R Notebook

Predicting number of Covid19 deaths using Time Series Analysis (ARIMA MODEL)

Importing required libraries

```
suppressMessages(library(tidyverse))
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
## Warning: package 'dplyr' was built under R version 4.2.1
```

```
## Warning: package 'stringr' was built under R version 4.2.1
```

```
## Warning: package 'forcats' was built under R version 4.2.2
```

```
suppressMessages(library(ggplot2))
suppressMessages(library(zoo))
```

```
## Warning: package 'zoo' was built under R version 4.2.2
```

```
suppressMessages(library(aTSA))
suppressMessages(library(tseries))
```

```
## Warning: package 'tseries' was built under R version 4.2.2
```

```
suppressMessages(library(forecast))
```

```
## Warning: package 'forecast' was built under R version 4.2.2
```

```
suppressMessages(library(lubridate))
```

```
## Warning: package 'lubridate' was built under R version 4.2.2
```

Load the dataset

```
covid_data <- read.csv("owid-covid-data.csv")
colnames(covid_data)
```

```
##  [1] "iso_code"                        "continent"
##  [3] "location"                        "date"
##  [5] "total_cases"                     "new_cases"
##  [7] "total_deaths"                    "new_deaths"
##  [9] "total_cases_per_million"         "new_cases_per_million"
## [11] "total_deaths_per_million"        "new_deaths_per_million"
## [13] "new_tests"                       "total_tests"
## [15] "total_tests_per_thousand"        "new_tests_per_thousand"
## [17] "new_tests_smoothed"              "new_tests_smoothed_per_thousand"
## [19] "tests_units"                     "stringency_index"
## [21] "population"                      "population_density"
## [23] "median_age"                      "aged_65_older"
## [25] "aged_70_older"                   "gdp_per_capita"
## [27] "extreme_poverty"                 "cardiovasc_death_rate"
## [29] "diabetes_prevalence"             "female_smokers"
## [31] "male_smokers"                    "handwashing_facilities"
## [33] "hospital_beds_per_thousand"      "life_expectancy"
```

```
str(covid_data)
```

```
## 'data.frame':    34033 obs. of  34 variables:
##  $ iso_code                      : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ continent                     : chr  "Asia" "Asia" "Asia" "Asia" ...
##  $ location                      : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghan
istan" ...
##  $ date                          : chr  "2019-12-31" "2020-01-01" "2020-01-02" "2020-01-0
3" ...
##  $ total_cases                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths                    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_cases_per_million       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases_per_million         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_deaths_per_million      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_deaths_per_million        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ new_tests                     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests                   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests_per_thousand      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_per_thousand        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed_per_thousand: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tests_units                   : chr  "" "" "" "" ...
##  $ stringency_index              : num  NA 0 0 0 0 0 0 0 0 0 ...
##  $ population                    : num  38928341 38928341 38928341 38928341 38928341 ...
##  $ population_density            : num  54.4 54.4 54.4 54.4 54.4 ...
##  $ median_age                    : num  18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6
...
##  $ aged_65_older                 : num  2.58 2.58 2.58 2.58 2.58 ...
##  $ aged_70_older                 : num  1.34 1.34 1.34 1.34 1.34 ...
##  $ gdp_per_capita                : num  1804 1804 1804 1804 1804 ...
##  $ extreme_poverty               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ cardiovasc_death_rate         : num  597 597 597 597 597 ...
##  $ diabetes_prevalence           : num  9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59
...
##  $ female_smokers                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ male_smokers                  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ handwashing_facilities        : num  37.7 37.7 37.7 37.7 37.7 ...
##  $ hospital_beds_per_thousand    : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ life_expectancy               : num  64.8 64.8 64.8 64.8 64.8 ...
```

We change the date column into a date type

```
covid_data$date <- as.Date(covid_data$date)
```

We pick the US rows only.

```
data_US <- covid_data[covid_data$location == "United States",c(3,4,7,8)]

summary(data_US)
```

```
##    location              date             total_deaths      new_deaths
##  Length:215         Min.   :2019-12-31   Min.   :     0   Min.   :   0.0
##  Class :character   1st Qu.:2020-02-22   1st Qu.:     0   1st Qu.:   0.0
##  Mode  :character   Median :2020-04-16   Median : 30985   Median : 500.0
##                     Mean   :2020-04-16   Mean   : 53419   Mean   : 713.1
##                     3rd Qu.:2020-06-08   3rd Qu.:110761   3rd Qu.:1167.0
##                     Max.   :2020-08-01   Max.   :153314   Max.   :4928.0
```

Checking for null or na rows

```
sum(is.null(data_US$date))
```

```
## [1] 0
```

```
sum(is.na(data_US$date))
```

```
## [1] 0
```

```
sum(is.na(data_US$total_deaths))
```

```
## [1] 0
```

```
sum(is.null(data_US$total_deaths))
```

```
## [1] 0
```

```
sum(is.na(data_US$new_deaths))
```

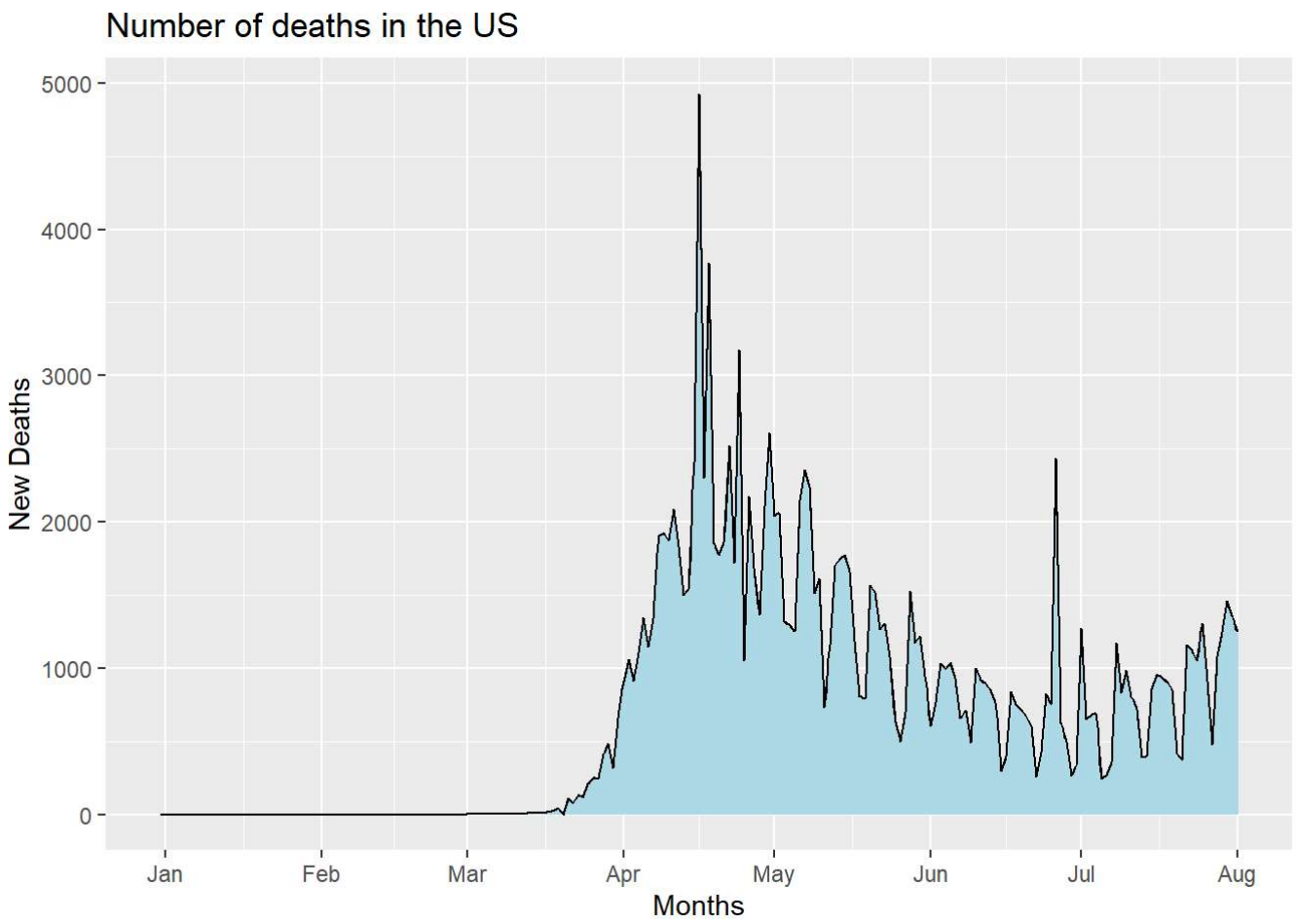
```
## [1] 0
```

```
sum(is.na(data_US$new_deaths))
```

```
## [1] 0
```

