Time Series Forcasting report for total industry

Kevork Sulahian

2021-04-22

```r
library(readxl)
library(forecast)
```

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

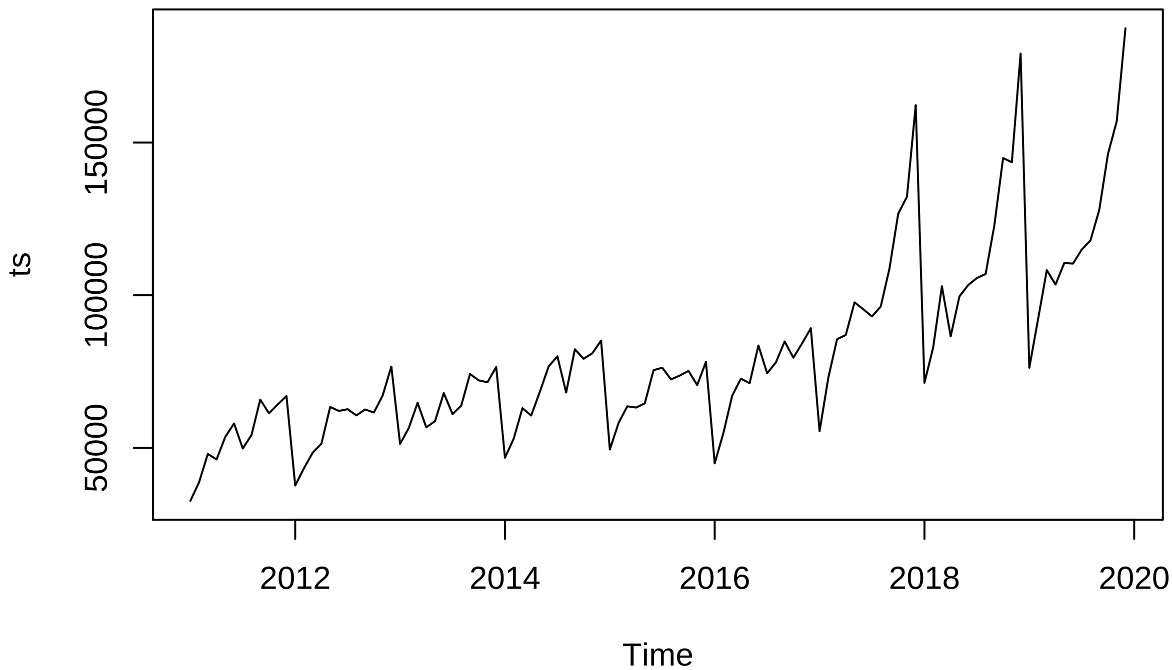
```r
df <- read_xls('economy.xls', sheet='2011-2019 NACE 2')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * `` -> ...6
## * ...
```

```r
df = df[11,]
df = df[-c(1,3)]
rownames(df) = df[1]
```

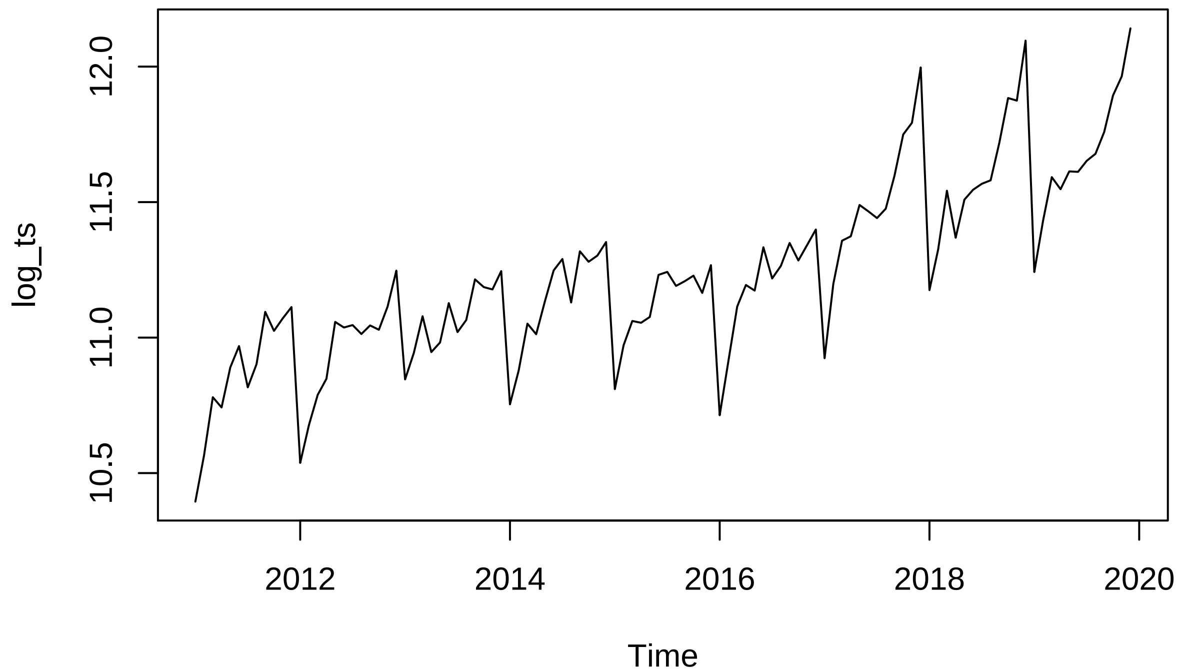
```
## Warning: Setting row names on a tibble is deprecated.
```

```r
df = df[-1]
df = t(df)
df[110] = "155770.7"
df[] <- sapply(df[],function(x) as.numeric(as.character(x)))
df = as.numeric(df)
# df= df * 1000000
df = df[-c(109:120)]
ts = ts(df, start = c(2011,1), frequency = c(12))
```



In this case, it appears that an additive model is not appropriate for describing this time series, since the size of the seasonal fluctuations and random fluctuations seem to increase with the level of the time series. Thus, we may need to transform the time series in order to get a transformed time series that can be described using an additive model. For example, we can transform the time series by calculating the natural log of the original data:

```
log_ts <- log(ts)
plot.ts(log_ts)
```



##Decomposing Time Series

Decomposing a time series means separating it into its constituent components, which are usually a trend component and an irregular component, and if it is a seasonal time series, a seasonal component.

###Decomposing Seasonal Data A seasonal time series consists of a trend component, a seasonal component and an irregular component. Decomposing the time series means separating the time series into these three components: that is, estimating these three components.

