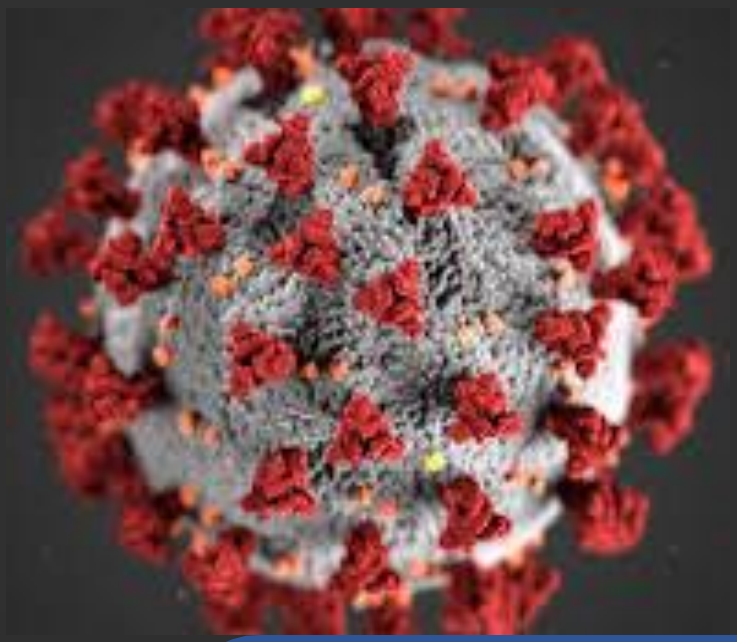


# MSIN0025- Data Analytics II: Final Assignment

Word Count : 1998

## Section 1: The Problem

The White House Office of Science and Technology Policy has brought together a coalition of research groups and companies to prepare the COVID-19 open research dataset in an attempt to answer key open scientific questions about COVID-19. The answers to these questions are intended to assist the National Academy of Sciences, Engineering and Medicine and the World Health Organization in the fight against the virus.[1]

My goal here is to predict the estimated number of confirmed cases and fatalities for the next two months. For the sake of clarity, I have chosen to focus on three countries that are among those most affected by the virus: Italy, France and the United Kingdom.

The data I used was found in Kaggle and comes from the Johns Hopkins University Center for Systems Science and Engineering. As it is a public and reliable Institution, the data should be accurate. However, these figures may not be entirely accurate because countries sometimes have difficulty in knowing exactly how many people are affected by the virus or how many deaths are caused by the virus. The dataset shows the number of Confirmed Cases and Fatalities recorded from 22 January 2020 to 6 April 2020 by Country/Region (or Province/State if relevant).

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 |
| **23253** | 32708 | | Zimbabwe | 2020-04-03 | 9 | 1 |
| **23254** | 32709 | | Zimbabwe | 2020-04-04 | 9 | 1 |
| **23255** | 32710 | | Zimbabwe | 2020-04-05 | 9 | 1 |
| **23256** | 32711 | | Zimbabwe | 2020-04-06 | 10 | 1 |

The two target attributes I chose are the number of deaths and new cases per day. Initially, the dataset showed the number of deaths and cases cumulated, so I had to derive the columns "Fatalities" and "ConfirmedCases" using the following code:

1 (Kaggle, 2020)

```r
#Create a new column with the number of fatalities per day in Italy
library(dplyr)

train_Italy <- train_Italy %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))
```

```r
#Create a new column with the number of new cases per day in Italy
library(dplyr)

train_Italy <- train_Italy %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))
```

I have chosen these attributes in order to be able to better observe trends, otherwise the curves obtained would necessarily have been increasing.

## Section 2: Understand the Data

The dataset I downloaded from Kaggle contained 32711 rows and 6 columns. However, as mentioned before, I chose to work only on Italy, United Kingdom and France. After excluding the rest of the countries and adding my two derived target attributes: Number of new cases per day (named "diff" in my dataset and number of deaths per day (named diff_fatalities), my dataset contained 228 columns and 8 rows.

As you will be able to notice, I separated the dataset into train_Italy, train_France and train_UK which respectively contained only the rows related to Italy, France and UK in order to deliver a more understandable and clear analysis.

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 1 | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 | 0 |
| 4 | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 | 0 |
| 73 | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 | 684 |
| 74 | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 | 708 |
| 75 | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 | 621 |
| 76 | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 | 439 |

| | X | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 15516 | NA | Italy | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 2 | 15517 | NA | Italy | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 3 | 15518 | NA | Italy | 2020-01-24 | 0 | 0 | 0 | 0 |
| 4 | 4 | 15519 | NA | Italy | 2020-01-25 | 0 | 0 | 0 | 0 |
| 73 | 73 | 15588 | NA | Italy | 2020-04-03 | 119827 | 14681 | 4585 | 766 |
| 74 | 74 | 15589 | NA | Italy | 2020-04-04 | 124632 | 15362 | 4805 | 681 |
| 75 | 75 | 15590 | NA | Italy | 2020-04-05 | 128948 | 15887 | 4316 | 525 |
| 76 | 76 | 15591 | NA | Italy | 2020-04-06 | 132547 | 16523 | 3599 | 636 |

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 1 | 13055 | | France | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 13056 | | France | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 13057 | | France | 2020-01-24 | 2 | 0 | 2 | 0 |
| 4 | 13058 | | France | 2020-01-25 | 3 | 0 | 1 | 0 |
| 73 | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 | 1120 |
| 74 | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 | 1053 |
| 75 | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 | 518 |
| 76 | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 | 833 |

As explained in the last section, the datasets show the number of Confirmed Cases and Fatalities recorded from 22 January 2020 to 6 April 2020 and the two target attributes I chose are the number of deaths per day and the number of new cases per day, they are derived from the number of Confirmed Cases and Fatalities.

Not surprisingly, the attributes "Fatality" and "ConfirmedCases" are highly positively correlated:

```
cor(train$ConfirmedCases, train$Fatalities, use ="complete.obs")
```

```
0.866822535211556
```

Now let's compare the evolution of the average number of new cases and deaths per day by month in France, Italy and the UK.

```
aggregate(train_France$diff ~
          train_France$month, data = train_France, FUN = mean)
```

| train_France$month | train_France$diff |
|---|---|
| Jan | 0.500000 |
| Feb | 3.275862 |
| Mar | 1678.322581 |
| Apr | 7647.000000 |

**Average number of new cases per day by month in France**

```
aggregate(train_France$diff_fatalities ~ train_France$month, data = train_France, FUN = mean)
```

| train_France$month | train_France$diff_fatalities |
|---|---|
| Jan | 0.00000000 |
| Feb | 0.06896552 |
| Mar | 113.58064516 |
| Apr | 898.00000000 |

**Average number of deaths per day by month in France**

```
aggregate(train_UK$diff ~
          train_UK$month, data = train_UK, FUN = mean)
```

| train_UK$month | train_UK$diff |
|---|---|
| Jan | 0.2000000 |
| Feb | 0.7241379 |
| Mar | 810.5483871 |
| Apr | 4409.6666667 |

**Average number of new cases per day by month in the UK**

```
aggregate(train_UK$diff_fatalities ~ train_UK$month, data = train_UK, FUN = mean)
```

| train_UK$month | train_UK$diff_fatalities |
|---|---|
| Jan | 0.00000 |
| Feb | 0.00000 |
| Mar | 57.70968 |
| Apr | 597.33333 |

**Average number of deaths per day by month in the UK**

```
aggregate(train_Italy$diff ~
          train_Italy$month, data = train_Italy, FUN = mean)
```

| train_Italy$month | train_Italy$diff |
|---|---|
| Jan | 0.20000 |
| Feb | 38.82759 |
| Mar | 3376.25806 |
| Apr | 4459.16667 |

**Average number of new cases per day by month in Italy**

```
In [86]: aggregate(train_Italy$diff_fatalities ~ train_Italy$month, data = train_Italy, FUN = mean)
```

| train_Italy$month | train_Italy$diff_fatalities |
|---|---|
| Jan | 0.0000 |
| Feb | 1.0000 |
| Mar | 399.9677 |
| Apr | 682.5000 |

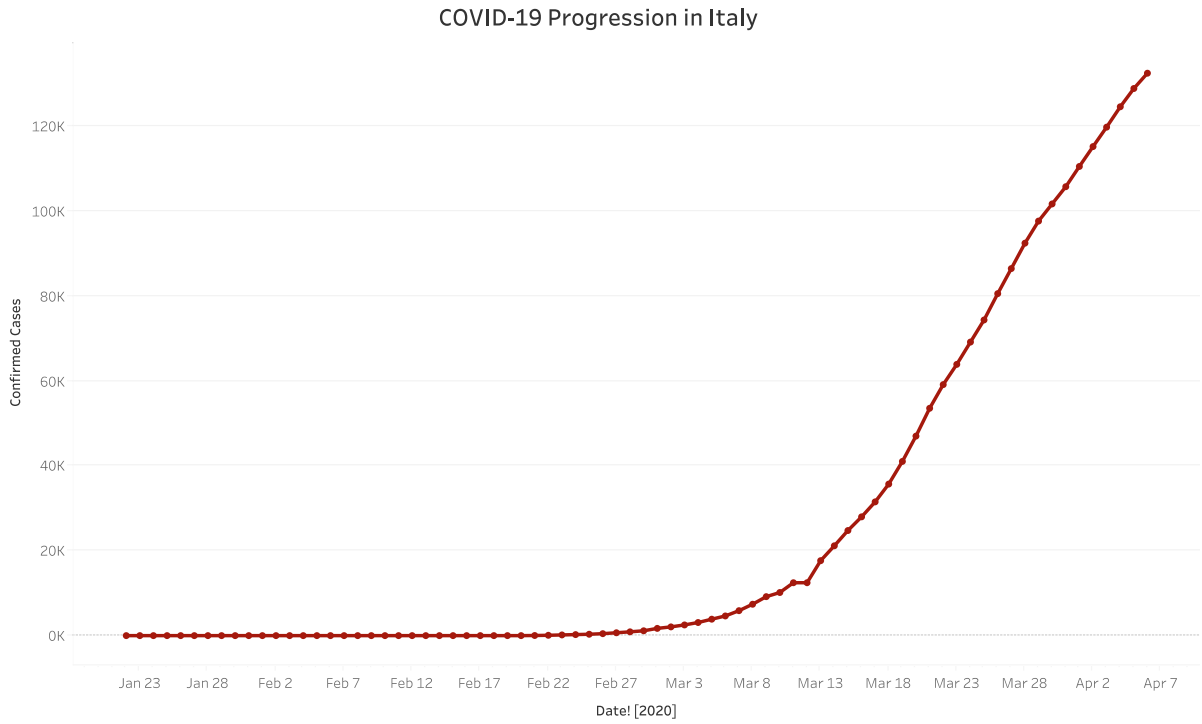
**Average number of deaths per day by month in Italy**

We can see that in Italy, the virus started to spread earlier. Indeed, in February the average number of new cases per day was already around 39 compared to 3.27 and 0.7 in France and the UK. The average number of deaths per day did not exceed 1 in February in any of these countries.

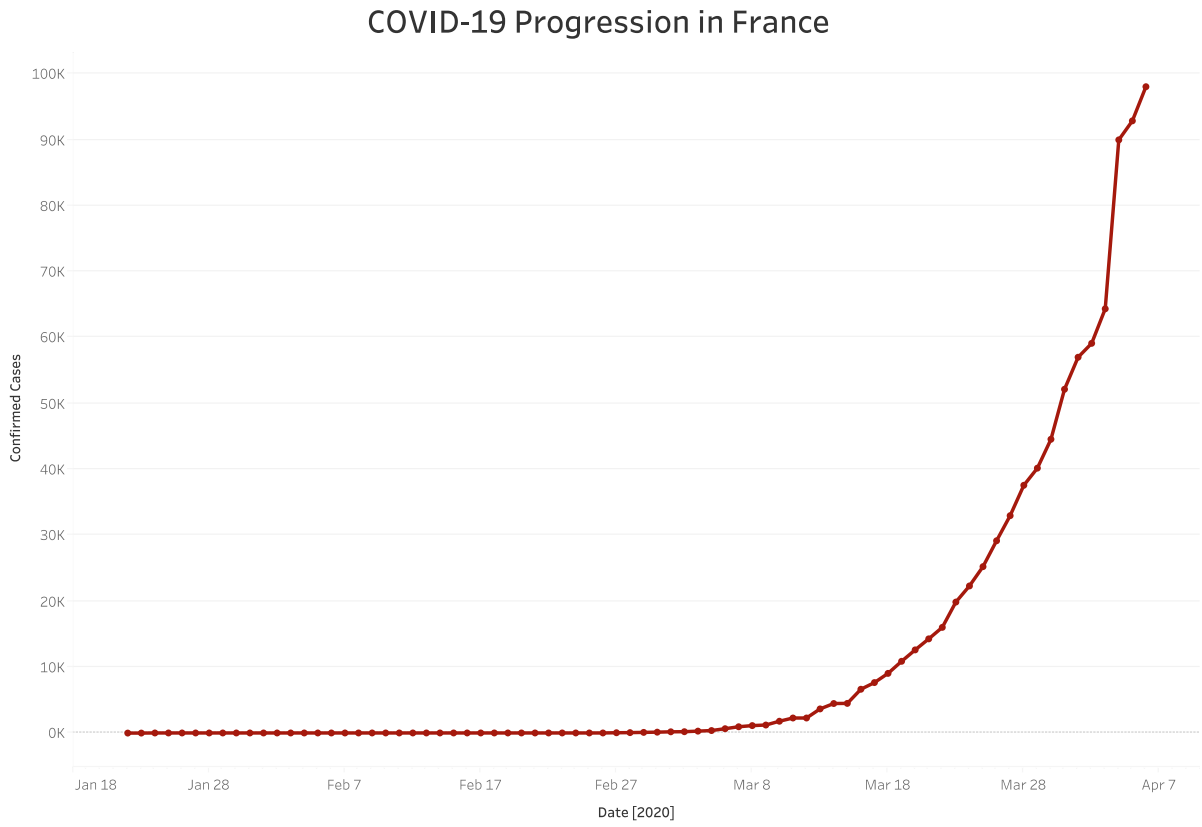
In March, the virus began to spread widely in all three countries. However, it developed much faster in Italy where the average number of new cases per day reached 3376 with an average of 400 deaths, compared to 1678 and 113 respectively in France. In the UK, the virus spread more slowly but still reached an average of 810 new cases per day and 57 deaths.

Finally, from the first to the 6th of April, both countries were largely affected by the virus with an average of 4409 cases and 597 deaths per day in the UK and 3376 cases and 400 deaths in Italy. The virus was particularly violent in France, which recorded almost twice as many cases as in Italy and the UK per day (7647 new cases and 898 deaths per day on average).

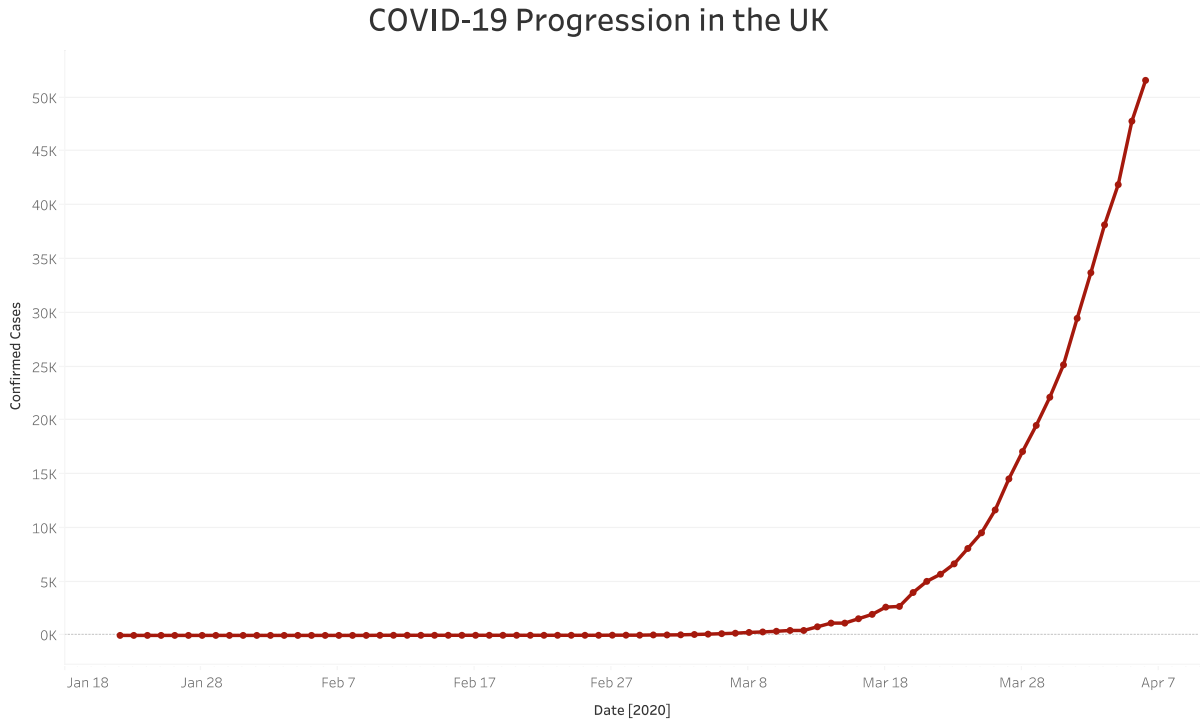
Using Tableau, I generated three plots showing the evolution of the number of cases in the three countries. I used the Confirmedcases variable which adds each day the number of new cases on that day to the number of cases recorded so far. Thus, the flatter the curve, the lower the number of new cases per day.




The trend of sum of Confirmed Cases for Date!.




Date vs. Confirmed Cases.




Date vs. Confirmed Cases.

We can notice that the virus arrived earlier in Italy but spread rather linearly, whereas in France and the UK it arrived later but then spread exponentially.

## Section 3: Prepare the Data

As stated in the last sections, in order to prepare my data for each country, I first created two variables: diff and diff_fatalities showing the number of new cases and the number of deaths per day, they are derived from the variable "ConfirmedCases" and "Fatalities".

Then, I created a time series for the diff and diff_fatalities variables in the training dataset of each country (France, Italy and UK) in order to be able to prepare models.

```
dl <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")
set.seed(25)
myts <- ts(train_Italy$diff,      # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
plot(myts, ylab = "Number of Cases")
```

Code for the time series of the variable diff in Italy

In order to create a test dataset, I downloaded a new dataset which was actually the same as the one I used for the training dataset but updated with the number of confirmed cases and deaths up to April 13th, which is one week longer than the training dataset.

```
test <- read.csv("test.csv")
n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | ForecastId | Province_State | Country_Region | Date | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|
| 1 | 1 | | Afghanistan | 2020-01-22 | 0 | 0 |
| 2 | 2 | | Afghanistan | 2020-01-23 | 0 | 0 |
| 3 | 3 | | Afghanistan | 2020-01-24 | 0 | 0 |
| 4 | 4 | | Afghanistan | 2020-01-25 | 0 | 0 |
| 25976 | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 |
| 25977 | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 |
| 25978 | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 |
| 25979 | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 |

I created again the diff and diff_fatalities variables:

```
#Create a new column with the number of new cases per day
library(dplyr)

test <- test %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

#Create a new column with the number of fatalities per day

test <- test %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))


n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | ForecastId | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | Afghanistan | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 2 | | Afghanistan | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 3 | | Afghanistan | 2020-01-24 | 0 | 0 | 0 | 0 |
| 4 | 4 | | Afghanistan | 2020-01-25 | 0 | 0 | 0 | 0 |
| 25976 | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 | 2 | 0 |
| 25977 | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 | 1 | 0 |
| 25978 | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 | 0 | 0 |
| 25979 | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 | 3 | 0 |

Then I only kept the additional values (compared to the training dataset), i.e. from the 7th to the 13th of April and the values of the countries I was interested in(Italy, France or UK).

```
test$Date <- as.Date(test$Date, "%Y-%m-%d")

#Take values between the 7th and the 13th of April and for country = Italy
test_Italy = test[which(test$Country_Region=='Italy' & test$Province_State =='' & test$Date >= as.Date("2020-4-7") ),]
```

Code for creating the test dataset and the variable diff for Italy

Finally, I created a time series for the diff and diff_fatalities variables of the test dataset in each country in order to be able to test my upcoming models.

To be able to prepare an ensemble model, I split my training dataset into a small training dataset which contained the value up to the 24th of March and a validation dataset which contained the remaining values (from the 24thof March to the 6thof April)

```
#Step 1. Split the training set into a smaller training set and a validation set.

train_Italy$Date <- as.Date(train_Italy$Date, "%Y-%m-%d")

smalltrainItaly.df <- train_Italy[which(train_Italy$Date <= as.Date("2020-3-24") ),]

validItaly.df <- train_Italy[which(train_Italy$Date >= as.Date("2020-3-24") ),]
```

Code to split the training dataset for Italy

# Section 4: Generate and Test Prediction Models

Using the training and testing datasets, I computed the following models to predict the future outcome of the independent variables diff and diff_fatalities:

- An exponential smoothing model
- A Holt's linear trend model
- A Time series linear model
- Three Ensemble Model(Stacking)

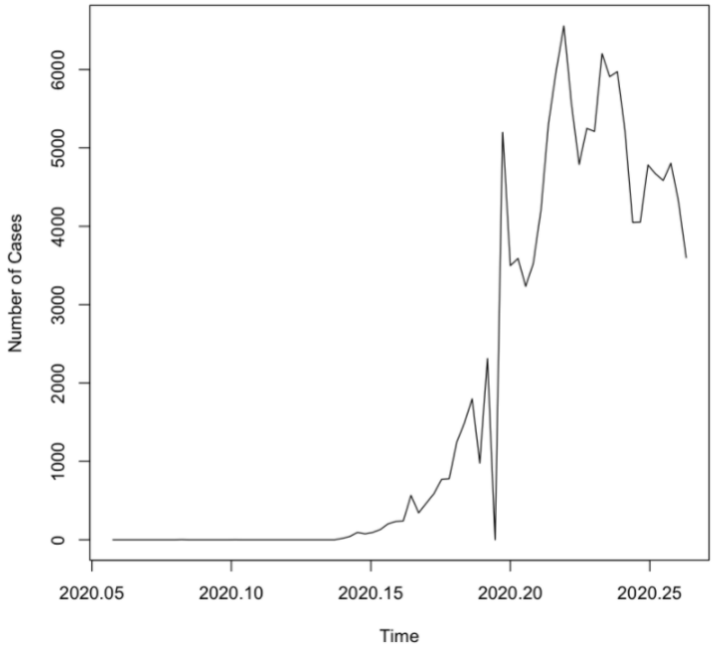
Each of these models were prepared and computed for predicting the future number of new cases(diff) and deaths(diff_fatalities) per day in Italy, France and the UK. Here are the forecasts I made for the number of new cases per day in Italy, refer to the appendix for the forecasts regarding deaths per day in Italy and the forecasts regarding the number of new cases and deaths per day in France and the UK.

## For predicting the number of new cases per day in Italy

### 1) Exponential smoothing model

I first developed an exponential smoothing model using the following time series myts which is constituted of the number of new cases per day from the 22nd of January to the 6th of April (the whole training dataset).

```
dl <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")
set.seed(25)
myts <- ts(train_Italy$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
plot(myts, ylab = "Number of Cases")
```




```
#Generate an exponential smoothing ets model

(myts.ANN <- ets(myts, model = "ANN"))
```

```
ETS(A,N,N)

Call:
 ets(y = myts, model = "ANN")

  Smoothing parameters:
    alpha = 0.5473

  Initial states:
    l = -0.0568

  sigma:  712.2709

     AIC     AICc      BIC
1331.515 1331.848 1338.507
```

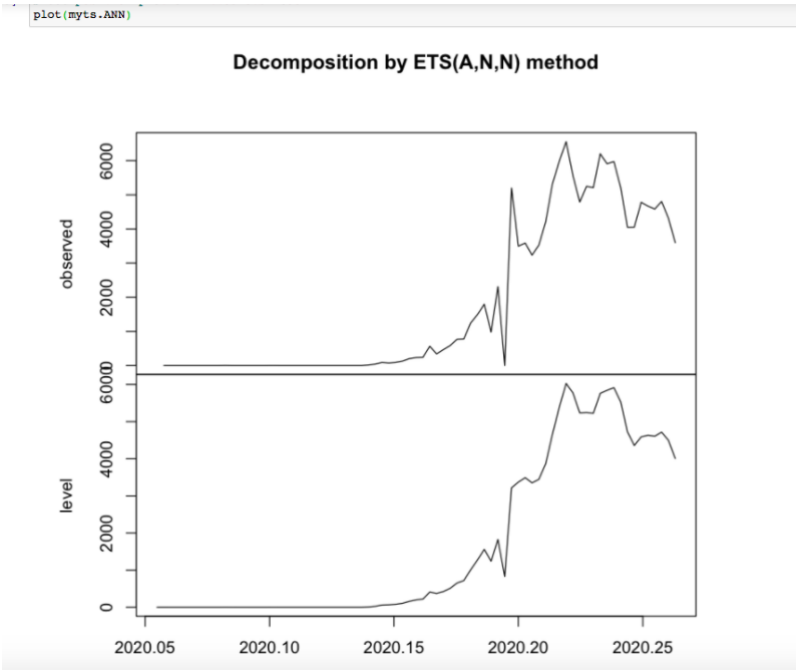
I obtained the following RMSE error:

```
#RMSE error for the ets model

rmse.ets <- function (etsmodel) cat("RMSE = ", sqrt(etsmodel$mse))

rmse.ets(myts.ANN)
```

```
RMSE =  702.8364
```

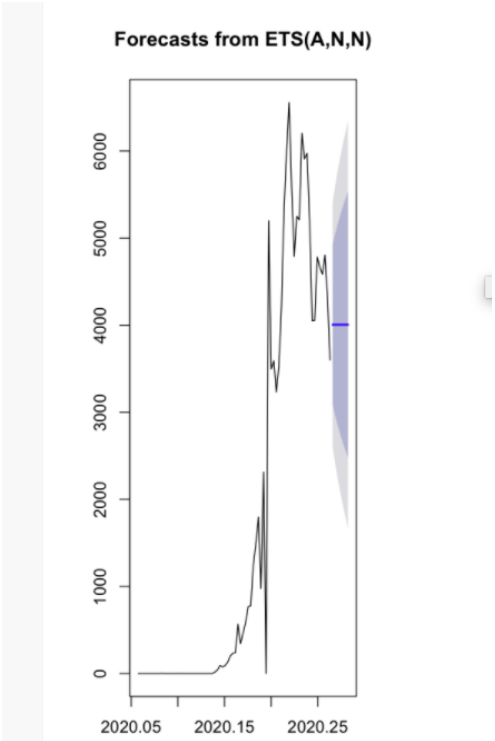
The following picture shows both the actual values (**"observed"**) and the exponentially smoothed average (**"level"**).

```
plot(myts.ANN)
```




Then I generated forecasts until April 13th:

```
#Generate forecasts to April 13th 2020  using the above models
myts.ANN.pred <- forecast(myts.ANN, h = 7)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(myts.ANN.pred)
```



According to this model, the number of new cases per day in Italy should be constant from the 7th to the 13th of April.

### 2) Holt's linear trend model

I developed the following Holt's linear model using the time series myts:

```
#Generate an additive trend (Holt's linear) ets model

(myts.AAN <- ets(myts, model = "AAN"))
```

```
ETS(A,Ad,N)

Call:
 ets(y = myts, model = "AAN")

  Smoothing parameters:
    alpha = 0.1409
    beta  = 0.1409
    phi   = 0.9021

  Initial states:
    l = -0.2626
    b = 0.069

  sigma:  688.9336

     AIC     AICc      BIC
1329.306 1330.523 1343.290
```

I obtained the following RMSE error:

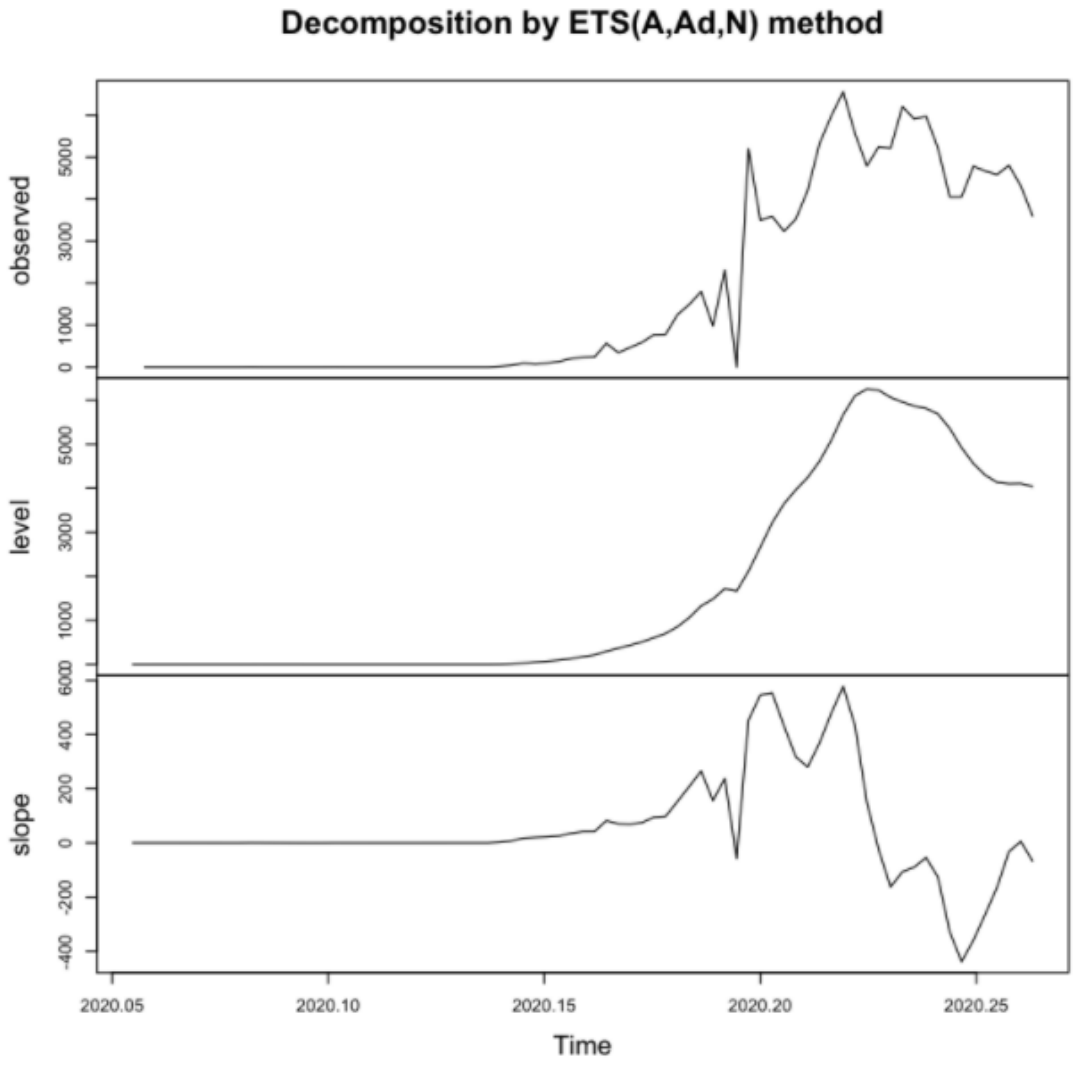
```
rmse.ets(myts.AAN)
```

```
RMSE =  665.8857
```

As this RMSE is lower than the exponential smoothing model's one therefore this model better fit the training dataset.

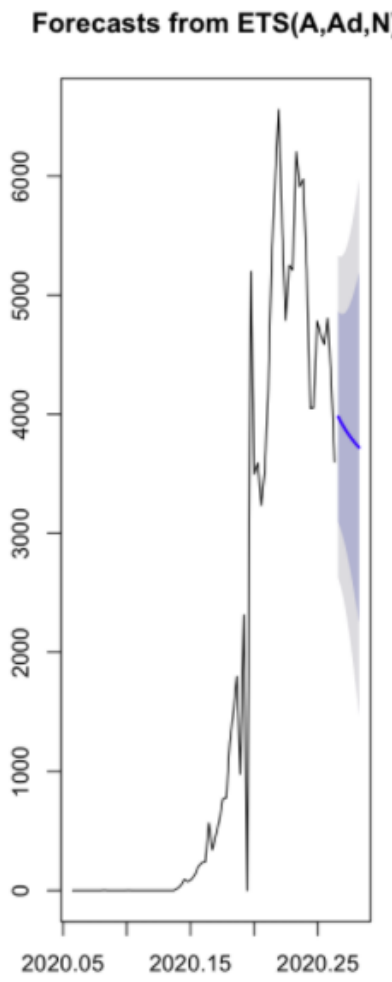
The following plot shows both the actual values(**"observed"**), the exponentially smoothed average(**"level")**, and the trend values(**"slope"**).

```
#Decomposition plot of fitted additive trend ets model

plot(myts.AAN)
```




Then I generated forecasts until April 13th:

```
#Generate forecast to April 13th 2020  using the above model
myts.AAN.pred <- forecast(myts.AAN, h = 7)

# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(myts.AAN.pred)
```




According to this model, the number of new cases per day in Italy should decrease from the 7th to the 13th of April.

