SEJ Analytix: Time-Series Modeling State Traffic Volume in 2020

COVID19 restrictions at State Level are Numerically Coded

#Clear the space

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

Introduction and Data

We will be analyzing the Daily Traffic Volume and the Daily Proportion of Traffic Volume in the US cumulative per state via Time Series Cross Correlation, on a temporal scale from Jan.1st 2020 to Dec. 31st 2020.

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

The COVID19 state policies were extracted and from
https://en.wikipedia.org/wiki/U.S._state_and_local_government_responses_to_the_COVID19_pandemic
(https://en.wikipedia.org/wiki/U.S._state_and_local_government_responses_to_the_COVID19_pandemic)
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 FIPS.State.Code - numeric and USPS codes for state were extracted from
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)

The Covid19 state policies were numerically coded with reference to the state specific traffic.
The structure of the imputed state cumulative traffic data is

date - character date of 2020
FIPS.State.Code - numeric code for state
Year.of.Data - two digit Month.of.Data - numeric month of the year
Day.of.Data - numeric day of the month
stateTraffic - state cumulative daily traffic, in million vehicles per day
Day.of.Week - numeric day, Sunday=1 Week.of.Year - numeric week, 1 through 53 priorTraffic - arithmetic mean of the daily traffic in 2018 and 2019, in million vehicles per day
offTraffic - difference (stateTraffic-priorTraffic), in million vehicles per day
prop Traffic - ratio (stateTraffic/priorTraffic), dimensionless
emergencyState - size after the emergency order in that state, otherwise base
lockState - size if stay home order; 0.5(size) if restricted; otherwise base
travelOutState - 0.6(size) if out-of-state travel restriction; 0.3(size) if limited out-of-state travel restrictions; otherwise base
clsRetailState - 0.8(size) if bars and restaurants closed at state level; 0.4* size if limited closings; otherwise base

The numerical enconding of the COVID19 policies is specific
size=0.9 of the maximum stateTraffic, in millions of vehicles per day
base=0.8 of the minimum statePrior, in millions of vehicles per day

```
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 the *.csv file for traffic volume.

```
trafficCovid<-read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/categorical&n
umerical_Covid & imputed_TrafficDaily.csv", header=T, all=T)
stateList<-read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/USPS_State_Abbre
viations.csv", header=T, all=T)
```

```
numericalCovid<-trafficCovid%>%select(-emer, -lock,-trv,-cls)
allStateCodes<-unique(numericalCovid$FIPS.State.Code)
```

Make state info and print it

```
for (n in allStateCodes) {
 name <- (stateList %>% dplyr::filter(FIPS.State.Code == n))$stateName
codeFIPS<-n
abbr<- (stateList %>% dplyr::filter(FIPS.State.Code == n))$stateCode

# if (n %in% allStateCodes) {
#   print(paste("FIPS code", codeFIPS,"The state name is", name, " and its abbreviation code i
s",abbr ))
# } else {
#   print("This state code is not included in our database. Please select another code.")
# }

}
```

Build functions for lagged CC2 time series

```
# function diagnosticPlots to make diagnostic plots
diagnosticPlots<-function(data1, data2) {
  # plot data1 and its growth, data2 and its growth
par(mfrow=c(2,2))
tsplot(data1, main=paste(colNames(data1)))
tsplot(diff(traffic20), main = paste(colNames(data1), "Growth") ,col=4)
tsplot(trafficPrior, main=paste(colNames(data2)))
tsplot(diff(trafficPrior), main = paste(colNames(data2), "Growth") ,col=4)
}


# function findLag to find the lag between two time series
findLag<-function(data1, data2){
ccf_result<-as.data.frame(ccf2(diff(data1), diff(data2), max.lag = 10, plot = FALSE, type = c("c
orrelation")))
# Extract the CCF values and lags
ccf_values <- ccf_result$CCF
lags <- ccf_result$LAG
#find the maximal cross-correlation, in abs. value
top2 <- ccf_result %>%
  mutate(abs_values = abs(CCF)) %>%
  arrange(desc(abs_values)) %>%
  slice(1:2)
# exclude top lags greater than one week as they could reflect seasonal traffic variations
if (top2$abs_values[1] - top2$abs_values[2] > 0.05) {
  top <- top2[1, ]
} else {
  if (abs(top2$LAG[2]) > 7) {
    top <- top2[1, ]
  } else {
    top <- top2[2, ]
  }
}
best_lag <- top$LAG
}

