DMV Commute Data Analysis Tutorial

Daniel Brewer and Joseph Kahadze

5/7/2019

The Washington D.C. metropolitan area is one of the fastest growing regions in America. Each day, millions of commuters travel to and from work on the highways surrounding the city. As the area changes, the commute changes as well. Between changing demographics and infrastructure, commuter variables are constantly changing each year. But the real question is, what are the trends? For example, how are the means of transportation to work changing for foreign born U.S. citizens? Or, what percentage of the population will be leaving for work between 5:00am and 5:29am in 2030? These are important questions that, with data analysis, can be answered and provide essential information about one of the most heavily traveled areas in the world.

Fortunately, data is collected through surveys that is publicly available online. The data set that will be used today is called "Means of Transportation to Work by Selected Characteristics". It provides percentages of the population based on method of transportation. In order to find and download the desired data sets for our analysis, careful instructions need to be followed:

1. Go to factfinder.census.gov
2. Click on "Advanced Search" and then "SHOW ME ALL" under the red Community Facts heading
3. In the search table with the box labeled "topic or table name" type in "S08*" without the quotes and hit the "GO" button
4. You will find many data sets containing information about commuting characteristics. In order to find data from the D.C. metro area, you will need to click on the blue "Geographies" button on the left side of the window
5. make sure "most requested geographic types" is selected via the radio button. Click on "select a geographic type" and scroll down the drop down until you see "Metropolitan Statistical Area/Metropolitan Statistical Area". Click on it
6. In the list of Metropolitan areas, scroll down until you reach "Washington-Arlington-Alexandria, DC-VA-MD-WV Metro Area". Select it, click "ADD TO YOUR SELECTIONS", and then close out of the pop up
7. Find "Show results from:" above the table and select "2006" from it. This will filter out data sets and provide us with one that ranges from 2006-2017
8. This will bring up the data table, with each version of the table (based on year) on the left hand side. The next thing to do is download each of these tables in .csv form to use in our data set. Make sure you are on the 2017 data set, and click the download button above the table
9. Check the "Use the data" option and make sure that both "Merge the annotations and data into a single file" and "Include descriptive data element names" are selected. Click "OK" and then "Download" once loaded.
10. Please save and unzip the .zip file in the same location that you are writing your R code in. This will make it easier to access the .csv files.
11. Repeat steps 9 and 10 for the 2016-2006 data sets

Now that you have all the data sets together in a folder, it is ready for data cleaning. Unfortunately, the provided .csv files are not ready for analysis right away.

To begin, for debugging and reproducible purposes, make sure knitr is echoing output

Data Cleansing

As mentioned, the .csv file comes as an elongated row of misplaced variables. It will be exhaustive, but also necessary for what we are attempting to accomplish.

First, import the following libraries. These will be essential throughout the tutorial. For a through explanation on the use of each of these libraries, go to https://cran.r-project.org and read their descriptions.

```
library(tidyverse)
library(data.table)
library(ggplot2)
library(broom)
```

The csv file in its original form is messy. Please go to Table 1 below to see the data frame for the 2017 table. As you can see, there is only a few rows with attributes within those rows. Overall, very messy. We are going to create 16 different tables for each type of observation. Each table will contain the observations, their percentages, their margin of errors, a subject ID representing the mode of transportation, and the year. The 16 observations are as followed:

1. Age
2. Sex
3. Race
4. Citizenship status
5. English ability
6. Earnings (in the past 12 months)
7. Occupation
8. Industry
9. Class of worker
10. Place of work
11. Departure time to work
12. Travel time to work
13. Housing Tenure
14. Number of vehicles owned
15. Poverty Status
16. Total number of commuters

After the data is cleaned, you will have a list of 16 clean tables ready for analysis based on each of these observations. For a visual representation, please refer to Table 2 below. This table is for the Age observation and is similar to the other tables

To begin, we are going to create a function that creates the total number of commuters table. Do not get this confused with "Total" as the mean of transporation category. That is for the total percentage of commuters in each observation, while the table created in function below contains integers representing the number of subjects for each mean of transportation. For a completed example, please refer to Table 3.

Please read through the comments of the code carefully to understand what each block of code is doing after reading the preliminary explanation.

The function will take in the full, dirty csv file shown in Table 1, and the year of the csv file. In each csv file, the "total" subject observation is in columns 4 through 11. Therefore, anything other than those need to be filtered out using the select method.

Once we have a data frame that has properly been filtered, we want to set the column names to include our attributes per observation. However, this only creates one long row with each column being a single measure of an attribute per mode of transportation. Each observation is not a row and there are multiple variables in columns.

In order to Tidy our total table up, we must use gather, spread, and separate to place our variables into separate columns and ensuring each observation is a row. This will give us our estimate, margin of error, year, and mean of transportation in each column, just like in Table 3. For more information on spread and gather, refer to this link: https://tidyr.tidyverse.org. After type conversion (so we can use the variables within the columns) and a str_replace_all call to remove unnecessary characters, the table is almost ready.

To ensure there is only one observation per table and for readability, we will convert the "Subject" column that contains the mean of transportation into a decimal number by creating a new table with the mean of transportation and its unique number. These numbers will be unique to each method of transportation and are the following:

1. Total
2. Car, truck or van – drove alone
3. Car, truck, or van – carpooled
4. Public transportation (excluding taxi cab)

This newly created table will be "left joined" with the total table and then "subject" will be filtered out in order to expose a purely numerical attribute. Left join combines the two tables based on a common attribute (Subject/mean of transportation). For a better grasp on the join functions, go to https://dplyr.tidyverse.org/reference/join.html

```
#Creates a custom table of the total amount of subjects per transporation method.
#Inputs are a full
create_total_table <- function(total, year1) {

  #Filter out any observations other than the "total" subject observation
  total <- total %>% select(c(4:11))

  #Set the column names to the attributes that are currently within the row
  total %>% setnames(old = colnames(total), new = as.character(total[1, ]))
  total <- total[-c(1), ]

  #Place the attribute names into columns and the values in rows while also
  #adding the observation year
  total <-
    total %>% gather(cols, percentage, na.rm = TRUE) %>%
    separate(cols, c("Subject", "Measure", "Observation"), sep = ";") %>%
    spread(Measure, percentage) %>% select(-c("Observation"))  %>%
    mutate(Year = as.integer(year1)) %>%
    type_convert()

  #Remove periods and replace with spaces for readability
  names(total) <- str_replace_all(names(total), c(" " = "."))

  #Adds in a unique subject_id, with each subject ID corresponding to a
  #transportation method. Integers were assigned directly in order to maintain
  #consistency with the original dataset
  unique <-
    total %>%
    select(Subject) %>%
    unique() %>%
    mutate(subject_id = as.integer(row_number()))
  unique[4, 2] <- as.integer(1)
  unique[3, 2] <- as.integer(4)
```

```
  unique[1, 2] <- as.integer(3)
  total <-
    total %>%
    left_join(unique, c("Subject")) %>%
    select(-Subject) %>%
    arrange(subject_id)
  names(total)[1] <- "Estimate"

  total

}
```

The next function to be created is the create_table function. This will take in our larger csv data frame, the name of the observation, and the year of the data frame. This will return a mostly clean singular observation table and will be extremely helpful when we create our 15 other tables.

This function is used due to the specific special characters that are observed within the data frame, and therefore there is a good bit of repetition between observations. As mentioned, much cleaning is required for data set. If this function does not make sense at first, it will during the creation of our "create_df_list" function.

First, a "table" variable is created by filtering out any entities that do not match the desired one. We will be using the str_detect function to identify the desired observations, and then arrange them by subject_id. A year column will be added representing the year, and type_convert() will be called so the variables can be used during analysis. As mentioned, str_replace_all and mutate was used in order to clean up special characters, which is a major problem with the data set. For more information on what str_replace and mutate do in terms of data cleansing, please visit the following resources:

https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_replace https://dplyr.tidyverse.org/reference/mutate.html

```
#Takes in a dataframe, an observation within the dataframe, and the year of
#observation. Returns a table tailored specifically to the observation
#within the original dataframe
create_table <- function(frame, identifier, year) {

  table <- frame %>%
    filter(str_detect(Observation, paste0("^", identifier))) %>%
    mutate(Observation =
             gsub(paste0("^", identifier, " - "), "", Observation)) %>%
    arrange(subject_id) %>%
    mutate(Year = as.integer(year)) %>%
    type_convert()
  names(table)[1] <- identifier
  names(table) <- str_replace_all(names(table), c("^ " = ""))
  names(table) <- str_replace_all(names(table), c("\\\\" = ""))
  names(table) <- str_replace_all(names(table), c(" " = "."))
  table
}
```

The next (and final) function in the data cleansing part of this tutorial is the create_df_list function. This takes in a given csv file (our data set) and the year that the data was recorded in. This will output a list of the 16 observation tables, completely in their clean form. We will divide the function up into two parts, because part 1 is fairly similar to our creation of the total table.

The function begins by reading in the data frame from the csv file, and then passes it to the create_total_table function that we created above to get our total subject total for the given csv file.

Opposite from the create_total_table function, we will filter out entities that are not percentages. This is because we have already handled the total subject observation. Regarding gather, spread, and separate: these functions have the same functionality as in the create_total_table. However, there will be some differences regarding cleaning observation names to remain consistency through all the years:

1. Mutate will be used to remove any "Workers 16 years and over" phrases. Some years had this present in their datasets, while others did not. This was filtered using regular expressions: https://stat545.com/block022_regular-expression.html
2. The phrase "Speak language other than English -" phrase was removed in order to maintain consistency between data sets due to the presence of the phrase
3. Similar to number 2, "NATIVITY AND" was removed
4. "Foreign born -" was removed similar as above
5. "One race -" was removed to maintain consistency, similar to above

As with the create_total_table function, unique subject ID's were created for better readability and maintaining one observation per table

Part 2 of the create_df_list (please follow the comments for further instruction when writing code):

1. The first table created is the age table. This makes use of the create_table function, and for the purposes of this tutorial we will filter out the median age, hence the use of the filter function.