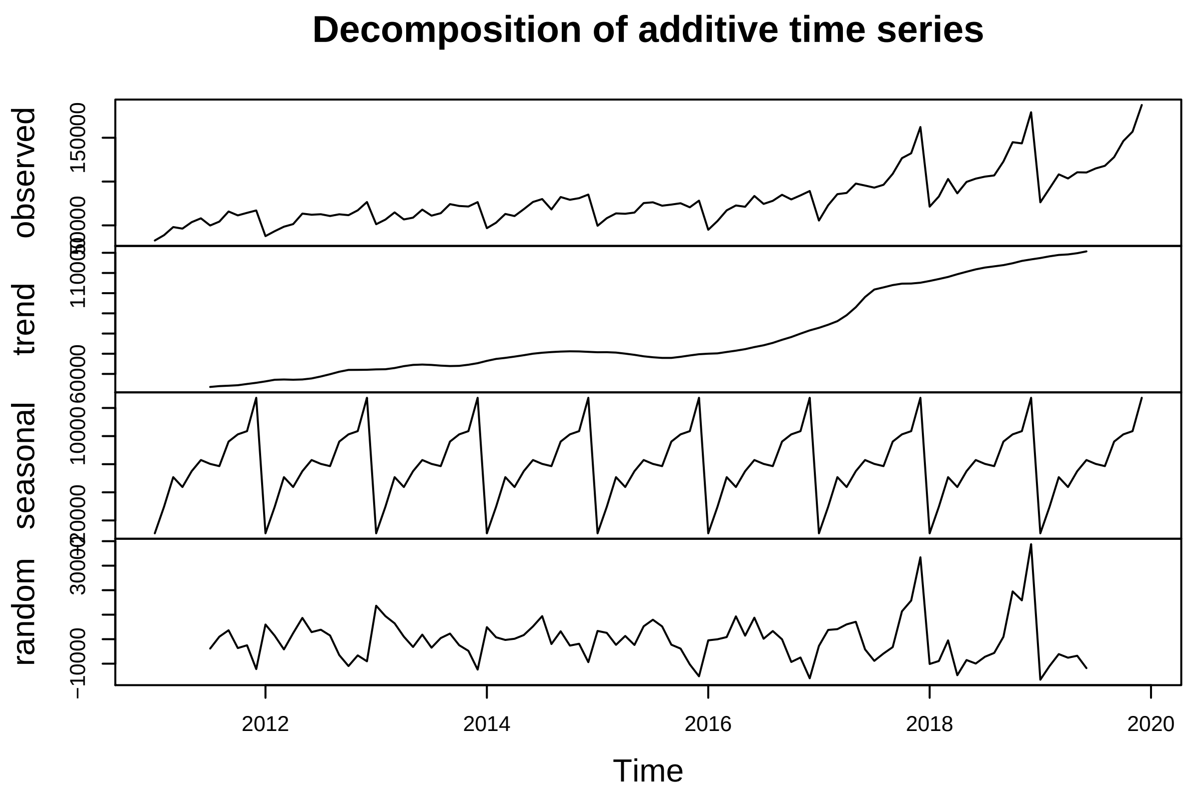
```
ts_components <- decompose(ts)
```

we can print out the estimated values of the seasonal component

```
ts_components$seasonal
```

```
##              Jan         Feb         Mar         Apr         May         Jun
## 2011 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2012 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2013 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2014 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2015 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2016 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2017 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2018 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
## 2019 -24588.4065 -15153.7133  -4601.1352  -8123.8237  -2480.5076   1488.2299
##              Jul         Aug         Sep         Oct         Nov         Dec
```

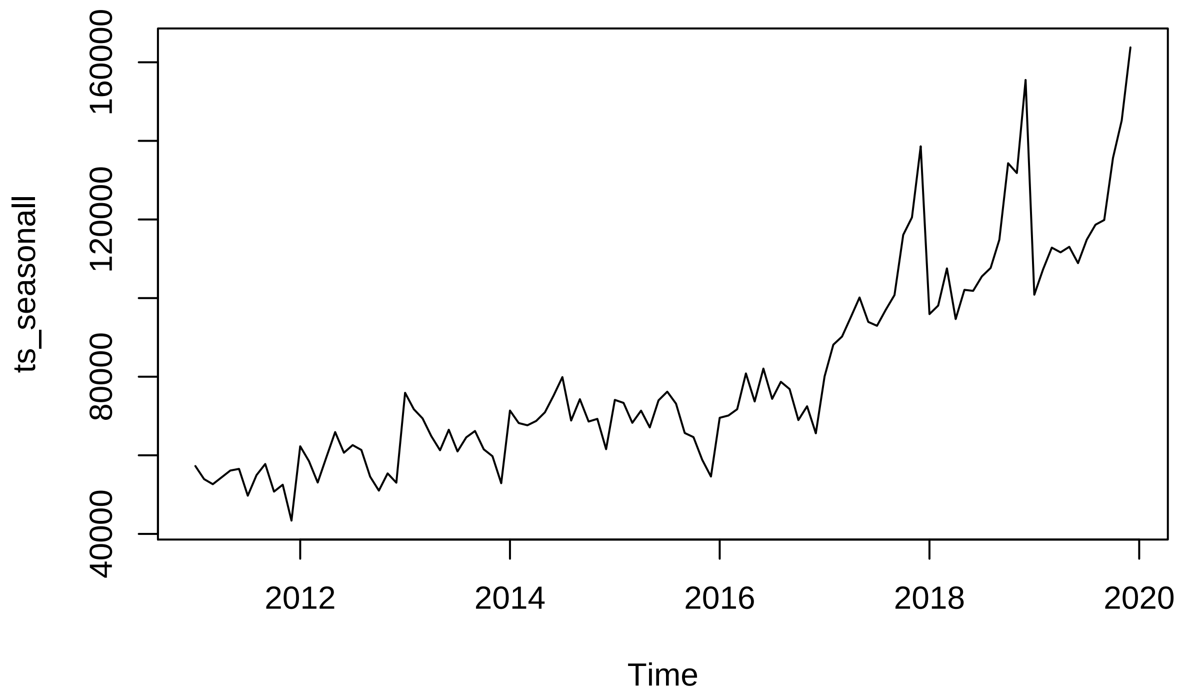
```
## 2011    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2012    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2013    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2014    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2015    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2016    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2017    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2018    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
## 2019    109.2596   -686.7487   8045.2492  10616.0039  11749.1299  23626.4622
```



The plot above shows the original time series (top), the estimated trend component (second from top), the estimated seasonal component (third from top), and the estimated irregular component (bottom)

Seasonally Adjusting

```
ts_seasonall <- ts - ts_components$seasonal
```



Holt-Winters Exponential Smoothing

```
ts_forcaste <- HoltWinters(ts)
ts_forcaste
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts)
##
## Smoothing parameters:
##  alpha: 0.2187957
##  beta : 0
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   116063.0702
## b      412.1641
## s1  -38745.2509
## s2  -23525.0504
## s3   -6821.8331
## s4  -12611.9619
## s5   -4592.2182
## s6   -3071.9176
## s7    2147.6401
```

