SEJ Analytix: Pre-Process State Traffic Volume in 2018, 2019, and 2020

Imputation for missing days. Make counts per day and per week

Emanuela Ene

#Clear the space

```
rm(list = ls())
```

Introduction and Data

We will be analyzing the Daily Traffic Volume at the FTS stations in the US on a temporal scale from Jan.1st 2019 to Dec. 31st 2020 ## Download

Download transportation data from the Bureau of Transportation Statistics (https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/) is in its original database format as it is collected through the FHWA Travel Monitoring Analysis System (TMAS).

- The TMG (2001) specified fixed width volume data format as listed below

Field Columns Length Description

1 Record Type
2-3 2 FIPS State Code
4-5 2 Functional Classification
6-11 6 Station Identification
12 1 Direction of Travel
13 1 Lane of Travel
14-15 2 Year of Data
16-17 2 Month of Data
18-19 2 Day of Data
20 1 Day of Week
21-25 5 Traffic Volume Counted, 00:01 - 01:00
26-30 5 Traffic Volume Counted, 01:01 - 02:00
31-35 5 Traffic Volume Counted, 02:01 - 03:00
36-40 5 Traffic Volume Counted, 03:01 - 04:00
41-45 5 Traffic Volume Counted, 04:01 - 05:00
46-50 5 Traffic Volume Counted, 05:01 - 06:00
51-55 5 Traffic Volume Counted, 06:01 - 07:00
56-60 5 Traffic Volume Counted, 07:01 - 08:00
61-65 5 Traffic Volume Counted, 08:01 - 09:00
66-70 5 Traffic Volume Counted, 09:01 - 10:00
71-75 5 Traffic Volume Counted, 10:01 - 11:00
76-80 5 Traffic Volume Counted, 11:01 - 12:00
81-85 5 Traffic Volume Counted, 12:01 - 13:00

86-90 5 Traffic Volume Counted, 13:01 - 14:00
91-95 5 Traffic Volume Counted, 14:01 - 15:00
96-100 5 Traffic Volume Counted, 15:01 - 16:00
101-105 5 Traffic Volume Counted, 16:01 - 17:00
106-110 5 Traffic Volume Counted, 17:01 - 18:00
111-115 5 Traffic Volume Counted, 18:01 - 19:00
116-120 5 Traffic Volume Counted, 19:01 - 20:00
121-125 5 Traffic Volume Counted, 20:01 - 21:00
126-130 5 Traffic Volume Counted, 21:01 - 22:00
131-135 5 Traffic Volume Counted, 22:01 - 23:00
136-140 5 Traffic Volume Counted, 23:01 - 24:00
141 1 Restrictions

In this format, each data field has its unique column with fixed width. The hourly count data for each of the 24 hours in a day takes 24 columns.

The tool asks for a zipped volume file (downloaded from the FHWA Office of Highway Policy Information website (https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/ (https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/)). The tool then converts it to different formats with a daily record or hourly record forms.

```
setwd("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation")
```

Load packages

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tibble' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## Warning: package 'stringr' was built under R version 4.3.3
```

```
## Warning: package 'forcats' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

Read in each *.csv file for traffic volume into an R data file that has all of the road data.

```
d20=read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/Data/all_vol_data_2020.
csv", header=T, all=T)
d19=read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/Data/all_vol_data_2019.
csv", header=T, all=T)
d18=read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/Data/all_vol_data_2018.
csv", header=T, all=T)
```

Data Exploration

Check data structure for each file

```
# str(d18)
# str(d19)
# str(d20)
```

Make map for data exploration across all data sets loaded.

```
data_list<-list(d18=d18, d19=d19,d20=d20)
data_names <- names(data_list)
data_map <- imap(data_list, ~ {
  data <- .x
  name <- .y
  # Perform operations on data
  list(name = name, data = data)
})
```

Find character counts

```
negatives<-tibble()

for (item in data_map) {
  name<-item$name
  cc<-item$data
  cc <- cc %>% dplyr::select(FIPS.State.Code, Year.of.Data  , contains("Traffic.Volume.Counte
d."))
  filtered<-cc[apply(cc, 1, function(row) any(grepl("-", row))), ]
  filtered<-filtered%>% mutate(across(everything(), as.character))
  negatives<-bind_rows(negatives, filtered)
}

occurrences<- table(year=negatives$Year.of.Data, state=negatives$FIPS.State.Code)
print(occurrences)
```

```
##     state
## year  51
##   18 933
##   19 909
##   20 900
```