Plotting the number of deaths in the US

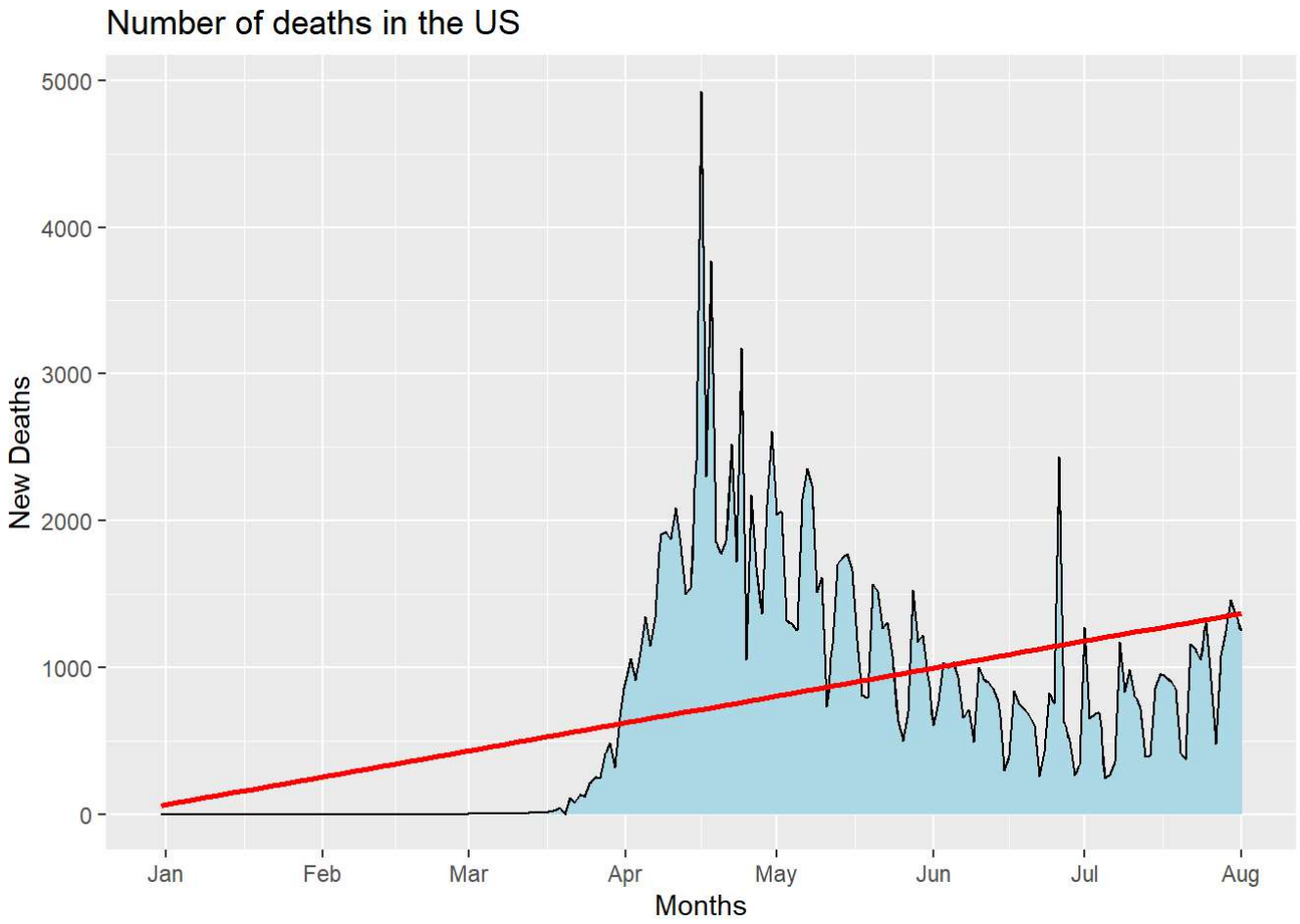
```
p <- ggplot(data_US, aes(x=date, y=new_deaths)) +
  geom_line() + scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  xlab("Months") + ylab("New Deaths")+ggtitle("Number of deaths in the US")+geom_area(fill="l
ightblue", color="black")
p
```



Plotting the mean to check if our data is stationary or not

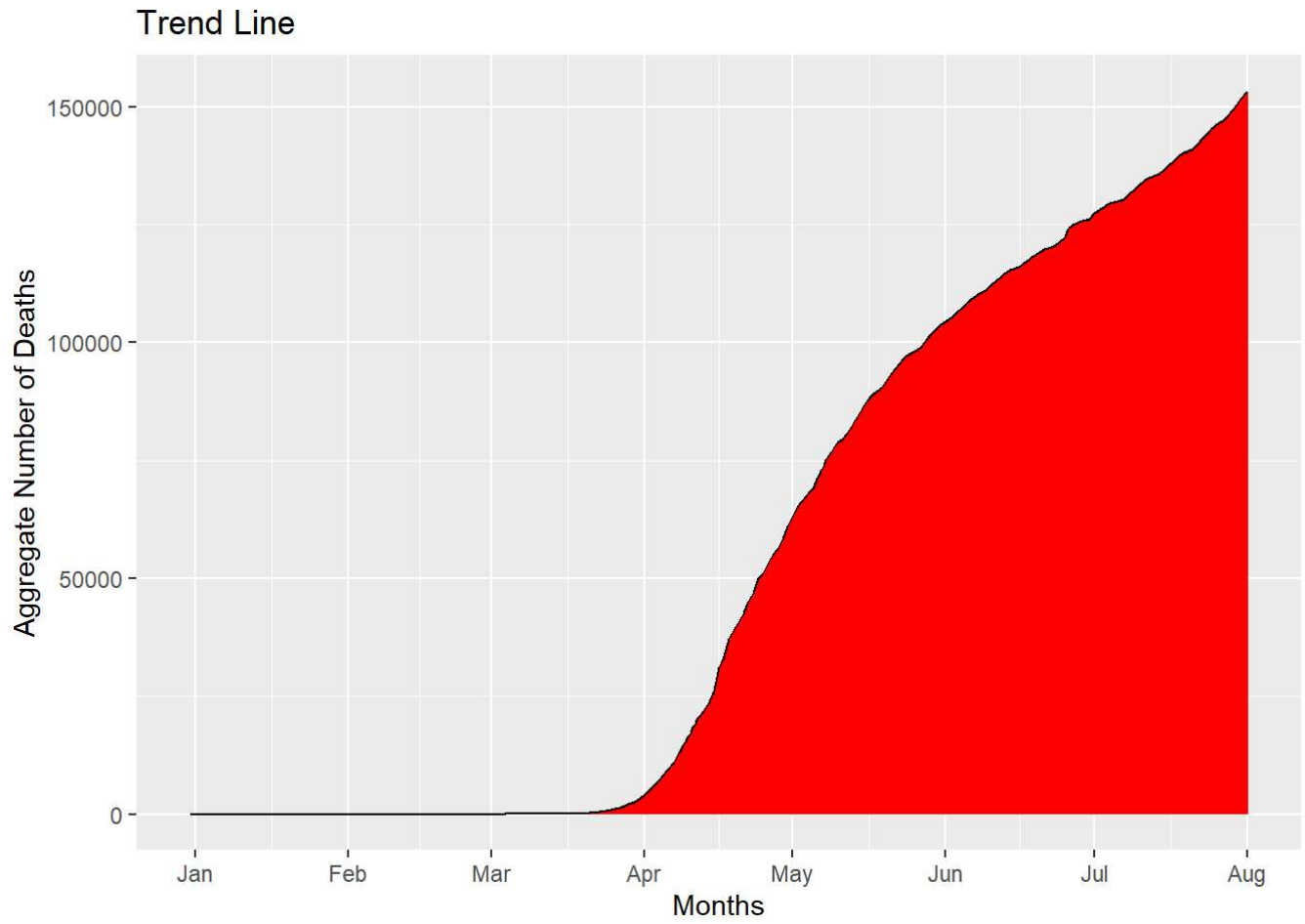
```
p + geom_smooth(method = lm, col = "red", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



General trend of our data

```
p1 <- ggplot(data_US, aes(x=date, y=total_deaths)) +
  geom_line() + scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  xlab("Months") + ylab("Aggregate Number of Deaths")+ggtitle("Trend Line")+geom_area(fill="r
ed", color="black")
p1
```



Here we can see the aggregated numbers of deaths