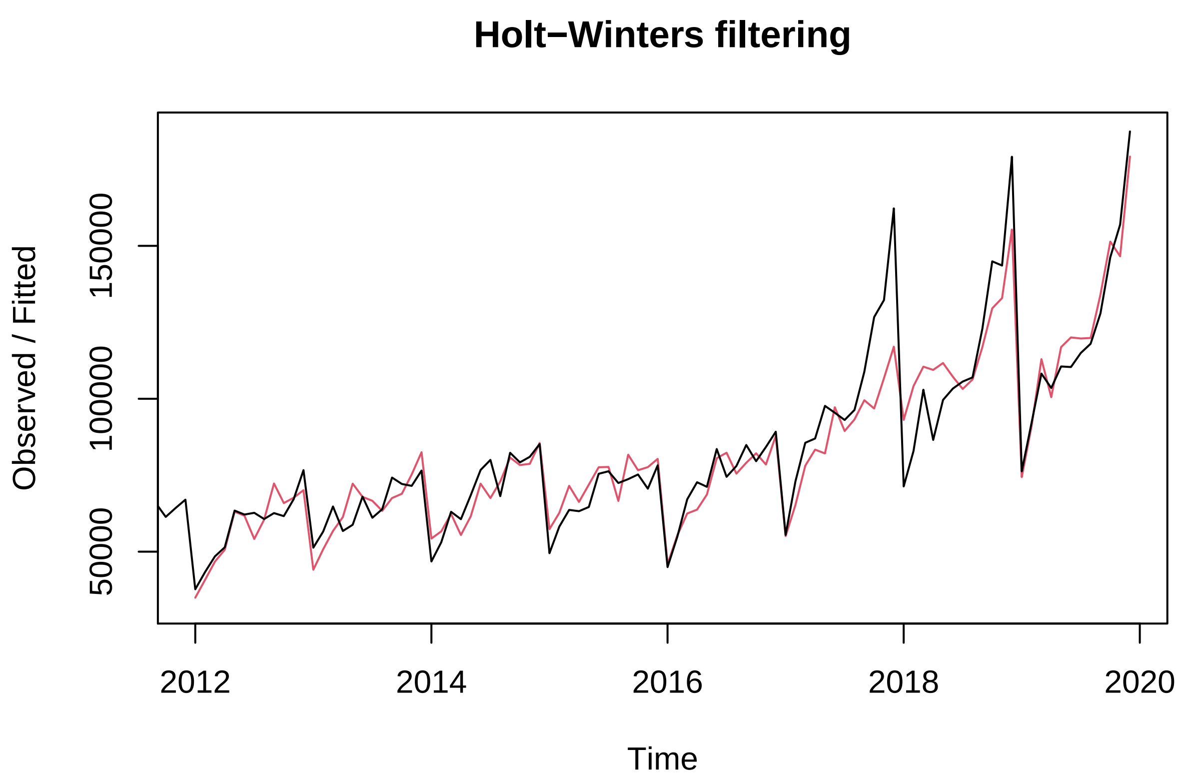
```
## s8     5161.0306
## s9    16027.7968
## s10   35096.6029
## s11   43103.5368
## s12   71346.4298
```

```
#
```

The value of alpha (0.41) is relatively low, indicating that the estimate of the level at the current time point is based upon both recent observations and some observations in the more distant past. The value of beta is 0.00, indicating that the estimate of the slope b of the trend component is not updated over the time series, and instead is set equal to its initial value. This makes good intuitive sense, as the level changes quite a bit over the time series, but the slope b of the trend component remains roughly the same. In contrast, the value of gamma (0.96) is high, indicating that the estimate of the seasonal component at the current time point is just based upon very recent observations

```
ts_forcaste$SSE
```

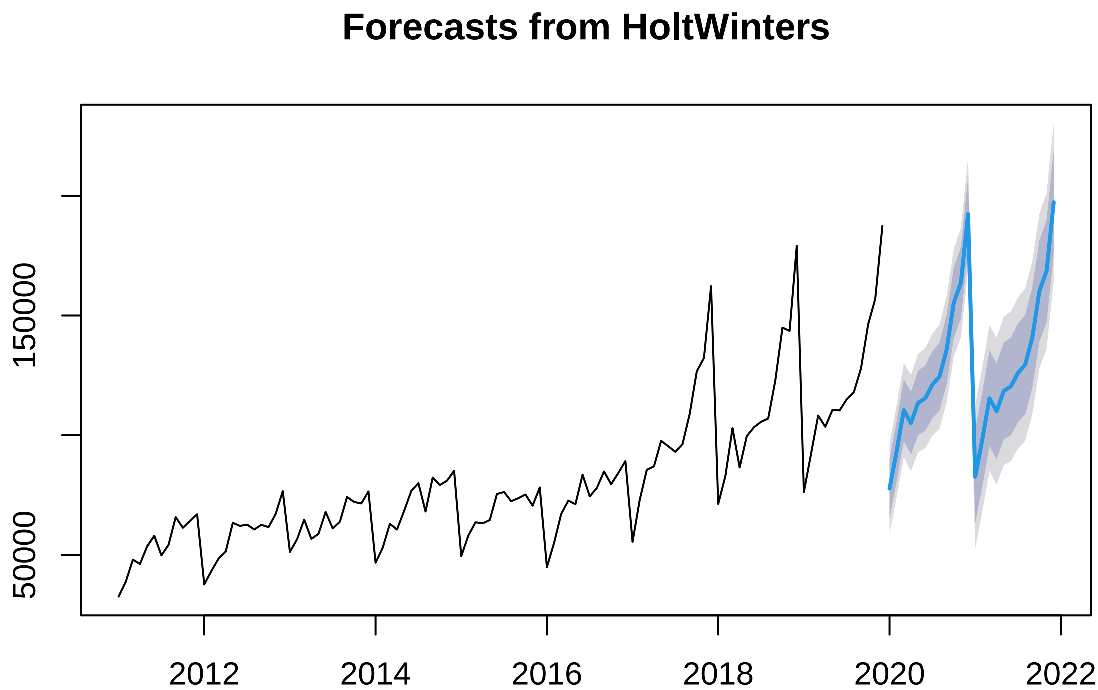
```
## [1] 8847392124
```




```
ts_forcaste2 = forecast:::forecast.HoltWinters(ts_forcaste, h= 24)
(as.data.frame(ts_forcaste2))[1]
```

```
##          Point Forecast
```