2. Next is the sex table, which our create_table function takes care of completely

3. The race table is created, except we will filter out smaller race categories and rename the RACE.AND.HISPANIC.OR.LATINO.ORIGIN column to simply Race. This similar method will be done to most of the tables due to the tendency to have periods within the observation name. This is done using the rename method.

4. Follow the code comments and create the tables until you reach the occupation table. The tables up until it will be similar to the ones described in the previous 3 steps.

5. The occupation table will be coded differently than the previous ones and the ones after. Due to the combination of two occupations after 2009, these two occupations had to be merged into a renamed one and had to have their values added. This ensured consistency between data sets. The two occupations to be combined are "Farming, fishing, and forestry occupations" and "Construction, extraction, and maintenance". A combination of mutate, join, and filter was used to create a separate table with the combined occupations and then join them back into the original data. In order to truly understand these three functions and occupation table creation, please visit these three resources:

   https://dplyr.tidyverse.org/reference/mutate.html https://dplyr.tidyverse.org/reference/join.html https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/filter

6. Follow the code comments for the creation of the remaining tables. The rest are similar to the previous (except the occupation table) and make use of gsub and column renaming for data cleansing and better graphical representation. Gsub is used to substitute certain strings (in our case an empty string) into other specific strings. For a better explanation, please visit this resource: http://www.endmemo.com/program/R/gsub.php

7. The final step of the function is to simply load all of the created tables into a list called year_frames and return it. This will return a list of our cleaned frames in a package ready for use.

```r
#Part 1
#Takes in a csv file and the year in which the csv file was recorded in.
#Returns a list of dataframes that are each a categorical observation
create_df_list <- function(csv, year) {
  df <- read_csv(csv, col_types = cols())

  total_table <- create_total_table(df, year)

  #Filter out entities that are not percentages
  df <- df %>% select(-c(1:11))
  df %>% setnames(old = colnames(df), new = as.character(df[1, ]))
  df <- df[-c(1), ]

  #Creates a table with proper attributes and filters common phrases seen
  #within observations in order to produce categorical entities
  df <-
    df %>% gather(cols, percentage, na.rm = TRUE) %>%
    separate(cols, c("Subject", "Measure", "Observation"), sep = ";") %>%
    spread(Measure, percentage) %>%
    mutate(Observation =
             gsub("^ ", "", Observation)) %>%
    mutate(Observation =
             gsub(
      "^Workers 16 years and over[[:alpha:]|[:space:]]* - ",
      "",
      Observation
    )) %>%
    mutate(Observation =
      gsub("Speak language other than English - ", "", Observation)) %>%
    mutate(Observation =
      gsub("^NATIVITY AND ", "", Observation)) %>%
    mutate(Observation =
      gsub("Foreign born - ", "", Observation)) %>%
    mutate(Observation =
      gsub("One race - ", "", Observation))

  #Adds in a unique subject_id, with each subject ID corresponding to a
  #transportation method. Integers were assigned directly in order to maintain
  #consistency with the original dataset
  unique <-
    df %>%
    select(Subject) %>%
    unique() %>%
    mutate(subject_id = as.integer(row_number()))
  unique[4, 2] <- as.integer(1)
  unique[3, 2] <- as.integer(4)
  unique[1, 2] <- as.integer(3)

  df <- df %>%
    left_join(unique, c("Subject")) %>%
    select(-Subject)

  #PART 2
```

```r
#Filters out median age (not needed for this tutorial, but can be included
#if you would like)
age_table <-
  create_table(df, "AGE", year) %>% filter(AGE != "Median age (years)")

sex_table <- create_table(df, "SEX", year)

#Filters out smaller race categories
race_table <-
  create_table(df, "RACE AND HISPANIC OR LATINO ORIGIN", year) %>%
  rename(Race = RACE.AND.HISPANIC.OR.LATINO.ORIGIN) %>%
  filter(
    Race == "American Indian and Alaska Native" |
      Race == "Asian" |
      Race == "Black or African American" |
      Race == "Some other race" |
      Race == "White" | Race == "Two or more races"
  )

nativity_table <- create_table(df, "CITIZENSHIP STATUS", year)

english_ability_table <-
  create_table(df,
    "LANGUAGE SPOKEN AT HOME AND ABILITY TO SPEAK ENGLISH", year) %>%
  rename(English.ability =
           LANGUAGE.SPOKEN.AT.HOME.AND.ABILITY.TO.SPEAK.ENGLISH)

earnings_string <-
  paste0(
    "EARNINGS IN THE PAST 12 MONTHS \\(IN ",
    year,
    " INFLATION-ADJUSTED DOLLARS\\) FOR WORKERS"
  )

earnings_table <- create_table(df, earnings_string, year)
colnames(earnings_table)[1] <- "Earnings"

#Cleans observation data by making them consistent throughout data years
earnings_table <-
  earnings_table %>%
  mutate(Earnings =
           gsub("^Workers 16 years and over with earnings - ", "", Earnings)) %>%
  filter(
    Earnings != "Median earnings (dollars)",
    Earnings != "Workers 16 years and over with earnings"
  ) %>%
  mutate(Earnings =
           gsub(",000", "k", Earnings))

poverty_status_table <-
  create_table(df, "POVERTY STATUS IN THE PAST 12 MONTHS", year)  %>%
  mutate(
    POVERTY.STATUS.IN.THE.PAST.12.MONTHS =
```

```
      gsub(
      "^Workers 16 years and over for whom poverty status is determined - ",
      "",
      POVERTY.STATUS.IN.THE.PAST.12.MONTHS
    )
  ) %>%
  mutate(
    POVERTY.STATUS.IN.THE.PAST.12.MONTHS = gsub(
      " of the poverty level",
      "",
      POVERTY.STATUS.IN.THE.PAST.12.MONTHS
    )
  ) %>%
  rename(Poverty.status =
           POVERTY.STATUS.IN.THE.PAST.12.MONTHS) %>%
  filter(Poverty.status !=
           "Workers 16 years and over for whom poverty status is determined")

occupation_table <- create_table(df, "OCCUPATION", year)
colnames(occupation_table)[1] <- "Occupation"

#After 2009 in the dataset, two observation categories were merged into one
#category. Therefore, entities that were observations of these two categories
#in the sub-2010 datasets had their estimate percentages added and were
#combined into one observation entity (Natural resources, construction, and
#maintenance)
occ_value <-
  occupation_table %>%
  filter(Occupation == "Farming, fishing, and forestry occupations") %>%
  select(Estimate, subject_id) %>%
  rename(merged_estimate = Estimate)
occupation_table <-
  occupation_table %>%
  full_join(occ_value, by = "subject_id")
occupation_table <-
  occupation_table  %>%
  mutate(Occupation = ifelse(
    str_detect(Occupation, "^Management"),
    "Management, Professional, etc.",
    Occupation
  )) %>%
  mutate(Occupation =
           gsub(" occupations", "", Occupation)) %>%
  mutate(Occupation = ifelse(
    str_detect(Occupation, "^Armed"),
    "Military specific",
    Occupation
  )) %>%
  mutate(
    Estimate = ifelse(
      Occupation == "Construction, extraction, and maintenance",
      Estimate + merged_estimate,
      Estimate
```

```r
      )
    ) %>%
    mutate(
      Occupation = ifelse(
        str_detect(Occupation, "^Construction, extraction"),
        "Natural resources, construction, and maintenance",
        Occupation
      )
    ) %>%
    filter(Occupation != "Farming, fishing, and forestry")

  industry_table <- create_table(df, "INDUSTRY", year)

  class_table <-
    create_table(df, "CLASS OF WORKER", year) %>%
    mutate(CLASS.OF.WORKER = gsub(" workers", "", CLASS.OF.WORKER))

  #Unneccessary strings were removed to enhance graphical representation
  place_of_work_table <-
    create_table(df, "PLACE OF WORK", year) %>%
    mutate(PLACE.OF.WORK =
             gsub("^Worked in state of residence - ", "", PLACE.OF.WORK)) %>%
    mutate(PLACE.OF.WORK =
             gsub("Worked ", "", PLACE.OF.WORK))

  departure_time_table <-
    create_table(df, "TIME LEAVING HOME TO GO TO WORK", year)
  colnames(departure_time_table)[1] <- "Departure.time"

  #Unneccessary strings were removed to enhance graphical representation
  travel_time_table <-
    create_table(df, "TRAVEL TIME TO WORK", year) %>%
    mutate(TRAVEL.TIME.TO.WORK =
             gsub(" minutes", "", TRAVEL.TIME.TO.WORK))

  #Unneccessary strings were removed to enhance graphical representation
  housing_tenure_table <-
    create_table(df, "HOUSING TENURE", year) %>%
    mutate(HOUSING.TENURE =
             gsub(" housing units", "", HOUSING.TENURE))
  colnames(housing_tenure_table)[1] <- "Housing.tenure"

  #Unneccessary strings were removed to enhance graphical representation
  vehicles_table <-
    create_table(df, "VEHICLES AVAILABLE", year) %>%
    mutate(VEHICLES.AVAILABLE =
             gsub(" vehicle available", "", VEHICLES.AVAILABLE)) %>%
    mutate(VEHICLES.AVAILABLE =
             gsub("No", "0", VEHICLES.AVAILABLE)) %>%
    mutate(VEHICLES.AVAILABLE =
             gsub(" vehicles available", "", VEHICLES.AVAILABLE))

  #A list of all clean, observation data frames is created and returned
```

```
  year_frames <- list(
    age_table,
    sex_table,
    race_table,
    nativity_table,
    english_ability_table,
    earnings_table,
    poverty_status_table,
    occupation_table,
    industry_table,
    class_table,
    place_of_work_table,
    departure_time_table,
    travel_time_table,
    housing_tenure_table,
    vehicles_table,
    total_table
  )
  year_frames
}
```

As of now, we have only created a method to create a list of observation data frames for a single year. It is now time to create lists for all years, and then merge them together to get a list of observation tables containing data from ALL years.