### 3) Time series linear model

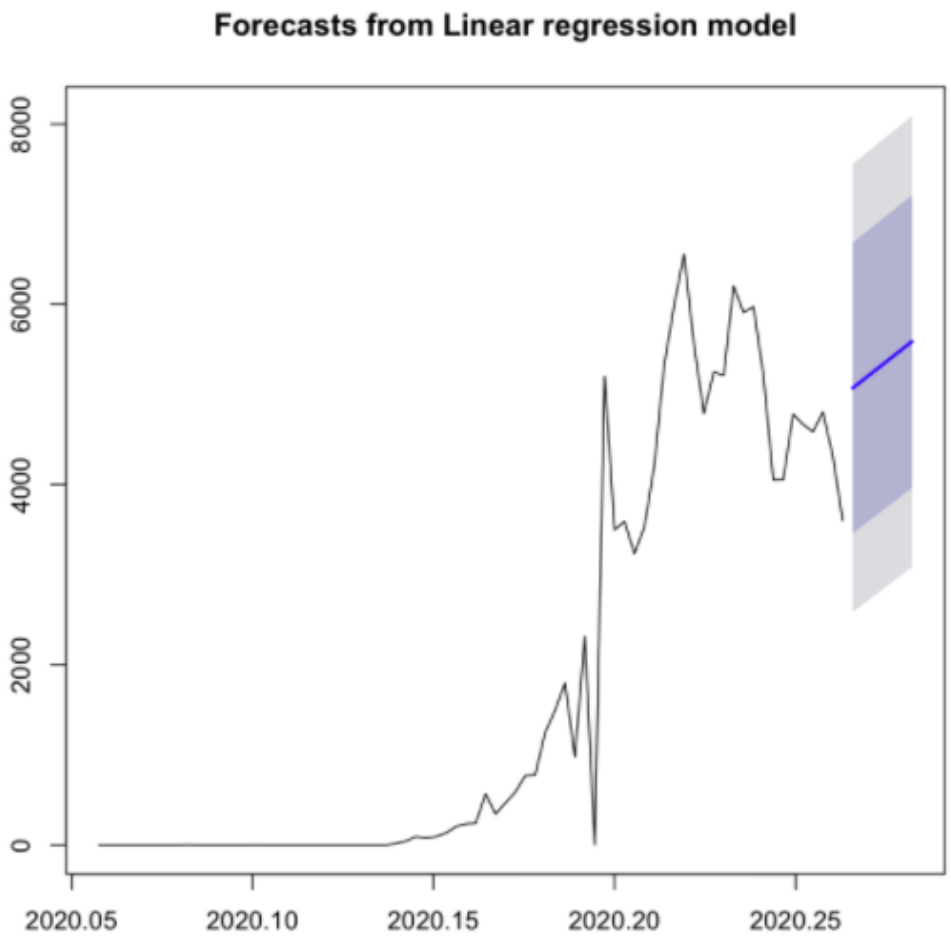
Again, using the time series myts, I have developed the following time series linear model:

```
tslm_model_diffItaly
```

```
Call:
tslm(formula = myts ~ trend)

Coefficients:
(Intercept)        trend
    -1582.5         86.4
```

The model predicted that the number of new cases per day will increase from the 7th to the 13th of April.

```
tslm_model_diffItaly.pred <- forecast(tslm_model_diffItaly, h = 7)
plot(tslm_model_diffItaly.pred)
```




My test dataset contained the number of confirmed cases up to April 13th, which is one week longer than the training dataset. From this dataset I created the following time series in order to test my models:

```
test_Italy
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **12195** | 16721 | | Italy | 2020-04-07 | 135586 | 17127 | 3039 | 604 |
| **12196** | 16722 | | Italy | 2020-04-08 | 139422 | 17669 | 3836 | 542 |
| **12197** | 16723 | | Italy | 2020-04-09 | 143626 | 18279 | 4204 | 610 |
| **12198** | 16724 | | Italy | 2020-04-10 | 147577 | 18849 | 3951 | 570 |
| **12199** | 16725 | | Italy | 2020-04-11 | 152271 | 19468 | 4694 | 619 |
| **12200** | 16726 | | Italy | 2020-04-12 | 156363 | 19899 | 4092 | 431 |
| **12201** | 16727 | | Italy | 2020-04-13 | 159516 | 20465 | 3153 | 566 |

```
d2 <- seq(from = as.Date("2020-4-7", "%Y-%m-%d"), to = as.Date("2020-4-13", "%Y-%m-%d"), by = "day")
```

```
Italytest.ts <- ts(test_Italy$diff,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

I tested the models and their accuracy, if we refer to the RMSE error rate, it seems that the Exponential smoothing model is the best fit to the training dataset but that the Holt's linear trend model better fit the test dataset.

```
# Evaluate the performance of the model
accuracy(tslm_model_diffItaly.pred, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 6.150051e-16 | 1199.660 | 999.2115 | NaN | Inf | NaN | 0.76557764 | NA |
| **Test set** | -1.477062e+03 | 1572.026 | 1477.0619 | -41.14607 | 41.14607 | NaN | 0.04214087 | 2.13482 |

```
accuracy(myts.ANN.pred, Italytest.ts)
```

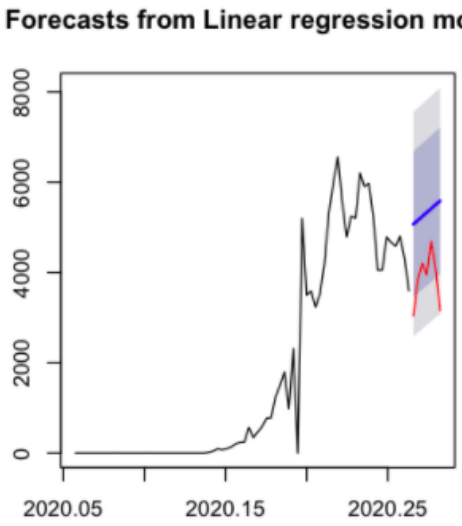
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 96.30564 | 702.8364 | 317.6979 | NaN | Inf | NaN | -0.08510095 | NA |
| **Test set** | -152.70565 | 562.3071 | 430.9171 | -6.160134 | 12.30551 | NaN | 0.07746870 | 0.6559577 |

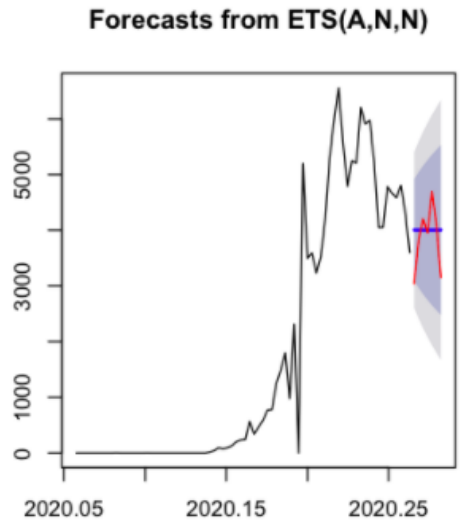
```
accuracy(myts.AAN.pred, Italytest.ts)
```

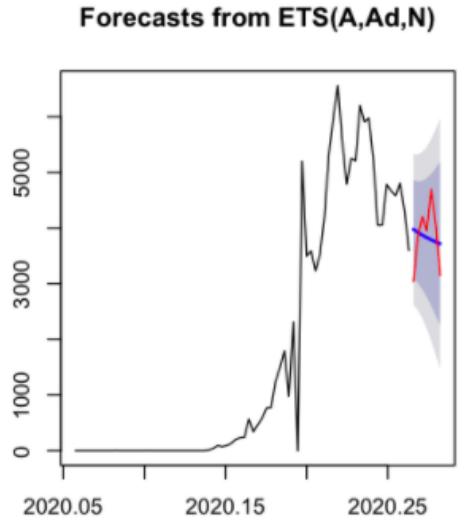
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 31.23647 | 665.8857 | 337.9145 | NaN | Inf | NaN | 0.1027531 | NA |
| **Test set** | 14.48641 | 569.3753 | 469.9547 | -1.813075 | 12.81046 | NaN | 0.1299212 | 0.6795859 |

Here are the plots of the validation forecasts against the actual validation data:

```
#Plot and compare the validation forecasts against the actual validation data

options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))

plot(tslm_model_diffItaly.pred)
lines(Italytest.ts, col="red")  # plots the actual validation data
plot(myts.ANN.pred)
lines(Italytest.ts, col="red")
plot(myts.AAN.pred)
lines(Italytest.ts, col="red")
```

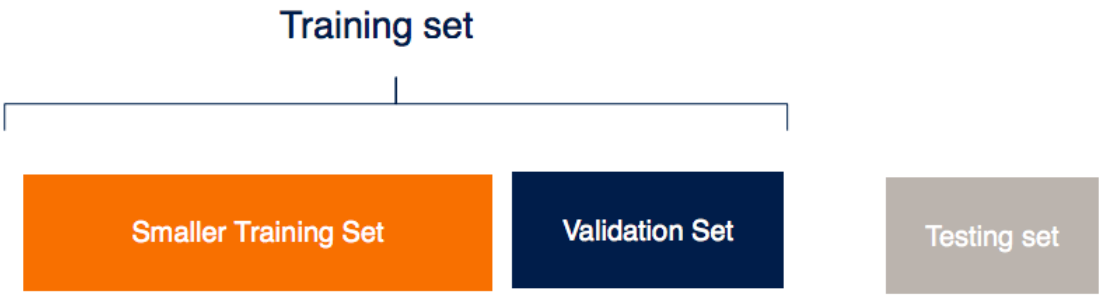



## 4) Ensemble Model --- Stacking

In order to develop an ensemble model, I have split the training dataset into a smaller training set composed of the training set without the last two weeks and a validation set composed of those two weeks.




```
#Step 1. Split the training set into a smaller training set and a validation set.

train_Italy$Date <- as.Date(train_Italy$Date, "%Y-%m-%d")

smalltrainItaly.df <- train_Italy[which(train_Italy$Date <= as.Date("2020-3-24") ),]

validItaly.df <- train_Italy[which(train_Italy$Date >= as.Date("2020-3-24") ),]
```

The second step was to train my models in the small training set to generate predictions in the validation set. I chose to use the Exponential smoothing model and the Holt's linear trend model for my stacker model because they are giving better RMSE scores than the time series linear model.

```
#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainItalydiff.df <- ts(smalltrainItaly.df$diff,     # random data
          start = c(2020, as.numeric(format(dl[1], "%j"))) ,
          frequency = 365)
```

```
#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainItalydiff.df.ANN <- ets(myts_smalltrainItalydiff.df, model = "ANN"))
myts_smalltrainItalydiff.df.ANN.pred <- forecast(myts_smalltrainItalydiff.df.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainItalydiff.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.5258

  Initial states:
    l = -0.0116

  sigma:  726.45

     AIC     AICc      BIC
1095.094 1095.501 1101.524
```

```
#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainItalydiff.df.AAN <- ets(myts_smalltrainItalydiff.df, model = "AAN"))
myts_smalltrainItalydiff.df.AAN.pred <- forecast(myts_smalltrainItalydiff.df.AAN , h = 14)
```

```
ETS(A,A,N)

Call:
 ets(y = myts_smalltrainItalydiff.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1121
    beta  = 0.1121

  Initial states:
    l = -0.3865
    b = 0.1201

  sigma:  686.3518

     AIC     AICc      BIC
1089.840 1090.893 1100.556
```

The third step was to fit a model to the predictions generated in step 2

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.df <- data.frame(Date = validItaly.df$Date, diff = validItaly.df$diff,
                         myts_smalltrainItalydiff.df.ANN.pred = myts_smalltrainItalydiff.df.ANN.pred,
                         myts_smalltrainItalydiff.df.AAN.pred = myts_smalltrainItalydiff.df.AAN.pred)
```

```
library(dplyr)

stacker.df <- stacker.df[,c("Date","diff","myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast",
                            "myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast")]

head(stacker.df)
```

| | Date | diff | myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast | myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 5249 | 5248.946 | 6723.002 |
| **2020.2329** | 2020-03-25 | 5210 | 5248.946 | 6762.216 |
| **2020.2356** | 2020-03-26 | 6203 | 5248.946 | 6801.431 |
| **2020.2384** | 2020-03-27 | 5909 | 5248.946 | 6840.645 |
| **2020.2411** | 2020-03-28 | 5974 | 5248.946 | 6879.859 |
| **2020.2438** | 2020-03-29 | 5217 | 5248.946 | 6919.073 |

I have done 3 stacker models: The first one is a random forest model, the second one a regression tree model and the third one a XGBoost model.

```
library(randomForest)
```

```
#Fit a random forest to the predictions as stacker model 1
stackerModel_1 <- randomForest(stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast
                      + stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast
                      + stacker.df$smalltslm_model_diffItaly.pred.Point.Forecast,
                  data = stacker.df,ntree = 1000, mtry = 3, nodesize = 5, importance = TRUE)
```

```
summary(stackerModel_1)
```

```
                Length Class  Mode
call              7    -none- call
type              1    -none- character
predicted        14    -none- numeric
mse            1000    -none- numeric
rsq            1000    -none- numeric
oob.times        14    -none- numeric
importance        6    -none- numeric
importanceSD      3    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y                14    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

```
#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModel_2 <- rpart(stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast
                      + stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast
                      , data= stacker.df, method= "anova")
```

```
summary(stackerModel_2)
```

```
Call:
rpart(formula = stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast +
    stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast,
    data = stacker.df, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=4901.429, MSE=559806.5
```

```
stacker.df.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.df[,c(-1,-2)])
stacker.y <- as.vector(stacker.df[,2])
stackerModel_3 <- xgboost(stacker.df.xgb, stacker.y, nrounds=200,
                      verbose = 0)

summary(stackerModel_3)
```

```
                Length Class              Mode
handle               1 xgb.Booster.handle externalptr
raw             100174 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                13 -none-             call
params               1 -none-             list
callbacks            1 -none-             list
feature_names        2 -none-             character
nfeatures            1 -none-             numeric
```

The fourth and last step was to re-train my models using the entire training set and generate predictions on the testing set.

```
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

myts <- ts(train_Italy$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
```

```
#Generate an exponential smoothing ets model
(myts.ANN <- ets(myts, model = "ANN"))
myts.ANN.pred.test <- forecast(myts.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts, model = "ANN")

  Smoothing parameters:
    alpha = 0.5473

  Initial states:
    l = -0.0568

  sigma:  712.2709

     AIC     AICc      BIC
1331.515 1331.848 1338.507
```

```
#Generate an additive trend (Holt's linear) ets model
(myts.AAN <- ets(myts, model = "AAN"))
myts.AAN.pred.test <- forecast(myts.AAN, h = 14)
```

```
ETS(A,Ad,N)

Call:
 ets(y = myts, model = "AAN")

  Smoothing parameters:
    alpha = 0.1409
    beta  = 0.1409
    phi   = 0.9021

  Initial states:
    l = -0.2626
    b = 0.069

  sigma:  688.9336

     AIC     AICc      BIC
1329.306 1330.523 1343.290
```

```
predict.variables <- data.frame('myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast' = myts.ANN.pred.test,
                                'myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast' = myts.AAN.pred.test)


predict.variables
```

```
# Predict from stacker model_1 --- random forest
stacker.predict.rt1 <- predict(stackerModel_1, predict.variables )
```

```
# Predict from stacker model_2 --- regression tree
stacker.predict.rt2 <- predict(stackerModel_2, predict.variables )
```

```
# Predict from stacker model_3 --- XGBoost
predict.variables.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables)
```

```
stacker.predict.rt3 <- predict(stackerModel_3, predict.variables.xgb)
```

I obtained the following scores:

```
accuracy(stacker.predict.rt1, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -1305.419 | 1386.809 | 1305.419 | -36.3328 | 36.3328 | 0.1050097 | 1.804676 |

```
accuracy(stacker.predict.rt2, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -1048.714 | 1180.115 | 1048.714 | -29.90806 | 29.90806 | 0.0774697 | 1.506205 |

```
accuracy(stacker.predict.rt3, Italytest.ts)
```

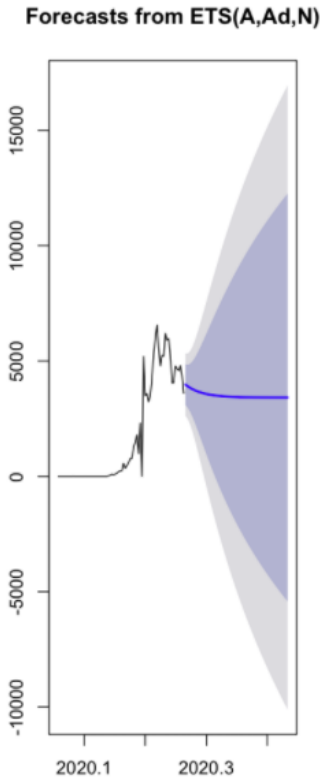
| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -1396.285 | 1497.492 | 1396.285 | -39.1201 | 39.1201 | 0.0774697 | 1.979775 |

Both of my ensemble models obtain a lower RMSE score than the exponential smoothing model (ANN) and the Holt's linear trend model (AAN).

Based on its fit to both the training and the test dataset, it seems that the Holt's linear trend model is the best one.

If we try to forecast for the next two months using this model, we can see that the number of cases per day should slowly decrease to about 3419 in early June.

```
#Generate forecasts to June 6th 2020  using the above model
myts.AAN.pred2months <- forecast(myts.AAN, h = 62)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(myts.AAN.pred2months)
```




```
myts.AAN.pred2months$mean[62]
```

3418.78888356871

## Section 5: Problem Conclusions and Recommendations

| | Italy: 2-month forecast of the number of cases per day | Italy: 2-month forecast of the number of deaths per day | United Kingdom: 2-month forecast of the number of cases per day | United Kingdom: 2-month forecast of the number of deaths per day | France: 2-month forecast of the number of cases per day | France: 2-month forecast of the number of deaths per day |
|---|---|---|---|---|---|---|
| Best model | Holt's linear trend model | Exponential smoothing model | Holt's linear trend model | Exponential smoothing model | Regression tree ensemble stacker model | XGBoost ensemble stacker model |
| 2-month forecast | Slightly increase | Stay constant | Sharply Increase | Stay constant | Stay constant | Stay constant |
| Estimated number of cases/deaths per day early June | 3419 | 623 | 17478 | 439 | 5582 | 833 |

As stated in the Introduction, my goal here was to predict the estimated number of cases and fatalities due to Covid-19 for the next two months in Italy, the United Kingdom and France.

According to my analysis, the number of deaths per day in both three countries is expecting to stay constant for the next two months. The country which is expected to suffer the most from Covid-19 according to the expected number of deaths is France with a forecast of around 833 deaths per day in early June.

Concerning the number of cases, it is expected to increase very sharply in the UK to reach a number of new cases per day of around 17478 at the beginning of June. In Italy, it is expected to increase slightly to around 3419. Finally, in France, it should remain constant at around 5582.

Giving these predictions to the health organisations and governments of these three countries can help to better prevent the virus by, for example, adjusting the number of hospital beds reserved for Covid-19 patients.

However, we can expect that these predictions are not entirely accurate, as all these countries are now under quarantine and this started only a few days before my dataset was created.

# Appendix

## Appendix 1: Generate and Test Prediction Models

### Appendix 1-A: Forecast of the number of fatalities per day in Italy

#### 1) Exponential smoothing model

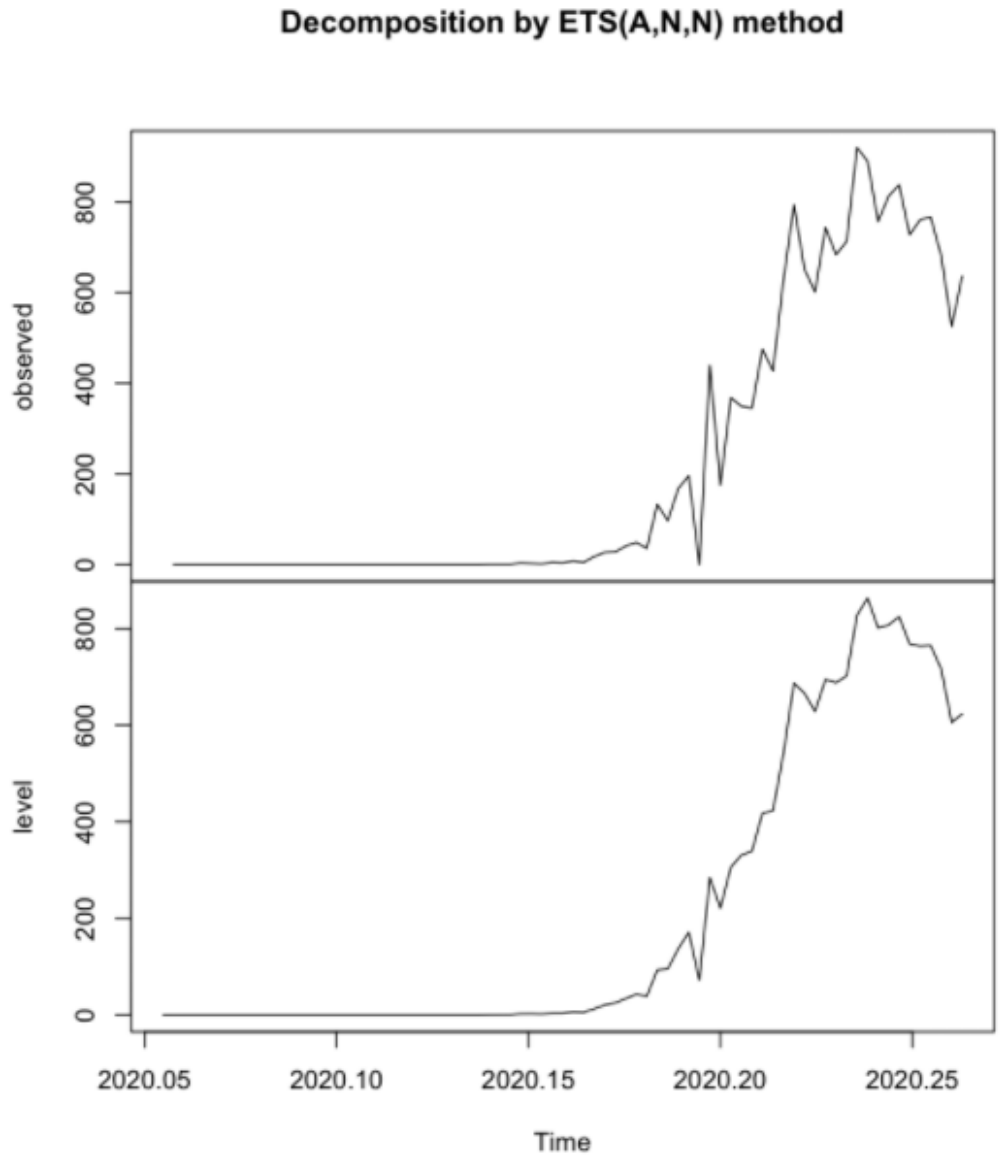
I first developed an exponential smoothing model using the time series mytsfatalities which is constituted of the number of deaths per day from the 22nd of January to the 6th of April (the whole training dataset).

```
#Generate an exponential smoothing ets model

(mytsfatalities.ANN <- ets(mytsfatalities, model = "ANN"))
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalities, model = "ANN")

  Smoothing parameters:
    alpha = 0.5786

  Initial states:
    l = -0.1223

  sigma:  80.4169

     AIC      AICc       BIC
999.9671 1000.3005 1006.9593
```

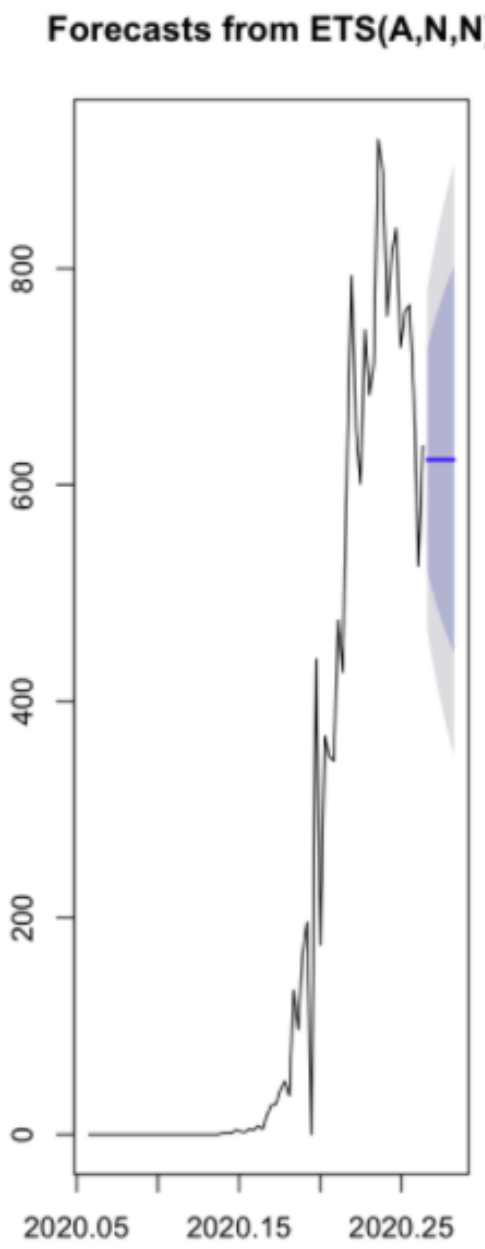
The following picture shows both the actual values ("**observed**") and the exponentially smoothed average ("**level**").

```
#Decomposition plot of fitted ets model¶
plot(mytsfatalities.ANN)
```




Then I generated forecasts until April 13th:

In [138]:
```
#Generate forecasts to May 7th 2020  using the above models
mytsfatalities.ANN.pred <- forecast(mytsfatalities.ANN, h = 7)


# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsfatalities.ANN.pred)
```




According to this model, the number of new cases per day in Italy should be constant from the 7th to the 13th of April.

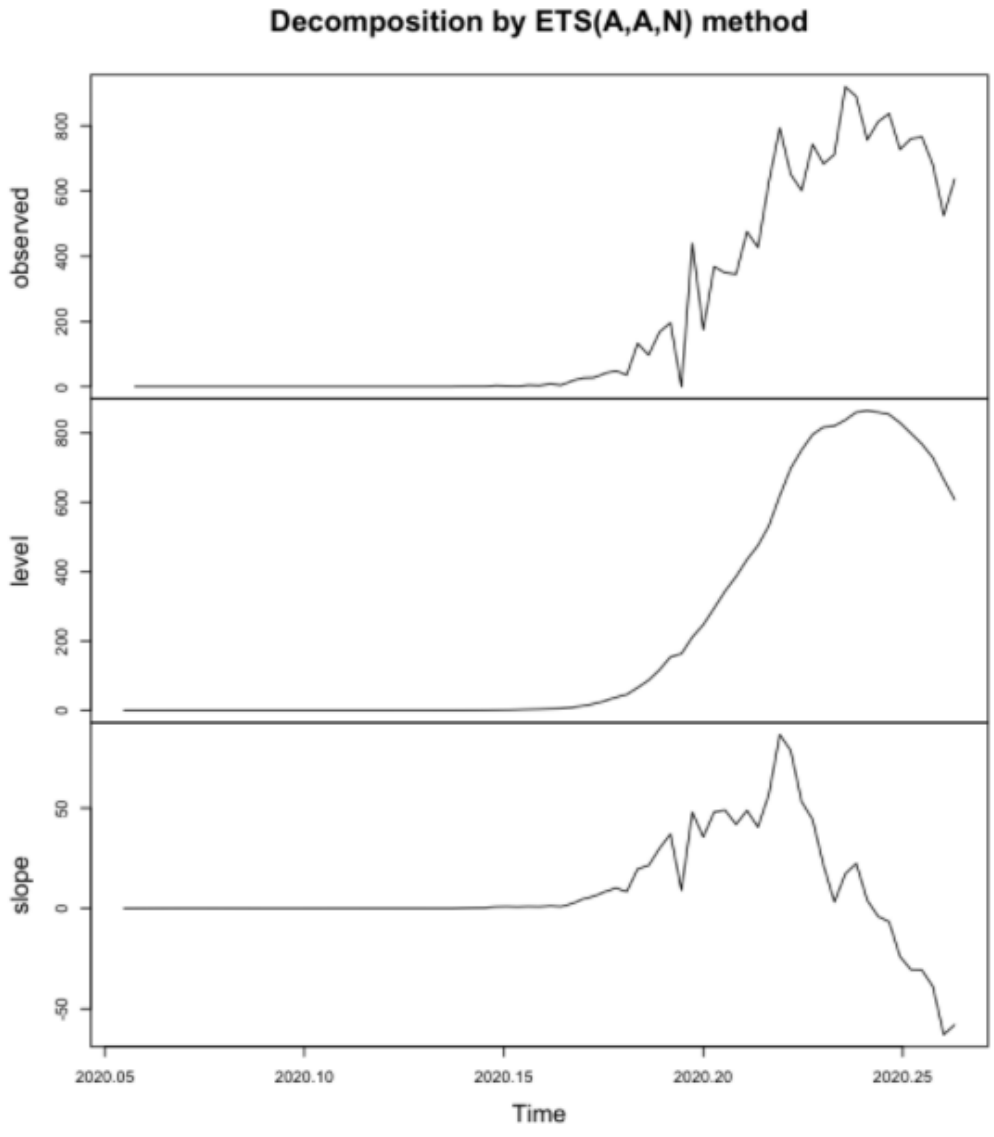
## 2) Holt's linear trend model

Using the mytsfatalities time series I developed the following Holt's linear model.

```
#Generate an additive trend (Holt's linear) ets model

(mytsfatalities.AAN <- ets(mytsfatalities, model = "AAN"))

rmse.ets(mytsfatalities.AAN)
```

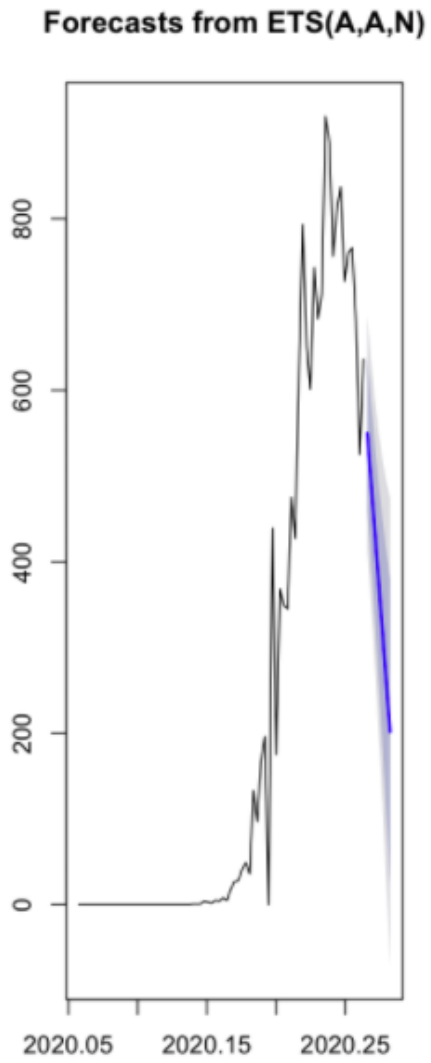
```
ETS(A,A,N)

Call:
 ets(y = mytsfatalities, model = "AAN")

  Smoothing parameters:
    alpha = 0.1459
    beta  = 0.1459

  Initial states:
    l = -0.038
    b = -0.0341

  sigma:  69.8549

     AIC     AICc      BIC
980.4824 981.3396 992.1361

RMSE =  67.99173
```

```
#Decomposition plot of fitted additive trend ets model

plot(mytsfatalities.AAN)
```



Which gave the following forecasts until April 13th:

```
mytsfatalities.AAN.pred <- forecast(mytsfatalities.AAN, h = 7)

# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsfatalities.AAN.pred)
```




According to this model, the number of new cases per day in Italy is expected to fall sharply from the 7th to the 13th of April.

```
#And print the final new cases per day forecasts for May 7th 2020 for each model:
mytsfatalities.AAN.pred$mean[7]
```

```
201.539355079269
```

On the 13th of April, there should be around 202 deaths according to the model.

### 3) Time series linear model

Again, using the mytsfatalities time series, I have also computed the following linear model:

```
#Create the test dataset time series for fatalities
Italytest2.ts <- ts(test_Italy$diff_fatalities,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

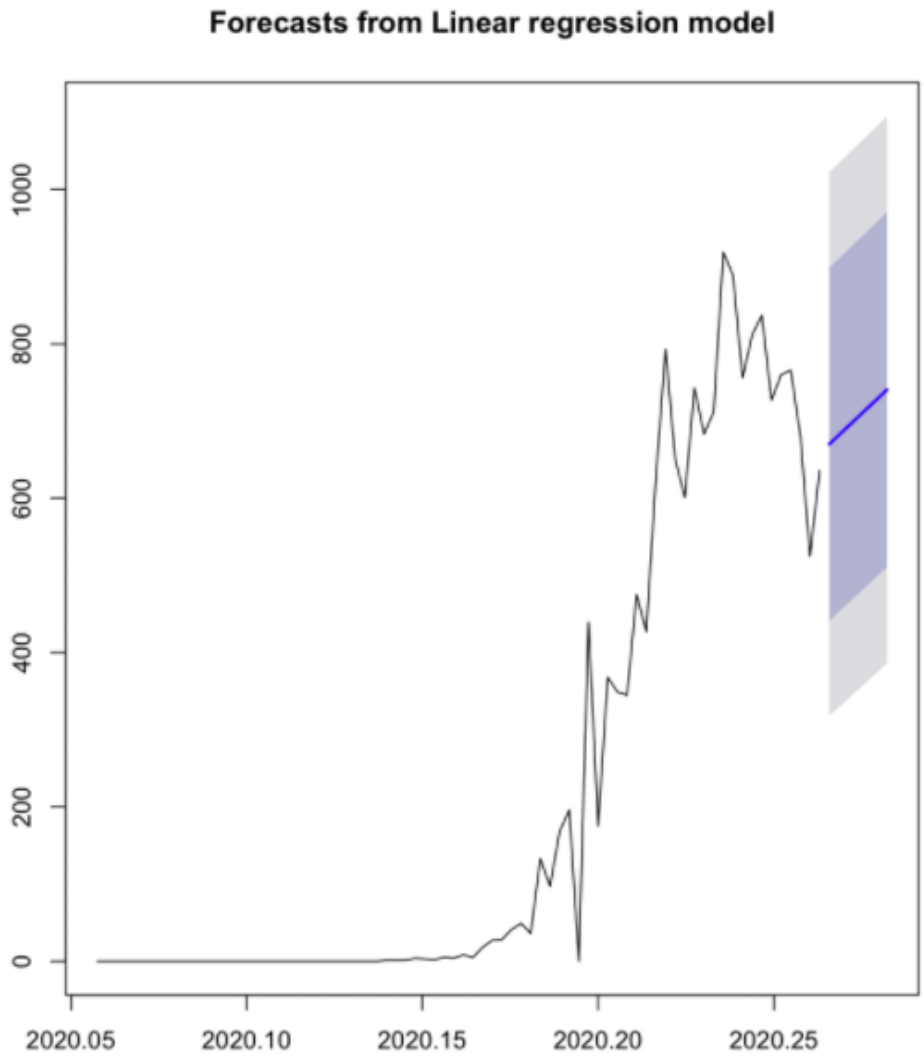
```
#Generate a tslm model
tslm_model_difffatalitiesItaly <- tslm(mytsfatalities~ trend)
tslm_model_difffatalitiesItaly.pred <- forecast(tslm_model_difffatalitiesItaly, h = 7)
```

```
tslm_model_difffatalitiesItaly
```

```
Call:
tslm(formula = mytsfatalities ~ trend)

Coefficients:
(Intercept)        trend
    -235.31        11.76
```

And made the following forecast for the week of April 7:

```
plot(tslm_model_difffatalitiesItaly.pred)
```



I tested the models and their accuracy, if we refer to the RMSE error rate, it seems again that the hold's linear trend model (AAN) model is the best fit to the training dataset but that the exponential smoothing model (ANN) better fit the test dataset.