# function maxLag to find the max lag between two time series when the data2 is a step function
maxLag<-function(data1, data2){
ccf_result<-as.data.frame(ccf2(diff(data1), diff(data2), max.lag = 10, plot = FALSE, type = c("c
orrelation")))
# Extract the CCF values and lags
ccf_values <- ccf_result$CCF
lags <- ccf_result$LAG
#find the maximal cross-correlation, in abs. value
top<- ccf_result %>%
  mutate(abs_values = abs(CCF)) %>%
  arrange(desc(abs_values)) %>%
  slice(1:1)
best_lag <- top$LAG

}
```

```
# function padSeries to pad lagged data series before calculating CC2
#returns diff1 and diff2
padSeries<-function(lag, data1, data2) {
m<-abs(lag)
#dy is the padded time-series diff(data1), where data1 is the data that we want to predict
#dx is the time-series that can explain it
  if(lag>0) {
        dy=c(diff(data1), rep(NA,m))
        dx=c( rep(NA,m),diff(data2))
    } else {
        dy=c(rep(NA,m),  diff(data1))
        dx=c(diff(data2),rep(NA,m))
      }
diff1<-dy
diff2<-dx
result<-cbind(diff1,diff2)
}
```

Cross correlation between traffic in 2020 and prior traffic

```
#convert to date format
date<-as.Date(numericalCovid$date)

#create empty lists for storing the results
priorLags<-list()
priorCC2F<-list()
priorSL<-list()
# list are useful for retrieving the results because if a FIPS.State.Code does not exit in our d
atabase, then the data store for it is NULL.

for (codeFIPS in allStateCodes) {
 state<- numericalCovid %>% dplyr::filter(FIPS.State.Code == codeFIPS)

#select data series for cross correlation
data1<-state$stateTraffic
data2<-state$priorTraffic

#make time series
library(astsa)
data1_ts<-ts(data1,date)
data2_ts<-ts(data2,date)
stateLag <- findLag(data1_ts,data2_ts)
# Print the result

padedDifferences<-padSeries(stateLag, data1_ts, data2_ts)

y<-padedDifferences[,1]
x<-padedDifferences[,2]

sl=lm(diff(y)~diff(x))

if (dim(summary(sl)$coef)[1]<2) {
  slope='NA'
} else {
  p_values <- summary(sl)$coefficients[, "Pr(>|t|)"]
  vals<-p_values [[2]]
  cc<-summary(sl)$coef[2]
 if(vals>0.05) {slope='NA'} else {slope=cc}
  }

priorLags[[codeFIPS]]<-stateLag
priorSL[[codeFIPS]]<-sl
priorCC2F[[codeFIPS]]<-slope
}

