```r
# Loading relevant libraries
library(readxl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(TTR)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
# Importing the death data from excel
uk_death <- read_excel('Vital statistics in the UK.xlsx', sheet = 3, skip = 5)
```

```r
# Checking the head of the data frame
head(uk_death)
```

```
## # A tibble: 6 x 7
##    Year Number of deaths: United~1 Number of deaths: En~2 Number of deaths: En~3
##   <dbl> <chr>                                       <dbl> <chr>
## 1  2021 667479                                     586334 549349
## 2  2020 689629                                     607922 569700
## 3  2019 604707                                     530841 496370
## 4  2018 616014                                     541589 505859
## 5  2017 607172                                     533253 498882
## 6  2016 597206                                     525048 490791
## # i abbreviated names: 1: `Number of deaths: United Kingdom`,
## #   2: `Number of deaths: England and Wales`, 3: `Number of deaths: England`
## # i 3 more variables: `Number of deaths: Wales` <chr>,
## #   `Number of deaths : Scotland` <chr>,
## #   `Number of deaths: Northern Ireland` <chr>
```

```r
# Selecting columns needed (Year,Number of deaths: United Kingdom)
death_uk <- uk_death %>%
  select(Year,`Number of deaths: United Kingdom`)
```

```r
# Checking the first 6 entries.
head(death_uk)
```

```
## # A tibble: 6 x 2
##    Year 'Number of deaths: United Kingdom'
##   <dbl> <chr>
## 1  2021 667479
## 2  2020 689629
## 3  2019 604707
## 4  2018 616014
## 5  2017 607172
## 6  2016 597206
```

```r
# Checking the last 6 entries.
tail(death_uk)
```

```
## # A tibble: 6 x 2
##    Year 'Number of deaths: United Kingdom'
##   <dbl> <chr>
## 1  1843 :
## 2  1842 :
## 3  1841 :
## 4  1840 :
## 5  1839 :
## 6  1838 :
```

Notice that some observations are ':'

```r
# Checking the structure of the data frame
str(death_uk)
```

```
## tibble [184 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Year                            : num [1:184] 2021 2020 2019 2018 2017 ...
##  $ Number of deaths: United Kingdom: chr [1:184] "667479" "689629" "604707" "616014" ...
```

from the above, the 'Number of deaths: United Kingdom' column is stored as 'chr'

```r
# Cleaning and preparing the data for time series analysis.
death_uk <- death_uk %>%
  rename(no_of_deaths =`Number of deaths: United Kingdom`) %>%
  filter(no_of_deaths != ':') %>%
  arrange(Year) %>%
  select(no_of_deaths)
```

```r
# Converting the data to a time series
death_uk$no_of_deaths <- as.integer(death_uk$no_of_deaths)
death_uk_ts = ts(death_uk, frequency = 1, start = 1887)
death_uk_ts
```

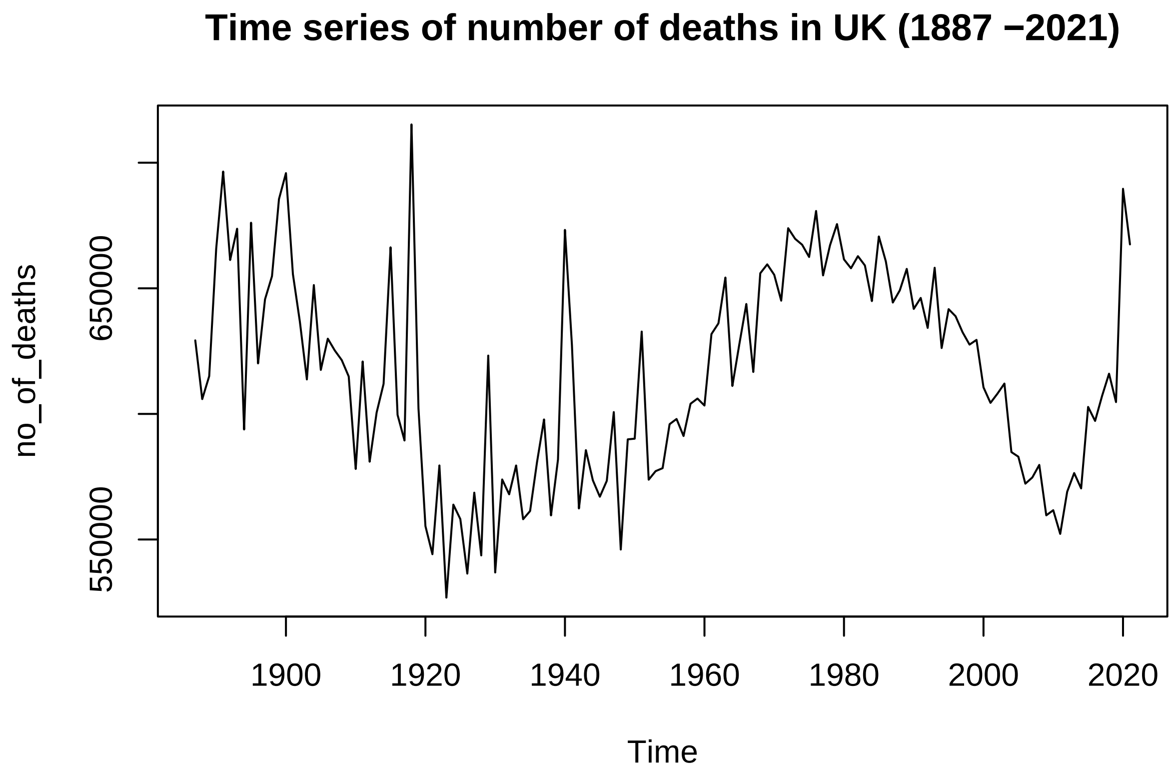
```
## Time Series:
## Start = 1887
## End = 2021
## Frequency = 1
##        no_of_deaths
```

```
##  [1,]      629287
##  [2,]      605899
##  [3,]      615033
##  [4,]      665758
##  [5,]      696490
##  [6,]      661273
##  [7,]      673722
##  [8,]      593808
##  [9,]      676110
## [10,]      620108
## [11,]      645630
## [12,]      654812
## [13,]      685510
## [14,]      695867
## [15,]      655646
## [16,]      636650
## [17,]      613726
## [18,]      651301
## [19,]      617516
## [20,]      629955
## [21,]      625271
## [22,]      621427
## [23,]      614910
## [24,]      578091
## [25,]      620868
## [26,]      580977
## [27,]      600554
## [28,]      611970
## [29,]      666322
## [30,]      599621
## [31,]      589416
## [32,]      715246
## [33,]      602188
## [34,]      555326
## [35,]      544140
## [36,]      579480
## [37,]      526858
## [38,]      563891
## [39,]      558132
## [40,]      536411
## [41,]      568655
## [42,]      543664
## [43,]      623231
## [44,]      536860
## [45,]      573908
## [46,]      567986
## [47,]      579467
## [48,]      558072
## [49,]      561324
## [50,]      580942
## [51,]      597798
## [52,]      559598
## [53,]      581857
## [54,]      673253
```

```
##  [55,]        627378
##  [56,]        562356
##  [57,]        585582
##  [58,]        573570
##  [59,]        567027
##  [60,]        573361
##  [61,]        600728
##  [62,]        546002
##  [63,]        589876
##  [64,]        590136
##  [65,]        632786
##  [66,]        573806
##  [67,]        577220
##  [68,]        578400
##  [69,]        595916
##  [70,]        597981
##  [71,]        591200
##  [72,]        604040
##  [73,]        606115
##  [74,]        603328
##  [75,]        631788
##  [76,]        636051
##  [77,]        654288
##  [78,]        611130
##  [79,]        627798
##  [80,]        643754
##  [81,]        616710
##  [82,]        655998
##  [83,]        659537
##  [84,]        655385
##  [85,]        645078
##  [86,]        673938
##  [87,]        669692
##  [88,]        667359
##  [89,]        662477
##  [90,]        680799
##  [91,]        655143
##  [92,]        667177
##  [93,]        675576
##  [94,]        661519
##  [95,]        657974
##  [96,]        662801
##  [97,]        659101
##  [98,]        644918
##  [99,]        670656
## [100,]        660735
## [101,]        644342
## [102,]        649178
## [103,]        657733
## [104,]        641799
## [105,]        646181
## [106,]        634238
## [107,]        658194
## [108,]        626222
```

