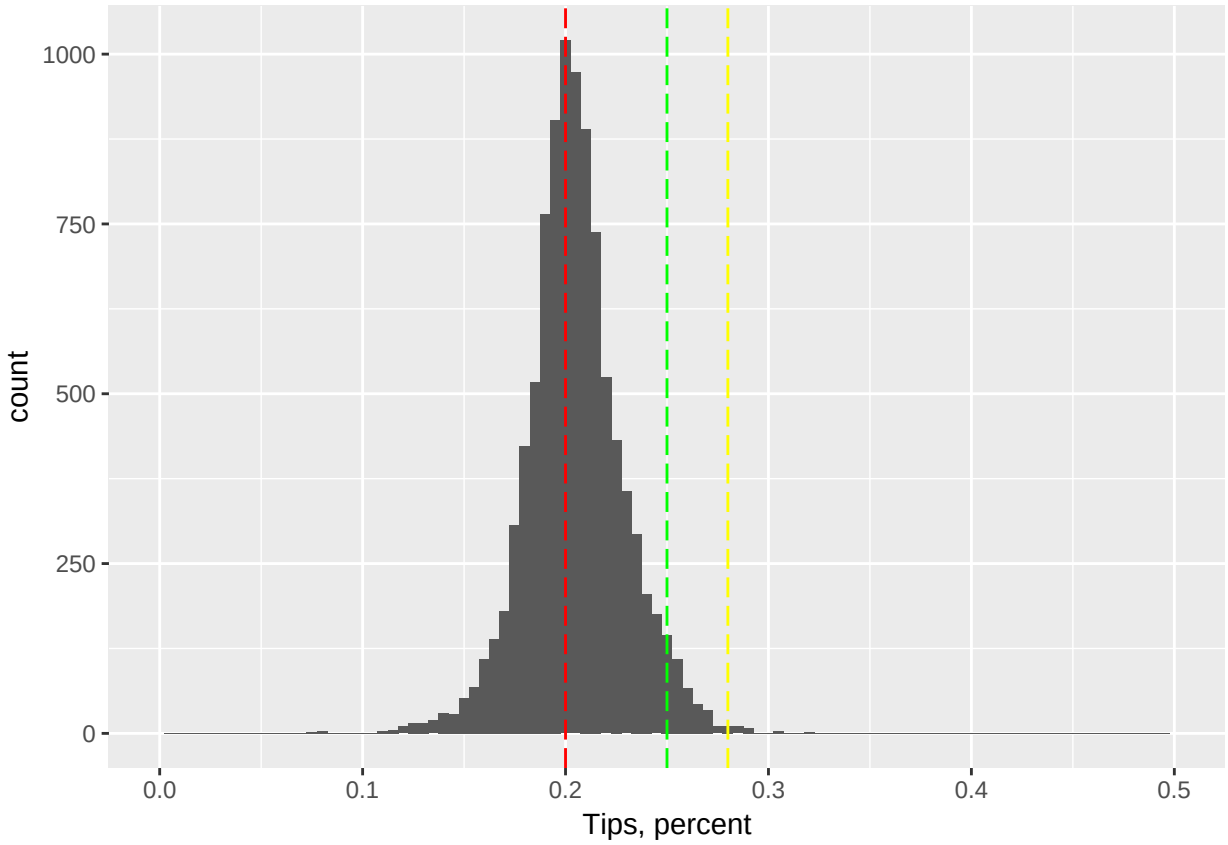
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06938 on 6088289 degrees of freedom
Multiple R-squared:  3.706e-08, Adjusted R-squared:  -1.272e-07
F-statistic: 0.2256 on 1 and 6088289 DF,  p-value: 0.6348
```

Acoording to the simple linear regression result, trip distance does not have significant impact on the percent of tips paid.

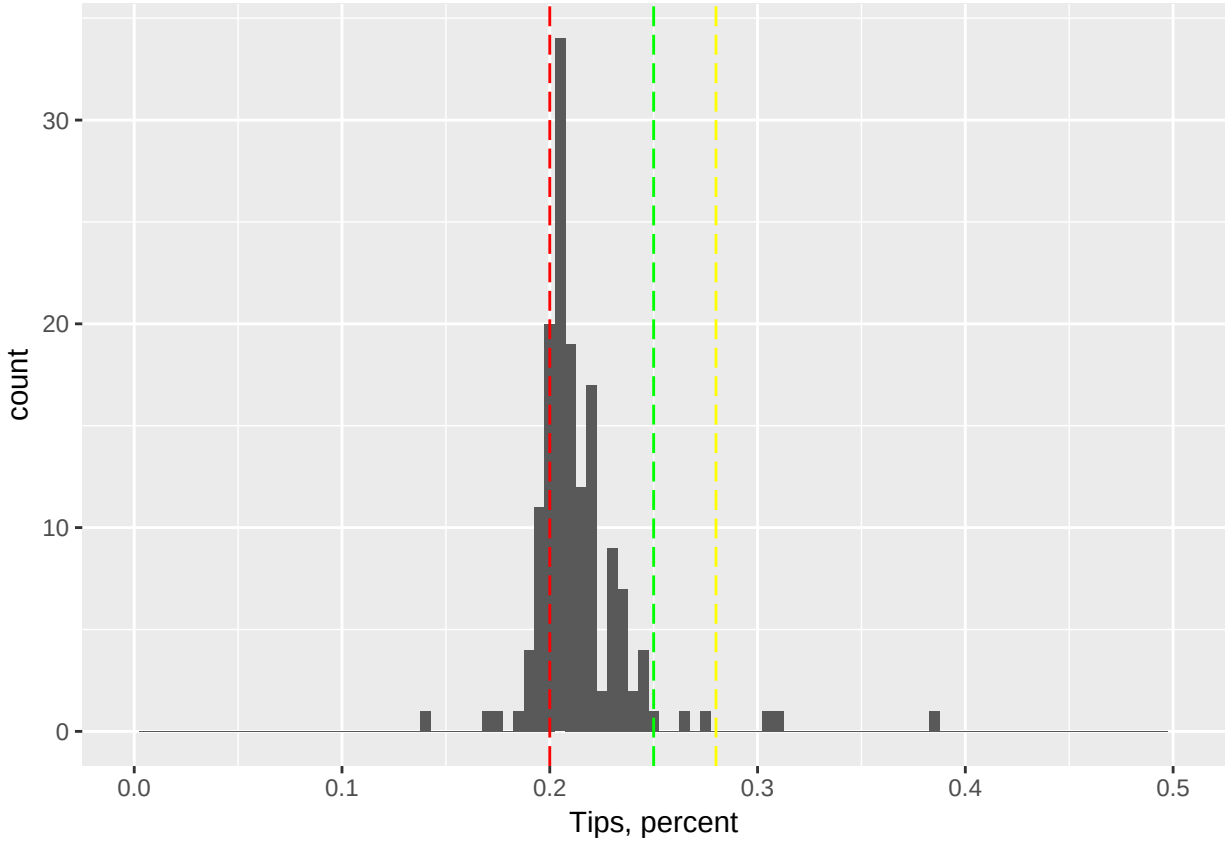
3.1.2 Aggregated Zone-level Tip Information