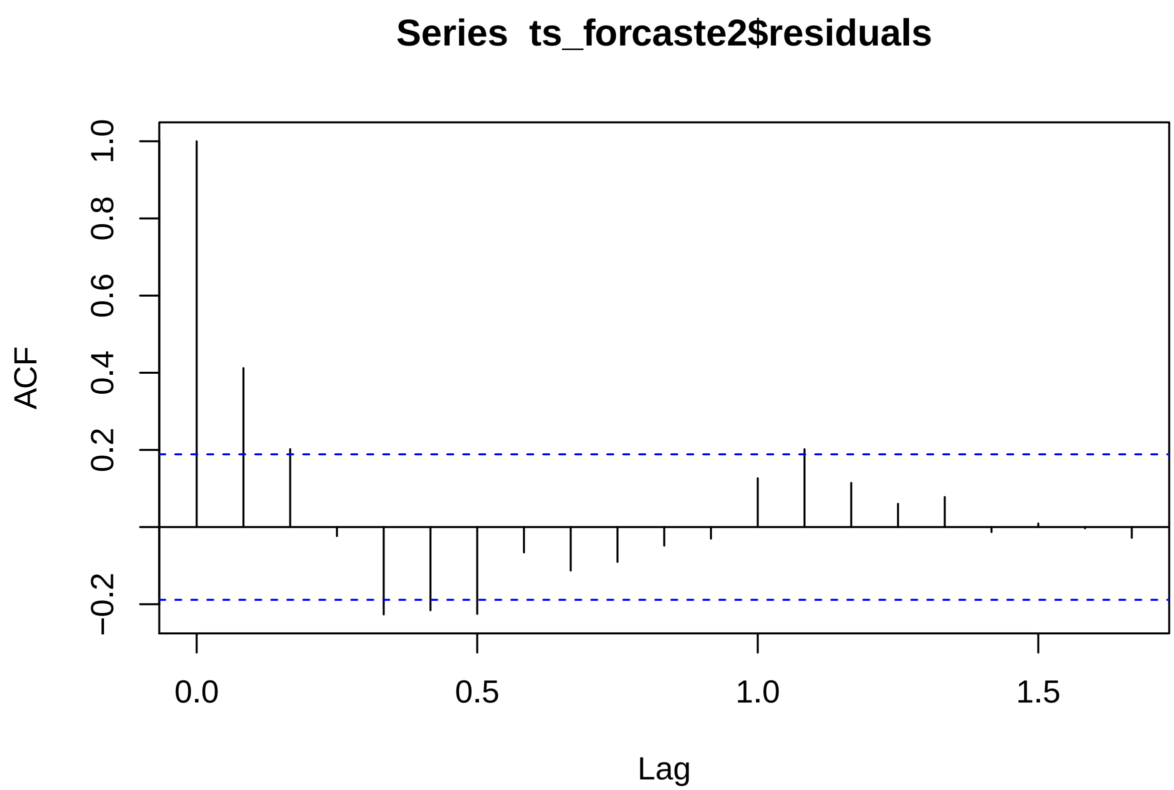
```
## Jan 2020       77729.98
## Feb 2020       93362.35
## Mar 2020      110477.73
## Apr 2020      105099.76
## May 2020      113531.67
## Jun 2020      115464.14
## Jul 2020      121095.86
## Aug 2020      124521.41
## Sep 2020      135800.34
## Oct 2020      155281.31
## Nov 2020      163700.41
## Dec 2020      192355.47
## Jan 2021       82675.95
## Feb 2021       98308.32
## Mar 2021      115423.70
## Apr 2021      110045.73
## May 2021      118477.64
## Jun 2021      120410.11
## Jul 2021      126041.83
## Aug 2021      129467.38
## Sep 2021      140746.31
## Oct 2021      160227.28
## Nov 2021      168646.38
## Dec 2021      197301.44
```



Growth

```r
year_2019 <- window(ts, 2019)
year_2020 <- (as.data.frame(ts_forcaste2))[1][c(1:12),]
year_2021 <- (as.data.frame(ts_forcaste2))[1][c(13:24),]

growth_HW_21 <- growth(sum(year_2021),sum(year_2020))
growth_HW_20 <- growth(sum(year_2020),sum(year_2019))
```

We can investigate whether the predictive model can be improved upon by checking whether the in-sample forecast errors show non-zero autocorrelations at lags 1-20, by making a correlogram and carrying out the



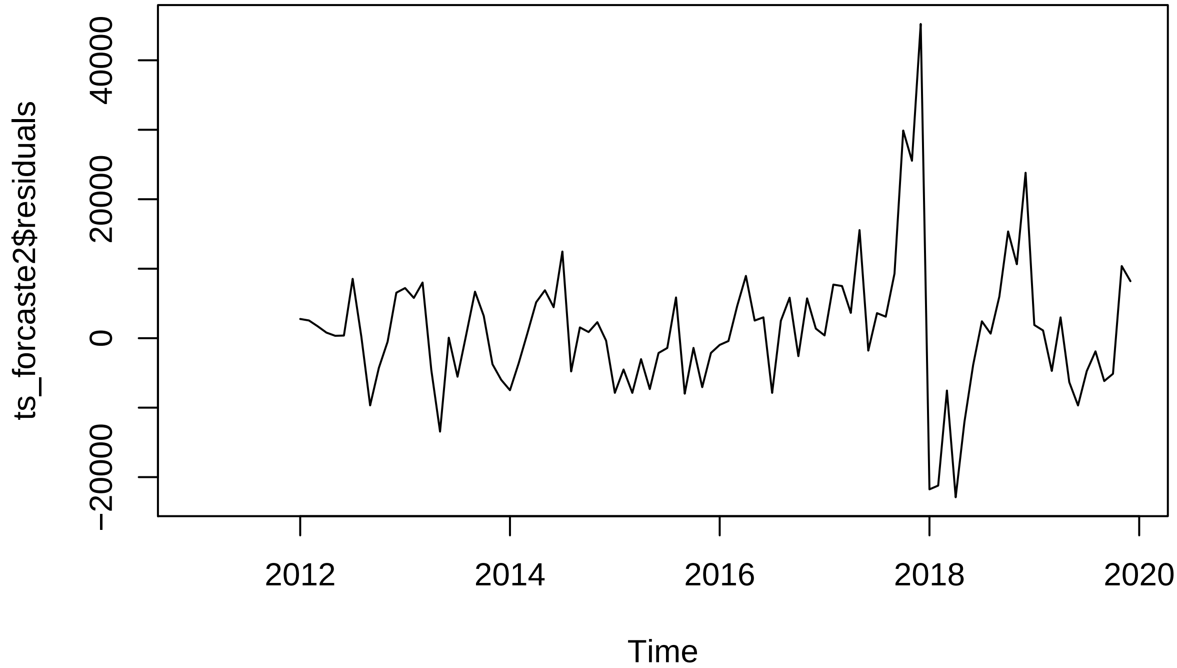
Ljung-Box test:

```
##
##  Box-Ljung test
##
## data:  ts_forcaste2$residuals
## X-squared = 48.555, df = 20, p-value = 0.0003554
```