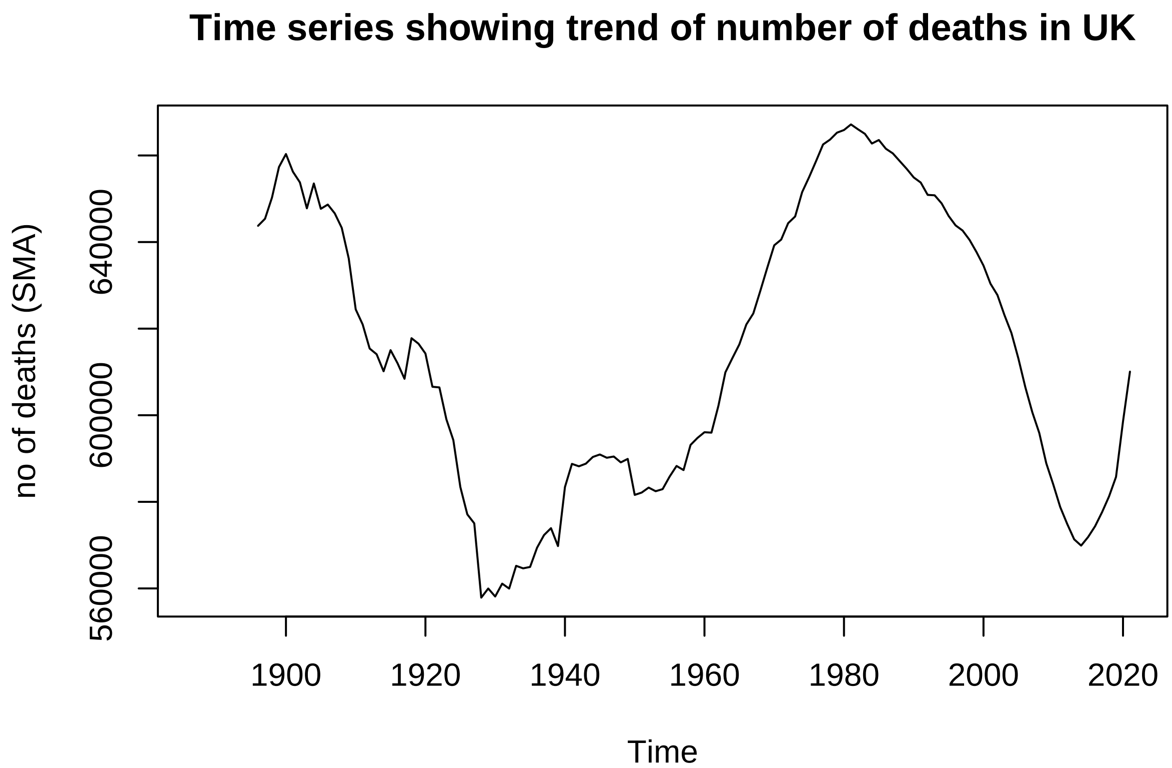
```
## [109,]       641712
## [110,]       638879
## [111,]       632517
## [112,]       627592
## [113,]       629476
## [114,]       610579
## [115,]       604393
## [116,]       608045
## [117,]       612085
## [118,]       584791
## [119,]       582964
## [120,]       572224
## [121,]       574687
## [122,]       579697
## [123,]       559617
## [124,]       561666
## [125,]       552232
## [126,]       569024
## [127,]       576458
## [128,]       570341
## [129,]       602782
## [130,]       597206
## [131,]       607172
## [132,]       616014
## [133,]       604707
## [134,]       689629
## [135,]       667479
```

```r
# Plotting the initial number of deaths from year 1887 - 2021
plot.ts(death_uk_ts, main='Time series of number of deaths in UK (1887 -2021)')
```



The time series appears non seasonal and can probably be described using an additive model. Time series is non seasonal, but has trend and irregular components.

```r
# Estimating trend by smoothing using a simple moving average of order 10.
plot.ts(SMA(death_uk_ts, n=10),
        main='Time series showing trend of number of deaths in UK',
        ylab = 'no of deaths (SMA)')
```



TIME SERIES MODELLING

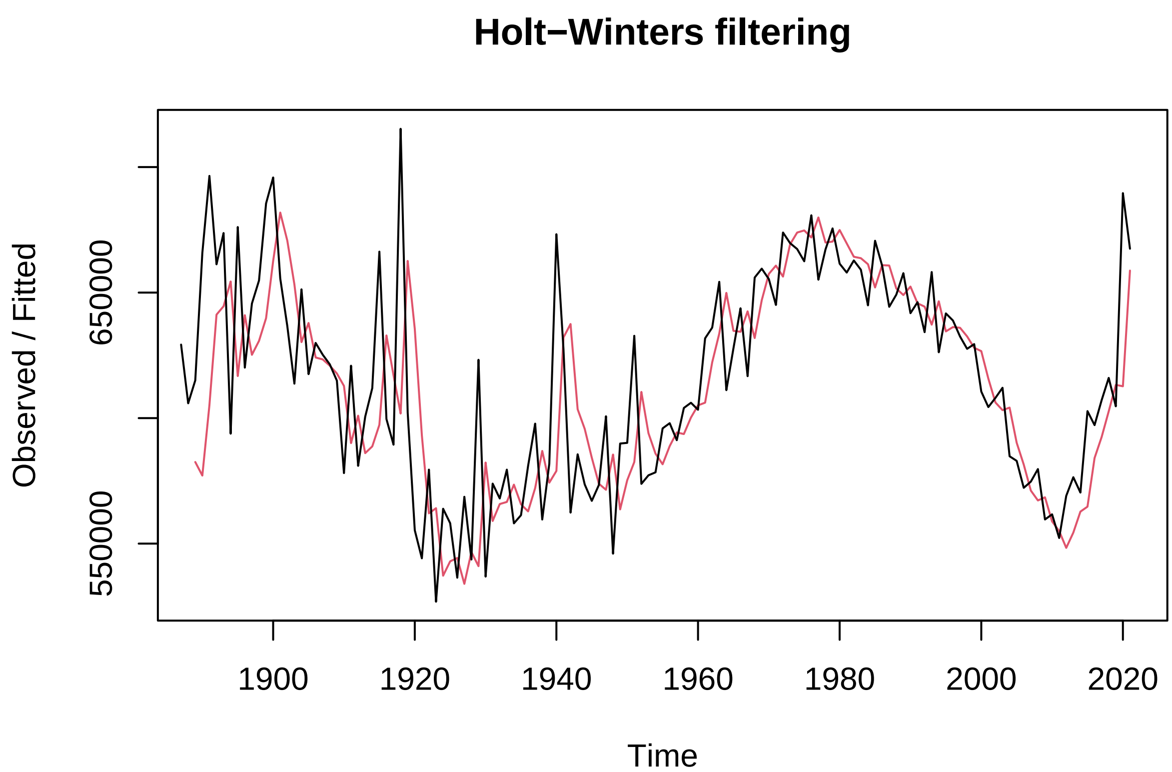
MODEL 1 -FORECASTING USING SMOOTHING The time series can be described by an additive model, it has trend with no seasonality, therefore: We can use Holt's Exponential Smoothing.

```
# Fitting a predictive model using Holt-Winters
death_uk_ts_forcast <- HoltWinters(death_uk_ts, gamma = FALSE)
death_uk_ts_forcast
```

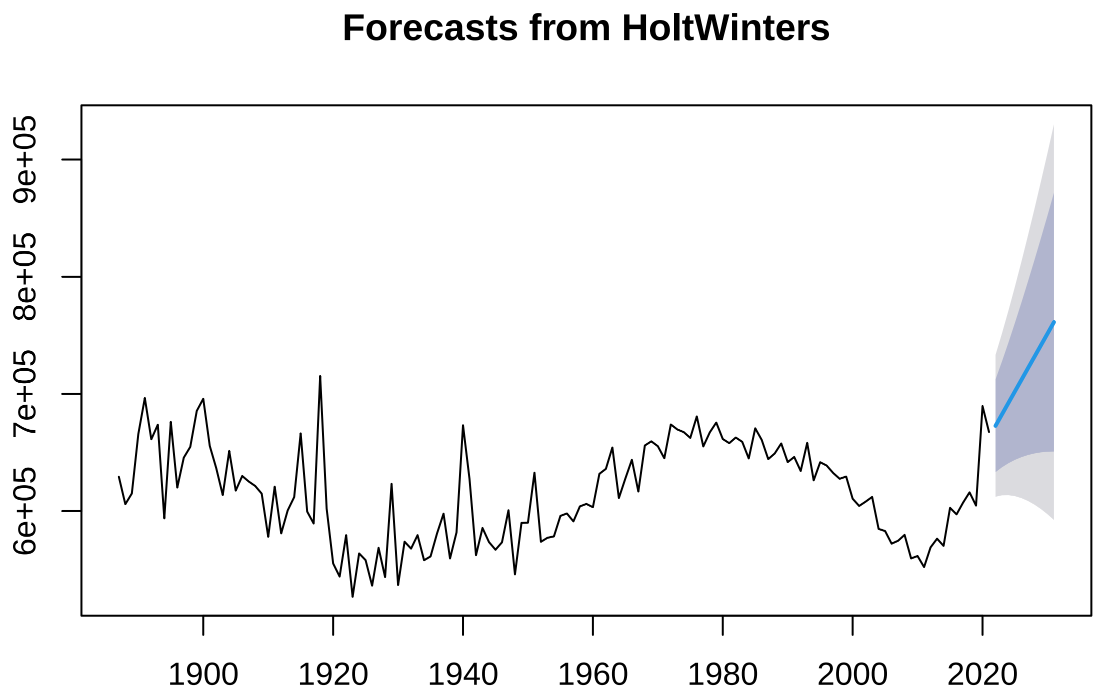
```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = death_uk_ts, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.4796249
##  beta : 0.1534181
##  gamma: FALSE
##
## Coefficients:
##          [,1]
## a 662949.045
## b   9829.915
```