```
data(taxi_zone_lookup)
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")
```

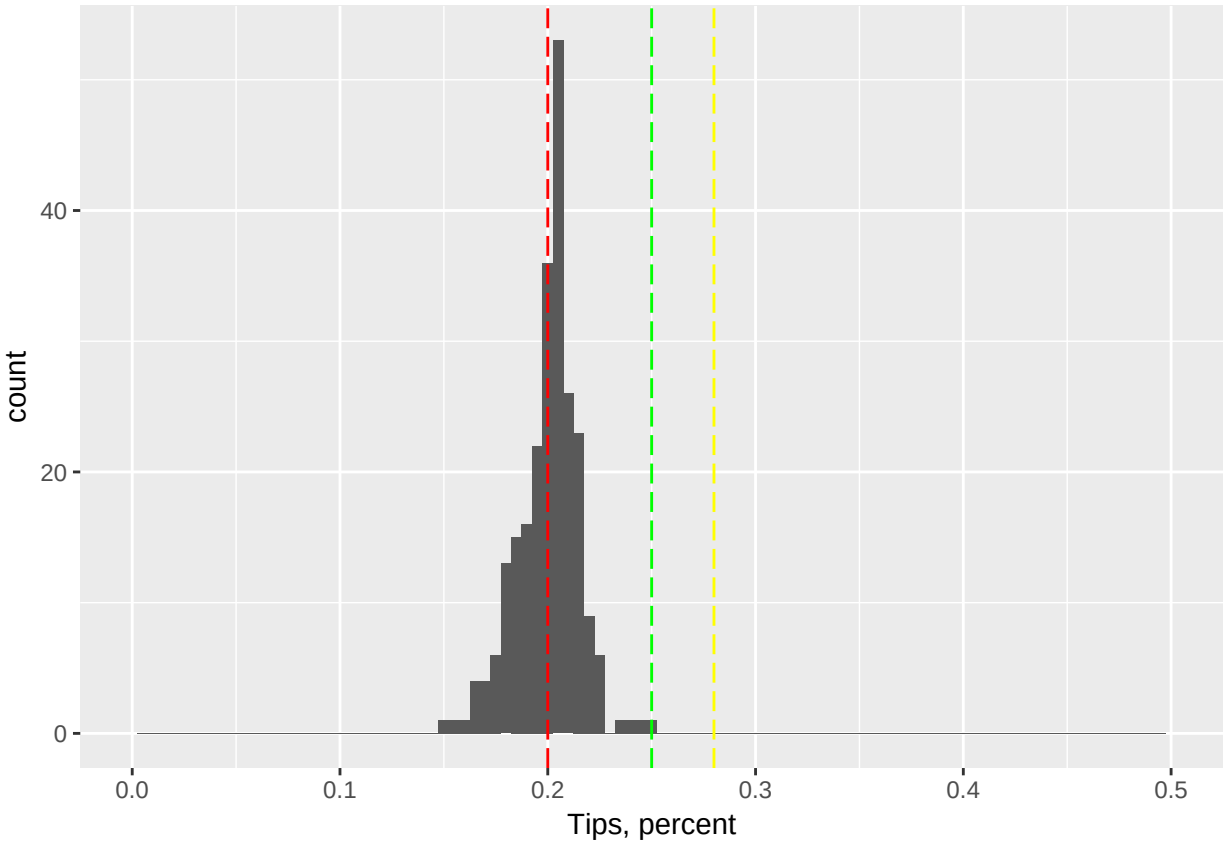
```
#zone
region_vis <- ggplot(data = tip_region, aes(x = avg_tip) ) +
  xlab("Tips, percent") +
  geom_histogram(binwidth = 0.005) +
  geom_vline(xintercept = c(0.20), col = "red",linetype = "longdash") +
  geom_vline(xintercept = c(0.25), col = "green",linetype = "longdash") +
  geom_vline(xintercept = c(0.28), col = "yellow",linetype = "longdash") +
  scale_x_continuous(limits = c(0, 0.5))
region_vis
```




