Final Project

"05/11/2020"

- Part1 USA national level analysis
- 1. Build Model 1 between USA cumulative positive cases and the date between 3/16/2020 to 3/29/2020.

```r
#1. Data pre-processing for training data of model 1 (M1)
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

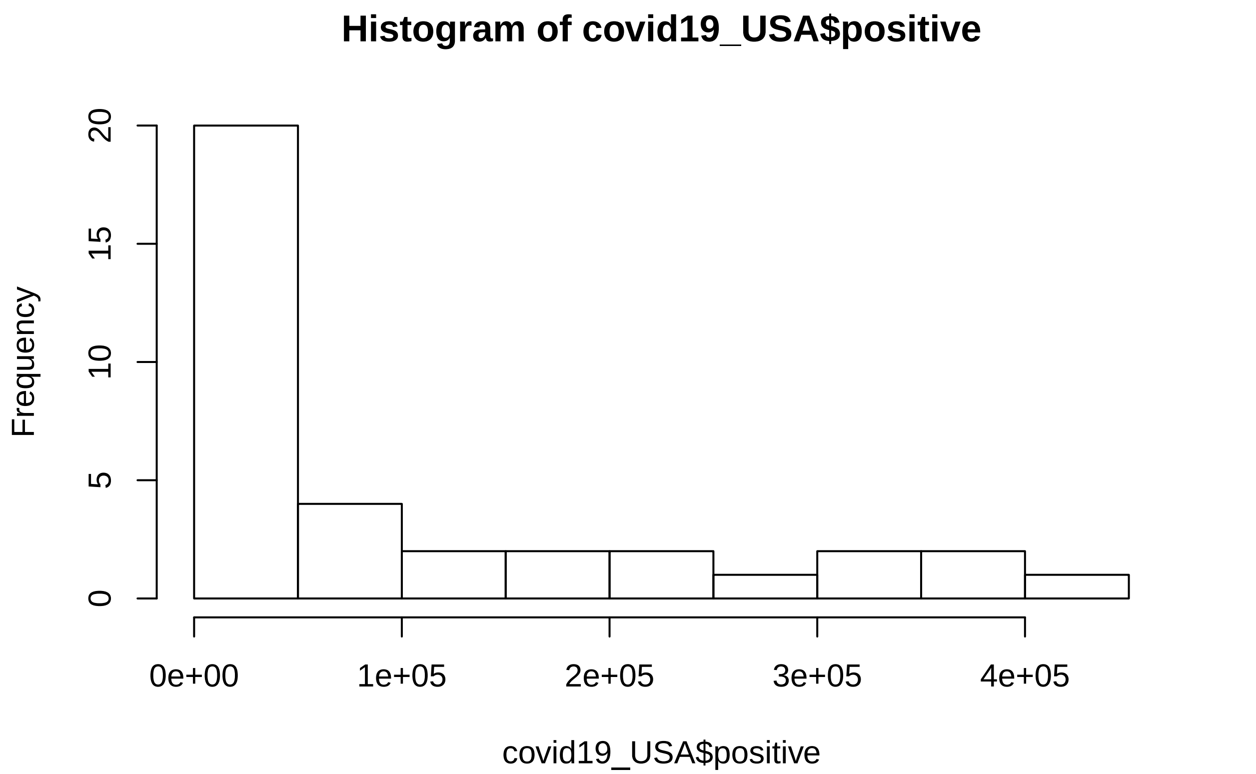
```r
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.6.2

##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```r
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```r
library(ggplot2)
covid19_USA = fread('/Users/likehang/Desktop/lr final project/COVID-19_0408_USA.csv', data.table = FALSE
covid19_USA$date = as.Date(as.character(covid19_USA$date),tryFormats = "%Y%m%d")
covid19_USA = covid19_USA[order(as.Date(covid19_USA$date, format="%d/%m/%Y")),]
hist(covid19_USA$positive)
```




```r
subset1 = covid19_USA$date>="2020-03-16" & covid19_USA$date <= "2020-03-29"
train1_USA = covid19_USA[subset1,] %>% dplyr::select('date','positive')
summary(train1_USA)
```

```
##       date               positive
##  Min.   :2020-03-16   Min.   :  4019
##  1st Qu.:2020-03-19   1st Qu.: 13048
##  Median :2020-03-22   Median : 37016
##  Mean   :2020-03-22   Mean   : 49770
##  3rd Qu.:2020-03-25   3rd Qu.: 76533
##  Max.   :2020-03-29   Max.   :139061
```

```r
#2. Build linear regression model 1 (M1)
M1 = lm(positive ~ date, data = train1_USA)
summary(M1)
```

```
##
## Call:
## lm(formula = positive ~ date, data = train1_USA)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -13088 -11299  -3543   9919  23113
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.867e+08  1.613e+07  -11.57 7.22e-08 ***
## date         1.018e+04  8.794e+02   11.58 7.20e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 13260 on 12 degrees of freedom
## Multiple R-squared:  0.9178, Adjusted R-squared:  0.911 
## F-statistic:   134 on 1 and 12 DF,  p-value: 7.203e-08
```

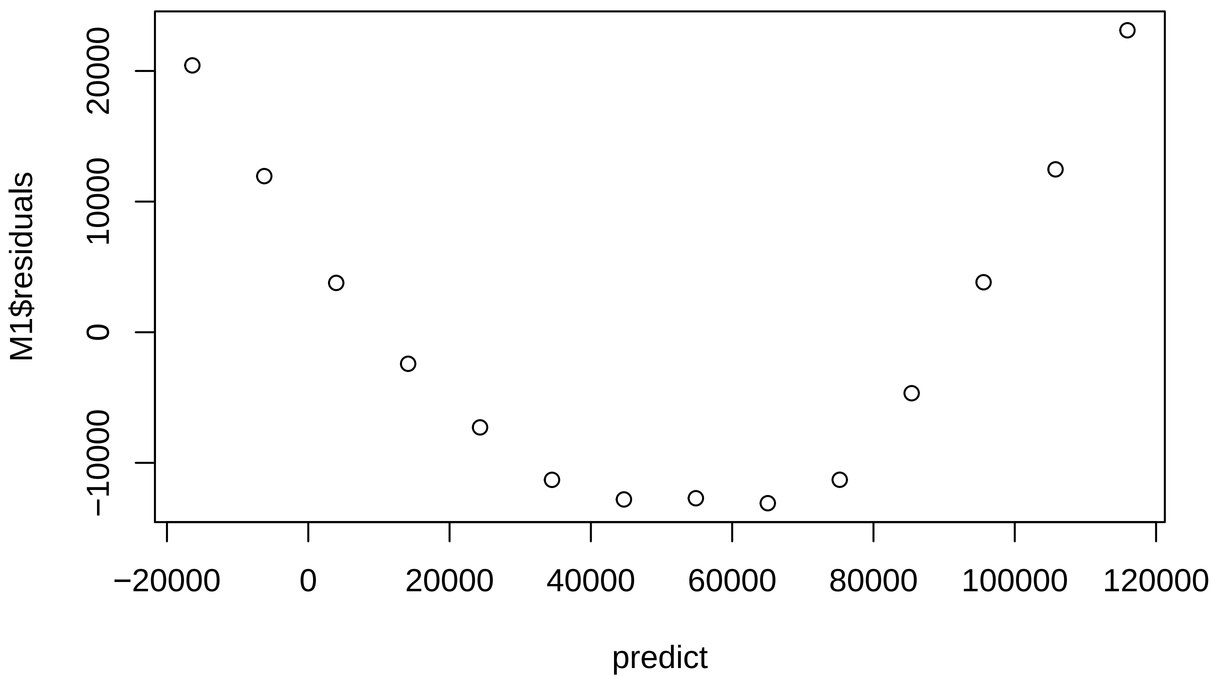
```r
anova(M1)
```

```
## Analysis of Variance Table
## 
## Response: positive
##           Df     Sum Sq    Mean Sq F value    Pr(>F)    
## date       1 2.3582e+10 2.3582e+10  134.04 7.203e-08 ***
## Residuals 12 2.1112e+09 1.7594e+08                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

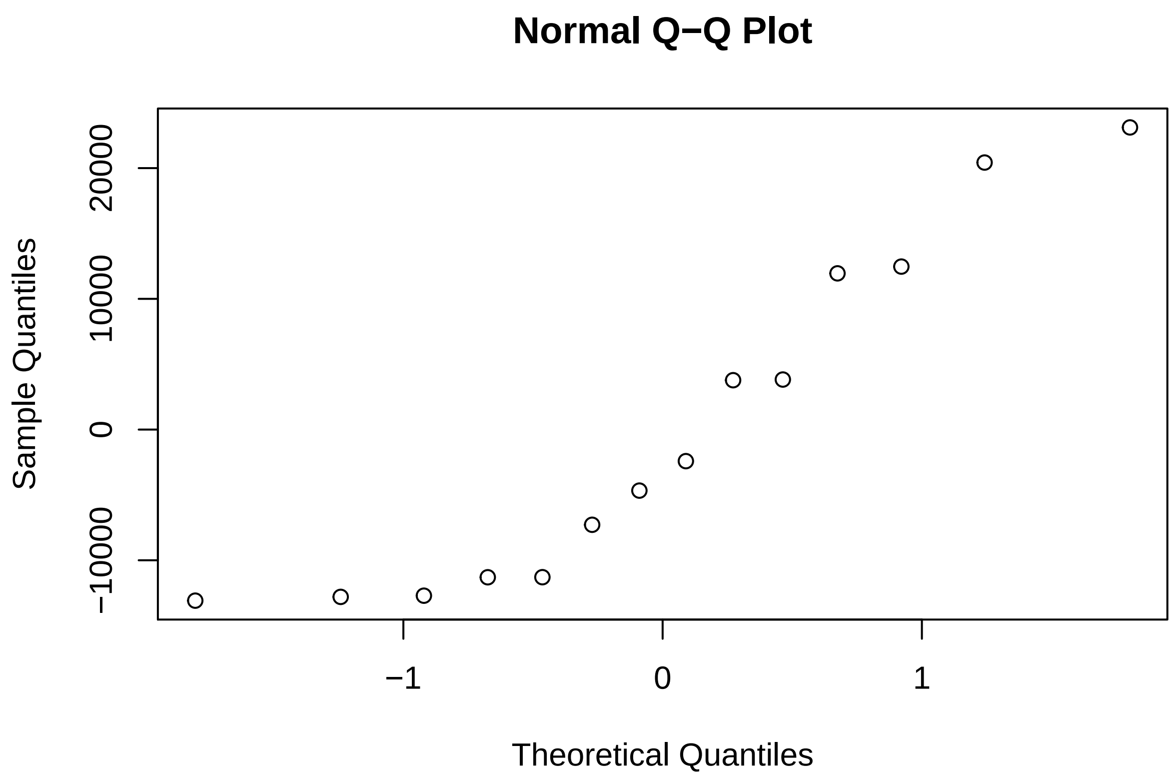
```r
confint(M1)
```

```
##                     2.5 %        97.5 %
## (Intercept) -2.218565e+08 -151562483.58
## date         8.265176e+03      12097.27
```