cc2TrafficPrior<-list(priorLags,priorSL,priorCC2F)
save(cc2TrafficPrior, file="cc2TrafficPrior.R")
```

Investigate the correlation between the traffic in 2020 and the prior traffic for a specific state

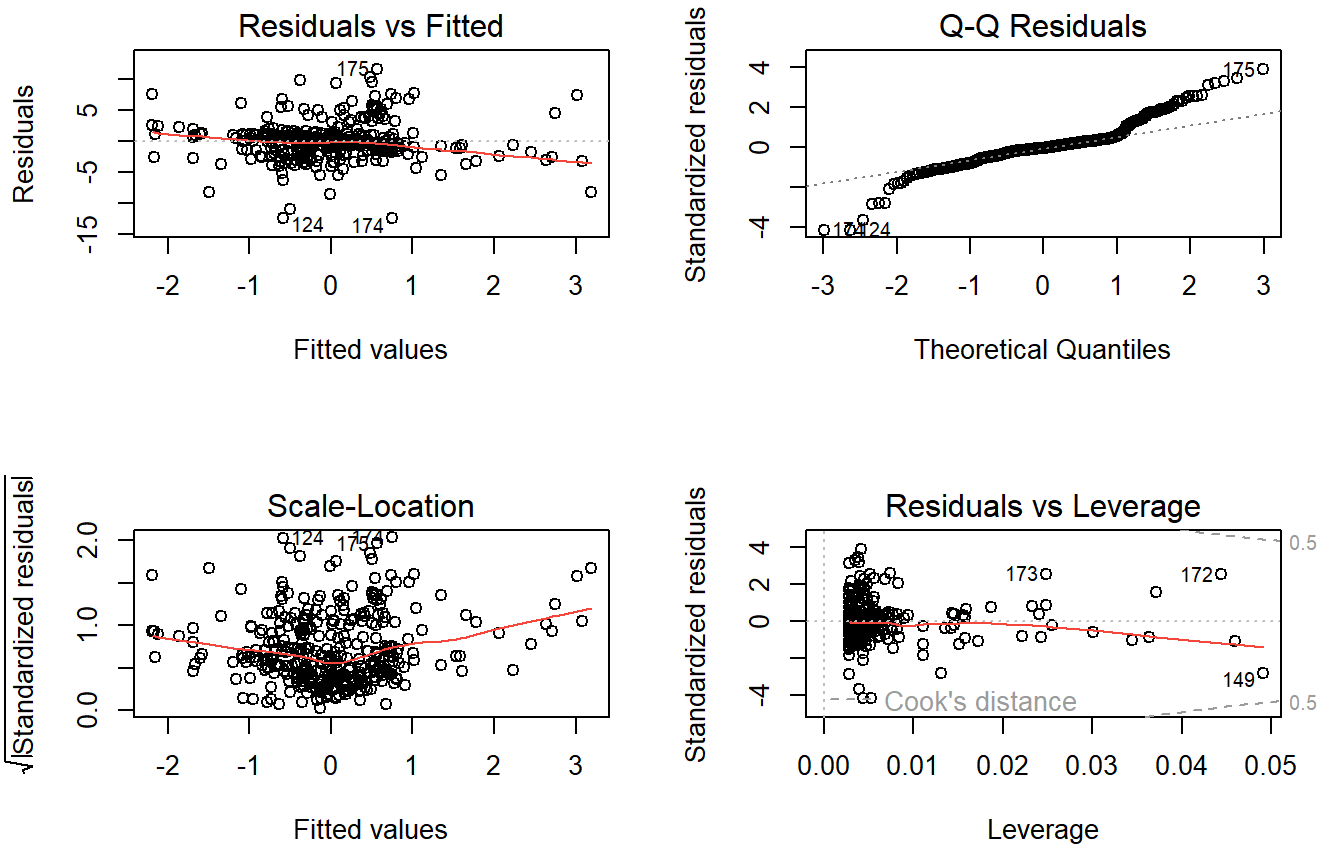
```
#choose a state
thisState=4

# extract the results
cc2model<-priorSL[[thisState]]
thisLag<-priorLags[[thisState]]
association<-priorCC2F[[thisState]]
print(thisLag)
```

```
## [1] -3
```

```
print(association)
```

```
## [1] 0.3046906
```

```
if(!is.null(cc2model)) {
  par(mfrow=c(2,2))
  plot(cc2model)
  } else {
    stop("This FIPS.State.Code is not in our database. Please try another state code.")
 }
```



Cross correlation between Traffic and Emergency Order timeline

```
#convert to date format
date<-as.Date(numericalCovid$date)

#create empty lists for storing the results
emerLags<-list()
emerCC2F<-list()
emerSL<-list()
# list are useful for retrieving the results because if a FIPS.State.Code does not exit in our d
atabase, then the data store for it is NULL.

for (codeFIPS in allStateCodes) {
 state<- numericalCovid %>% dplyr::filter(FIPS.State.Code == codeFIPS)

#select data series for cross correlation
data2<-state$emergency

#make time series
library(astsa)
data2_ts<-ts(data2,date)
stateLag <- findLag(data1_ts,data2_ts)
# Print the result

padedDifferences<-padSeries(stateLag, data1_ts, data2_ts)

y<-padedDifferences[,1]
x<-padedDifferences[,2]

sl=lm(diff(y)~diff(x))

if (dim(summary(sl)$coef)[1]<2) {
  slope='NA'
} else {
  p_values <- summary(sl)$coefficients[, "Pr(>|t|)"]
  vals<-p_values [[2]]
  cc<-summary(sl)$coef[2]
 if(vals>0.05) {slope='NA'} else {slope=cc}
  }

emerLags[[codeFIPS]]<-stateLag
emerSL[[codeFIPS]]<-sl
emerCC2F[[codeFIPS]]<-slope
}