The first step is to create a list of lists called "year_frame_double" by calling our create_df_list for each year and placing the returned object within the list. If we had more data, we can do this with many more years. However, we will only be using from 2006 and forward for this tutorial.

The next step is to join (remember from above!) all the years together for each observation frame. This will be done with a double for loop, which will go through each entity and continuously join it year by year. At the end, we will have a list containing 16 data frames of our observations from the years 2006-2017. If you are still unsure what this would look like, please refer to Table 2 and Table 3. Both of these were called by indexing directly from our "cleaned data" list!

Now, we have our 16 observations from all of our desired years, cleaned and packaged for data analysis. If you are still unsure about lists and for loops in R, please go to the following resources:

https://www.tutorialspoint.com/r/r_lists.htm https://warwick.ac.uk/fac/sci/moac/degrees/moac/ch923/r_introduction/r_programming/

```
#A list of lists is created containing each observed year and a list of its
#clean observation tables
year_frame_double <- list(
  create_df_list("ACS_17_1YR_S0802/ACS_17_1YR_S0802.csv", 2017),
  create_df_list("ACS_16_1YR_S0802/ACS_16_1YR_S0802.csv", 2016),
  create_df_list("ACS_15_1YR_S0802/ACS_15_1YR_S0802.csv", 2015),
  create_df_list("ACS_14_1YR_S0802/ACS_14_1YR_S0802.csv", 2014),
  create_df_list("ACS_13_1YR_S0802/ACS_13_1YR_S0802.csv", 2013),
  create_df_list("ACS_12_1YR_S0802/ACS_12_1YR_S0802.csv", 2012),
  create_df_list("ACS_11_1YR_S0802/ACS_11_1YR_S0802.csv", 2011),
  create_df_list("ACS_10_1YR_S0802/ACS_10_1YR_S0802.csv", 2010),
  create_df_list("ACS_09_1YR_S0802/ACS_09_1YR_S0802.csv", 2009),
  create_df_list("ACS_08_1YR_S0802/ACS_08_1YR_S0802.csv", 2008),
  create_df_list("ACS_07_1YR_S0802/ACS_07_1YR_S0802.csv", 2007),
```

```r
  create_df_list("ACS_06_EST_S0802/ACS_06_EST_S0802.csv", 2006)

)
```

```
## Warning: package 'bindrcpp' was built under R version 3.5.2
```

```r
cleaned_data <- vector("list", 16)

#Each observation table within each observed year dataframe list is continously
#joined with a corresponding observation table in the previous year. This returns
#a list of dataframes, with each dataframe being an observation in all years
for (i in 1:16) {
  joined_df <- year_frame_double[[1]][[i]]

  for (j in 2:12) {
    joined_df <-
      full_join(as.data.frame(joined_df),
                as.data.frame(year_frame_double[[j]][[i]]))
  }
  cleaned_data[[i]] <- joined_df
}
```

Table 1

```r
og_2017 <- read_csv("ACS_17_1YR_S0802/ACS_17_1YR_S0802.csv", col_types = cols())
head(og_2017)
```

```
## # A tibble: 2 x 811
##   GEO.id GEO.id2 `GEO.display-la~ HC01_EST_VC01 HC01_MOE_VC01 HC02_EST_VC01
##   <chr>  <chr>   <chr>            <chr>         <chr>         <chr>
## 1 Id     Id2     Geography        Total; Estim~ Total; Margi~ Car, truck, ~
## 2 310M3~ 47900   Washington-Arli~ 3320895       14865         2204896
## # ... with 805 more variables: HC02_MOE_VC01 <chr>, HC03_EST_VC01 <chr>,
## #   HC03_MOE_VC01 <chr>, HC04_EST_VC01 <chr>, HC04_MOE_VC01 <chr>,
## #   HC01_EST_VC03 <chr>, HC01_MOE_VC03 <chr>, HC02_EST_VC03 <chr>,
## #   HC02_MOE_VC03 <chr>, HC03_EST_VC03 <chr>, HC03_MOE_VC03 <chr>,
## #   HC04_EST_VC03 <chr>, HC04_MOE_VC03 <chr>, HC01_EST_VC04 <chr>,
## #   HC01_MOE_VC04 <chr>, HC02_EST_VC04 <chr>, HC02_MOE_VC04 <chr>,
## #   HC03_EST_VC04 <chr>, HC03_MOE_VC04 <chr>, HC04_EST_VC04 <chr>,
## #   HC04_MOE_VC04 <chr>, HC01_EST_VC05 <chr>, HC01_MOE_VC05 <chr>,
## #   HC02_EST_VC05 <chr>, HC02_MOE_VC05 <chr>, HC03_EST_VC05 <chr>,
## #   HC03_MOE_VC05 <chr>, HC04_EST_VC05 <chr>, HC04_MOE_VC05 <chr>,
## #   HC01_EST_VC06 <chr>, HC01_MOE_VC06 <chr>, HC02_EST_VC06 <chr>,
## #   HC02_MOE_VC06 <chr>, HC03_EST_VC06 <chr>, HC03_MOE_VC06 <chr>,
## #   HC04_EST_VC06 <chr>, HC04_MOE_VC06 <chr>, HC01_EST_VC07 <chr>,
## #   HC01_MOE_VC07 <chr>, HC02_EST_VC07 <chr>, HC02_MOE_VC07 <chr>,
## #   HC03_EST_VC07 <chr>, HC03_MOE_VC07 <chr>, HC04_EST_VC07 <chr>,
## #   HC04_MOE_VC07 <chr>, HC01_EST_VC08 <chr>, HC01_MOE_VC08 <chr>,
## #   HC02_EST_VC08 <chr>, HC02_MOE_VC08 <chr>, HC03_EST_VC08 <chr>,
## #   HC03_MOE_VC08 <chr>, HC04_EST_VC08 <chr>, HC04_MOE_VC08 <chr>,
```

```
## #   HC01_EST_VC10 <chr>, HC01_MOE_VC10 <chr>, HC02_EST_VC10 <chr>,
## #   HC02_MOE_VC10 <chr>, HC03_EST_VC10 <chr>, HC03_MOE_VC10 <chr>,
## #   HC04_EST_VC10 <chr>, HC04_MOE_VC10 <chr>, HC01_EST_VC13 <chr>,
## #   HC01_MOE_VC13 <chr>, HC02_EST_VC13 <chr>, HC02_MOE_VC13 <chr>,
## #   HC03_EST_VC13 <chr>, HC03_MOE_VC13 <chr>, HC04_EST_VC13 <chr>,
## #   HC04_MOE_VC13 <chr>, HC01_EST_VC14 <chr>, HC01_MOE_VC14 <chr>,
## #   HC02_EST_VC14 <chr>, HC02_MOE_VC14 <chr>, HC03_EST_VC14 <chr>,
## #   HC03_MOE_VC14 <chr>, HC04_EST_VC14 <chr>, HC04_MOE_VC14 <chr>,
## #   HC01_EST_VC17 <chr>, HC01_MOE_VC17 <chr>, HC02_EST_VC17 <chr>,
## #   HC02_MOE_VC17 <chr>, HC03_EST_VC17 <chr>, HC03_MOE_VC17 <chr>,
## #   HC04_EST_VC17 <chr>, HC04_MOE_VC17 <chr>, HC01_EST_VC18 <chr>,
## #   HC01_MOE_VC18 <chr>, HC02_EST_VC18 <chr>, HC02_MOE_VC18 <chr>,
## #   HC03_EST_VC18 <chr>, HC03_MOE_VC18 <chr>, HC04_EST_VC18 <chr>,
## #   HC04_MOE_VC18 <chr>, HC01_EST_VC19 <chr>, HC01_MOE_VC19 <chr>,
## #   HC02_EST_VC19 <chr>, HC02_MOE_VC19 <chr>, HC03_EST_VC19 <chr>,
## #   HC03_MOE_VC19 <chr>, HC04_EST_VC19 <chr>, ...
```

Table 2

```r
head(as.data.frame(cleaned_data[1]))
```

```
##                 AGE Estimate Margin.of.Error subject_id Year
## 1    16 to 19 years      2.7             0.1          1 2017
## 2    20 to 24 years      8.1             0.2          1 2017
## 3    25 to 44 years     45.6             0.2          1 2017
## 4    45 to 54 years     22.0             0.2          1 2017
## 5    55 to 59 years      9.4             0.2          1 2017
## 6 60 years and over     12.3             0.2          1 2017
```

Table 3

```r
head(as.data.frame(cleaned_data[16]))
```

```
##   Estimate .Margin.of.Error Year subject_id
## 1  3320895            14865 2017          1
## 2  2204896            17843 2017          2
## 3   304964            11011 2017          3
## 4   424417             9334 2017          4
## 5  3249197            16368 2016          1
## 6  2142125            16813 2016          2
```

Machine Learning and Data Exploration

When working with large quantities of data, it's crucial you get a broad view of the data before diving into specific trends and correlations. With so much data to work with, we will be split the data analysis and machine learning parts of this tutorial into two parts. Here in Part I, we will do conduct a broad analysis of our data to get a gist of the overall trend and to see if there's anything worth diving deeper into.