```
data_US$total_deaths
```

```
##   [1]      0      0      0      0      0      0      0      0      0      0
##  [11]      0      0      0      0      0      0      0      0      0      0
##  [21]      0      0      0      0      0      0      0      0      0      0
##  [31]      0      0      0      0      0      0      0      0      0      0
##  [41]      0      0      0      0      0      0      0      0      0      0
##  [51]      0      0      0      0      0      0      0      0      0      0
##  [61]      0      1      2      6      9     11     12     14     17     21
##  [71]     26     28     30     40     47     57     69     85    108    150
##  [81]    150    260    340    471    590    801   1050   1296   1707   2191
##  [91]   2509   3170   4079   5138   6053   7157   8501   9647  10989  12895
## [101]  14817  16690  18777  20608  22108  23649  26057  30985  33284  37054
## [111]  38910  40682  42539  45063  46784  49963  51017  53189  54876  56245
## [121]  58355  60966  63006  65068  66385  67682  68934  71078  73431  75670
## [131]  77180  78794  79528  80684  82387  84133  85906  87568  88754  89562
## [141]  90353  91921  93439  94702  96007  97087  97720  98220  98916 100442
## [151] 101617 102836 103781 104383 105147 106181 107175 108211 109143 109802
## [161] 110514 111007 112006 112924 113820 114669 115436 115732 116127 116963
## [171] 117717 118434 119112 119719 119975 120402 121228 121979 124416 125039
## [181] 125539 125804 126140 127410 128062 128740 129434 129676 129947 130306
## [191] 131480 132309 133291 134097 134814 135205 135605 136466 137419 138358
## [201] 139266 140119 140534 140906 142066 143190 144242 145546 146460 146935
## [211] 148011 149256 150713 152070 153314
```

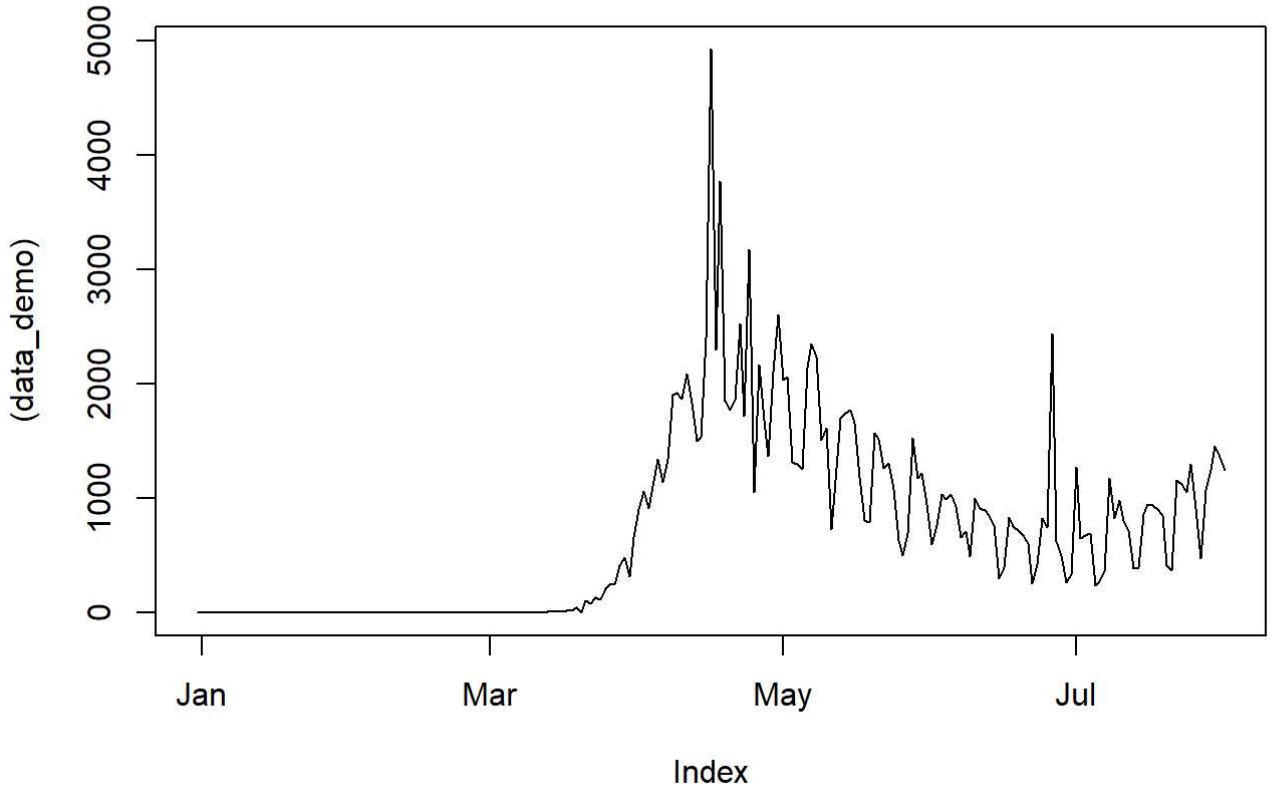
Change the data to zoo type providing the sequence as dates ranging from 2019/12/31 to 2020/08/01

```
data_US_Analysis <- data_US[,c(2,4)]
data_demo <- zoo(data_US_Analysis$new_deaths, seq(from = as.Date("2019-12-31"), to = as.Date
("2020-08-01"), by = 1))

summary(data_demo)
```

```
##       Index              data_demo
##  Min.   :2019-12-31   Min.   :   0.0
##  1st Qu.:2020-02-22   1st Qu.:   0.0
##  Median :2020-04-16   Median : 500.0
##  Mean   :2020-04-16   Mean   : 713.1
##  3rd Qu.:2020-06-08   3rd Qu.:1167.0
##  Max.   :2020-08-01   Max.   :4928.0
```

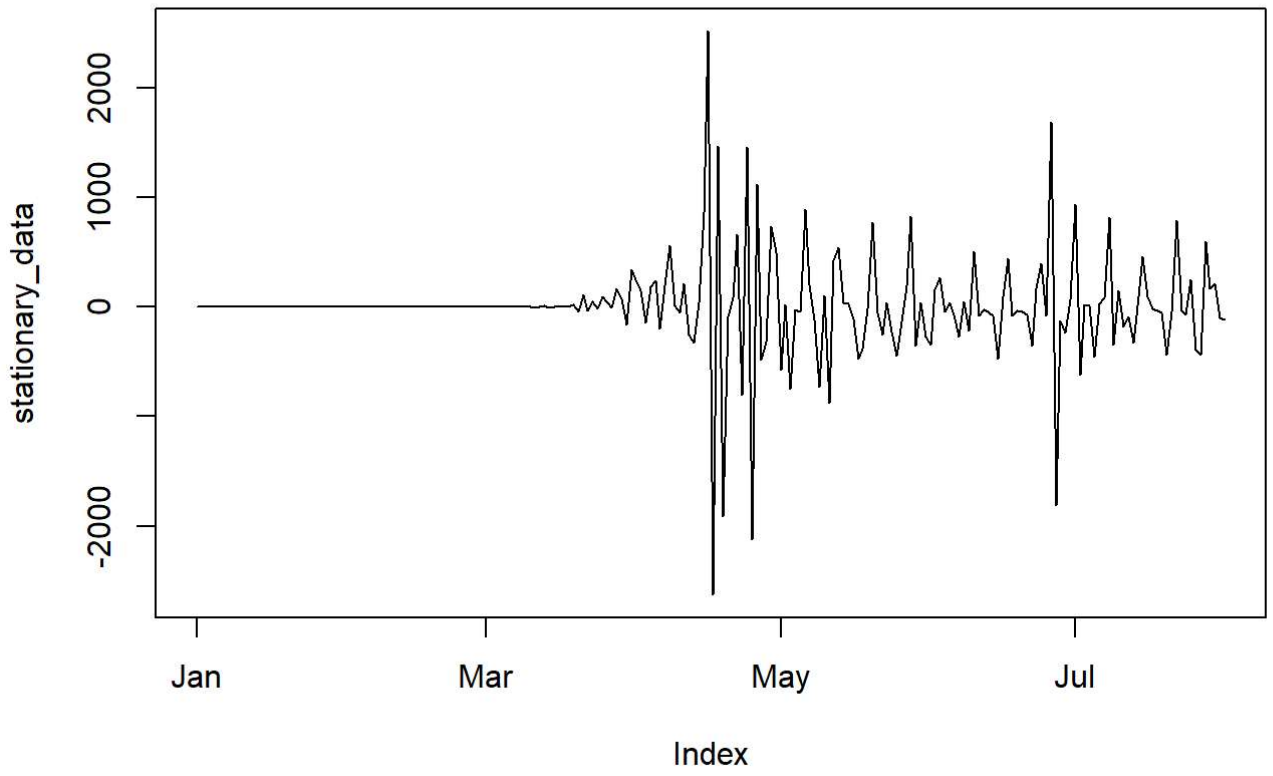
Plotting the dataframe

```
plot((data_demo))
```



Converting the data to stationary by using diff function

```
stationary_data <- diff(data_demo)
plot(stationary_data)
```



To check if it's stationary we conduct a quantitative test. We use the Augmented Dickey-Fuller Test.

H_0 = The null hypothesis for this test is that there is a unit root.

H_A = The alternative hypothesis is that the time series is stationary (or trend-stationary).

We select a significance level of 0.05 and since our p-value is 0.01 and smaller then 0.05, we come to the conclusion to reject the null hypothesis.