An alpha value approximately 0.48, is just right in the middle of 0 and 1, which means that 48% of the weight is given to the most recent observation when estimating the level. A beta value of 0.15 means more weight (85%) is given to the previous trend estimate (not the most recent).

```r
# plotting both observed and fitted data from HoltWinters forecast.
plot(death_uk_ts_forcast)
```




```r
# forecasting the next 10 years.
death_uk_ts_forcast2 <- forecast(death_uk_ts_forcast, h=10)
plot(death_uk_ts_forcast2)
```



The forecast in blue. The purple area is the 80% prediction interval The grey area is the 95% prediction interval

```
# forcasted data with the 80% and 85% intervals.
death_uk_ts_forcast2
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2022       672779.0 633216.8 712341.1 612273.9 733284.0
## 2023       682608.9 637396.5 727821.3 613462.5 751755.3
## 2024       692438.8 640872.7 744004.9 613575.2 771302.3
## 2025       702268.7 643729.7 760807.7 612741.0 791796.4
## 2026       712098.6 646035.3 778161.9 611063.5 813133.7
## 2027       721928.5 647843.1 796014.0 608624.6 835232.5
## 2028       731758.4 649195.5 814321.4 605489.3 858027.6
## 2029       741588.4 650126.4 833050.3 601709.4 881467.3
## 2030       751418.3 650663.6 852173.0 597327.2 905509.3
## 2031       761248.2 650829.5 871666.9 592377.4 930119.0
```

```
# Sum of square error
death_uk_ts_forcast$SSE
```

```
## [1] 127326590931
```

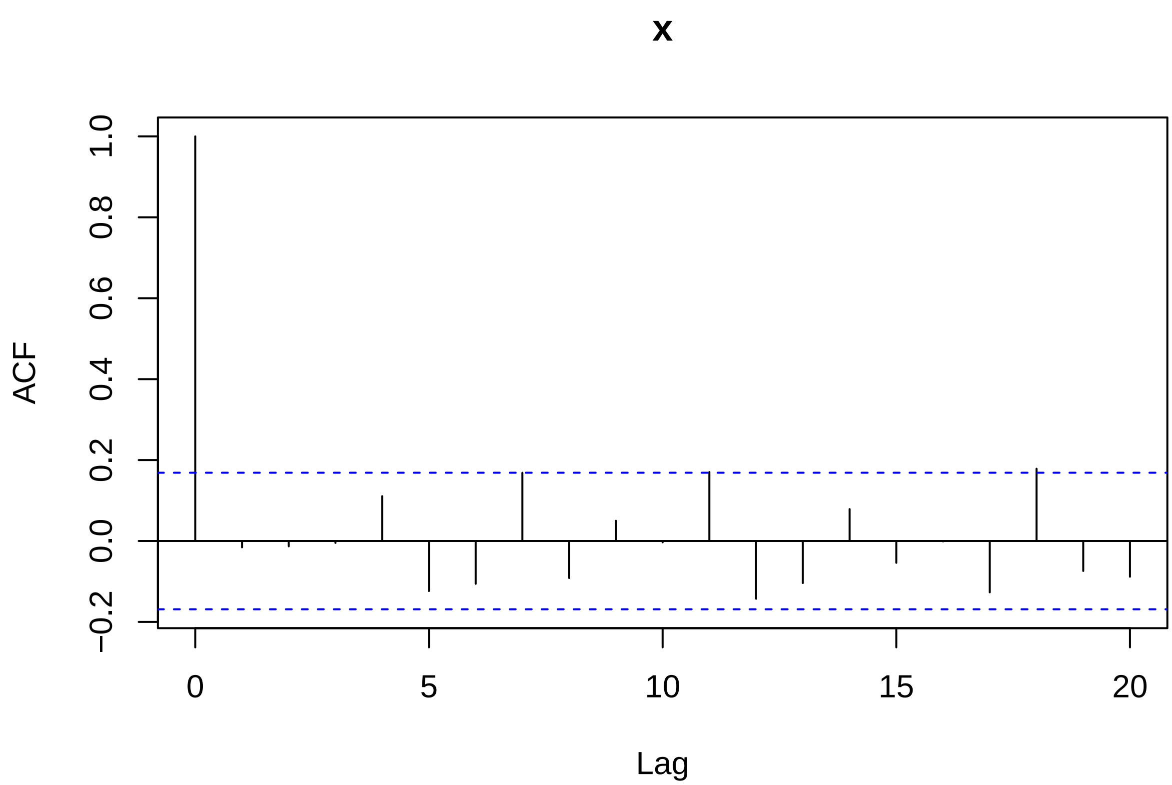
```r
# Root Mean Square Error for Holt-Winters
RMSE_HW = sqrt(mean(death_uk_ts_forcast2$residuals^2, na.rm = TRUE))
RMSE_HW
```

```
## [1] 30940.96
```

```r
# Mean absolute percentage error MAPE for Holt-Winters
MAPE_HW = mean((abs(death_uk_ts_forcast2$residuals/death_uk_ts)*100), na.rm=TRUE)
MAPE_HW
```

```
## [1] 3.537774
```

```r
# ACF and Ljung box test
acf(death_uk_ts_forcast2$residuals, lag.max=20 , na.action = na.pass)
```

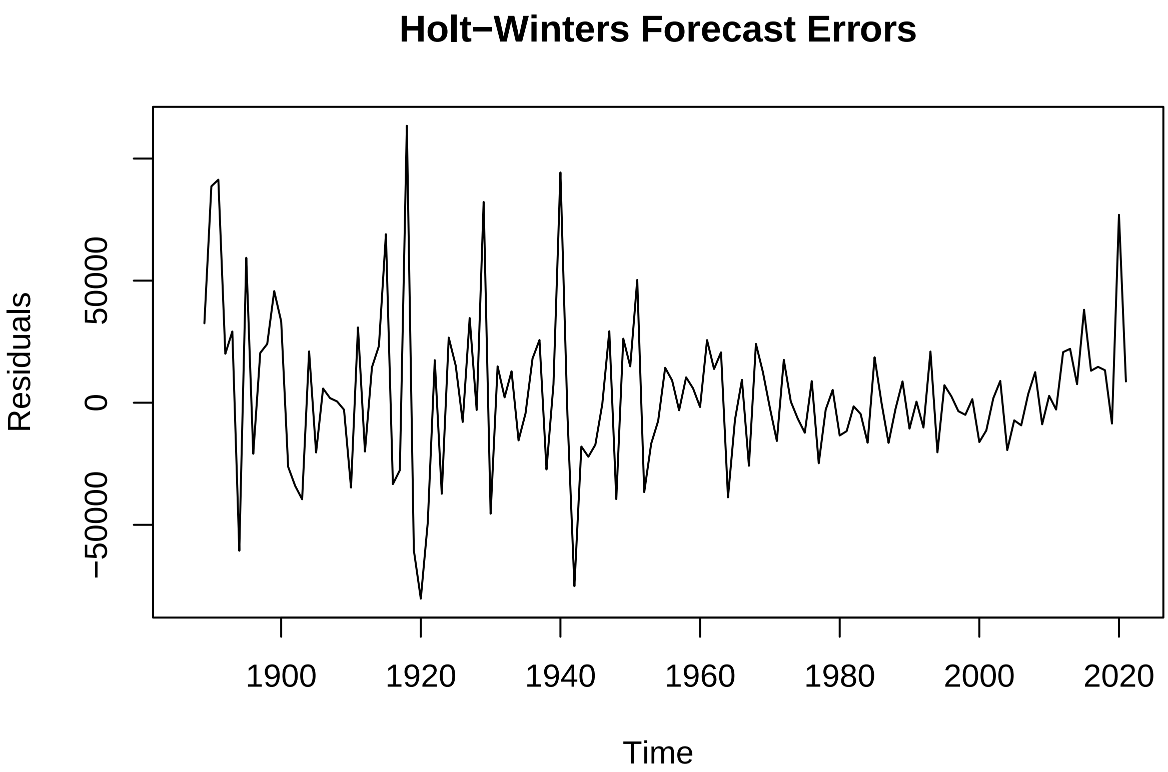



```r
Box.test(death_uk_ts_forcast2$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  death_uk_ts_forcast2$residuals
## X-squared = 31.015, df = 20, p-value = 0.055
```

The P-value for the LJung-test is 0.055, there is little evidence of non-zero auto correlations in the in-sample forecast errors at lags 1-20.

```
#  plotting the forecast errors to check for constant variance
plot.ts(death_uk_ts_forcast2$residuals, main='Holt-Winters Forecast Errors',
        ylab = 'Residuals')
```