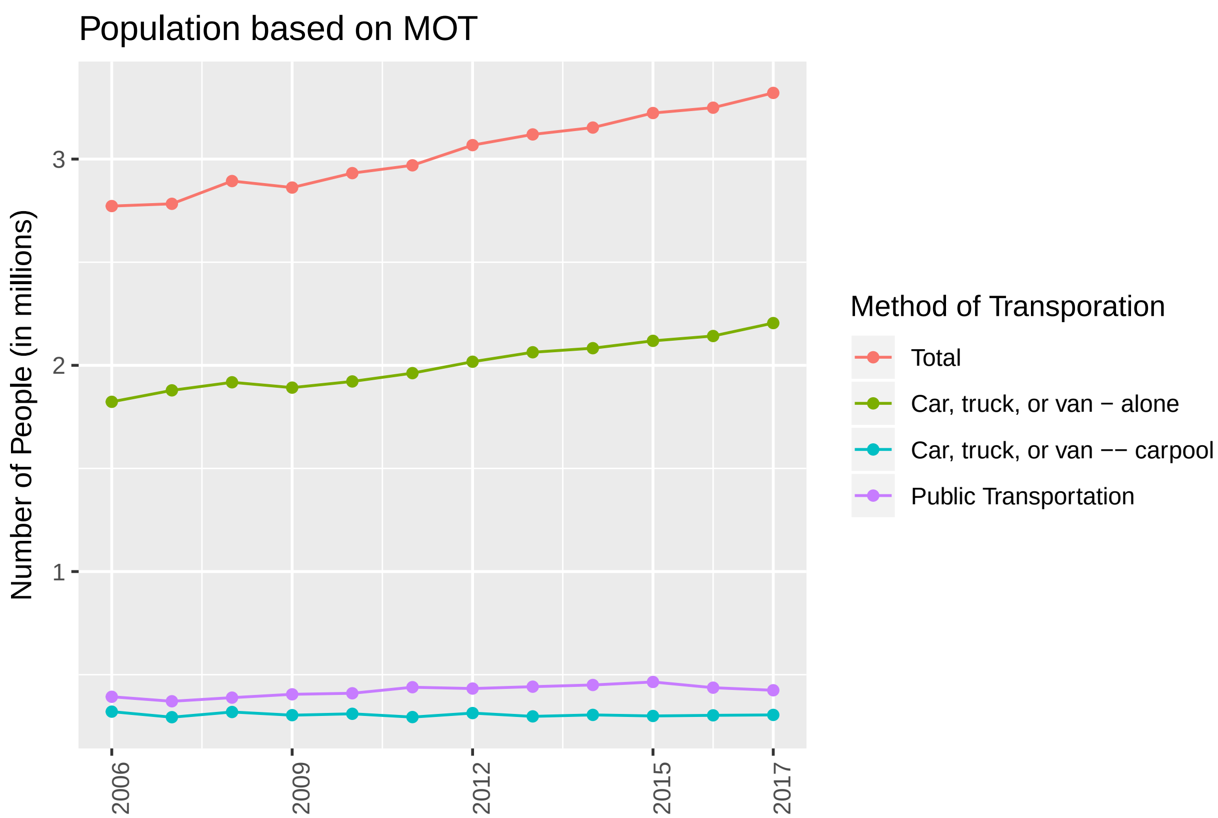
The first step in the general analysis will be to plot the number of people commuting (by the method of transportation) using the ggplot2 package and conduct a linear regression using the lm() function.

You can read more about the ggplot2 package here: https://ggplot2.tidyverse.org/

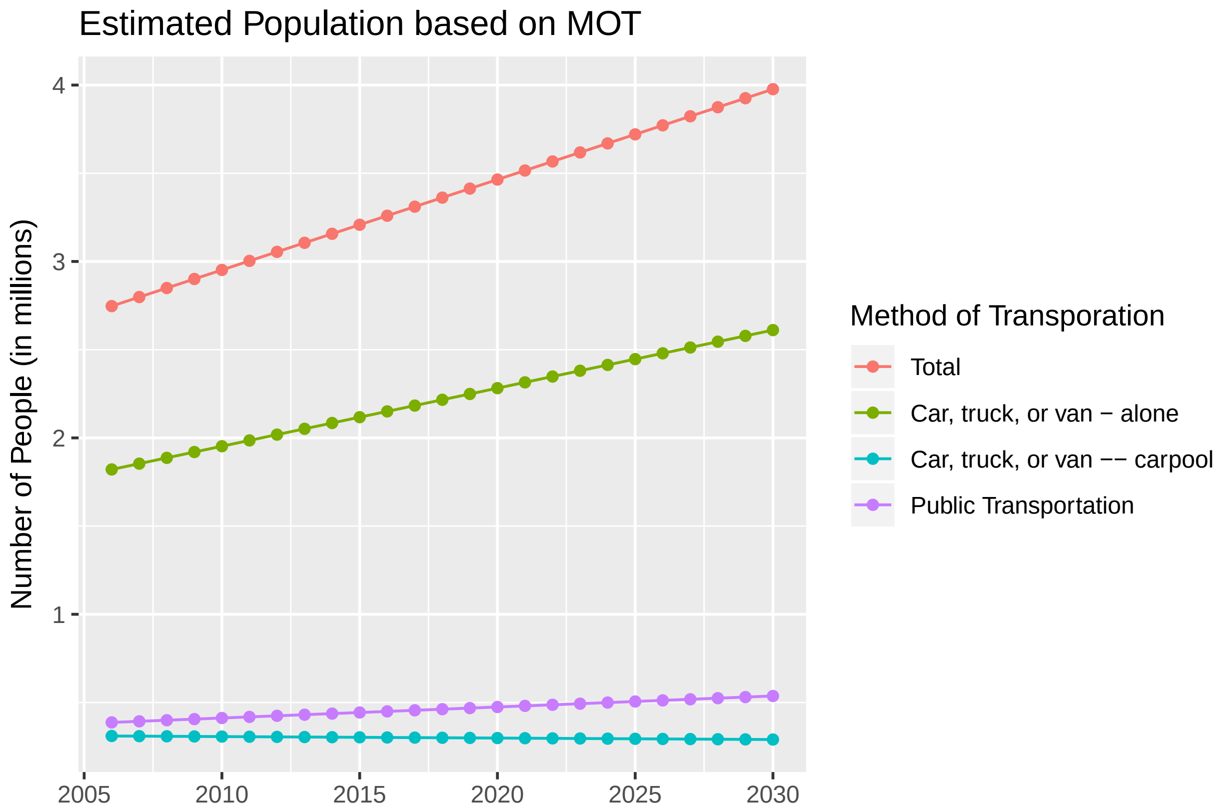
You can also use this useful cheat sheet to learn the syntax of different plots: https://www.rstudio.com/wp-content/uploads/2018/08/data-visualization-2.1.png

You can read more about linear regression in R and lm() function here: https://www.r-bloggers.com/r-tutorial-series-simple-linear-regression/

```
#Total estimate

fit_1 <-
  lm(Estimate ~ Year * as.factor(subject_id), data = cleaned_data[[16]])

linear_pred_mode <- cleaned_data[[16]]

linear_predicted_population <-
  linear_pred_mode %>% ggplot(mapping = aes(
    y = Estimate,
    x = Year,
    color = as.factor(subject_id)
  )) + scale_color_discrete(
    name = "Method of Transporation",
    labels = c(
      "Total",
      "Car, truck, or van - alone",
      "Car, truck, or van -- carpool",
      "Public Transportation"
    )
  ) + geom_line()  + scale_x_continuous(breaks = c(2006, 2009, 2012, 2015, 2017)) +
  theme(axis.text.x = element_text(angle = 90)) + ylab("Number of People (in millions)") +
  xlab("") + scale_y_continuous(
    label = function(x)
      format(x / 1000000)
  ) + ggtitle("Population based on MOT") + geom_point()

linear_predicted_population
```




```r
#Linear Regression
new_dat <-
  summarize_at(linear_pred_mode, vars(Estimate, .Margin.of.Error), mean)
new_dat <-
  cbind(expand.grid(Year = seq(2006, 2030, 1), subject_id = 1:4), new_dat)

new_dat$Estimate <- predict(fit_1, new_dat)

ggplot(new_dat,
       aes(
         x = Year,
         y = Estimate,
         color = factor(subject_id),
         group = subject_id
       )) + scale_color_discrete(
         name = "Method of Transporation",
         labels = c(
           "Total",
           "Car, truck, or van - alone",
           "Car, truck, or van -- carpool",
           "Public Transportation"
         )
       ) + ylab("Number of People (in millions)") + xlab("") + scale_y_continuous(
         label = function(x)
           format(x / 1000000)
       ) + ggtitle("Estimated Population based on MOT ") + geom_point() + geom_line()
```