```
adf.test(as.matrix(stationary_data))
```

```
## Warning in adf.test(as.matrix(stationary_data)): p-value smaller than printed p-
## value
```

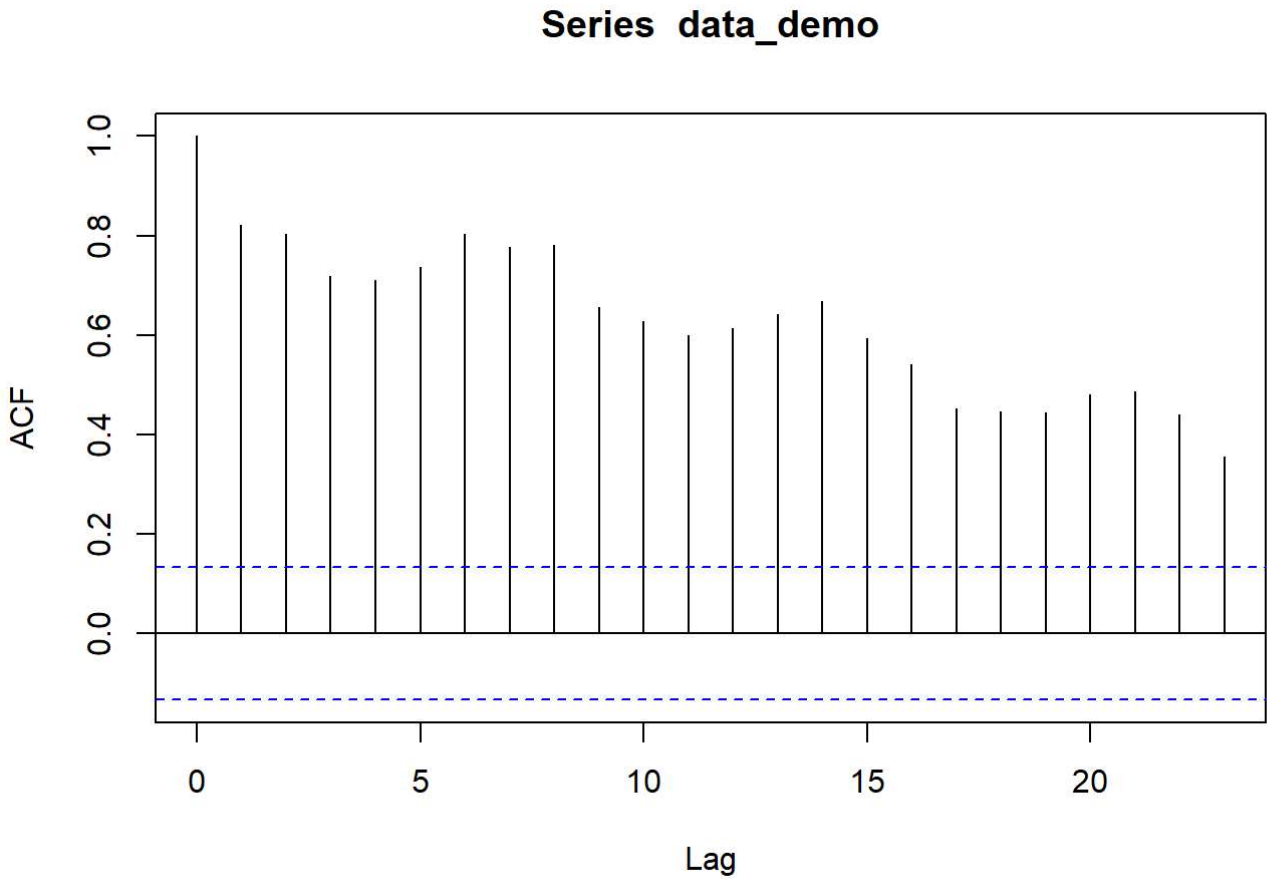
```
##
##  Augmented Dickey-Fuller Test
##
## data:  as.matrix(stationary_data)
## Dickey-Fuller = -10.478, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Modeling

We use the Auto Correlation Graph when our data isn't stationary.

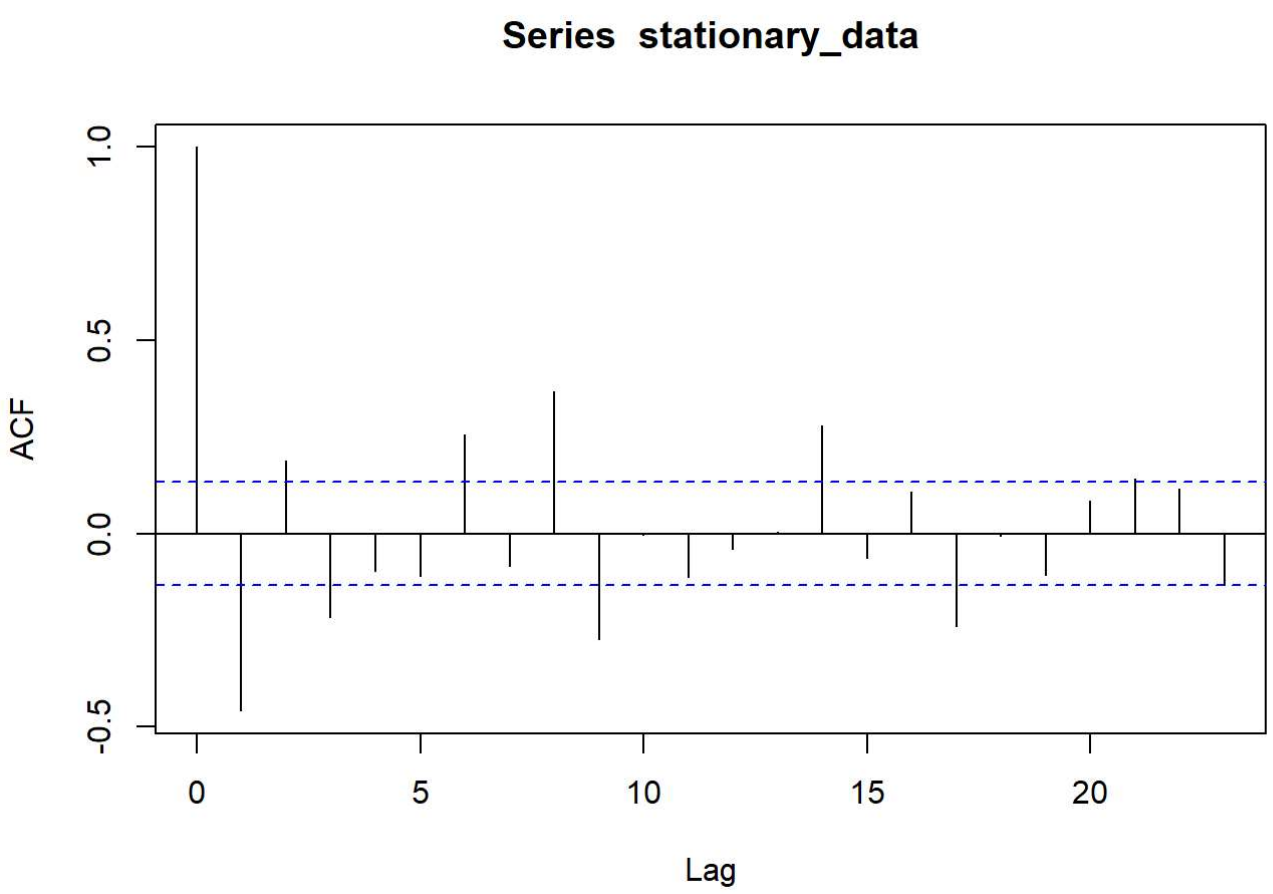
Here we see that our values are all exceeding the blue line. The goal is to have the values under the blue line and they should be inverted as well.

```
acf(data_demo)
```

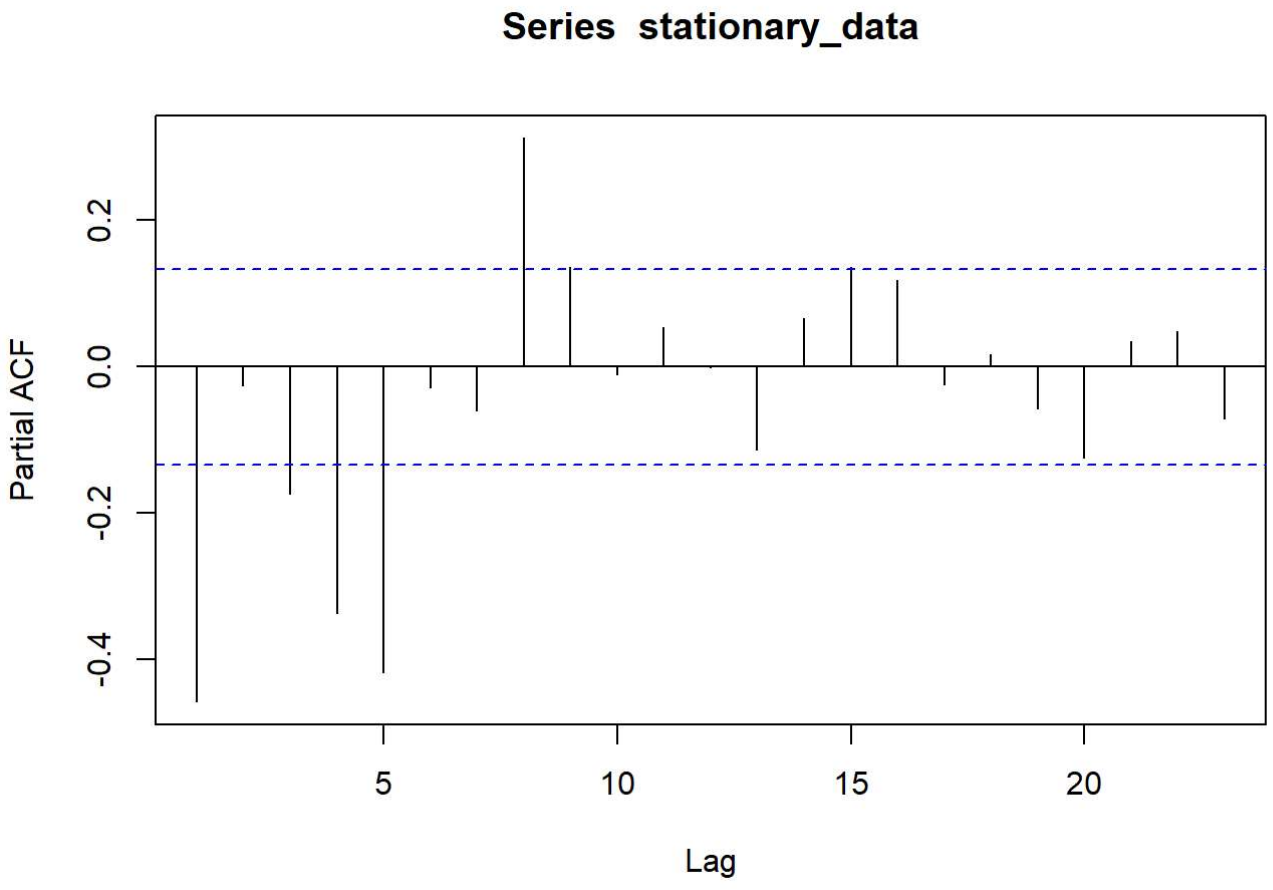


To select the p and q values we select the number before the first inverted line

```
acf(stationary_data)
```




```
pacf(stationary_data)
```