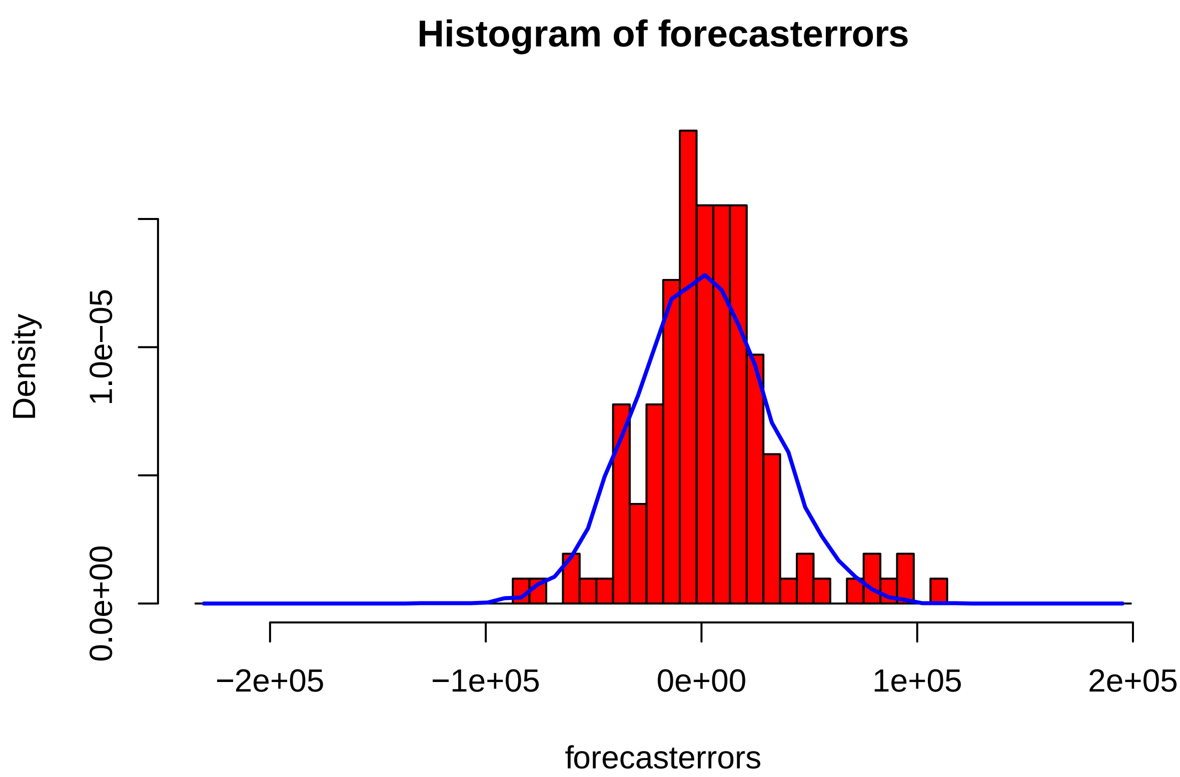



```
# function to plot forecast errors and overlay normal distributed data
plotForecastErrors <- function(forecasterrors)
{
# make a histogram of the forecast errors:
mybinsize <- IQR(forecasterrors)/4
mysd <- sd(forecasterrors)
mymin <- min(forecasterrors) - mysd*5
mymax <- max(forecasterrors) + mysd*3
# generate normally distributed data with mean 0 and standard deviation mysd
mynorm <- rnorm(10000, mean=0, sd=mysd)
mymin2 <- min(mynorm)
mymax2 <- max(mynorm)
if (mymin2 < mymin) { mymin <- mymin2 }
if (mymax2 > mymax) { mymax <- mymax2 }
# make a red histogram of the forecast errors, with the normally distributed data overlaid:
mybins <- seq(mymin, mymax, mybinsize)
hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
# freq=FALSE ensures the area under the histogram = 1
# generate normally distributed data with mean 0 and standard deviation mysd
```

```
myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
# plot the normal curve as a blue line on top of the histogram of forecast errors:
points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
}
```

```
# removing NA values from the residuals
death_uk_ts_forcast2$residuals <- death_uk_ts_forcast2$residuals[!is.na(death_uk_ts_forcast2$residuals)]
```

```
# plotting if the forecast errors to check if normally distributed with mean zero
plotForecastErrors(death_uk_ts_forcast2$residuals)
```




```
# library to import Augmented Dickey-Fuller Test
library(tseries)
```

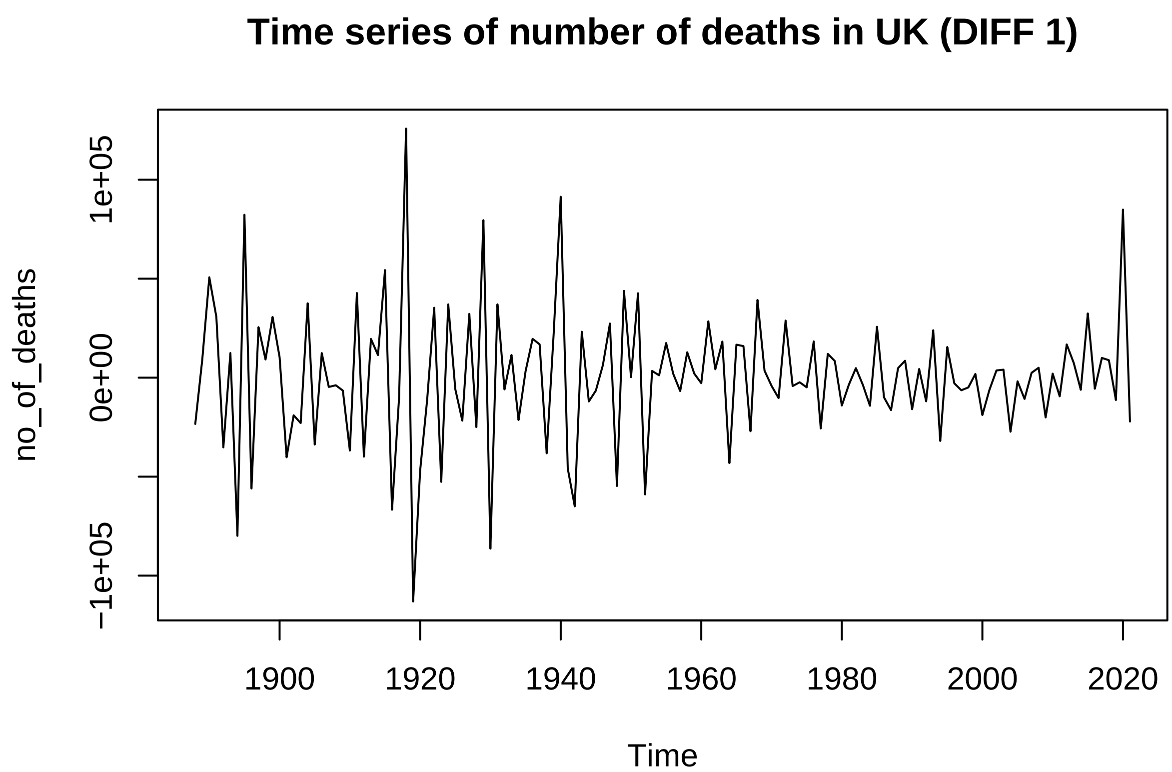
```
# Augmented Dickey-Fuller Test
adf.test(death_uk_ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  death_uk_ts
## Dickey-Fuller = -2.3315, Lag order = 5, p-value = 0.4386
## alternative hypothesis: stationary
```

Test if series is stationary, P-value is greater than 0.05, therefore we fail to reject null hypothesis.

```
# Differencing the time series to make it stationary
death_uk_ts_diff1 <- diff(death_uk_ts, differences = 1)

# Plotting the series with difference 1.
plot(death_uk_ts_diff1, main='Time series of number of deaths in UK (DIFF 1)')
```



The plot appears stationary in mean

```
# Augmented Dickey-Fuller Test for difference 1
adf.test(death_uk_ts_diff1)
```