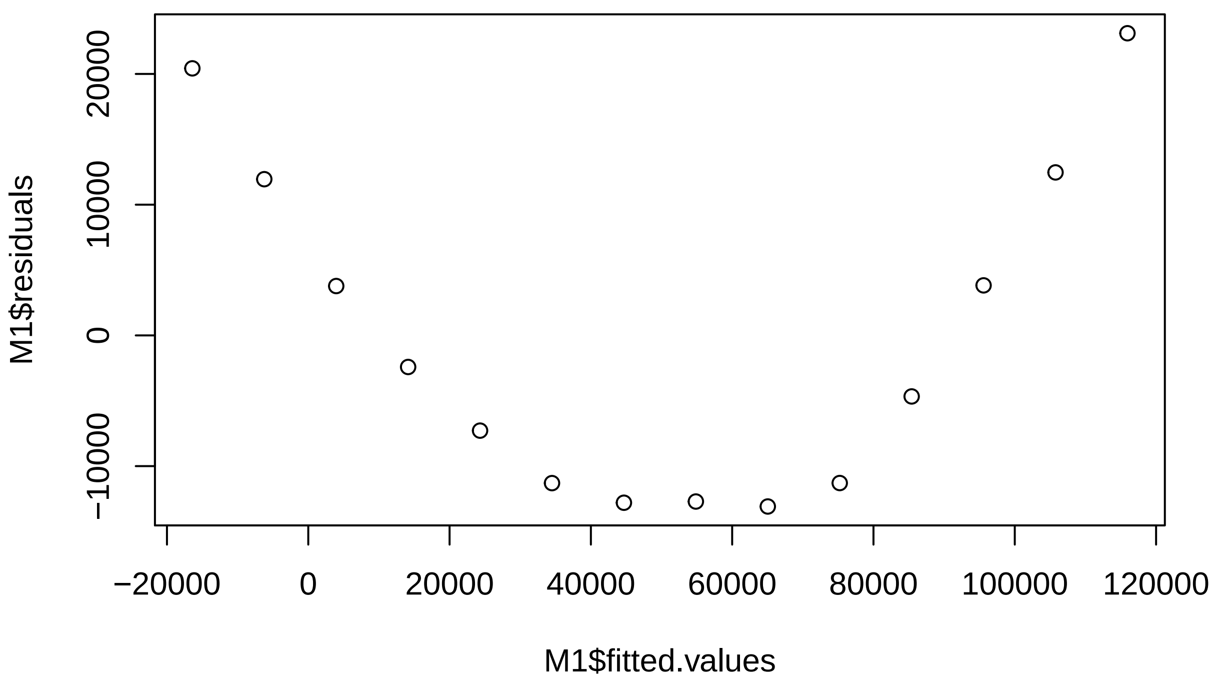
```r
#3. Assumption diagnosis of model 1 (M1)
##3.1 Prediction value vs residuals, test the assumption of linearity
predict = predict(M1, newdata = train1_USA)
plot(predict, M1$residuals)
```




```r
##3.2 QQ Plot, test the assumption that error terms are normally distributed
qqnorm(M1$residuals)
```




```
##3.3 Residuals vs. Fits Plot, test the assumption of homoscedasticity
plot(M1$fitted.values, M1$residuals)
```

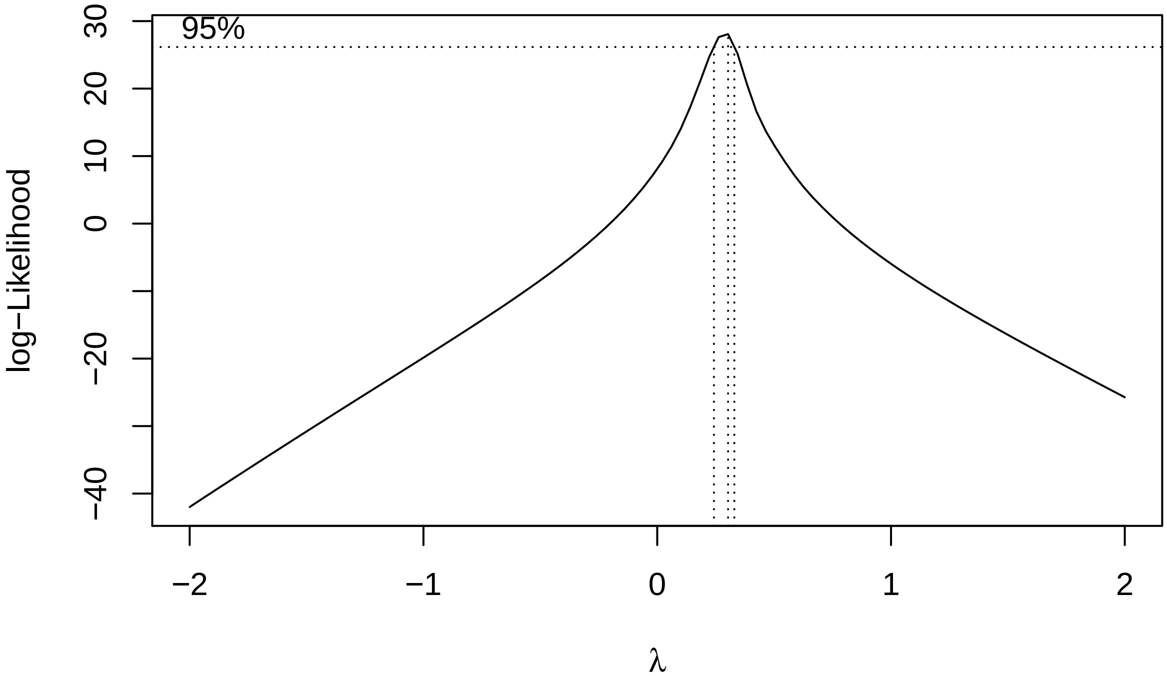


- 2. Build Model 2 between USA cumulative positive cases and the date between 3/16/2020 to 3/29/2020.

```
# Because the M1 assumption tests are not satisfied, try transformed regression model 2 (M2)
#1. Transform "positive" variable and build transformed regression model 2
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
boxcox_fit = boxcox(M1)
```




```
lam1 = boxcox_fit$x[which.max(boxcox_fit$y)]
lam1
```

```
## [1] 0.3030303
```

```
train1_USA$positive = (train1_USA$positive^lam1-1)/lam1
M2 = lm(positive ~ date, data = train1_USA)
summary(M2)
```

```
##
## Call:
## lm(formula = positive ~ date, data = train1_USA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5352 -0.5055  0.1068  0.3284  1.9612
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.140e+05  1.081e+03  -105.5   <2e-16 ***
## date         6.220e+00  5.894e-02   105.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.889 on 12 degrees of freedom
## Multiple R-squared:  0.9989, Adjusted R-squared:  0.9988
```

```
## F-statistic: 1.114e+04 on 1 and 12 DF,  p-value: < 2.2e-16
```

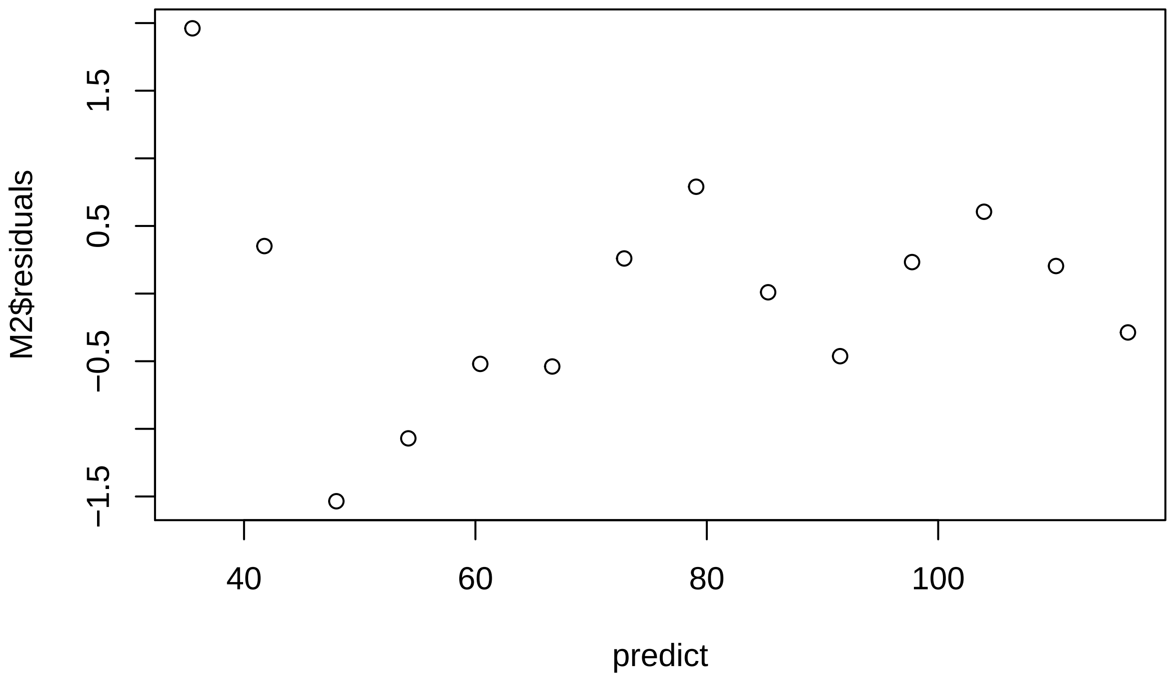
```r
anova(M2)
```

```
## Analysis of Variance Table
##
## Response: positive
##           Df Sum Sq Mean Sq F value    Pr(>F)
## date       1 8802.2  8802.2   11138 < 2.2e-16 ***
## Residuals 12    9.5     0.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

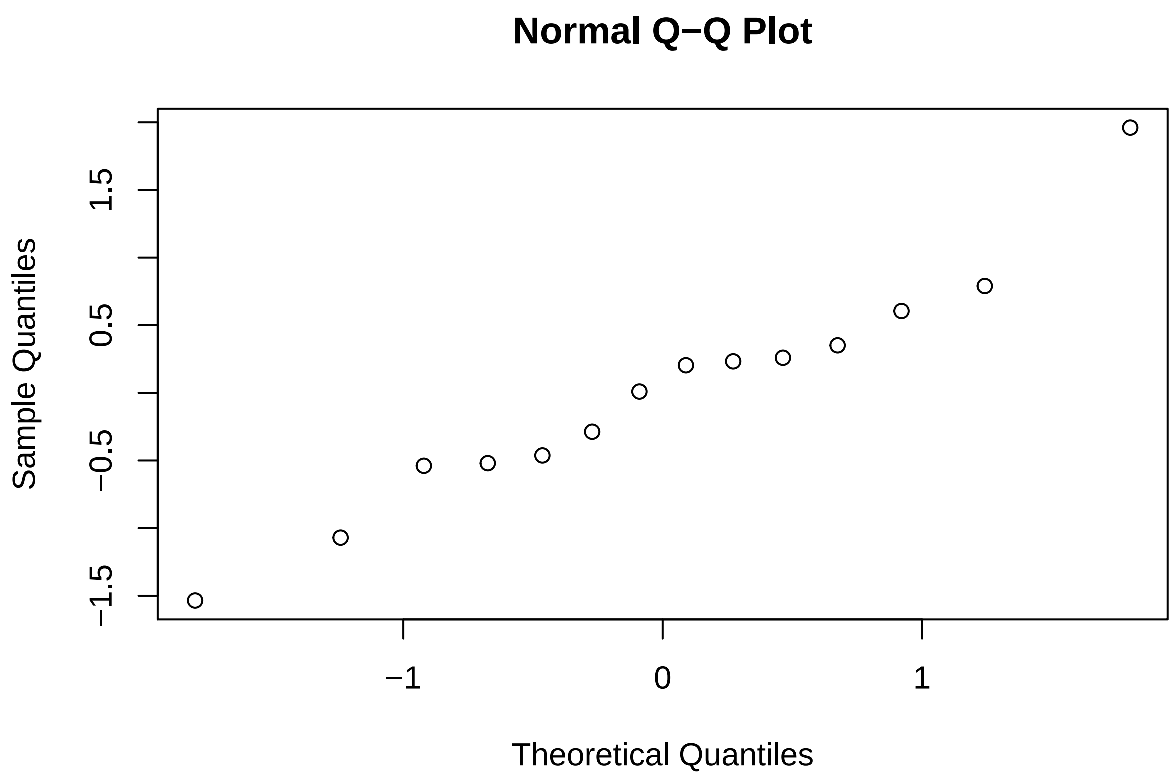
```r
confint(M2)
```