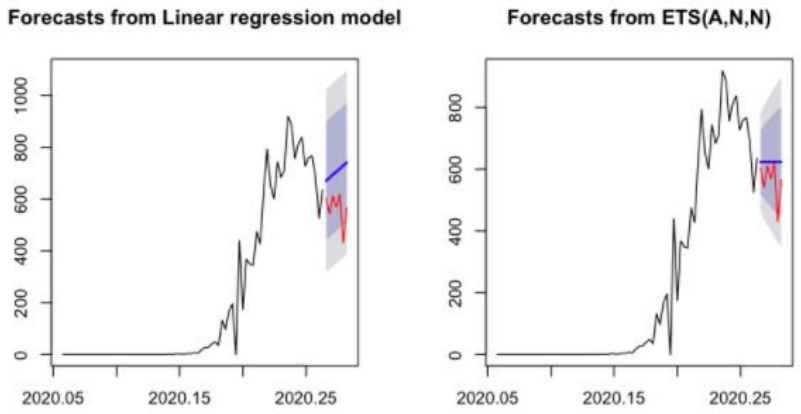
```
# Evaluate the performance of the model
accuracy(tslm_model_difffatalitiesItaly.pred, Italytest2.ts)

accuracy(mytsfatalities.ANN.pred, Italytest2.ts)

accuracy(mytsfatalities.AAN.pred, Italytest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | -6.335963e-15 | 169.9328 | 146.0583 | NaN | Inf | NaN | 0.85011080 | NA |
| Test set | -1.422614e+02 | 159.5435 | 142.2614 | -27.15274 | 27.15274 | NaN | -0.02173393 | 1.555295 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 14.17581 | 79.35176 | 37.60560 | NaN | Inf | NaN | -0.1570501 | NA |
| Test set | -60.06495 | 84.69719 | 60.06495 | -12.14942 | 12.14942 | NaN | -0.3570856 | 0.7840892 |

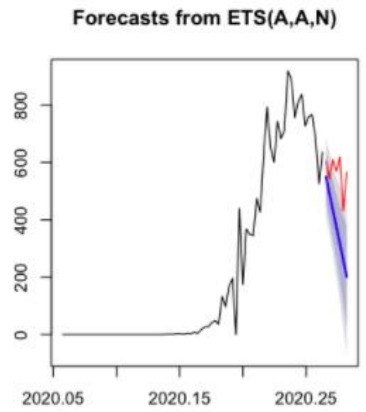
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | -5.229916 | 67.99173 | 35.6246 | NaN | Inf | NaN | -0.09313776 | NA |
| Test set | 187.561846 | 216.30039 | 187.5618 | 33.4565 | 33.4565 | NaN | 0.19469444 | 2.373198 |

Here is the plot of the validation forecasts against the actual validation data:









## 4) Ensemble Model --- Stacking

Here again the first step is to split the training dataset into a smaller training set composed of the training set without the last two weeks and a validation set composed of those 2 weeks.

```
#Step 1. Split the training set into a smaller training set and a validation set.

train_Italy$Date <- as.Date(train_Italy$Date, "%Y-%m-%d")

smalltrainItaly.df <- train_Italy[which(train_Italy$Date <= as.Date("2020-3-24") ),]

validItaly.df <- train_Italy[which(train_Italy$Date >= as.Date("2020-3-24") ),]
```

Then, the second step was to train my models in the small training set in order to generate predictions in the validation set. Again I chose to use the Exponential smoothing model and the Holt's linear trend model for my stacker model because according to their RMSE scores they are the two models which better fit the training dataset.

```
#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainItalydifffatalities.df <- ts(smalltrainItaly.df$diff_fatalities,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainItalydifffatalities.df.ANN <- ets(myts_smalltrainItalydifffatalities.df, model = "ANN"))
myts_smalltrainItalydifffatalities.df.ANN.pred <- forecast(myts_smalltrainItalydifffatalities.df.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainItalydifffatalities.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.5624

  Initial states:
    l = -0.1301

  sigma:  76.8266

     AIC     AICc      BIC
812.0205 812.4272 818.4499
```

```
#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainItalydifffatalities.df.AAN <- ets(myts_smalltrainItalydifffatalities.df, model = "AAN"))
myts_smalltrainItalydifffatalities.df.AAN.pred <- forecast(myts_smalltrainItalydifffatalities.df.AAN , h = 14)
```

```
ETS(A,A,N)

Call:
 ets(y = myts_smalltrainItalydifffatalities.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1034
    beta  = 0.1034

  Initial states:
    l = 0.1843
    b = -0.0733

  sigma:  61.6949

     AIC     AICc      BIC
786.2822 787.3348 796.9979
```

The third step was to fit a model to the predictions generated in step 2

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.df <- data.frame(Date = validItaly.df$Date, diff_fatalities = validItaly.df$diff_fatalities,
                         myts_smalltrainItalydifffatalities.df.ANN.pred = myts_smalltrainItalydifffatalities.df.ANN.pre
                         myts_smalltrainItalydifffatalities.df.AAN.pred =
                         myts_smalltrainItalydifffatalities.df.AAN.pred)



library(dplyr)

stacker.df <- stacker.df[,c("Date","diff_fatalities","myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast",
                            "myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast")]

head(stacker.df)
```

| | Date | diff_fatalities | myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast | myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 743 | 892.9591 | 846.9981 |
| **2020.2329** | 2020-03-25 | 683 | 892.9591 | 899.5975 |
| **2020.2356** | 2020-03-26 | 712 | 892.9591 | 952.1970 |
| **2020.2384** | 2020-03-27 | 919 | 892.9591 | 1004.7964 |
| **2020.2411** | 2020-03-28 | 889 | 892.9591 | 1057.3958 |
| **2020.2438** | 2020-03-29 | 756 | 892.9591 | 1109.9952 |

I chose to do 3 stacker models again: The first one is a random forest model, the second one a regression tree model and the third one a XGBoost model.

```
library(rpart)

library(randomForest)

#Fit a random forest to the predictions as stacker model 1
stackerModel_1.fatalities <- randomForest(stacker.df$diff_fatalities ~ stacker.df$myts_smalltrainItalydifffatalities.df
                        + stacker.df$myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast,
                    data = stacker.df,ntree = 1000, mtry = 2, nodesize = 5, importance = TRUE)



summary(stackerModel_1.fatalities)
```

```
                Length Class  Mode
call               7   -none- call
type               1   -none- character
predicted         14   -none- numeric
mse             1000   -none- numeric
rsq             1000   -none- numeric
oob.times         14   -none- numeric
importance         4   -none- numeric
importanceSD       2   -none- numeric
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            11   -none- list
coefs              0   -none- NULL
y                 14   -none- numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
```

```r
#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModel_2fatalities <- rpart(stacker.df$diff_fatalities ~
                          stacker.df$myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast
                        + stacker.df$myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast
                        , data= stacker.df, method= "anova")


summary(stackerModel_2fatalities)
```

```
Call:
rpart(formula = stacker.df$diff_fatalities ~ stacker.df$myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast
+
    stacker.df$myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast,
    data = stacker.df, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=746.1429, MSE=9597.98
```

```r
stacker.dffatalities.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.dffatalities[,c(-1,-2)])
stacker.y <- as.vector(stacker.dffatalities[,2])
stackerModel_3fatalities <- xgboost(stacker.dffatalities.xgb, stacker.y, nrounds=200,
                        verbose = 0)

summary(stackerModel_3fatalities)
```

```
               Length Class              Mode
handle              1 xgb.Booster.handle externalptr
raw             94126 -none-             raw
niter               1 -none-             numeric
evaluation_log      2 data.table         list
call               13 -none-             call
params              1 -none-             list
callbacks           1 -none-             list
feature_names       2 -none-             character
nfeatures           1 -none-             numeric
```

Finally, the fourth and last step was to re-train my models using the entire training set and to generate predictions on the testing set.

```
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

mytsfatalities <- ts(train_Italy$diff_fatalities,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model
(mytsfatalities.ANN <- ets(mytsfatalities, model = "ANN"))
mytsfatalities.ANN.pred.test <- forecast(mytsfatalities.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalities, model = "ANN")

  Smoothing parameters:
    alpha = 0.5786

  Initial states:
    l = -0.1223

  sigma:  80.4169

     AIC      AICc       BIC
999.9671 1000.3005 1006.9593
```

```
#Generate an additive trend (Holt's linear) ets model
(mytsfatalities.AAN <- ets(mytsfatalities, model = "AAN"))
mytsfatalities.AAN.pred.test <- forecast(mytsfatalities.AAN, h = 14)
```

```
ETS(A,A,N)

Call:
 ets(y = mytsfatalities, model = "AAN")

  Smoothing parameters:
    alpha = 0.1459
    beta  = 0.1459

  Initial states:
    l = -0.038
    b = -0.0341

  sigma:  69.8549

     AIC     AICc      BIC
980.4824 981.3396 992.1361
```

```
predict.variables.fatalities <- data.frame('myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast' =
                                     mytsfatalities.ANN.pred.test,
                                   'myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast' =
                                     mytsfatalities.AAN.pred.test)


# Predict from stacker model 1 --- Random forest
stacker.predict.rt.fatalities1 <- predict(stackerModel_1.fatalities, predict.variables.fatalities )
```

```
# Predict from stacker model 2 --- Regression tree
stacker.predict.rt.fatalities2 <- predict(stackerModel_2fatalities, predict.variables.fatalities )
```

```
# Predict from stacker model_3 --- XGBoost
predict.variables.fatalities.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.fatalities)

colnames(predict.variables.fatalities.xgb) =
c('myts_smalltrainItalydifffatalities.df.ANN.pred.Point.Forecast',
  'myts_smalltrainItalydifffatalities.df.AAN.pred.Point.Forecast')

stacker.predict.rt.fatalities3 <- predict(stackerModel_3fatalities, predict.variables.fatalities.xgb)
```

I obtained the following scores:

```
accuracy(stacker.predict.rt.fatalities1, Italytest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -224.151 | 239.5885 | 224.151 | -41.78362 | 41.78362 | 0.2311971 | 2.310264 |

```
accuracy(stacker.predict.rt.fatalities2, Italytest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -183 | 192.4919 | 183 | -34.27221 | 34.27221 | -0.3570856 | 1.805588 |

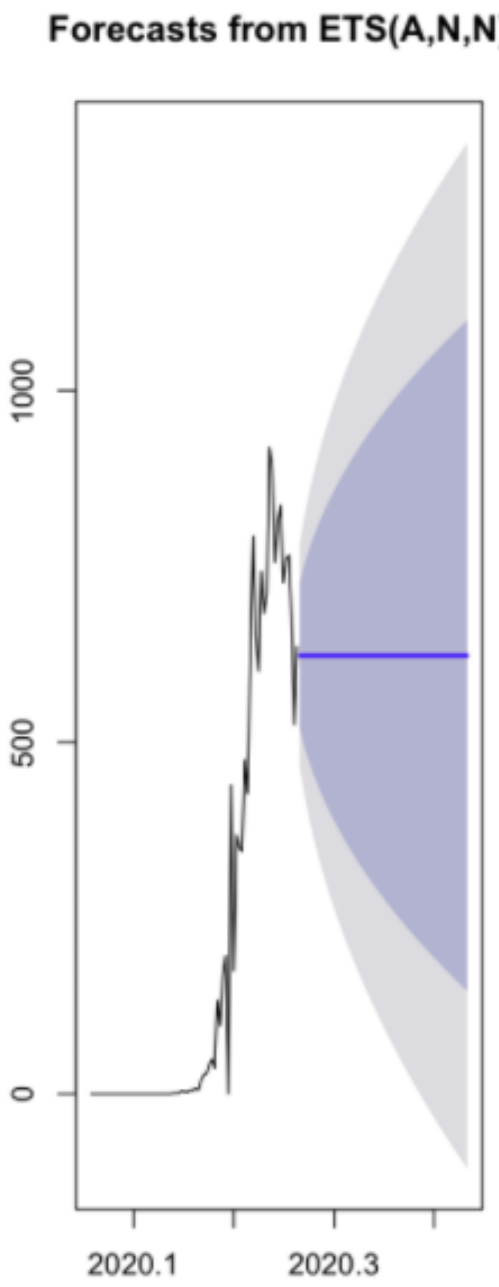
```
accuracy(stacker.predict.rt.fatalities3, Italytest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -179.9581 | 189.5055 | 179.9581 | -33.70645 | 33.70645 | -0.3570856 | 1.777545 |

Both of my ensemble models obtain a lower RMSE score than the exponential smoothing model (ANN) and the hold's trend linear model (AAN). Overall, considering its RMSE score with the training and the test dataset, it seems that the exponential smoothing model is the best one.

If we try to forecast for the next two months using this model, we can see that the number of deaths per day caused by Covid-19 should stay constant and be equal to about 623 in early June.

```r
#Generate forecasts to June 6th 2020  using the above model
mytsfatalities.ANN.pred2months <- forecast(mytsfatalities.ANN, h = 62)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(mytsfatalities.ANN.pred2months)
```




```r
mytsfatalities.ANN.pred2months$mean[62]
```

623.207808576663

## Appendix 1-B: Forecast of the number of cases per day in the UK

### UK FORECAST

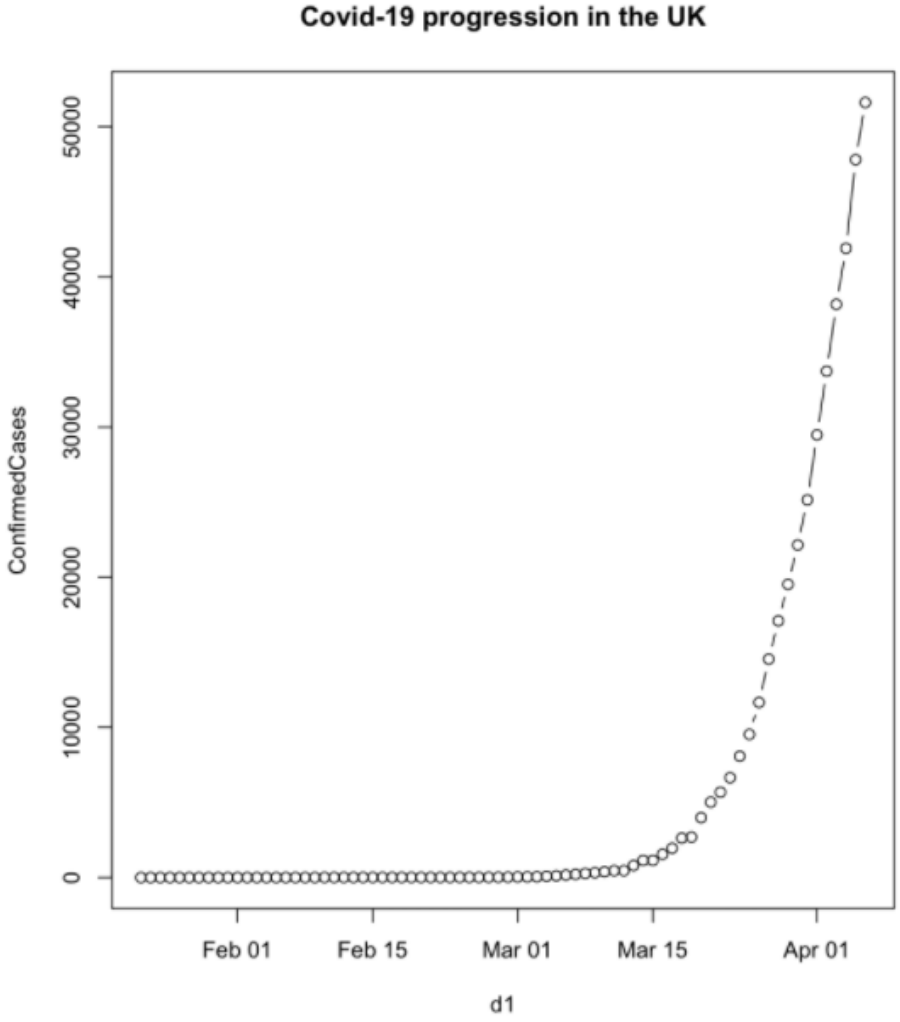
In [126]:
```
train <- read.csv("train.csv")
```

In [127]:
```
train_UK = train[which(train$Country_Region=='United Kingdom' & train$Province_State=='' ),]
```

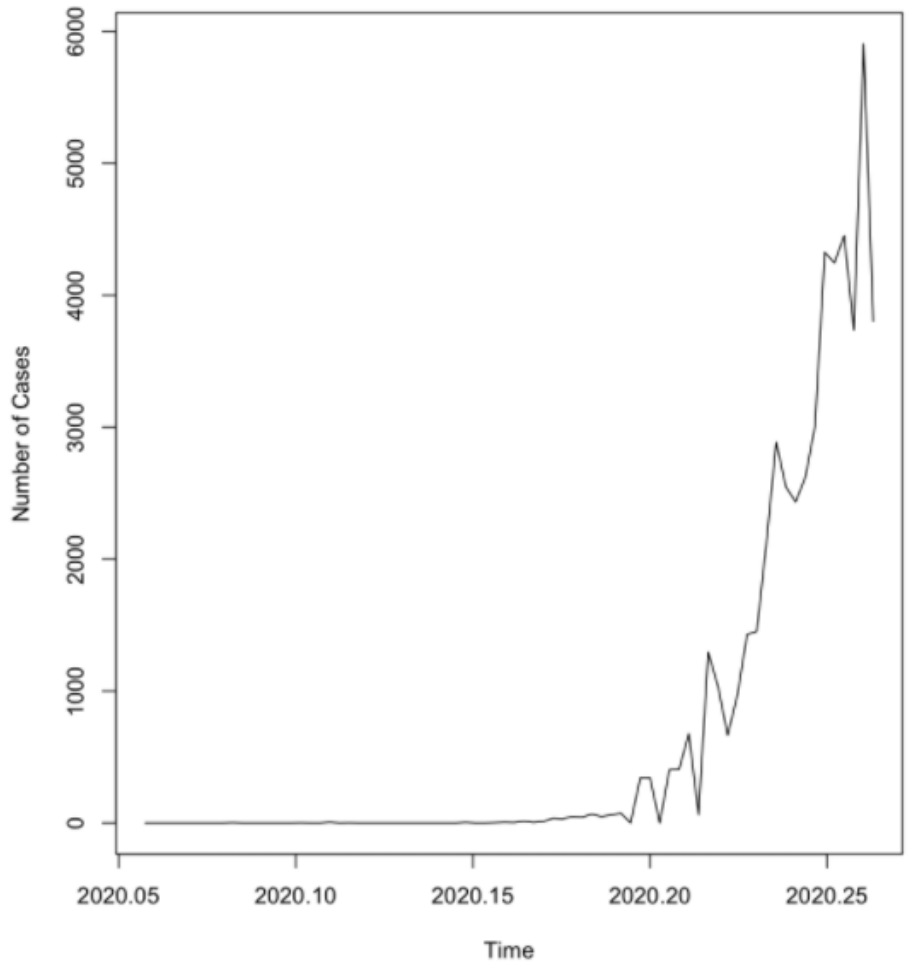
In [128]:
```
#Create a new column with the number of new cases per day in the UK
library(dplyr)

train_UK <- train_UK %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

n <- nrow(train_UK)
train_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff |
|---|---|---|---|---|---|---|---|
| 1 | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 |
| 2 | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 |
| 3 | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 |
| 4 | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 |
| 73 | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 |
| 74 | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 |
| 75 | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 |
| 76 | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 |

In [129]:
```
# A plot with the progression of Covid-19 in the UK

train_UK$Date <- as.Date(train_UK$Date, "%Y-%m-%d")

dl <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")

plot(x = dl, y = train_UK$ConfirmedCases, type = "b", ylab = "ConfirmedCases",main = "Covid-19 progression in the UK")
```




30]:
```
set.seed(25)
mytsUK <- ts(train_UK$diff,
             start = c(2020, as.numeric(format(d1[1], "%j"))),
             frequency = 365)
plot(mytsUK, ylab = "Number of Cases")
```



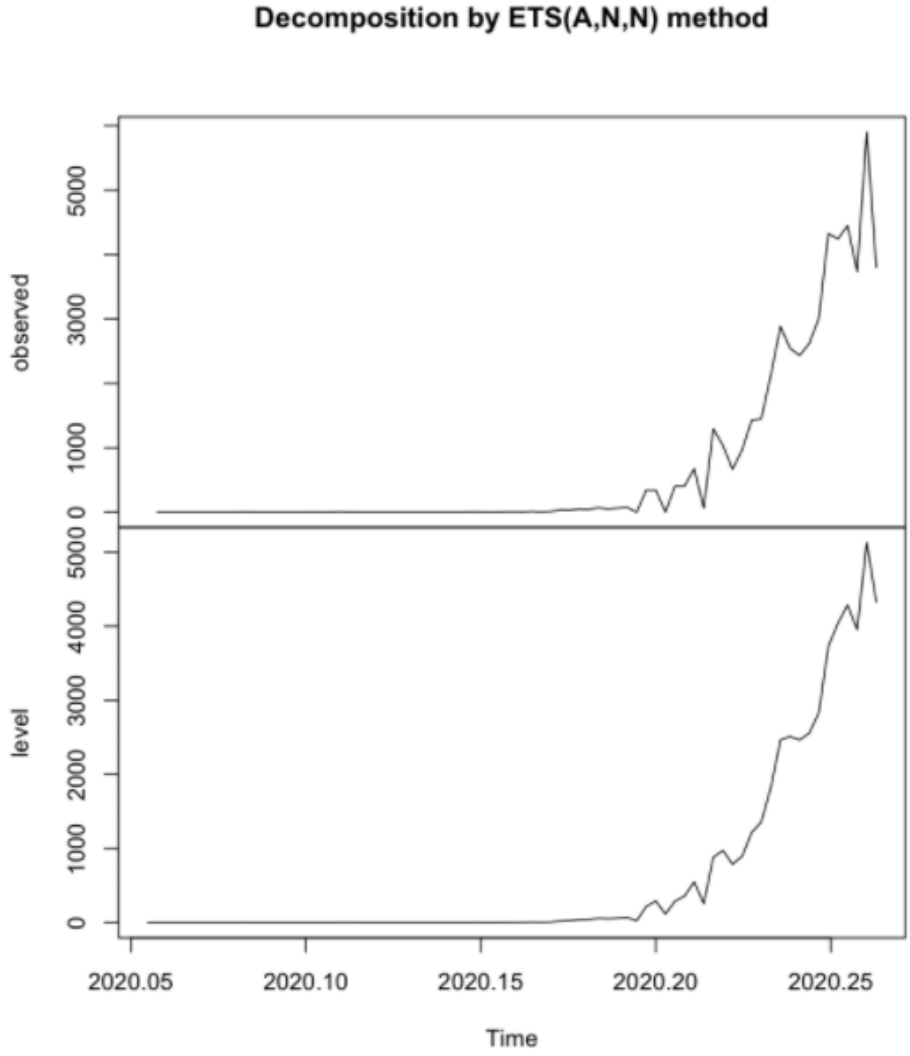

31]:
```
#Generate an exponential smoothing ets model

(mytsUK.ANN <- ets(mytsUK, model = "ANN"))
```

```
ETS(A,N,N)

Call:
 ets(y = mytsUK, model = "ANN")

  Smoothing parameters:
    alpha = 0.6032

  Initial states:
    l = -0.2799

  sigma:  414.2175

     AIC     AICc      BIC
1249.120 1249.454 1256.113
```

2]:
```
#RMSE error for the ets model

rmse.ets(mytsUK.ANN)
```

```
RMSE =  408.7309
```

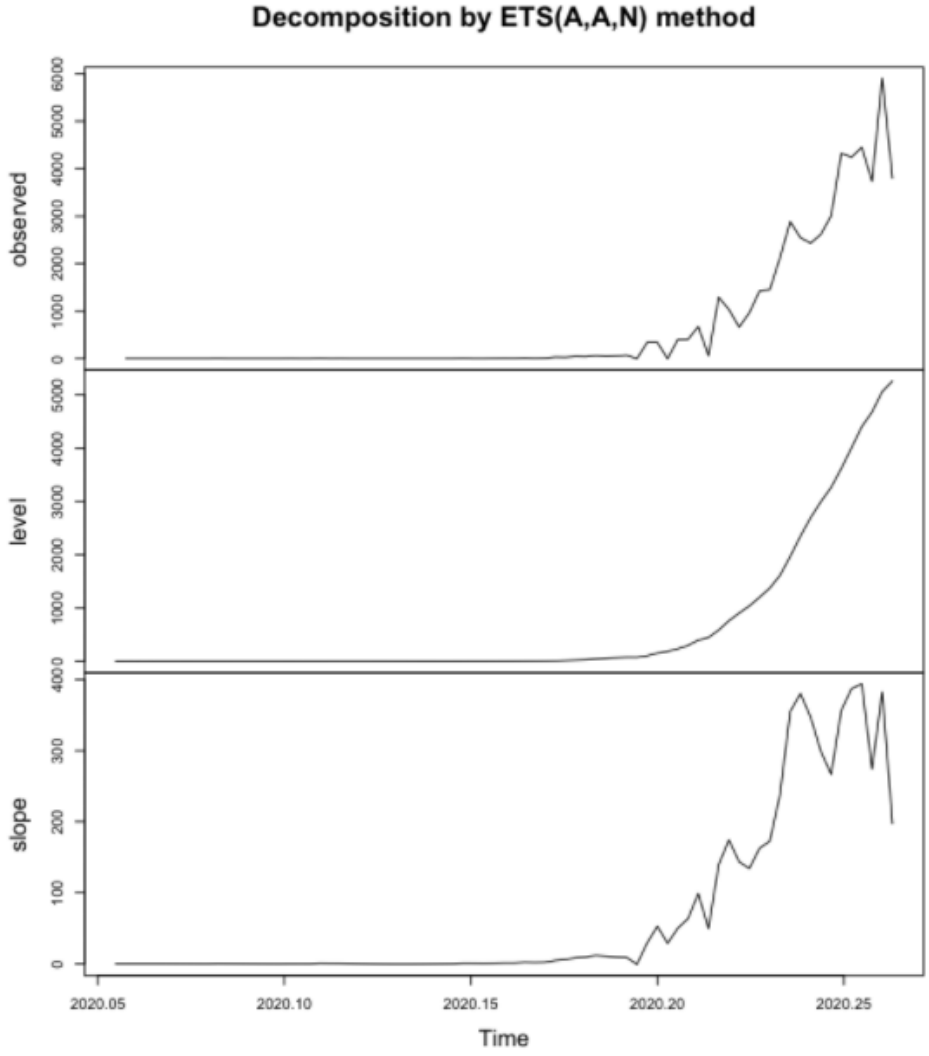
3]:
```
#Decomposition plot of fitted ets model¶
plot(mytsUK.ANN)
```



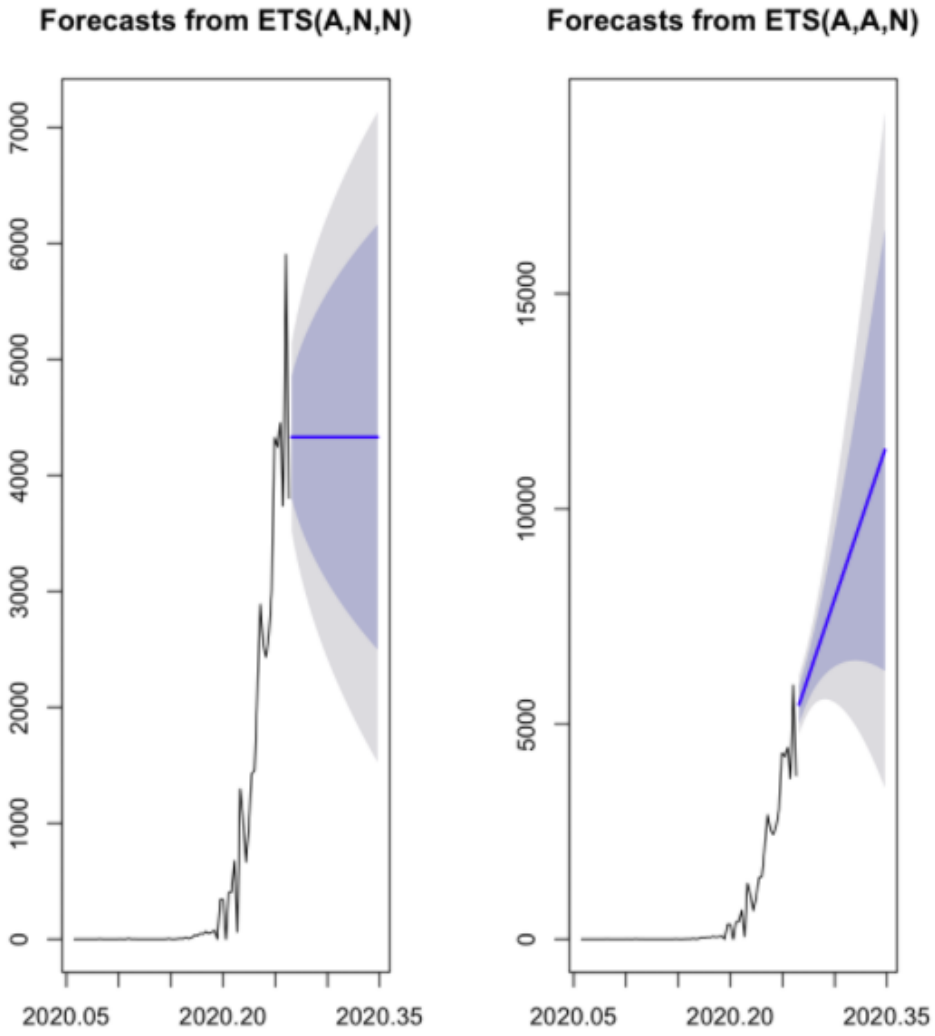

```
: #Generate an additive trend (Holt's linear) ets model

(mytsUK.AAN <- ets(mytsUK, model = "AAN"))

rmse.ets(mytsUK.AAN)
```

```
ETS(A,A,N)

Call:
 ets(y = mytsUK, model = "AAN")

  Smoothing parameters:
    alpha = 0.1132
    beta  = 0.1132

  Initial states:
    l = -0.3188
    b = 0.068

  sigma:  345.6478

     AIC     AICc      BIC
1223.530 1224.388 1235.184

RMSE =  336.4288
```

```
: #Decomposition plot of fitted additive trend ets model

plot(mytsUK.AAN)
```


Decomposition by ETS(A,A,N) method
observed
level
slope
0
1000
2000
3000
4000
5000
6000
100
200
300
2020.05
2020.10
2020.15
2020.20
2020.25
Time

```r
#Generate forecasts to May 7th 2020  using the above models
mytsUK.ANN.pred <- forecast(mytsUK.ANN, h = 31)
mytsUK.AAN.pred <- forecast(mytsUK.AAN, h = 31)

# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsUK.ANN.pred)
plot(mytsUK.AAN.pred)
```



```r
#And print the final new cases per day forecasts for May 7th 2020 for each model:
mytsUK.ANN.pred$mean[31]
mytsUK.AAN.pred$mean[31]
```

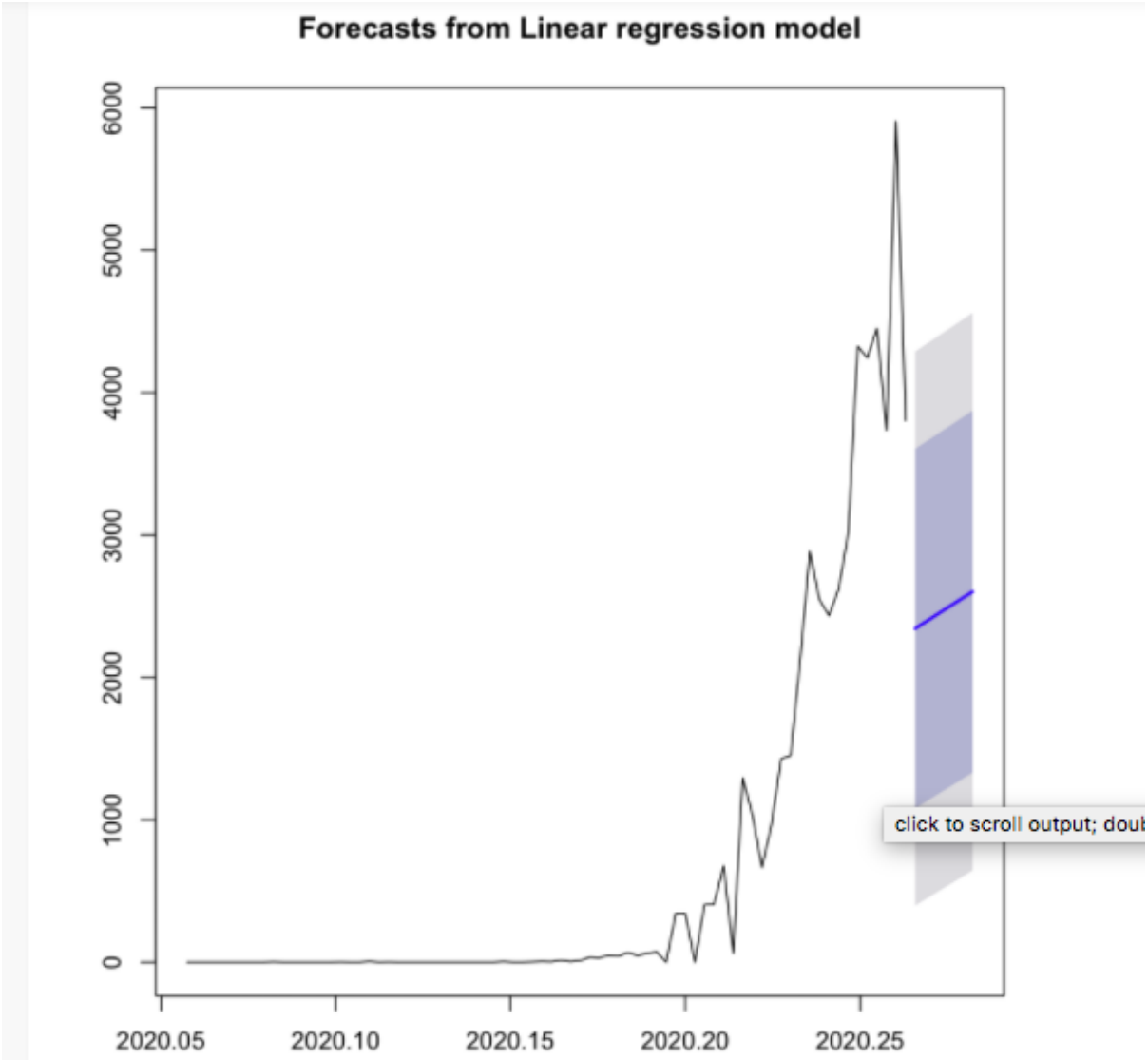
4328.85211721567

11365.4178179248

```r
#Generate a tslm model
tslm_model_diffUK <- tslm(mytsUK~ trend)
tslm_model_diffUK.pred <- forecast(tslm_model_diffUK, h = 7)

tslm_model_diffUK
```

```
Call:
tslm(formula = mytsUK ~ trend)

Coefficients:
(Intercept)        trend
    -984.60        43.21
```

```r
plot(tslm_model_diffUK.pred)
```




```
]: #I want the test data set to contain the Confirmed Cases and Fatalities from the 7th to the 13th of April
   test <- read.csv("test.csv")
   n <- nrow(test)
   test[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 |
| **25976** | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 |
| **25977** | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 |
| **25978** | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 |
| **25979** | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 |