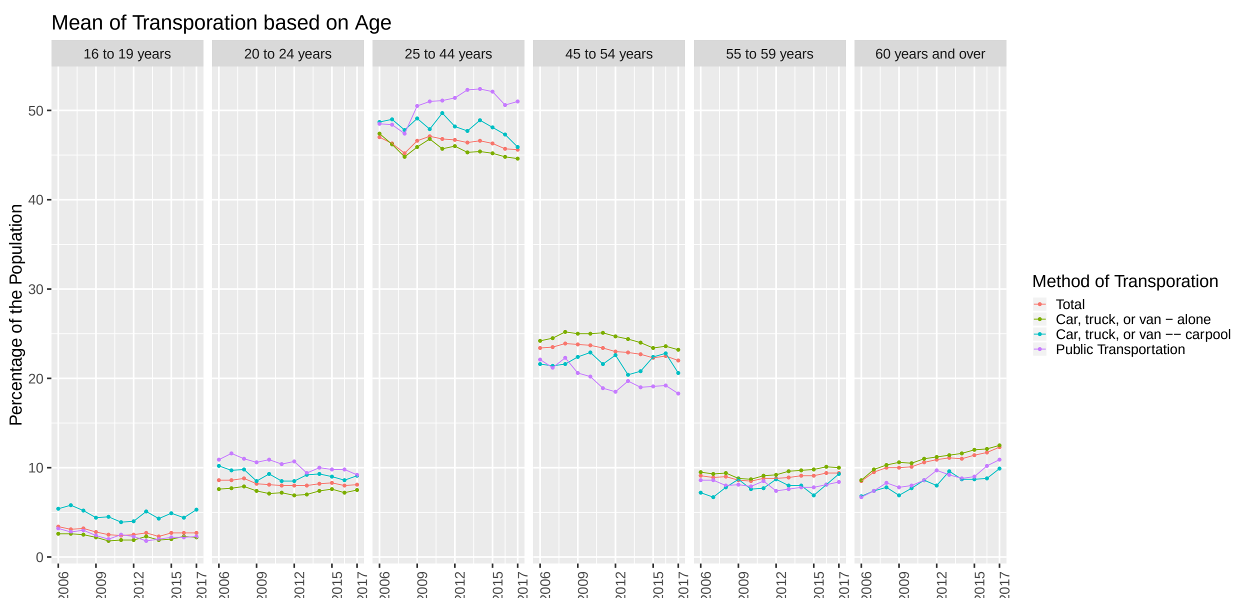
One of the fundamental principles of programming is to code efficiently by using functions to avoid rewriting the same blocks of code. This principle is paramount in data science where we will often be executing the same commands/calculations on different tables.

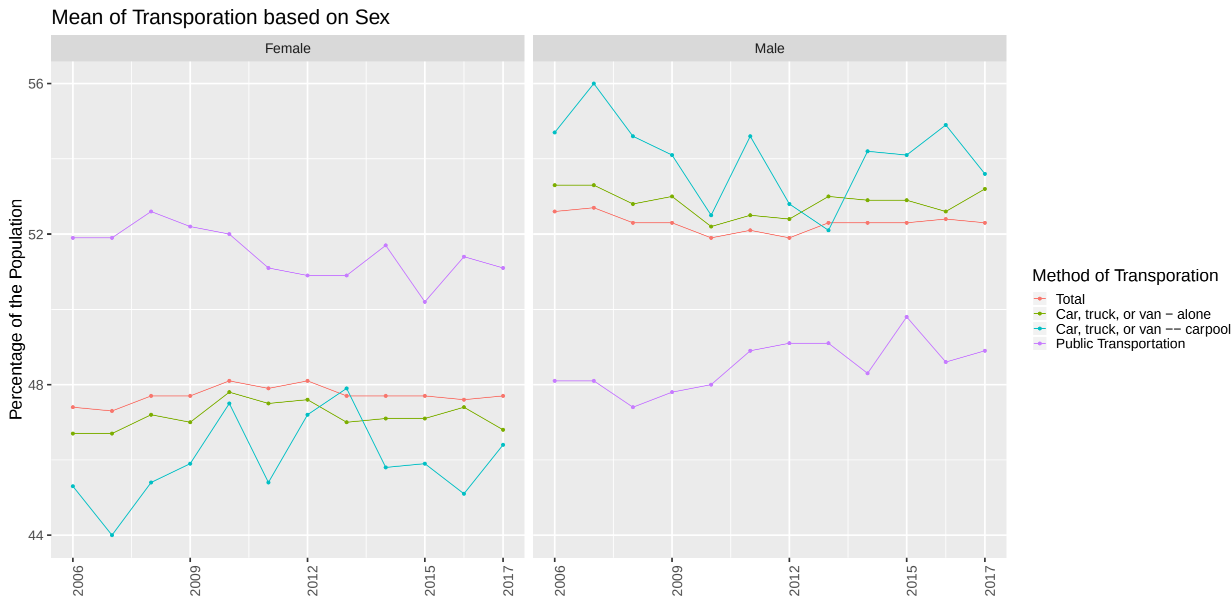
Now that we have conducted an analysis and linear regression on the number of commuters across all attributes, we will do a broad analysis of the effect different attributes have on the percentage of commuters. The perfect tool for this task is faceting, which allows us to visualize data across many different categories. Faceting will give us an insight into general trends among the plethora of attributes, where we can hopefully find some interesting association worth examining further.

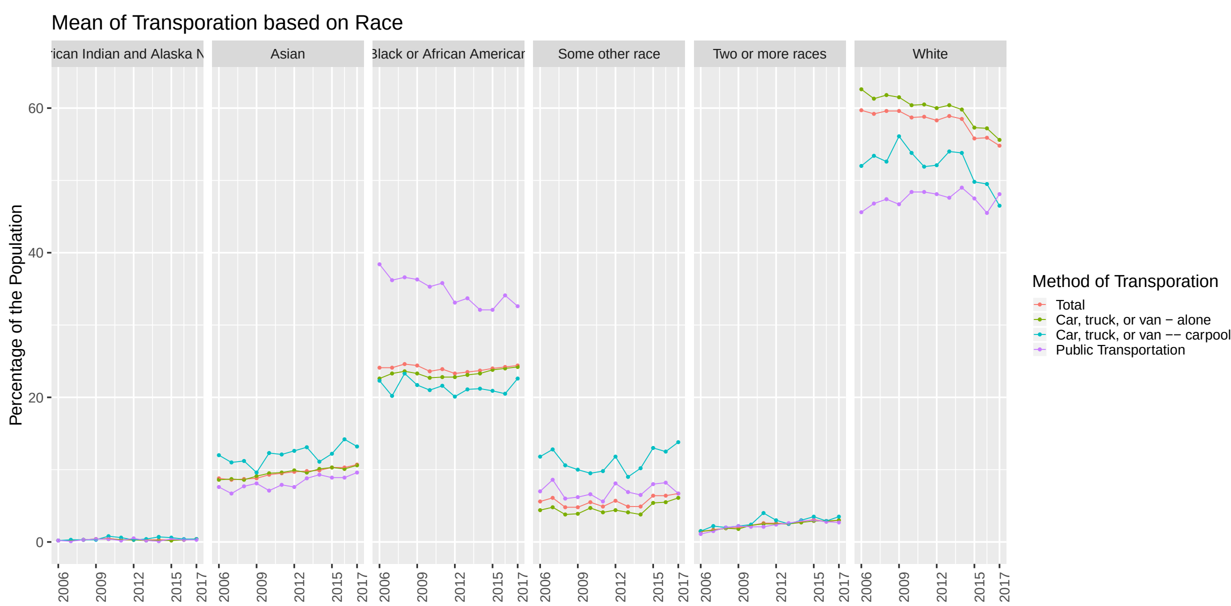
You can read further on faceting here: https://ggplot2.tidyverse.org/reference/facet_grid.html

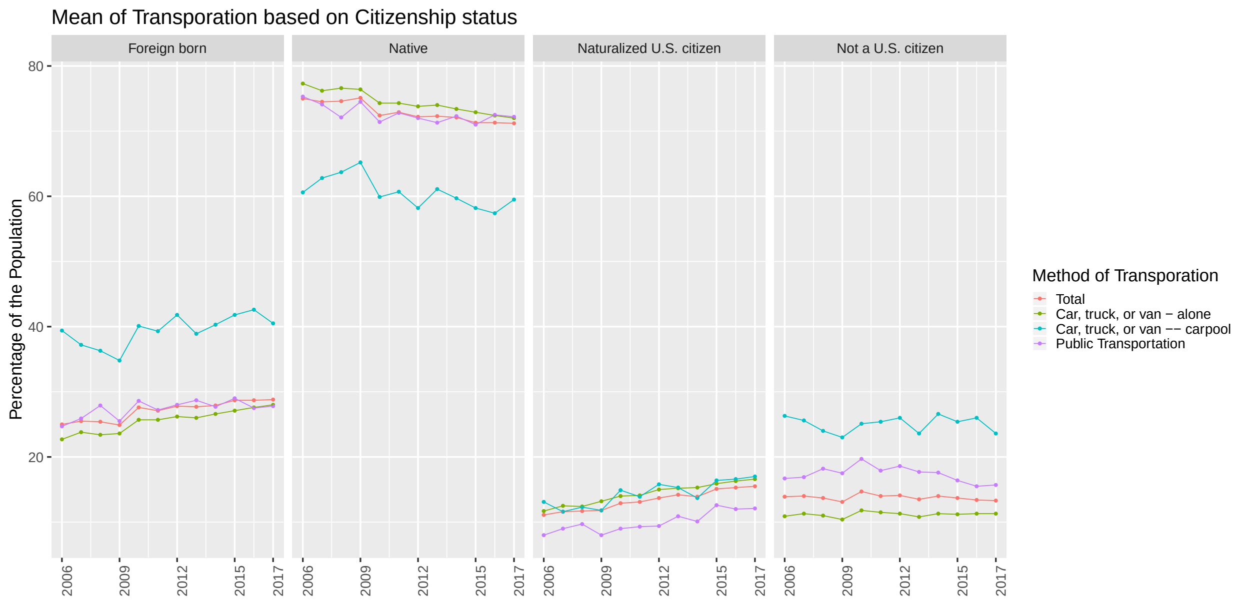
```
create_facet <- function(passed_frame) {
  param <- colnames(passed_frame)[1]

  title <- param
  title <- str_replace_all(title, "\\.", " ")
  title <- str_replace_all(title, "[A-Z]", tolower)
  title <- str_replace_all(title, "^[a-z]", toupper)

  facet <-
    passed_frame %>% ggplot(mapping = aes(
      y = Estimate,
      x = Year,
      color = as.factor(subject_id)
    )) + geom_point() + facet_grid(param, as.table = TRUE) +
    scale_color_manual(values =
```

