SEJ Analytix: Daily State Traffic Volume in 2020

Will make prior traffic and Numerically Coded COVID19 restrictions

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 cumulative per state on a temporal scale from Jan.1st 2018 to Dec. 31st 2020.

The state cumulative daily traffic was computed by SEJ Analytix from the original data at https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/ (https://www.fhwa.dot.gov/policyinformation/tables/tmasdata/).

For modeling purposes, the data were imputed to obtain non-zero state traffic for each state in each day in each calendar year calendar. The median of the week was employed for the missing days of 2020, separately for each state. The median of the weekday of the month was employed for the missing days of 2018 and 2019, separately for each state. Less than 1% of the original data suffered the imputation process.

The imputation was important for the following states: (1) Maine, missing 4 days in December 2020 due to state historic snow storm; (2) Louisiana, missing 31 days in 2019; (3) Missouri, missing 39 days in 2018.

The structure of the imputed state cumulative daily traffic data is

X - control index
date - full calendar date
FIPS.State.Code - numeric code for state, as in https://www.bls.gov/respondents/mwr/electronic-data-interchange/appendix-d-usps-state-abbreviations-and-fips-codes.htm (https://www.bls.gov/respondents/mwr/electronic-data-interchange/appendix-d-usps-state-abbreviations-and-fips-codes.htm) Year.of.Data - two digit code for year
Month.of.Data - numeric month of the year
Day.of.Data - numeric day of the month
stateTraffic - state cumulative daily traffic, million of vehicles Day.of.Week - numeric day of the week, Sunday=1
Week.of.Year - numeric week of year

The COVID19 state policies were extracted from https://en.wikipedia.org/wiki/U.S._state_and_local_government_responses_to_the_COVID19_pandemic#Initial_pandemic_responses,_including_full_lockdc (https://en.wikipedia.org/wiki/U.S._state_and_local_government_responses_to_the_COVID19_pandemic#Initial_pandemic_responses,_including_full_lockd and https://www.nytimes.com/interactive/2020/us/states-reopen-map-coronavirus.html (https://www.nytimes.com/interactive/2020/us/states-reopen-map-coronavirus.html)

The structure COVID19 state policies data is

FIPS.State.Code - numeric code for state
Year.of.Data - two digit code for year
State - two-letter USPS state code
stateName - fullstate name
Emergency - character, state order date
StartLock - character, state order date; 'N' if no state stay home order
LiftLock - character, state order date; 'NA' if no state stay home order OutState - character, out-of-state travel restriction; 'R' if limited out-of-state travel restrictions CloSch - character, state schools closing
ClsRest - character, bars and restaurants closing at state level
ClsStores - character, non-essential stores closing at state level

We will model the COVID19 state policies numerically as follows:

```
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 become errors
```

Read the *.csv file for traffic volume.

```
daySums<-read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/stateDaily_18_19_20_imputed_data.csv", header=
T, all=T)
covid<-read.table("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/lockdownsUS.txt", header=T, all=T)
```

Make list of unique states

```
allStateCodes<-unique(daySums$FIPS.State.Code)
```

Make prior daily traffic data

The year 2020 has 366 days, the other years of interest have 365 days.

```
#isolate traffic in 2020
dailyTraffic <- daySums %>%filter(Year.of.Data == 20)
#calculate mean traffic, per calendar day, in the prior years
priorTraffic <- daySums  %>%
  filter(Year.of.Data %in% c(18, 19)) %>%
  group_by(FIPS.State.Code, Month.of.Data, Day.of.Data) %>%
  summarise(priorTraffic = mean(stateTraffic))
```

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

```
# create prior traffic for Feb. 29
additional_rows <- priorTraffic %>%
  filter(Month.of.Data == 2, Day.of.Data == 28) %>%
  mutate(Day.of.Data = 29)
priorTraffic <- bind_rows(priorTraffic, additional_rows) %>%
  arrange(FIPS.State.Code, Month.of.Data, Day.of.Data)
# include prior traffic reference column in dailyTraffic
dailyTraffic <- dailyTraffic%>%left_join(priorTraffic, by = c("FIPS.State.Code", "Month.of.Data", "Day.of.Data"))
```