```
#Create a new column with the number of new cases per day
library(dplyr)

test <- test %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

#Create a new column with the number of fatalities per day

test <- test %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))


n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 | 0 | 0 |
| **25976** | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 | 2 | 0 |
| **25977** | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 | 1 | 0 |
| **25978** | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 | 0 | 0 |
| **25979** | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 | 3 | 0 |

```
test$Date <- as.Date(test$Date, "%Y-%m-%d")

#Take values between the 7th and the 13th of April and for country = Italy
test_UK = test[which(test$Country_Region=='United Kingdom' & test$Province_State =='' & test$Date >= as.Date("2020-4-7"

n <- nrow(test_UK)
test_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **25309** | 34733 | | United Kingdom | 2020-04-07 | 55242 | 6159 | 3634 | 786 |
| **25310** | 34734 | | United Kingdom | 2020-04-08 | 60733 | 7097 | 5491 | 938 |
| **25311** | 34735 | | United Kingdom | 2020-04-09 | 65077 | 7978 | 4344 | 881 |
| **25312** | 34736 | | United Kingdom | 2020-04-10 | 73758 | 8958 | 8681 | 980 |
| **25312.1** | 34736 | | United Kingdom | 2020-04-10 | 73758 | 8958 | 8681 | 980 |
| **25313** | 34737 | | United Kingdom | 2020-04-11 | 78991 | 9875 | 5233 | 917 |
| **25314** | 34738 | | United Kingdom | 2020-04-12 | 84279 | 10612 | 5288 | 737 |
| **25315** | 34739 | | United Kingdom | 2020-04-13 | 88621 | 11329 | 4342 | 717 |

```
test_UK

d2 <- seq(from = as.Date("2020-4-7", "%Y-%m-%d"), to = as.Date("2020-4-13", "%Y-%m-%d"), by = "day")

UKtest.ts <- ts(test_UK$diff,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **25309** | 34733 | | United Kingdom | 2020-04-07 | 55242 | 6159 | 3634 | 786 |
| **25310** | 34734 | | United Kingdom | 2020-04-08 | 60733 | 7097 | 5491 | 938 |
| **25311** | 34735 | | United Kingdom | 2020-04-09 | 65077 | 7978 | 4344 | 881 |
| **25312** | 34736 | | United Kingdom | 2020-04-10 | 73758 | 8958 | 8681 | 980 |
| **25313** | 34737 | | United Kingdom | 2020-04-11 | 78991 | 9875 | 5233 | 917 |
| **25314** | 34738 | | United Kingdom | 2020-04-12 | 84279 | 10612 | 5288 | 737 |
| **25315** | 34739 | | United Kingdom | 2020-04-13 | 88621 | 11329 | 4342 | 717 |

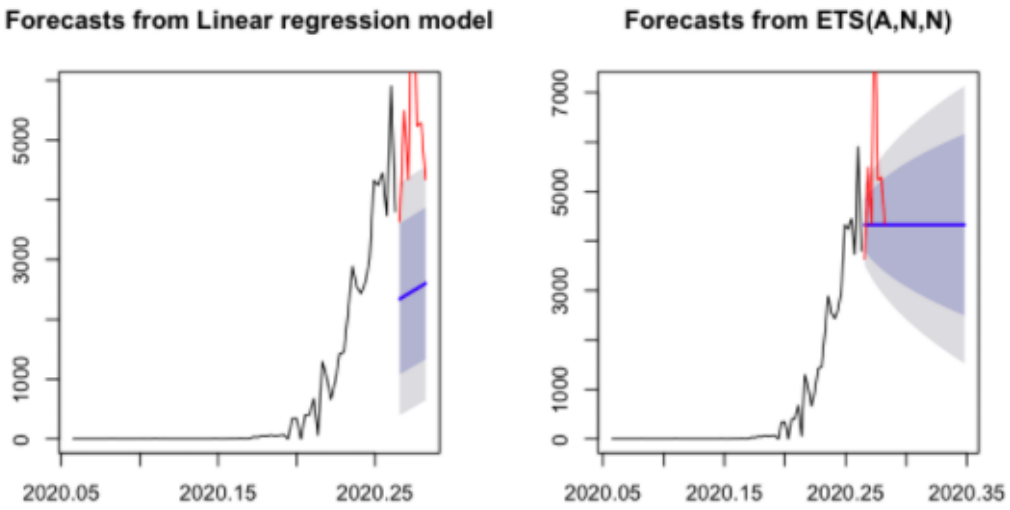
```
# Evaluate the performance of the model
accuracy(tslm_model_diffUK.pred, UKtest.ts)

accuracy(mytsUK.ANN.pred, UKtest.ts)

accuracy(mytsUK.AAN.pred, UKtest.ts)
```

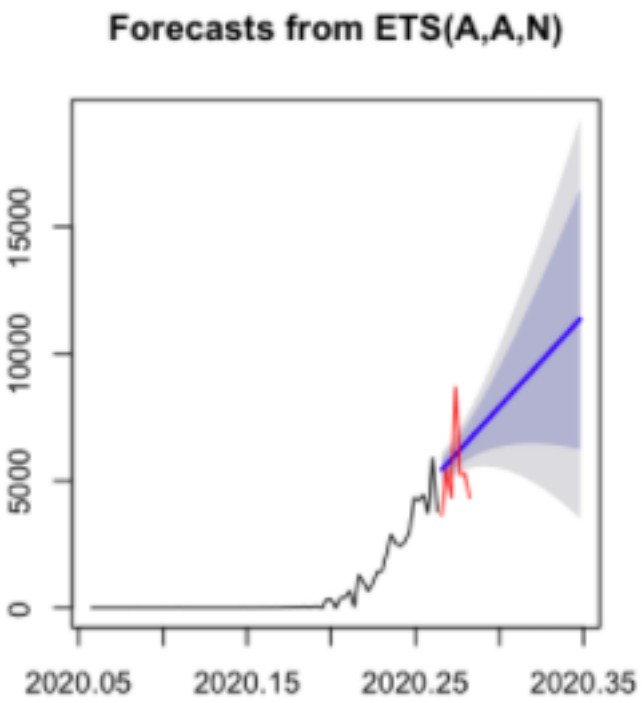
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 6.210237e-14 | 938.2649 | 738.808 | NaN | Inf | NaN | 0.8611603 | NA |
| **Test set** | 2.815226e+03 | 3193.3843 | 2815.226 | 50.1854 | 50.1854 | NaN | -0.2507913 | 1.503803 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 94.4357 | 408.7309 | 177.154 | NaN | Inf | NaN | -0.1938854 | NA |
| **Test set** | 958.7193 | 1793.3388 | 1157.248 | 12.60649 | 18.06959 | NaN | -0.2435489 | 0.8782954 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 22.92113 | 336.4288 | 149.8998 | NaN | Inf | NaN | -0.2263572 | NA |
| **Test set** | -753.56575 | 1695.1238 | 1507.8123 | -21.51218 | 30.20065 | NaN | -0.2331874 | 0.6825993 |

```
#Plot and compare the validation forecasts against the actual validation data
options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8
par(mfrow = c(2, 2))

plot(tslm_model_diffUK.pred)
lines(UKtest.ts, col="red")  # plots the actual validation data

plot(mytsUK.ANN.pred)
lines(UKtest.ts, col="red")

plot(mytsUK.AAN.pred)
lines(UKtest.ts, col="red")
```




Forecasts from ETS(A,A,N)
0
5000
10000
15000
2020.05
2020.15
2020.25
2020.35

Ensemble Model --- Stacking

```
#Step 1. Split the training set into a smaller training set and a validation set.

train_UK$Date <- as.Date(train_UK$Date, "%Y-%m-%d")

smalltrainUK.df <- train_UK[which(train_UK$Date <= as.Date("2020-3-24") ),]

validUK.df <- train_UK[which(train_UK$Date >= as.Date("2020-3-24") ),]
```

```
#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainUKdiff.df <- ts(smalltrainUK.df$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainUKdiff.df.ANN <- ets(myts_smalltrainUKdiff.df, model = "ANN"))
myts_smalltrainUKdiff.df.ANN.pred <- forecast(myts_smalltrainUKdiff.df.ANN, h = 14)



#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainUKdiff.df.AAN <- ets(myts_smalltrainUKdiff.df, model = "AAN"))
myts_smalltrainUKdiff.df.AAN.pred <- forecast(myts_smalltrainUKdiff.df.AAN , h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainUKdiff.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.4562

  Initial states:
    l = -0.0879

  sigma:  184.0034

     AIC     AICc      BIC
922.0693 922.4761 928.4987

ETS(A,A,N)

Call:
 ets(y = myts_smalltrainUKdiff.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1214
    beta  = 0.1214

  Initial states:
    l = -0.3591
    b = 0.0973

  sigma:  152.5298

     AIC     AICc      BIC
900.3322 901.3848 911.0479
```

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.df.diffUK <- data.frame(Date = validUK.df$Date, diff = validUK.df$diff,
                        myts_smalltrainUKdiff.df.ANN.pred = myts_smalltrainUKdiff.df.ANN.pred,
                        myts_smalltrainUKdiff.df.AAN.pred = myts_smalltrainUKdiff.df.AAN.pred)



library(dplyr)

stacker.df.diffUK <- stacker.df.diffUK[,c("Date","diff","myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast",
                          "myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast")]

head(stacker.df.diffUK)
```

| | Date | diff | myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast | myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 1427 | 1122.1 | 1374.836 |
| **2020.2329** | 2020-03-25 | 1452 | 1122.1 | 1535.537 |
| **2020.2356** | 2020-03-26 | 2129 | 1122.1 | 1696.237 |
| **2020.2384** | 2020-03-27 | 2885 | 1122.1 | 1856.937 |
| **2020.2411** | 2020-03-28 | 2546 | 1122.1 | 2017.637 |
| **2020.2438** | 2020-03-29 | 2433 | 1122.1 | 2178.337 |

```
library(randomForest)

Fit a random forest to the predictions as stacker model 1
tackerModelUK_1 <- randomForest(stacker.df.diffUK$diff ~ stacker.df.diffUK$myts_smalltrainUKdiff.df.ANN.pred.Point.Fore
                      + stacker.df.diffUK$myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast,
                  data = stacker.df.diffUK,  nodesize = 5, importance = TRUE)


Fit a Regression tree to the predictions as stacker model_2
ibrary(rpart)
tackerModelUK_2 <- rpart(stacker.df.diffUK$diff ~ stacker.df.diffUK$myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast
                      + stacker.df.diffUK$myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast
                      , data= stacker.df.diffUK, method= "anova")


Fit a XGBoost model to the predictions as stacker model_3
ibrary(xgboost)

ibrary(foreach)
ibrary(parallel)
ibrary(Matrix)
ibrary(readr)
ibrary(caret)

tacker.df.diffUK.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.df.diffUK[,c(-1,-2)])
tacker.diffUK.y <- as.vector(stacker.df.diffUK[,2])

tackerModelUK_3 <- xgboost(stacker.df.diffUK.xgb, stacker.diffUK.y, nrounds=200,
                    verbose = 0)



ummary(stackerModelUK_1)

ummary(stackerModelUK_2)

ummary(stackerModelUK_3)
```

```
                Length Class  Mode
```

```
                Length Class  Mode
call              5    -none- call
type              1    -none- character
predicted        14    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times        14    -none- numeric
importance        4    -none- numeric
importanceSD      2    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y                14    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call

Call:
rpart(formula = stacker.df.diffUK$diff ~ stacker.df.diffUK$myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast +
    stacker.df.diffUK$myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast,
    data = stacker.df.diffUK, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=3211.286, MSE=1478872


                Length Class              Mode
handle               1 xgb.Booster.handle externalptr
raw             101182 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                13 -none-             call
params               1 -none-             list
callbacks            1 -none-             list
feature_names        2 -none-             character
nfeatures            1 -none-             numeric
```

```r
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

mytsdiffUK <- ts(train_UK$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model
(mytsdiffUK.ANN <- ets(mytsdiffUK, model = "ANN"))
mytsdiffUK.ANN.pred.test <- forecast(mytsdiffUK.ANN, h = 14)


#Generate an additive trend (Holt's linear) ets model
(mytsdiffUK.AAN <- ets(mytsdiffUK, model = "AAN"))
mytsdiffUK.AAN.pred.test <- forecast(mytsdiffUK.AAN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsdiffUK, model = "ANN")

  Smoothing parameters:
    alpha = 0.6032

  Initial states:
    l = -0.2799

  sigma:  414.2175

     AIC     AICc      BIC
1249.120 1249.454 1256.113

ETS(A,A,N)

Call:
 ets(y = mytsdiffUK, model = "AAN")

  Smoothing parameters:
    alpha = 0.1132
    beta  = 0.1132

  Initial states:
    l = -0.3188
    b = 0.068

  sigma:  345.6478

     AIC     AICc      BIC
1223.530 1224.388 1235.184
```

```
#Create a variable with the predicted values
predict.variables.diffUK <- data.frame('myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast' = mytsdiffUK.ANN.pred.test,
                                       'myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast' = mytsdiffUK.AAN.pred.test)


#Only select the forecast values
predict.variables.diffUK <- predict.variables.diffUK%>% select(1,6)

predict.variables.diffUK
```

| | myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast.Point.Forecast | myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast.Point.Forecast |
|---|---|---|
| 2020.2658 | 4328.852 | 5449.550 |
| 2020.2685 | 4328.852 | 5646.746 |
| 2020.2712 | 4328.852 | 5843.942 |
| 2020.2740 | 4328.852 | 6041.137 |
| 2020.2767 | 4328.852 | 6238.333 |
| 2020.2795 | 4328.852 | 6435.528 |
| 2020.2822 | 4328.852 | 6632.724 |
| 2020.2849 | 4328.852 | 6829.919 |
| 2020.2877 | 4328.852 | 7027.115 |
| 2020.2904 | 4328.852 | 7224.311 |
| 2020.2932 | 4328.852 | 7421.506 |
| 2020.2959 | 4328.852 | 7618.702 |
| 2020.2986 | 4328.852 | 7815.897 |
| 2020.3014 | 4328.852 | 8013.093 |

```
# Predict from stacker model_1 --- random forest
stacker.predict.diffUK.rt1 <- predict(stackerModelUK_1, predict.variables.diffUK )

# Predict from stacker model_2 --- regression tree
stacker.predict.diffUK.rt2 <- predict(stackerModelUK_2, predict.variables.diffUK )

# Predict from stacker model_3 --- XGBoost
predict.variables.diffUK.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.diffUK)

colnames(predict.variables.diffUK.xgb) =
c('myts_smalltrainUKdiff.df.ANN.pred.Point.Forecast','myts_smalltrainUKdiff.df.AAN.pred.Point.Forecast')

stacker.predict.diffUK.rt3 <- predict(stackerModelUK_3, predict.variables.diffUK.xgb)
```

```
accuracy(stacker.predict.diffUK.rt1, UKtest.ts)

accuracy(stacker.predict.diffUK.rt2, UKtest.ts)

accuracy(stacker.predict.diffUK.rt3, UKtest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 2520.535 | 2925.597 | 2520.535 | 44.31913 | 44.31913 | -0.2681637 | 1.392481 |

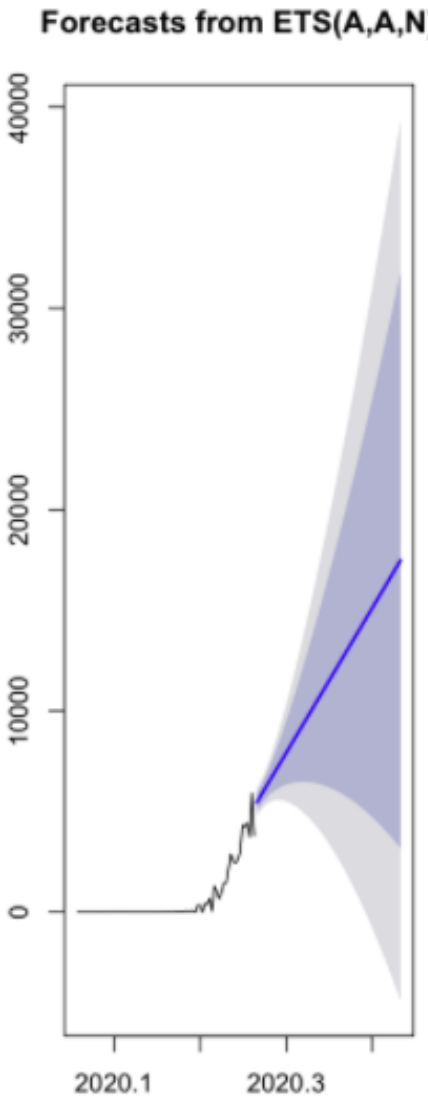
| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 2076.286 | 2570.58 | 2076.286 | 35.1686 | 35.1686 | -0.2435499 | 1.23615 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 1485.571 | 2122.226 | 1533.571 | 23.2429 | 24.56376 | -0.2435499 | 1.037537 |

Overall, considering RMSE scores for both the training and test dataset, the holt's trend linear model (AAN) is the best model.

If we try to forecast for the next two months using this model, we can see that the number of cases per day caused by Covid-19 should sharply increase and be equal to about 17478 in early June:

```
#Generate forecasts to June 6th 2020  using the above model
mytsUK.AAN.pred2months <- forecast(mytsUK.AAN, h = 62)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(mytsUK.AAN.pred2months)
```




```
mytsUK.AAN.pred2months$mean[62]
```

```
17478.4807794676
```

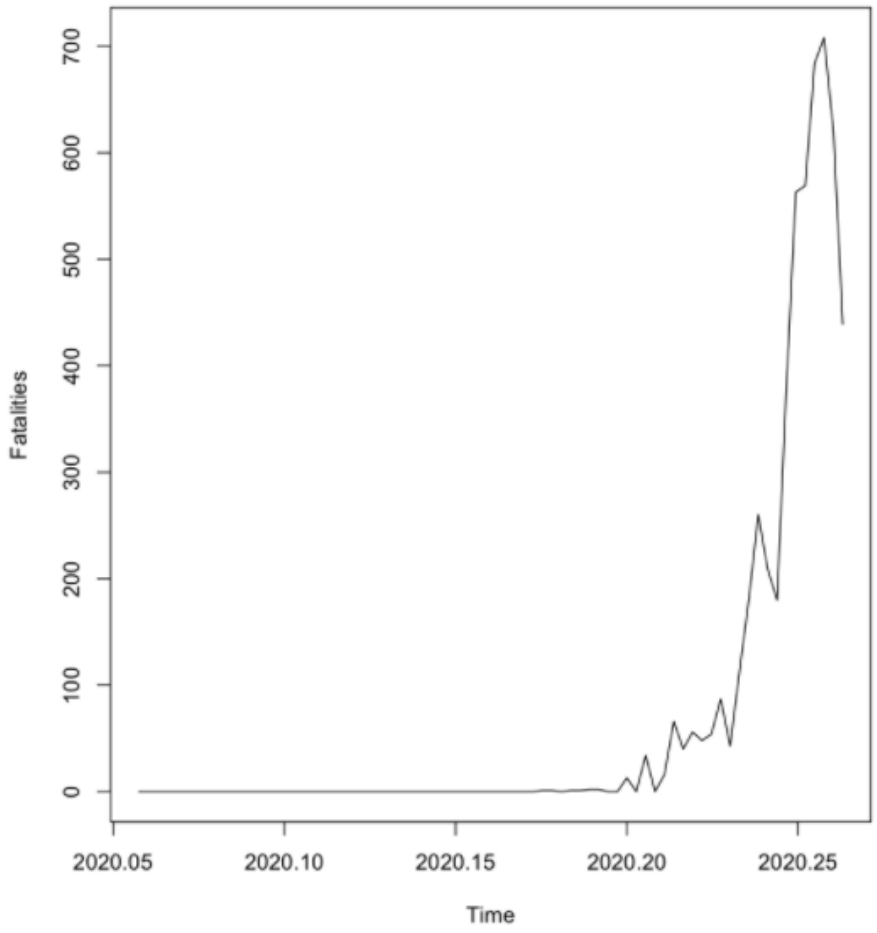
## Appendix 1-C: Forecast of the number of fatalities per day in the UK

### UK forecast fatalities per day

In [138]:
```
#Create a new column with the number of new cases per day in the Uk
library(dplyr)

train_UK <- train_UK %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))

n <- nrow(train_UK)
train_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 1 | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 | 0 |
| 4 | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 | 0 |
| 73 | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 | 684 |
| 74 | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 | 708 |
| 75 | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 | 621 |
| 76 | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 | 439 |

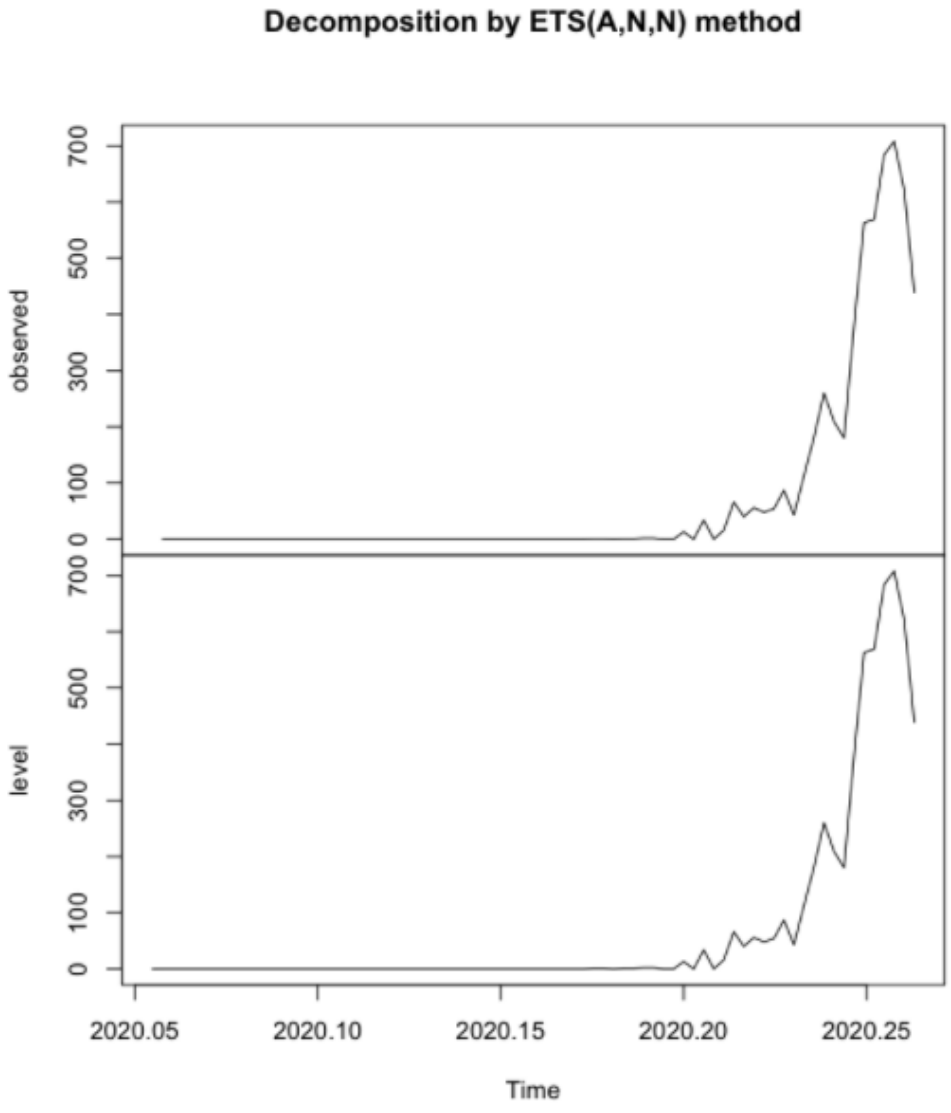
In [223]:
```
d1 <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")
set.seed(25)
mytsfatalitiesUK <- ts(train_UK$diff_fatalities,
            start = c(2020, as.numeric(format(d1[1], "%j"))) ,
            frequency = 365)
plot(mytsfatalitiesUK, ylab = "Fatalities")
```




In [211]:
```
#Generate an exponential smoothing ets model

(mytsfatalitiesUK.ANN <- ets(mytsfatalitiesUK, model = "ANN"))

#RMSE error for the ets model

rmse.ets(mytsfatalitiesUK.ANN)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalitiesUK, model = "ANN")

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = -0.1248

  sigma:  46.0233

     AIC     AICc      BIC
915.1395 915.4728 922.1317

RMSE =  45.41371
```

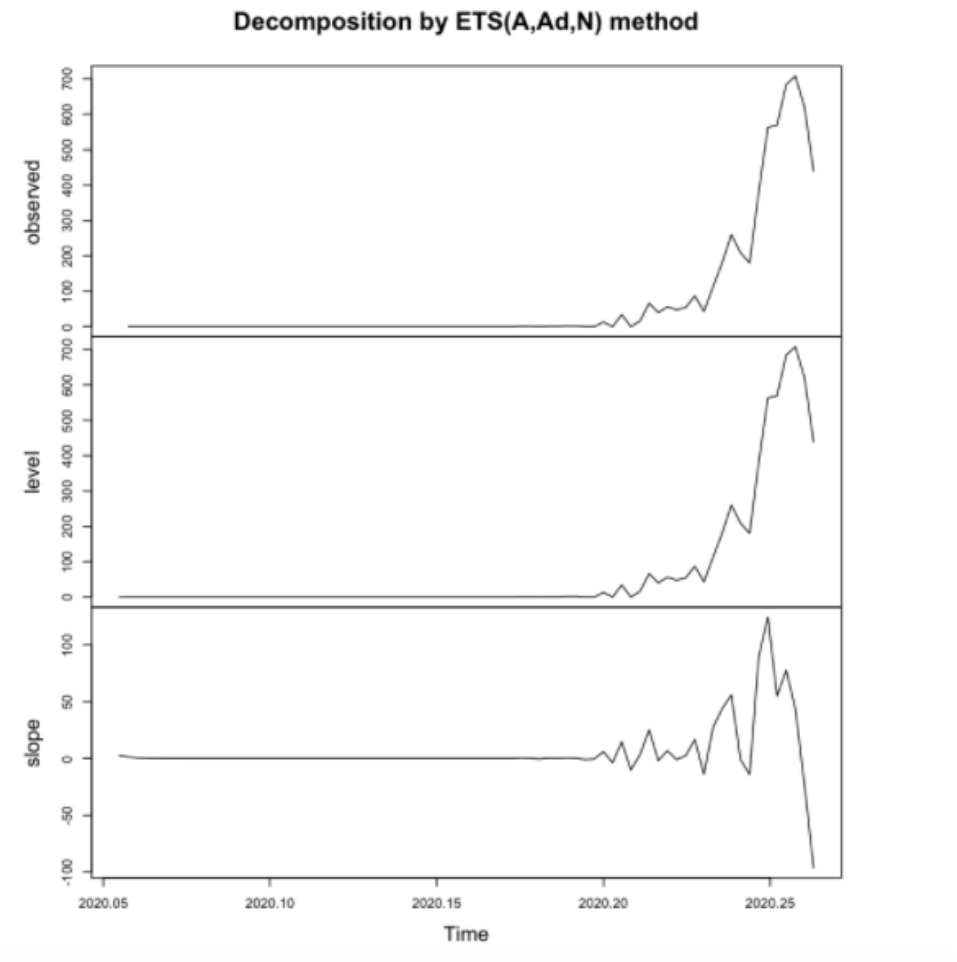
[212]:
```
#Decomposition plot of fitted ets model¶
plot(mytsfatalitiesUK.ANN)
```



```r
#Generate an additive trend (Holt's linear) ets model

(mytsfatalitiesUK.AAN <- ets(mytsfatalitiesUK, model = "AAN"))

rmse.ets(mytsfatalitiesUK.AAN)
```

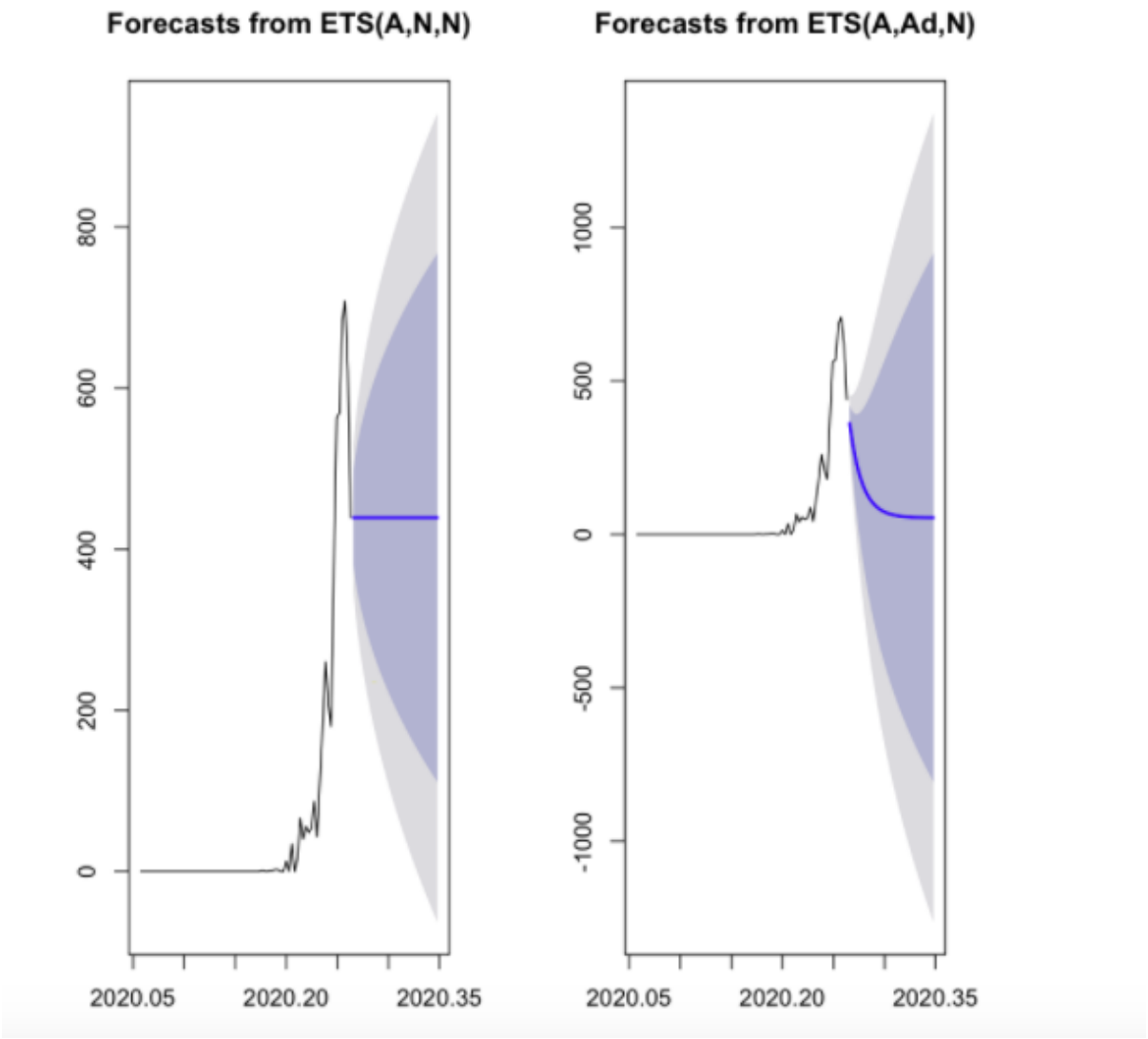
```
ETS(A,Ad,N)

Call:
 ets(y = mytsfatalitiesUK, model = "AAN")

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.476
    phi   = 0.8

  Initial states:
    l = -0.7114
    b = 2.5812

  sigma:  45.7301

     AIC     AICc      BIC
917.0228 918.2402 931.0072

RMSE =  44.20025
```

```r
#Decomposition plot of fitted additive trend ets model

plot(mytsfatalitiesUK.AAN)
```



```r
#Generate forecasts to May 7th 2020  using the above models
mytsfatalitiesUK.ANN.pred <- forecast(mytsfatalitiesUK.ANN, h = 31)
mytsfatalitiesUK.AAN.pred <- forecast(mytsfatalitiesUK.AAN, h = 31)
```

```r
# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsfatalitiesUK.ANN.pred)
plot(mytsfatalitiesUK.AAN.pred)
```

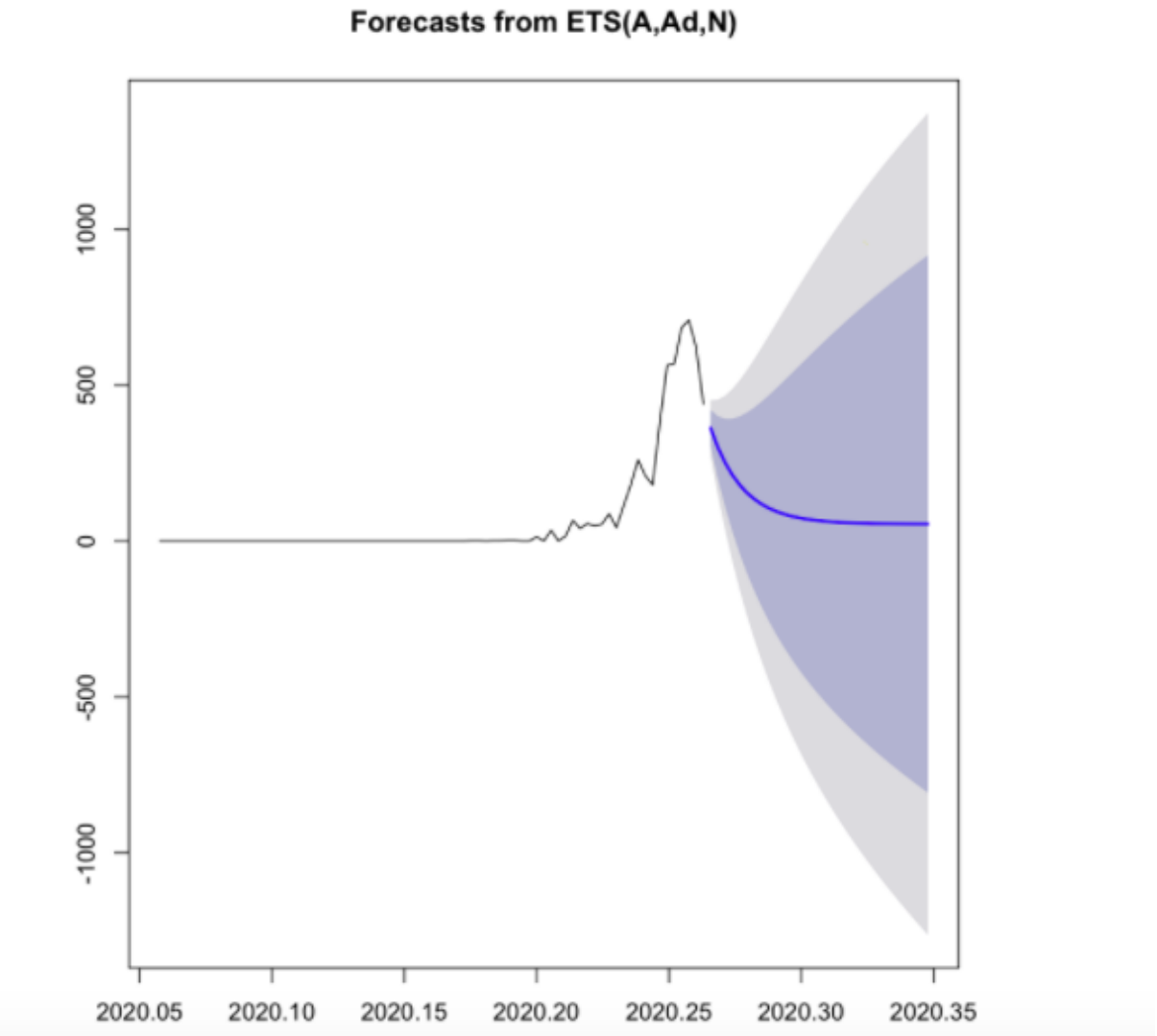