cc2Emer<-list(emerLags, emerSL,emerCC2F)
save(cc2Emer, file="cc2Emer.R")
```

Investigate the correlation between the Traffic in 2020 and the Emergency Order for a

specific state

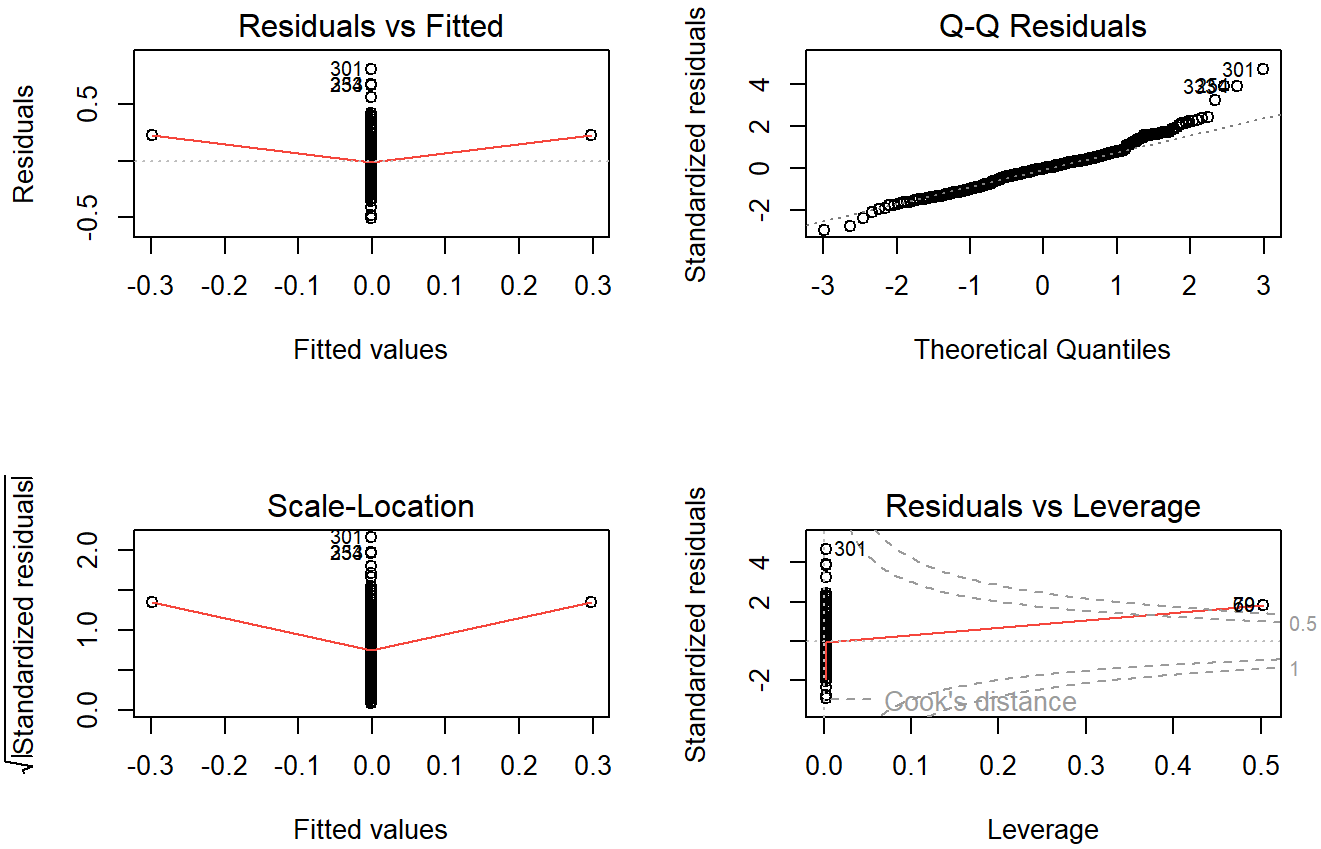
```
#choose a state
thisState=4

# extract the results
cc2model<-emerSL[[thisState]]
thisLag<-emerLags[[thisState]]
association<-emerCC2F[[thisState]]
print(thisLag)
```

```
## [1] -3
```

```
print(association)
```

```
## [1] -0.02609872
```

```
if(is.null(association)) {
   print("This FIPS.State.Code is not in our database. Please try another state code.")
}else{
    if(association=="NA"){
    print("It was not found a significant association between the changes in state traffic and t
he COVID19 Emergency Order")
    }else{
      par(mfrow=c(2,2))
      plot(cc2model)
    }
}
```



Crosscorrelation of Traffic in 2020 with the LockDown Timeline