```
tip_pickup <- tip_region %>%
  group_by(LocationID) %>%
  summarise(avg_tip = mean(avg_tip), num_trips=sum(trips)) %>%
  left_join(taxi_zone_lookup, by = "LocationID") %>%
  arrange(desc(avg_tip)) %>%
  filter(Zone != "Unknown")

region_pickup_vis <- region_vis %+% tip_pickup
region_pickup_vis
```




```
tip_dropoff <- tip_region %>%
  group_by(DOLocationID) %>%
  summarise(avg_tip = mean(avg_tip)) %>%
  rename(LocationID = DOLocationID) %>%
  left_join(taxi_zone_lookup, by = "LocationID")

region_dropoff_vis <- region_pickup_vis %+% tip_dropoff
region_dropoff_vis
```



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

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

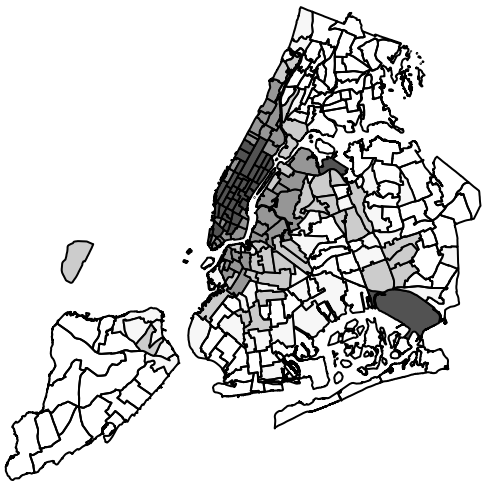
```
data("taxi_zones")
names(taxi_zones)
```