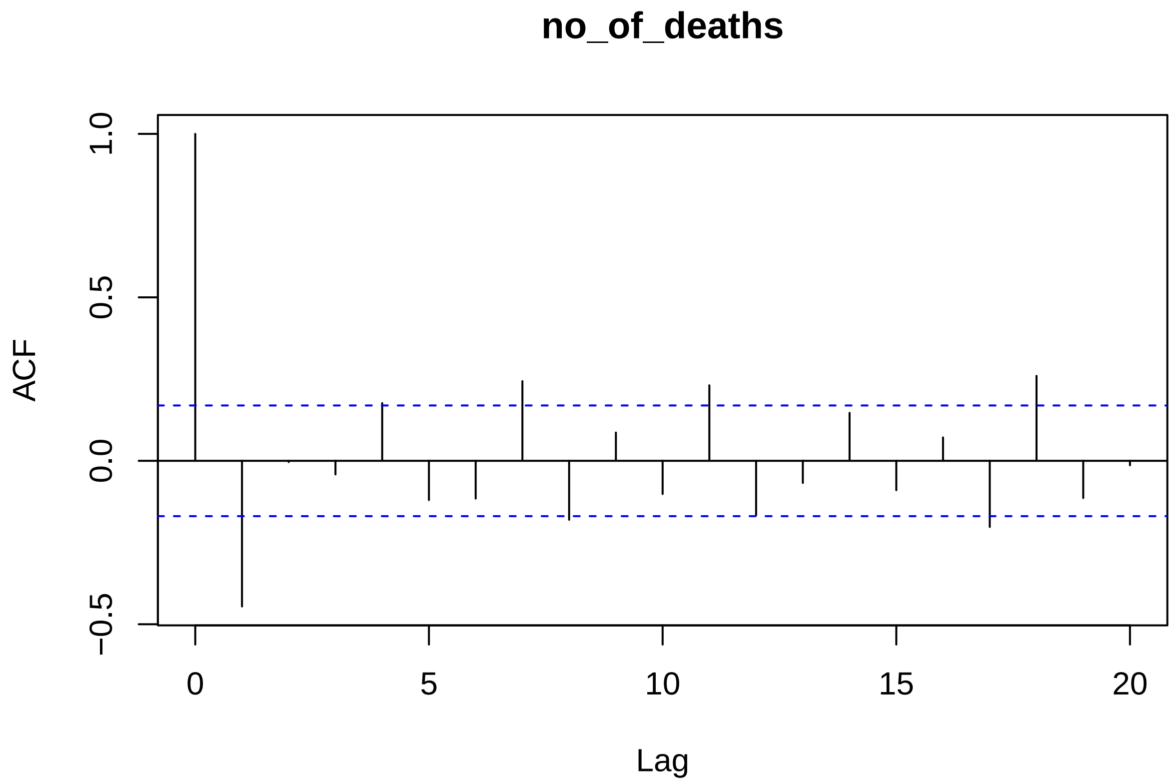
```
## Warning in adf.test(death_uk_ts_diff1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  death_uk_ts_diff1
## Dickey-Fuller = -6.1896, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Test if series is stationary, P-value is less than 0.05, therefore we reject null hypothesis. Difference 1 is stationary.

SELECTING ARIMA MODEL.

```r
# Plotting the correlogram for diff1
acf(death_uk_ts_diff1, lag.max = 20)
```

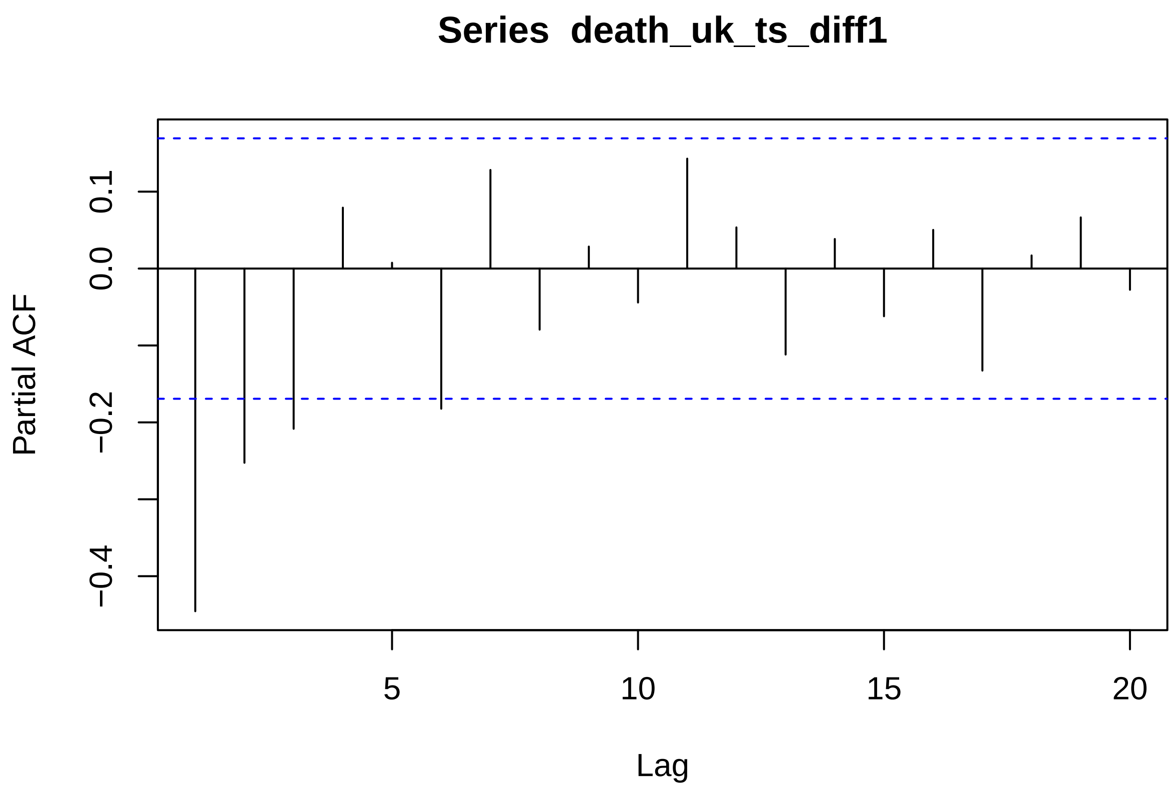



```r
acf(death_uk_ts_diff1, lag.max = 20, plot = FALSE)
```

```
##
## Autocorrelations of series 'death_uk_ts_diff1', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000 -0.446 -0.004 -0.042  0.177 -0.120 -0.116  0.244 -0.180  0.086 -0.102
##     11     12     13     14     15     16     17     18     19     20
##  0.231 -0.166 -0.068  0.147 -0.090  0.072 -0.202  0.260 -0.114 -0.014
```

from the correlogram, the autocorrelation at lag 1 (-0.446) exceeds the significance bounds. so a Moving average model of order 1 - ARMA(0,1) can be used which is also a ARIMA(0,1,1) with difference 1.

```r
# Plotting the partial correlogram for diff1
pacf(death_uk_ts_diff1, lag.max = 20)
```




```r
pacf(death_uk_ts_diff1, lag.max = 20, plot = FALSE)
```

```
##
## Partial autocorrelations of series 'death_uk_ts_diff1', by lag
##
##      1      2      3      4      5      6      7      8      9     10     11
## -0.446 -0.253 -0.208  0.079  0.008 -0.182  0.128 -0.079  0.029 -0.044  0.143
##     12     13     14     15     16     17     18     19     20
##  0.054 -0.112  0.038 -0.062  0.050 -0.133  0.017  0.067 -0.028
```

The partial autocorrelation at lags 1,2,and 3 exceeds the significance bounds. an Auto regressive model of order 3 is possible. ARIMA(3,1,0)

From the principle of parsimony (fewer is better).

MODEL 2 - MOVING AVERAGE MODEL OF ORDER 1 - ARIMA(0,1,1)

```r
# Moving average model of order 1 and difference 1.
death_uk_ts_ma <- arima(death_uk_ts, order = c(0,1,1))
death_uk_ts_ma
```

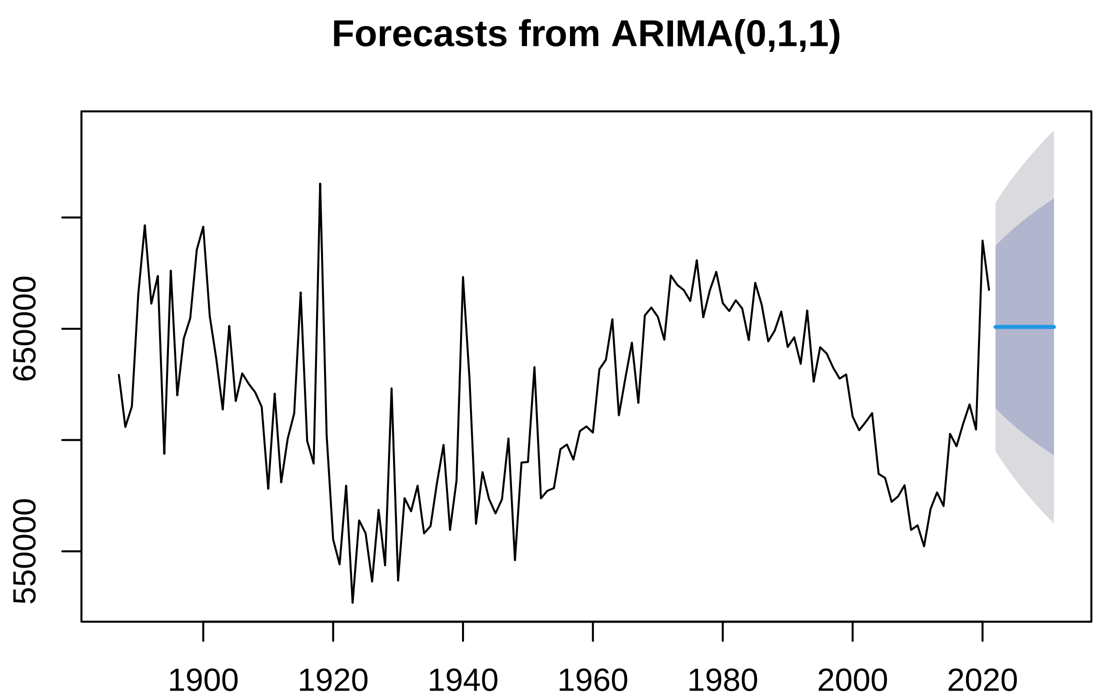
```
##
## Call:
## arima(x = death_uk_ts, order = c(0, 1, 1))
##
```