```
#convert to date format
date<-as.Date(numericalCovid$date)

#create empty lists for storing the results
lockLags<-list()
lockCC2F<-list()
lockSL<-list()
# list are useful for retrieving the results because if a FIPS.State.Code does not exit in our d
atabase, then the data store for it is NULL.

for (codeFIPS in allStateCodes) {
 state<- numericalCovid %>% dplyr::filter(FIPS.State.Code == codeFIPS)

#select data series for cross correlation
data2<-state$lockDown

#make time series
library(astsa)
data2_ts<-ts(data2,date)

stateLag <- findLag(data1_ts,data2_ts)
# Print the result

padedDifferences<-padSeries(stateLag, data1_ts, data2_ts)

y<-padedDifferences[,1]
x<-padedDifferences[,2]

sl=lm(diff(y)~diff(x))

if (dim(summary(sl)$coef)[1]<2) {
  slope='NA'
} else {
  p_values <- summary(sl)$coefficients[, "Pr(>|t|)"]
  vals<-p_values [[2]]
  cc<-summary(sl)$coef[2]
 if(vals>0.05) {slope='NA'} else {slope=cc}
  }

lockLags[[codeFIPS]]<-stateLag
lockSL[[codeFIPS]]<-sl
lockCC2F[[codeFIPS]]<-slope
}

cc2Lock<-list(lockLags,lockSL,lockCC2F)
save(cc2Lock, file="cc2Lock.R")
```

Correlation between the Traffic in 2020 and the LockDown order for a specific state

```
#choose a state
thisState=4

# extract the results
cc2model<-lockSL[[thisState]]
thisLag<-lockLags[[thisState]]
association<-lockCC2F[[thisState]]
print(thisLag)
```

```
## [1] -1
```

```
print(association)
```

```
## [1] "NA"
```

```
if(is.null(association)) {
   print("This FIPS.State.Code is not in our database. Please try another state code.")
}else{
    if(association=="NA"){
    print("There is no significant association between the changes in state traffic and the COVI
D19 Stay Home Order")
    }else{
      par(mfrow=c(2,2))
      plot(cc2model)
    }
}
```

```
## [1] "There is no significant association between the changes in state traffic and the COVID19
Stay Home Order"
```

Crosscorrelation Traffic and travelOutState Order

```
#convert to date format
date<-as.Date(numericalCovid$date)

#create empty lists for storing the results
trvLags<-list()
trvCC2F<-list()
trvSL<-list()
# list are useful for retrieving the results because if a FIPS.State.Code does not exit in our d
atabase, then the data store for it is NULL.

for (codeFIPS in allStateCodes) {
 state<- numericalCovid %>% dplyr::filter(FIPS.State.Code == codeFIPS)

#select data series for cross correlation
data2<-state$travelBan

#make time series
library(astsa)
data2_ts<-ts(data2,date)
stateLag <- maxLag(data1_ts,data2_ts)
# Print the result

padedDifferences<-padSeries(stateLag, data1_ts, data2_ts)

y<-padedDifferences[,1]
x<-padedDifferences[,2]

sl=lm(diff(y)~diff(x))

if (dim(summary(sl)$coef)[1]<2) {
  slope='NA'
} else {
  p_values <- summary(sl)$coefficients[, "Pr(>|t|)"]
  vals<-p_values [[2]]
  cc<-summary(sl)$coef[2]
 if(vals>0.05) {slope='NA'} else {slope=cc}
  }

trvLags[[codeFIPS]]<-stateLag
trvSL[[codeFIPS]]<-sl
trvCC2F[[codeFIPS]]<-slope
}

cc2Trv<-list(trvLags,trvSL,trvCC2F)
save(cc2Trv, file="cc2Trv.R")
```

Investigate the correlation between the Traffic in 2020 and the travelOutState order for a

specific state

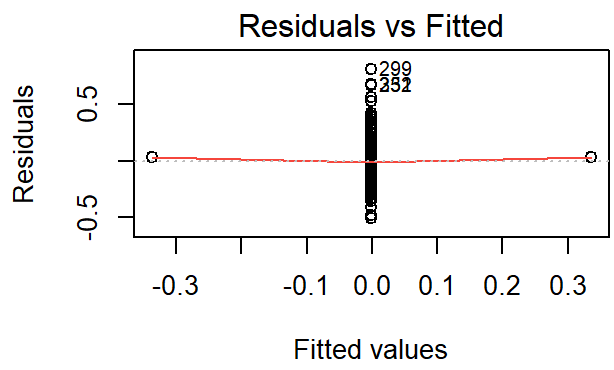
```
#choose a state
thisState=4

