# Log-linear regression analysis to investigate the associations between transmission controls measures and the number of reported cases in the first seven days of the outbreaks in cities

Chieh-Hsi Wu

## Contents



Here we investigate the associations between transmission control measures and the number of reported cases in the first week of the outbreaks in cities.

```
library(readxl)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
```

```
##     as.Date, as.Date.numeric
```

```
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
library(R330)
```

```
## Loading required package: s20x
```

```
## Loading required package: leaps
```

```
## Loading required package: rgl
```

```
# function to obtain regression coefficients
bs = function(formula, data, indices) {
  # allows boot to select sample
  d <- data[indices,]
  fit <- lm(formula, data=d)
  return(coef(fit))
}


getSgnChgIndex = function(ci = NULL, a = NULL){
  colnames(ci) = a
  rownames(ci) = c("CI level",
                   "lower index", "upper index",
                   "95% CI lower bound", "95% CI upper bound")

  ci = t(ci[4:5,])
  cprCIBndSgn = apply(ci,1,
        function(z){
          sgn = z>0;
          return(sgn[1]==sgn[2])
        })
  return(max(which(cprCIBndSgn)))
}
```

```
covid2019FilePath = "/Users/chwu/Documents/research/nCov-2019_TCM/nCoV-data.xlsx"
covid2019.df = read_excel(path = covid2019FilePath, sheet = "3resp-7days")
covid2019Dist.df = read_excel(path = covid2019FilePath, sheet = "dist-296")

covid2019v2.df = read_excel(path = "/Users/chwu/Documents/research/nCov-2019_TCM/nCoV-data_0323.xlsx",
                            sheet = "3resp-7days")
```

## Processing the data

Some of the cities have a inflow from Wuhan recorded as 0, which causes calculations to run into an error when we use it as an offset variable. To resolve this issue, 0 values are changed to $10^{-6}$, which is equivalent to only one person arriving to a city from Wuhan.

The arrival time is processed so that 31 December 2019 is coded as day 0.

```
covid2019.df$new.arr.time  = covid2019.df$arr.time - 1
```

The timing of suspending intra-city public transport is processed so that 31 December 2019 is coded as day 0.

```
bus.resp.tab = table(covid2019.df$Bus.resp)
bus.resp.tab
```

```
##
##   0   1
## 207  89
```

```
bus.date.tab1 = table(covid2019.df$Bus.date[which(covid2019.df$Bus.resp==1)])
bus.date.tab1
```

```
##
## 24 25 26 27 28 29 30 31 32 33
##  3 10  5 14 20 20 11  2  3  1
```

```
covid2019.df$new.Bus.date = covid2019.df$Bus.date - 1
new.bus.date.tab1 = table(covid2019.df$new.Bus.date[which(covid2019.df$Bus.resp==1)])
new.bus.date.tab1
```

```
##
## 23 24 25 26 27 28 29 30 31 32
##  3 10  5 14 20 20 11  2  3  1
```

```
covid2019.df$new.Bus.date[which(covid2019.df$Bus.resp==0)] = 0
new.bus.date.tab = table(covid2019.df$new.Bus.date)
new.bus.date.tab
```

```
##
##   0  23  24  25  26  27  28  29  30  31  32
## 207   3  10   5  14  20  20  11   2   3   1
```

```
## Sanity check
## Should return 0(s).
# bus.date.tab1 - new.bus.date.tab1
# (as.numeric(names(bus.date.tab1)) - 1) - as.numeric(names(new.bus.date.tab1))
# bus.date.tab1 - new.bus.date.tab[-1]
# (as.numeric(names(bus.date.tab1)) - 1) - as.numeric(names(new.bus.date.tab[-1]))
# bus.resp.tab["0"] - new.bus.date.tab["0"]
## Sanity check complete
```

The timing of suspending inter-city passenger traffic is processed so that 31 December 2019 is coded as day 0.

```
##
##   0   1
## 125 171
```

```
##
## 24 25 26 27 28 29 30 31 32
##  1 10 14 48 66 21  8  1  2
```

```
##
## 23 24 25 26 27 28 29 30 31
##  1 10 14 48 66 21  8  1  2
```

```
##
##   0  23  24  25  26  27  28  29  30  31
## 125   1  10  14  48  66  21   8   1   2
```

The timing of closure of entertainment venues and banning public gathering is processed so that 31 December 2019 is coded as day 0.

```
##
##   0   1
## 127 169
```

```
##
## 24 25 26 27 28 29 30 32 35
##  5 64 42 27  5  7 16  2  1
```

```
##
## 23 24 25 26 27 28 29 31 34
##  5 64 42 27  5  7 16  2  1
```

```
##
##   0  23  24  25  26  27  28  29  31  34
## 127   5  64  42  27   5   7  16   2   1
```

## Fitting a log-linear regression model

First we create a new response variable which is incidence per capita divided by the inflow Wuhan.

```
covid2019.df$std.sevendays.cucase =
  covid2019.df$sevendays.cucase/
  (covid2019.df$Pop_million_2018*covid2019.df$new.totalflow_million)
```

After some exploration we found that there's non-linear relationship between dependent variable and the timing of closure of entertainment venues and banning public gathering. Therefore we have used a square term of that timeing variable to take care of that. The log-linear model summarised below.

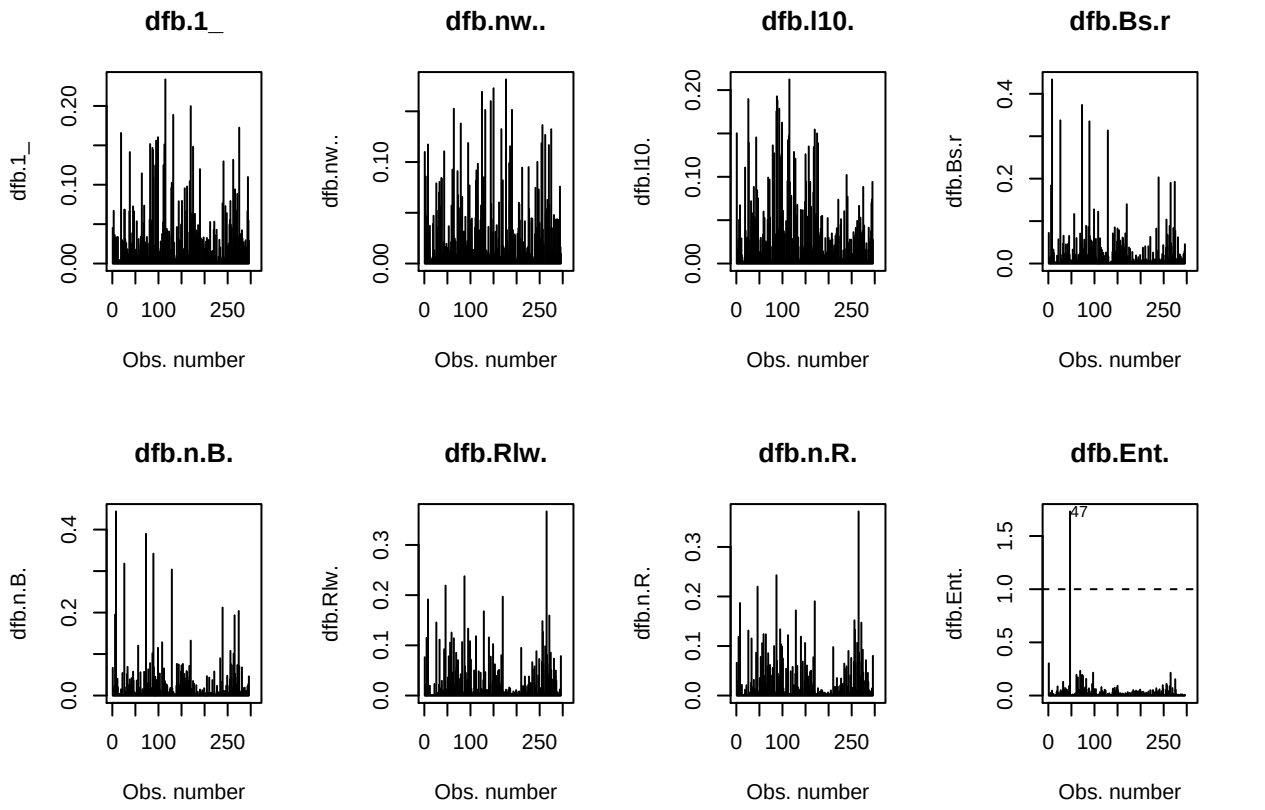
```
yfc.resp.date.lm = lm(log(std.sevendays.cucase) ~ new.arr.time + log10.Dis.WH +
                        Bus.resp + new.Bus.date +
                        Railway.resp + new.Railway.date +
                        Enter.resp + new.Enter.date + I(new.Enter.date^2),
                      data = covid2019.df)
summary(yfc.resp.date.lm)
```