```
#And print the final new cases per day forecasts for May 7th 2020 for each model:
mytsfatalitiesUK.ANN.pred$mean[31]
mytsfatalitiesUK.AAN.pred$mean[31]
```

439.018203919078

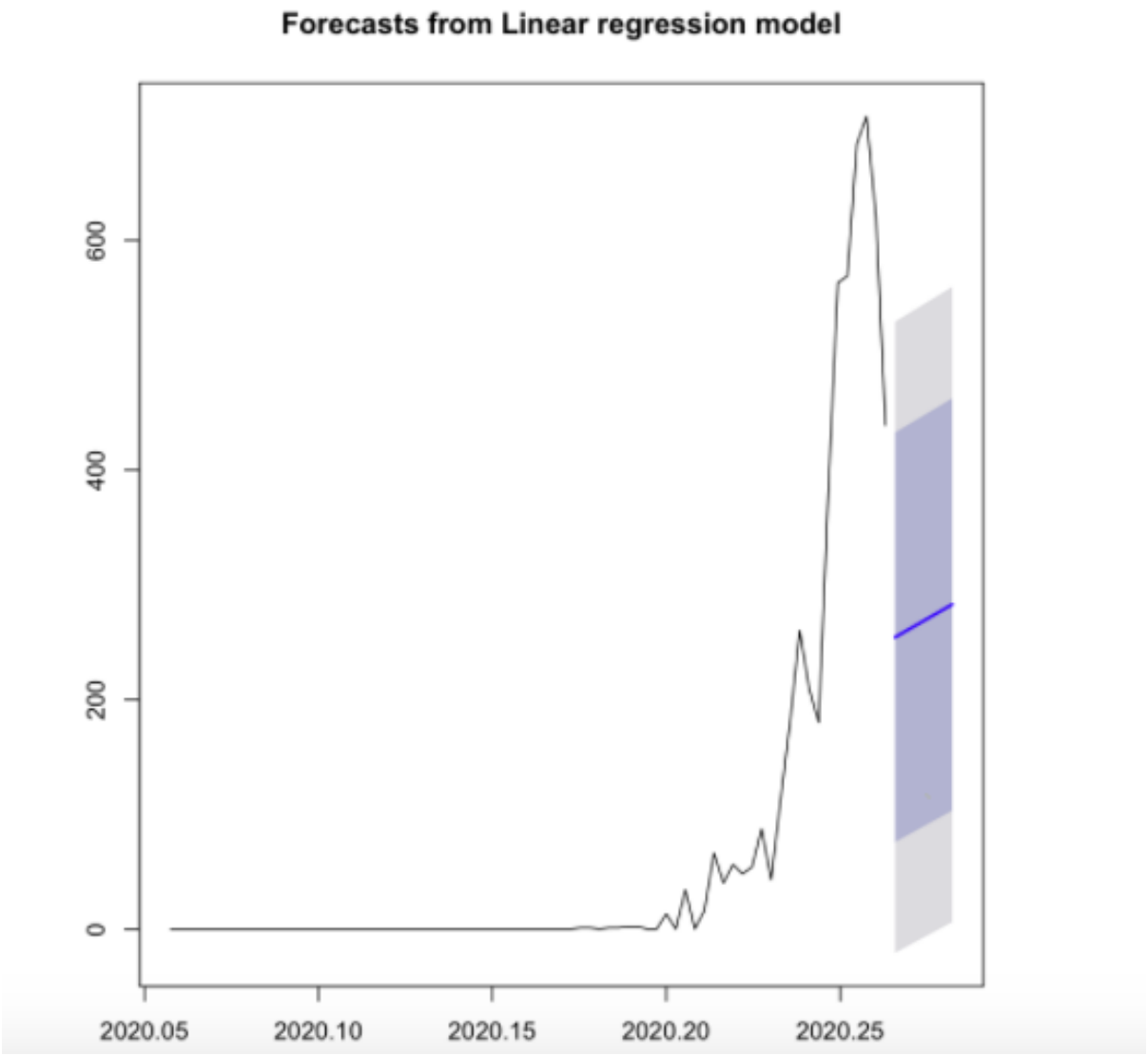
54.3253435600929

```
plot(mytsfatalitiesUK.AAN.pred)
```



```r
#Generate a tslm model
tslm_model_difffatalitiesUK <- tslm(mytsfatalitiesUK~ trend)
tslm_model_difffatalitiesUK.pred <- forecast(tslm_model_difffatalitiesUK, h = 7)

tslm_model_difffatalitiesUK



plot(tslm_model_difffatalitiesUK.pred)
```

```
Call:
tslm(formula = mytsfatalitiesUK ~ trend)

Coefficients:
(Intercept)        trend
    -112.741        4.765
```



```
#Create the test dataset time series for fatalities
UKtest2.ts <- ts(test_UK$diff_fatalities,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

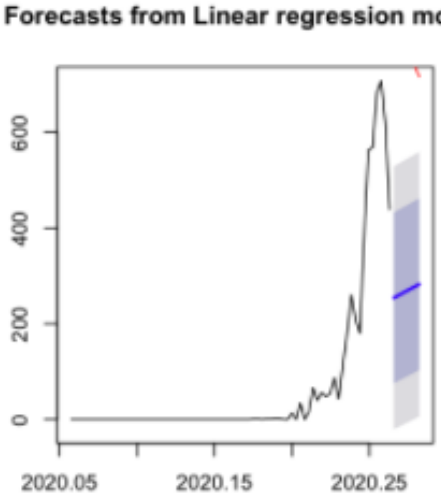
```
# Evaluate the performance of the model
accuracy(tslm_model_difffatalitiesUK.pred, UKtest2.ts)

accuracy(mytsfatalitiesUK.ANN.pred, UKtest2.ts)

accuracy(mytsfatalitiesUK.AAN.pred, UKtest2.ts)
```

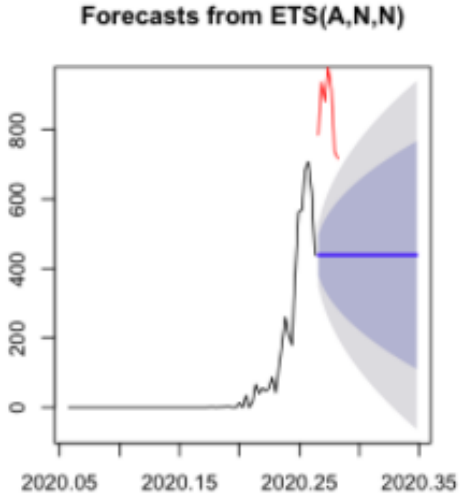
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 4.594826e-15 | 132.6716 | 96.07829 | NaN | Inf | NaN | 0.9245477 | NA |
| Test set | 5.824275e+02 | 591.0391 | 582.42745 | 67.97266 | 67.97266 | NaN | 0.3057407 | 5.427982 |

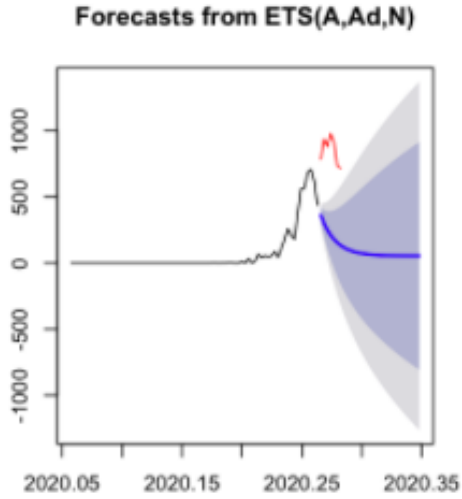
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 5.778776 | 45.41371 | 18.33112 | NaN | Inf | NaN | 0.2935970 | NA |
| Test set | 411.838939 | 422.90189 | 411.83894 | 47.71977 | 47.71977 | NaN | 0.2648284 | 3.914095 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.2008582 | 44.20025 | 19.41047 | NaN | Inf | NaN | 0.09804653 | NA |
| Test set | 623.0176023 | 631.81555 | 623.01760 | 73.19613 | 73.19613 | NaN | 0.15209638 | 5.970723 |

```
#Plot and compare the validation forecasts against the actual validation data

options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))

plot(tslm_model_difffatalitiesUK.pred)
lines(UKtest2.ts, col="red")  # plots the actual validation data
plot(mytsfatalitiesUK.ANN.pred)
lines(UKtest2.ts, col="red")
plot(mytsfatalitiesUK.AAN.pred)
lines(UKtest2.ts, col="red")
```


Forecasts from Linear regression model
0
200
400
600
2020.05
2020.15
2020.25


Forecasts from ETS(A,N,N)
0
200
400
600
800
2020.05
2020.15
2020.25
2020.35


Forecasts from ETS(A,Ad,N)
-1000
-500
0
500
1000
2020.05
2020.15
2020.25
2020.35

Ensemble model (Stacking)

In [264]:
```
#Step 1. Split the training set into a smaller training set and a validation set.

train_UK$Date <- as.Date(train_UK$Date, "%Y-%m-%d")

smalltrainUK.df <- train_UK[which(train_UK$Date <= as.Date("2020-3-24") ),]

validUK.df <- train_UK[which(train_UK$Date >= as.Date("2020-3-24") ),]




#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainUKdifffatalities.df <- ts(smalltrainUK.df$diff_fatalities,
           start = c(2020, as.numeric(format(d1[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainUKdifffatalities.df.ANN <- ets(myts_smalltrainUKdifffatalities.df, model = "ANN"))
myts_smalltrainUKdifffatalities.df.ANN.pred <- forecast(myts_smalltrainUKdifffatalities.df.ANN, h = 14)



#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainUKdifffatalities.df.AAN <- ets(myts_smalltrainUKdifffatalities.df, model = "AAN"))
myts_smalltrainUKdifffatalities.df.AAN.pred <- forecast(myts_smalltrainUKdifffatalities.df.AAN , h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainUKdifffatalities.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.5497

  Initial states:
    l = -0.0557

  sigma:  9.884

     AIC     AICc      BIC
553.6402 554.0470 560.0696

ETS(A,A,N)

Call:
 ets(y = myts_smalltrainUKdifffatalities.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1477
    beta  = 0.1477

  Initial states:
    l = -0.0072
    b = -0.0209

  sigma:  8.3457

     AIC     AICc      BIC
534.2251 535.2777 544.9407
```

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.diffatalitiesUK <- data.frame(Date = validUK.df$Date, diff_fatalities = validUK.df$diff_fatalities,
                         myts_smalltrainUKdifffatalities.df.ANN.pred = myts_smalltrainUKdifffatalities.df.ANN.pred,
                         myts_smalltrainUKdifffatalities.df.AAN.pred =
                         myts_smalltrainUKdifffatalities.df.AAN.pred)



library(dplyr)

stacker.diffatalitiesUK <- stacker.diffatalitiesUK[,c("Date","diff_fatalities","myts_smalltrainUKdifffatalities.df.ANN.
                           "myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast")]

head(stacker.diffatalitiesUK)
```

| | Date | diff_fatalities | myts_smalltrainUKdifffatalities.df.ANN.pred.Point.Forecast | myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 87 | 71.03488 | 86.48488 |
| **2020.2329** | 2020-03-25 | 43 | 71.03488 | 96.33341 |
| **2020.2356** | 2020-03-26 | 113 | 71.03488 | 106.18193 |
| **2020.2384** | 2020-03-27 | 181 | 71.03488 | 116.03046 |
| **2020.2411** | 2020-03-28 | 260 | 71.03488 | 125.87899 |
| **2020.2438** | 2020-03-29 | 209 | 71.03488 | 135.72752 |

```
library(rpart)

library(randomForest)

#Fit a random forest to the predictions as stacker model 1
stackerModel_1.UKfatalities <- randomForest(stacker.diffatalitiesUK$diff_fatalities
                                            ~ stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.ANN.pred.Point
                        +stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast,
                  data = stacker.diffatalitiesUK,ntree = 1000, mtry = 2, nodesize = 5, importance = TRUE)



summary(stackerModel_1.UKfatalities)



#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModel_2UKfatalities <- rpart(stacker.diffatalitiesUK$diff_fatalities ~
                                      stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.ANN.pred.Point.Forecast
                           + stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast
                           , data= stacker.diffatalitiesUK, method= "anova")


summary(stackerModel_2UKfatalities)

stacker.diffatalitiesUK.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.diffatalitiesUK[,c(-1,-2)])
stacker.y.difffatalitiesUK <- as.vector(stacker.diffatalitiesUK[,2])
stackerModel_3UKfatalities <- xgboost(stacker.diffatalitiesUK.xgb, stacker.y.difffatalitiesUK, nrounds=200,
                       verbose = 0)

summary(stackerModel_3UKfatalities)
```

```
                Length Class  Mode
call               7   -none- call
type               1   -none- character
predicted         14   -none- numeric
mse             1000   -none- numeric
rsq             1000   -none- numeric
oob.times         14   -none- numeric
importance         4   -none- numeric
importanceSD       2   -none- numeric
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            11   -none- list
coefs              0   -none- NULL
y                 14   -none- numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call

Call:
rpart(formula = stacker.diffatalitiesUK$diff_fatalities ~ stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.
ANN.pred.Point.Forecast +
    stacker.diffatalitiesUK$myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast,
    data = stacker.diffatalitiesUK, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=359.8571, MSE=51360.12


               Length Class              Mode
handle              1 xgb.Booster.handle externalptr
raw             88726 -none-             raw
niter               1 -none-             numeric
evaluation_log      2 data.table         list
call               13 -none-             call
params              1 -none-             list
callbacks           1 -none-             list
feature_names       2 -none-             character
nfeatures           1 -none-             numeric
```

```r
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

mytsfatalitiesUK <- ts(train_UK$diff_fatalities,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model
(mytsfatalitiesUK.ANN <- ets(mytsfatalitiesUK, model = "ANN"))
mytsfatalitiesUK.ANN.pred.test <- forecast(mytsfatalitiesUK.ANN, h = 14)


#Generate an additive trend (Holt's linear) ets model
(mytsfatalitiesUK.AAN <- ets(mytsfatalitiesUK, model = "AAN"))
mytsfatalitiesUK.AAN.pred.test <- forecast(mytsfatalitiesUK.AAN, h = 14)



predict.variables.UKfatalities <- data.frame('myts_smalltrainUKdifffatalities.df.ANN.pred.Point.Forecast' =
                                    mytsfatalitiesUK.ANN.pred.test,
                                  'myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast' =
                                    mytsfatalitiesUK.AAN.pred.test)


#Only select the forecast values
predict.variables.UKfatalities <- predict.variables.UKfatalities%>% select(1,6)


predict.variables.UKfatalities
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalitiesUK, model = "ANN")

  Smoothing parameters:
    alpha = 0.9999

  Initial states:
    l = -0.1248

  sigma:  46.0233

     AIC     AICc      BIC
915.1395 915.4728 922.1317

ETS(A,Ad,N)

Call:
 ets(y = mytsfatalitiesUK, model = "AAN")

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.476
    phi   = 0.8

  Initial states:
    l = -0.7114
    b = 2.5812

  sigma:  45.7301

     AIC     AICc      BIC
917.0228 918.2402 931.0072
```

| | myts_smalltrainUKdifffatalities.df.ANN.pred.Point.Forecast.Point.Forecast | myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast.Point.Forecast |
|---|---|---|
| 2020.2658 | 439.0182 | 362.00222 |
| 2020.2685 | 439.0182 | 300.39059 |
| 2020.2712 | 439.0182 | 251.10128 |
| 2020.2740 | 439.0182 | 211.66983 |
| 2020.2767 | 439.0182 | 180.12467 |
| 2020.2795 | 439.0182 | 154.88854 |
| 2020.2822 | 439.0182 | 134.69964 |
| 2020.2849 | 439.0182 | 118.54851 |
| 2020.2877 | 439.0182 | 105.62761 |
| 2020.2904 | 439.0182 | 95.29089 |
| 2020.2932 | 439.0182 | 87.02151 |
| 2020.2959 | 439.0182 | 80.40601 |
| 2020.2986 | 439.0182 | 75.11360 |
| 2020.3014 | 439.0182 | 70.87968 |

```
# Predict from stacker model 1 --- Random forest
stacker.predict.rt.UKfatalities1 <- predict(stackerModel_1.UKfatalities, predict.variables.UKfatalities )


# Predict from stacker model 2 --- Regression tree
stacker.predict.rt.UKfatalities2 <- predict(stackerModel_2UKfatalities, predict.variables.UKfatalities )


# Predict from stacker model_3 --- XGBoost
predict.variables.UKfatalities.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.UKfatalities)

colnames(predict.variables.UKfatalities.xgb) =
c('myts_smalltrainUKdifffatalities.df.ANN.pred.Point.Forecast',
  'myts_smalltrainUKdifffatalities.df.AAN.pred.Point.Forecast')

stacker.predict.rt.UKfatalities3 <- predict(stackerModel_3UKfatalities, predict.variables.UKfatalities.xgb)
```

```
accuracy(stacker.predict.rt.UKfatalities1, UKtest2.ts)

accuracy(stacker.predict.rt.UKfatalities2, UKtest2.ts)

accuracy(stacker.predict.rt.UKfatalities3, UKtest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 693.386 | 704.4792 | 693.386 | 80.96286 | 80.96286 | 0.4853349 | 6.408115 |

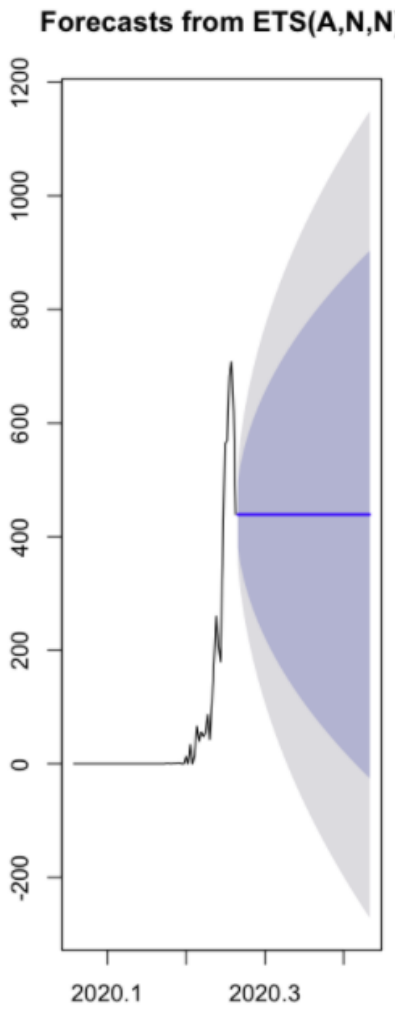
| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 491 | 500.3156 | 491 | 57.14662 | 57.14662 | 0.2648284 | 4.619413 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 418.0004 | 430.373 | 418.0004 | 49.61202 | 49.61202 | -0.2411651 | 4.164773 |

Overall, considering RMSE scores for both the training and test dataset, the exponential smoothing model (ANN) is the best model.

If we try to forecast for the next two months using this model, we can see that the number of deaths per day caused by Covid-19 should stay constant and be equal to about 439 in early June:

```
#Generate forecasts to June 6th 2020  using the above model
mytsfatalitiesUK.ANN.pred2months <- forecast(mytsfatalitiesUK.ANN, h = 62)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(mytsfatalitiesUK.ANN.pred2months)
```




```
mytsfatalitiesUK.ANN.pred2months$mean[62]
```

439.018203919078

## Appendix 1-D: Forecast of the number of cases per day in France

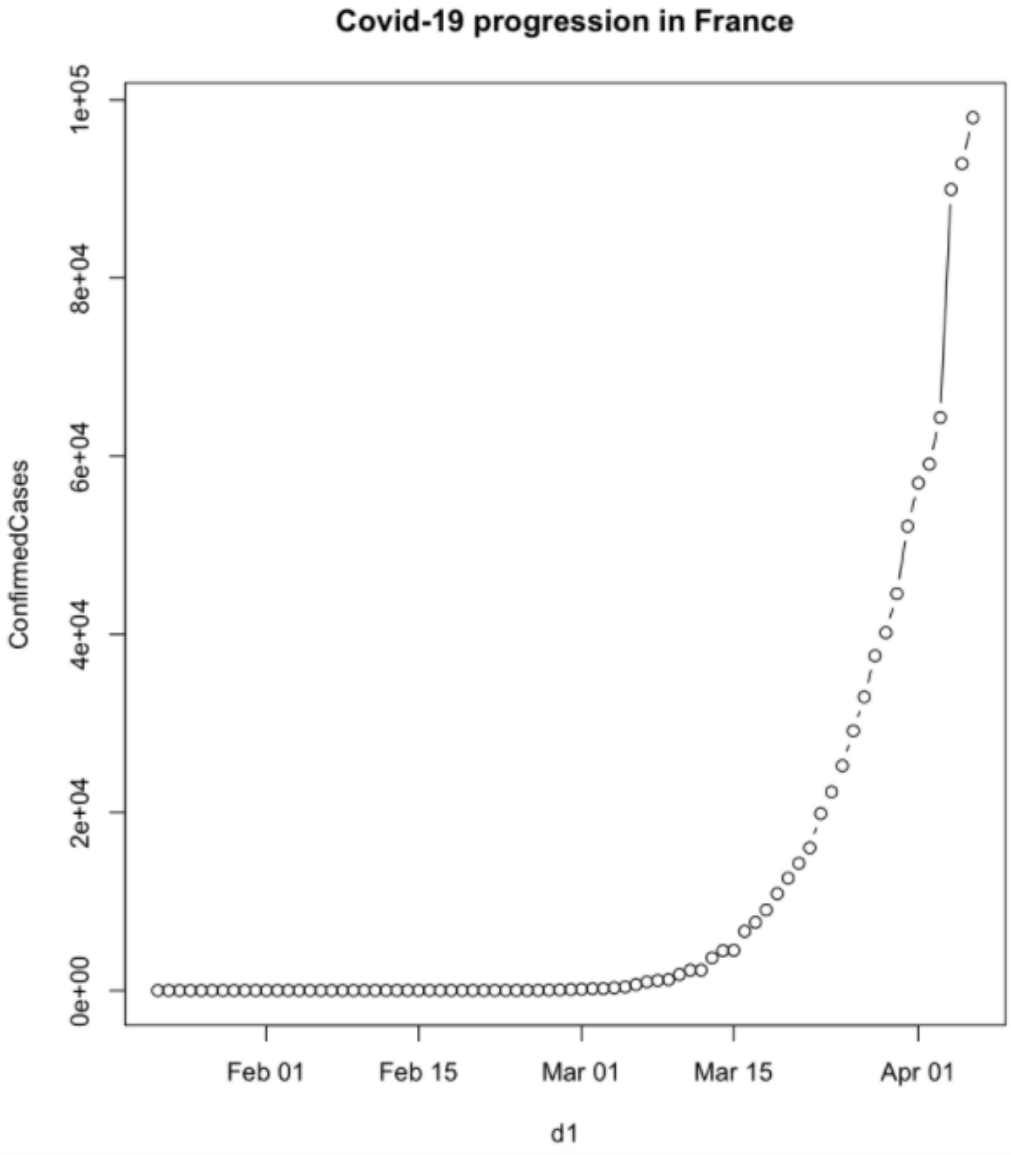
**France FORECAST**

In [141]:
```
train_France = train[which(train$Country_Region=='France' & train$Province_State =='' ),]
```

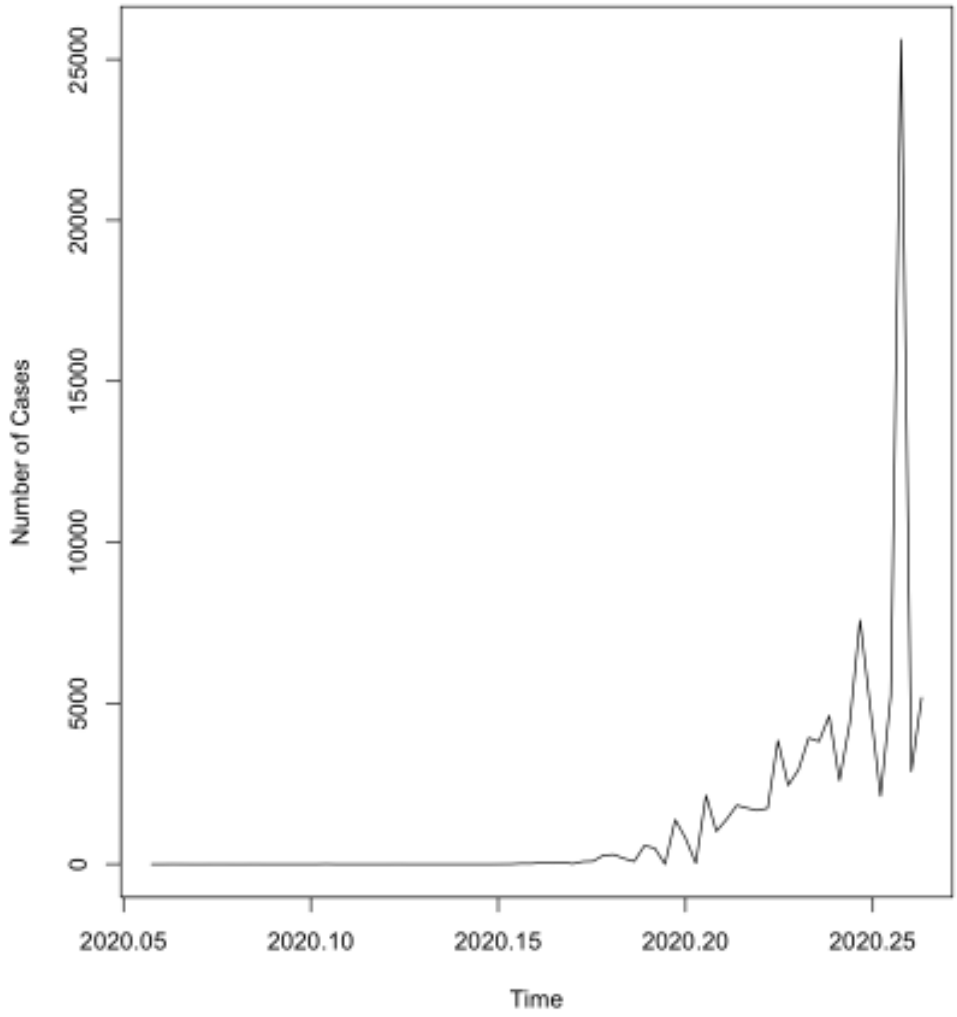
In [142]:
```
#Create a new column with the number of new cases per day in Italy
library(dplyr)

train_France <- train_France %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff |
|---|---|---|---|---|---|---|---|
| 1 | 13055 | | France | 2020-01-22 | 0 | 0 | 0 |
| 2 | 13056 | | France | 2020-01-23 | 0 | 0 | 0 |
| 3 | 13057 | | France | 2020-01-24 | 2 | 0 | 2 |
| 4 | 13058 | | France | 2020-01-25 | 3 | 0 | 1 |
| 73 | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 |
| 74 | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 |
| 75 | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 |
| 76 | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 |

In [143]:
```
# A plot with the progression of Covid-19 in France

train_France$Date <- as.Date(train_France$Date, "%Y-%m-%d")

d1 <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")

plot(x = d1, y = train_France$ConfirmedCases, type = "b", ylab = "ConfirmedCases",main = "Covid-19 progression in Franc
```



```
set.seed(25)
mytsFrance <- ts(train_France$diff,
           start = c(2020, as.numeric(format(d1[1], "%j"))),
           frequency = 365)
plot(mytsFrance, ylab = "Number of Cases")
```



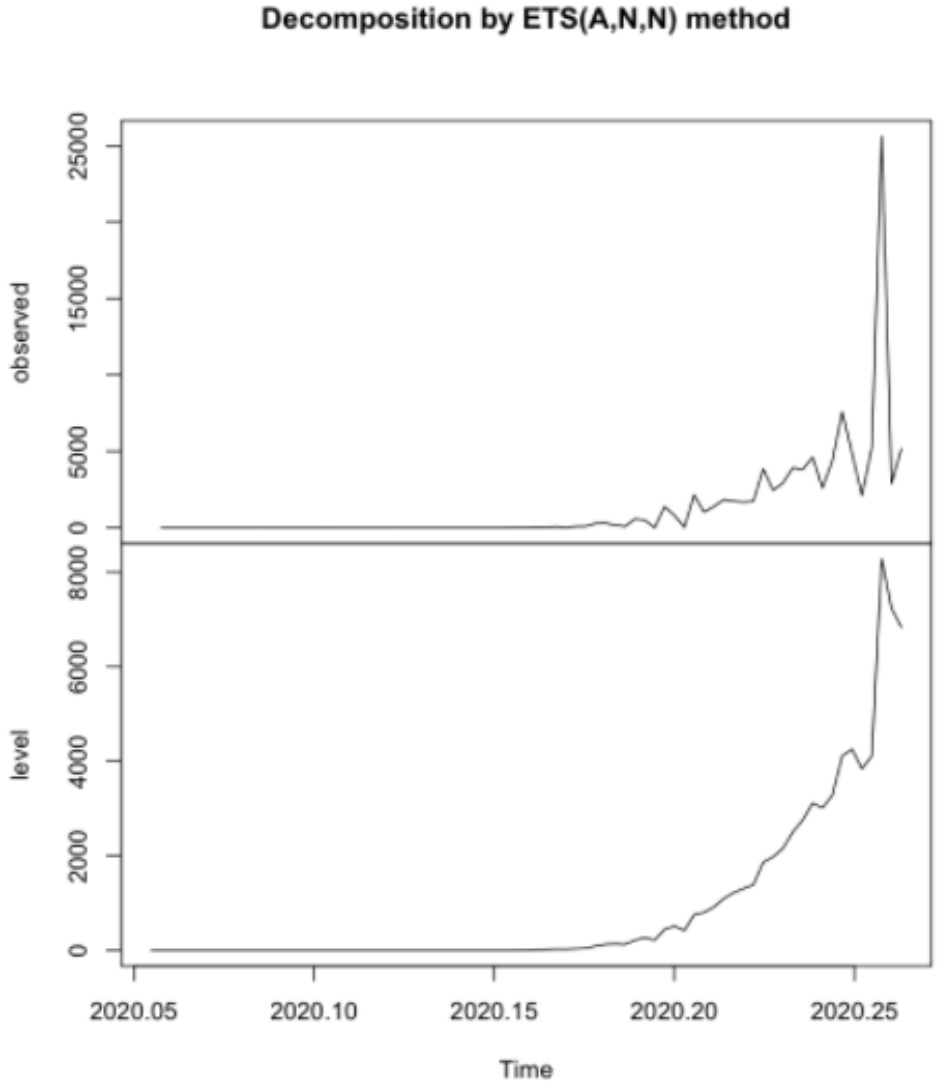

```
#Generate an exponential smoothing ets model

(mytsFrance.ANN <- ets(mytsFrance, model = "ANN"))
```

```
ETS(A,N,N)

Call:
 ets(y = mytsFrance, model = "ANN")

  Smoothing parameters:
    alpha = 0.1938

  Initial states:
    l = 0.5781

  sigma:  2716.468

     AIC     AICc      BIC
1534.986 1535.320 1541.978
```

```
#RMSE error for the ets model

rmse.ets(mytsFrance.ANN)
```

```
RMSE =  2680.486
```

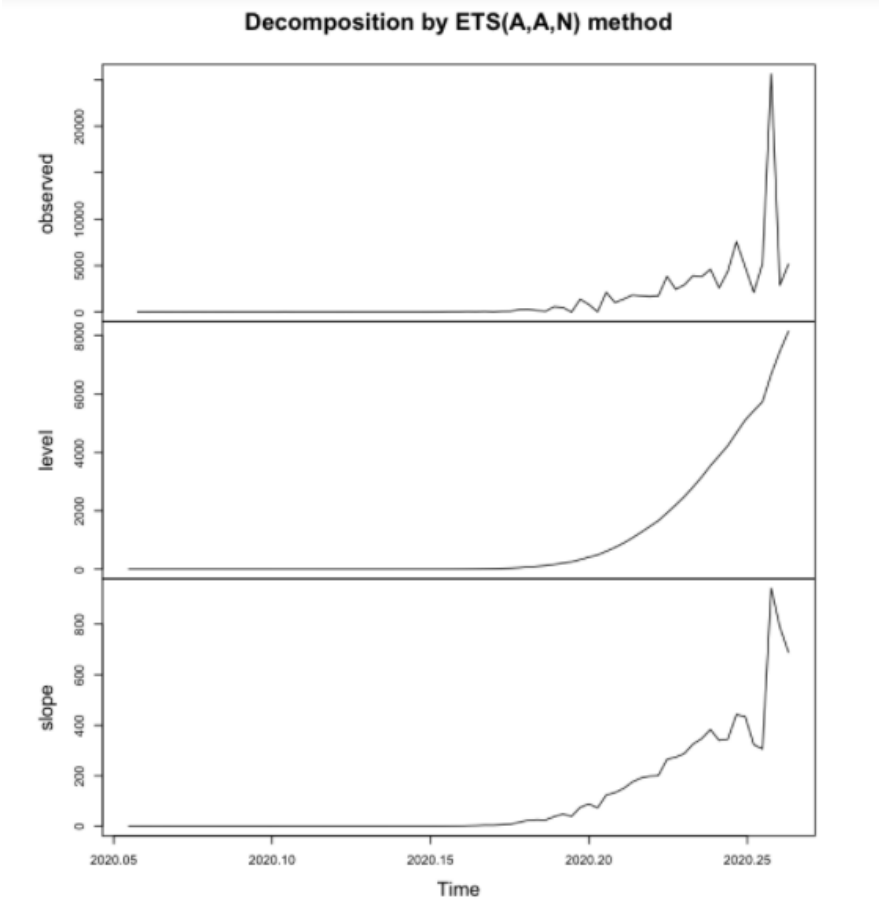
```
#Decomposition plot of fitted ets model¶
plot(mytsFrance.ANN)
```



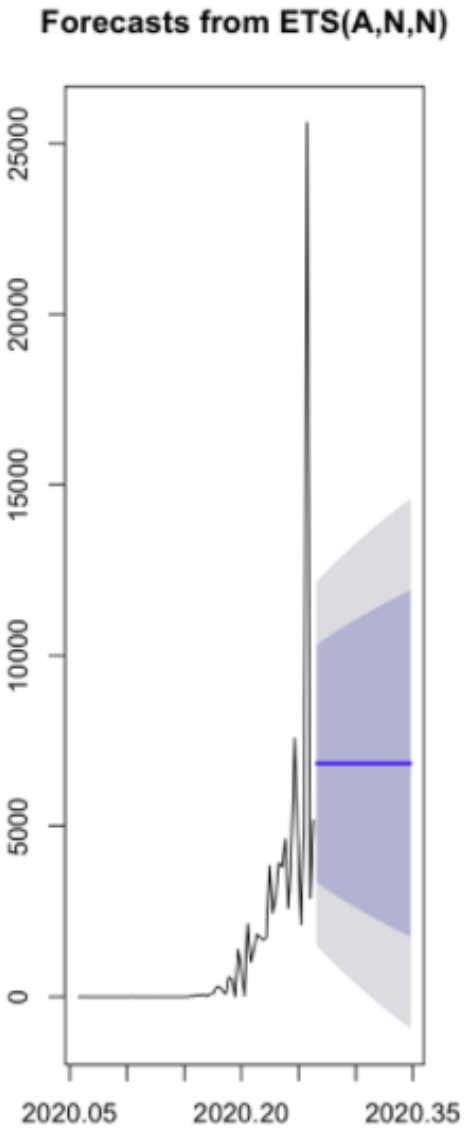