arima has a auto.arima function which gives us the ideal arima model based on our data.

```
arima_funct <- auto.arima(stationary_data)
arima_funct
```

```
## Series: stationary_data
## ARIMA(3,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2
##       0.6082  -0.1778  -0.6058  -1.2614  0.8282
## s.e.  0.0599   0.0686   0.0563   0.0460  0.0476
##
## sigma^2 = 126057:  log likelihood = -1558.96
## AIC=3129.92   AICc=3130.32   BIC=3150.11
```

lets use the auto.arima function to forecast 3 weeks

```
forecast1 <- forecast(arima_funct, h=21)
additional_deaths <- round(sum(forecast1$upper[,2]))
# [1] 18589

total_number_of_deaths <- round(sum(data_US_Analysis$new_deaths)+additional_deaths)
# [1] 171903

additional_deaths
```

```
## [1] 18589
```

```
total_number_of_deaths
```

```
## [1] 171903
```

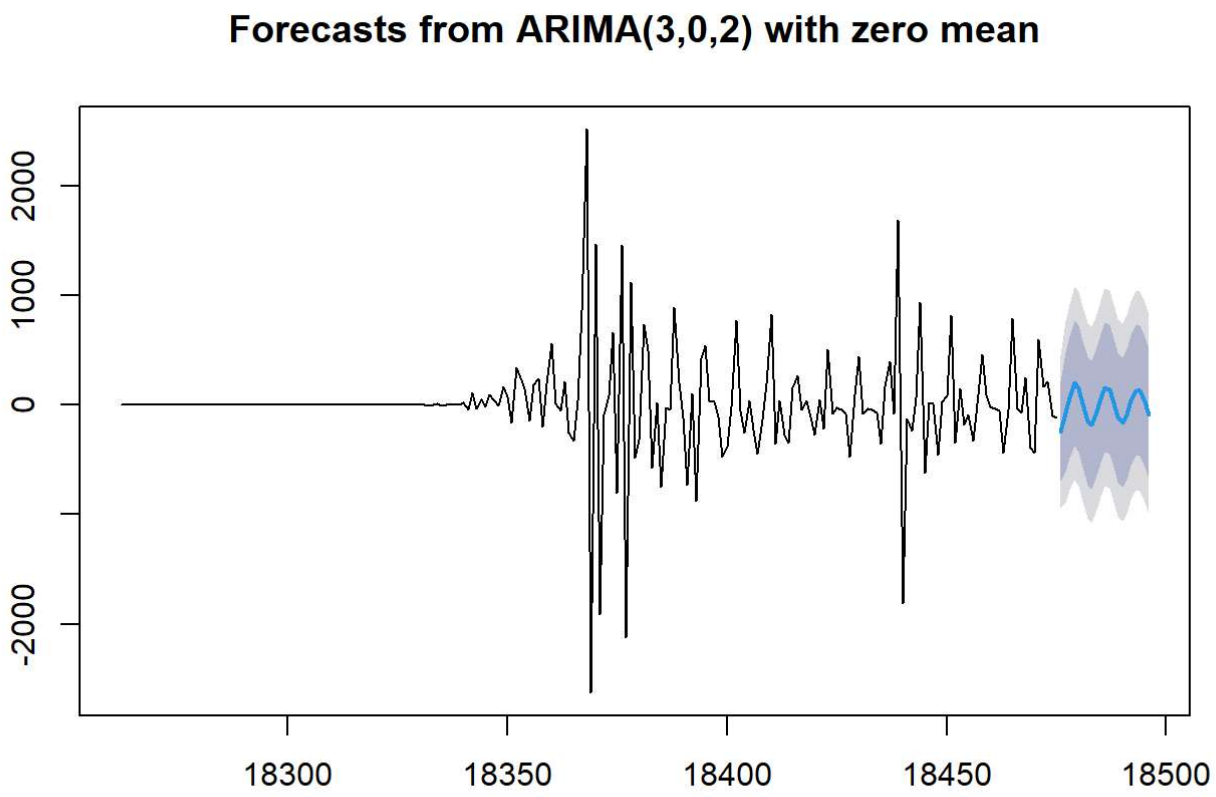
```
forecast1
```

```
##       Point Forecast     Lo 80    Hi 80      Lo 95     Hi 95
## 18476     -240.63831 -695.6474 214.3708  -936.5147  455.2381
## 18477      -67.25592 -610.7369 476.2251  -898.4384  763.9265
## 18478       70.34770 -485.1972 625.8926  -779.2849  919.9803
## 18479      200.53359 -375.6366 776.7038  -680.6427 1081.7099
## 18480      150.19919 -429.8157 730.2140  -736.8570 1037.2553
## 18481       13.06974 -566.9487 593.0882  -873.9919  900.1314
## 18482     -140.25233 -725.6887 445.1840 -1035.6000  755.0953
## 18483     -178.62119 -764.1164 406.8740 -1074.0588  716.8164
## 18484      -91.61313 -677.1609 493.9346  -987.1311  803.9048
## 18485       61.01629 -527.0520 649.0846  -838.3566  960.3892
## 18486      161.61703 -427.5961 750.8301  -739.5067 1062.7407
## 18487      142.94659 -446.3195 732.2127  -758.2582 1044.1513
## 18488       21.23253 -569.0528 611.5179  -881.5310  923.9961
## 18489     -110.42078 -702.3988 481.5573 -1015.7731  794.9315
## 18490     -157.53559 -750.0768 435.0057 -1063.7492  748.6781
## 18491      -89.03929 -681.7287 503.6501  -995.4796  817.4010
## 18492       40.75887 -553.2624 634.7801  -867.7183  949.2360
## 18493      136.06443 -459.1375 731.2663  -774.2183 1046.3472
## 18494      129.44873 -465.8029 724.7003  -780.9100 1039.8075
## 18495       29.84002 -565.9823 625.6623  -881.3915  941.0715
## 18496      -87.30467 -684.4631 509.8538 -1000.5797  825.9704
```

```
forecast1$upper[, 2]
```

```
## Time Series:
## Start = 18476
## End = 18496
## Frequency = 1
##  [1]  455.2381  763.9265  919.9803 1081.7099 1037.2553  900.1314  755.0953
##  [8]  716.8164  803.9048  960.3892 1062.7407 1044.1513  923.9961  794.9315
## [15]  748.6781  817.4010  949.2360 1046.3472 1039.8075  941.0715  825.9704
```

```
plot(forecast1)
```