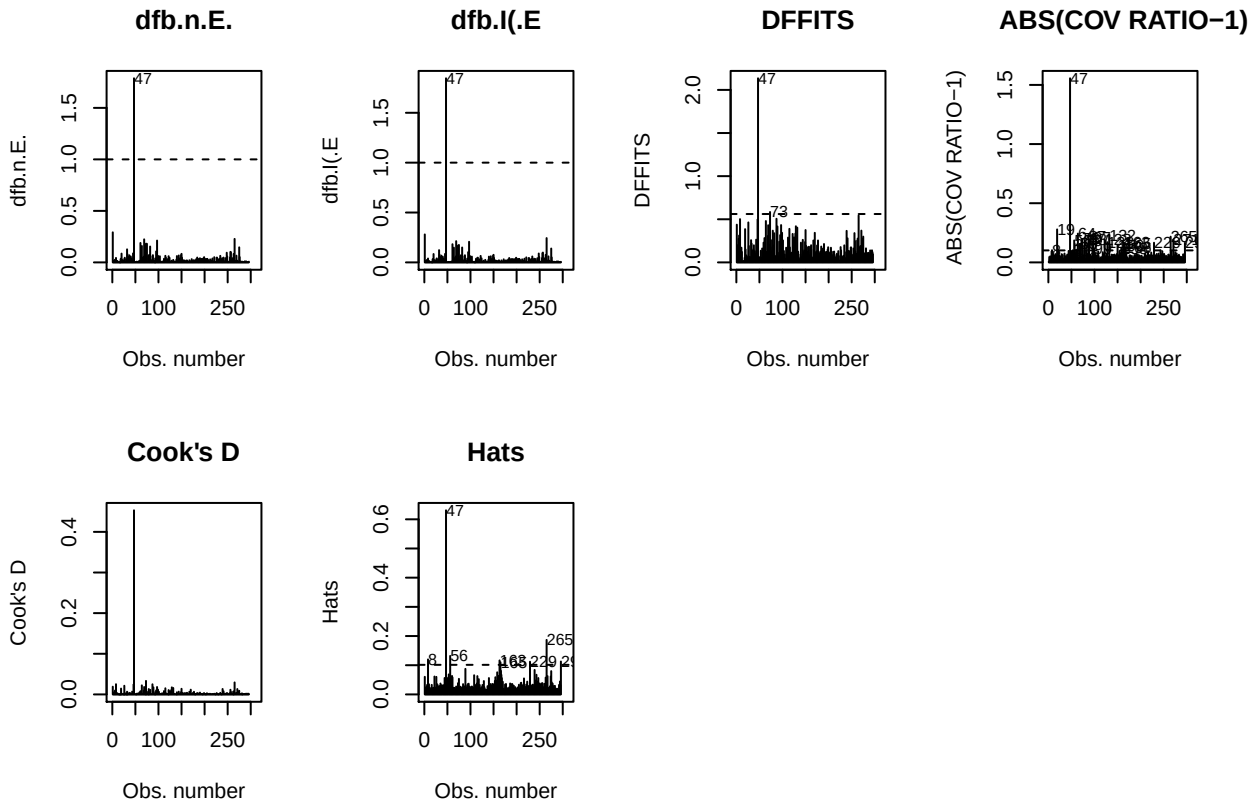
```
##
## Call:
## lm(formula = log(std.sevendays.cucase) ~ new.arr.time + log10.Dis.WH +
##     Bus.resp + new.Bus.date + Railway.resp + new.Railway.date +
##     Enter.resp + new.Enter.date + I(new.Enter.date^2), data = covid2019.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9423 -1.9987 -0.3722  1.3091  9.4776
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -3.06426    2.57925  -1.188  0.23580
## new.arr.time          0.36826    0.08104   4.544 8.15e-06 ***
## log10.Dis.WH         -0.37351    0.71627  -0.521  0.60244
## Bus.resp            -16.27164    5.46314  -2.978  0.00315 **
## new.Bus.date          0.56858    0.19935   2.852  0.00466 **
## Railway.resp          1.81901    5.51638   0.330  0.74183
## new.Railway.date     -0.01526    0.20601  -0.074  0.94099
```

```
## Enter.resp          -89.74984   32.81630  -2.735  0.00663 **
## new.Enter.date        6.76256    2.44913   2.761  0.00613 **
## I(new.Enter.date^2)  -0.12579    0.04546  -2.767  0.00603 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.939 on 286 degrees of freedom
## Multiple R-squared:  0.1714, Adjusted R-squared:  0.1453
## F-statistic: 6.572 on 9 and 286 DF,  p-value: 1.58e-08
```

The influential plots indicates an apparent influential point.

```
par(mfrow = c(2, 4))
influenceplots(yfc.resp.date.lm)
```






For interpretabiliy, instead of having a squared we discretise the timing variable, and re-fit the model with the timing variable.