# extract the results
cc2model<-trvSL[[thisState]]
thisLag<-trvLags[[thisState]]
association<-trvCC2F[[thisState]]
print(thisLag)
```

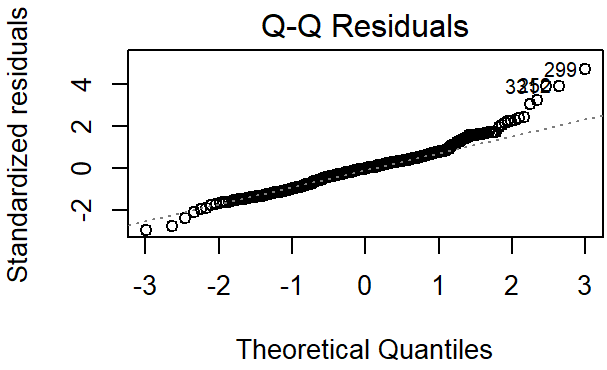
```
## [1] -1
```

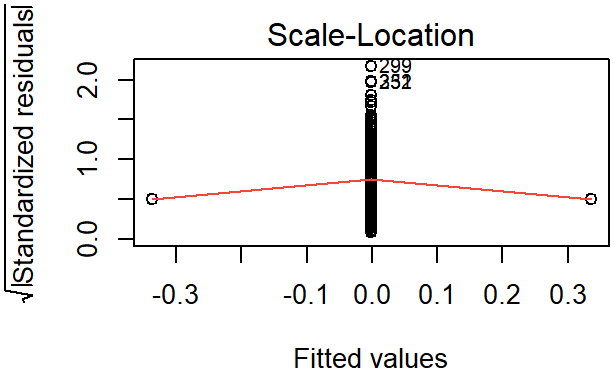
```
print(association)
```

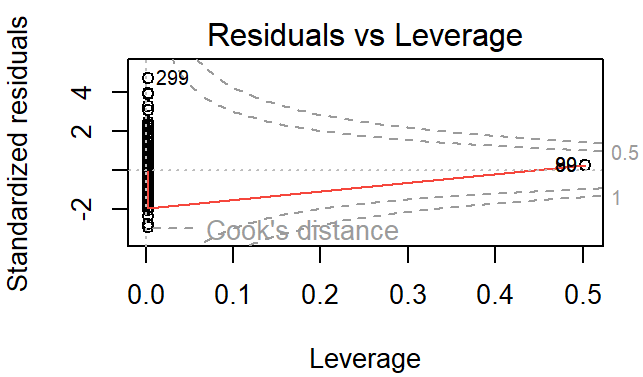
```
## [1] 0.0568367
```

```
if(is.null(association)) {
   print("This FIPS.State.Code is not in our database. Please try another state code.")
}else{
    if(association=="NA"){
    print("There is no significant association between the changes in state traffic and the COVI
D19 Ban on Travel Out-of-State ")
    }else{
      par(mfrow=c(2,2))
      plot(cc2model)
    }
}
```









Crosscorrelation with closeRetailState Order

```
#convert to date format
date<-as.Date(numericalCovid$date)

#create empty lists for storing the results
clsLags<-list()
clsCC2F<-list()
clsSL<-list()
# list are useful for retrieving the results because if a FIPS.State.Code does not exit in our d
atabase, then the data store for it is NULL.

for (codeFIPS in allStateCodes) {
 state<- numericalCovid %>% dplyr::filter(FIPS.State.Code == codeFIPS)

#select data series for cross correlation
data2<-state$closedStores

#make time series
library(astsa)
data2_ts<-ts(data2,date)
stateLag <- maxLag(data1_ts,data2_ts)
# Print the result

padedDifferences<-padSeries(stateLag, data1_ts, data2_ts)

y<-padedDifferences[,1]
x<-padedDifferences[,2]

sl=lm(diff(y)~diff(x))

if (dim(summary(sl)$coef)[1]<2) {
  slope='NA'
} else {
  p_values <- summary(sl)$coefficients[, "Pr(>|t|)"]
  vals<-p_values [[2]]
  cc<-summary(sl)$coef[2]
 if(vals>0.05) {slope='NA'} else {slope=cc}
  }