```
##                     2.5 %        97.5 %
## (Intercept) -1.163802e+05 -1.116691e+05
## date         6.091806e+00  6.348636e+00
```

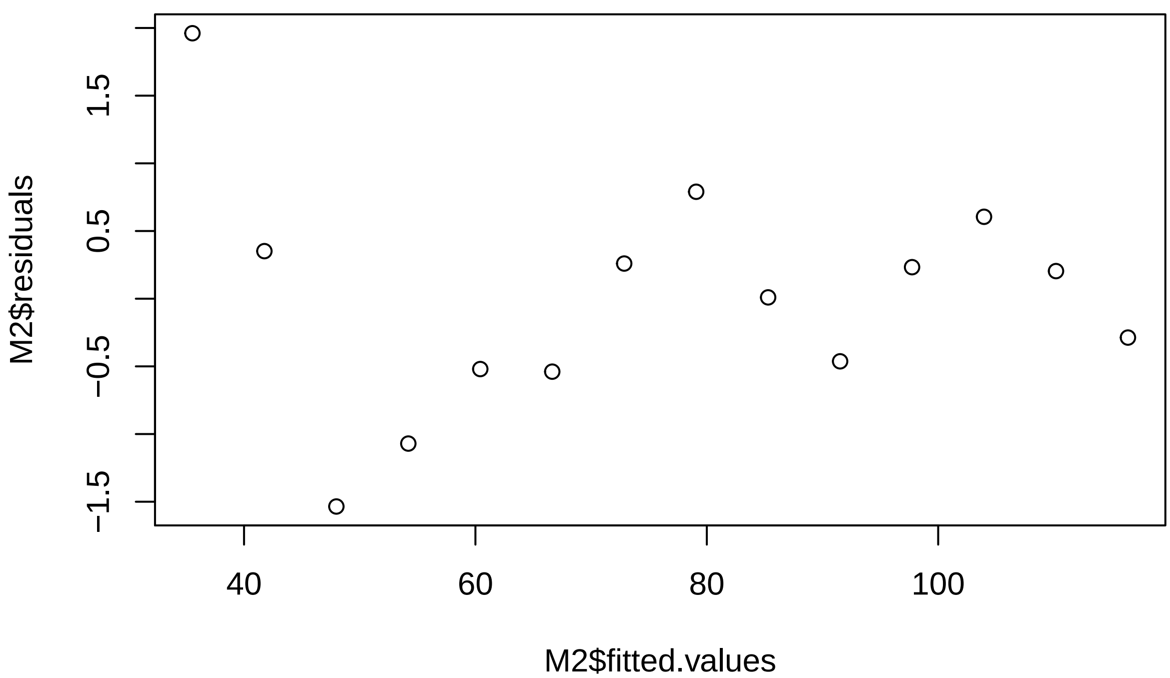
```r
#2. Assumption diagnosis
##2.1 Prediction value vs residuals, test the assumption of linearity
predict = predict(M2, newdata = train1_USA)
plot(predict, M2$residuals)
```




```r
##2.2 QQ Plot, test the assumption that error terms are normally distributed
qqnorm(M2$residuals)
```




```
##2.3 Residuals vs. Fits Plot, test the assumption of homoscedasticity
plot(M2$fitted.values, M2$residuals)
```



- 3. Model 2 prediction for the date from 03/30/2020 to 04/08/2020

```
#1. Create date sequence from 03/30/2020 to 04/08/2020 and make prediction by model 2
pred_date = seq.Date(from = as.Date('2020-03-30'), to = as.Date('2020-04-08'), by = 'days')
new_USA = data.frame(date = pred_date)
pred_USA = predict(M2, newdata = new_USA, interval = "prediction")
```

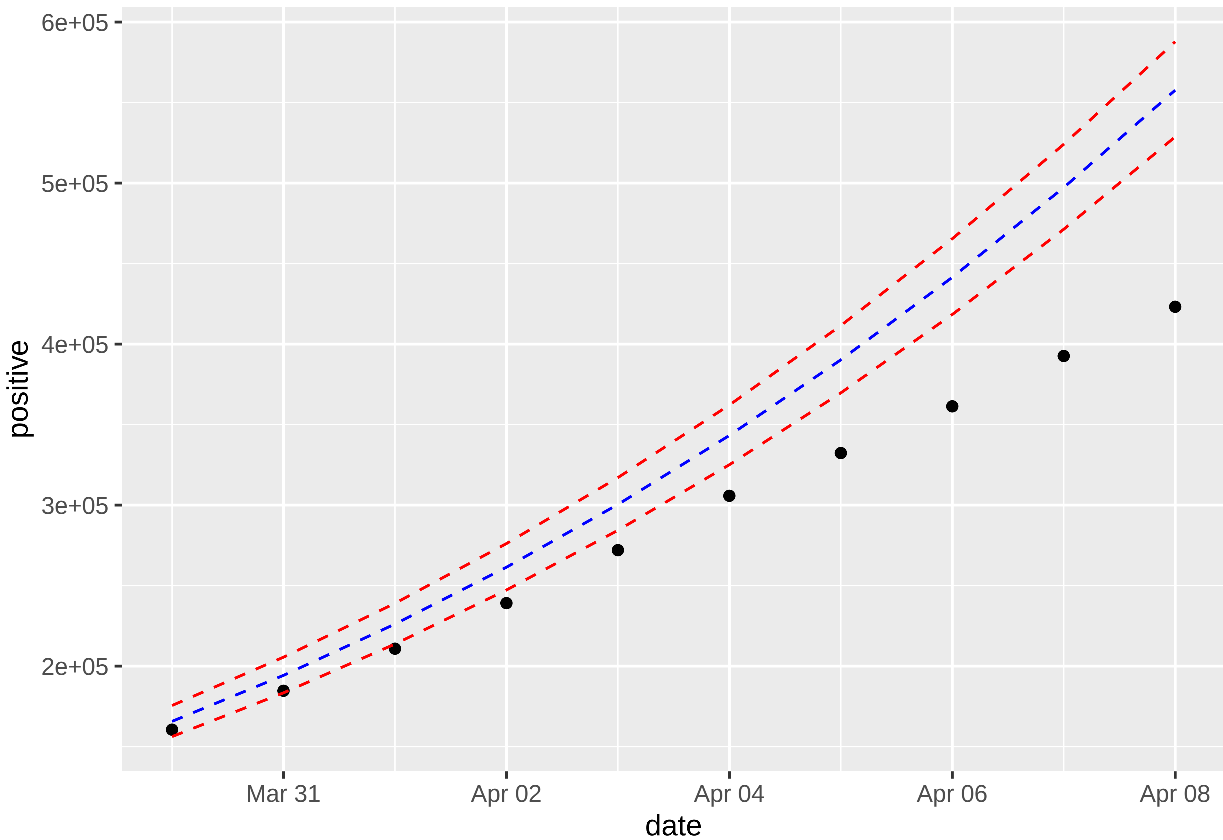
```r
pred_USA
```

```
##         fit      lwr      upr
## 1  122.6229 120.3987 124.8472
## 2  128.8432 126.5604 131.1259
## 3  135.0634 132.7165 137.4103
## 4  141.2836 138.8675 143.6997
## 5  147.5038 145.0138 149.9939
## 6  153.7240 151.1557 156.2923
## 7  159.9443 157.2938 162.5947
## 8  166.1645 163.4283 168.9007
## 9  172.3847 169.5596 175.2098
## 10 178.6049 175.6879 181.5220
```

```r
#2. Transform fitted values and prediction intervals to original scale
pred_USA = (pred_USA*lam1 + 1)^(1/lam1)
pred_USA = data.frame(pred_USA)

#3. Filter the test data for model 2 prediction and combind predict data, test data together
test_USA = covid19_USA[covid19_USA$date>="2020-03-30" & covid19_USA$date <= "2020-04-08", ] %>% dplyr::s
test_USA = cbind(new_USA, pred_USA, test_USA)

#4. Calculate prediction error
test_USA$pred_error = (test_USA$fit - test_USA$positive)^2
M2_pred_error = mean(test_USA$pred_error)
M2_pred_error
```

```
## [1] 4186229169
```

```r
#5. Visualization of model 2 prediction
p <- ggplot(test_USA, aes(date, positive), xlab = "Date", ylab = "USA cumulative count of positive cases
  geom_point()
p + geom_line(aes(y = lwr), color = "red", linetype = "dashed")+
    geom_line(aes(y = upr), color = "red", linetype = "dashed")+
    geom_line(aes(y = fit), color = "blue", linetype = "dashed")
```



- 4. Build Model 3 between USA cumulative positive cases and the date between 3/23/2020 to 3/29/2020.

```
#1. Data processing for training data of model 3 (M4)
subset2 = covid19_USA$date>="2020-03-23" & covid19_USA$date <= "2020-03-29"
train2_USA = covid19_USA[subset2,] %>% dplyr::select('date','positive')
summary(train2_USA)
```

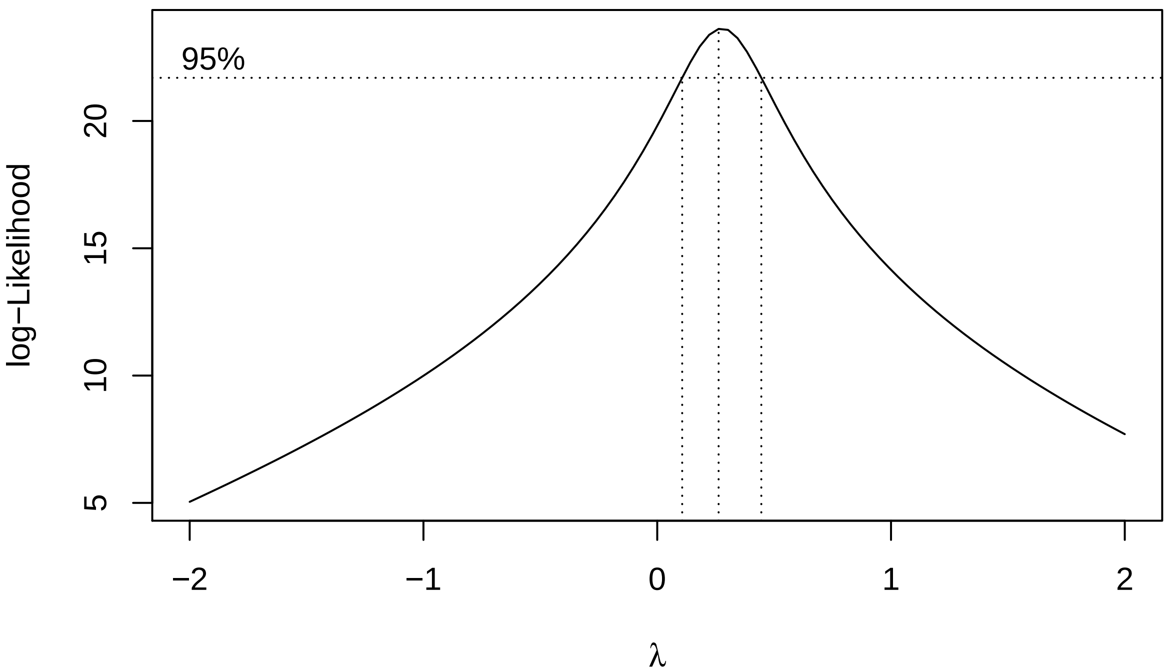
```
##       date               positive
##  Min.   :2020-03-23   Min.   : 42152
##  1st Qu.:2020-03-24   1st Qu.: 57941
##  Median :2020-03-26   Median : 80735
##  Mean   :2020-03-26   Mean   : 85068
##  3rd Qu.:2020-03-27   3rd Qu.:108824
##  Max.   :2020-03-29   Max.   :139061
```