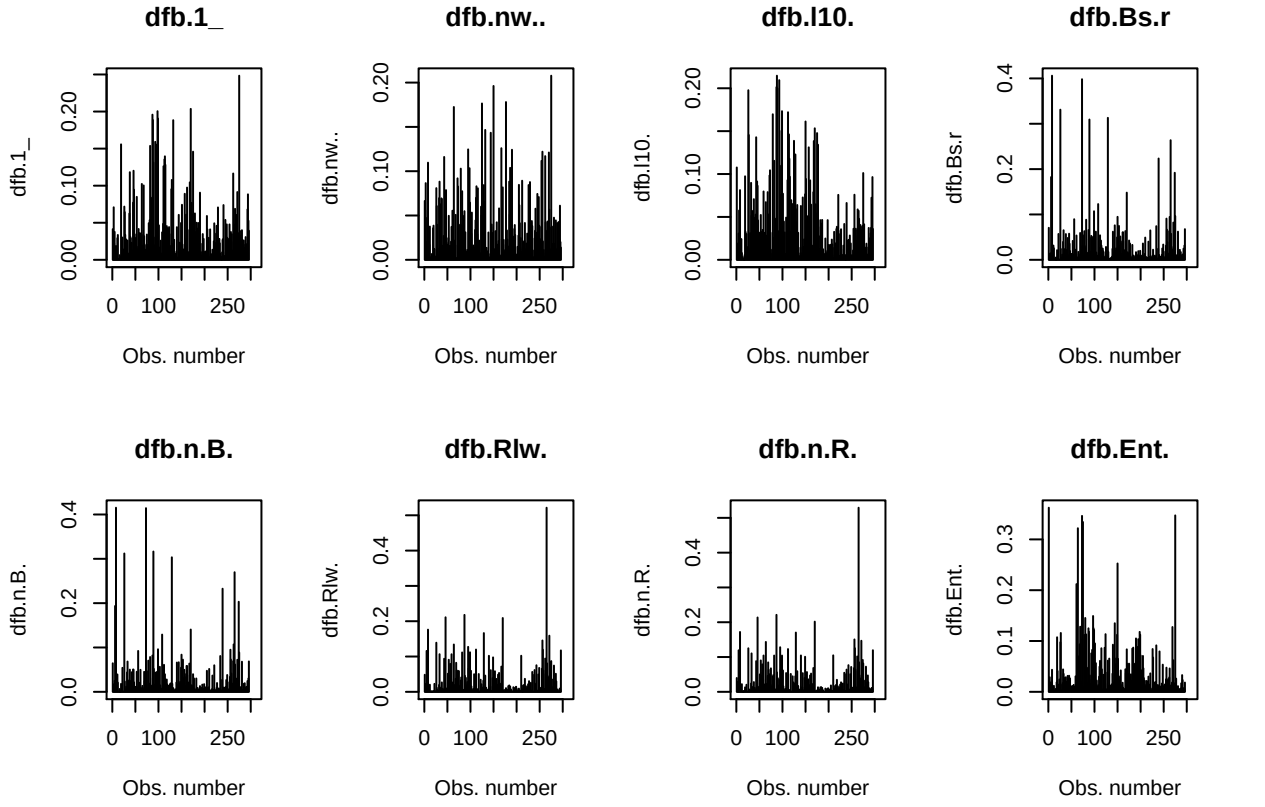
```
covid2019.df$new.Enter.date.cat = cut(covid2019.df$new.Enter.date, c(-1, c(23, 24, 25, 36) -1))
covid2019.df$new.Enter.date.cat = as.numeric(covid2019.df$new.Enter.date.cat) -1

yfc.resp.date.lm1 = lm(log(std.sevendays.cucase) ~ new.arr.time + log10.Dis.WH +
                         Bus.resp + new.Bus.date +
                         Railway.resp + new.Railway.date +
                         Enter.resp +  new.Enter.date.cat,
                       data = covid2019.df)
summary(yfc.resp.date.lm1)
```

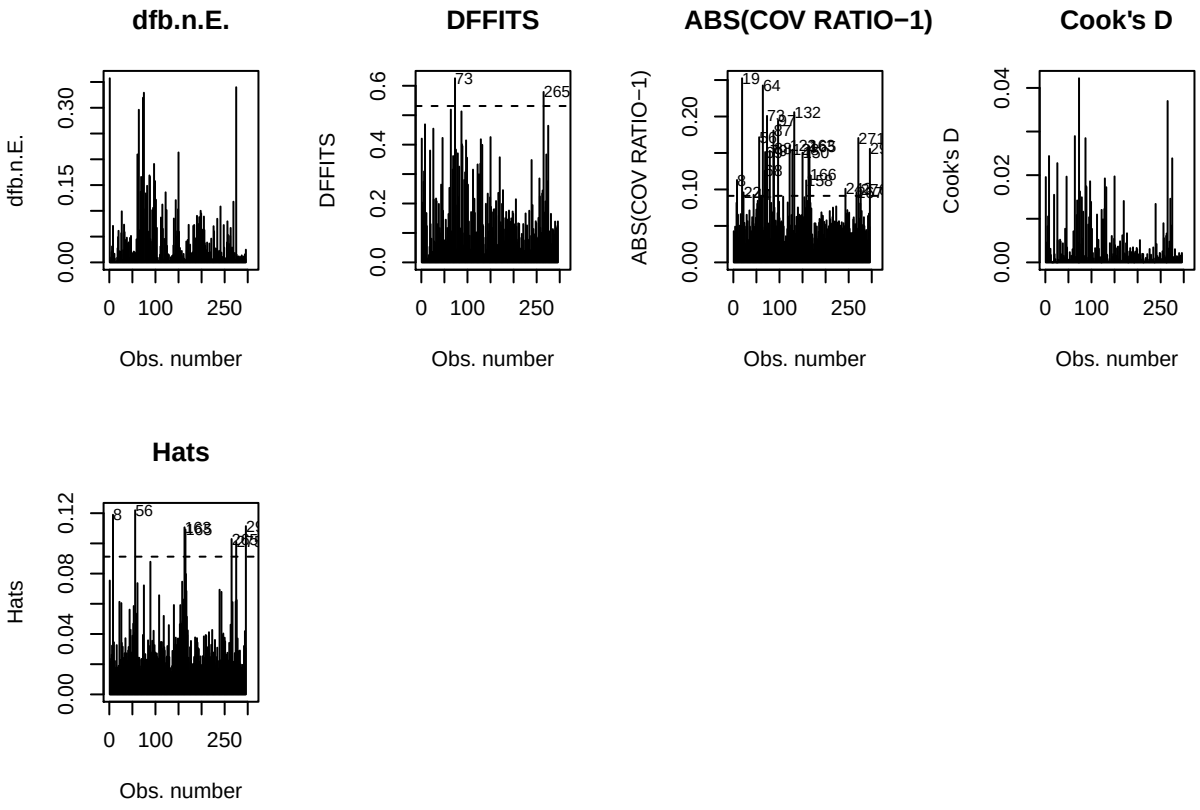
```
##
## Call:
## lm(formula = log(std.sevendays.cucase) ~ new.arr.time + log10.Dis.WH +
##     Bus.resp + new.Bus.date + Railway.resp + new.Railway.date +
##     Enter.resp + new.Enter.date.cat, data = covid2019.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9124 -1.9975 -0.3785  1.2936  9.4273
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.64226    2.45119  -0.670  0.50341
## new.arr.time         0.32736    0.08049   4.067 6.16e-05 ***
## log10.Dis.WH        -0.51421    0.68769  -0.748  0.45523
## Bus.resp           -15.40124    5.44481  -2.829  0.00500 **
```

```
## new.Bus.date          0.54029    0.19862   2.720  0.00692 **
## Railway.resp          2.64510    5.50740   0.480  0.63139
## new.Railway.date     -0.04779    0.20575  -0.232  0.81648
## Enter.resp           -2.93703    1.18615  -2.476  0.01386 *
## new.Enter.date.cat    1.29960    0.43526   2.986  0.00307 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.928 on 287 degrees of freedom
## Multiple R-squared:  0.1748, Adjusted R-squared:  0.1518
## F-statistic:   7.6 on 8 and 287 DF,  p-value: 3.208e-09
```

After the discretising the timing variable, the observation flagged as an influential is no longer problematic. The Cook's distances do not indicate presence of outliers. Here are a number of observations indicated as having large hat values. However, they are not too far away from they cutoff, and their hat values are quite similar to each other. So it is hard to justigy removing only a subset of those points, but there are quite a few of them, so we leave them in the model.

```
par(mfrow = c(2,4))
influenceplots(yfc.resp.date.lm1)
```









The studentized Breusch-Pagan test provides evidence for heterscedasticity in the residuals.

```
lmtest::bptest(yfc.resp.date.lm1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  yfc.resp.date.lm1
## BP = 30.671, df = 8, p-value = 0.0001607
```

After some exploration, we found that heteroscedasticity occurs when we include either the log10 of distance to Wuhan, and the binary and timing varibles for closure of entertainment venues and banning public gathrings. Coincidentally, the models above show that we have no evidence for the the associations between those three variables and the dependent variable.