```
Loading required package: sp

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

[1] "OBJECTID"   "Shape_Leng" "Shape_Area" "zone"       "LocationID"
[6] "borough"
```

```
#names(taxi_zones)
newobj <- merge(taxi_zones, tip_pickup, by.x = "LocationID", by.y = "LocationID")
library(RColorBrewer)
cols <- brewer.pal(n = 4, name = "Greys")
lcols <- cut(newobj$num_trips,
             breaks = quantile(newobj$num_trips, na.rm = TRUE),
             labels = cols)
plot(newobj, col = as.character(lcols))
```

Table 3.1: Taxi zone with the highest tip percent, threshold = 1000

avg_tip	num_trips	Borough	Zone
0.2220842	1514	Brooklyn	Prospect Heights
0.2209707	1890	Brooklyn	Bushwick South
0.2203979	1049	Brooklyn	Crown Heights North
0.2203508	1812	Brooklyn	Clinton Hill
0.2202720	2637	Queens	Steinway
0.2196133	1177	Brooklyn	Bedford
0.2189780	3321	Brooklyn	Carroll Gardens
0.2148377	3429	Brooklyn	Greenpoint
0.2143333	5566	Queens	Long Island City/Hunters Point
0.2137938	4234	Brooklyn	East Williamsburg



3.1.3 What are the zones with the highest percent ?

```
#pick a threshold for the cutoff of number of trips
pickup_zone_1000 <- tip_pickup %>%
  filter(num_trips >= 1000) %>%
  arrange(desc(avg_tip))

library(knitr)
kable(pickup_zone_1000[1:10,2:5], caption = "Taxi zone with the highest tip percent, thr
```

If we focus on the pick-up zones that have more than 10000 trips per month, then we observe that all pick-up zones that have the highest percent tips are in Manhattan besides LaGuardia Airport.

```
#pick a threshold for the cutoff of number of trips
pickup_zone_10000 <- tip_pickup %>%
```

Table 3.2: Taxi zone with the highest tip percent, threshold = 10000

avg_tip	num_trips	Borough	Zone
0.2111362	39854	Manhattan	Hudson Sq
0.2100087	224163	Manhattan	Midtown East
0.2089467	63884	Manhattan	Central Park
0.2087323	177105	Queens	LaGuardia Airport
0.2085443	107154	Manhattan	Greenwich Village North
0.2076574	33450	Manhattan	World Trade Center
0.2076139	88966	Manhattan	UN/Turtle Bay South
0.2075438	213399	Manhattan	Penn Station/Madison Sq West
0.2073151	27055	Manhattan	Financial District South
0.2064646	56759	Manhattan	SoHo

```
  filter(num_trips >= 10000) %>%
  arrange(desc(avg_tip))
kable(pickup_zone_10000[1:10,2:5], caption = "Taxi zone with the highest tip perce
```

3.1.4 Do taxi drivers go to zones that offer high tips?

Pick-up zones with higher tips should attract more taxi drivers.

```
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)
```

```
Call:
lm(formula = trips ~ avg_tip + LocationID, data = tip_region)

Residuals:
   Min     1Q Median     3Q    Max
 -3780   -663   -183    246  62515

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2907.26    1456.57  -1.996   0.0460 *
avg_tip       18382.03     643.53  28.564   <2e-16 ***
LocationID10   -936.47    1471.76  -0.636   0.5246
LocationID100  -146.65    1456.66  -0.101   0.9198
LocationID106 -1327.88    1478.24  -0.898   0.3691
LocationID107   367.36    1457.42   0.252   0.8010
```