Find the zero count stations

```
filtered<-d20%>%filter(FIPS.State.Code==23, Month.of.Data==12, Day.of.Data%in%c(5, 6, 12, 17))
#print(filtered)
d20 <- d20 %>%filter(!(Month.of.Data == 12 & Day.of.Data %in% c(5, 6, 12, 17)))
```

Replace the ‘000-1’ count by zero, and convert all traffic count columns to numeric.

```
library(purrr)

# Define function
process_dataset <-function(dataset){
  # select and process traffic counts
  cc <- dataset %>% dplyr::select(contains("Traffic.Volume.Counted."))
  cc<-cc %>%mutate_all(~ ifelse(. == "000-1", "00000", .))
  cc <- cc %>% mutate(across(contains("Traffic.Volume.Counted."), as.integer))

  # make daily counts per station
  dailyTraffic<-cc%>%mutate(dailyCount = rowSums(across(contains("Traffic.Volume.Counted."))))
  dailyTraffic<-dailyTraffic%>% dplyr::select(- contains("Traffic.Volume.Counted."))

  # select descriptive columns
  simple_data<-dataset %>% dplyr::select(- contains("Traffic.Volume.Counted."), -Unknown, -Lane.
of.Travel, -Restrictions, -Source.File)

  # make counts per station per calendar day
  processed_data<-bind_cols(simple_data, dailyTraffic)
  processed_data<-processed_data%>%mutate(date = as.Date(paste(2000+Year.of.Data, Month.of.Data,
Day.of.Data, sep = "-")))
}

#make list for processed datasets
corrected_data<-list()

for (item in data_map) {
  name<-item$name
  original_data<-item$data
  corrected_data[[name]]<- process_dataset (original_data)
   }
#corrected_data$d18
#corrected_data$d19
#corrected_data$d20
```

Remove the original data files.

```
rm(d18, d19, d20)
```

Data Imputation

Make Daily Counts and Impute Missing Days

```
combined<-bind_rows(corrected_data$d18, corrected_data$d19,corrected_data$d20)
allStateCodes<-unique(combined$FIPS.State.Code)
```

Make daily counts per state

```
library(lubridate)
#make day.of.week sum by grouping by Day.of.Week, for each year
daySums <- combined %>% group_by(FIPS.State.Code, Year.of.Data, Month.of.Data, Day.of.Data) %>%
summarize(stateTraffic=sum(dailyCount)/1000000)
```

```
## `summarise()` has grouped output by 'FIPS.State.Code', 'Year.of.Data',
## 'Month.of.Data'. You can override using the `.groups` argument.
```

```
daySums_dated<-daySums%>%mutate(date = make_date(year =Year.of.Data+2000 , month = Month.of.Dat
a, day = Day.of.Data))
daySums_dated<-daySums_dated%>%mutate(Day.of.Week=wday(date), Week.of.Year = isoweek(date))
#summary(daySums_dated)
```