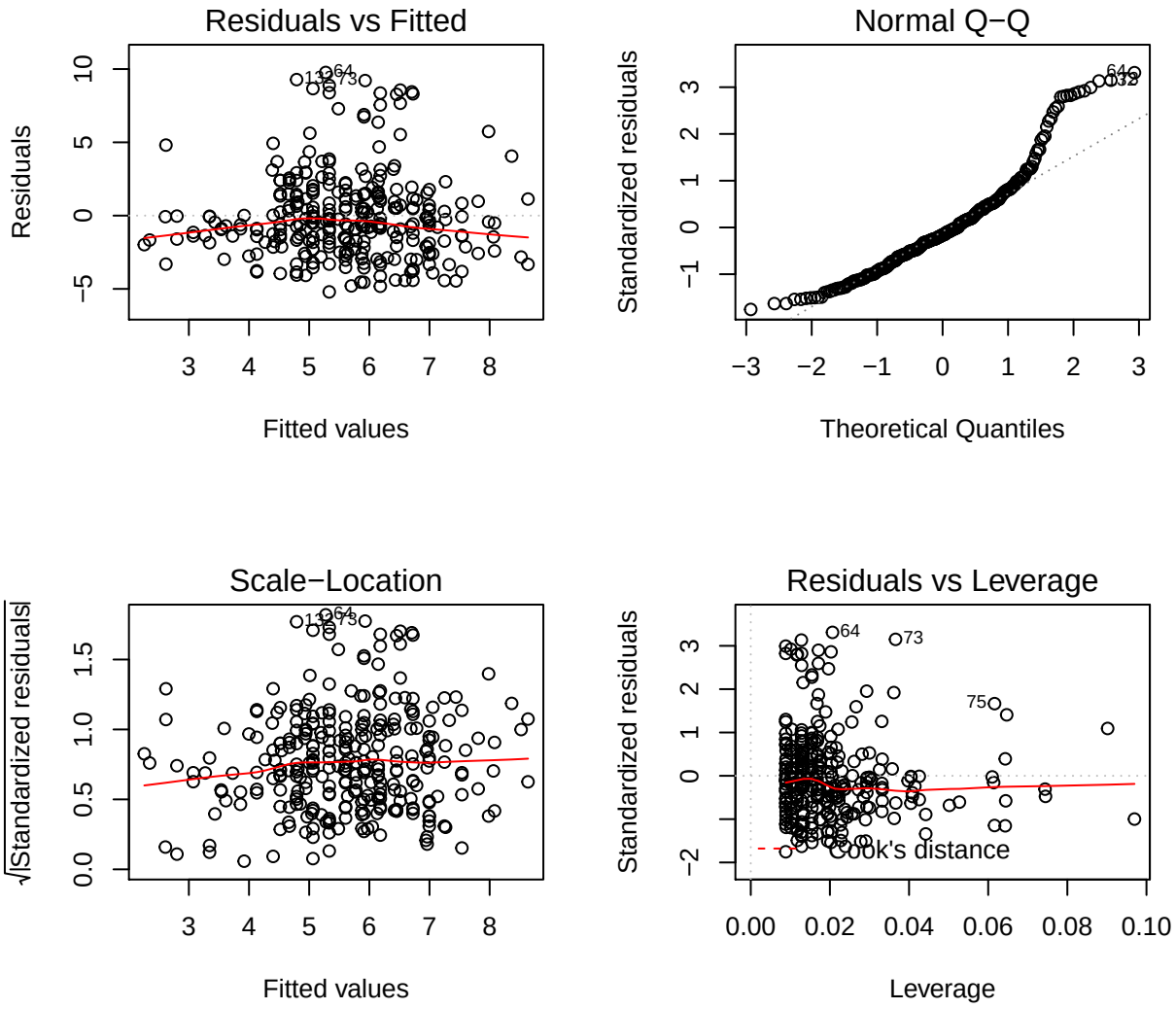
The conclusions regarding the rest of the variables do not change.

```
##
## Call:
## lm(formula = log(std.sevendays.cucase) ~ new.arr.time + Bus.resp +
##     new.Bus.date + Enter.resp + new.Enter.date.cat, data = covid2019.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2050 -1.8544 -0.4602  1.3450  9.7662
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.17630    1.93005  -0.609 0.542692
## new.arr.time         0.27125    0.07816   3.471 0.000598 ***
```

```
## Bus.resp            -12.70925    4.62522  -2.748 0.006375 **
## new.Bus.date          0.46292    0.17051   2.715 0.007027 **
## Enter.resp           -3.41094    1.18132  -2.887 0.004177 **
## new.Enter.date.cat    1.51027    0.43274   3.490 0.000558 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.981 on 290 degrees of freedom
## Multiple R-squared:  0.1357, Adjusted R-squared:  0.1208
## F-statistic: 9.105 on 5 and 290 DF,  p-value: 4.665e-08
```