```
LocationID112  -990.71    1463.11  -0.677   0.4983
LocationID113   -75.78    1458.00  -0.052   0.9586
LocationID114  -151.25    1457.84  -0.104   0.9174
LocationID115  -932.89    2053.75  -0.454   0.6497
LocationID116  -545.87    1464.83  -0.373   0.7094
LocationID117 -1355.28    2053.90  -0.660   0.5094
LocationID12   -913.56    1464.52  -0.624   0.5328
LocationID125  -594.29    1459.17  -0.407   0.6838
LocationID127 -1036.80    1523.19  -0.681   0.4961
LocationID129  -862.22    1478.89  -0.583   0.5599
LocationID13   -236.51    1458.24  -0.162   0.8712
LocationID130  -801.18    1523.08  -0.526   0.5989
LocationID132   -41.13    1455.22  -0.028   0.9775
LocationID133 -1390.30    1623.97  -0.856   0.3920
LocationID134 -1464.23    1677.23  -0.873   0.3827
LocationID135 -4184.09    2058.02  -2.033   0.0421 *
LocationID137  -185.17    1457.69  -0.127   0.8989
LocationID138  -146.01    1455.46  -0.100   0.9201
LocationID14  -1130.61    1778.69  -0.636   0.5250
LocationID140    68.23    1457.41   0.047   0.9627
LocationID141   232.75    1457.51   0.160   0.8731
LocationID142   354.11    1457.96   0.243   0.8081
LocationID143   -76.35    1459.87  -0.052   0.9583
LocationID144  -233.85    1458.16  -0.160   0.8726
LocationID145  -967.14    1460.82  -0.662   0.5080
LocationID146  -693.04    1462.82  -0.474   0.6357
LocationID148   -16.40    1456.93  -0.011   0.9910
LocationID151  -160.33    1460.48  -0.110   0.9126
LocationID152  -613.12    1466.90  -0.418   0.6760
LocationID157  -921.67    2053.70  -0.449   0.6536
LocationID158   -41.50    1458.24  -0.028   0.9773
LocationID159 -1560.66    2054.04  -0.760   0.4474
LocationID161   575.57    1456.82   0.395   0.6928
LocationID162   350.15    1456.48   0.240   0.8100
LocationID163   266.21    1457.50   0.183   0.8551
LocationID164   155.58    1456.71   0.107   0.9149
LocationID165 -2688.69    2055.25  -1.308   0.1908
LocationID166  -555.16    1461.49  -0.380   0.7041
LocationID168 -1234.45    1530.99  -0.806   0.4201
LocationID17  -1090.44    1476.37  -0.739   0.4602
LocationID170   415.75    1456.42   0.285   0.7753
LocationID177 -1538.40    2054.03  -0.749   0.4539
LocationID178  -445.39    2053.60  -0.217   0.8283
LocationID179  -834.77    1469.84  -0.568   0.5701
LocationID181  -886.06    1459.97  -0.607   0.5439
```