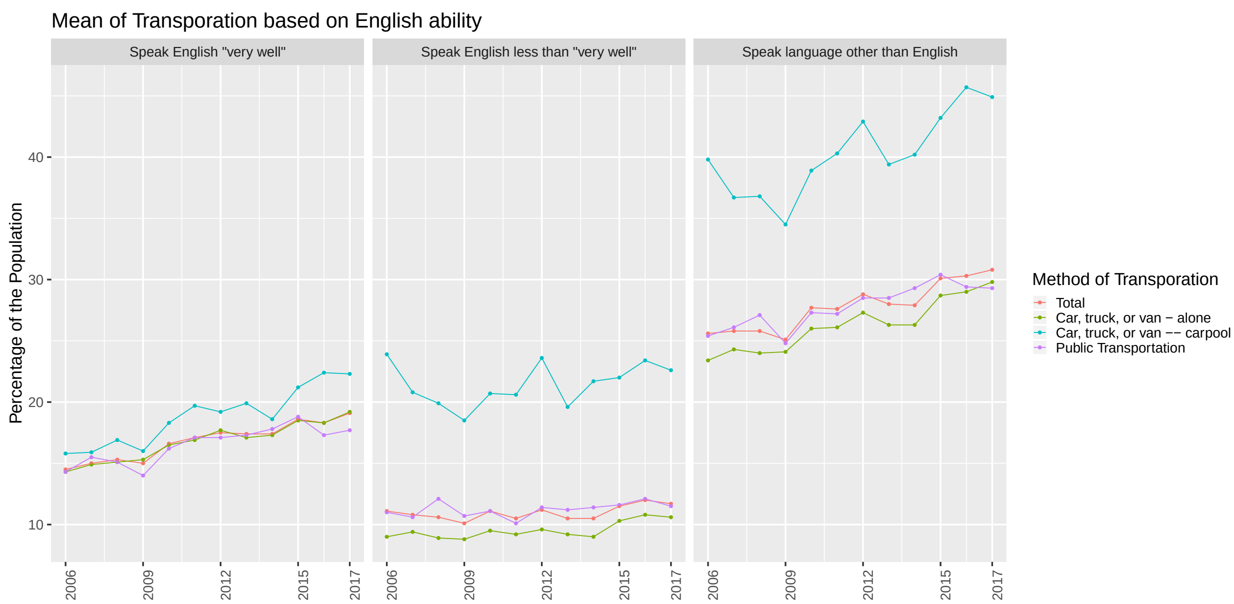
```r
    c("#999999", "#E69F00", "#56B4E9", "#000000")) +
    ggtitle(paste0("Mean of Transporation based on ", title)) +
    scale_color_discrete(name = "Method of Transporation",
                         labels = c( "Total", "Car, truck, or van - alone",
                                     "Car, truck, or van -- carpool",
                                     "Public Transportation")) +
    geom_line() +
    theme_grey(base_size = 20) +
    scale_x_continuous(breaks = c(2006, 2009, 2012, 2015, 2017)) +
    theme(axis.text.x = element_text(angle = 90)) +
    ylab("Percentage of the Population") +
    xlab("")

  facet

}

facet_data <- vector("list", 15)

for (i in 1:6) {
  d1 <- create_facet(cleaned_data[[i]])
  print(d1)

}
```

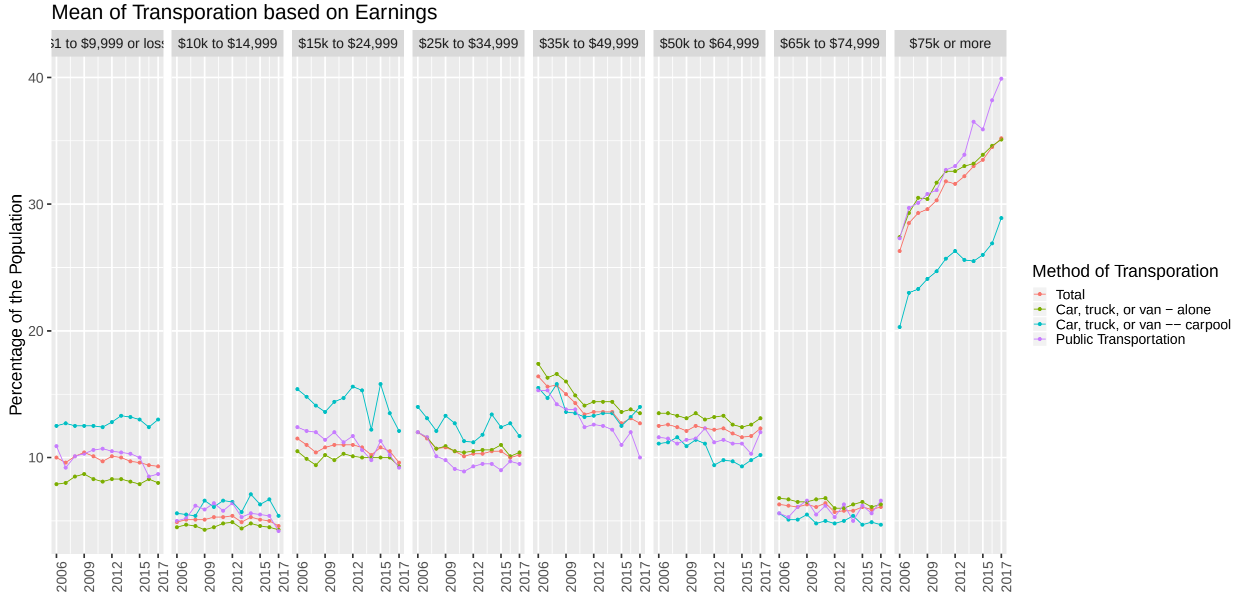



Mean of Transporation based on Sex
Female
Male
56
52
48
44
Percentage of the Population
2006
2009
2012
2015
2017
Method of Transporation
Total
Car, truck, or van − alone
Car, truck, or van −− carpool
Public Transportation


Mean of Transporation based on Race
ican Indian and Alaska N
Asian
Black or African American
Some other race
Two or more races
White
60
40
20
0
Percentage of the Population
2006
2009
2012
2015
2017
Method of Transporation
Total
Car, truck, or van − alone
Car, truck, or van −− carpool
Public Transportation

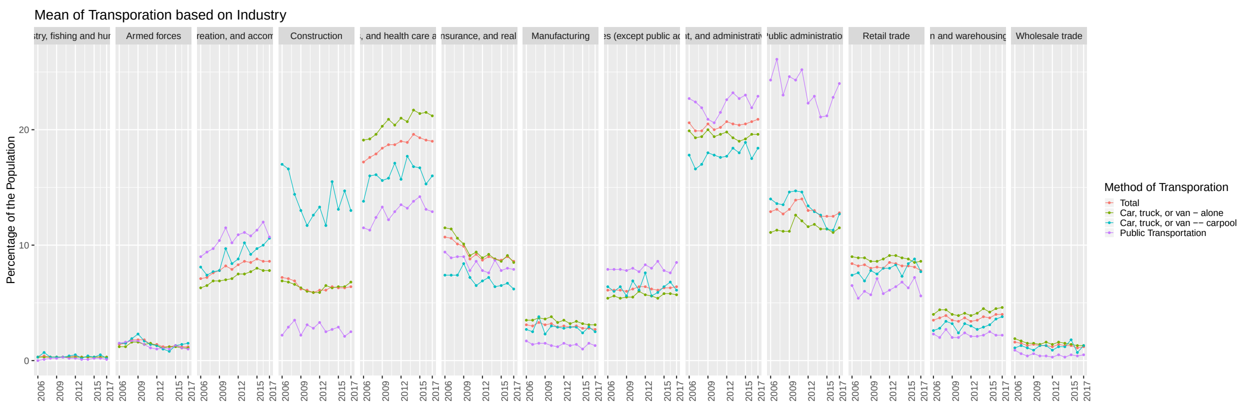

Mean of Transporation based on Citizenship status
Foreign born
Native
Naturalized U.S. citizen
Not a U.S. citizen
Percentage of the Population
80
60
40
20
2006
2009
2012
2015
2017
Method of Transporation
Total
Car, truck, or van – alone
Car, truck, or van –– carpool
Public Transportation


Mean of Transporation based on English ability
Speak English "very well"
Speak English less than "very well"
Speak language other than English
Percentage of the Population
40
30
20
10
2006
2009
2012
2015
2017
Method of Transporation
Total
Car, truck, or van – alone
Car, truck, or van –– carpool
Public Transportation