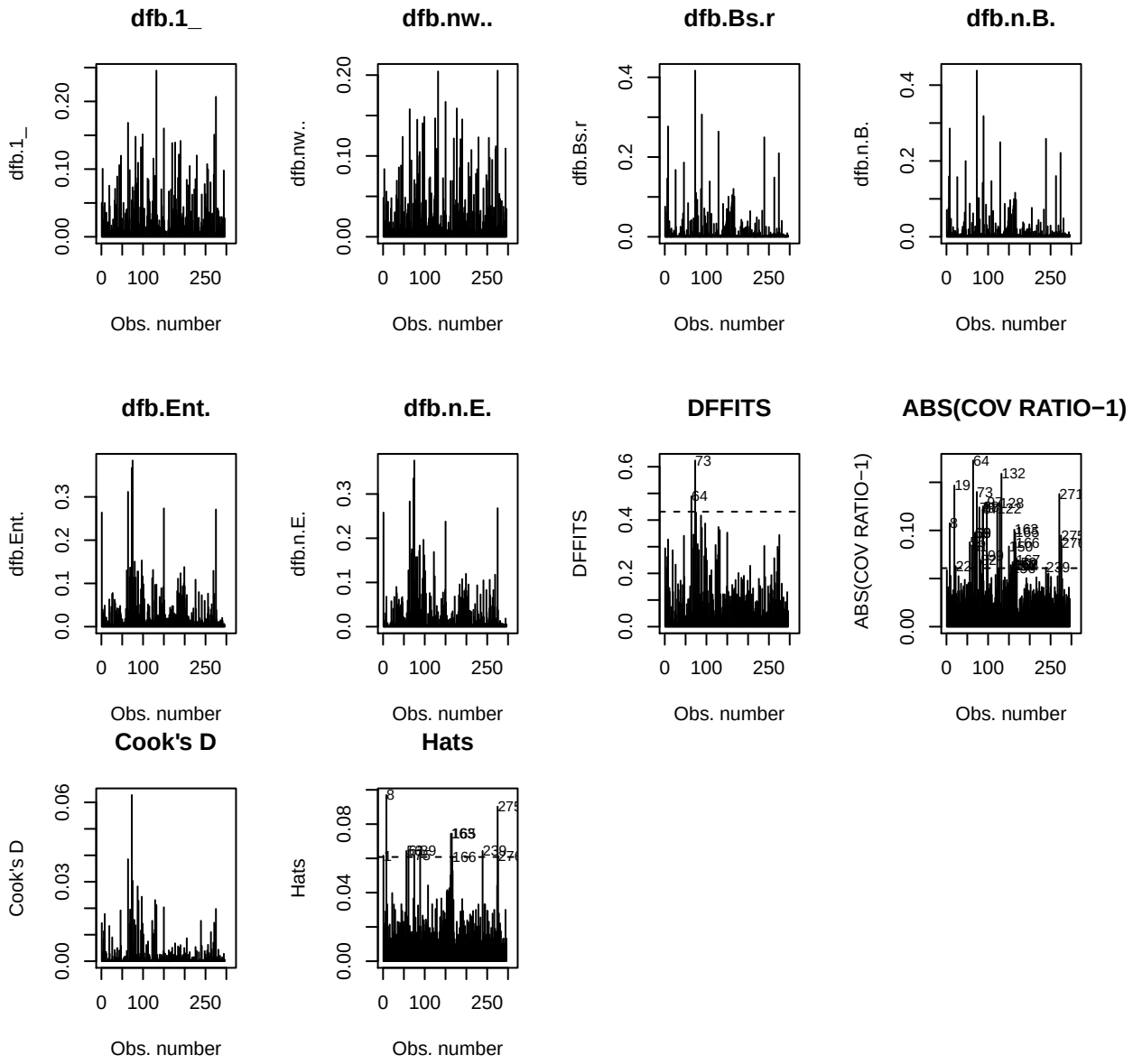
The plot of studentised residuals does not indicate evident non-linearity in the residuals

```
par(mfrow = c(2,2))
plot(yfc.resp.date.lm2)
```




The influence plots do provide strong indication any points that are particularly influential and need to be removed.

```
par(mfrow = c(2,4))
influenceplots(yfc.resp.date.lm2)
```




The studentized Breusch-Pagan test dose not provide evidence for heteroscedasticity in the residuals.

```
lmtest::bptest(yfc.resp.date.lm2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  yfc.resp.date.lm2
## BP = 8.1589, df = 5, p-value = 0.1477
```

```
yfc.resp.date.lm2.est = coef(summary(yfc.resp.date.lm2))
yfc.resp.date.lm2.est.tab = cbind(yfc.resp.date.lm2.est[,"Estimate"],
yfc.resp.date.lm2.est[,"Estimate"]-1.96*yfc.resp.date.lm2.est[,"Std. Error"],
yfc.resp.date.lm2.est[,"Estimate"]+1.96*yfc.resp.date.lm2.est[,"Std. Error"])
colnames(yfc.resp.date.lm2.est.tab) = c("Coefficient", "95% CI upper bound", "95% CI lower bound")
round(yfc.resp.date.lm2.est.tab, 2)
```

```
##                    Coefficient 95% CI upper bound 95% CI lower bound
```

```
## (Intercept)              -1.18             -4.96              2.61
## new.arr.time              0.27              0.12              0.42
## Bus.resp                -12.71            -21.77             -3.64
## new.Bus.date              0.46              0.13              0.80
## Enter.resp               -3.41             -5.73             -1.10
## new.Enter.date.cat        1.51              0.66              2.36
```

# Diagnostics

## Check for spatial correlation in the residuals

Here we check whether cities that are closer together are going have more similar residuals

```
covid2019DistMat = as.matrix(covid2019Dist.df[,-1], nrow = 296, ncol = 296)
rownames(covid2019DistMat) = covid2019Dist.df$code
colnames(covid2019DistMat) = covid2019Dist.df$code
```

```
all(covid2019.df$Code == rownames(covid2019DistMat))
```

```
## [1] TRUE
```

```
all(covid2019.df$Code == colnames(covid2019DistMat))
```

```
## [1] TRUE
```

```
yfc.resp.date.lm2.res = residuals(yfc.resp.date.lm2)
res.dist = as.matrix(dist(yfc.resp.date.lm2.res))
cityDist = covid2019DistMat
# 262 is NA by mistake in the input file
diag(cityDist)[262] = 0
```

```
## Only extract the values of the upper triangle of the matrix as
## The lower triangle repeats the upper triange values.
res.dist.unique = res.dist[upper.tri(res.dist)]
city.dist.unique = cityDist[upper.tri(cityDist)]
length(res.dist.unique) == length(city.dist.unique)
```

```
## [1] TRUE
```

```
## Sanity checks
## The codes below should all return TRUE(s).
# res.dist[1,2] == res.dist.unique[1]
# res.dist[1:2,3] == res.dist.unique[1+1:2]
# res.dist[1:4,5] == res.dist.unique[6+1:4]
# res.dist[1:5,6] == res.dist.unique[10+1:5]

# cityDist[1,2] == city.dist.unique[1]
# cityDist[1:2,3] == city.dist.unique[1+1:2]
# cityDist[1:4,5] == city.dist.unique[6+1:4]
#cityDist[1:5,6] == city.dist.unique[10+1:5]
## Sanity checks complete
```

There is no evident correlation between the pairiwse residual differences and pairwise city differences.

```
cor((res.dist.unique), city.dist.unique)
```

```
## [1] -0.09319644
```

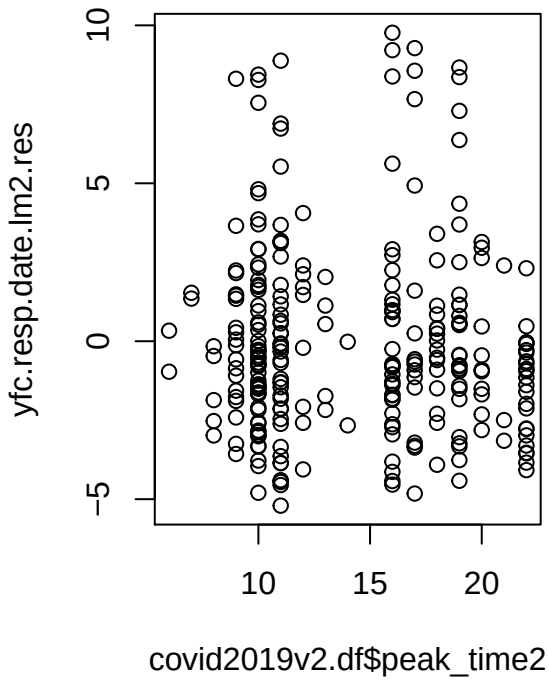
Here we check whether the peak times of inflow from Wuhan correlations with the residuals. The peak inflow

is calculated for the period from 11 January 2020 to 23 January 2020. 11 January is 15 days before the Chinese New Year, while 23 January is day of Wuhan shutdown.

```
all(covid2019.df$Code == covid2019v2.df$Code)
```

```
## [1] TRUE
```

```
par(mfrow = c(1,2))
plot(x = covid2019v2.df$peak_time2,
     y = yfc.resp.date.lm2.res)
```