```
#2. Build simple linear regression M3
M3 = lm(positive ~ date, data = train2_USA)
summary(M3)
```

```
##
## Call:
## lm(formula = positive ~ date, data = train2_USA)
##
## Residuals:
##      17      16      15      14      13      12      11
##  6238.0  -344.7 -4755.4 -4333.1 -2039.9   396.4  4838.7
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.005e+08  1.613e+07  -18.64 8.19e-06 ***
## date         1.638e+04  8.789e+02   18.64 8.18e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4651 on 5 degrees of freedom
## Multiple R-squared:  0.9858, Adjusted R-squared:  0.983
## F-statistic: 347.5 on 1 and 5 DF,  p-value: 8.177e-06
```

```
#3. Transfrom variable "positive" and build transformed regression model 3 (M4)
library(MASS)
boxcox_fit = boxcox(M3)
```




```
lam2 = boxcox_fit$x[which.max(boxcox_fit$y)]
lam2
```

```
## [1] 0.2626263
```

```
train2_USA$positive = (train2_USA$positive^lam2-1)/lam2
M4 = lm(positive ~ date, data = train2_USA)
summary(M4)
```

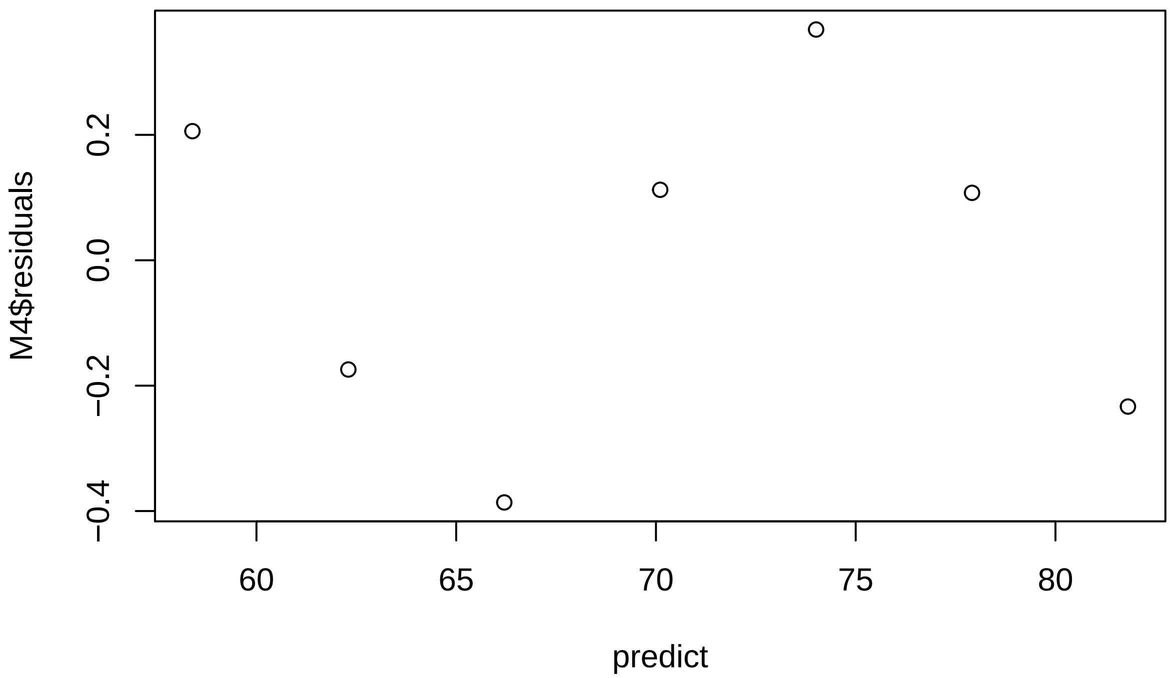
```
##
## Call:
## lm(formula = positive ~ date, data = train2_USA)
##
## Residuals:
##      17      16      15      14      13      12      11
##  0.2059 -0.1742 -0.3863  0.1124  0.3680  0.1075 -0.2334
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.154e+04  1.024e+03  -69.86 1.14e-08 ***
```

```
## date         3.903e+00  5.581e-02   69.93 1.13e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2953 on 5 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.9988
## F-statistic:  4891 on 1 and 5 DF,  p-value: 1.132e-08
```

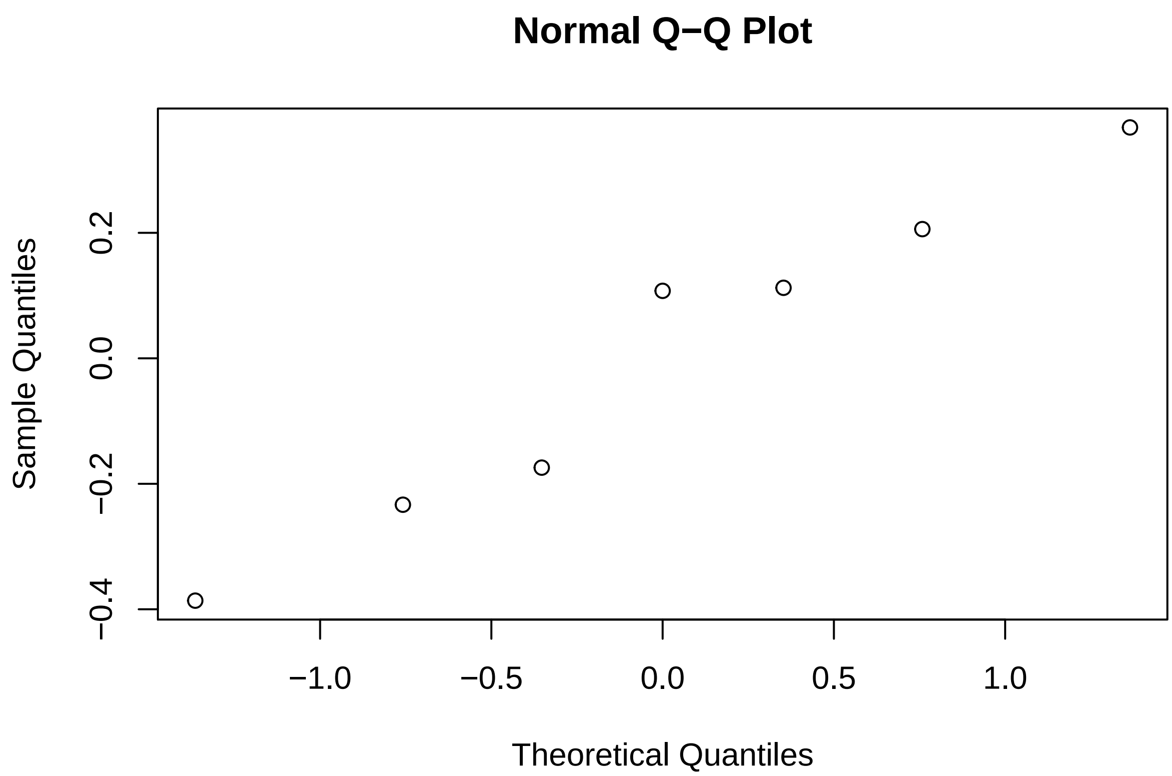
```r
anova(M4)
```

```
## Analysis of Variance Table
##
## Response: positive
##           Df Sum Sq Mean Sq F value    Pr(>F)
## date       1 426.52  426.52  4890.7 1.132e-08 ***
## Residuals  5   0.44    0.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

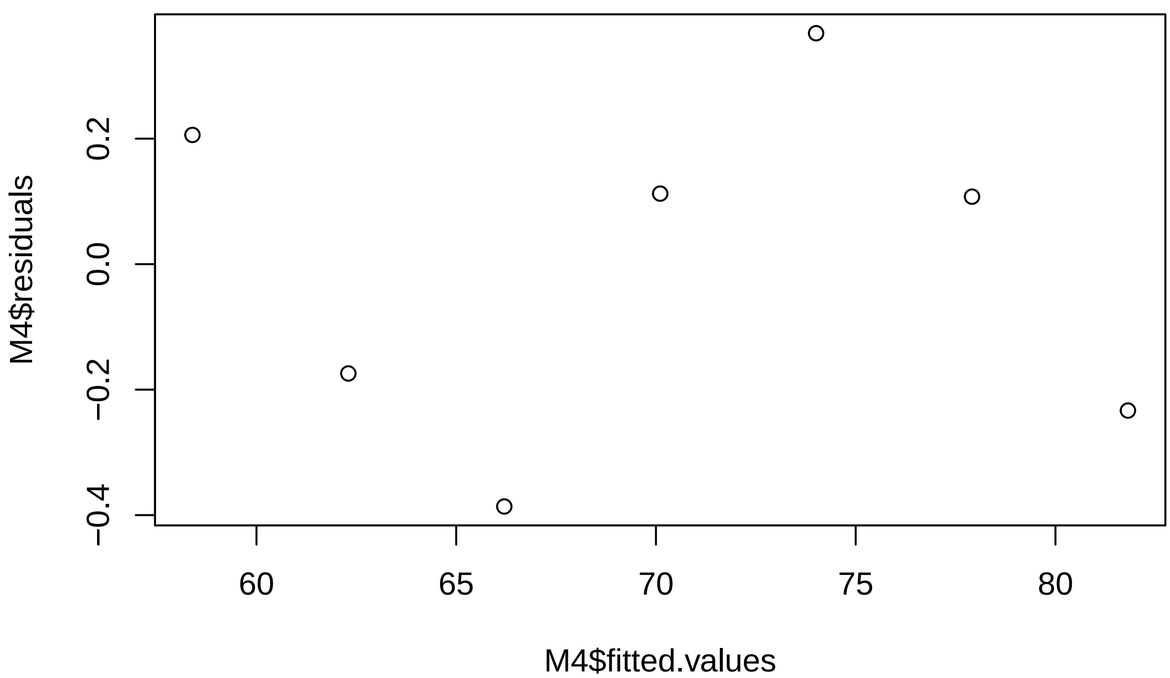
```r
#4. Assumption diagnosis for model 3 (M4)
##4.1 Prediction value vs residuals, test the assumption of linearity
predict = predict(M4, newdata = train2_USA)
plot(predict, M4$residuals)
```




```r
##4.2 QQ Plot, test the assumption that error terms are normally distributed
qqnorm(M4$residuals)
```




```r
##4.3 Residuals vs. Fits Plot, test the assumption of homoscedasticity
plot(M4$fitted.values, M4$residuals)
```