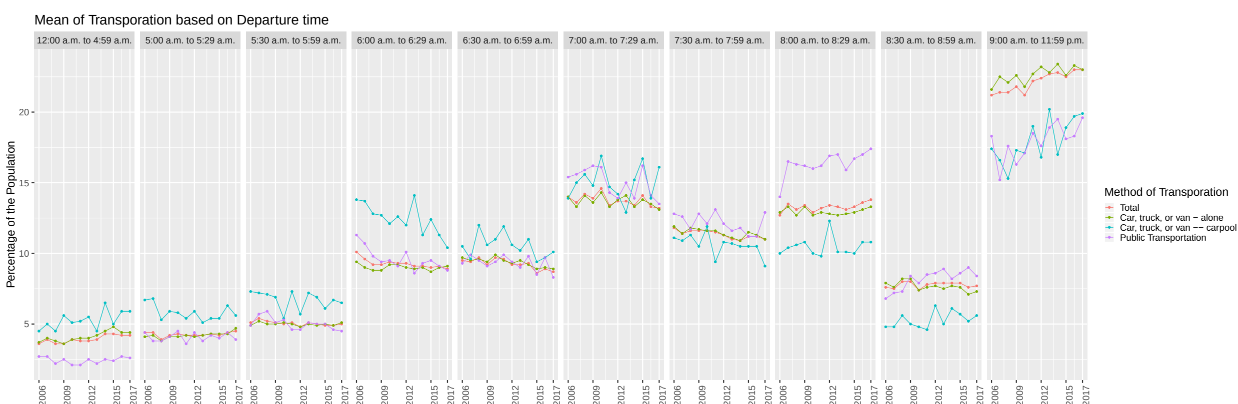



```r
for (i in 10:11) {
  d1 <- create_facet(cleaned_data[[i]])
  print(d1)

}
```

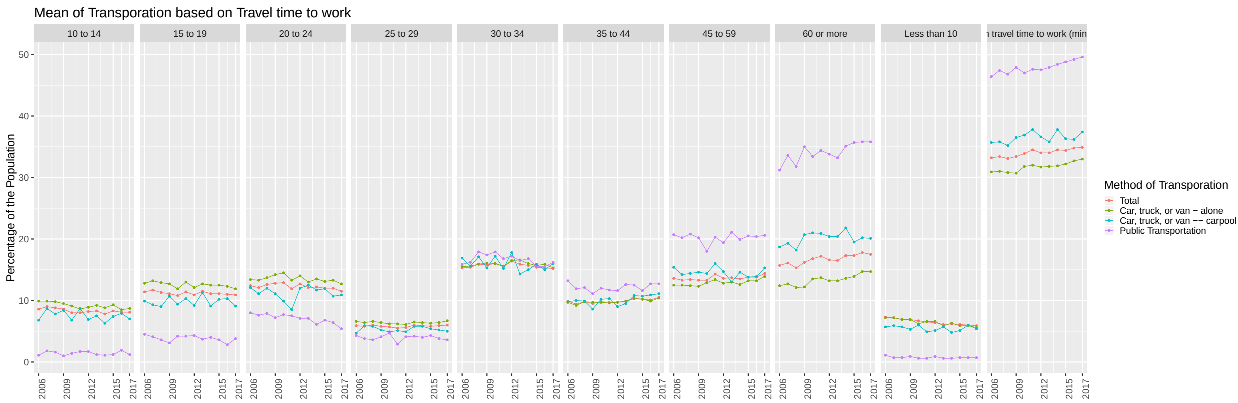

```
create_facet(cleaned_data[[9]])
```




```
create_facet(cleaned_data[[12]])
```



```r
create_facet(cleaned_data[[13]])
```



Hypothesis testing is the formal method we use to accept or reject a hypothesis. Depending on the problem at hand, we will use a specific formula to calculate the Test Statistic which we use to calculate the p-value. We compare the p-value to our level of significance level to decide whether to reject or not reject our hypothesis (referred to as the null hypothesis).

For the purposes of this tutorial, we will be testing the proportion of the population which uses a particular method of transportation to work over our time period, 2006 to 2017. The null hypothesis will be that the proportion remains the same across all the years.

For more information about hypothesis testing visit: https://stattrek.com/hypothesis-test/hypothesis-testing.aspx

Applying the principle of efficient coding discussed above, we will be writing a function which will take two vectors, "Observed" and "Expected" and return a chi-square test statistic. Although R has a built-in function which can perform the chi-square test, it does not provide the same statistical insight as writing the method yourself.

For more information about the chi-square test you can visit: https://www.spss-tutorials.com/chi-square-independence-test/

```r
# chi sqr hypothesis test function which returns the Chi Square Value
chisqr <- function(observed, expected) {
  sum <- 0
  for (num in  1:length(observed)){
    res <- observed[num] - expected[num]
    res <- (res^2)/expected[num]
    sum <- sum + res
  }

  return(sum)
}
```

We will be conducting three tests for the three different methods of transportation to work, driving alone, carpool, and public transport. In order to conduct the test first you will have to retrieve the correct table from the set of tables in cleaned_data, and then clean it up a bit so it only includes the relevant information. Next, we must add a new column, "Expected", which is the mean of all of the Observed values. Finally, we can use the chi_sqr() function we defined to calculate the test-statistic and then feed that into pchisq() function to get the p-value.

```r
# number of nows in data table 16 (total summary)
num_rows <- nrow(data.frame(cleaned_data[[16]]))

# Testing proportion that drove alone from 2006 to 2017
drove_alone <- data.frame(cleaned_data[[16]]) %>%
  select(1,3) %>%
  slice(seq(2, num_rows, by=4)) %>%
  mutate(Expected = mean(Estimate))

alone_res <- chisqr(drove_alone$Estimate, drove_alone$Expected)
alone_res
```

```
## [1] 79661.77
```

```r
alone_pval <- pchisq(alone_res, df=nrow(drove_alone)-1, lower.tail=FALSE)
alone_pval
```

```
## [1] 0
```

```r
# Testing proportion that carpooled alone from 2006 to 2017
drove_carpool <- data.frame(cleaned_data[[16]]) %>%
  select(1,3) %>%
  slice(seq(3, num_rows, by=4)) %>%
  mutate(Expected =  mean(Estimate))

carpool_res <- chisqr(drove_carpool$Estimate, drove_carpool$Expected)
carpool_res
```

```
## [1] 2842.952
```

```r
carpool_pval <- pchisq(carpool_res, df=nrow(drove_carpool)-1, lower.tail=FALSE)
carpool_pval
```

```
## [1] 0
```

```r
# Testing proportion that took public transportation from 2006 to 2017
public <- data.frame(cleaned_data[[16]]) %>%
  select(1,3) %>%
  slice(seq(4, num_rows, by=4)) %>%
  mutate(Expected =  mean(Estimate))

public_res <- chisqr(public$Estimate, public$Expected)
public_res
```

```
## [1] 20469.33
```

```r
public_pval <- pchisq(public_res, df=nrow(public)-1, lower.tail=FALSE)
public_pval
```

```
## [1] 0
```

All three of the tests resulted in a p-value of approximately 0, thus we reject the null hypothesis at a significance level of 0.05 (or any other reasonable significance level) for all three methods of transportation. We can conclude that there is sufficient evidence that there are at least two years for each of the methods of transportation where the proportions of commuters (out of the total) are not equal.

Now it is time to return to the data analysis/machine learning to dive deeper into some interesting trends which can be observed in Part I of the analysis.

First, we will write a function, get_y(), which will take a linear regression model with an x value and return a y value. This will be useful later on.

```
# function which predicts y value based on year
get_y <- function(fit_stats, year) {
  b0 <- fit_stats$estimate[1]
  b1 <- fit_stats$estimate[2]
  x <- year
  y <- b0 + (b1*x)
  return (y)
}
```

Next, we will write two more functions, graph_model() and model_stats(). graph_model() will take a table and return a line plot of the data with a linear regression line calculated using the lm() function we previously used. model_stats() will also perform a linear regression but instead return a tidied version of the linear regression model statistics.

```
# function which returns a ggplot2 represnetation of the table with the linear
# regression graphed
graph_model <- function(table) {

  model_fit <- lm(Estimate~Year, data=table)

  pred_model <- table %>% ggplot(mapping = aes(
      y = Estimate,
      x = Year
    )) + geom_line()  +
    scale_x_continuous(breaks = c(2006, 2009, 2012, 2015, 2017)) +
    theme(axis.text.x = element_text(angle = 90)) +
    ylab("Percentage of Population") + xlab("Year") +
    geom_point() +
    geom_smooth(method=lm)

  return (pred_model)

}

# function which tidies the linear regression statistics
model_stats <- function(table) {

  model_fit <- lm(Estimate~Year, data=table)

  fit_stats <- model_fit %>%
    tidy()

  return (fit_stats)
}
```