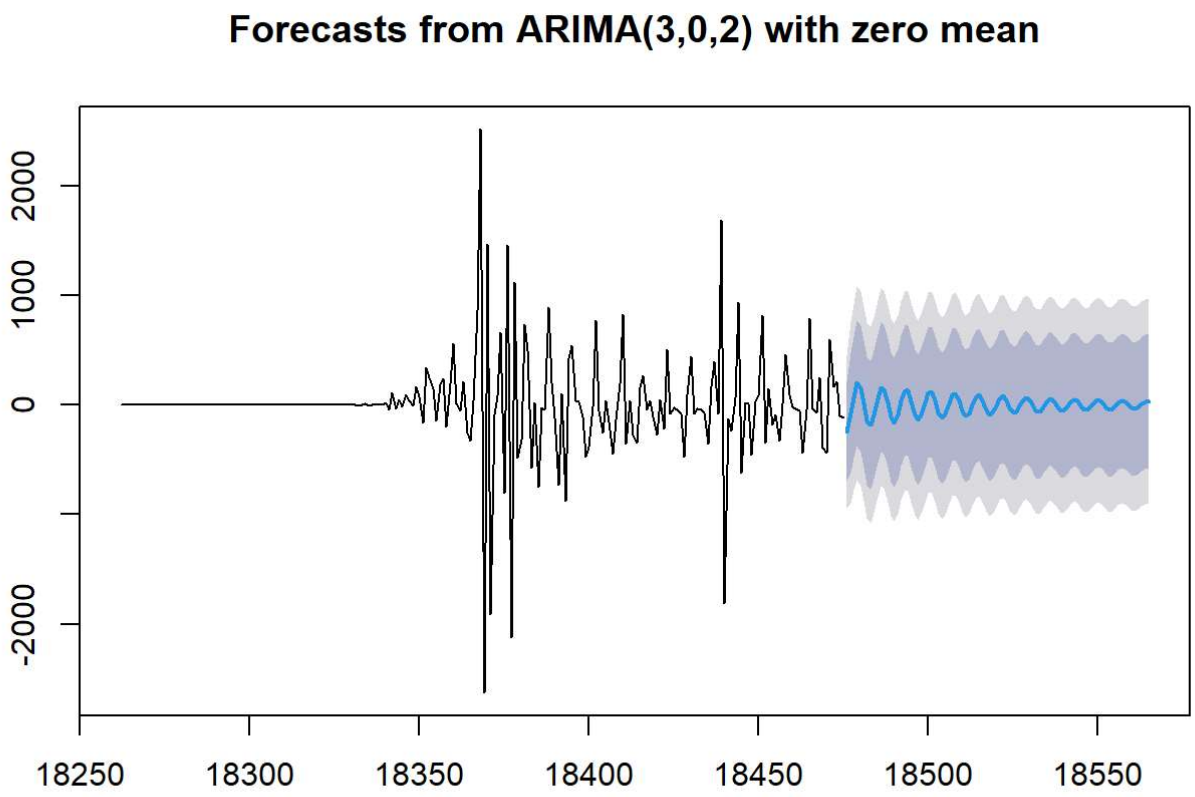
lets use the auto.arima function to forecast 3 months

```
forecast2 <- forecast(object = arima_funct, h = 90)
additional_deaths2 <- round(sum(forecast2$upper[,2]),0)
# [1] 82653

total_number_of_deaths2 <- round(sum(data_US_Analysis$new_deaths)+additional_deaths2,0)
# [1] 235967

# additional_deaths2
# total_number_of_deaths2
forecast2
```

```
##       Point Forecast     Lo 80    Hi 80      Lo 95     Hi 95
## 18476    -240.638307 -695.6474 214.3708  -936.5147  455.2381
## 18477     -67.255919 -610.7369 476.2251  -898.4384  763.9265
## 18478      70.347696 -485.1972 625.8926  -779.2849  919.9803
## 18479     200.533588 -375.6366 776.7038  -680.6427 1081.7099
## 18480     150.199189 -429.8157 730.2140  -736.8570 1037.2553
## 18481      13.069740 -566.9487 593.0882  -873.9919  900.1314
## 18482    -140.252327 -725.6887 445.1840 -1035.6000  755.0953
## 18483    -178.621186 -764.1164 406.8740 -1074.0588  716.8164
## 18484     -91.613131 -677.1609 493.9346  -987.1311  803.9048
## 18485      61.016293 -527.0520 649.0846  -838.3566  960.3892
## 18486     161.617032 -427.5961 750.8301  -739.5067 1062.7407
## 18487     142.946586 -446.3195 732.2127  -758.2582 1044.1513
## 18488      21.232527 -569.0528 611.5179  -881.5310  923.9961
## 18489    -110.420784 -702.3988 481.5573 -1015.7731  794.9315
## 18490    -157.535588 -750.0768 435.0057 -1063.7492  748.6781
## 18491     -89.039295 -681.7287 503.6501  -995.4796  817.4010
## 18492      40.758871 -553.2624 634.7801  -867.7183  949.2360
## 18493     136.064435 -459.1375 731.2663  -774.2183 1046.3472
## 18494     129.448734 -465.8029 724.7003  -780.9100 1039.8075
## 18495      29.840017 -565.9823 625.6623  -881.3915  941.0715
## 18496     -87.304672 -684.4631 509.8538 -1000.5797  825.9704
## 18497    -136.829681 -734.4825 460.8232 -1050.8609  777.2015
## 18498     -85.771624 -683.4796 511.9363  -999.8870  828.3438
## 18499      25.059448 -573.5834 623.7023  -890.4858  940.6047
## 18500     113.390613 -486.1958 712.9770  -803.5976 1030.3788
## 18501     116.470836 -483.1922 716.1339  -800.6346 1033.5763
## 18502      35.490384 -564.5199 635.5007  -882.1461  953.1269
## 18503     -67.823597 -668.8243 533.1771  -986.9748  851.3276
## 18504    -118.123719 -719.5702 483.3228 -1037.9568  801.7094
## 18505     -81.282343 -682.7451 520.1804 -1001.1402  838.5755
## 18506      12.660900 -589.4393 614.7611  -908.1718  933.4936
## 18507      93.718765 -509.1327 696.5703  -828.2630 1015.7006
## 18508     103.991621 -498.9548 706.9380  -818.1353 1026.1186
## 18509      38.910403 -564.2384 642.0592  -883.5261  961.3469
## 18510     -51.606398 -655.4828 552.2700  -975.1556  871.9428
## 18511    -101.308321 -705.5754 502.9587 -1025.4550  822.8384
## 18512     -76.011238 -680.2802 528.2577 -1000.1609  848.1384
## 18513       3.051436 -601.6454 607.7483  -921.7525  927.8554
## 18514      76.749564 -528.5375 682.0366  -848.9571 1002.4562
## 18515      92.186490 -513.2058 697.5787  -833.6811 1018.0540
## 18516      40.570045 -564.9346 646.0747  -885.4694  966.6095
## 18517     -38.217091 -644.2498 567.8156  -965.0641  888.6299
## 18518     -86.308163 -692.6756 520.0593 -1013.6671  841.0508
## 18519     -70.274757 -676.6426 536.0931  -997.6343  857.0848
## 18520      -4.238937 -610.8886 602.4107  -932.0295  923.5516
## 18521      62.207804 -544.8996 669.3152  -866.2828  990.6984
## 18522      81.163364 -526.0523 688.3790  -847.4928 1009.8195
## 18523      40.868626 -566.4054 648.1426  -887.8768  969.6141
## 18524     -27.265274 -634.9177 580.3871  -956.5894  902.0589
## 18525     -73.022201 -680.9558 534.9114 -1002.7764  856.7320
## 18526     -64.322793 -672.2608 543.6153  -994.0838  865.4382
## 18527      -9.616689 -617.7363 598.5029  -939.6554  920.4220
## 18528      49.829493 -558.6407 658.2997  -880.7454  980.4044
## 18529      70.985360 -537.5904 679.5611  -859.7509 1001.7216
```

```
## 18530      40.137736 -568.4655 648.7409  -890.6406  970.9160
## 18531     -18.400566 -627.2712 590.4701  -949.5879  912.7868
## 18532     -61.334539 -670.4372 547.7682  -992.8767  870.2077
## 18533     -58.348057 -667.4609 550.7648  -989.9057  873.2096
## 18534     -13.431912 -622.6588 595.7949  -945.1640  918.3002
## 18535      39.365764 -570.1263 648.8578  -892.7719  971.5034
## 18536      61.680130 -547.9105 671.2708  -870.6084  993.9686
## 18537      38.650547 -570.9512 648.2523  -893.6548  970.9559
## 18538     -11.310969 -621.0990 598.4771  -943.9013  921.2794
## 18539     -51.120755 -661.0971 558.8556  -983.9991  881.7576
## 18540     -52.495858 -662.4875 557.4958  -985.3976  880.4059
## 18541     -15.984129 -626.0452 594.0769  -948.9920  917.0238
## 18542      30.584911 -579.6742 640.8441  -902.7259  963.8958
## 18543      53.248049 -557.1003 663.5964  -880.1992  986.6953
## 18544      36.629955 -573.7217 646.9816  -896.8223  970.0822
## 18545      -5.720609 -616.2000 604.7587  -939.3682  927.9270
## 18546     -42.252929 -652.8829 568.3770  -976.1309  891.6250
## 18547     -46.872494 -657.5215 563.7765  -980.7795  887.0345
## 18548     -17.527872 -628.2175 593.1618  -951.4971  916.4413
## 18549      23.273533 -587.5622 634.1092  -910.9191  957.4661
## 18550      45.668991 -565.2452 656.5832  -888.6436  979.9816
## 18551      34.255875 -576.6587 645.1704  -900.0573  968.5691
## 18552      -1.387197 -612.3878 609.6134  -935.8320  933.0576
## 18553     -34.603503 -645.7228 576.5158  -969.2299  900.0229
## 18554     -41.552471 -652.6930 569.5880  -976.2112  893.1063
## 18555     -18.277928 -629.4411 592.8852  -952.9713  916.4154
## 18556      17.236967 -594.0325 628.5065  -917.6190  952.0930
## 18557      38.907899 -572.4290 650.2448  -896.0512  973.8670
## 18558      31.671728 -579.6653 643.0087  -903.2875  966.6310
## 18559       1.900662 -609.4932 613.2945  -933.1455  936.9468
## 18560     -28.048174 -639.5343 583.4380  -963.2355  907.1392
## 18561     -36.584672 -648.0926 574.9233  -971.8054  898.6360
## 18562     -18.414202 -629.9340 593.1056  -953.6530  916.8246
## 18563      12.299183 -599.2970 623.8953  -923.0564  947.6548
## 18564      32.919320 -578.7336 644.5722  -902.5231  968.3617
## 18565      28.990168 -582.6636 640.6440  -906.4536  964.4339
```

```
plot(forecast2)
```