We do not find evident correlation between the residuals peak times of inflow from Wuhan.

```
cor(x = covid2019v2.df$peak_time2,
    y = yfc.resp.date.lm2.res, use="pairwise.complete.obs")
```

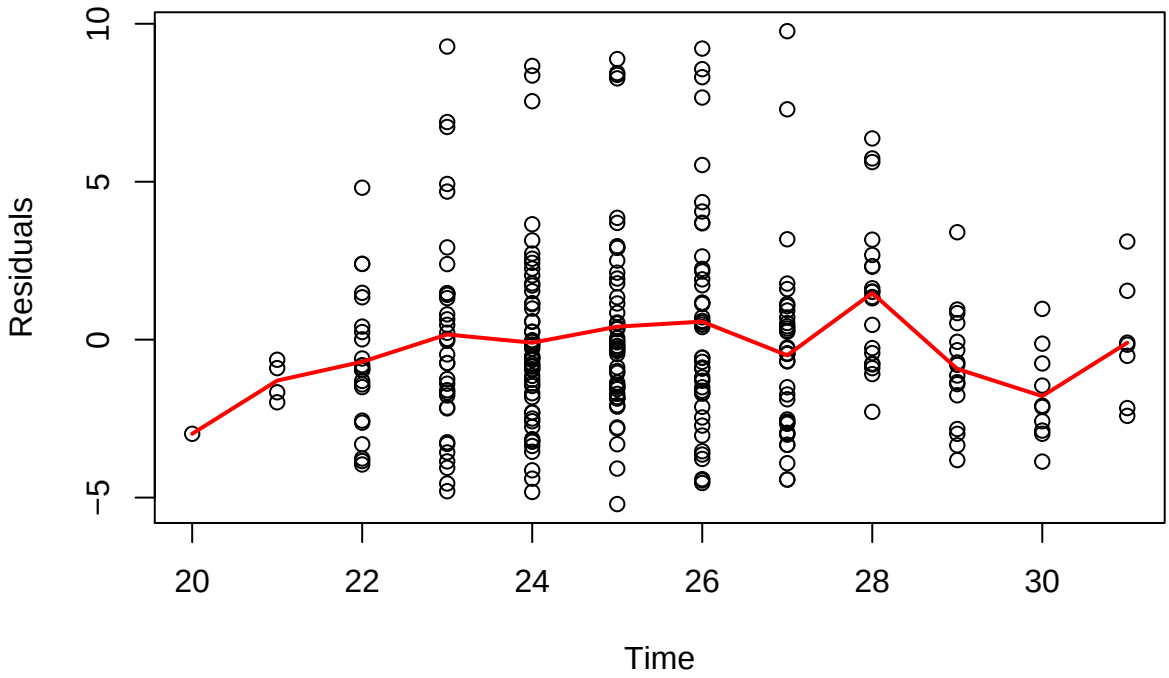
```
## [1] -0.04064901
```

### Check for temporal correlation in the residuals

Here we check whether there's some temporal autocorrelation in the data. To this end we evaluate the strength of evidence for the association between the mean residuals with arrival time on day j and the residuals with arrival time on day j+1.

```
## Get the unqiue arrival times
arr.time.level = sort(unique(covid2019.df$arr.time))

## Calculate the mean arrival time for each arrival time value
res.mean.by.time = vector(length = length(arr.time.level))
for(i in 1:length(arr.time.level)){
  # get the residual values for a given arrival time
  res.per.arrT = yfc.resp.date.lm2.res[covid2019.df$arr.time == arr.time.level[i]]
  res.mean.by.time[i] = mean(res.per.arrT)
}
names(res.mean.by.time) = arr.time.level
```

```
plot(covid2019.df$arr.time,
     yfc.resp.date.lm2.res,
     xlab = "Time", ylab = "Residuals")
lines(x = arr.time.level, y = res.mean.by.time,
      col = "red", lwd = 2)
```




```
arr.time.level.prev = arr.time.level[-length(arr.time.level)]
arr.time.level.prev
```

```
##  [1] 20 21 22 23 24 25 26 27 28 29 30
```

```
names(arr.time.level.prev) = arr.time.level[-1]
```

The earliest arrival time is 20 January. So the residuals with this arrival time will not have a covariate value.

```
prev.day = arr.time.level.prev[as.character(covid2019.df$arr.time)]
table(covid2019.df$arr.time - prev.day, useNA = "always")
```

```
##
##    1 <NA>
##  295    1
```

```
# Sanity check
# table(covid2019.df$arr.time - 1)[-1] == table(prev.day)
```

There is not evidence for an association between the mean residuals with arrival time on day j and the residuals with arrival time on day j + 1.

```
prev.mean = res.mean.by.time[as.character(prev.day)]
summary(lm(yfc.resp.date.lm2.res ~ prev.mean))
```

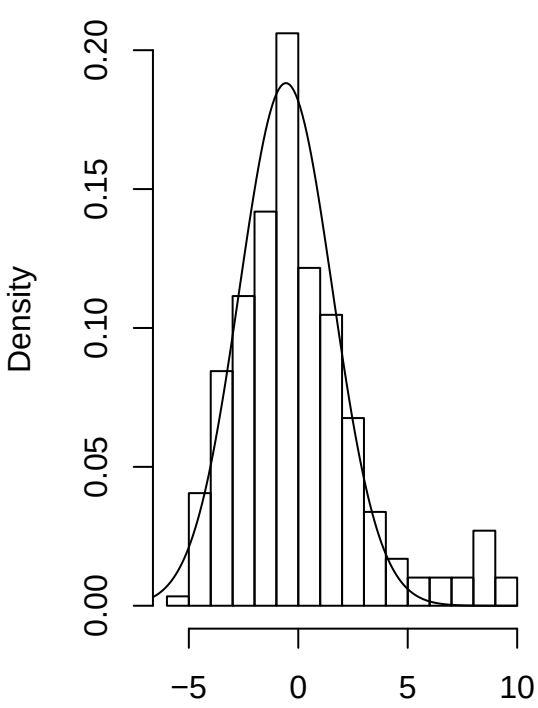
```
##
## Call:
## lm(formula = yfc.resp.date.lm2.res ~ prev.mean)
##
## Residuals:
```

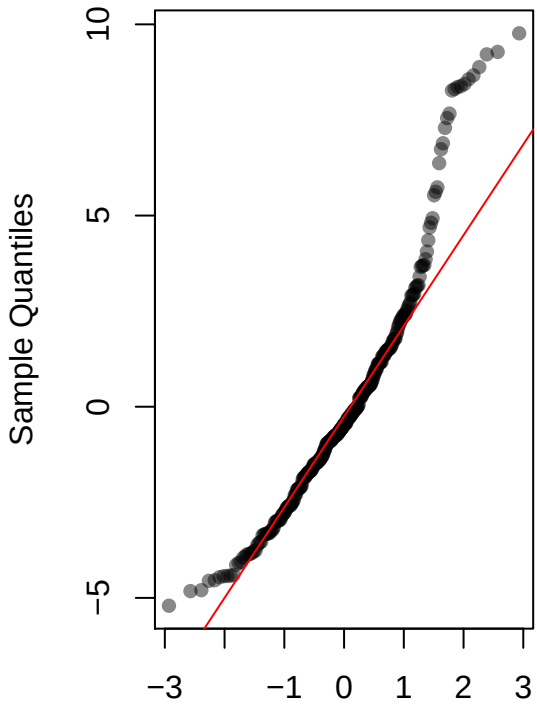
```
##     Min      1Q  Median      3Q     Max
## -5.2156 -1.8542 -0.4795  1.3810  9.7176
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01607    0.17393   0.092    0.926
## prev.mean    0.05705    0.22191   0.257    0.797
##
## Residual standard error: 2.961 on 293 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.0002255,  Adjusted R-squared:  -0.003187
## F-statistic: 0.06609 on 1 and 293 DF,  p-value: 0.7973
```