Make weekly counts per year per state

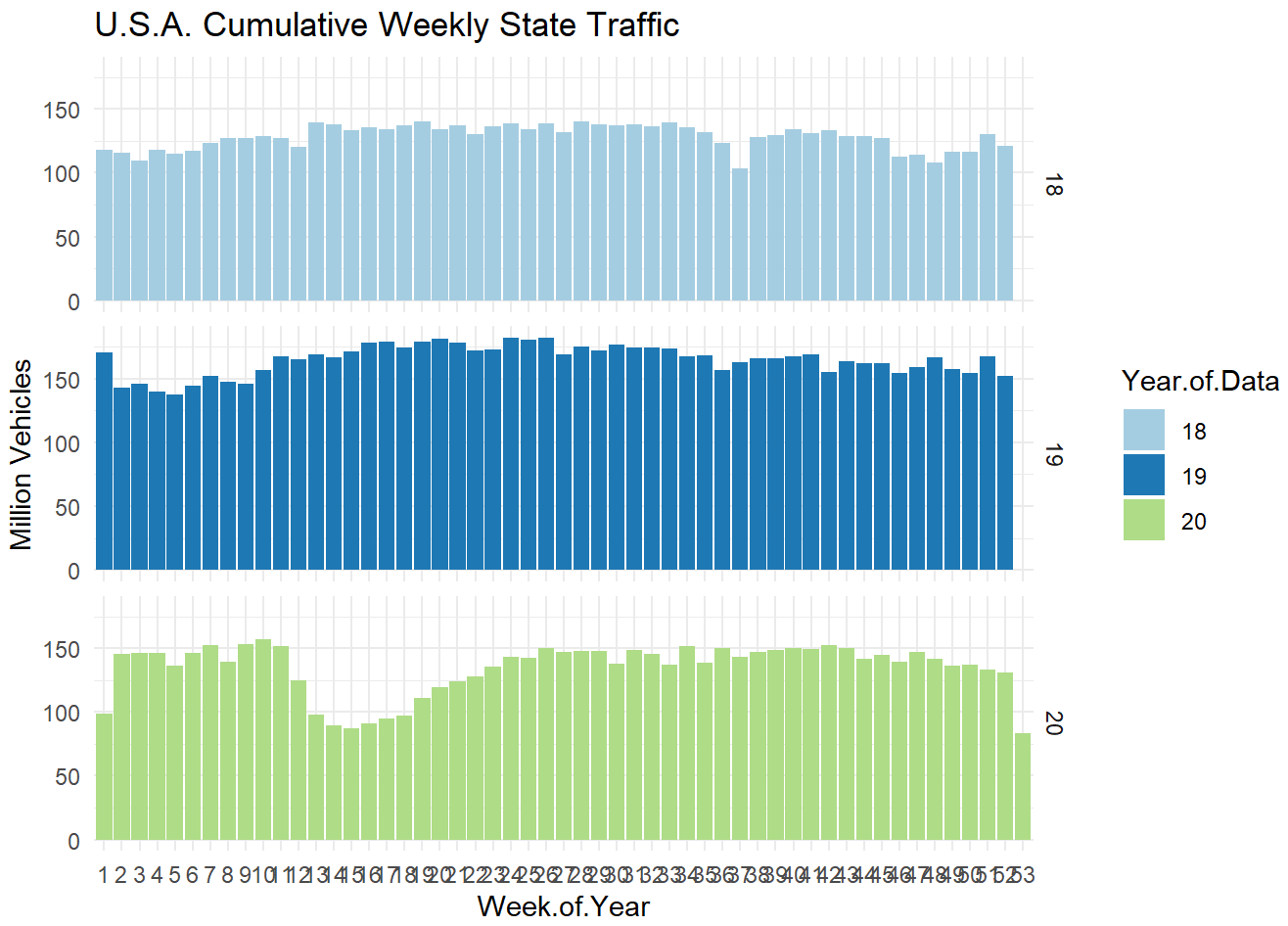
```
#make week sum by grouping by Year.of.Data,  at state level
state_week_sums <- daySums_dated%>% group_by(FIPS.State.Code,Year.of.Data, Week.of.Year  ) %>% s
ummarize(stateCount_Week = sum(stateTraffic))
```

```
## `summarise()` has grouped output by 'FIPS.State.Code', 'Year.of.Data'. You can
## override using the `.groups` argument.
```

```
#summary(state_week_sums)
#rows_with_na_all <- state_week_sums %>%filter(if_all(stateCount_Week , is.na))
#write.csv(state_week_sums,"stateWeeklyTraffic_18_19_20_fhwa_data.csv")
```

Visualize the weekly counts per year.

```
library(RColorBrewer)
# Create the plot
ggplot(state_week_sums, aes(x = factor(Week.of.Year), y = stateCount_Week, fill = factor(Year.o
f.Data))) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_grid(Year.of.Data ~ .)+
  labs(title = "U.S.A. Cumulative Weekly State Traffic",
       x = "Week.of.Year",
       y = "Million Vehicles",
       fill = "Year.of.Data") +
  theme_minimal()+
  scale_fill_brewer(palette = "Paired")
```



Make Median Count for each type of day chosen, on years

```
#make medians for day of the week in a given year and month; for the years with drastic changes in week counts, such as 2020
weeklyAverage<-daySums_dated%>%group_by(FIPS.State.Code, Year.of.Data, Week.of.Year) %>% summarize(week.median=median(stateTraffic))
```

```
## `summarise()` has grouped output by 'FIPS.State.Code', 'Year.of.Data'. You can
## override using the `.groups` argument.
```

```
#make medians for weekdays year long
dayAverage<-daySums_dated%>%group_by(FIPS.State.Code, Year.of.Data, Day.of.Week) %>% summarize(day.median=median(stateTraffic))
```

```
## `summarise()` has grouped output by 'FIPS.State.Code', 'Year.of.Data'. You can
## override using the `.groups` argument.
```

Make dates for each year in the data.

```
# make reference dates for each year
date18<-data.frame(date=seq.Date(from = as.Date("2018-01-01"), to=as.Date("2018-12-31"), by = "d
ay"))
date19<-data.frame(date=seq.Date(from = as.Date("2019-01-01"), to=as.Date("2019-12-31"), by = "d
ay"))
date20<-data.frame(date=seq.Date(from = as.Date("2020-01-01"), to=as.Date("2020-12-31"), by = "d
ay"))
allDates_list <- list(date18 = date18, date19 = date19, date20 = date20)
```