```
#Generate an additive trend (Holt's linear) ets model

(mytsFrance.AAN <- ets(mytsFrance, model = "AAN"))

rmse.ets(mytsFrance.AAN)
```

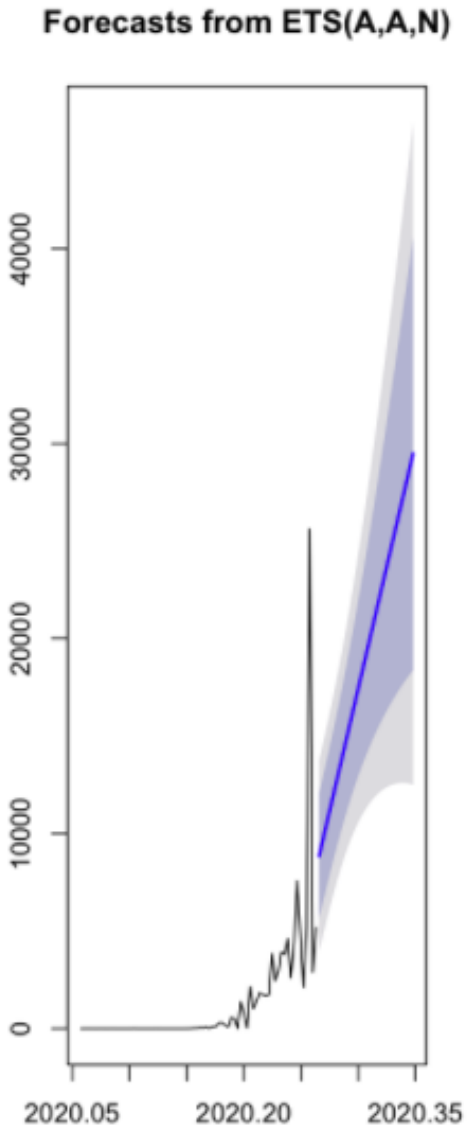
```
ETS(A,A,N)

Call:
 ets(y = mytsFrance, model = "AAN")

  Smoothing parameters:
    alpha = 0.0325
    beta  = 0.0325

  Initial states:
    l = 0.7136
    b = 0.1813

  sigma:  2502.313

     AIC     AICc      BIC
1524.422 1525.279 1536.076

RMSE =  2435.572
```

```
#Decomposition plot of fitted additive trend ets model

plot(mytsFrance.AAN)
```




```
#Generate forecasts to May 7th 2020  using the above models
mytsFrance.ANN.pred <- forecast(mytsFrance.ANN, h = 31)
mytsFrance.AAN.pred <- forecast(mytsFrance.AAN, h = 31)

# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsFrance.ANN.pred)
plot(mytsFrance.AAN.pred)
```



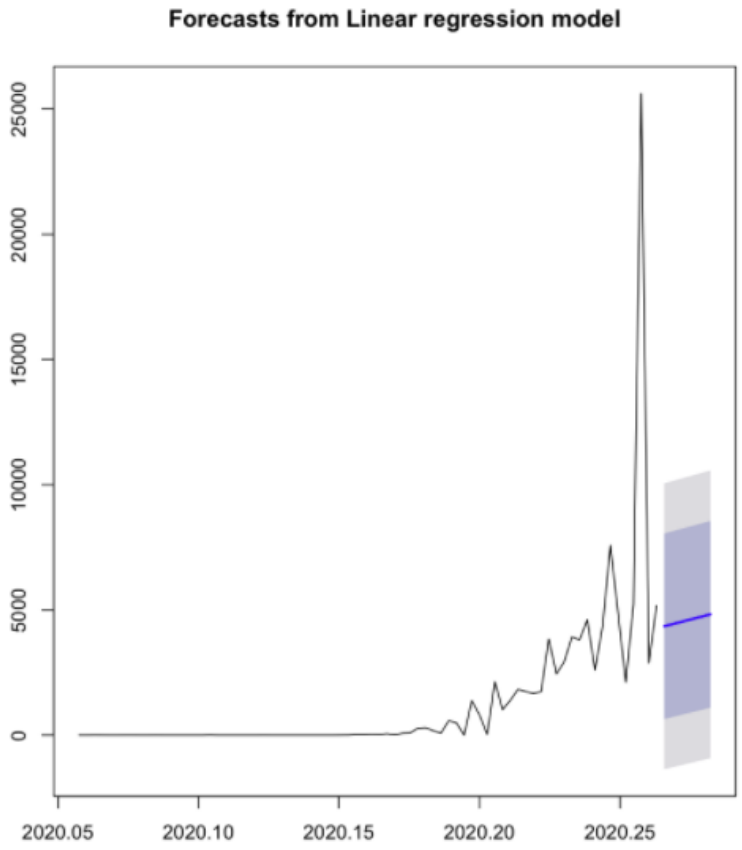

```
#And print the final new cases per day forecasts for May 7th 2020 for each model:
mytsFrance.ANN.pred$mean[31]
mytsFrance.AAN.pred$mean[31]
```

```
6834.2728043123

29485.8273483421
```

```r
#Generate a tslm model
tslm_model_diffFrance <- tslm(mytsFrance~ trend)
tslm_model_diffFrance.pred <- forecast(tslm_model_diffFrance, h = 7)

tslm_model_diffFrance
```

```
Call:
tslm(formula = mytsFrance ~ trend)

Coefficients:
(Intercept)        trend
   -1769.87        79.47
```

```r
plot(tslm_model_diffFrance.pred)
```




```r
#I want the test data set to contain the Confirmed Cases and Fatalities from the 7th to the 13th of April
test <- read.csv("test.csv")
n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 |
| **25976** | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 |
| **25977** | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 |
| **25978** | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 |
| **25979** | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 |

```r
#Create a new column with the number of new cases per day
library(dplyr)

test <- test %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

#Create a new column with the number of fatalities per day

test <- test %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))


n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 | 0 | 0 |
| **25976** | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 | 2 | 0 |
| **25977** | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 | 1 | 0 |
| **25978** | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 | 0 | 0 |
| **25979** | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 | 3 | 0 |

```
test$Date <- as.Date(test$Date, "%Y-%m-%d")

#Take values between the 7th and the 13th of April and for country = Italy
test_France = test[which(test$Country_Region=='France' & test$Province_State =='' 
                         & test$Date >= as.Date("2020-4-7") ),]

n <- nrow(test_France)
test_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **10286** | 14099 | | France | 2020-04-07 | 109069 | 10328 | 11059 | 1417 |
| **10287** | 14100 | | France | 2020-04-08 | 112950 | 10869 | 3881 | 541 |
| **10288** | 14101 | | France | 2020-04-09 | 117749 | 12210 | 4799 | 1341 |
| **10289** | 14102 | | France | 2020-04-10 | 124869 | 13197 | 7120 | 987 |
| **10289.1** | 14102 | | France | 2020-04-10 | 124869 | 13197 | 7120 | 987 |
| **10290** | 14103 | | France | 2020-04-11 | 129654 | 13832 | 4785 | 635 |
| **10291** | 14104 | | France | 2020-04-12 | 132591 | 14393 | 2937 | 561 |
| **10292** | 14105 | | France | 2020-04-13 | 136779 | 14967 | 4188 | 574 |

```
d2 <- seq(from = as.Date("2020-4-7", "%Y-%m-%d"), to = as.Date("2020-4-13", "%Y-%m-%d"), by = "day")

Francetest.ts <- ts(test_France$diff,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

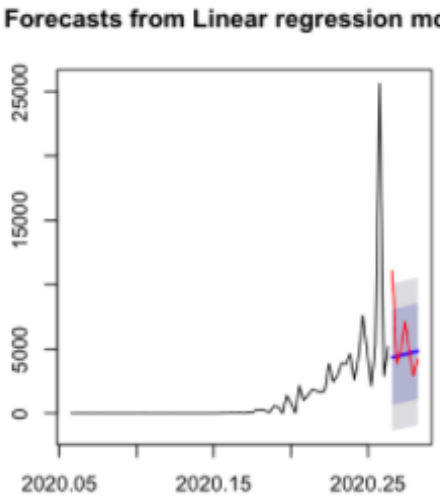
```
# Evaluate the performance of the model
accuracy(tslm_model_diffFrance.pred, Francetest.ts)

accuracy(mytsFrance.ANN.pred, Francetest.ts)

accuracy(mytsFrance.AAN.pred, Francetest.ts)
```

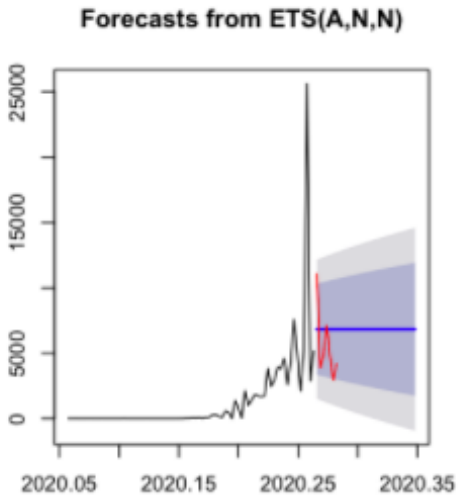
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 5.264575e-14 | 2753.392 | 1328.558 | NaN | Inf | NaN | 0.13193292 | NA |
| **Test set** | 9.509423e+02 | 2816.175 | 1906.617 | 1.97508 | 27.96011 | NaN | -0.06743931 | 0.6442363 |

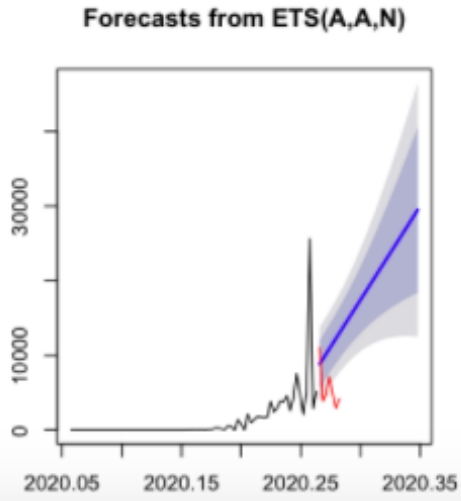
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 463.9443 | 2680.486 | 752.0617 | -Inf | Inf | NaN | -0.1487562 | NA |
| **Test set** | -1295.8442 | 2858.199 | 2584.5455 | -45.00015 | 57.0615 | NaN | -0.1059291 | 1.288446 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 278.6956 | 2435.572 | 650.6012 | NaN | Inf | NaN | -0.2038935 | NA |
| **Test set** | -5367.1298 | 6451.143 | 6000.8200 | -137.3887 | 143.1188 | NaN | 0.1654693 | 3.789235 |

```
#Plot and compare the validation forecasts against the actual validation data

options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))

plot(tslm_model_diffFrance.pred)
lines(Francetest.ts, col="red")  # plots the actual validation data
plot(mytsFrance.ANN.pred)
lines(Francetest.ts, col="red")
plot(mytsFrance.AAN.pred)
lines(Francetest.ts, col="red")
```



```
#Step 1. Split the training set into a smaller training set and a validation set.

train_France$Date <- as.Date(train_France$Date, "%Y-%m-%d")

smalltrainFrance.df <- train_France[which(train_France$Date <= as.Date("2020-3-24") ),]

validFrance.df <- train_France[which(train_France$Date >= as.Date("2020-3-24") ),]


#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainFrancediff.df <- ts(smalltrainFrance.df$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainFrancediff.df.ANN <- ets(myts_smalltrainFrancediff.df, model = "ANN"))
myts_smalltrainFrancediff.df.ANN.pred <- forecast(myts_smalltrainFrancediff.df.ANN, h = 14)



#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainFrancediff.df.AAN <- ets(myts_smalltrainFrancediff.df, model = "AAN"))
myts_smalltrainFrancediff.df.AAN.pred <- forecast(myts_smalltrainFrancediff.df.AAN , h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainFrancediff.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.4298

  Initial states:
    l = 0.386

  sigma:  413.2779

     AIC     AICc      BIC
1024.024 1024.431 1030.454

ETS(A,A,N)

Call:
 ets(y = myts_smalltrainFrancediff.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.0889
    beta  = 0.0889

  Initial states:
    l = 0.7768
    b = 0.017

  sigma:  336.1653

      AIC      AICc       BIC
 999.9028 1000.9555 1010.6185
```

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.df.diffFrance <- data.frame(Date = validFrance.df$Date, diff = validFrance.df$diff,
                        myts_smalltrainFrancediff.df.ANN.pred = myts_smalltrainFrancediff.df.ANN.pred,
                        myts_smalltrainFrancediff.df.AAN.pred = myts_smalltrainFrancediff.df.AAN.pred)



library(dplyr)

stacker.df.diffFrance <- stacker.df.diffFrance[,c("Date","diff","myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast",
                          "myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast")]

head(stacker.df.diffFrance)
```

| | Date | diff | myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast | myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 2448 | 2537.117 | 2963.586 |
| **2020.2329** | 2020-03-25 | 2929 | 2537.117 | 3245.399 |
| **2020.2356** | 2020-03-26 | 3922 | 2537.117 | 3527.212 |
| **2020.2384** | 2020-03-27 | 3809 | 2537.117 | 3809.025 |
| **2020.2411** | 2020-03-28 | 4611 | 2537.117 | 4090.838 |
| **2020.2438** | 2020-03-29 | 2599 | 2537.117 | 4372.651 |

```
library(randomForest)

#Fit a random forest to the predictions as stacker model 1
stackerModelFrance_1 <- randomForest(stacker.df.diffFrance$diff ~
                                       stacker.df.diffFrance$myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast
                     + stacker.df.diffFrance$myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast,
                   data = stacker.df.diffFrance,  nodesize = 5, importance = TRUE)


#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModelFrance_2 <- rpart(stacker.df.diffFrance$diff
                              ~ stacker.df.diffFrance$myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast
                            + stacker.df.diffFrance$myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast
                            , data= stacker.df.diffFrance, method= "anova")


#Fit a XGBoost model to the predictions as stacker model_3
library(xgboost)

library(foreach)
library(parallel)
library(Matrix)
library(readr)
library(caret)

stacker.df.diffFrance.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.df.diffFrance[,c(-1,-2)])
stacker.diffFrance.y <- as.vector(stacker.df.diffFrance[,2])

stackerModelFrance_3 <- xgboost(stacker.df.diffFrance.xgb, stacker.diffFrance.y, nrounds=200,
                      verbose = 0)



summary(stackerModelFrance_1)

summary(stackerModelFrance_2)

summary(stackerModelFrance_3)
```

```
                Length Class  Mode
call              5    -none- call
type              1    -none- character
predicted        14    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times        14    -none- numeric
importance        4    -none- numeric
importanceSD      2    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y                14    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call

Call:
rpart(formula = stacker.df.diffFrance$diff ~ stacker.df.diffFrance$myts_smalltrainFrancediff.df.ANN.pred.Point.Foreca
st +
    stacker.df.diffFrance$myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast,
    data = stacker.df.diffFrance, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=5582.429, MSE=3.281802e+07


               Length Class              Mode
handle              1 xgb.Booster.handle externalptr
raw            129190 -none-             raw
niter               1 -none-             numeric
evaluation_log      2 data.table         list
call               13 -none-             call
params              1 -none-             list
callbacks           1 -none-             list
feature_names       2 -none-             character
nfeatures           1 -none-             numeric
```

```r
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

mytsdiffFrance <- ts(train_France$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model
(mytsdiffFrance.ANN <- ets(mytsdiffFrance, model = "ANN"))
mytsdiffFrance.ANN.pred.test <- forecast(mytsdiffFrance.ANN, h = 14)


#Generate an additive trend (Holt's linear) ets model
(mytsdiffFrance.AAN <- ets(mytsdiffFrance, model = "AAN"))
mytsdiffFrance.AAN.pred.test <- forecast(mytsdiffFrance.AAN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsdiffFrance, model = "ANN")

  Smoothing parameters:
    alpha = 0.1938

  Initial states:
    l = 0.5781

  sigma:  2716.468

     AIC     AICc      BIC
1534.986 1535.320 1541.978

ETS(A,A,N)

Call:
 ets(y = mytsdiffFrance, model = "AAN")

  Smoothing parameters:
    alpha = 0.0325
    beta  = 0.0325

  Initial states:
    l = 0.7136
    b = 0.1813

  sigma:  2502.313

     AIC     AICc      BIC
1524.422 1525.279 1536.076
```

```
predict.variables.diffFrance <- data.frame('myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast' = mytsdiffFrance.ANN.
                                'myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast' = mytsdiffFrance.AAN.pred.test)

#Only select the forecast values
predict.variables.diffFrance <- predict.variables.diffFrance%>% select(1,6)

predict.variables.diffFrance
```

| | myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast.Point.Forecast | myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast.Point.Forecast |
|---|---|---|
| 2020.2658 | 6834.273 | 8841.084 |
| 2020.2685 | 6834.273 | 9529.242 |
| 2020.2712 | 6834.273 | 10217.400 |
| 2020.2740 | 6834.273 | 10905.558 |
| 2020.2767 | 6834.273 | 11593.716 |
| 2020.2795 | 6834.273 | 12281.875 |
| 2020.2822 | 6834.273 | 12970.033 |
| 2020.2849 | 6834.273 | 13658.191 |
| 2020.2877 | 6834.273 | 14346.349 |
| 2020.2904 | 6834.273 | 15034.507 |
| 2020.2932 | 6834.273 | 15722.665 |
| 2020.2959 | 6834.273 | 16410.823 |
| 2020.2986 | 6834.273 | 17098.981 |
| 2020.3014 | 6834.273 | 17787.139 |

```
# Predict from stacker model_1 --- random forest
stacker.predict.diffFrance.rt1 <- predict(stackerModelFrance_1, predict.variables.diffFrance )

# Predict from stacker model_2 --- regression tree
stacker.predict.diffFrance.rt2 <- predict(stackerModelFrance_2, predict.variables.diffFrance )

# Predict from stacker model_3 --- XGBoost
predict.variables.diffFrance.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.diffFrance)

colnames(predict.variables.diffFrance.xgb) =
c('myts_smalltrainFrancediff.df.ANN.pred.Point.Forecast','myts_smalltrainFrancediff.df.AAN.pred.Point.Forecast')

stacker.predict.diffFrance.rt3 <- predict(stackerModelFrance_3, predict.variables.diffFrance.xgb)


accuracy(stacker.predict.diffFrance.rt1, Francetest.ts)


accuracy(stacker.predict.diffFrance.rt2, Francetest.ts)


accuracy(stacker.predict.diffFrance.rt3, Francetest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 724.5627 | 2695.672 | 1748.506 | -2.389854 | 27.99558 | -0.0801732 | 0.6316943 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | -44 | 2547.946 | 2048.041 | -18.44025 | 38.75925 | -0.1059291 | 0.7863667 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 367.43 | 2573.926 | 1871.714 | -9.711095 | 32.74405 | -0.1059291 | 0.6730795 |

Overall, considering RMSE scores for the test dataset, the regression tree ensemble stacker model seems to be the best model.

If we try to forecast for the next two months using this model, we can see that the number of cases per day caused by Covid-19 should stay constant and be equal to about 5582 in early June.

```
stacker.predict.diffFrance.rt2 <- predict(stackerModelFrance_2, predict.variables.diffFrance )

stacker.predict.diffFrance.rt2[62]
```

5582.42857142857

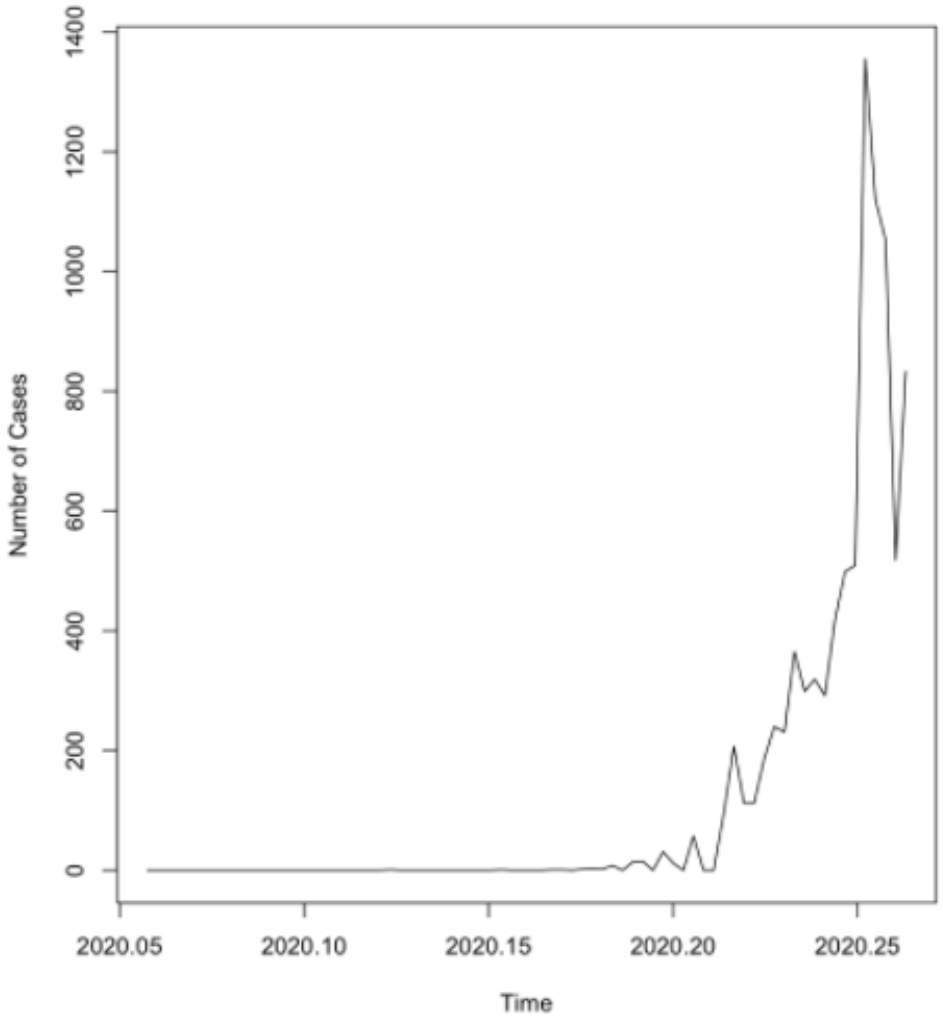
## Appendix 1-E: Forecast of the number of fatalities per day in France

### France forecast fatalities per day

In [152]:
```
#Create a new column with the number of new cases per day in France
library(dplyr)

train_France <- train_France %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))

n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **1** | 13055 | | France | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 13056 | | France | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 13057 | | France | 2020-01-24 | 2 | 0 | 2 | 0 |
| **4** | 13058 | | France | 2020-01-25 | 3 | 0 | 1 | 0 |
| **73** | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 | 1120 |
| **74** | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 | 1053 |
| **75** | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 | 518 |
| **76** | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 | 833 |

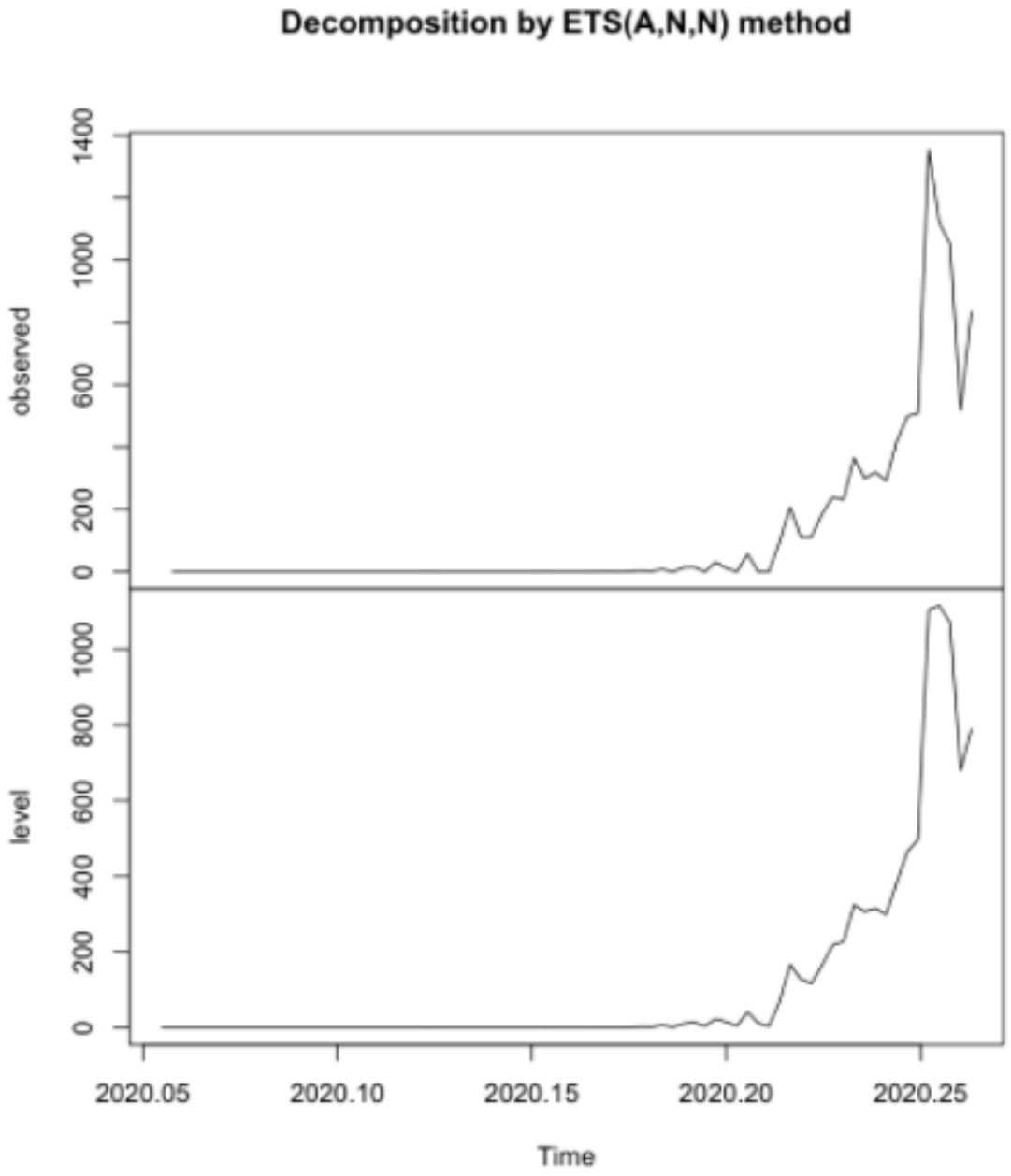
In [156]:
```
dl <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")
set.seed(25)
mytsfatalitiesFrance <- ts(train_France$diff_fatalities,
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
plot(mytsfatalitiesFrance, ylab = "Number of Cases")
```




[157]:
```
#Generate an exponential smoothing ets model

(mytsfatalitiesFrance.ANN <- ets(mytsfatalitiesFrance, model = "ANN"))

#RMSE error for the ets model

rmse.ets(mytsfatalitiesFrance.ANN)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalitiesFrance, model = "ANN")

  Smoothing parameters:
    alpha = 0.7079

  Initial states:
    l = -0.1293

  sigma:  125.6663

     AIC     AICc      BIC
1067.821 1068.154 1074.813

RMSE =  124.0018
```

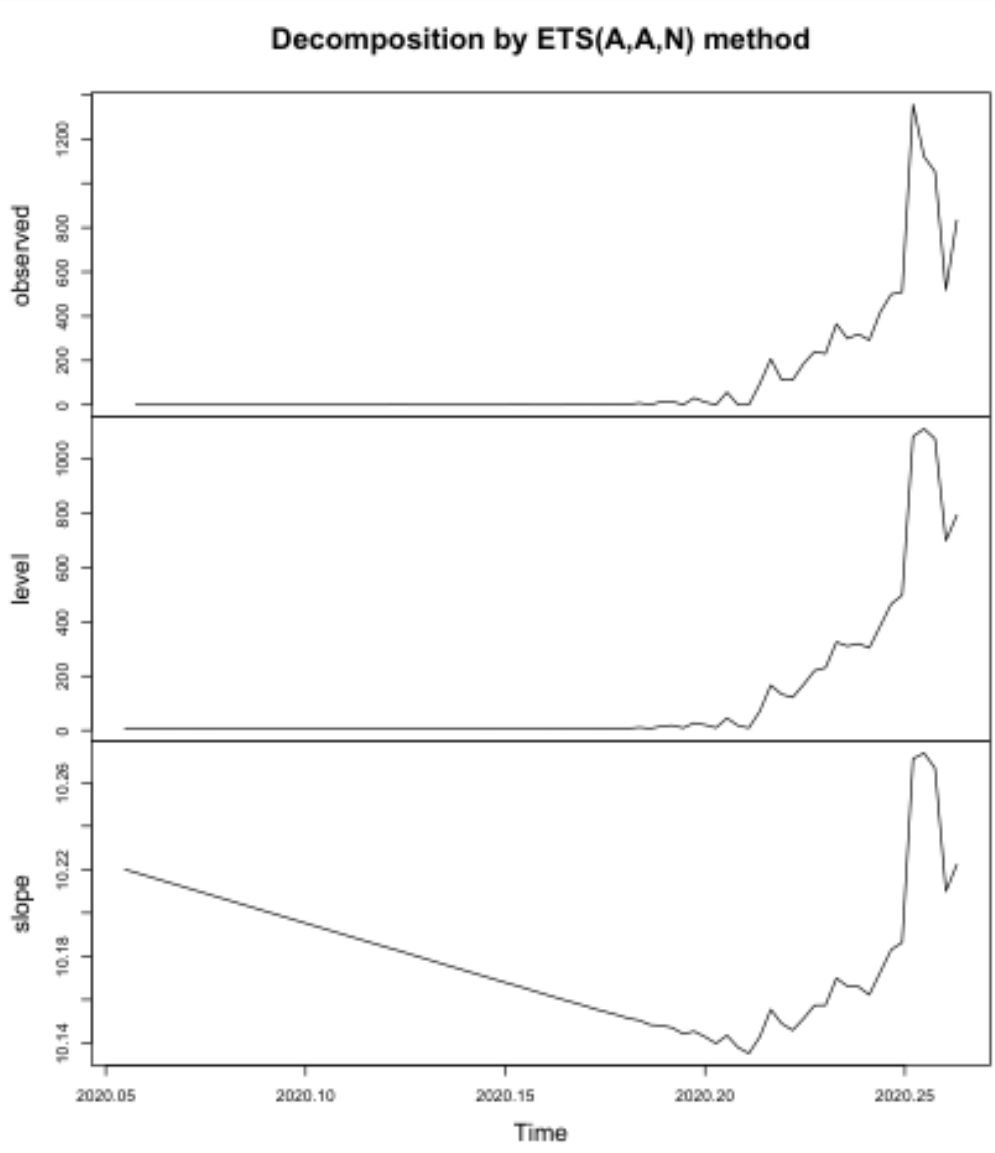
158]:
```
#Decomposition plot of fitted ets model¶
plot(mytsfatalitiesFrance.ANN)
```



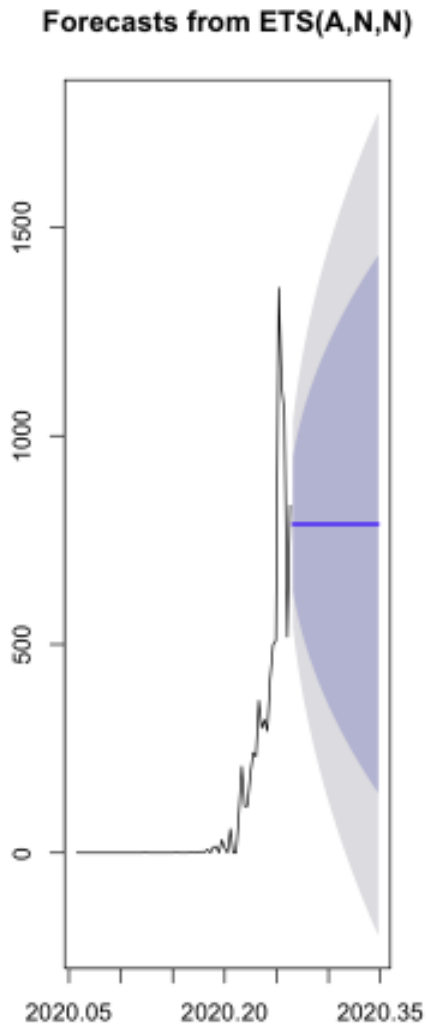

```
#Generate an additive trend (Holt's linear) ets model

(mytsfatalitiesFrance.AAN <- ets(mytsfatalitiesFrance, model = "AAN"))

rmse.ets(mytsfatalitiesFrance.AAN)
```

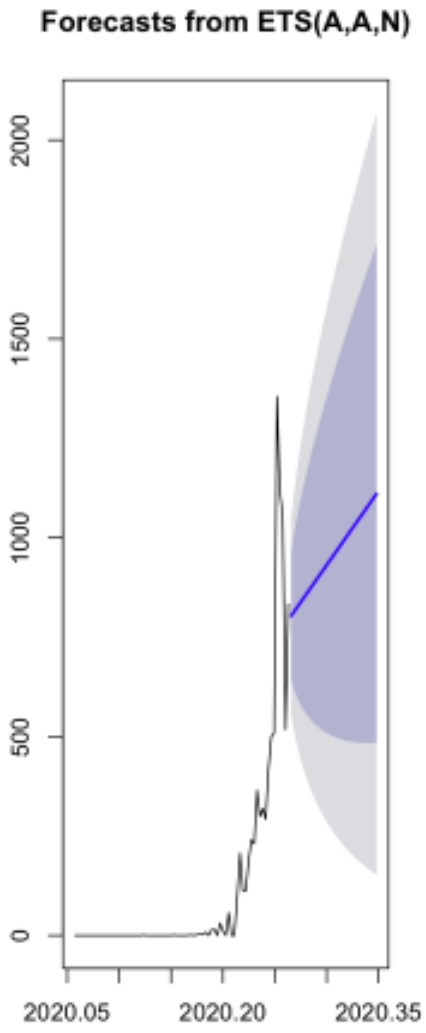
```
ETS(A,A,N)

Call:
 ets(y = mytsfatalitiesFrance, model = "AAN")

  Smoothing parameters:
    alpha = 0.6798
    beta  = 1e-04

  Initial states:
    l = 4.6159
    b = 10.22

  sigma:  126.4808

     AIC     AICc      BIC
1070.720 1071.578 1082.374

RMSE =  123.1074
```

```
#Decomposition plot of fitted additive trend ets model

plot(mytsfatalitiesFrance.AAN)
```