Below we are just adding the dates and make changes to the x axis for the projected graphs

```
delta <- (forecast1$lower[, 2] + forecast1$upper[, 2])/2

predicted_21 <- data.frame(date = seq(from = 18476, by = 1, length=21), new_deaths = delta)
original <- data.frame(date = as.numeric(data_US_Analysis$date), new_deaths = c(0, stationary
_data))

z <- rbind(original,predicted_21)
z$date <- as.Date(z$date)

z
```

date <date>	new_deaths <dbl>
2019-12-31	0.00000
2020-01-01	0.00000
2020-01-02	0.00000
2020-01-03	0.00000
2020-01-04	0.00000
2020-01-05	0.00000

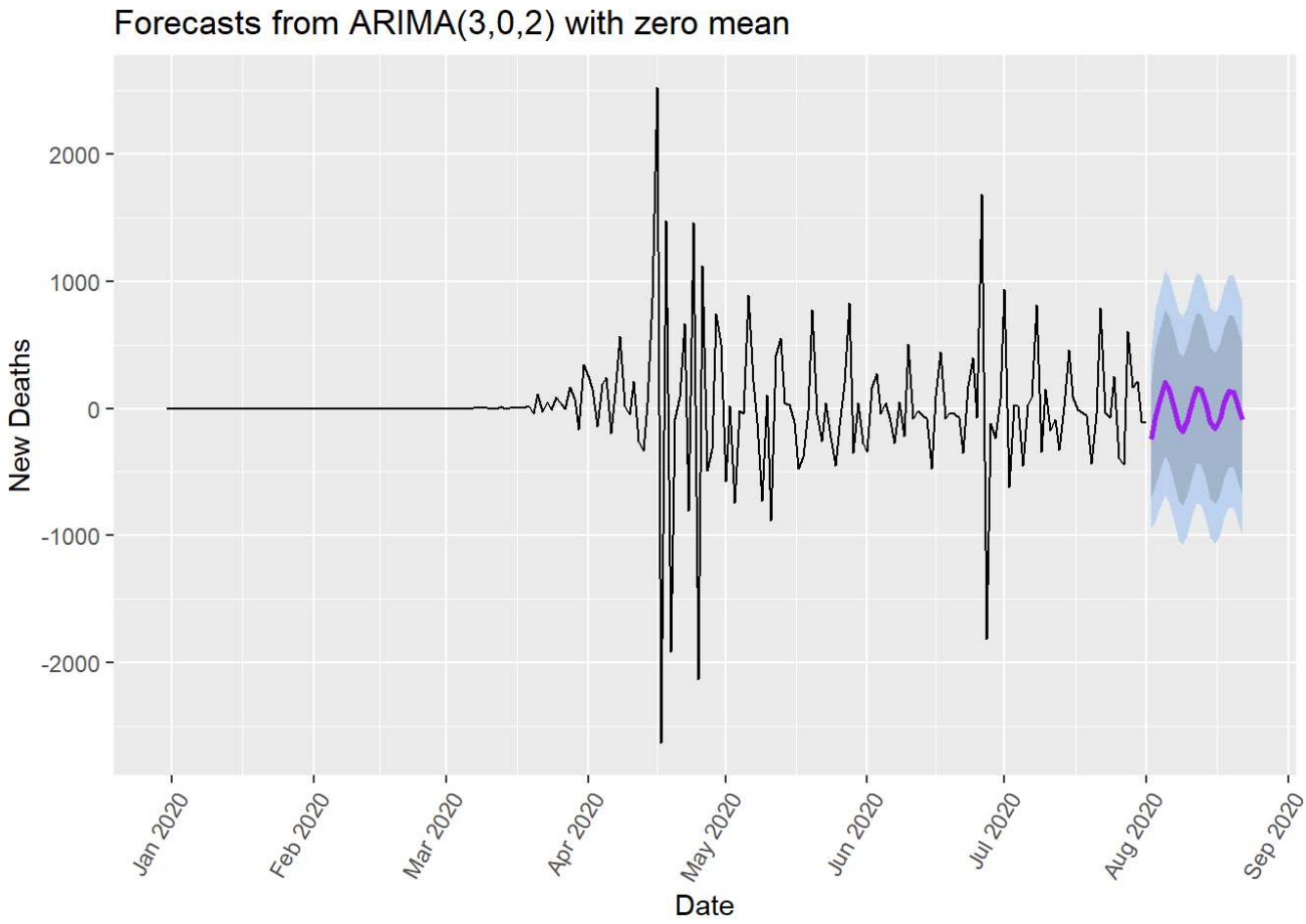
date <date>	new_deaths <dbl>
2020-01-06	0.00000
2020-01-07	0.00000
2020-01-08	0.00000
2020-01-09	0.00000

1-10 of 236 rows Previous **1** 2 3 4 5 6 … 24 Next

```
forecast1_data <- as.data.frame(forecast1)
forecast1_data <- data.frame(date=seq(from=18476, by=1, length=21), forecast1_data, y = delt
a)
forecast1_data$date <- as.Date(forecast1_data$date)

graph <- ggplot(data=z,aes(x=date,y=new_deaths),show.legend = FALSE)+
  theme(axis.text.x=element_text(angle=60, hjust=1))+
  geom_line(data=original,aes(x=as.Date(date),y=new_deaths))+
  geom_ribbon(data=forecast1_data,aes(x = date,ymin =Lo.95, ymax =Hi.95), inherit.aes = FALS
E,fill = "lightsteelblue2")+
  geom_ribbon(data=forecast1_data,aes(x = date,ymin =Lo.80, ymax =Hi.80), inherit.aes = FALS
E,fill = "lightsteelblue3")+
  geom_line(data=forecast1_data,aes(x=date,y=y),size=1,color='purple')+
  ggtitle("Forecasts from ARIMA(3,0,2) with zero mean")+
  scale_x_date(date_breaks = "1 month", date_labels =  "%b %Y")+
  labs(x="Date",y="New Deaths")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
```

```
graph
```



Here, the forecasts are shown as a purple line, with the 80% prediction intervals as a dark shaded area and the 95% prediction intervals as a light shaded area.

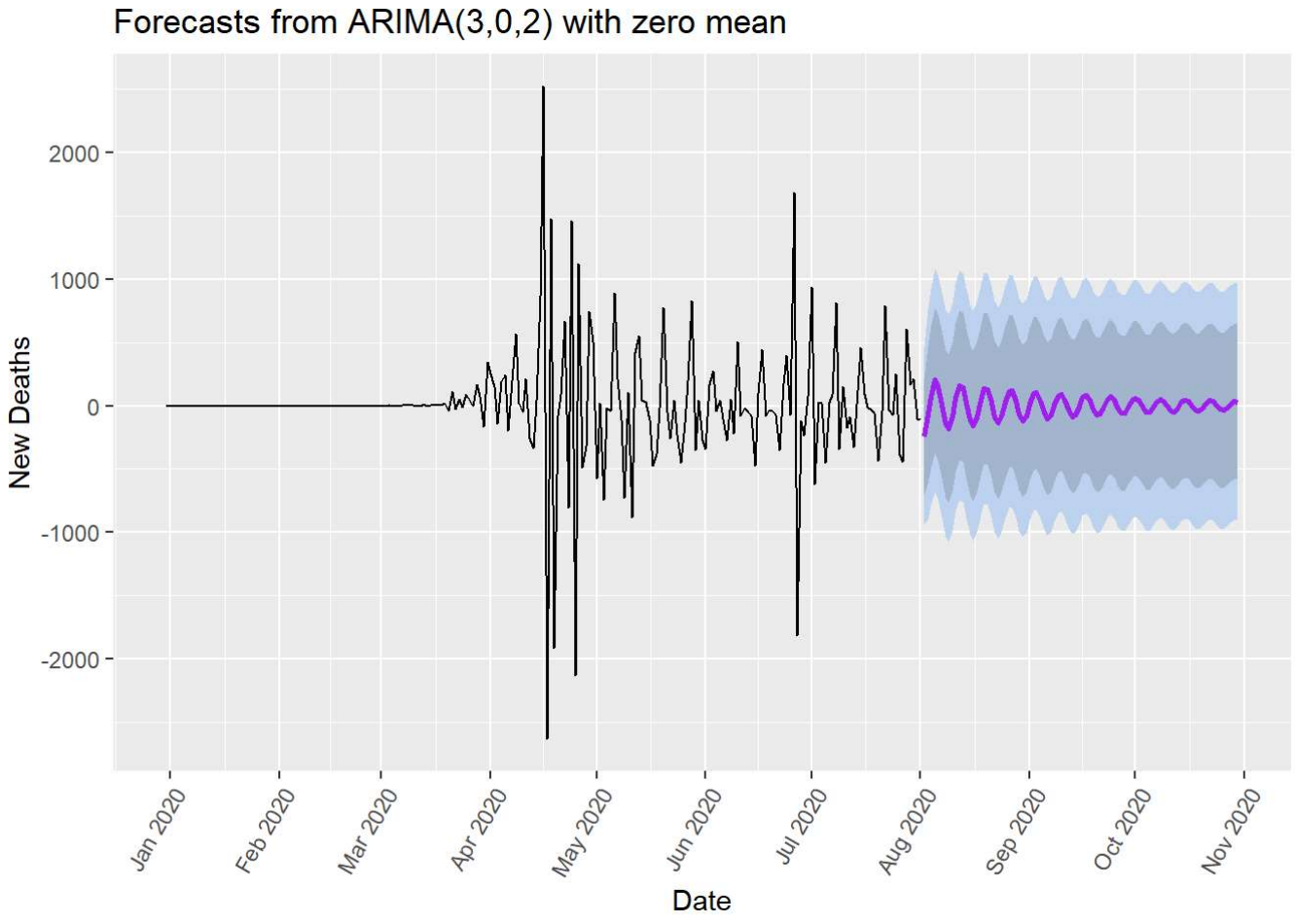
```
delta2 <- (forecast2$lower[,2]+forecast2$upper[,2])/2

predicted_22 <- data.frame(date=seq(from=18476,by=1,length=nrow(forecast2$lower)),new_deaths=
delta2) # prediction from ARIMA
original <- data.frame(date=as.numeric(data_US_Analysis$date),new_deaths=c(0,stationary_dat
a)) # Previous Observations
z2 <- rbind(original,predicted_22) # combine the two data sets
z2$date <- as.Date(z2$date) # convert numeric column to date (creating the base layer of ggpl
ot)
z2
```

date <date>	new_deaths <dbl>
2019-12-31	0.000000
2020-01-01	0.000000
2020-01-02	0.000000
2020-01-03	0.000000
2020-01-04	0.000000
2020-01-05	0.000000
2020-01-06	0.000000

date <date>	new_deaths <dbl>
2020-01-07	0.000000
2020-01-08	0.000000
2020-01-09	0.000000