Build functions for Imputation.

```
#function complete_dates
complete_dates <- function(dateX,allDates) {
completed<-tibble()
for (state in allStateCodes){
  data<-dateX%>%filter(FIPS.State.Code == state)
  complete_data <- allDates %>%
    left_join(data, by = "date") %>% mutate(
      FIPS.State.Code = state,
      Year.of.Data = ifelse(is.na(Year.of.Data), year(date)-2000, Year.of.Data),
      Month.of.Data = month(date),
      Day.of.Data = day(date),
      Day.of.Week=wday(date),
      Week.of.Year=isoweek(date)
    )

  completed<-bind_rows(completed,complete_data)
  } # complete dates for states with missing dates

  completed<-completed %>% mutate(FIPS.State.Code=as.integer(FIPS.State.Code), Year.of.Data=as.i
nteger(Year.of.Data), Month.of.Data= as.integer(Month.of.Data), Day.of.Week=as.integer(Day.of.We
ek),Week.of.Year=as.integer(Week.of.Year))
}

#function impute_counts with median weekday count
imputeCount_weekday <- function(dateX,dayAverage) {
complete_data <- dateX %>%
  left_join(dayAverage, by = c("FIPS.State.Code", "Year.of.Data", "Day.of.Week")) %>%
  mutate(stateTraffic = coalesce(stateTraffic, day.median))
}

#function impute_counts with median weekly count
imputeCount_weekAverage <- function(dateX,weeklyAverage) {
complete_data <- dateX %>%
  left_join(weeklyAverage, by = c("FIPS.State.Code", "Year.of.Data", "Week.of.Year")) %>%
  mutate(stateTraffic = coalesce(stateTraffic, week.median))
}

# function to extract the data frame dateXX based on the suffix
get_by_suffix <- function(suffix, data_list) {
  df_name <- str_c("date", suffix)
  data_list[[df_name]]
}
```

Complete daily counts in Prior years 2018 and 2019

```
imputedPrior<-tibble()
for (i in c(18,19)) {
  yearData<-daySums_dated%>%filter(Year.of.Data==i)
  medians<-dayAverage%>%filter(Year.of.Data==i)
  allDates<- get_by_suffix(i, allDates_list )
  complete_year<-complete_dates(yearData,allDates)
  imputed_year<-imputeCount_weekday(complete_year,medians)
  imputedPrior<-bind_rows(imputedPrior, imputed_year)
}
#summary(imputedPrior)
```

Complete daily counts in year 2020

```
  yearData<-daySums_dated%>%filter(Year.of.Data==20)
  medians<-weeklyAverage%>%filter(Year.of.Data==20)
  allDates<- get_by_suffix(20, allDates_list)
  complete_year<-complete_dates(yearData,allDates)
  imputed2020<-imputeCount_weekAverage(complete_year,medians)
  #summary(imputed2020)
  #rows_with_na_all <- imputed2020 %>%filter(if_all(stateTraffic, is.na))
```

Make imputed day counts per year per state

```
imputedDays<-bind_rows(imputedPrior,imputed2020 )%>%select(-day.median,-week.median)
#summary(imputedDays)
write.csv(imputedDays ,"stateDaily_18_19_20_imputed_data.csv")
```

Make weekly counts per year per state

```
#make week sum by grouping by Year.of.Data,  at state level
imputedWeeks <- imputedDays%>% group_by(FIPS.State.Code,Year.of.Data, Week.of.Year  ) %>% summar
ize(stateCount_Week = sum(stateTraffic))
```

```
## `summarise()` has grouped output by 'FIPS.State.Code', 'Year.of.Data'. You can
## override using the `.groups` argument.
```

```
#summary(imputedWeeks)
write.csv(imputedWeeks ,"stateWeekly_18_19_20_imputed_data.csv")
```

Data Visualization

Visualize the weekly counts per year.

```
library(RColorBrewer)
# Create the plot
ggplot(imputedWeeks  , aes(x = factor(Week.of.Year), y = stateCount_Week, fill = factor(Year.of.
Data))) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_grid(Year.of.Data ~ .)+
  labs(title = "U.S.A. Imputed Weekly State Traffic",
       x = "Week.of.Year",
       y = "Million Vehicles",
       fill = "Year.of.Data") +
  theme_minimal()+
  scale_fill_brewer(palette = "Paired")
```