- 5. Model 3 prediction for the date from 03/30/2020 to 04/08/2020

```r
#1. Predict the cumulative count of positive cases between 3/30/2020 to 4/08/2020 by model 3 (M4)
pred2_USA = predict(M4, newdata = new_USA, interval = "prediction")
pred2_USA
```

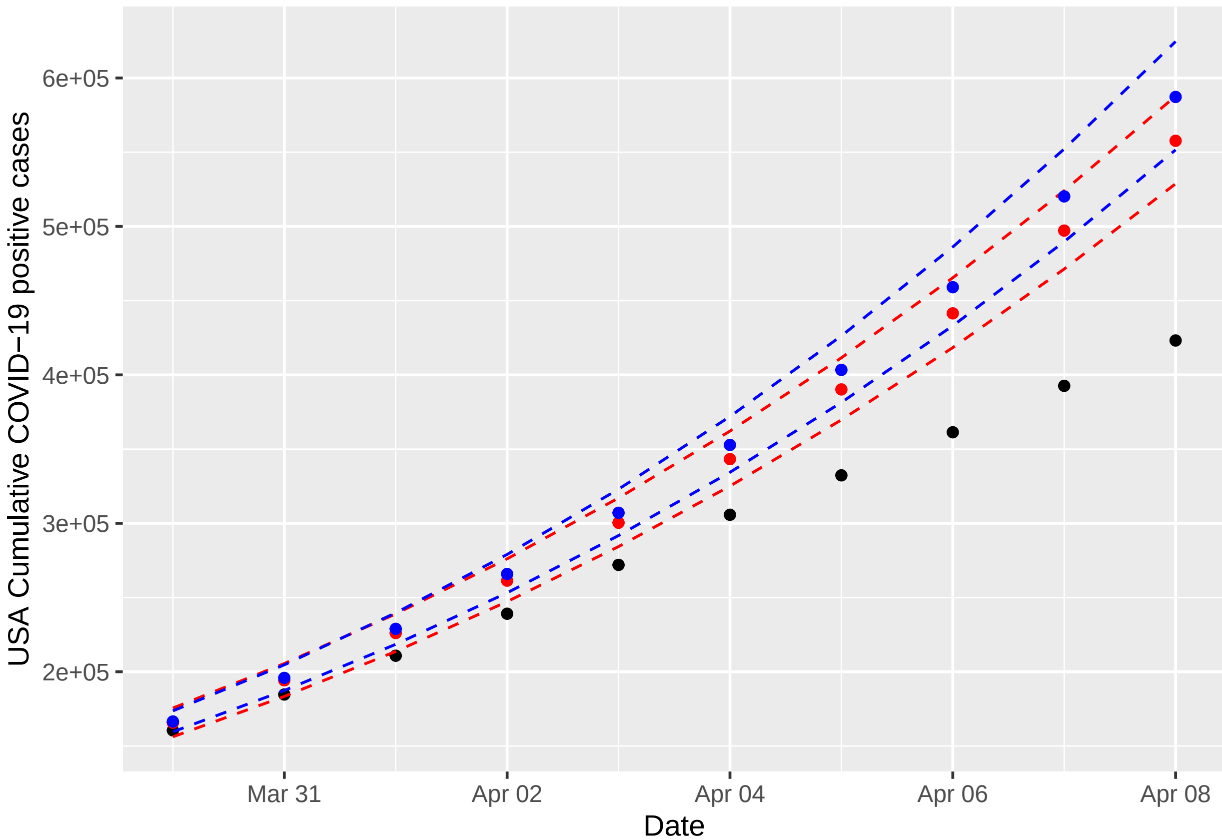
```
##          fit       lwr       upr
## 1   85.71772  84.72379  86.71165
## 2   89.62064  88.53753  90.70375
## 3   93.52356  92.34054  94.70657
## 4   97.42647  96.13532  98.71763
## 5  101.32939  99.92376 102.73502
## 6  105.23231 103.70729 106.75732
## 7  109.13522 107.48698 110.78347
## 8  113.03814 111.26362 114.81266
## 9  116.94106 115.03783 118.84429
## 10 120.84397 118.81006 122.87789
```

```
#2. Transform fitted values and prediction intervals to original scale
pred2_USA = (pred2_USA*lam2 + 1)^(1/lam2)
pred2_USA = data.frame(pred2_USA)

#3. Filter the test data for model 3 (M4) prediction and combind predict data, test data together
test2_USA = covid19_USA[covid19_USA$date>="2020-03-30" & covid19_USA$date <= "2020-04-08", ] %>% dplyr:
test2_USA = cbind(new_USA, pred2_USA, test2_USA)

#4. Calculate prediction error of model 3 (M4)
test2_USA$pred_error = (test2_USA$fit - test2_USA$positive)^2
M4_pred_error = mean(test2_USA$pred_error)
M4_pred_error
```

```
## [1] 6244159238
```

```
#5. Visualiztion of model 2 and model 3 prediction by ggplot2
names(test2_USA)[names(test2_USA)=="fit"] = "fit2"
names(test2_USA)[names(test2_USA)=="upr"] = "upr2"
names(test2_USA)[names(test2_USA)=="lwr"] = "lwr2"
test_USA = merge(test_USA, test2_USA, by = c("date","positive"))

p2 <- ggplot(data = test_USA, aes(x = date, y = positive)) + geom_point() +labs(x ="Date", y = "USA Cum

p2 + geom_line(aes(y = lwr), color = "red", linetype = "dashed")+
    geom_line(aes(y = upr), color = "red", linetype = "dashed")+
    geom_point(aes(y = fit), color = "red", linetype = "dashed")+
    geom_line(aes(y = lwr2), color = "blue", linetype = "dashed")+
    geom_line(aes(y = upr2), color = "blue", linetype = "dashed")+
    geom_point(aes(y = fit2), color = "blue", linetype = "dashed")
```

```
## Warning: Ignoring unknown parameters: linetype

## Warning: Ignoring unknown parameters: linetype
```



- Part2 State level analysis
- 1. Select 5 states with top 5 cumulative positive cases on March 29th

```
#1. Data pre-processing
covid19_state = fread('/Users/likehang/Desktop/lr final project/COVID-19_0408_States.csv', data.table =
covid19_state$date = as.Date(as.character(covid19_state$date),tryFormats = "%Y%m%d")
covid19_state = covid19_state[order(as.Date(covid19_state$date,format="%d/%m/%Y")),]

#2. Find the top5 states with the largest cumulative count of positive cases on March 29, 2020
date_0329 = covid19_state[covid19_state$date == "2020-03-29",]
date_0329 = date_0329[order(date_0329[["positive"]], decreasing = TRUE)[1:5],]
date_0329$state
```

```
## [1] "NY" "NJ" "CA" "MI" "MA"
```

- 2. Build Model 4 between state cumulative positive cases and the date between 3/23/2020 to 3/29/2020.

```
#1. #1. Data processing for training data of model 4 (M6)
subset1 = covid19_state$state %in% c('NJ', 'NY', "CA", "MI", "MA")
subset2 = covid19_state$date>="2020-03-23" & covid19_state$date <= "2020-03-29"
train1_state= covid19_state[subset1 & subset2,] %>% dplyr::select('date','positive','state')
train1_state$state = as.factor(train1_state$state)

#2. Build simple linear regression M5
M5 = lm(positive ~ date + state, data = train1_state)
summary(M5)
```

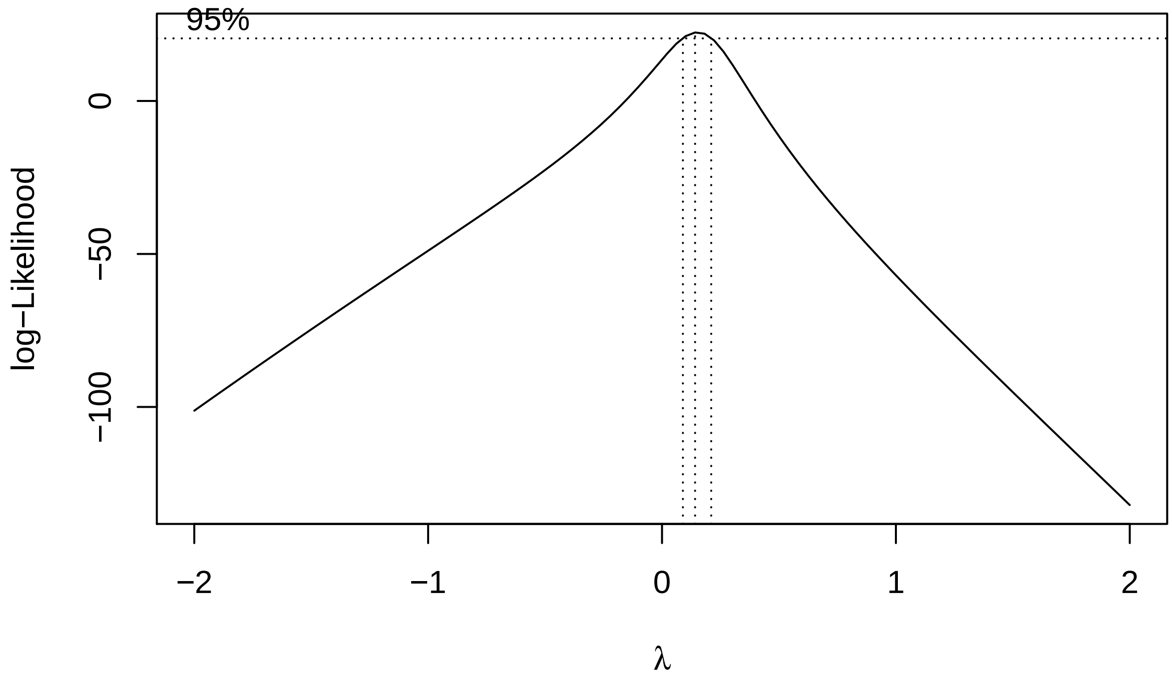
```
##
```

```
## Call:
## lm(formula = positive ~ date + state, data = train1_state)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11630.9  -2046.5   -326.6   2200.1  14568.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.803e+07  7.828e+06  -4.858 3.76e-05 ***
## date         2.073e+03  4.266e+02   4.859 3.75e-05 ***
## stateMA     -6.833e+02  2.698e+03  -0.253    0.802
## stateMI     -3.367e+02  2.698e+03  -0.125    0.902
## stateNJ      3.958e+03  2.698e+03   1.467    0.153
## stateNY      3.538e+04  2.698e+03  13.111 1.02e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5048 on 29 degrees of freedom
## Multiple R-squared:  0.9094, Adjusted R-squared:  0.8938
## F-statistic: 58.24 on 5 and 29 DF,  p-value: 3.092e-14
```

```r
anova(M5)
```

```
## Analysis of Variance Table
##
## Response: positive
##           Df     Sum Sq    Mean Sq F value    Pr(>F)
## date       1  601638498  601638498  23.609 3.755e-05 ***
## state      4 6819746419 1704936605  66.903 3.212e-14 ***
## Residuals 29  739031130   25483832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#3. Transfrom variable "positive" and build transformed regression model 4 (M6)
library(MASS)
boxcox_fit = boxcox(M5)
```




```r
lam3 = boxcox_fit$x[which.max(boxcox_fit$y)]
lam3
```

```
## [1] 0.1414141
```