```
LocationID186   347.47    1456.51   0.239   0.8114
LocationID188 -1101.31    1516.92  -0.726   0.4678
LocationID189 -1135.25    1471.38  -0.772   0.4404
LocationID190 -1267.45    1591.05  -0.797   0.4257
LocationID193  -918.34    1479.93  -0.621   0.5349
LocationID194 -1070.92    1540.42  -0.695   0.4869
LocationID195 -1280.82    1568.81  -0.816   0.4143
LocationID196 -1132.43    1552.63  -0.729   0.4658
LocationID197  -872.05    1778.57  -0.490   0.6239
LocationID198 -1965.61    1677.73  -1.172   0.2414
LocationID200 -1357.03    2053.91  -0.661   0.5088
LocationID202 -1286.33    1623.87  -0.792   0.4283
LocationID206 -1097.95    2053.77  -0.535   0.5929
LocationID209  -727.16    1459.76  -0.498   0.6184
LocationID210 -2751.74    2055.35  -1.339   0.1807
LocationID211  -390.09    1458.59  -0.267   0.7891
LocationID215  -754.50    1516.75  -0.497   0.6189
LocationID216  -758.87    1676.84  -0.453   0.6509
LocationID217 -1072.02    1530.88  -0.700   0.4838
LocationID219  -689.67    1676.80  -0.411   0.6809
LocationID220 -2136.47    2054.56  -1.040   0.2984
LocationID221  -834.38    2053.69  -0.406   0.6845
LocationID223 -1089.04    1466.82  -0.742   0.4578
LocationID224  -472.24    1460.49  -0.323   0.7464
LocationID225 -1284.59    1503.46  -0.854   0.3929
LocationID226  -647.57    1460.79  -0.443   0.6576
LocationID228 -1331.63    1517.09  -0.878   0.3801
LocationID229    22.10    1457.62   0.015   0.9879
LocationID230   129.93    1456.52   0.089   0.9289
LocationID231    37.85    1457.24   0.026   0.9793
LocationID232  -709.19    1459.45  -0.486   0.6270
LocationID233  -224.75    1457.82  -0.154   0.8775
LocationID234   657.47    1457.03   0.451   0.6518
LocationID236   522.15    1457.81   0.358   0.7202
LocationID237   687.51    1457.72   0.472   0.6372
LocationID238    85.83    1458.25   0.059   0.9531
LocationID239   267.90    1458.00   0.184   0.8542
LocationID24   -575.82    1463.15  -0.394   0.6939
LocationID243  -776.99    1486.37  -0.523   0.6012
LocationID244  -563.98    1463.81  -0.385   0.7000
LocationID245 -1055.00    2053.76  -0.514   0.6075
LocationID246   -28.13    1458.16  -0.019   0.9846
LocationID247 -1116.20    1540.46  -0.725   0.4687
LocationID249   183.07    1457.44   0.126   0.9000
LocationID25   -925.19    1461.22  -0.633   0.5266
```

LocationID255	-712.61	1459.30	-0.488	0.6253
LocationID256	-730.24	1460.07	-0.500	0.6170
LocationID257	-1384.43	1677.16	-0.825	0.4091
LocationID26	-1078.96	2053.76	-0.525	0.5993
LocationID260	-599.78	1470.18	-0.408	0.6833
LocationID261	-637.95	1458.13	-0.438	0.6618
LocationID262	-163.52	1458.97	-0.112	0.9108
LocationID263	45.49	1457.67	0.031	0.9751
LocationID264	-248.23	1459.69	-0.170	0.8650
LocationID265	-488.28	1590.89	-0.307	0.7589
LocationID28	-1056.99	1676.94	-0.630	0.5285
LocationID33	-882.15	1460.78	-0.604	0.5459
LocationID34	-1406.30	1591.15	-0.884	0.3768
LocationID35	-1293.82	2053.87	-0.630	0.5287
LocationID36	-1152.73	1494.51	-0.771	0.4405
LocationID37	-1100.63	1472.98	-0.747	0.4550
LocationID38	-907.22	2053.69	-0.442	0.6587
LocationID39	-1458.54	2053.98	-0.710	0.4777
LocationID4	-606.70	1459.29	-0.416	0.6776
LocationID40	-1066.91	1463.49	-0.729	0.4660
LocationID41	-519.09	1462.81	-0.355	0.7227
LocationID42	-575.20	1465.29	-0.393	0.6947
LocationID43	-307.29	1459.36	-0.211	0.8332
LocationID45	-730.91	1459.53	-0.501	0.6165
LocationID48	319.92	1456.55	0.220	0.8262
LocationID49	-1095.55	1471.36	-0.745	0.4565
LocationID50	-363.85	1458.10	-0.250	0.8030
LocationID52	-930.55	1462.76	-0.636	0.5247
LocationID54	-1667.06	1540.91	-1.082	0.2793
LocationID56	-961.27	2053.73	-0.468	0.6398
LocationID6	-972.02	2053.72	-0.473	0.6360
LocationID61	-1109.13	1476.38	-0.751	0.4525
LocationID62	-1391.55	1591.14	-0.875	0.3818
LocationID65	-734.58	1459.59	-0.503	0.6148
LocationID66	-858.00	1463.19	-0.586	0.5576
LocationID68	270.91	1457.16	0.186	0.8525
LocationID69	-1571.95	2054.06	-0.765	0.4441
LocationID7	-753.88	1461.49	-0.516	0.6060
LocationID70	-1311.82	1494.61	-0.878	0.3801
LocationID74	-517.69	1461.93	-0.354	0.7233
LocationID75	-302.63	1459.76	-0.207	0.8358
LocationID76	-1424.36	2053.96	-0.693	0.4880
LocationID77	-1579.84	2054.06	-0.769	0.4418
LocationID79	517.61	1456.75	0.355	0.7224
LocationID8	-792.30	2053.66	-0.386	0.6997