Make offset traffic and normalized traffic

```
dailyTraffic <- dailyTraffic %>% mutate(offTraffic=stateTraffic-priorTraffic, propTraffic=stateTraffic/priorTraffic)
```

Make time series for 2020. When differencing for calculating the growth rate, Jan.1st is the reference and must be dropped from the series of dates.

```
Date<-seq.Date(from = as.Date("2020-01-01"), to=as.Date("2020-12-31"), by = "day")
#make a vector for the lockdown timeline
lock=rep(0,366)
```

Create time-vectors for emergency, startLock, liftLock, outState, clsStores for each state and store them in a 51 column data frame.

```
#make data frame for the days in 2020
tf <- data.frame(
  date = Date)
timeline <- tf %>%
  mutate(
    Month.of.Data = as.integer(format(date, "%m")),
    Day.of.Data = as.integer(format(date, "%d"))
  )
```

Create 12/31/20 as a false date for the states that did not have Lock Down.

```
covid_temp<-covid %>% mutate(StartLock = ifelse(StartLock=='N'|StartLock=='R','12/31/2020',StartLock ), LiftLock = ifelse(i
s.na(LiftLock),'12/31/2020',LiftLock ))
```

Create time series for the daily state Covid policies

```
# Initialize an empty list to store results
covidPolicy <- tibble()

for (state in allStateCodes){
  policy <- covid_temp %>% filter(FIPS.State.Code== state)
  startLock <- strptime(as.character(policy$StartLock), "%m/%d/%Y")
format(startLock, "%Y-%m-%d")
liftLock <- strptime(as.character(policy$LiftLock), "%m/%d/%Y")
format(liftLock, "%Y-%m-%d")
emergencyOrder<- strptime(as.character(policy$Emergency), "%m/%d/%Y")
format(emergencyOrder, "%Y-%m-%d")
travel=policy$OutState
retail=policy$ClsStores
policyState<- timeline %>%  mutate(FIPS.State.Code= state, emer = ifelse((date >=emergencyOrder), 'Y', 'N'), lock = ifelse
((date >=startLock & date <=liftLock), 'Y', 'N'), trv= ifelse((date >=startLock), travel, 'N'),  cls= ifelse((date >=startLo
ck), retail, 'N'))

covidPolicy<- bind_rows(covidPolicy,policyState)
}
```

Make data set for modeling

```
covidState<-left_join(dailyTraffic,covidPolicy,by=c('FIPS.State.Code','Month.of.Data', 'Day.of.Data'))
write_csv(covidState, "categoricalCovid&Traffic_Daily.csv")
allStateCodes<-unique(covidState$FIPS.State.Code)
```

Make Numerical Codes for the COVID19 restrictions at State Level

```
# Initialize an empty list to store results

results<- tibble()

for (n in allStateCodes){

  state<-covidState %>% dplyr::filter(FIPS.State.Code == n)

  # define the numerical scale of the Covid19 restrictions
  size=0.9*max(state$stateTraffic)
  base=0.8*min(state$priorTraffic)

  state<-state%>%dplyr::mutate(emergency=if_else(emer=='N',base,size),
                               lockDown=if_else(lock=='N',base,if_else(lock=='R', 0.5*size, size)),
                               travelBan= if_else(trv=='N',base,if_else(trv=='R', 0.3*size, 0.6*size)),
                               closedStores= if_else(cls=='N',base,if_else(cls=='R', 0.4*size, 0.8*size)) )

  results<- bind_rows(results,state)
}

timelineCovid<-results%>%rename(date=date.x)%>%select(-X, -date.y)
write_csv(timelineCovid, "categorical&numerical_Covid & imputed_TrafficDaily.csv")
numericalCovid<-timelineCovid%>%select(-emer, -lock,-trv,-cls)
```