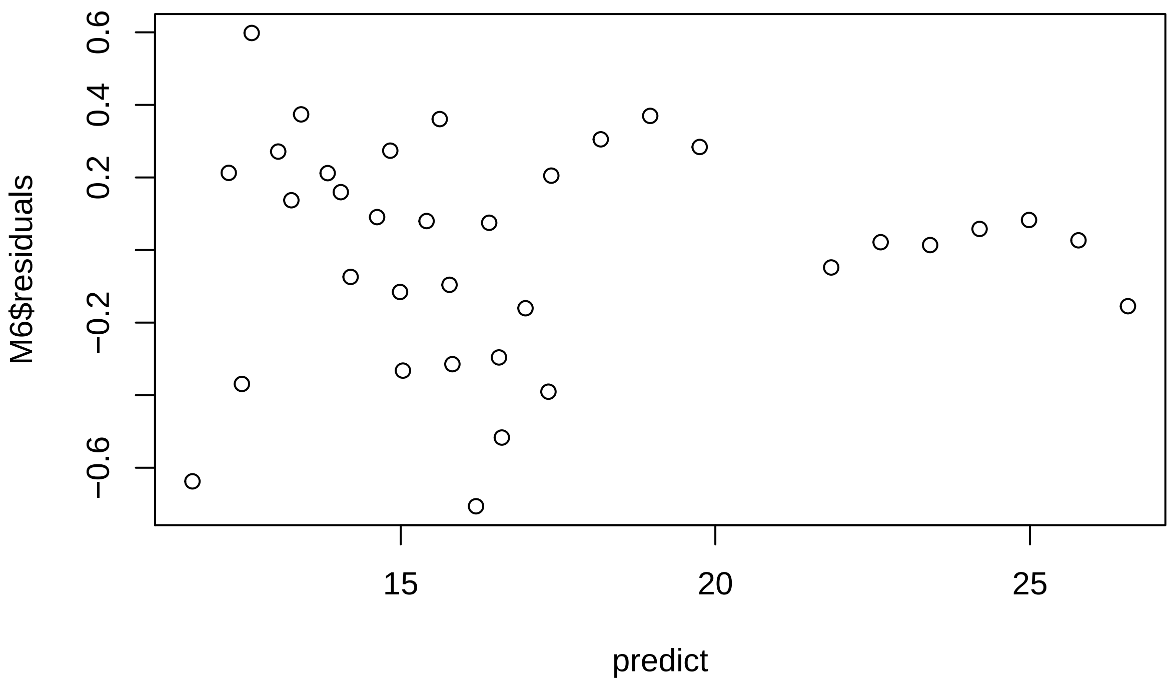
```r
train1_state$positive = (train1_state$positive^lam3-1)/lam3
M6 = lm(positive ~ date + state, data = train1_state)
summary(M6)
```

```
##
## Call:
## lm(formula = positive ~ date + state, data = train1_state)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70615 -0.15756  0.05813  0.21229  0.59809
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.441e+04  5.096e+02 -28.266  < 2e-16 ***
## date         7.860e-01  2.778e-02  28.295  < 2e-16 ***
## stateMA     -9.415e-01  1.757e-01  -5.359 9.35e-06 ***
## stateMI     -3.645e-01  1.757e-01  -2.075    0.047 *
## stateNJ      2.404e+00  1.757e-01  13.681 3.52e-14 ***
## stateNY      9.210e+00  1.757e-01  52.421  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3287 on 29 degrees of freedom
## Multiple R-squared:  0.9946, Adjusted R-squared:  0.9937
## F-statistic:  1072 on 5 and 29 DF,  p-value: < 2.2e-16
```

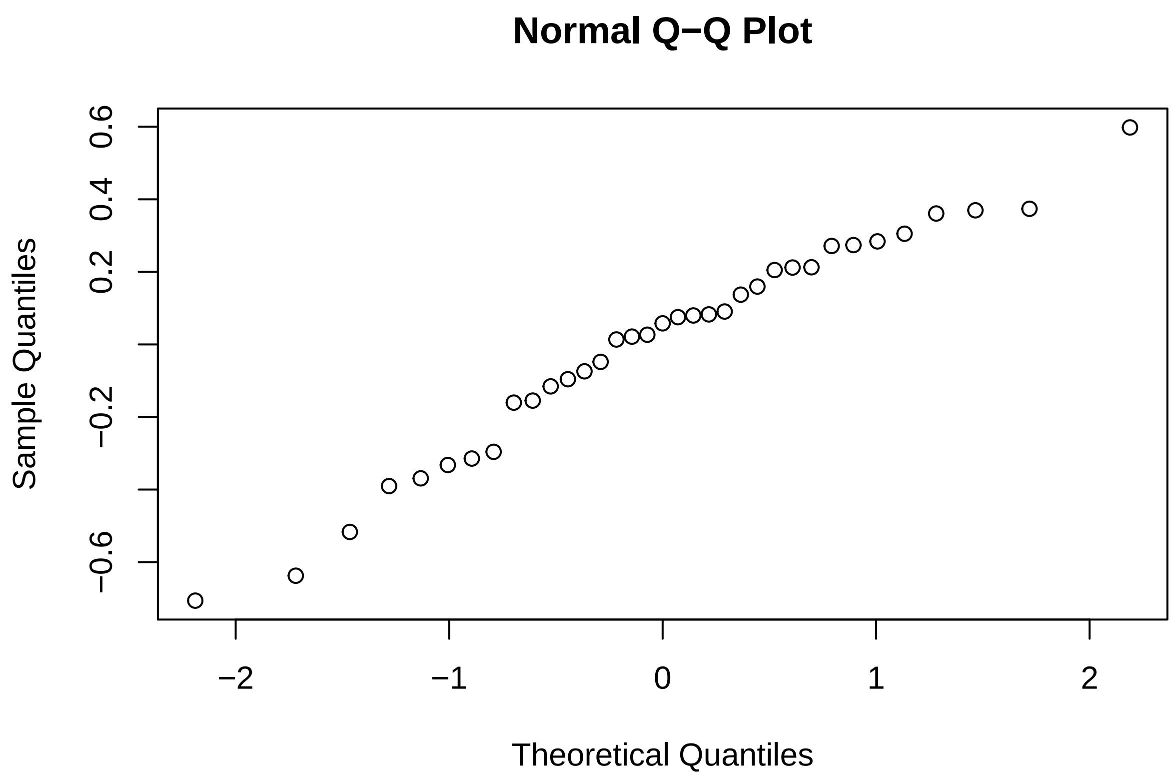
```r
confint(M6)
```

```
##                     2.5 %        97.5 %
## (Intercept) -1.544810e+04 -1.336340e+04
## date         7.291867e-01  8.428127e-01
## stateMA     -1.300857e+00 -5.822221e-01
## stateMI     -7.238550e-01 -5.220481e-03
## stateNJ      2.044233e+00  2.762868e+00
## stateNY      8.850253e+00  9.568888e+00
```

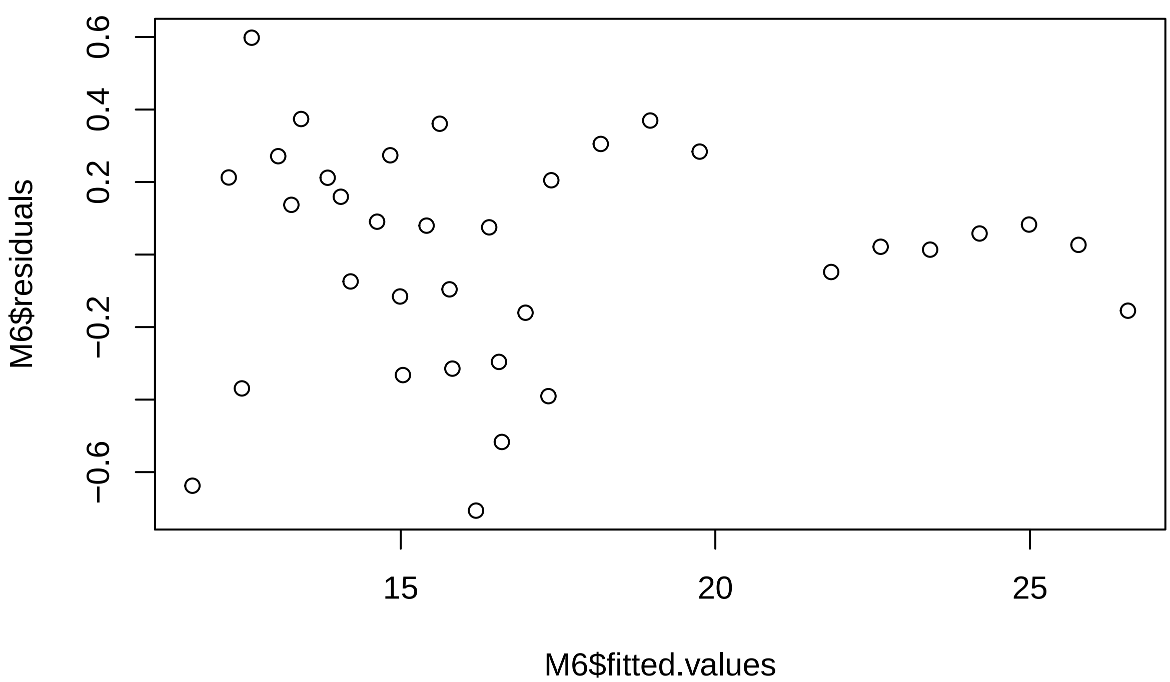
```r
#4. Assumption diagnosis for model 4 (M6)
##4.1 Prediction value vs residuals, test the assumption of linearity
predict = predict(M6, newdata = train1_state)
plot(predict, M6$residuals)
```




```r
##4.2 QQ Plot, test the assumption that error terms are normally distributed
qqnorm(M6$residuals)
```




```
##4.3 Residuals vs. Fits Plot, test the assumption of homoscedasticity
plot(M6$fitted.values, M6$residuals)
```