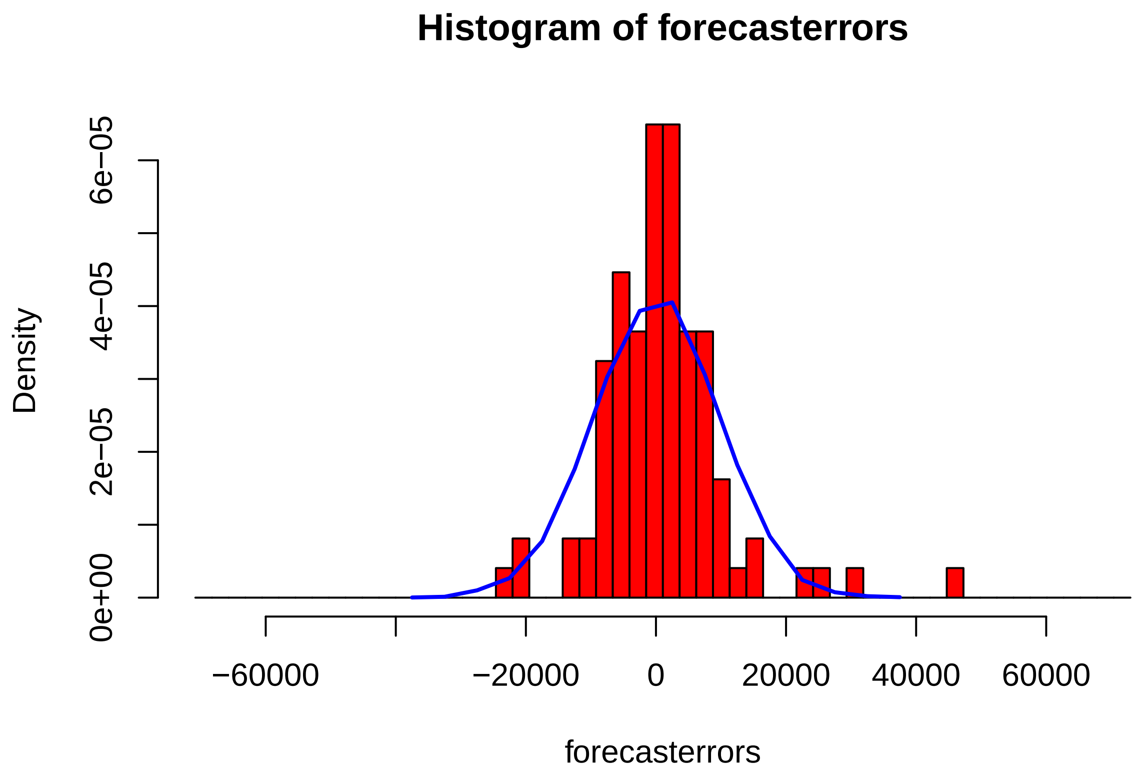
The correlogram shows that the autocorrelations for the in-sample forecast errors do not exceed the significance bounds for lags 1-20. Furthermore, the p-value for Ljung-Box test is 0.2, indicating that there is little evidence of non-zero autocorrelations at lags 1-20.

We can check whether the forecast errors have constant variance over time, and are normally distributed with mean zero, by making a time plot of the forecast errors and a histogram (with overlaid normal curve):

```r
plot.ts(ts_forcaste2$residuals)
```



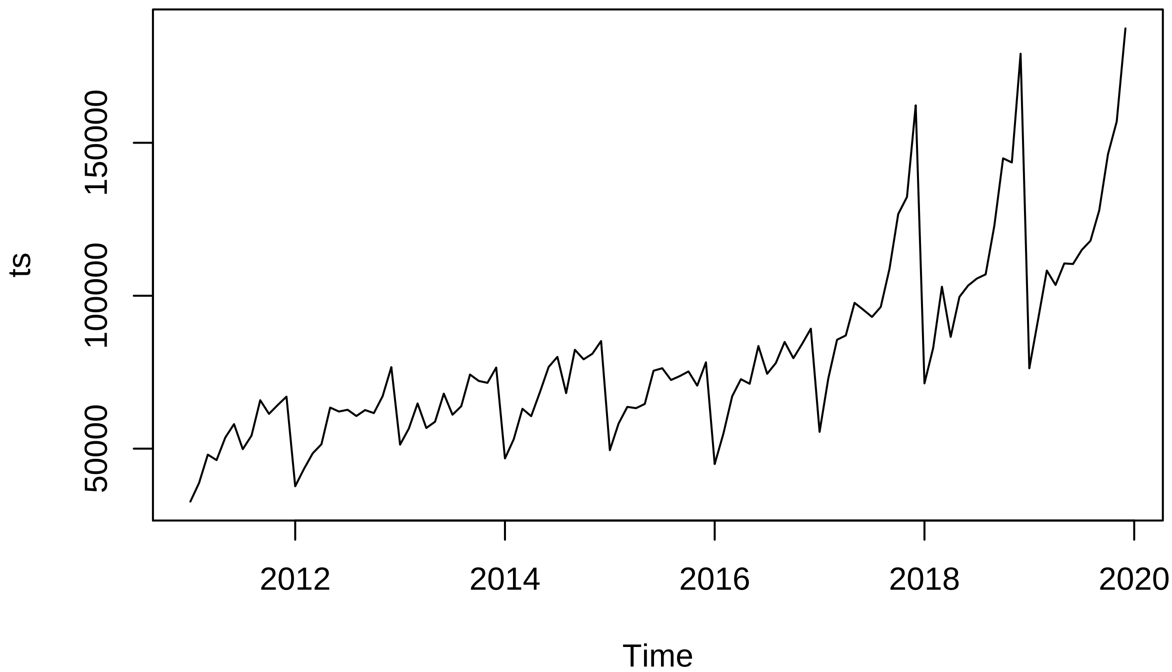

```r
plotForecastErrors(ts_forcaste2$residuals)
```



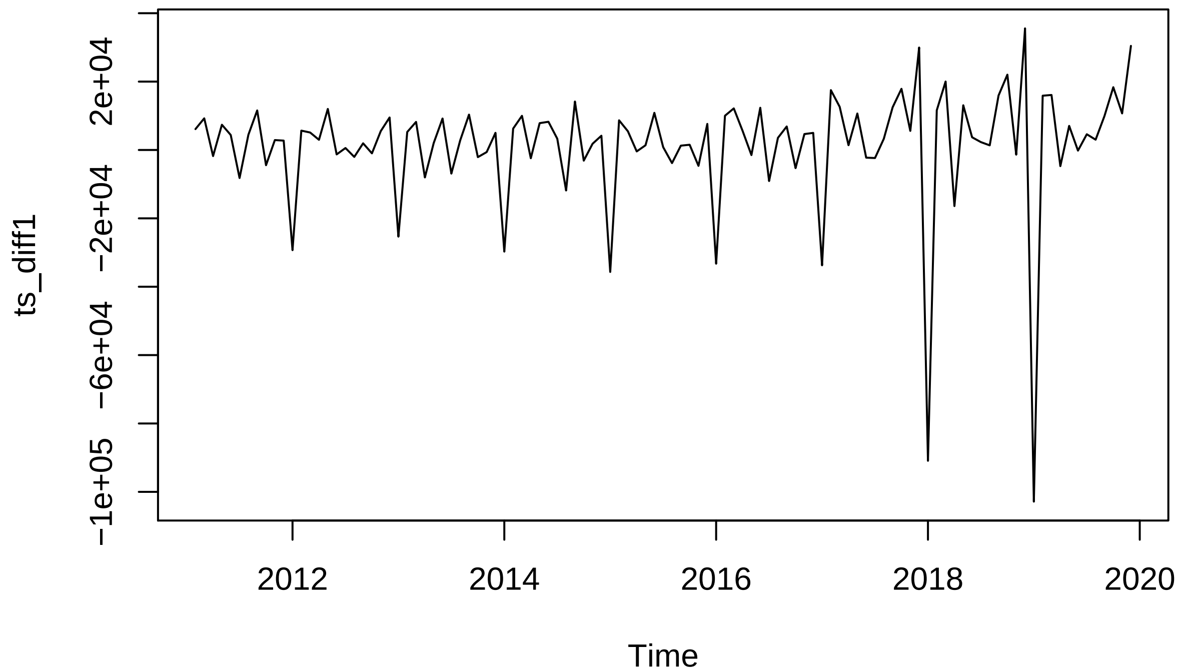
From the time plot, it appears plausible that the forecast errors have constant variance over time. From the histogram of forecast errors, it seems plausible that the forecast errors are normally distributed with mean zero.