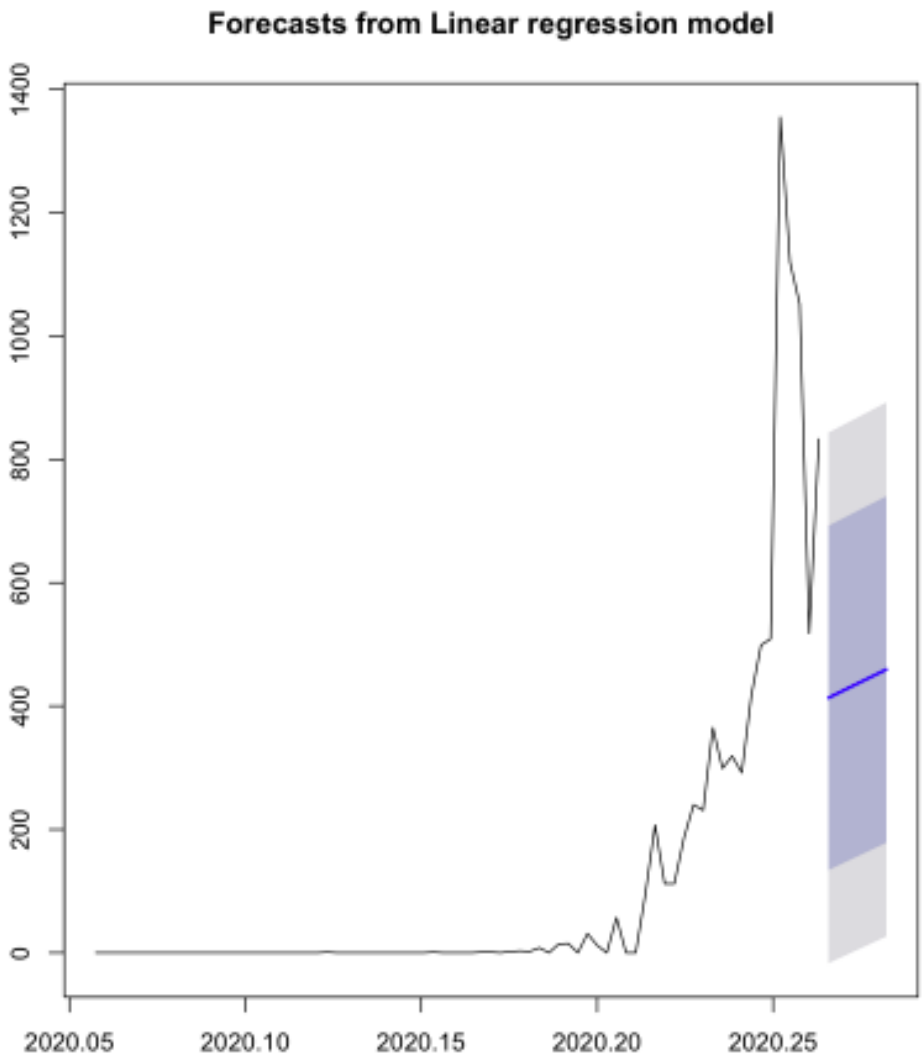

```r
#Generate forecasts to May 7th 2020  using the above models
mytsfatalitiesFrance.ANN.pred <- forecast(mytsfatalitiesFrance.ANN, h = 31)
mytsfatalitiesFrance.AAN.pred <- forecast(mytsfatalitiesFrance.AAN, h = 31)

# plot the forecasts for the ANN and AAN models
par(mfrow = c(1, 2))
plot(mytsfatalitiesFrance.ANN.pred)
plot(mytsfatalitiesFrance.AAN.pred)
```


Forecasts from ETS(A,N,N)
1500
1000
500
0
2020.05
2020.20
2020.35


Forecasts from ETS(A,A,N)
2000
1500
1000
500
0
2020.05
2020.20
2020.35

```r
#Generate a tslm model
tslm_model_difffatalitiesFrance <- tslm(mytsfatalitiesFrance~ trend)
tslm_model_difffatalitiesFrance.pred <- forecast(tslm_model_difffatalitiesFrance, h = 7)

tslm_model_difffatalitiesFrance



plot(tslm_model_difffatalitiesFrance.pred)
```

```
Call:
tslm(formula = mytsfatalitiesFrance ~ trend)

Coefficients:
(Intercept)        trend
   -178.845        7.691
```



```
#Create the test dataset time series for fatalities
Francetest2.ts <- ts(test_France$diff_fatalities,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

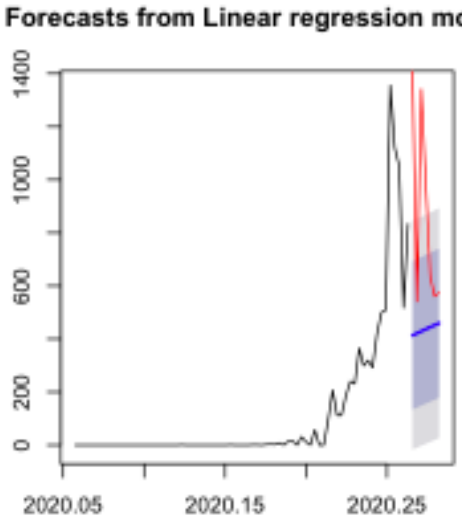
```
# Evaluate the performance of the model
accuracy(tslm_model_difffatalitiesFrance.pred, Francetest2.ts)

accuracy(mytsfatalitiesFrance.ANN.pred, Francetest2.ts)

accuracy(mytsfatalitiesFrance.AAN.pred, Francetest2.ts)
```

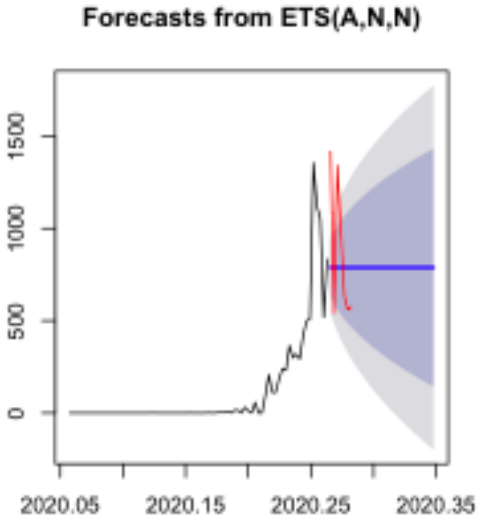
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 5.438632e-15 | 207.5269 | 139.7325 | NaN | Inf | NaN | 0.7772824 | NA |
| **Test set** | 4.287252e+02 | 562.9735 | 428.7252 | 40.89904 | 40.89904 | NaN | -0.1292809 | 1.060755 |

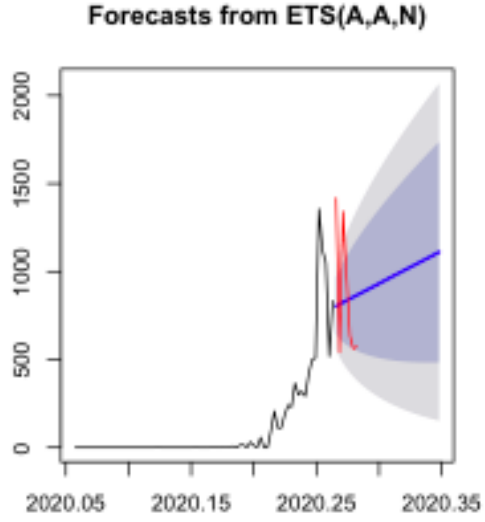
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 14.65269 | 124.0018 | 36.72435 | NaN | Inf | NaN | 0.005265166 | NA |
| **Test set** | 76.95247 | 363.0405 | 317.45577 | -5.983718 | 36.1959 | NaN | -0.164150298 | 0.7082471 |

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 0.2975472 | 123.1074 | 45.14395 | -Inf | Inf | NaN | 0.03085738 | NA |
| **Test set** | 30.6767426 | 369.5500 | 335.74916 | -12.76206 | 40.55177 | NaN | -0.11809380 | 0.7323956 |

```
#Plot and compare the validation forecasts against the actual validation data

options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))

plot(tslm_model_difffatalitiesFrance.pred)
lines(Francetest2.ts, col="red")  # plots the actual validation data
plot(mytsfatalitiesFrance.ANN.pred)
lines(Francetest2.ts, col="red")
plot(mytsfatalitiesFrance.AAN.pred)
lines(Francetest2.ts, col="red")
```


Forecasts from Linear regression model
0
200
600
1000
1400
2020.05
2020.15
2020.25


Forecasts from ETS(A,N,N)
0
500
1000
1500
2020.05
2020.15
2020.25
2020.35


Forecasts from ETS(A,A,N)
0
500
1000
1500
2000
2020.05
2020.15
2020.25
2020.35

Ensemble model -- Stacking

```
#Step 1. Split the training set into a smaller training set and a validation set.

train_France$Date <- as.Date(train_France$Date, "%Y-%m-%d")

smalltrainFrance.df <- train_France[which(train_France$Date <= as.Date("2020-3-24") ),]

validFrance.df <- train_France[which(train_France$Date >= as.Date("2020-3-24") ),]




#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainFrancedifffatalities.df <- ts(smalltrainFrance.df$diff_fatalities,
           start = c(2020, as.numeric(format(d1[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainFrancedifffatalities.df.ANN <- ets(myts_smalltrainFrancedifffatalities.df, model = "ANN"))
myts_smalltrainFrancedifffatalities.df.ANN.pred <- forecast(myts_smalltrainFrancedifffatalities.df.ANN, h = 14)



#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainFrancedifffatalities.df.AAN <- ets(myts_smalltrainFrancedifffatalities.df, model = "AAN"))
myts_smalltrainFrancedifffatalities.df.AAN.pred <- forecast(myts_smalltrainFrancedifffatalities.df.AAN , h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainFrancedifffatalities.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.7911

  Initial states:
    l = -0.0902

  sigma:  27.8179

     AIC     AICc      BIC
684.0206 684.4274 690.4500

ETS(A,A,N)

Call:
 ets(y = myts_smalltrainFrancedifffatalities.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1695
    beta  = 0.1695

  Initial states:
    l = -0.0359
    b = -0.0541

  sigma:  26.0529

     AIC     AICc      BIC
677.6609 678.7136 688.3766
```

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.diffatalitiesFrance <- data.frame(Date = validFrance.df$Date, diff_fatalities = validFrance.df$diff_fatalities,
                          myts_smalltrainFrancedifffatalities.df.ANN.pred = myts_smalltrainFrancedifffatalities.df.ANN.p
                          myts_smalltrainFrancedifffatalities.df.AAN.pred =
                          myts_smalltrainFrancedifffatalities.df.AAN.pred)



library(dplyr)

stacker.diffatalitiesFrance <- stacker.diffatalitiesFrance[,c("Date","diff_fatalities","myts_smalltrainFrancedifffatali
                           "myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast")]

head(stacker.diffatalitiesFrance)
```

| | Date | diff_fatalities | myts_smalltrainFrancedifffatalities.df.ANN.pred.Point.Forecast | myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| **2020.2301** | 2020-03-24 | 240 | 225.6176 | 255.4262 |
| **2020.2329** | 2020-03-25 | 231 | 225.6176 | 295.3526 |
| **2020.2356** | 2020-03-26 | 365 | 225.6176 | 335.2790 |
| **2020.2384** | 2020-03-27 | 299 | 225.6176 | 375.2054 |
| **2020.2411** | 2020-03-28 | 319 | 225.6176 | 415.1318 |
| **2020.2438** | 2020-03-29 | 292 | 225.6176 | 455.0582 |

```
library(rpart)

library(randomForest)

#Fit a random forest to the predictions as stacker model 1
stackerModel_1.Francefatalities <- randomForest(stacker.diffatalitiesFrance$diff_fatalities
                                                ~ stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.ANN.pr
                        +stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast,
                    data = stacker.diffatalitiesFrance,ntree = 1000, mtry = 2, nodesize = 5, importance = TRUE)



summary(stackerModel_1.Francefatalities)



#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModel_2Francefatalities <- rpart(stacker.diffatalitiesFrance$diff_fatalities ~
                                        stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.ANN.pred.Point.For
                        + stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast
                        , data= stacker.diffatalitiesFrance, method= "anova")


summary(stackerModel_2Francefatalities)

stacker.diffatalitiesFrance.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.diffatalitiesFrance[,c(-1,-2)])
stacker.y.difffatalitiesFrance <- as.vector(stacker.diffatalitiesFrance[,2])
stackerModel_3Francefatalities <- xgboost(stacker.diffatalitiesFrance.xgb, stacker.y.difffatalitiesFrance, nrounds=200,
                      verbose = 0)

summary(stackerModel_3Francefatalities)
```

```
                Length Class  Mode
call               7   -none- call
type               1   -none- character
predicted         14   -none- numeric
mse             1000   -none- numeric
rsq             1000   -none- numeric
oob.times         14   -none- numeric
importance         4   -none- numeric
importanceSD       2   -none- numeric
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            11   -none- list
coefs              0   -none- NULL
y                 14   -none- numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call

Call:
rpart(formula = stacker.diffatalitiesFrance$diff_fatalities ~
    stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.ANN.pred.Point.Forecast +
        stacker.diffatalitiesFrance$myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast,
    data = stacker.diffatalitiesFrance, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=575.0714, MSE=123926.1

                Length Class              Mode
handle             1   xgb.Booster.handle externalptr
raw            97070   -none-             raw
niter              1   -none-             numeric
evaluation_log     2   data.table         list
call              13   -none-             call
params             1   -none-             list
callbacks          1   -none-             list
feature_names      2   -none-             character
nfeatures          1   -none-             numeric
```

```r
#Step 4. Re-train the base models using the entire training set and generate predictions for the testing set

mytsfatalitiesFrance <- ts(train_France$diff_fatalities,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)


#Generate an exponential smoothing ets model
(mytsfatalitiesFrance.ANN <- ets(mytsfatalitiesFrance, model = "ANN"))
mytsfatalitiesFrance.ANN.pred.test <- forecast(mytsfatalitiesFrance.ANN, h = 14)


#Generate an additive trend (Holt's linear) ets model
(mytsfatalitiesFrance.AAN <- ets(mytsfatalitiesFrance, model = "AAN"))
mytsfatalitiesFrance.AAN.pred.test <- forecast(mytsfatalitiesFrance.AAN, h = 14)



predict.variables.Francefatalities <- data.frame('myts_smalltrainFrancedifffatalities.df.ANN.pred.Point.Forecast' =
                                    mytsfatalitiesFrance.ANN.pred.test,
                                'myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast' =
                                    mytsfatalitiesFrance.AAN.pred.test)




#Only select the forecast values
predict.variables.Francefatalities <- predict.variables.Francefatalities%>% select(1,6)
```

```
ETS(A,N,N)

Call:
 ets(y = mytsfatalitiesFrance, model = "ANN")

  Smoothing parameters:
    alpha = 0.7079

  Initial states:
    l = -0.1293

  sigma:  125.6663

     AIC     AICc      BIC
1067.821 1068.154 1074.813

ETS(A,A,N)

Call:
 ets(y = mytsfatalitiesFrance, model = "AAN")

  Smoothing parameters:
    alpha = 0.6798
    beta  = 1e-04

  Initial states:
    l = 4.6159
    b = 10.22

  sigma:  126.4808

     AIC     AICc      BIC
1070.720 1071.578 1082.374
```

```
# Predict from stacker model 1 --- Random forest
stacker.predict.rt.Francefatalities1 <- predict(stackerModel_1.Francefatalities, predict.variables.Francefatalities )


# Predict from stacker model 2 --- Regression tree
stacker.predict.rt.Francefatalities2 <- predict(stackerModel_2Francefatalities, predict.variables.Francefatalities )


# Predict from stacker model_3 --- XGBoost
predict.variables.Francefatalities.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.Francefatalities)

colnames(predict.variables.Francefatalities.xgb) =
c('myts_smalltrainFrancedifffatalities.df.ANN.pred.Point.Forecast',
  'myts_smalltrainFrancedifffatalities.df.AAN.pred.Point.Forecast')

stacker.predict.rt.Francefatalities3 <- predict(stackerModel_3Francefatalities, predict.variables.Francefatalities.xgb)

accuracy(stacker.predict.rt.Francefatalities1, Francetest2.ts)

accuracy(stacker.predict.rt.Francefatalities2, Francetest2.ts)

accuracy(stacker.predict.rt.Francefatalities3, Francetest2.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 550.8633 | 666.907 | 550.8633 | 56.96634 | 56.96634 | -0.06985188 | 1.235573 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 290.0714 | 458.2773 | 304.1327 | 22.67324 | 25.24261 | -0.1641503 | 0.8698834 |

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| Test set | 32.14389 | 356.2442 | 323.857 | -12.00889 | 39.06577 | -0.1641503 | 0.7031065 |

Overall, considering RMSE scores for the test dataset, the XGBoost ensemble stacker model seems to be the best model.

If we try to forecast for the next two months using this model, we can see that the number of deaths per day caused by Covid-19 should stay constant and be equal to about 833 in early June.

```
stacker.predict.rt.Francefatalities3[62]
```

```
832.998962402344
```

```
# Predict from stacker model_3 --- XGBoost
predict.variables.Francefatalities.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables.Francefatalities)
```

## Appendix 2: Rest of the code

### Appendix 2-A: Understand the data

# Understand the Data

## Italy

```
train <- read.csv("train.csv")
train_Italy = train[which(train$Country_Region=='Italy' & train$Province_State=='' ),]

#Creation of the column diff with the number of new cases per day
train_Italy <- train_Italy %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

n <- nrow(train_Italy)
train_Italy[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff |
|---|---|---|---|---|---|---|---|
| **1** | 15516 | | Italy | 2020-01-22 | 0 | 0 | 0 |
| **2** | 15517 | | Italy | 2020-01-23 | 0 | 0 | 0 |
| **3** | 15518 | | Italy | 2020-01-24 | 0 | 0 | 0 |
| **4** | 15519 | | Italy | 2020-01-25 | 0 | 0 | 0 |
| **73** | 15588 | | Italy | 2020-04-03 | 119827 | 14681 | 4585 |
| **74** | 15589 | | Italy | 2020-04-04 | 124632 | 15362 | 4805 |
| **75** | 15590 | | Italy | 2020-04-05 | 128948 | 15887 | 4316 |
| **76** | 15591 | | Italy | 2020-04-06 | 132547 | 16523 | 3599 |

```
#Create a new column with the number of fatalities per day in Italy
library(dplyr)

train_Italy <- train_Italy %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))

n <- nrow(train_Italy)
train_Italy[c(1:4,(n-3):n),]
```

| | X | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 15516 | NA | Italy | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 2 | 15517 | NA | Italy | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 3 | 15518 | NA | Italy | 2020-01-24 | 0 | 0 | 0 | 0 |
| **4** | 4 | 15519 | NA | Italy | 2020-01-25 | 0 | 0 | 0 | 0 |
| **73** | 73 | 15588 | NA | Italy | 2020-04-03 | 119827 | 14681 | 4585 | 766 |
| **74** | 74 | 15589 | NA | Italy | 2020-04-04 | 124632 | 15362 | 4805 | 681 |
| **75** | 75 | 15590 | NA | Italy | 2020-04-05 | 128948 | 15887 | 4316 | 525 |
| **76** | 76 | 15591 | NA | Italy | 2020-04-06 | 132547 | 16523 | 3599 | 636 |

```
mean(train_Italy$diff_fatalities)
```

217.407894736842

```
mean(train_Italy$diff)
```

1744.03947368421

```
library(lubridate)
```

```
train_Italy = train_Italy %>%
  dplyr::mutate(year = lubridate::year(Date),
                month = lubridate::month(Date),
                day = lubridate::day(Date),)
n <- nrow(train_Italy)
train_Italy[c(1:4,(n-3):n),]
```

| | X | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 15516 | NA | Italy | 2020-01-22 | 0 | 0 | 0 | 0 | 2020 | 1 | 22 |
| **2** | 2 | 15517 | NA | Italy | 2020-01-23 | 0 | 0 | 0 | 0 | 2020 | 1 | 23 |
| **3** | 3 | 15518 | NA | Italy | 2020-01-24 | 0 | 0 | 0 | 0 | 2020 | 1 | 24 |
| **4** | 4 | 15519 | NA | Italy | 2020-01-25 | 0 | 0 | 0 | 0 | 2020 | 1 | 25 |
| **73** | 73 | 15588 | NA | Italy | 2020-04-03 | 119827 | 14681 | 4585 | 766 | 2020 | 4 | 3 |
| **74** | 74 | 15589 | NA | Italy | 2020-04-04 | 124632 | 15362 | 4805 | 681 | 2020 | 4 | 4 |
| **75** | 75 | 15590 | NA | Italy | 2020-04-05 | 128948 | 15887 | 4316 | 525 | 2020 | 4 | 5 |
| **76** | 76 | 15591 | NA | Italy | 2020-04-06 | 132547 | 16523 | 3599 | 636 | 2020 | 4 | 6 |

```
train_Italy$month <- month(train_Italy$Date, label = TRUE, abbr = TRUE, locale = Sys.getlocale("LC_TIME"))
```

```
n <- nrow(train_Italy)
train_Italy[c(1:4,(n-3):n),]
```

| | X | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 15516 | NA | Italy | 2020-01-22 | 0 | 0 | 0 | 0 | 2020 | Jan | 22 |
| **2** | 2 | 15517 | NA | Italy | 2020-01-23 | 0 | 0 | 0 | 0 | 2020 | Jan | 23 |
| **3** | 3 | 15518 | NA | Italy | 2020-01-24 | 0 | 0 | 0 | 0 | 2020 | Jan | 24 |
| **4** | 4 | 15519 | NA | Italy | 2020-01-25 | 0 | 0 | 0 | 0 | 2020 | Jan | 25 |
| **73** | 73 | 15588 | NA | Italy | 2020-04-03 | 119827 | 14681 | 4585 | 766 | 2020 | Apr | 3 |
| **74** | 74 | 15589 | NA | Italy | 2020-04-04 | 124632 | 15362 | 4805 | 681 | 2020 | Apr | 4 |
| **75** | 75 | 15590 | NA | Italy | 2020-04-05 | 128948 | 15887 | 4316 | 525 | 2020 | Apr | 5 |
| **76** | 76 | 15591 | NA | Italy | 2020-04-06 | 132547 | 16523 | 3599 | 636 | 2020 | Apr | 6 |

```
aggregate(train_Italy$diff ~
          train_Italy$month, data = train_Italy, FUN = mean)
```

| train_Italy$month | train_Italy$diff |
|---|---|
| Jan | 0.20000 |
| Feb | 38.82759 |
| Mar | 3376.25806 |
| Apr | 4459.16667 |

```
aggregate(train_Italy$diff_fatalities ~ train_Italy$month, data = train_Italy, FUN = mean)
```

| train_Italy$month | train_Italy$diff_fatalities |
|---|---|
| Jan | 0.0000 |
| Feb | 1.0000 |
| Mar | 399.9677 |
| Apr | 682.5000 |

# United Kingdom

```
train <- read.csv("train.csv")
train_UK = train[which(train$Country_Region=='United Kingdom' & train$Province_State =='' ),]
```

```
#Create a new column with the number of new cases per day in Italy
library(dplyr)

train_UK <- train_UK %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

n <- nrow(train_UK)
train_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff |
|---|---|---|---|---|---|---|---|
| **1** | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 |
| **2** | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 |
| **3** | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 |
| **4** | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 |
| **73** | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 |
| **74** | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 |
| **75** | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 |
| **76** | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 |

```
#Create a new column with the number of fatalities per day in Italy
library(dplyr)

train_UK <- train_UK %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))

n <- nrow(train_UK)
train_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **1** | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 | 0 |
| **4** | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 | 0 |
| **73** | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 | 684 |
| **74** | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 | 708 |
| **75** | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 | 621 |
| **76** | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 | 439 |

```
mean(train_UK$diff)
```

679.052631578947

```
mean(train_UK$diff_fatalities)
```

70.6973684210526

```
library(lubridate)
```

```
train_UK = train_UK %>%
  dplyr::mutate(year = lubridate::year(Date),
                month = lubridate::month(Date),
                day = lubridate::day(Date),)
```

```
train_UK$month <- month(train_UK$Date, label = TRUE, abbr = TRUE, locale = Sys.getlocale("LC_TIME"))
```

```
n <- nrow(train_UK)
train_UK[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 31887 | | United Kingdom | 2020-01-22 | 0 | 0 | 0 | 0 | 2020 | Jan | 22 |
| **2** | 31888 | | United Kingdom | 2020-01-23 | 0 | 0 | 0 | 0 | 2020 | Jan | 23 |
| **3** | 31889 | | United Kingdom | 2020-01-24 | 0 | 0 | 0 | 0 | 2020 | Jan | 24 |
| **4** | 31890 | | United Kingdom | 2020-01-25 | 0 | 0 | 0 | 0 | 2020 | Jan | 25 |
| **73** | 31959 | | United Kingdom | 2020-04-03 | 38168 | 3605 | 4450 | 684 | 2020 | Apr | 3 |
| **74** | 31960 | | United Kingdom | 2020-04-04 | 41903 | 4313 | 3735 | 708 | 2020 | Apr | 4 |
| **75** | 31961 | | United Kingdom | 2020-04-05 | 47806 | 4934 | 5903 | 621 | 2020 | Apr | 5 |
| **76** | 31962 | | United Kingdom | 2020-04-06 | 51608 | 5373 | 3802 | 439 | 2020 | Apr | 6 |

```
aggregate(train_UK$diff ~
          train_UK$month, data = train_UK, FUN = mean)
```

| train_UK$month | train_UK$diff |
|---|---|
| Jan | 0.2000000 |
| Feb | 0.7241379 |
| Mar | 810.5483871 |
| Apr | 4409.6666667 |

```
aggregate(train_UK$diff_fatalities ~ train_UK$month, data = train_UK, FUN = mean)
```

| train_UK$month | train_UK$diff_fatalities |
|---|---|
| Jan | 0.00000 |
| Feb | 0.00000 |
| Mar | 57.70968 |
| Apr | 597.33333 |

# France

```
train_France = train[which(train$Country_Region=='France' & train$Province_State =='' ),]
```

```
#Create a new column with the number of new cases per day in Italy
library(dplyr)

train_France <- train_France %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff |
|---|---|---|---|---|---|---|---|
| **1** | 13055 | | France | 2020-01-22 | 0 | 0 | 0 |
| **2** | 13056 | | France | 2020-01-23 | 0 | 0 | 0 |
| **3** | 13057 | | France | 2020-01-24 | 2 | 0 | 2 |
| **4** | 13058 | | France | 2020-01-25 | 3 | 0 | 1 |
| **73** | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 |
| **74** | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 |
| **75** | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 |
| **76** | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 |

```
#Create a new column with the number of fatalities per day in Italy
library(dplyr)

train_France <- train_France %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))

n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| **1** | 13055 | | France | 2020-01-22 | 0 | 0 | 0 | 0 |
| **2** | 13056 | | France | 2020-01-23 | 0 | 0 | 0 | 0 |
| **3** | 13057 | | France | 2020-01-24 | 2 | 0 | 2 | 0 |
| **4** | 13058 | | France | 2020-01-25 | 3 | 0 | 1 | 0 |
| **73** | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 | 1120 |
| **74** | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 | 1053 |
| **75** | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 | 518 |
| **76** | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 | 833 |

```
mean(train_France$diff)
```

1289.60526315789

```
mean(train_France$diff_fatalities)
```

117.25

```
library(lubridate)

train_France = train_France %>%
  dplyr::mutate(year = lubridate::year(Date),
                month = lubridate::month(Date),
                day = lubridate::day(Date),)
n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 13055 | | France | 2020-01-22 | 0 | 0 | 0 | 0 | 2020 | 1 | 22 |
| **2** | 13056 | | France | 2020-01-23 | 0 | 0 | 0 | 0 | 2020 | 1 | 23 |
| **3** | 13057 | | France | 2020-01-24 | 2 | 0 | 2 | 0 | 2020 | 1 | 24 |
| **4** | 13058 | | France | 2020-01-25 | 3 | 0 | 1 | 0 | 2020 | 1 | 25 |
| **73** | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 | 1120 | 2020 | 4 | 3 |
| **74** | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 | 1053 | 2020 | 4 | 4 |
| **75** | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 | 518 | 2020 | 4 | 5 |
| **76** | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 | 833 | 2020 | 4 | 6 |

```
train_France$month <- month(train_France$Date, label = TRUE, abbr = TRUE, locale = Sys.getlocale("LC_TIME"))

n <- nrow(train_France)
train_France[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 13055 | | France | 2020-01-22 | 0 | 0 | 0 | 0 | 2020 | Jan | 22 |
| **2** | 13056 | | France | 2020-01-23 | 0 | 0 | 0 | 0 | 2020 | Jan | 23 |
| **3** | 13057 | | France | 2020-01-24 | 2 | 0 | 2 | 0 | 2020 | Jan | 24 |
| **4** | 13058 | | France | 2020-01-25 | 3 | 0 | 1 | 0 | 2020 | Jan | 25 |
| **73** | 13127 | | France | 2020-04-03 | 64338 | 6507 | 5233 | 1120 | 2020 | Apr | 3 |
| **74** | 13128 | | France | 2020-04-04 | 89953 | 7560 | 25615 | 1053 | 2020 | Apr | 4 |
| **75** | 13129 | | France | 2020-04-05 | 92839 | 8078 | 2886 | 518 | 2020 | Apr | 5 |
| **76** | 13130 | | France | 2020-04-06 | 98010 | 8911 | 5171 | 833 | 2020 | Apr | 6 |

```
aggregate(train_France$diff ~
          train_France$month, data = train_France, FUN = mean)
```

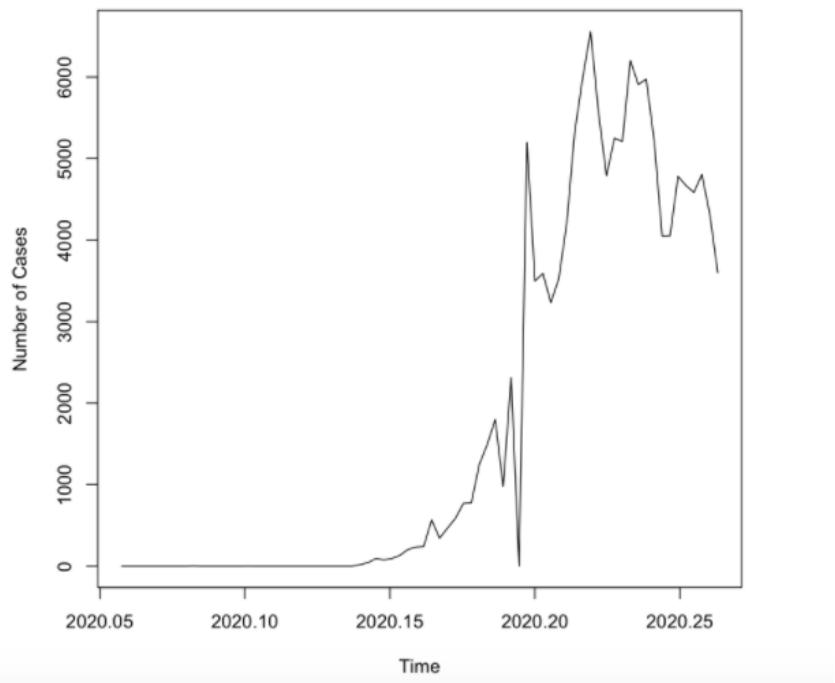
| train_France$month | train_France$diff |
|---|---|
| Jan | 0.500000 |
| Feb | 3.275862 |
| Mar | 1678.322581 |
| Apr | 7647.000000 |

```
aggregate(train_France$diff_fatalities ~ train_France$month, data = train_France, FUN = mean)
```

| train_France$month | train_France$diff_fatalities |
|---|---|
| Jan | 0.00000000 |
| Feb | 0.06896552 |
| Mar | 113.58064516 |
| Apr | 898.00000000 |

## Appendix 2-B: Full code for forecasting the number of cases per day in Italy

```
dl <- seq(from = as.Date("2020-1-22", "%Y-%m-%d"), to = as.Date("2020-4-6", "%Y-%m-%d"), by = "day")
set.seed(25)
myts <- ts(train_Italy$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
plot(myts, ylab = "Number of Cases")
```



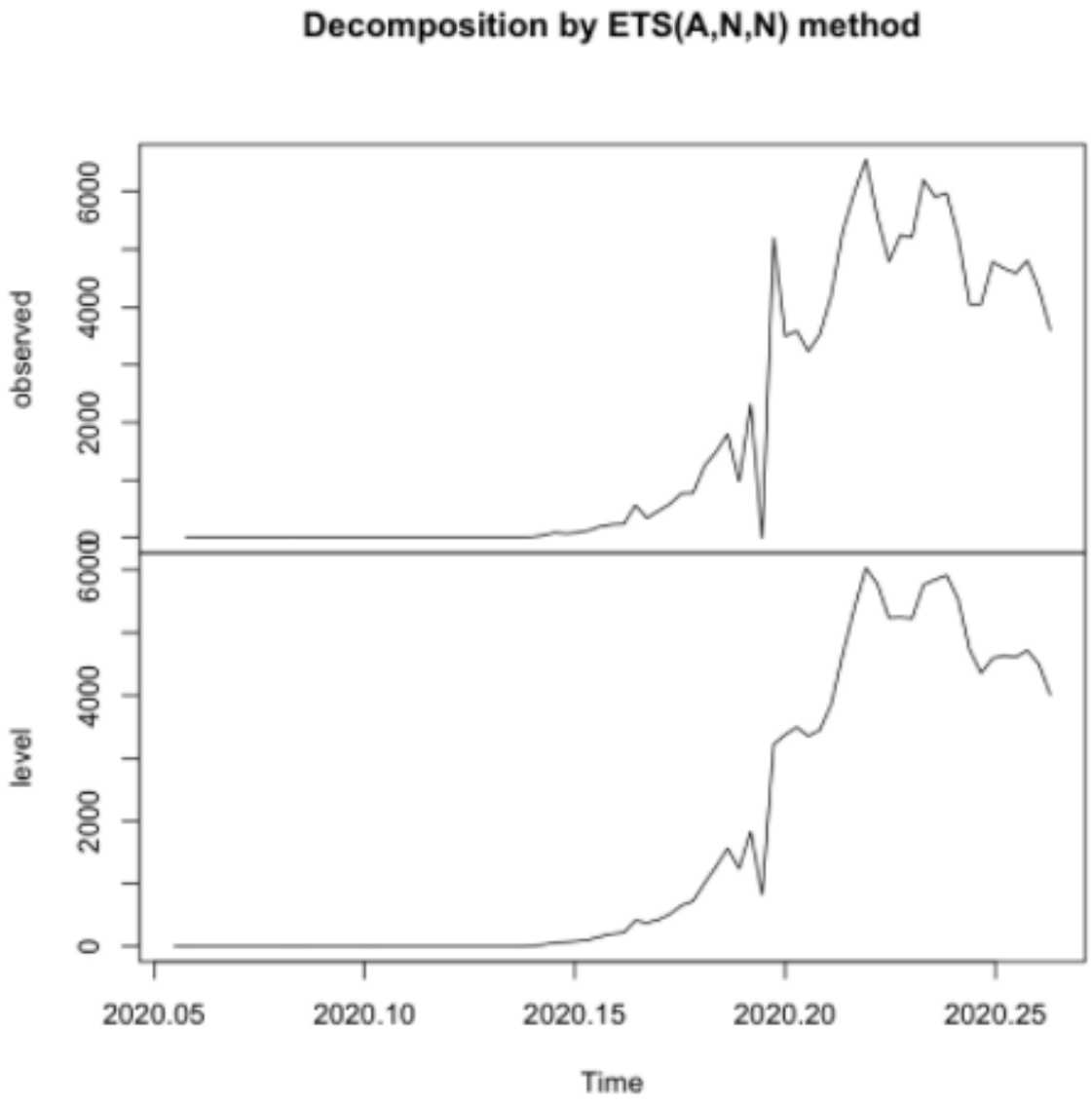