- 5. Model 4 prediction for the date from 03/30/2020 to 04/08/2020 in each state

```
#1. Predict the cumulative positive cases in each state between 3/30/2020 to 4/08/2020 by model 4 (M6)
pred_date = seq.Date(from = as.Date('2020-03-30'), to = as.Date('2020-04-08'), by = 'days')
state_5 = factor(rep(c('NJ', 'NY', "CA", "MI", "MA"), 10))
new_state = data.frame(date = rep(pred_date,each = 5), state = state_5)
```

```r
pred1_state = predict(M6, newdata = new_state, interval = "prediction")
pred1_state
```

```
##         fit      lwr      upr
## 1  20.53711 19.78340 21.29082
## 2  27.34313 26.58942 28.09684
## 3  18.13356 17.37985 18.88727
## 4  17.76902 17.01531 18.52273
## 5  17.19202 16.43831 17.94573
## 6  21.32311 20.55037 22.09585
## 7  28.12913 27.35639 28.90187
## 8  18.91956 18.14682 19.69230
## 9  18.55502 17.78228 19.32776
## 10 17.97802 17.20528 18.75076
## 11 22.10911 21.31372 22.90449
## 12 28.91513 28.11974 29.71051
## 13 19.70556 18.91017 20.50094
## 14 19.34102 18.54564 20.13640
## 15 18.76402 17.96863 19.55940
## 16 22.89511 22.07377 23.71644
## 17 29.70113 28.87979 30.52246
## 18 20.49156 19.67022 21.31289
## 19 20.12702 19.30568 20.94835
## 20 19.55002 18.72868 20.37135
## 21 23.68111 22.83081 24.53141
## 22 30.48713 29.63683 31.33743
## 23 21.27756 20.42726 22.12786
## 24 20.91302 20.06272 21.76332
## 25 20.33602 19.48572 21.18632
## 26 24.46711 23.58513 25.34908
## 27 31.27313 30.39115 32.15510
## 28 22.06356 21.18158 22.94553
## 29 21.69902 20.81704 22.58099
## 30 21.12202 20.24004 22.00399
## 31 25.25311 24.33702 26.16919
## 32 32.05913 31.14304 32.97521
## 33 22.84955 21.93347 23.76564
## 34 22.48502 21.56893 23.40110
## 35 21.90802 20.99193 22.82410
## 36 26.03911 25.08674 26.99147
## 37 32.84513 31.89276 33.79749
## 38 23.63555 22.68319 24.58791
## 39 23.27102 22.31866 24.22338
## 40 22.69402 21.74166 23.64638
## 41 26.82510 25.83454 27.81567
## 42 33.63112 32.64056 34.62169
## 43 24.42155 23.43099 25.41212
## 44 24.05702 23.06645 25.04759
## 45 23.48001 22.48945 24.47058
## 46 27.61110 26.58061 28.64160
## 47 34.41712 33.38663 35.44762
## 48 25.20755 24.17706 26.23805
## 49 24.84302 23.81252 25.87351
## 50 24.26601 23.23552 25.29651
```

```r
#2. Transform fitted values and prediction intervals to original scale
train1_state$positive = (train1_state$positive*lam3 + 1)^(1/lam3)
pred1_state = (pred1_state*lam3 + 1)^(1/lam3)
pred1_state = data.frame(pred1_state)

#3. Filter the test data for model 4 (M6) prediction and combind predict data, test data together
subset1 = covid19_state$state %in% c('NJ', 'NY', "CA", "MI", "MA")
subset2 = covid19_state$date>="2020-03-30" & covid19_state$date <= "2020-04-08"
test1_state= covid19_state[subset1 & subset2,] %>% dplyr::select('positive','state', 'date')
test1_state = merge(cbind(new_state, pred1_state), test1_state, by=c("date","state"))
test1_state = test1_state[order(test1_state$state),]

test1_state$pred_error = (test1_state$fit - test1_state$positive)^2
M6_pred_error_CA = mean(test1_state$pred_error[test1_state$state == "CA"])
M6_pred_error_NY = mean(test1_state$pred_error[test1_state$state == "NY"])
M6_pred_error_NJ = mean(test1_state$pred_error[test1_state$state == "NJ"])
M6_pred_error_MI = mean(test1_state$pred_error[test1_state$state == "MI"])
M6_pred_error_MA = mean(test1_state$pred_error[test1_state$state == "MA"])
M6_pred_error_CA
```

```
## [1] 209466042
```

```r
M6_pred_error_NY
```

```
## [1] 3732561929
```

```r
M6_pred_error_NJ
```

```
## [1] 158225864
```

```r
M6_pred_error_MI
```

```
## [1] 107138808
```

```r
M6_pred_error_MA
```

```
## [1] 96893270
```

```r
ratio_CA = sqrt(M6_pred_error_CA) / mean(test1_state$positive[test1_state$state == "CA"])
ratio_MI = sqrt(M6_pred_error_MI) / mean(test1_state$positive[test1_state$state == "MI"])
ratio_MA = sqrt(M6_pred_error_MA) / mean(test1_state$positive[test1_state$state == "MA"])
ratio_NY = sqrt(M6_pred_error_NY) / mean(test1_state$positive[test1_state$state == "NY"])
ratio_NJ = sqrt(M6_pred_error_NJ) / mean(test1_state$positive[test1_state$state == "NJ"])
ratio_CA
```

```
## [1] 1.262931
```

```r
ratio_MA
```

```
## [1] 0.8985912
```

```r
ratio_MI
```

```
## [1] 0.7755607
```

```r
ratio_NY
```

```
## [1] 0.5678732
```

```r
ratio_NJ
```

```
## [1] 0.3960026
```

```r
test1_state = rbind.fill(test1_state, train1_state )
test1_state = test1_state[order(test1_state$state),]
```

```r
#4. Filter the test data for model 4 (M6) prediction and combind predict data, test data together
pred_date = seq.Date(from = as.Date('2020-03-23'), to = as.Date('2020-04-08'), by = 'days')
state_5 = factor(rep(c('NJ', 'NY', "CA", "MI", "MA"), 17))
new_state = data.frame(date = rep(pred_date,each = 5), state = state_5)
pred2_state = predict(M6, newdata = new_state, interval = "prediction")
pred2_state
```

```
##         fit      lwr      upr
## 1  15.03511 14.29654 15.77368
## 2  21.84113 21.10256 22.57970
## 3  12.63156 11.89299 13.37013
## 4  12.26702 11.52845 13.00559
## 5  11.69002 10.95145 12.42859
## 6  15.82111 15.09355 16.54867
## 7  22.62713 21.89957 23.35469
## 8  13.41756 12.69000 14.14512
## 9  13.05302 12.32546 13.78058
## 10 12.47602 11.74846 13.20358
## 11 16.60711 15.88623 17.32799
## 12 23.41313 22.69225 24.13401
## 13 14.20356 13.48268 14.92443
## 14 13.83902 13.11814 14.55990
## 15 13.26202 12.54114 13.98290
## 16 17.39311 16.67447 18.11174
## 17 24.19913 23.48049 24.91776
## 18 14.98956 14.27092 15.70819
## 19 14.62502 13.90639 15.34365
## 20 14.04802 13.32938 14.76665
## 21 18.17911 17.45823 18.89998
## 22 24.98513 24.26425 25.70600
## 23 15.77556 15.05468 16.49643
## 24 15.41102 14.69014 16.13190
## 25 14.83402 14.11314 15.55490
## 26 18.96511 18.23755 19.69267
## 27 25.77113 25.04357 26.49869
## 28 16.56156 15.83400 17.28912
## 29 16.19702 15.46946 16.92458
## 30 15.62002 14.89246 16.34758
## 31 19.75111 19.01254 20.48968
## 32 26.55713 25.81856 27.29570
## 33 17.34756 16.60899 18.08613
## 34 16.98302 16.24445 17.72159
## 35 16.40602 15.66745 17.14459
## 36 20.53711 19.78340 21.29082
## 37 27.34313 26.58942 28.09684
## 38 18.13356 17.37985 18.88727
## 39 17.76902 17.01531 18.52273
## 40 17.19202 16.43831 17.94573
## 41 21.32311 20.55037 22.09585
## 42 28.12913 27.35639 28.90187
```

```
## 43 18.91956 18.14682 19.69230
## 44 18.55502 17.78228 19.32776
## 45 17.97802 17.20528 18.75076
## 46 22.10911 21.31372 22.90449
## 47 28.91513 28.11974 29.71051
## 48 19.70556 18.91017 20.50094
## 49 19.34102 18.54564 20.13640
## 50 18.76402 17.96863 19.55940
## 51 22.89511 22.07377 23.71644
## 52 29.70113 28.87979 30.52246
## 53 20.49156 19.67022 21.31289
## 54 20.12702 19.30568 20.94835
## 55 19.55002 18.72868 20.37135
## 56 23.68111 22.83081 24.53141
## 57 30.48713 29.63683 31.33743
## 58 21.27756 20.42726 22.12786
## 59 20.91302 20.06272 21.76332
## 60 20.33602 19.48572 21.18632
## 61 24.46711 23.58513 25.34908
## 62 31.27313 30.39115 32.15510
## 63 22.06356 21.18158 22.94553
## 64 21.69902 20.81704 22.58099
## 65 21.12202 20.24004 22.00399
## 66 25.25311 24.33702 26.16919
## 67 32.05913 31.14304 32.97521
## 68 22.84955 21.93347 23.76564
## 69 22.48502 21.56893 23.40110
## 70 21.90802 20.99193 22.82410
## 71 26.03911 25.08674 26.99147
## 72 32.84513 31.89276 33.79749
## 73 23.63555 22.68319 24.58791
## 74 23.27102 22.31866 24.22338
## 75 22.69402 21.74166 23.64638
## 76 26.82510 25.83454 27.81567
## 77 33.63112 32.64056 34.62169
## 78 24.42155 23.43099 25.41212
## 79 24.05702 23.06645 25.04759
## 80 23.48001 22.48945 24.47058
## 81 27.61110 26.58061 28.64160
## 82 34.41712 33.38663 35.44762
## 83 25.20755 24.17706 26.23805
## 84 24.84302 23.81252 25.87351
## 85 24.26601 23.23552 25.29651
```

```r
subset1 = covid19_state$state %in% c('NJ', 'NY', "CA", "MI", "MA")
subset2 = covid19_state$date>="2020-03-23" & covid19_state$date <= "2020-04-08"
test2_state= covid19_state[subset1 & subset2,] %>% dplyr::select('positive','state', 'date')
test2_state = merge(cbind(new_state, pred2_state), test2_state, by=c("date","state"))
test2_state = test2_state[order(test2_state$state),]

#5. Calculate prediction error of model 4 (M6) for each state
##5.1 MSPE
test2_state$pred_error = (test2_state$fit - test2_state$positive)^2
M6_pred_error_CA = mean(test2_state$pred_error[test2_state$state == "CA"])
```

```r
M6_pred_error_NY = mean(test2_state$pred_error[test2_state$state == "NY"])
M6_pred_error_NJ = mean(test2_state$pred_error[test2_state$state == "NJ"])
M6_pred_error_MI = mean(test2_state$pred_error[test2_state$state == "MI"])
M6_pred_error_MA = mean(test2_state$pred_error[test2_state$state == "MA"])
M6_pred_error_CA
```

```
## [1] 89314055
```

```r
M6_pred_error_NY
```

```
## [1] 7905271244
```

```r
M6_pred_error_NJ
```

```
## [1] 682594229
```

```r
M6_pred_error_MI
```

```
## [1] 120863850
```

```r
M6_pred_error_MA
```

```
## [1] 81348089
```

```r
#5.2 Ratio of root MSPE to mean cumulative positive cases in each state
ratio_CA = sqrt(M6_pred_error_CA) / mean(test2_state$positive[test2_state$state == "CA"])
ratio_MI = sqrt(M6_pred_error_MI) / mean(test2_state$positive[test2_state$state == "MI"])
ratio_MA = sqrt(M6_pred_error_MA) / mean(test2_state$positive[test2_state$state == "MA"])
ratio_NY = sqrt(M6_pred_error_NY) / mean(test2_state$positive[test2_state$state == "NY"])
ratio_NJ = sqrt(M6_pred_error_NJ) / mean(test2_state$positive[test2_state$state == "NJ"])
ratio_CA
```

```
## [1] 1.164003
```

```r
ratio_MA
```

```
## [1] 1.19614
```

```r
ratio_MI
```

```
## [1] 1.209432
```

```r
ratio_NY
```

```
## [1] 1.122183
```

```r
ratio_NJ
```

```
## [1] 1.204391
```