### Check the normality assumption

The disribution of the residuals is fairly symmetric albeit the longer right tail. As a result, we see moderate departure from normality in the Q-Q plots.

```
#pdf(file = "/Users/chwu/Documents/research/nCov-2019_TCM/logLinearNormality.pdf",
#    width = 9, height = 4.5)
par(mfrow = c(1,2), mar = c(5,4,1,2)+0.2)

hist(yfc.resp.date.lm2.res,
     xlab = "Residuals", main = "",
     nclass = 20, prob = T)
yfc.resp.date.lm2.res2 =
  yfc.resp.date.lm2.res[which(yfc.resp.date.lm2.res < abs(min(yfc.resp.date.lm2.res)))]
normFitDens = dnorm(-100:100/10, mean = -0.5642705, sd = 2.12)
lines(-100:100/10, normFitDens)

qqnorm(yfc.resp.date.lm2.res, main ="", col = "#00000077", pch = 16)
qqline(yfc.resp.date.lm2.res, col="red")
```




```
#dev.off()
```

The Shapiro-Wilk test provides evidence for departure of normality in the errors.

```
shapiro.test(yfc.resp.date.lm2.res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  yfc.resp.date.lm2.res
## W = 0.92325, p-value = 3.351e-11
```

# Bootstrap analysis

By central limit theorem, the moderate departure from normality, should not be a problem. However, just to be safe, we evaluate the bootstrap esimtates of the regression coefficients, 95% CI and p-values. The bootstrap esimtates does not assume a parametric distribution for the errors and therefore is a suitable alternative when departure of normality would be a problem.

## Simulate bootstrap replicates

We simulated 10000 bootstrap replicates.

```
set.seed(777)
# bootstrapping with 1000 replications
results <- boot(data=covid2019.df, statistic=bs,
   R=10000, formula=log(std.sevendays.cucase) ~ new.arr.time +
                           Bus.resp + new.Bus.date +
                           Enter.resp + new.Enter.date.cat)
```

The bias is generally very small compare to the coefficients, which indicates the bootstrap esimates are very similar to the least squre estimates above. This confirms that the moderate departure of normality is not an issue this case.

```
# view results
print(results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = covid2019.df, statistic = bs, R = 10000, formula = log(std.sevendays.cucase) ~
##     new.arr.time + Bus.resp + new.Bus.date + Enter.resp + new.Enter.date.cat)
##
##
## Bootstrap Statistics :
##        original        bias    std. error
## t1*  -1.1763006 -0.0085818633  1.67019155
## t2*   0.2712503  0.0005880381  0.06638612
## t3* -12.7092503  0.0521117064  4.27279149
## t4*   0.4629233 -0.0019890404  0.16199629
## t5*  -3.4109429 -0.0041846681  1.26479530
## t6*   1.5102706 -0.0008723285  0.47393112
```

The bootstrap estimates for the coefficients are

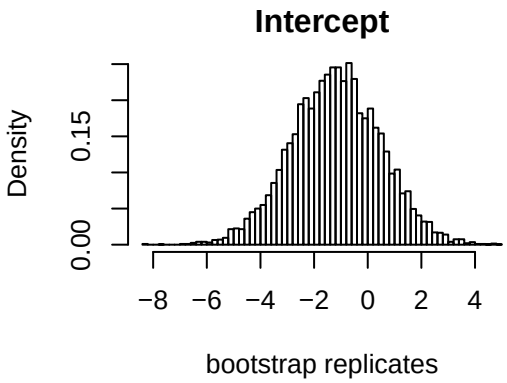
```
round(colMeans(results$t), 2)
```

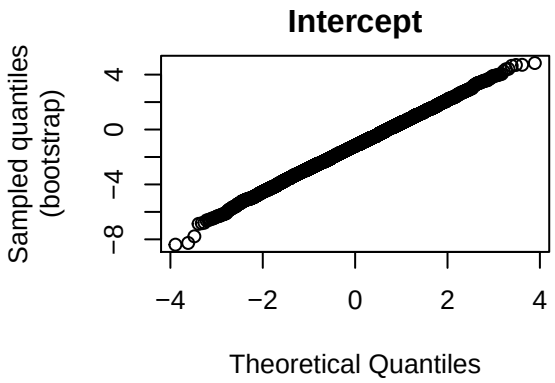
```
## [1]  -1.18   0.27 -12.66   0.46  -3.42   1.51
```

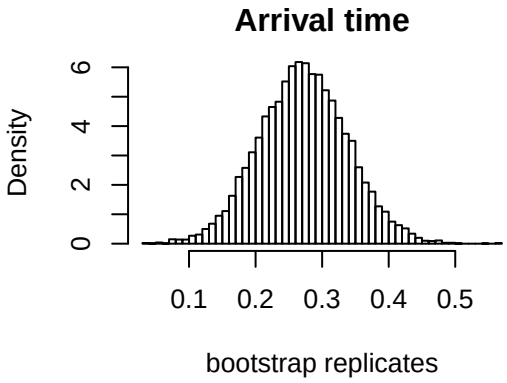
## Normality of the bootstrap statistics

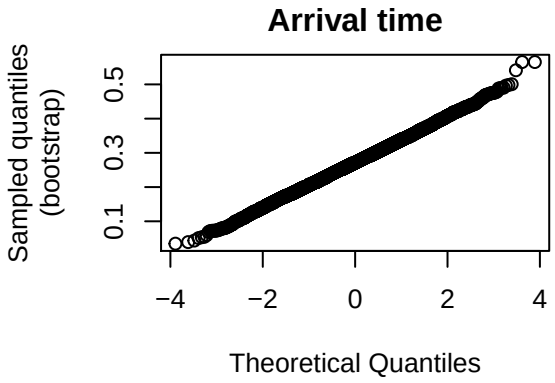
### Intercept and the adjusting variable

```
par(mfrow = c(2,2), mar = c(5, 5, 2, 2) + 0.2)
hist(results$t[,1], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main = "Intercept")
qqnorm(results$t[,1], main =  "Intercept", ylab = "Sampled quantiles\n(bootstrap)")

hist(results$t[,2], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main = "Arrival time")
qqnorm(results$t[,2], main =  "Arrival time", ylab = "Sampled quantiles\n(bootstrap)")
```