```
#Generate an exponential smoothing ets model

(myts.ANN <- ets(myts, model = "ANN"))
```

```
ETS(A,N,N)

Call:
 ets(y = myts, model = "ANN")

  Smoothing parameters:
    alpha = 0.5473

  Initial states:
    l = -0.0568

  sigma:  712.2709

     AIC     AICc      BIC
1331.515 1331.848 1338.507
```

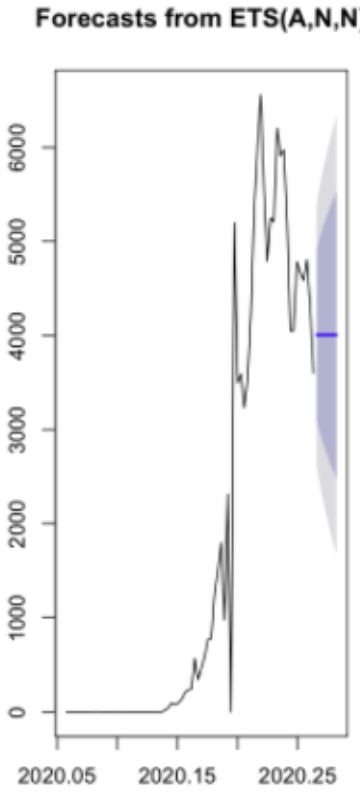
```
#RMSE error for the ets model

rmse.ets <- function (etsmodel) cat("RMSE = ", sqrt(etsmodel$mse))

rmse.ets(myts.ANN)
```

```
RMSE =  702.8364
```

```
#Decomposition plot of fitted ets model¶
plot(myts.ANN)
```




```
#Generate forecasts to April 13th 2020  using the above model
myts.ANN.pred <- forecast(myts.ANN, h = 7)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(myts.ANN.pred)
```




```
#And print the final new cases per day forecasts for May 7th 2020 for each model:
myts.ANN.pred$mean[7]
```

4005.4199357241

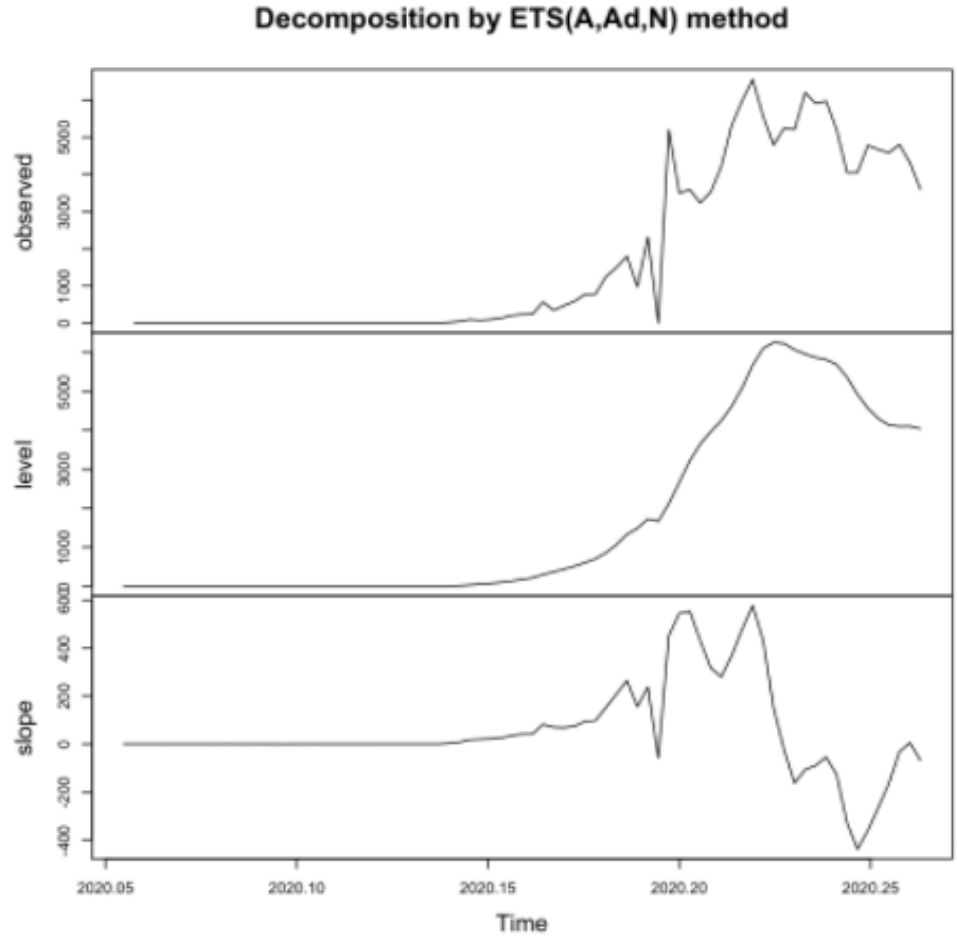
```
#Generate an additive trend (Holt's linear) ets model

(myts.AAN <- ets(myts, model = "AAN"))
```

```
ETS(A,Ad,N)

Call:
 ets(y = myts, model = "AAN")

  Smoothing parameters:
    alpha = 0.1409
    beta  = 0.1409
    phi   = 0.9021

  Initial states:
    l = -0.2626
    b = 0.069

  sigma:  688.9336

     AIC     AICc      BIC
1329.306 1330.523 1343.290
```

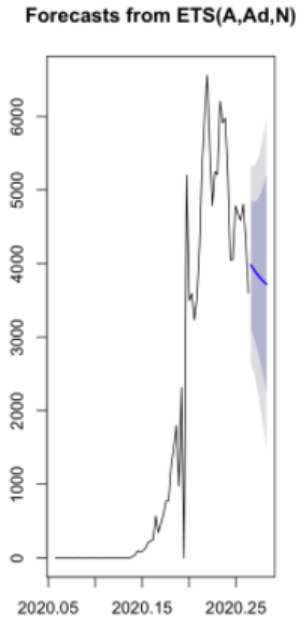
```
rmse.ets(myts.AAN)
```

```
RMSE =  665.8857
```

```
#Decomposition plot of fitted additive trend ets model

plot(myts.AAN)
```




```
#Generate forecast to April 13th 2020  using the above model
myts.AAN.pred <- forecast(myts.AAN, h = 7)

# plot the forecasts for the AAN models
par(mfrow = c(1, 2))
plot(myts.AAN.pred)
```




```
#And print the final new cases per day forecasts for April 13th 2020 for each model:
myts.ANN.pred$mean[7]
myts.AAN.pred$mean[7]
```

```
4005.4199357241
```

```
3719.9811236209
```

```
(diffItaly.ets <- ets(myts))
```

```
ETS(A,Ad,N)

Call:
 ets(y = myts)

  Smoothing parameters:
    alpha = 0.1409
    beta  = 0.1409
    phi   = 0.9021

  Initial states:
    l = -0.2626
    b = 0.069

  sigma:  688.9336

     AIC     AICc      BIC
1329.306 1330.523 1343.290
```

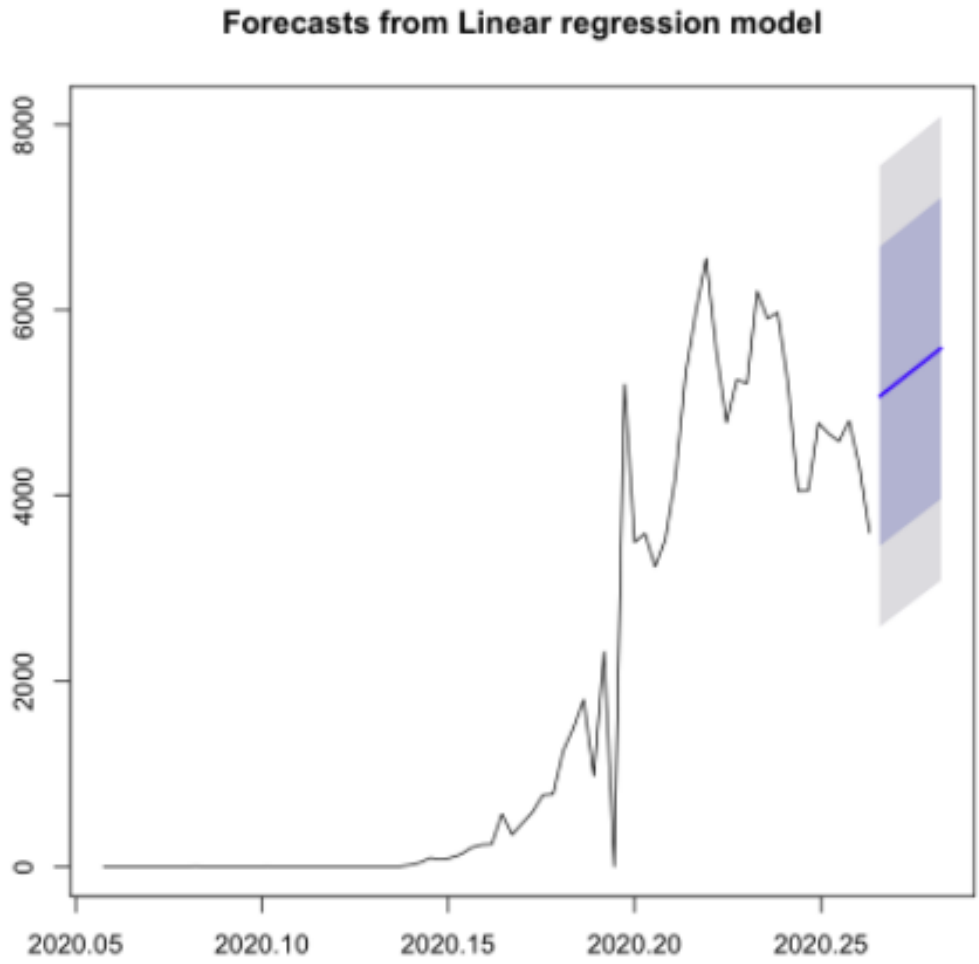
```
# The best fitting model is the AAN model
```

```
#Generate a tslm model
tslm_model_diffItaly <- tslm(myts~ trend)
tslm_model_diffItaly.pred <- forecast(tslm_model_diffItaly, h = 7)
```

```
tslm_model_diffItaly
```

```
Call:
tslm(formula = myts ~ trend)

Coefficients:
(Intercept)        trend
    -1582.5         86.4
```

```
plot(tslm_model_diffItaly.pred)
```




```r
#I want the test data set to contain the Confirmed Cases and Fatalities from the 7th to the 13th of April
test <- read.csv("test.csv")
n <- nrow(test)
test[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities |
|---|---|---|---|---|---|---|
| **1** | 1 | | Afghanistan | 2020-01-22 | 0 | 0 |
| **2** | 2 | | Afghanistan | 2020-01-23 | 0 | 0 |
| **3** | 3 | | Afghanistan | 2020-01-24 | 0 | 0 |
| **4** | 4 | | Afghanistan | 2020-01-25 | 0 | 0 |
| **25976** | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 |
| **25977** | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 |
| **25978** | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 |
| **25979** | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 |

```
#Create a new column with the number of new cases per day
library(dplyr)

test  <- test  %>%
  mutate(diff = ConfirmedCases - lag(ConfirmedCases, default = first(ConfirmedCases)))

#Create a new column with the number of fatalities per day

test  <- test  %>%
  mutate(diff_fatalities = Fatalities - lag(Fatalities, default = first(Fatalities)))


n <- nrow(test )
test [c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | Afghanistan | 2020-01-22 | 0 | 0 | 0 | 0 |
| 2 | 2 | | Afghanistan | 2020-01-23 | 0 | 0 | 0 | 0 |
| 3 | 3 | | Afghanistan | 2020-01-24 | 0 | 0 | 0 | 0 |
| 4 | 4 | | Afghanistan | 2020-01-25 | 0 | 0 | 0 | 0 |
| 25976 | 35648 | | Zimbabwe | 2020-04-10 | 13 | 3 | 2 | 0 |
| 25977 | 35649 | | Zimbabwe | 2020-04-11 | 14 | 3 | 1 | 0 |
| 25978 | 35650 | | Zimbabwe | 2020-04-12 | 14 | 3 | 0 | 0 |
| 25979 | 35651 | | Zimbabwe | 2020-04-13 | 17 | 3 | 3 | 0 |

```
test$Date <- as.Date(test$Date, "%Y-%m-%d")
```

```
#Take values between the 7th and the 13th of April and for country = Italy
test_Italy = test[which(test$Country_Region=='Italy' & test$Province_State =='' & test$Date >= as.Date("2020-4-7") ),]

n <- nrow(test_Italy )
test_Italy[c(1:4,(n-3):n),]
```

| | Id | Province_State | Country_Region | Date | ConfirmedCases | Fatalities | diff | diff_fatalities |
|---|---|---|---|---|---|---|---|---|
| 12195 | 16721 | | Italy | 2020-04-07 | 135586 | 17127 | 3039 | 604 |
| 12196 | 16722 | | Italy | 2020-04-08 | 139422 | 17669 | 3836 | 542 |
| 12197 | 16723 | | Italy | 2020-04-09 | 143626 | 18279 | 4204 | 610 |
| 12198 | 16724 | | Italy | 2020-04-10 | 147577 | 18849 | 3951 | 570 |
| 12198.1 | 16724 | | Italy | 2020-04-10 | 147577 | 18849 | 3951 | 570 |
| 12199 | 16725 | | Italy | 2020-04-11 | 152271 | 19468 | 4694 | 619 |
| 12200 | 16726 | | Italy | 2020-04-12 | 156363 | 19899 | 4092 | 431 |
| 12201 | 16727 | | Italy | 2020-04-13 | 159516 | 20465 | 3153 | 566 |

```
d2 <- seq(from = as.Date("2020-4-7", "%Y-%m-%d"), to = as.Date("2020-4-13", "%Y-%m-%d"), by = "day")
```

```
Italytest.ts <- ts(test_Italy$diff,
           start = c(2020, as.numeric(format(d2[1], "%j"))) ,
           frequency = 365)
```

```
# Evaluate the performance of the model
accuracy(tslm_model_diffItaly.pred, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 6.150051e-16 | 1199.660 | 999.2115 | NaN | Inf | NaN | 0.76557764 | NA |
| Test set | -1.477062e+03 | 1572.026 | 1477.0619 | -41.14607 | 41.14607 | NaN | 0.04214087 | 2.13482 |

```
accuracy(myts.ANN.pred, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 96.30564 | 702.8364 | 317.6979 | NaN | Inf | NaN | -0.08510095 | NA |
| **Test set** | -586.84851 | 1573.1109 | 865.0600 | -Inf | Inf | NaN | 0.04547857 | 0 |

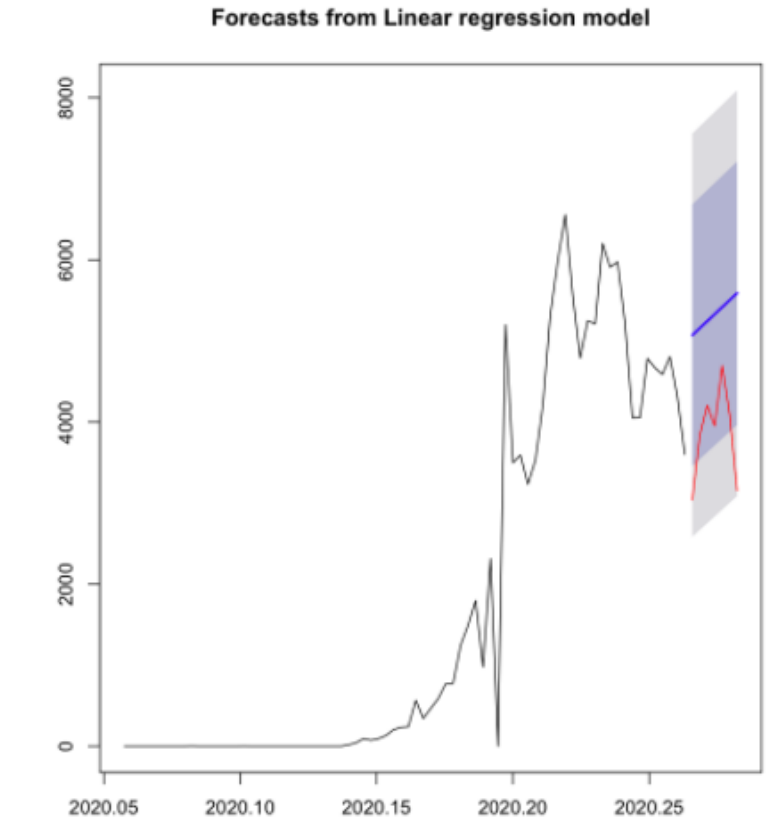
```
accuracy(myts.AAN.pred, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| **Training set** | 31.23647 | 665.8857 | 337.9145 | NaN | Inf | NaN | 0.10275309 | NA |
| **Test set** | -419.65645 | 1568.2239 | 904.0976 | -Inf | Inf | NaN | 0.06776878 | 0 |

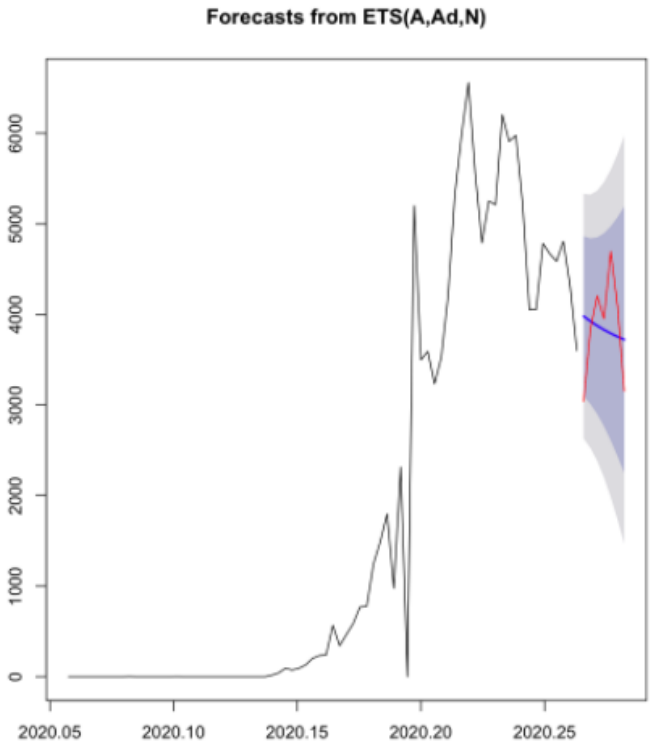
```
#Plot and compare the validation forecasts against the actual validation data

options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))

plot(tslm_model_diffItaly.pred)
lines(Italytest.ts, col="red")  # plots the actual validation data
plot(myts.ANN.pred)
lines(Italytest.ts, col="red")
plot(myts.AAN.pred)
lines(Italytest.ts, col="red")
```


```
#Plot and compare the validation forecasts against the actual validation data
options(repr.plot.width=7, repr.plot.height=8)  # change plot size to 7 x 8

par(mfrow = c(2, 2))
```

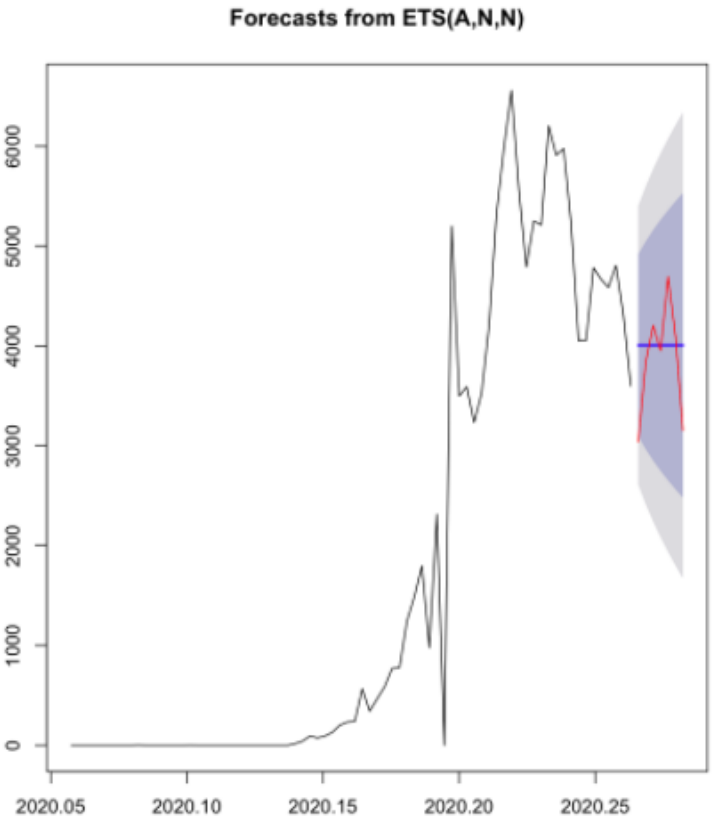
```
plot(tslm_model_diffItaly.pred)
lines(Italytest.ts, col="red")  # plots the actual validation data
```

```
plot(myts.AAN.pred)
lines(Italytest.ts, col="red")
```




```
plot(myts.ANN.pred)
lines(Italytest.ts, col="red")
```



Ensemble Model --- Stacking

```
#Step 1. Split the training set into a smaller training set and a validation set.

train_Italy$Date <- as.Date(train_Italy$Date, "%Y-%m-%d")

smalltrainItaly.df <- train_Italy[which(train_Italy$Date <= as.Date("2020-3-24") ),]

validItaly.df <- train_Italy[which(train_Italy$Date >= as.Date("2020-3-24") ),]
```

```
#Step 2. Train the base models in the small training set and generate predictions in the validation set

myts_smalltrainItalydiff.df <- ts(smalltrainItaly.df$diff,     # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
```

```
#Generate an exponential smoothing ets model (MODEL 1)
(myts_smalltrainItalydiff.df.ANN <- ets(myts_smalltrainItalydiff.df, model = "ANN"))
myts_smalltrainItalydiff.df.ANN.pred <- forecast(myts_smalltrainItalydiff.df.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts_smalltrainItalydiff.df, model = "ANN")

  Smoothing parameters:
    alpha = 0.5258

  Initial states:
    l = -0.0116

  sigma:  726.45

     AIC     AICc      BIC
1095.094 1095.501 1101.524
```

```
#Generate an additive trend (Holt's linear) ets model (Model 2)
(myts_smalltrainItalydiff.df.AAN <- ets(myts_smalltrainItalydiff.df, model = "AAN"))
myts_smalltrainItalydiff.df.AAN.pred <- forecast(myts_smalltrainItalydiff.df.AAN , h = 14)
```

```
ETS(A,A,N)

Call:
 ets(y = myts_smalltrainItalydiff.df, model = "AAN")

  Smoothing parameters:
    alpha = 0.1121
    beta  = 0.1121

  Initial states:
    l = -0.3865
    b = 0.1201

  sigma:  686.3518

     AIC     AICc      BIC
1089.840 1090.893 1100.556
```

```
#Step 3. Fit a stacker model to the predictions generated in Step 2

stacker.df <- data.frame(Date = validItaly.df$Date, diff = validItaly.df$diff,
                         myts_smalltrainItalydiff.df.ANN.pred = myts_smalltrainItalydiff.df.ANN.pred,
                         myts_smalltrainItalydiff.df.AAN.pred = myts_smalltrainItalydiff.df.AAN.pred)
```

```
library(dplyr)

stacker.df <- stacker.df[,c("Date","diff","myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast",
                            "myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast")]

head(stacker.df)
```

| | Date | diff | myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast | myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast |
|---|---|---|---|---|
| 2020.2301 | 2020-03-24 | 5249 | 5248.946 | 6723.002 |
| 2020.2329 | 2020-03-25 | 5210 | 5248.946 | 6762.216 |
| 2020.2356 | 2020-03-26 | 6203 | 5248.946 | 6801.431 |
| 2020.2384 | 2020-03-27 | 5909 | 5248.946 | 6840.645 |
| 2020.2411 | 2020-03-28 | 5974 | 5248.946 | 6879.859 |
| 2020.2438 | 2020-03-29 | 5217 | 5248.946 | 6919.073 |

```
library(randomForest)
```

```
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine
```

```
#Fit a random forest to the predictions as stacker model 1
stackerModel_1 <- randomForest(stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast
                        + stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast,
                  data = stacker.df,  nodesize = 5, importance = TRUE)
```

```
#Fit a Regression tree to the predictions as stacker model_2
library(rpart)
stackerModel_2 <- rpart(stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast
                        + stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast
                        , data= stacker.df, method= "anova")
```

```
#Fit a XGBoost model to the predictions as stacker model_3
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```
library(foreach)
library(parallel)
library(Matrix)
library(readr)
library(caret)
```

```
stacker.df.xgb  <- sparse.model.matrix(~ 0 + ., data = stacker.df[,c(-1,-2)])
stacker.y <- as.vector(stacker.df[,2])
stackerModel_3 <- xgboost(stacker.df.xgb, stacker.y, nrounds=200,
                    verbose = 0)

summary(stackerModel_3)
```

```
                Length Class              Mode
handle               1 xgb.Booster.handle externalptr
raw             100174 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                13 -none-             call
params               1 -none-             list
callbacks            1 -none-             list
feature_names        2 -none-             character
nfeatures            1 -none-             numeric
```

```
summary(stackerModel_1)
```

```
                 Length Class  Mode
call               5    -none- call
type               1    -none- character
predicted         14    -none- numeric
mse              500    -none- numeric
rsq              500    -none- numeric
oob.times         14    -none- numeric
importance         4    -none- numeric
importanceSD       2    -none- numeric
localImportance    0    -none- NULL
proximity          0    -none- NULL
ntree              1    -none- numeric
mtry               1    -none- numeric
forest            11    -none- list
coefs              0    -none- NULL
y                 14    -none- numeric
test               0    -none- NULL
inbag              0    -none- NULL
terms              3    terms  call
```

```
summary(stackerModel_2)
```

```
Call:
rpart(formula = stacker.df$diff ~ stacker.df$myts_smalltrainItalydiff.df.ANN.pred.Point.Fo
    stacker.df$myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast,
    data = stacker.df, method = "anova")
  n= 14

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 14 observations
  mean=4901.429, MSE=559806.5
```

```
summary(stackerModel_3)
```

```
                Length Class              Mode
handle               1 xgb.Booster.handle externalptr
raw             100174 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                13 -none-             call
params               1 -none-             list
callbacks            1 -none-             list
feature_names        2 -none-             character
nfeatures            1 -none-             numeric
```

```r
#Step 4. Re-train the base models using the entire training set and generate predictions

myts <- ts(train_Italy$diff,      # random data
           start = c(2020, as.numeric(format(dl[1], "%j"))) ,
           frequency = 365)
```

```r
#Generate an exponential smoothing ets model
(myts.ANN <- ets(myts, model = "ANN"))
myts.ANN.pred.test <- forecast(myts.ANN, h = 14)
```

```
ETS(A,N,N)

Call:
 ets(y = myts, model = "ANN")

  Smoothing parameters:
    alpha = 0.5473

  Initial states:
    l = -0.0568

  sigma:  712.2709

     AIC     AICc      BIC
1331.515 1331.848 1338.507
```

```r
#Generate an additive trend (Holt's linear) ets model
(myts.AAN <- ets(myts, model = "AAN"))
myts.AAN.pred.test <- forecast(myts.AAN, h = 14)
```

```
ETS(A,Ad,N)

Call:
 ets(y = myts, model = "AAN")

  Smoothing parameters:
    alpha = 0.1409
    beta  = 0.1409
    phi   = 0.9021

  Initial states:
    l = -0.2626
    b = 0.069

  sigma:  688.9336

     AIC     AICc      BIC
1329.306 1330.523 1343.290
```

```r
myts.ANN.pred.test
```

```
          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
2020.2658        4005.42 3092.608 4918.232 2609.3946 5401.445
2020.2685        4005.42 2964.860 5045.980 2414.0203 5596.820
2020.2712        4005.42 2851.164 5159.675 2240.1384 5770.701
2020.2740        4005.42 2747.705 5263.135 2081.9114 5928.928
2020.2767        4005.42 2652.133 5358.707 1935.7458 6075.094
2020.2795        4005.42 2562.878 5447.962 1799.2429 6211.597
2020.2822        4005.42 2478.833 5532.006 1670.7073 6340.133
2020.2849        4005.42 2399.180 5611.660 1548.8880 6461.952
2020.2877        4005.42 2323.294 5687.546 1432.8307 6578.009
2020.2904        4005.42 2250.687 5760.153 1321.7878 6689.052
2020.2932        4005.42 2180.967 5829.872 1215.1605 6795.679
2020.2959        4005.42 2113.816 5897.024 1112.4606 6898.379
2020.2986        4005.42 2048.967 5961.873 1013.2836 6997.556
2020.3014        4005.42 1986.200 6024.639  917.2901 7093.550
```

```r
predict.variables <- data.frame('myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast' = myts.ANN.pred.test,
                                'myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast' = myts.AAN.pred.test)
```

```
#Only select the forecast values
predict.variables <- predict.variables%>% select(1,6)
```

```
predict.variables
```

| | myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast.Point.Forecast | myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast.Point.Forecast |
|---|---|---|
| 2020.2658 | 4005.42 | 3978.527 |
| 2020.2685 | 4005.42 | 3923.631 |
| 2020.2712 | 4005.42 | 3874.109 |
| 2020.2740 | 4005.42 | 3829.435 |
| 2020.2767 | 4005.42 | 3789.134 |
| 2020.2795 | 4005.42 | 3752.778 |
| 2020.2822 | 4005.42 | 3719.981 |
| 2020.2849 | 4005.42 | 3690.395 |
| 2020.2877 | 4005.42 | 3663.705 |
| 2020.2904 | 4005.42 | 3639.627 |
| 2020.2932 | 4005.42 | 3617.907 |
| 2020.2959 | 4005.42 | 3598.312 |
| 2020.2986 | 4005.42 | 3580.636 |
| 2020.3014 | 4005.42 | 3564.690 |

```
# Predict from stacker model_1 --- random forest
stacker.predict.rt1 <- predict(stackerModel_1, predict.variables )
```

```
# Predict from stacker model_2 --- regression tree
stacker.predict.rt2 <- predict(stackerModel_2, predict.variables )
```

```
# Predict from stacker model_3 --- XGBoost
predict.variables.xgb <- sparse.model.matrix(~ 0 + ., data = predict.variables)
```

```
colnames(predict.variables.xgb) =
c('myts_smalltrainItalydiff.df.ANN.pred.Point.Forecast','myts_smalltrainItalydiff.df.AAN.pred.Point.Forecast')
```

```
stacker.predict.rt3 <- predict(stackerModel_3, predict.variables.xgb)
```

```
accuracy(stacker.predict.rt1, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| **Test set** | -1305.419 | 1386.809 | 1305.419 | -36.3328 | 36.3328 | 0.1050087 | 1.804676 |

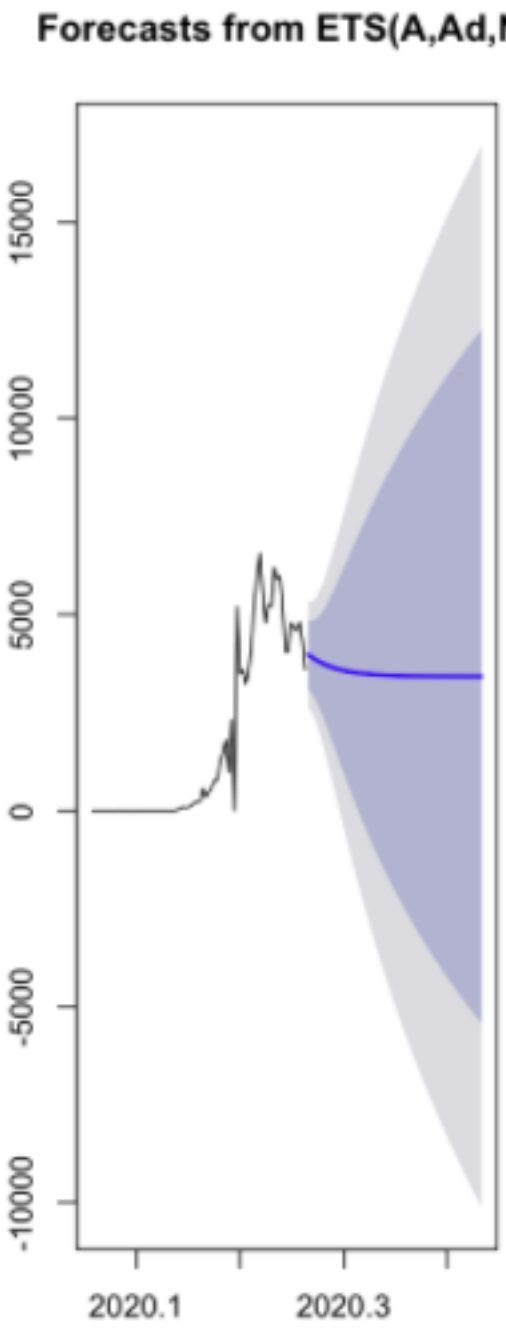
```
accuracy(stacker.predict.rt2, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| **Test set** | -1048.714 | 1180.115 | 1048.714 | -29.90806 | 29.90806 | 0.0774687 | 1.506205 |

```
accuracy(stacker.predict.rt3, Italytest.ts)
```

| | ME | RMSE | MAE | MPE | MAPE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|
| **Test set** | -1396.285 | 1497.492 | 1396.285 | -39.1201 | 39.1201 | 0.0774687 | 1.978775 |

```
#Generate forecasts to June 6th 2020  using the above model
myts.AAN.pred2months <- forecast(myts.AAN, h = 62)

# plot the forecasts for the ANN model
par(mfrow = c(1, 2))
plot(myts.AAN.pred2months)
```




```
myts.AAN.pred2months$mean[62]
```

3416.78888356871

## Resources


- Kaggle.com. 2020. *COVID19 Global Forecasting (Week 3)* | *Kaggle*. [online] Available at: https://www.kaggle.com/c/covid19-global-forecasting-week-3/data
- GitHub. 2020. *Cssegisanddata/COVID-19*. [online] Available at: https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series