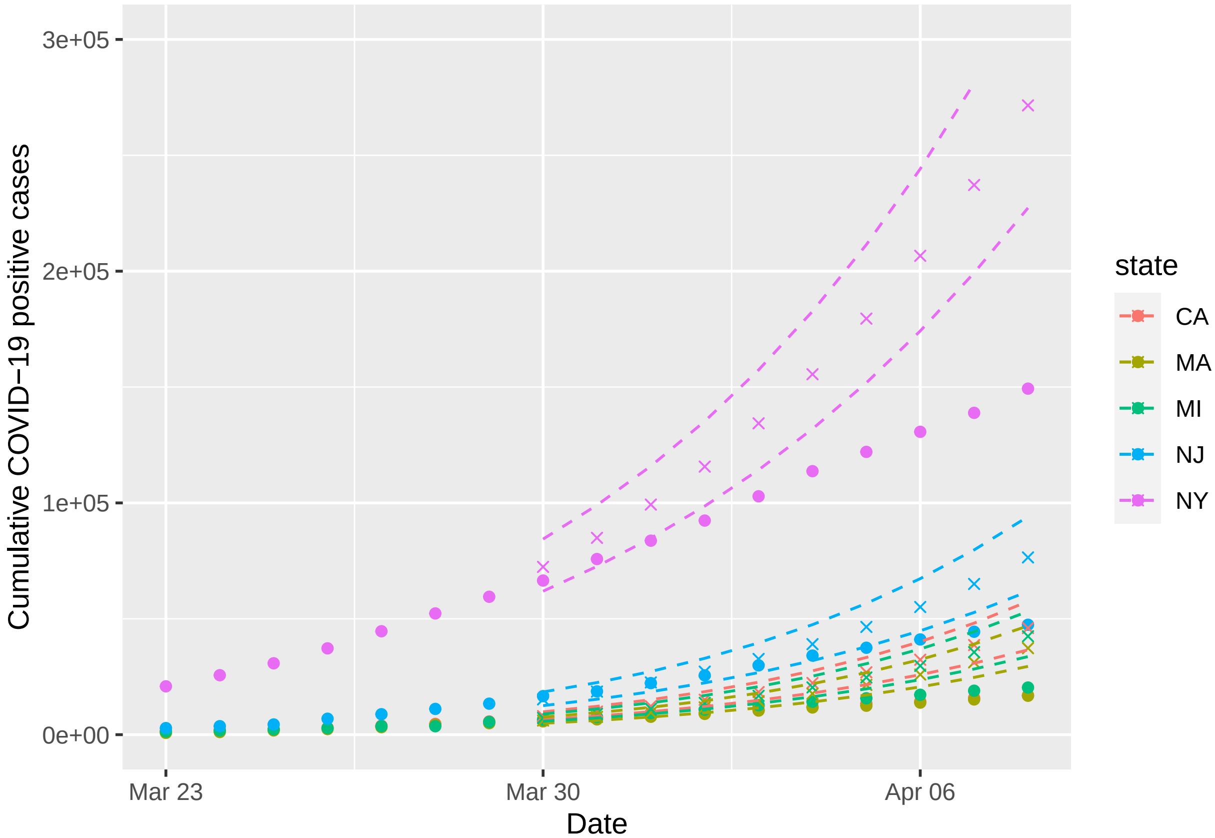
```r
#6. Visualiztion of model 4 prediction by ggplot2
lp1 <- ggplot(data=test1_state, aes(x = date, y = positive, group= state, colour= state))+ geom_point()
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```r
# Change the legend
lp1 + scale_shape_discrete(name  ="state",
                          breaks=c("NJ", "NY", "CA", "MI", "MA"),
                          labels=c("NJ", "NY", "CA", "MI", "MA"))
```

```
## Warning: Removed 35 rows containing missing values (geom_point).
```

```
## Warning: Removed 35 row(s) containing missing values (geom_path).

## Warning: Removed 36 row(s) containing missing values (geom_path).
```

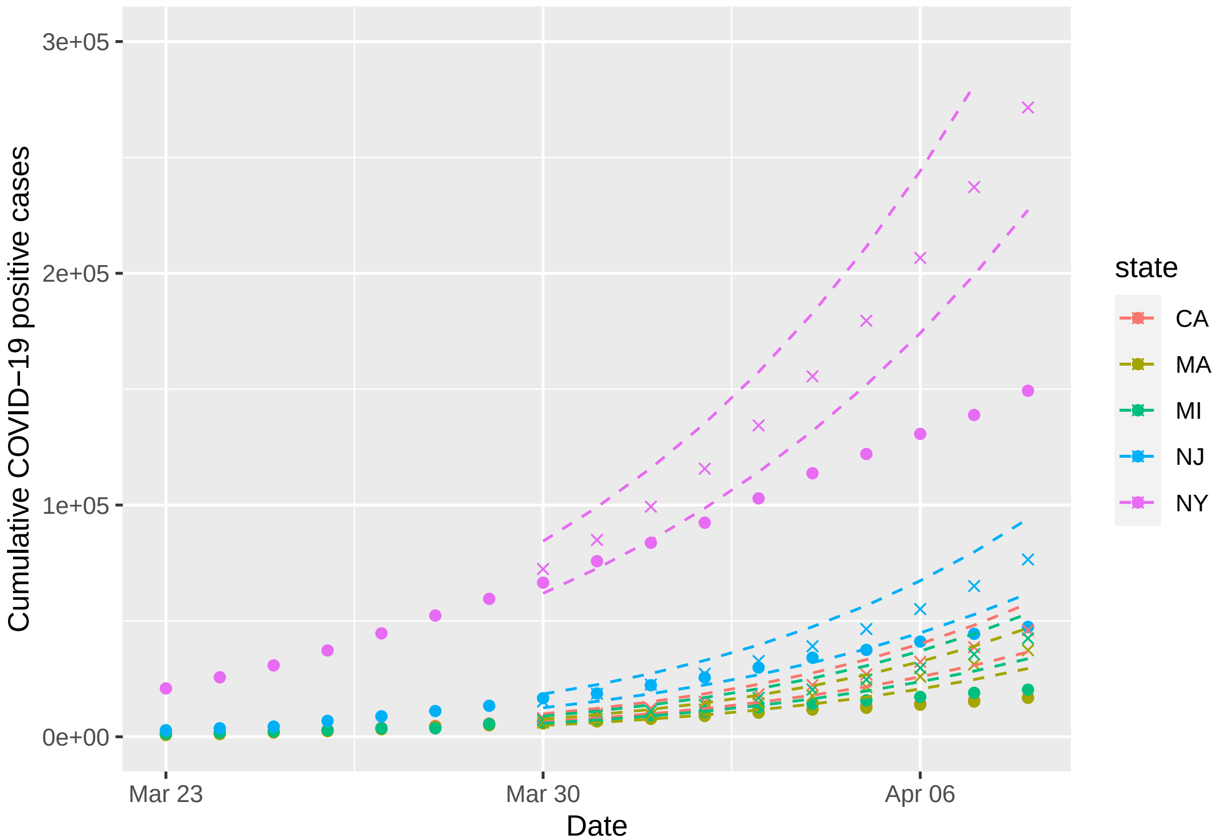



```
lp2 <- ggplot(data=test1_state, aes(x = date, y = positive, group= state, colour= state))+ geom_point()
```

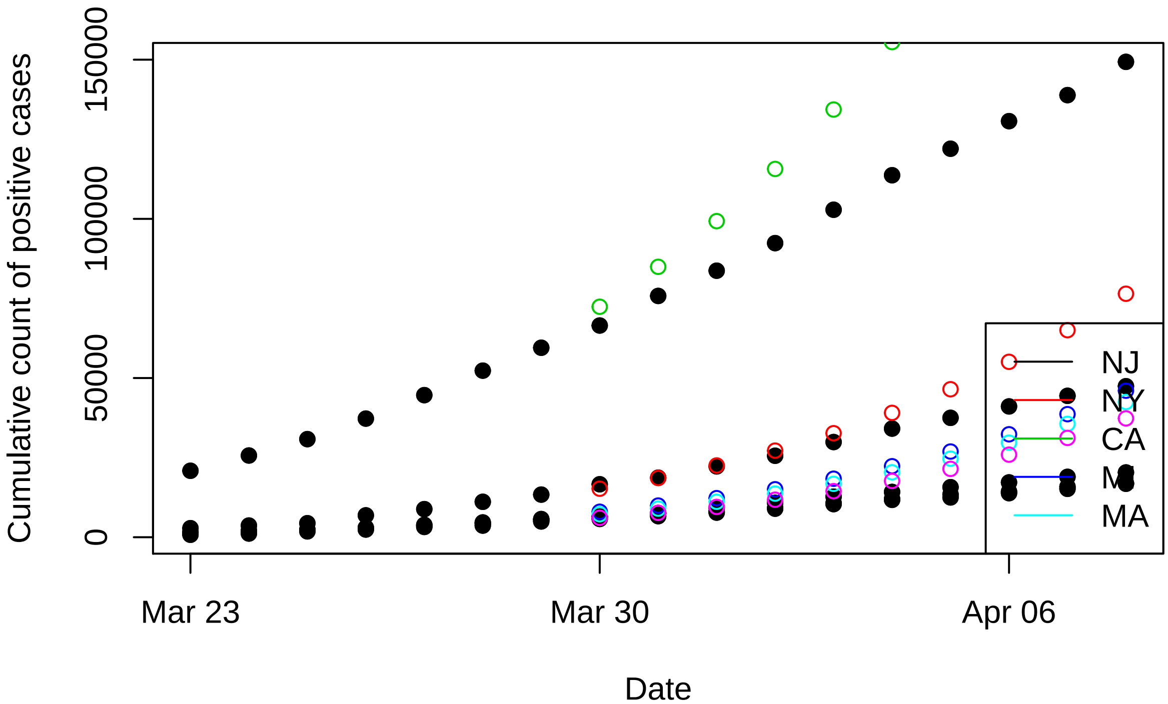
```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```
# Change the legend
lp2 + scale_shape_discrete(name  ="state",
                           breaks=c("NJ", "NY", "CA", "MI", "MA"),
                           labels=c("NJ", "NY", "CA", "MI", "MA"))
```

```
## Warning: Removed 35 rows containing missing values (geom_point).

## Warning: Removed 35 row(s) containing missing values (geom_path).

## Warning: Removed 36 row(s) containing missing values (geom_path).
```




```
##5.1 Visualiztion of model 4 prediction by plot
xmax = max(test1_state$date)
xmin = min(test1_state$date)
ymax = max(test1_state$fit)
ymin = min(test1_state$positive)

plot(test1_state$date, test1_state$positive, pch = 19, xlab = "Date", ylab = "Cumulative count of positi
lines(test1_state$date, test1_state$positive, type = "p")
count = 1
for(current_state in c("NJ", "NY", "CA", "MI", "MA")){
  count = count + 1
  lines(test1_state$date[test1_state$state == current_state],
        test1_state$fit[test1_state$state == current_state], type = "p", col = count)
}
legend('bottomright', legend = c("NJ", "NY", "CA", "MI", "MA"), col = 1:5, lty = rep(1,5))
```


Cumulative count of positive cases
0
50000
100000
150000
Mar 23
Mar 30
Apr 06
Date
NJ
NY
CA
MI
MA