Thus,there is little evidence of autocorrelation at lags 1-20 for the forecast errors, and the forecast errors appear to be normally distributed with mean zero and constant variance over time. This suggests that Holt-Winters exponential smoothing provides an adequate predictive model of the log of total productivity, which probably cannot be improved upon. Furthermore, the assumptions upon which the prediction intervals were based are probably valid.

```
plot.ts(ts)
```

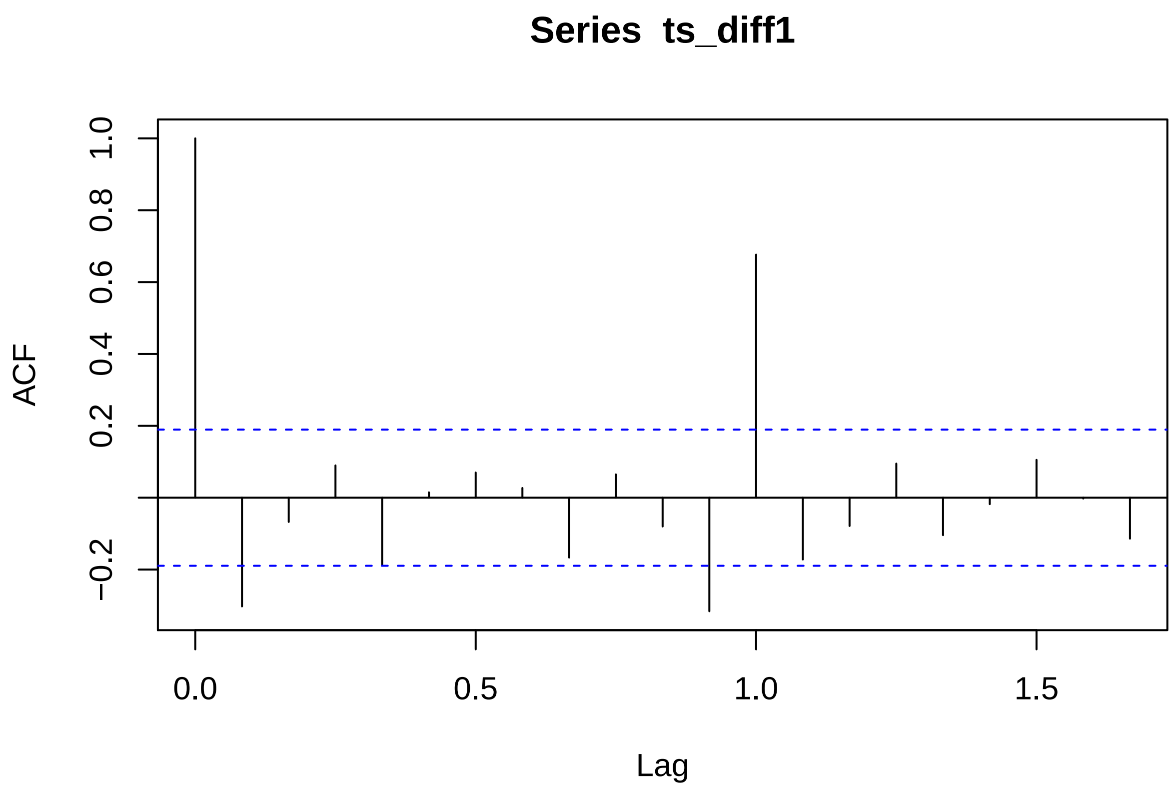



```r
ts_diff1 <-  diff(ts, differences = 1)

plot.ts(ts_diff1)
```



The time series of differences (above) does appear to be stationary in mean and variance, as the level of the series stays roughly constant over time, and the variance of the series appears roughly constant over time

```
acf(ts_diff1, lag.max=20)             # plot a correlogram
```