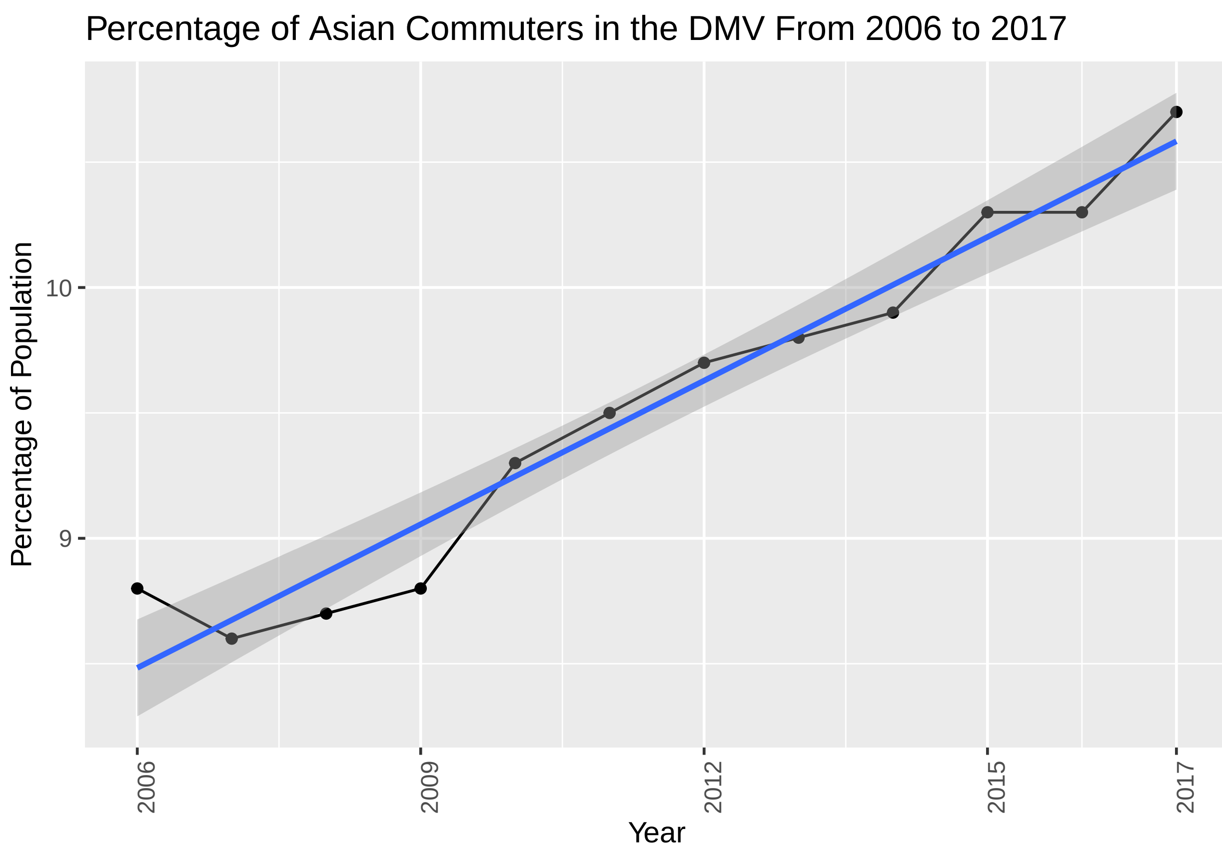
Two interesting trends we noticed from the general analysis performed earlier were the increase in the percentage of Asian Americans and the decrease in the percentage of Whites commuting from 2006 to 2017. In order to further examine these trends, we will graph their respective tables individually and perform a linear regression on each. This will give us greater insight into their particular trends.

To perform the linear regressions, we must retrieve the race attribute table from the cleaned_data set and slice it so it excludes the data for the specific methods of transportation, as we are only interested in the total numbers of commuters across all methods of transportation. Then we can use the functions defined above to plot the data and perform a linear regression.

```
num_rows_race <- nrow(data.frame(cleaned_data[[3]]))

#  Asian American

asian_total <- data.frame(cleaned_data[[3]]) %>%
  select(2,5) %>%
  slice(seq(2, num_rows_race, by=24))

asian_total
```

```
##    Estimate Year
## 1      10.7 2017
## 2      10.3 2016
## 3      10.3 2015
## 4       9.9 2014
## 5       9.8 2013
## 6       9.7 2012
## 7       9.5 2011
## 8       9.3 2010
## 9       8.8 2009
## 10      8.7 2008
## 11      8.6 2007
## 12      8.8 2006
```

```
asian_model <- graph_model(asian_total) +
  ggtitle("Percentage of Asian Commuters in the DMV From 2006 to 2017")
asian_model
```




```
asian_fit_stats <- model_stats(asian_total)
asian_fit_stats
```

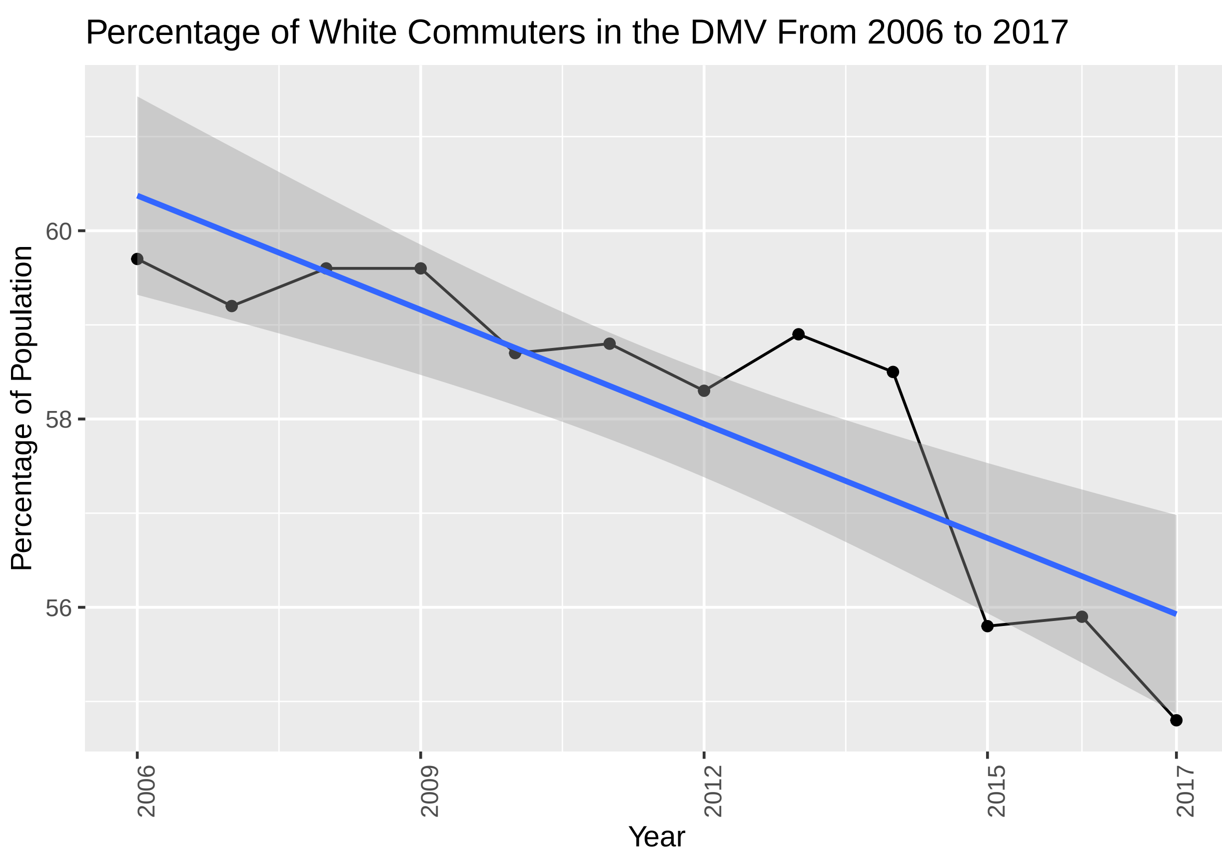
```
## # A tibble: 2 x 5
##   term        estimate std.error statistic      p.value
##   <chr>          <dbl>     <dbl>     <dbl>        <dbl>
## 1 (Intercept) -374.       26.9         -13.9 0.0000000703
## 2 Year           0.191     0.0133      14.3 0.0000000553
```

```
# White

white_total <- data.frame(cleaned_data[[3]]) %>%
  select(2,5) %>%
  slice(seq(5, num_rows_race, by=24))

white_total
```

```
##    Estimate Year
## 1      54.8 2017
## 2      55.9 2016
## 3      55.8 2015
## 4      58.5 2014
## 5      58.9 2013
## 6      58.3 2012
## 7      58.8 2011
```

```
## 8       58.7 2010
## 9       59.6 2009
## 10      59.6 2008
## 11      59.2 2007
## 12      59.7 2006
```

```r
white_model <- graph_model(white_total) +
  ggtitle("Percentage of White Commuters in the DMV From 2006 to 2017")
white_model
```




```r
white_fit_stats <- model_stats(white_total)
white_fit_stats
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  871.     147.          5.95 0.000142
## 2 Year          -0.404    0.0728     -5.55 0.000244
```

From the linear regression fit statistics, we can see the slope for each model. For Asian Americans, the model had a slope of 0.19, meaning the percentage of Asian American commuters in the DMV has increased by about 0.19 per year on average from 2006 to 2017. For Whites, the model had a slope of -0.40, meaning the percentage of White commuters in the DMV has decreased by about 0.40 per year on average from 2006 to 2017.

Finally, we can use the get_y() function we defined above to compare the fit of these two models by comparing their residual sum of squares (RSS). The goal of a linear regression model is to minimize RSS, therefore comparing their RSS is a useful way to compare the model's utility.

We will calculate the RSS the two races by using the formula residual = (observed_y - model_y) and then taking the sum squares of the residuals.

For further reading on residual sum of squares visit: https://www.tutorialspoint.com/statistics/residual_sum_of_squares.htm

```
asian_total <- asian_total %>%
  mutate(Model_Val = get_y(asian_fit_stats, Year)) %>%
  mutate(Residual = (Estimate - Model_Val)^2)

asian_rss <- sum(asian_total$Residual)
asian_rss
```

```
## [1] 0.2548485
```

```
white_total <- white_total %>%
  mutate(Model_Val = get_y(white_fit_stats, Year)) %>%
  mutate(Residual = (Estimate - Model_Val)^2)

white_rss <- sum(white_total$Residual)
white_rss
```

```
## [1] 7.587483
```

The RSS for the Asian American regression model was significantly lower then the RSS for the White regression model, therefore we can conclude that the Asian American linear regression model is a better fit.

It is possible to continue analysis on this data set ad nauseam but we will end it here as we believe the above tools, along with the linked materials, are sufficient to enable you to effectively perform the steps of the data science pipeline. If you would like to perform your own analysis, feel free to use this data set or find a new data set on sites such as: https://www.kaggle.com/data sets

Thank you for reading our data science tutorial.

C