clsLags[[codeFIPS]]<-stateLag
clsSL[[codeFIPS]]<-sl
clsCC2F[[codeFIPS]]<-slope

}

cc2Cls<-list(clsLags,clsSL,clsCC2F)

save(cc2Cls, file="cc2Cls.R")
```

Investigate the correlation between the traffic in 2020 and the closeRetailState order in a specific state

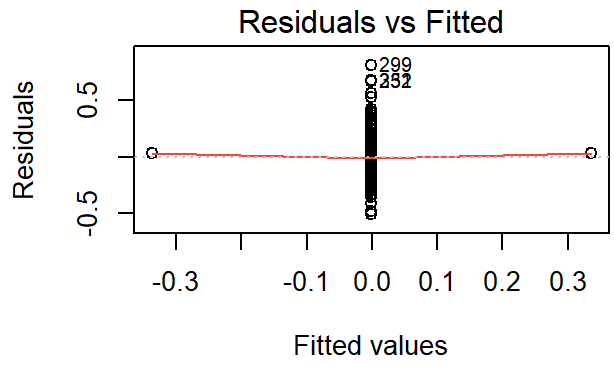
```
#choose a state
thisState=4

# extract the results
cc2model<-clsSL[[thisState]]
thisLag<-clsLags[[thisState]]
association<-clsCC2F[[thisState]]
print(thisLag)
```

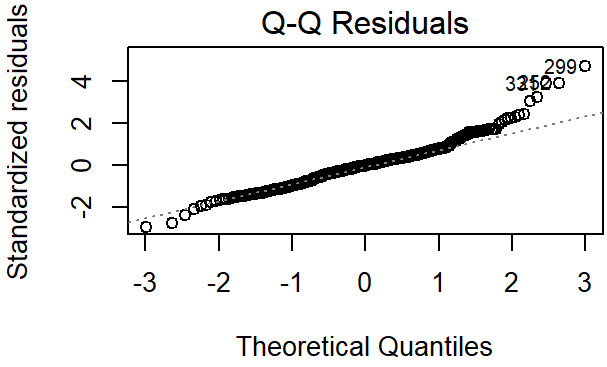
```
## [1] -1
```

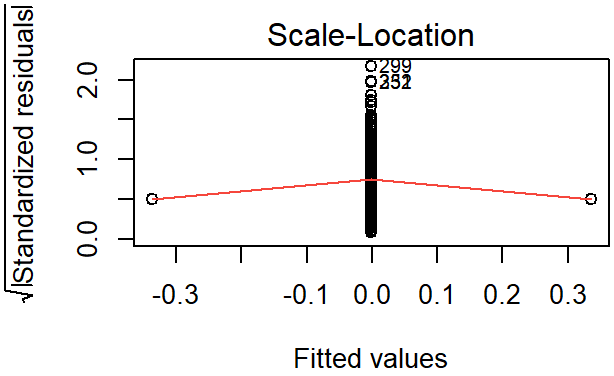
```
print(association)
```

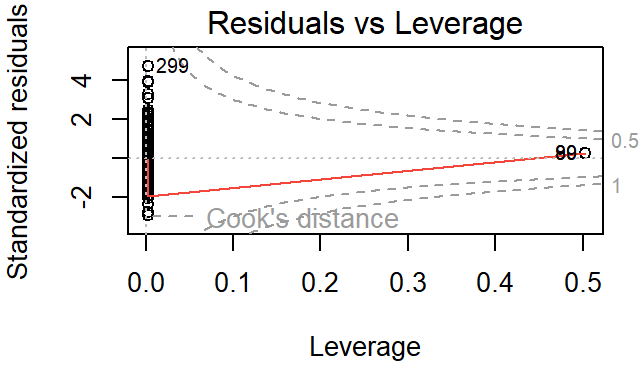
```
## [1] 0.03876723
```

```
if(is.null(association)) {
   print("This FIPS.State.Code is not in our database. Please try another state code.")
}else{
    if(association=="NA"){
    print("There is no significant association between the changes in state traffic and the COVI
D19 Ban on Travel Out-of-State ")
    }else{
      par(mfrow=c(2,2))
      plot(cc2model)
    }
}
```


Residuals vs Fitted
Residuals
Fitted values
299
252


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
299


Scale-Location
√|Standardized residuals|
Fitted values
299
252


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance
299
0.5
1