1-10 of 305 rows Previous 1 2 3 4 5 6 ... 31 Next

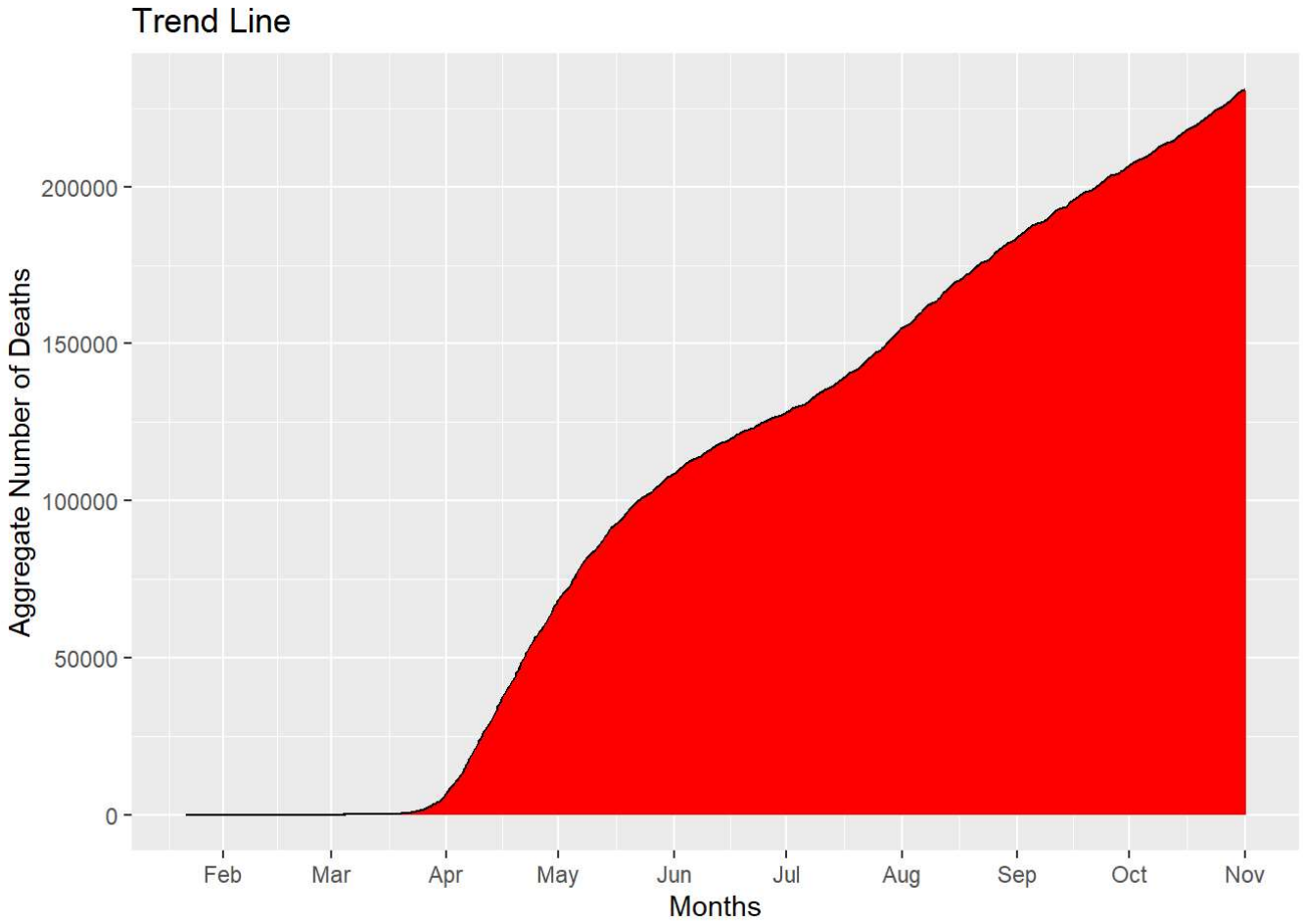
```
forecast2_data <- as.data.frame(forecast2) # convert forecast data into data.frame
forecast2_data <- data.frame(date=seq(from=18476,by=1,length=90),forecast2_data, y = delta2)
#add a date and delta of Lo.95 & Hi.95
forecast2_data$date <- as.Date(forecast2_data$date) # covert to date

graph1 <- ggplot(data=z2,aes(x=date,y=new_deaths),show.legend = FALSE)+
  theme(axis.text.x=element_text(angle=60, hjust=1))+
  geom_line(data=original,aes(x=as.Date(date),y=new_deaths))+
  geom_ribbon(data=forecast2_data,aes(x = date,ymin =Lo.95, ymax =Hi.95), inherit.aes = FALS
E,fill = "lightsteelblue2")+
  geom_ribbon(data=forecast2_data,aes(x = date,ymin =Lo.80, ymax =Hi.80), inherit.aes = FALS
E,fill = "lightsteelblue3")+
  geom_line(data=forecast2_data,aes(x=date,y=y),size=1,color='purple')+
  ggtitle("Forecasts from ARIMA(3,0,2) with zero mean")+
  scale_x_date(date_breaks = "1 month", date_labels =  "%b %Y")+
  labs(x="Date",y="New Deaths")

graph1
```



```
covid_new_data <- read.csv("owid-covid-data1.csv")

covid_new_data$date <- as.Date(covid_new_data$date)
```

```
new_data_US <- covid_new_data[covid_new_data$location == "United States",c(3,4,8,9)]

new_data_US <- new_data_US %>%
  filter(date < '2020-11-02')

new_data_US[is.na(new_data_US)] <- 0
#head(new_data_US)
```

```
p2 <- ggplot(new_data_US, aes(x=date, y=total_deaths)) +
  geom_line() + scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  xlab("Months") + ylab("Aggregate Number of Deaths")+ggtitle("Trend Line")+geom_area(fill="r
ed", color="black")
p2
```



Comparing with actual records

```
temp_21 <- new_data_US %>%
  filter(date > '2020-08-01' & date < '2020-08-22')

additional_21 <- round(sum(temp_21$new_deaths))
# [1] 18589

total_21 <- round(sum(data_US_Analysis$new_deaths) + additional_21)
# [1] 171903

additional_21
```

```
## [1] 19536
```

```
total_21
```

```
## [1] 172850
```

```
print(paste0("Predicted deaths in next 21 days: ", additional_deaths))
```

```
## [1] "Predicted deaths in next 21 days: 18589"
```

```
print(paste0("Predicted total deaths in next 21 days: ", total_number_of_deaths))
```

```
## [1] "Predicted total deaths in next 21 days: 171903"
```

```
print(paste0("Difference with actual additional deaths: ", abs(additional_deaths - additional
_21)))
```

```
## [1] "Difference with actual additional deaths: 947"
```

```
print(paste0("Difference with actual total deaths: ", abs(total_number_of_deaths - total_2
1)))
```

```
## [1] "Difference with actual total deaths: 947"
```

```
temp_90 <- new_data_US %>%
  filter(date > '2020-08-01' & date < '2020-11-01')

additional_90 <- round(sum(temp_90$new_deaths))
# [1] 82653

total_90 <- round(sum(data_US_Analysis$new_deaths) + additional_90)
# [1] 235967

additional_90
```

```
## [1] 75454
```

```
total_90
```

```
## [1] 228768
```

```
# additional_deaths2
# total_number_of_deaths2

print(paste0("Predicted deaths in next 90 days: ", additional_deaths2))
```

```
## [1] "Predicted deaths in next 90 days: 82653"
```

```
print(paste0("Predicted total deaths in next 90 days: ", total_number_of_deaths2))
```

```
## [1] "Predicted total deaths in next 90 days: 235967"
```

```
print(paste0("Difference with actual additional deaths: ", abs(additional_deaths2 - additiona
l_90)))
```

```
## [1] "Difference with actual additional deaths: 7199"
```

```
print(paste0("Difference with actual total deaths: ", abs(total_number_of_deaths2 - total_9
0)))
```

```
## [1] "Difference with actual total deaths: 7199"
```