```
## Coefficients:
##           ma1
##       -0.5913
## s.e.   0.0695
##
## sigma^2 estimated as 812067647:  log likelihood = -1564.86,  aic = 3133.73
```

```r
# forecasting the next 10 years using moving average.
death_uk_ts_ma_forecast <- forecast(death_uk_ts_ma, h =10)
death_uk_ts_ma_forecast
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2022         650827 614306.9 687347.1 594974.3 706679.7
## 2023         650827 611374.2 690279.8 590489.2 711164.8
## 2024         650827 608645.0 693009.1 586315.1 715338.9
## 2025         650827 606081.9 695572.2 582395.2 719258.8
## 2026         650827 603657.8 697996.2 578688.0 722966.1
## 2027         650827 601352.4 700301.6 575162.2 726491.9
## 2028         650827 599149.8 702504.3 571793.5 729860.6
## 2029         650827 597037.2 704616.8 568562.6 733091.4
## 2030         650827 595004.6 706649.5 565454.0 736200.1
## 2031         650827 593043.4 708610.6 562454.6 739199.5
```

```r
# 10 year forecast plot
plot(death_uk_ts_ma_forecast)
```



```r
# Evaluation for ARIMA(0,1,1)
AIC(death_uk_ts_ma)
```

```
## [1] 3133.728
```

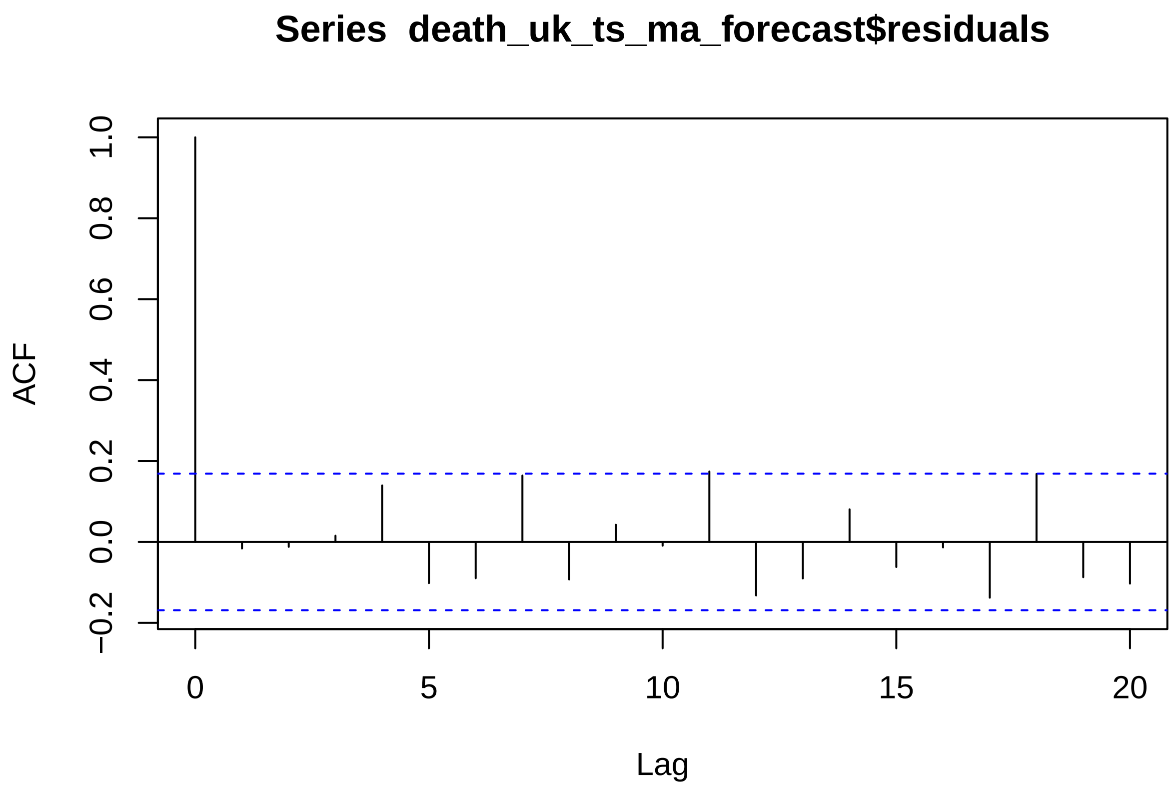
```r
BIC(death_uk_ts_ma)
```

```
## [1] 3139.524
```

```r
accuracy(death_uk_ts_ma)
```

```
##                    ME     RMSE      MAE         MPE     MAPE     MASE
## Training set 456.7854 28391.11 20199.13 -0.09581522 3.280053 0.858626
##                     ACF1
## Training set -0.01616067
```

```r
# ACF and Ljung box test
acf(death_uk_ts_ma_forecast$residuals, lag.max=20 , na.action = na.pass)
```



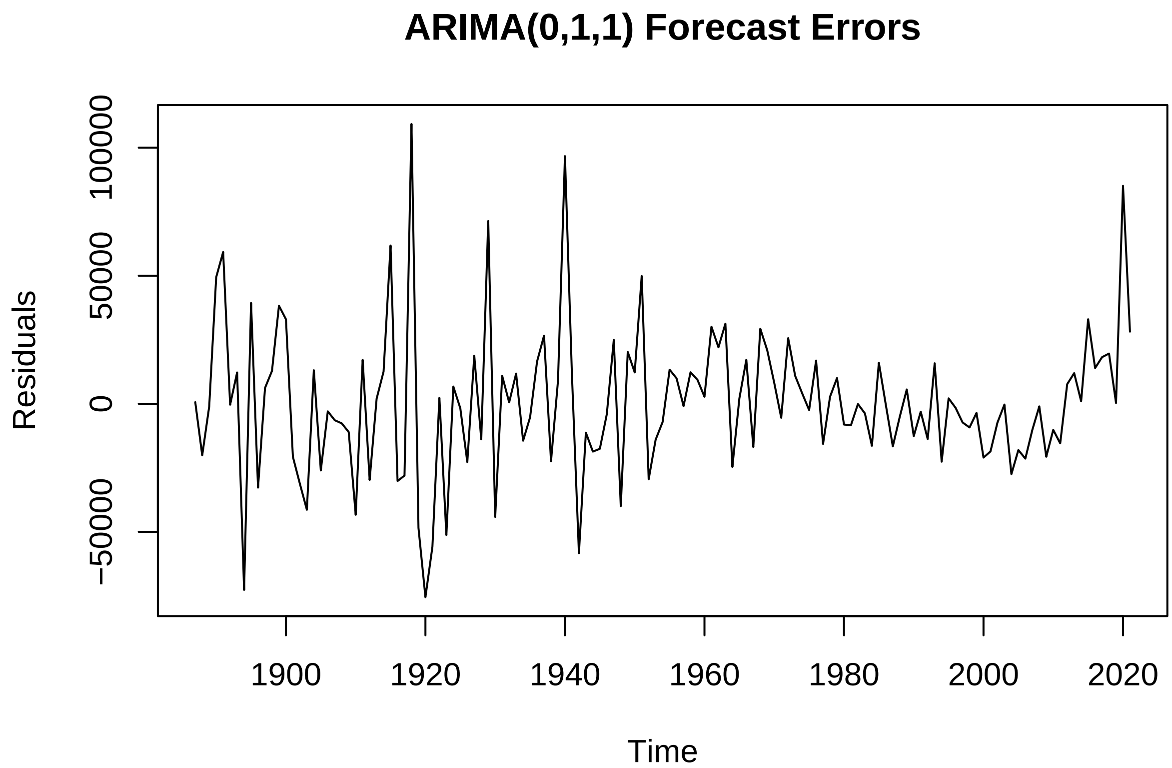

