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.

Introduction

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

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.

2.3 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 Source Code

2.7.1 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"), ".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/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(ye
    message("Downloading Uber 2014 data...")
    etl::smart_download(obj, small$src,...)
  } else if (2014 %in% remote_small$year && !(2015 %in% remote_small$year)) {
    message("Downloading Uber 2014 data...")
    #file paths
    small <- remote_small %>%
      mutate_(month_abb = ~tolower(month.abb[month]),
                src = ~file.path(path, paste0("uber-raw-data-",month_abb,substr(ye
    etl::smart_download(obj, small$src,...)}
  else {warning("The Uber data you requested are not currently available. Only d
  }
#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)
}
```

2.7.2 ETL Transform

```
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$src)
    # check that the sys support command line, and then remove the blank 2nd r
    if(length(src_small_green_1) != 0) {
      if (.Platform$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 blan
                in the green taxi datasets. This might affect loading data into
      }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 = TRUE))
    file_remian <- setdiff(in_raw,in_load)
    file.copy(file.path(attr(obj, "raw_dir"),file_remian), file.path(attr(obj, "load_
#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"),
  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$OS.type == "unix"){
        cmds_3 <- paste("cut -d, -f2-7 ",file.path(tempdir(),"uber.csv"), " > ", f
        lapply(cmds_3, system)
      } else {
        message("Windows system does not currently support removing the 2nd blank
                in the green taxi datasets. This might affect loading data into SQ
      }else {
        "You did not request for any green taxi data, or all the green taxi data y
    }
  #LYFT----------------------------------------------------------------
  lyft <- function(obj, years, months){
    valid_months <- etl::valid_year_month(years, months = 1, begin = "2015-01-01")
    message("Transforming lyft data from raw to load directory...")
    src <- list.files(attr(obj, "raw_dir"), "lyft", full.names = TRUE)
    src_year <- valid_months %>% distinct_(~year)
    remote <- data_frame(src)
    remote <- remote %>%
      mutate_(lcl = ~file.path(attr(obj, "load_dir"),basename(src)),
              basename = ~basename(src), year = ~substr(basename,6,9))
    class(remote$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)
}
```

2.7.3 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 = "2016
    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
  } 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 director
  } 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)
}
```

2.7.4 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
}
```

2.7.5 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_pickup_datetime) AS the_month, count(*) AS num_trips FROM yellow_old GROUP BY the_year, the_month;);

Chapter 3

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

3.1 New York City Taxi Fare Pricing Model

3.2 Transportation Network in New York City

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.