```
LocationID80   -960.44    1462.94  -0.657   0.5115
LocationID82  -1052.65    1494.43  -0.704   0.4812
LocationID83   -982.97    1540.37  -0.638   0.5234
LocationID87   -390.35    1457.74  -0.268   0.7889
LocationID88   -670.38    1458.49  -0.460   0.6458
LocationID89   -997.81    1516.87  -0.658   0.5107
LocationID9   -1433.27    2053.96  -0.698   0.4853
LocationID90    157.84    1457.70   0.108   0.9138
LocationID91    323.71    1778.61   0.182   0.8556
LocationID92   -175.80    1778.47  -0.099   0.9213
LocationID93  -1127.50    1479.02  -0.762   0.4459
LocationID95  -1065.11    1497.03  -0.711   0.4768
LocationID97   -844.35    1460.75  -0.578   0.5633
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1452 on 9499 degrees of freedom
Multiple R-squared:  0.1573,    Adjusted R-squared:  0.1437
F-statistic: 11.59 on 153 and 9499 DF,  p-value: < 2.2e-16
```

Each one percent increase in tips is associated with 18382.03 increase in the number of trips, controlling the pick-up zone.

3.1.5 Which pick-up zone has the highest price per minute

estmate slow traffic time

3.2 New York City Taxi Consumer

3.2.1 Does taxi fare change through time? Is taxi ride becoming more expensive?

3.3 New York City Taxi Fare & Limousine Commission

3.3.1 Should there be a flat rate between Manhattan and the JFK Airport?

People in Manhattan benefit from the $52 flat rate.

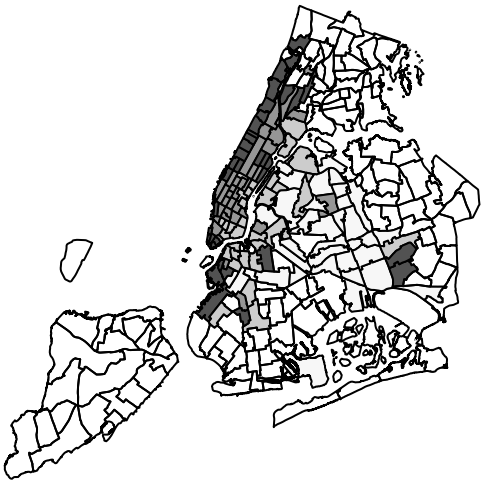
Why is there a flat rate to and from JFK airport and any location in Manhattan? Why is the flat rate $52? Does TLC make profit from the $52 flat rate? Does $52 reduce the cogestion on the road to JFK airport and make taking a train a more preferable choice?

If there is no flat rate between JFK and Manhattan,

```
jfk_trip <- dataset %>%
  filter(RatecodeID == 2) %>%
  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,
         est_diff = est_fare - fare_amount)

to_jfk <- jfk_trip %>%
  filter(DOLocationID == 132)

from_jfk <- jfk_trip %>%
  filter(PULocationID == 132)
```

```
to_jkf_zone <- to_jfk %>%
  group_by(PULocationID) %>%
  summarise(num_trips = n(),
            avg_dis = mean(trip_distance),
            avg_fare = mean(est_fare))

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

```
cols <- brewer.pal(n = 4, name = "Greys")
lcols <- cut(to_jkf_fare$avg_fare,
             breaks = quantile(to_jkf_fare$avg_fare, na.rm = TRUE),
             labels = cols)
plot(to_jkf_fare, col = as.character(lcols))
```



```
to_jkf_zone_above <- to_jkf_zone %>%
  filter(avg_fare >= 52) %>%
  arrange(desc(avg_fare))
kable(to_jkf_zone_above[1:10,], caption = "Zones that woulde have paid more than $
```

\begin{table}

\caption{Zones that woulde have paid more than $52}
\end{table}

PULocationID	num_trips	avg_dis	avg_fare
47	1	27.08000	76.54000
195	4	26.14000	74.19000
220	1	24.30000	69.59000
54	1	25.80000	67.80000
127	6	22.85833	65.06250
243	8	21.54250	63.82125
17	1	23.70000	62.55000
13	1266	22.09787	62.44055
244	74	20.42216	60.27892
133	1	18.74000	60.19000

Imagine you are travelling to New York City and you do not know much about the city. Travellers tend to gather around Mahanttan, and without the flat rate, passengers would have paid more than $52 to take a taxi to go to the JFK Airport. The $52 flat rate is nice for people who are not very familiar with New York City, and it incentivize tourists to take taxi to the JFK Airport. It also helps taxi drivers to get

more tips to JFK Airport.

3.3.2 However, are taxi drivers happy with the flat rate?

What the expected fare from JKF Airport how much time it would take for a cb driver to do a round trip

Conclusion

Chapter 4

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.