```r
Box.test(death_uk_ts_ma_forecast$residuals, lag=20, type="Ljung-Box")
```

```
##
```

```
##  Box-Ljung test
##
## data:  death_uk_ts_ma_forecast$residuals
## X-squared = 31.243, df = 20, p-value = 0.05206
```

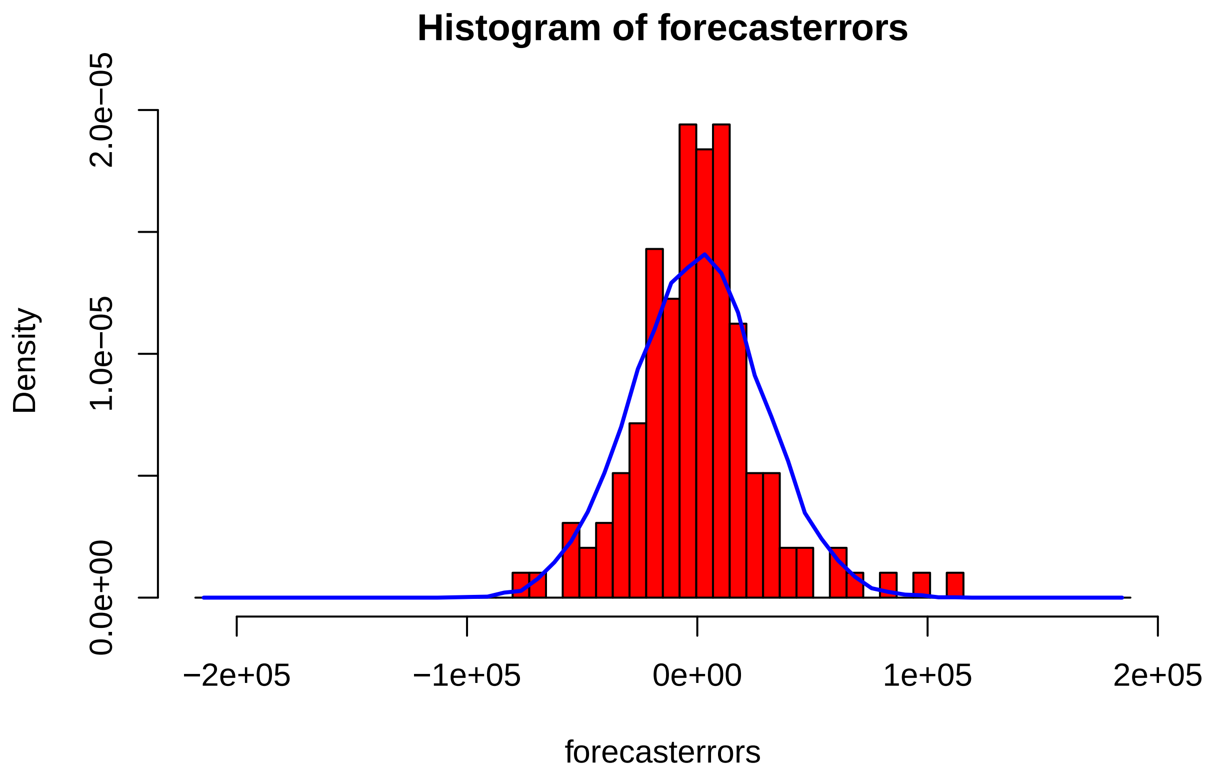
The P-value for the LJung-test is 0.052, there is little evidence of non-zero auto correlations in the forecast errors at lags 1-20.

```
#  plotting the forecast errors to check for constant variance
plot.ts(death_uk_ts_ma_forecast$residuals, main = 'ARIMA(0,1,1) Forecast Errors',
        ylab = 'Residuals')
```



Appears to have mean 0 and constant Variance

```
# plotting if the forecast errors to check if normally distributed with mean zero
plotForecastErrors(death_uk_ts_ma_forecast$residuals)
```



Appears normally distributed with mean 0

MODEL 3 - ARIMA(3,1,0)

```
# ARIMA model (3,1,0)
death_uk_ts_ar <- arima(death_uk_ts, order = c(3,1,0))
death_uk_ts_ar
```

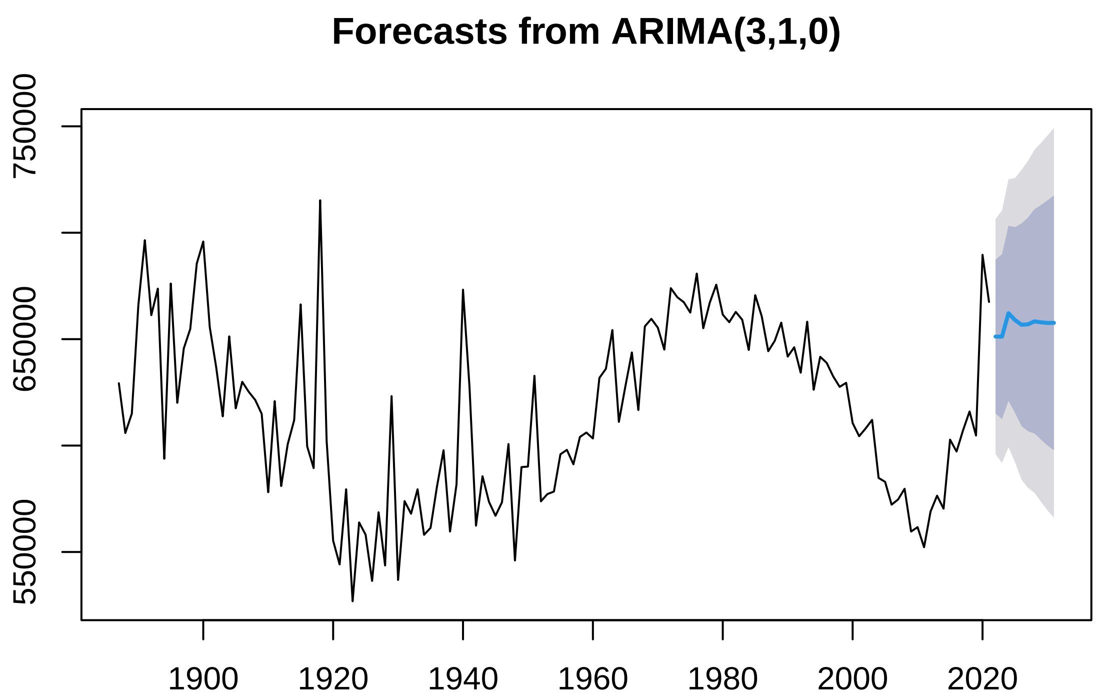
```
##
## Call:
## arima(x = death_uk_ts, order = c(3, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3
##       -0.6108  -0.3798  -0.2166
## s.e.   0.0844   0.0968   0.0871
##
## sigma^2 estimated as 794265312:  log likelihood = -1563.41,  aic = 3134.83
```

```
# forecasting the next 10 years.
death_uk_ts_ar_forecast <- forecast(death_uk_ts_ar, h =10)
death_uk_ts_ar_forecast
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2022       651205.7 615088.1 687323.3 595968.6 706442.8
## 2023       651163.3 612406.4 689920.1 591889.8 710436.8
```

```
## 2024       662167.0 621021.6 703312.5 599240.6 725093.5
## 2025       658987.0 615355.4 702618.6 592258.2 725715.7
## 2026       656759.6 609186.9 704332.2 584003.5 729515.6
## 2027       656944.3 606743.0 707145.7 580168.0 733720.7
## 2028       658366.2 605682.6 711049.8 577793.6 738938.8
## 2029       657910.0 602803.2 713016.9 573631.4 742188.7
## 2030       657608.6 600025.6 715191.7 569543.0 745674.3
## 2031       657658.0 597808.2 717507.8 566125.6 749190.4
```