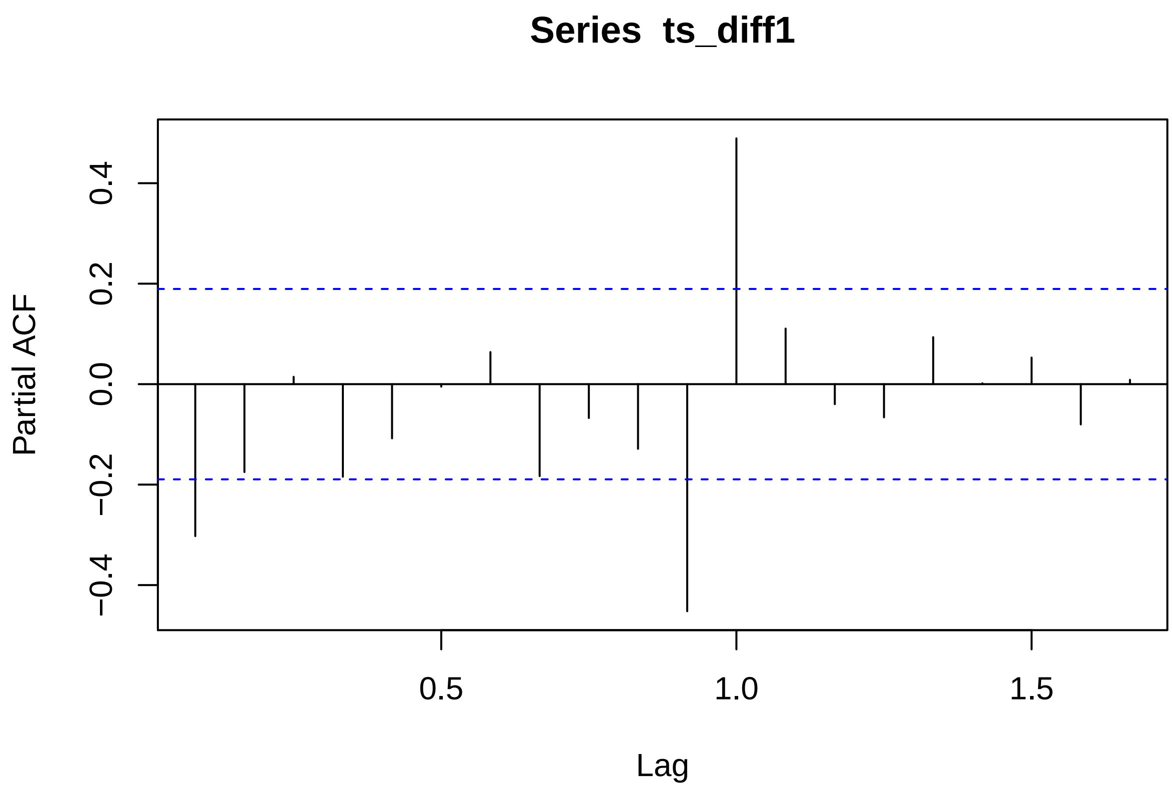


We see from the correlogram that the autocorrelation exceeds the significance bound 3 times but all the others do not exceed

```
acf(ts_diff1, lag.max=20, plot=FALSE) # get the autocorrelation values
```

```
##
## Autocorrelations of series 'ts_diff1', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  1.000 -0.303 -0.067  0.090 -0.187  0.015  0.070  0.027 -0.167  0.065 -0.080
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.316  0.676 -0.172 -0.079  0.095 -0.104 -0.018  0.105 -0.002 -0.114
```

```
pacf(ts_diff1, lag.max=20)             # plot a partial correlogram
```




```
pacf(ts_diff1, lag.max=20, plot=FALSE) # get the partial autocorrelation values
```

```
##
## Partial autocorrelations of series 'ts_diff1', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333 0.9167
## -0.303 -0.175  0.015 -0.184 -0.108 -0.005  0.064 -0.183 -0.068 -0.129 -0.452
## 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
##  0.489  0.111 -0.040 -0.066  0.094  0.002  0.053 -0.080  0.009
```

Arima, 1,1,1

```
ts_arima = Arima(ts, order=c(1,1,1),seasonal = list(order = c(1,1,1)))
ts_arima
```

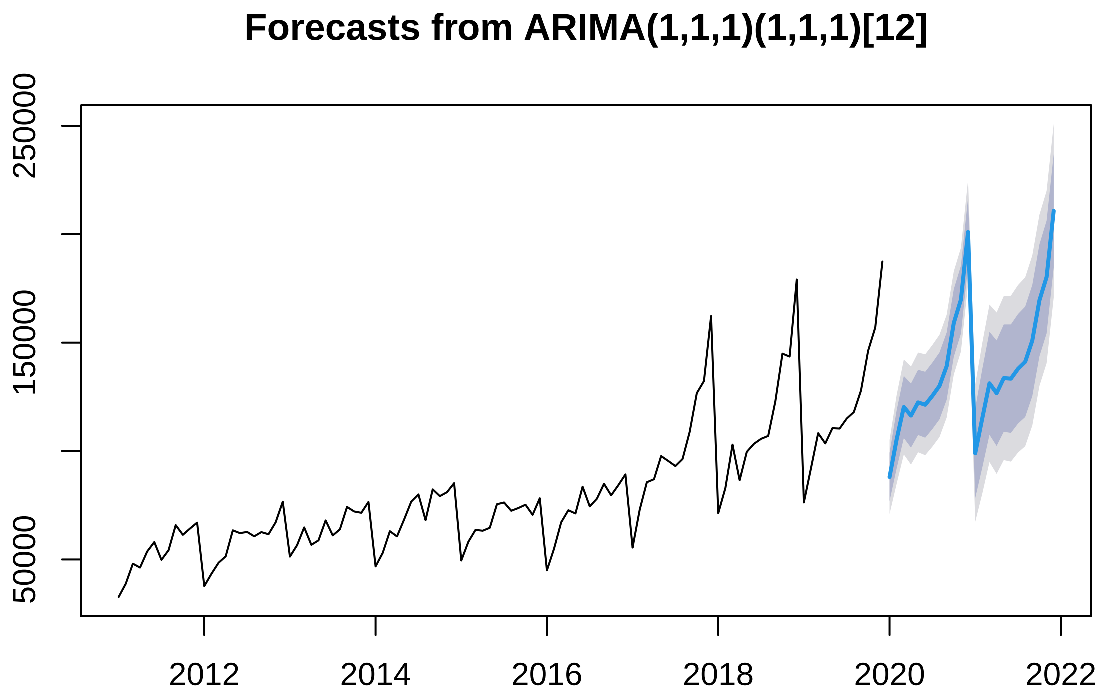
```
## Series: ts
## ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sar1    sma1
##       0.6177  -0.9530  -0.8754  0.9998
## s.e.  0.1160   0.0637   0.1033  0.1453
```

```
##
## sigma^2 estimated as 72394542:  log likelihood=-995.44
## AIC=2000.87   AICc=2001.55   BIC=2013.64
```

```
ts_arima_forecast = forecast(ts_arima,h = 24)
ts_arima_forecast
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       88039.87  76896.84  99182.9  70998.08 105081.7
## Feb 2020      105436.20  92059.69 118812.7  84978.60 125893.8
## Mar 2020      120324.97 106011.07 134638.9  98433.75 142216.2
## Apr 2020      116361.94 101584.89 131139.0  93762.39 138961.5
## May 2020      122438.66 107400.40 137476.9  99439.62 145437.7
## Jun 2020      121352.80 106147.59 136558.0  98098.44 144607.2
## Jul 2020      125469.91 110145.13 140794.7 102032.68 148907.2
## Aug 2020      130059.01 114639.96 145478.1 106477.61 153640.4
## Sep 2020      139121.13 123622.13 154620.1 115417.45 162824.8
## Oct 2020      159073.19 143502.80 174643.6 135260.34 182886.0
## Nov 2020      169793.04 154156.94 185429.1 145879.69 193706.4
## Dec 2020      201056.74 185359.63 216753.8 177050.08 225063.4
## Jan 2021       98936.41  78260.52 119612.3  67315.37 130557.4
## Feb 2021      115187.92  92475.13 137900.7  80451.71 149924.1
## Mar 2021      131238.08 107501.18 154975.0  94935.63 167540.5
## Apr 2021      126680.82 102348.39 151013.2  89467.59 163894.0
## May 2021      133651.21 108925.06 158377.4  95835.83 171466.6
## Jun 2021      133379.82 108362.67 158397.0  95119.38 171640.3
## Jul 2021      137932.36 112679.79 163184.9  99311.88 176552.8
## Aug 2021      141150.30 115693.91 166606.7 102218.11 180082.5
## Sep 2021      150977.24 125335.73 176618.8 111761.93 190192.5
## Oct 2021      169539.36 143724.16 195354.6 130058.42 209020.3
## Nov 2021      180236.21 154254.57 206217.8 140500.72 219971.7
## Dec 2021      210789.18 184645.87 236932.5 170806.44 250771.9
```

```
forecast:::plot.forecast(ts_arima_forecast)
```