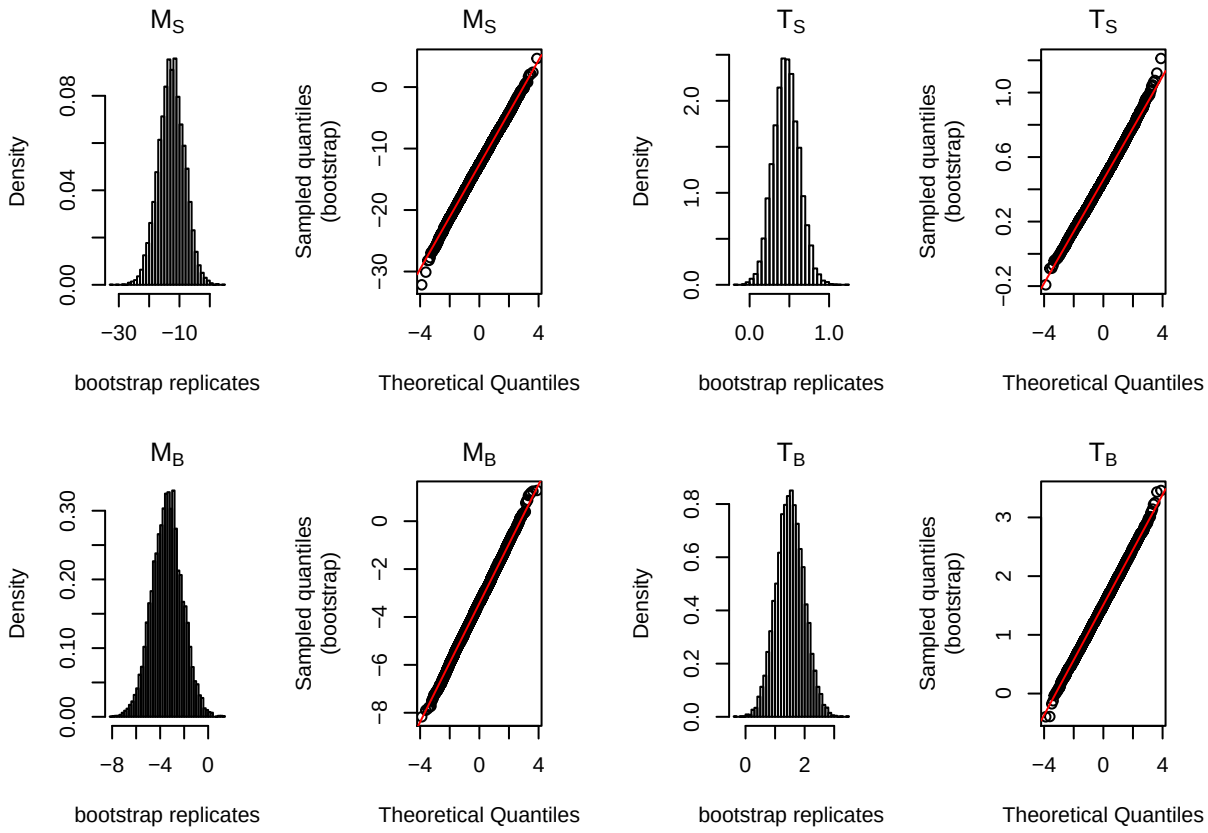




**Transmission control measure variables**

```
par(mfrow = c(2,4), mar = c(5, 5, 2, 2) + 0.2)
hist(results$t[,3], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main = expression(M[S]))
qqnorm(results$t[,3], main = expression(M[S]),
       ylab = "Sampled quantiles\n(bootstrap)")
qqline(results$t[,3], col = "red")

hist(results$t[,4], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main  = expression("T"[S]))
qqnorm(results$t[,4], main = expression("T"[S]),
       ylab = "Sampled quantiles\n(bootstrap)")
qqline(results$t[,4], col = "red")

hist(results$t[,5], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main = expression(M[B]))
qqnorm(results$t[,5], main = expression(M[B]),
       ylab = "Sampled quantiles\n(bootstrap)")
qqline(results$t[,5], col = "red")

hist(results$t[,6], nclass = 50, prob = T,
     xlab = "bootstrap replicates", main  = expression("T"[B]))
qqnorm(results$t[,6], main = expression("T"[B]),
       ylab = "Sampled quantiles\n(bootstrap)")
qqline(results$t[,6], col = "red")
```




## Confidence intervals

The confidence intervals calculated by using the adjusted bootstrap percentile for each coefficient is given by

```
# get 95% confidence intervals
boot.ci.results.tab = sapply(c(1:6),
       function(varIndex  = NULL){
         boot.ci(results, type = "bca", index = varIndex)$bca
       })
boot.ci.tab = t(boot.ci.results.tab)[,4:5]
rownames(boot.ci.tab) = names(coef(yfc.resp.date.lm2))
colnames(boot.ci.tab) = paste(c("95% lower bound", "95% lower bound"))
round(boot.ci.tab,2)
```

```
##                    95% lower bound 95% lower bound
## (Intercept)                  -4.45            2.14
## new.arr.time                  0.14            0.40
## Bus.resp                    -21.37           -4.74
## new.Bus.date                  0.17            0.80
## Enter.resp                   -5.82           -0.86
## new.Enter.date.cat            0.57            2.44
```

## P-values suggested by the confidence intervals

The p-value is defined as 1 minus the highest confidence level that produces a confidence interval excluding 0. While we might be able to assume that the bootstrap statistics follow a normal distribution, this method is chosen so that the p-values and confidence intervals are consistent. The p-values are calculated with sufficient precision to round up to 2 decimal places.

**Intercept**

```
a1 = c(500:520)/1000

ci1 = sapply(a1, function(a){
  boot.ci(results, type= c("bca"),
        conf = a, index=1)$bca
})

round(1 - a1[getSgnChgIndex(ci = ci1, a = a1)],2)
```

```
## [1] 0.49
```

**Arrival time**

```
boot.ci(results, type= c("bca"),
        conf = 0.999, index=2)$bca
```

```
##       conf
## [1,] 0.999 4.51 9995.47 0.05238094 0.4971083
```

```
1 - 0.999
```

```
## [1] 0.001
```

**Binary variable for suspension of the intra-city public transport**

```
a3 = c(9980:9990)/10000
ci3 = sapply(a3, function(a){
   boot.ci(results, type= c("bca"),
         conf = a, index=3)$bca
  })
#ci3[,c(3:4)]
#1-a3[3]

1 - a3[getSgnChgIndex(ci = ci3, a = a3)]
```

```
## [1] 0.0018
```

**Timing of suspension of the intra-city public transport**

```
a4 = c(9970:9980)/10000
ci4 = sapply(a4, function(a){
   boot.ci(results, type= c("bca"),
         conf = a, index=4)$bca
  })

1 - a4[getSgnChgIndex(ci = ci4, a = a4)]
```

```
## [1] 0.0023
```

**Binary variable for closure of entertainment venues and banning of public gatherings**

```
a5 = c(9900:9910)/10000
```

```
ci5 = sapply(a5, function(a){
  boot.ci(results, type= c("bca"),
        conf = a, index=5)$bca
})

1 - a5[getSgnChgIndex(ci = ci5, a = a5)]
```

```
## [1] 0.0098
```

**Timing of closure of entertainment venues and banning of public gatherings**

```
a6 = c(990:999)/1000

ci6 = sapply(a6, function(a){
  boot.ci(results, type= c("bca"),
        conf = a, index=6)$bca
})

1 - a6[getSgnChgIndex(ci = ci6, a = a6)]
```

```
## [1] 0.002
```