```
# 10 year forecast plot
plot(death_uk_ts_ar_forecast)
```




```
# Evaluation for ARIMA(3,1,0)
AIC(death_uk_ts_ar)
```

```
## [1] 3134.827
```

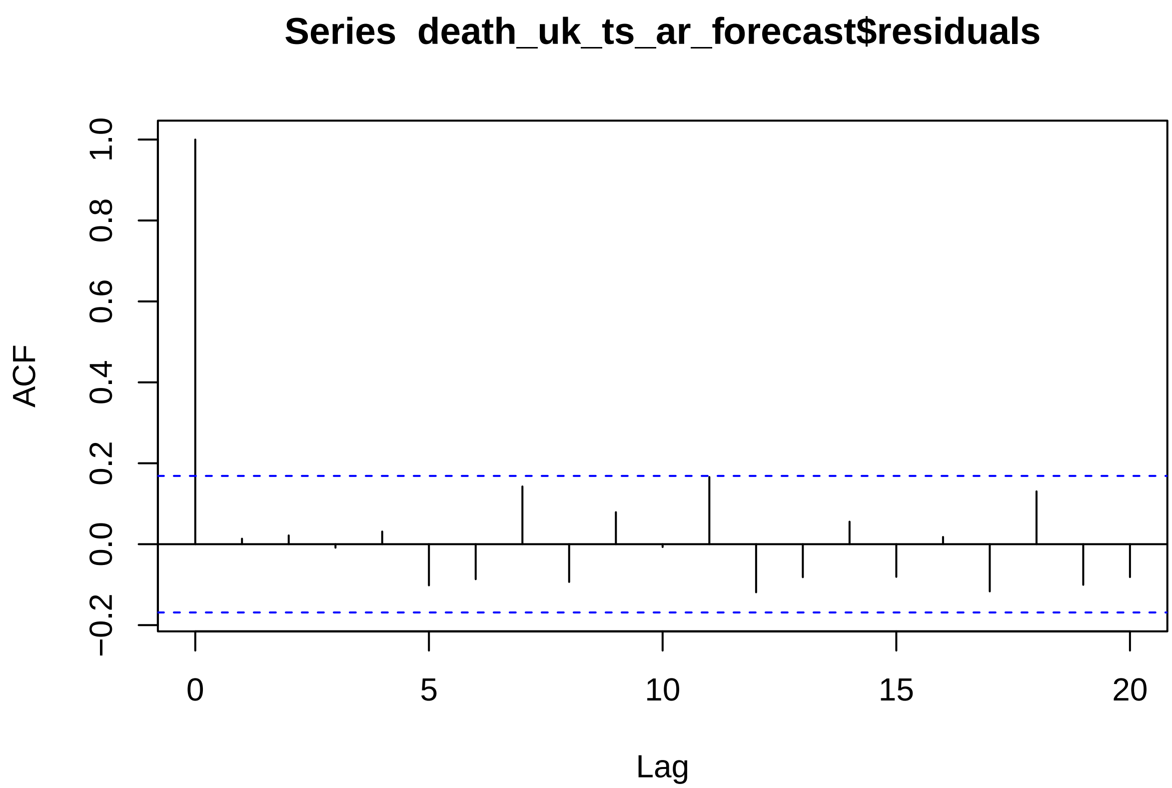
```
BIC(death_uk_ts_ar)
```

```
## [1] 3146.418
```

```
accuracy(death_uk_ts_ar)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 535.029 28078.19 20038.88 -0.07547298 3.256082 0.8518145
##                    ACF1
## Training set 0.01360795
```

```r
# ACF and Ljung box test
acf(death_uk_ts_ar_forecast$residuals, lag.max=20 , na.action = na.pass)
```

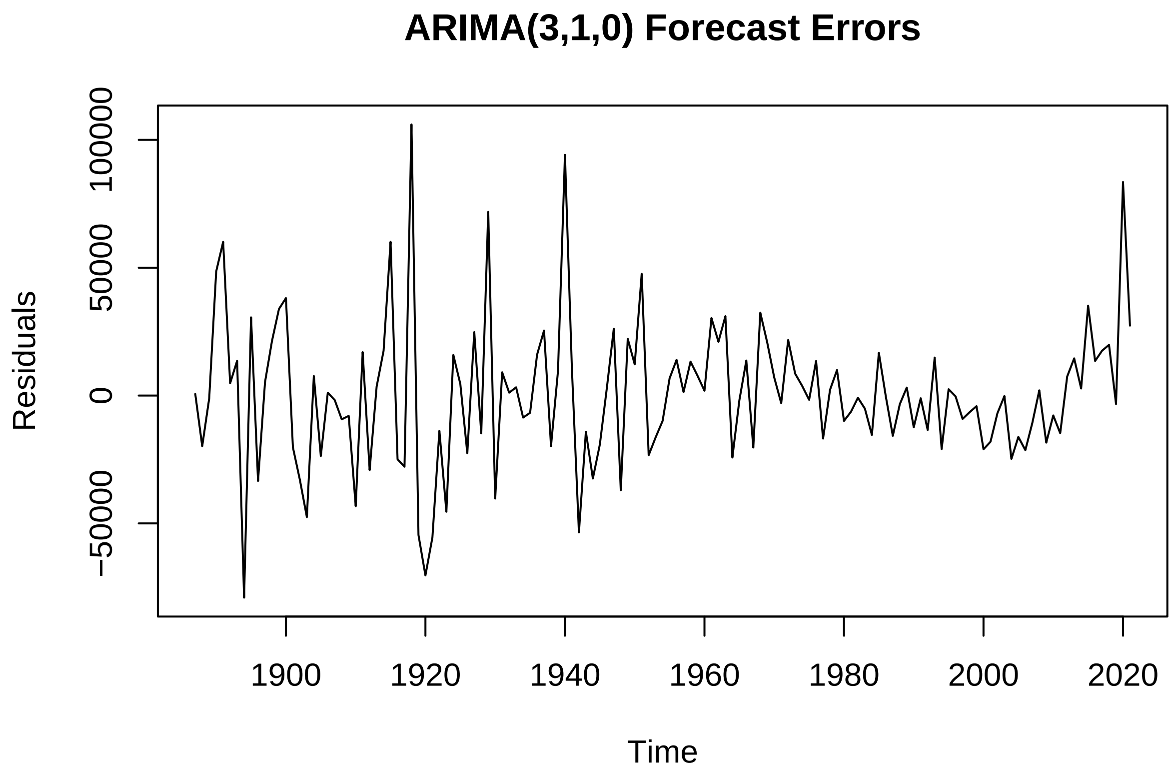



```r
Box.test(death_uk_ts_ar_forecast$residuals, lag=20, type="Ljung-Box")
```

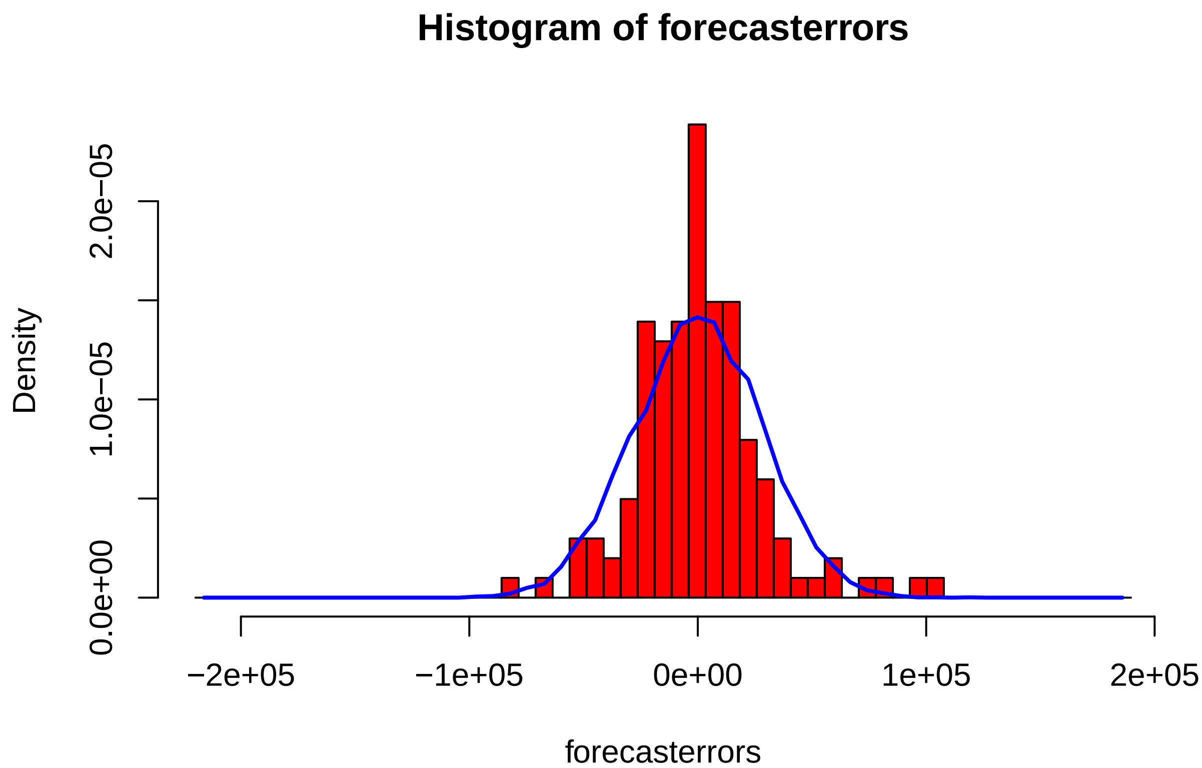
```
##
##  Box-Ljung test
##
## data:  death_uk_ts_ar_forecast$residuals
## X-squared = 24.19, df = 20, p-value = 0.2342
```

The P-value for the LJung-test is 0.057, there is little evidence of non-zero auto correlations in the forecast errors at lags 1-20.

```r
# plotting the forecast errors to check for constant variance
plot.ts(death_uk_ts_ar_forecast$residuals,main ='ARIMA(3,1,0) Forecast Errors',
        ylab = 'Residuals')
```




```
# plotting if the forecast errors to check if normally distributed with mean zero
plotForecastErrors(death_uk_ts_ar_forecast$residuals)
```



Histogram of forecasterrors
Density
0.0e+00
1.0e−05
2.0e−05
−2e+05
−1e+05
0e+00
1e+05
2e+05
forecasterrors