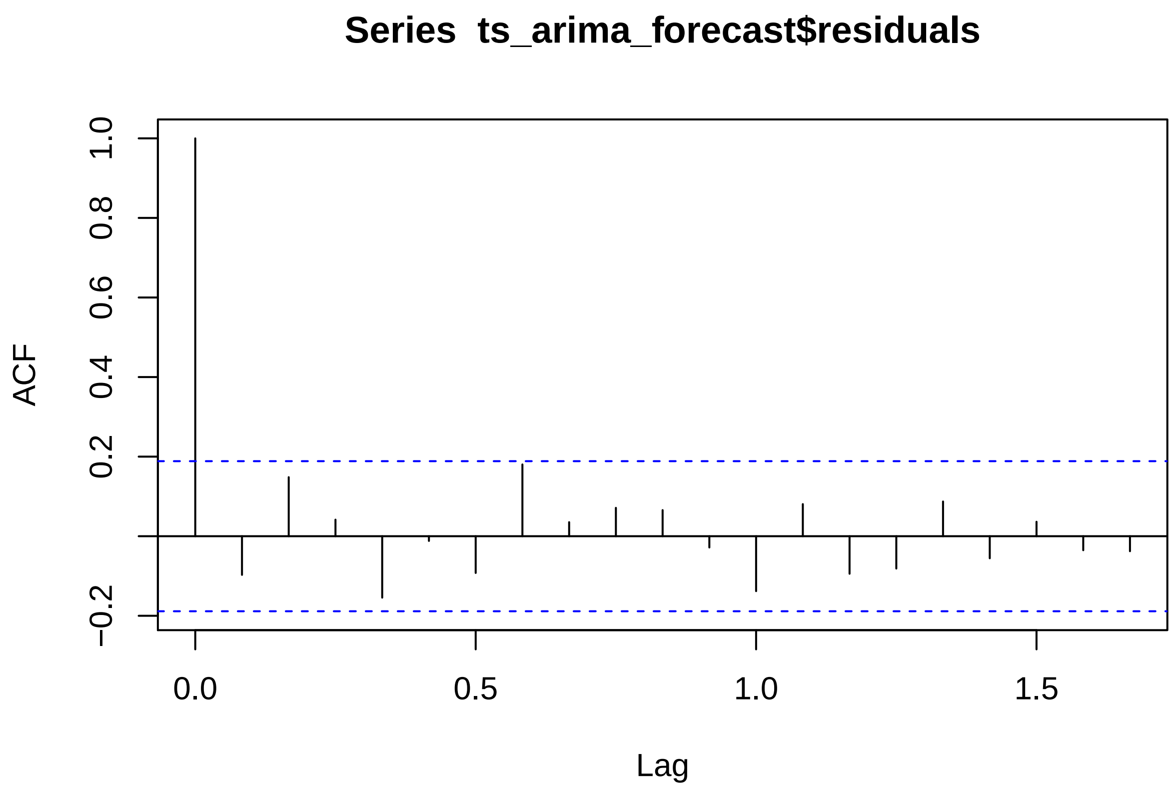
Growth

```
year_2020 <- (as.data.frame(ts_arima_forecast))[1][c(1:12),]
year_2021 <- (as.data.frame(ts_arima_forecast))[1][c(13:24),]

growth_ARIMA_21 <- growth(sum(year_2021), sum(year_2020))
growth_ARIMA_20 <- growth(sum(year_2020), sum(year_2019))
```

As in the case of exponential smoothing models, it is a good idea to investigate whether the forecast errors of an ARIMA model are normally distributed with mean zero and constant variance, and whether the are correlations between successive forecast errors.

For example, we can make a correlogram of the forecast errors for our ARIMA(0,1,1) model, and perform the Ljung-Box test for lags 1-20, by typing:

```
acf(ts_arima_forecast$residuals, lag.max=20)
```

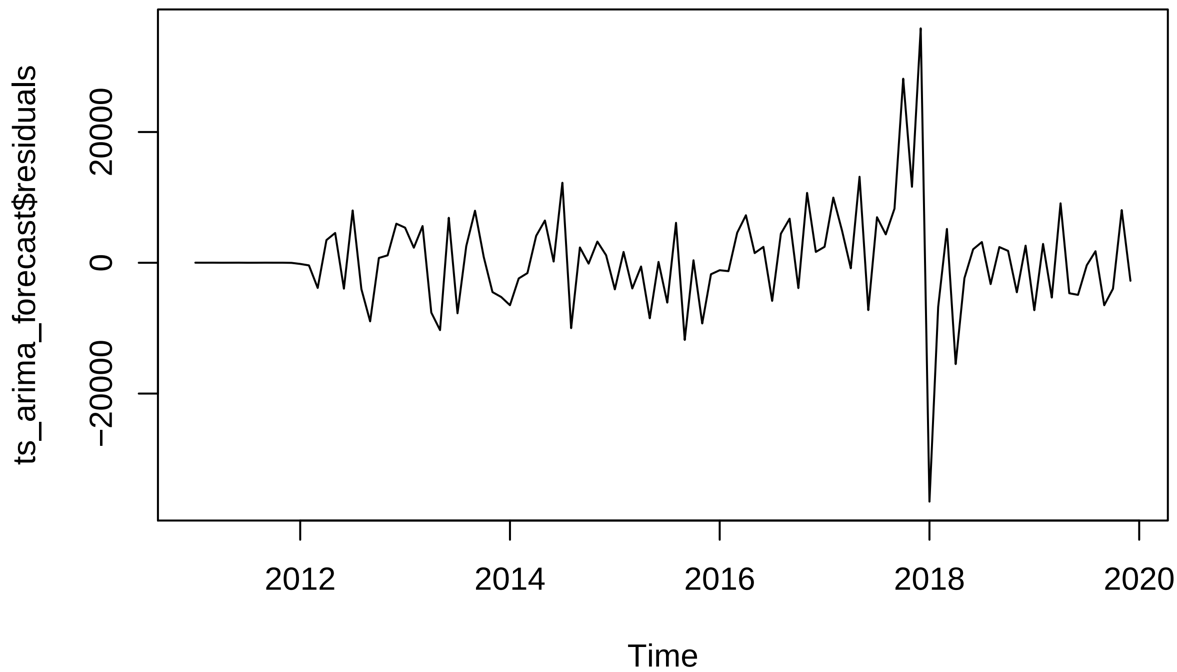



```
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

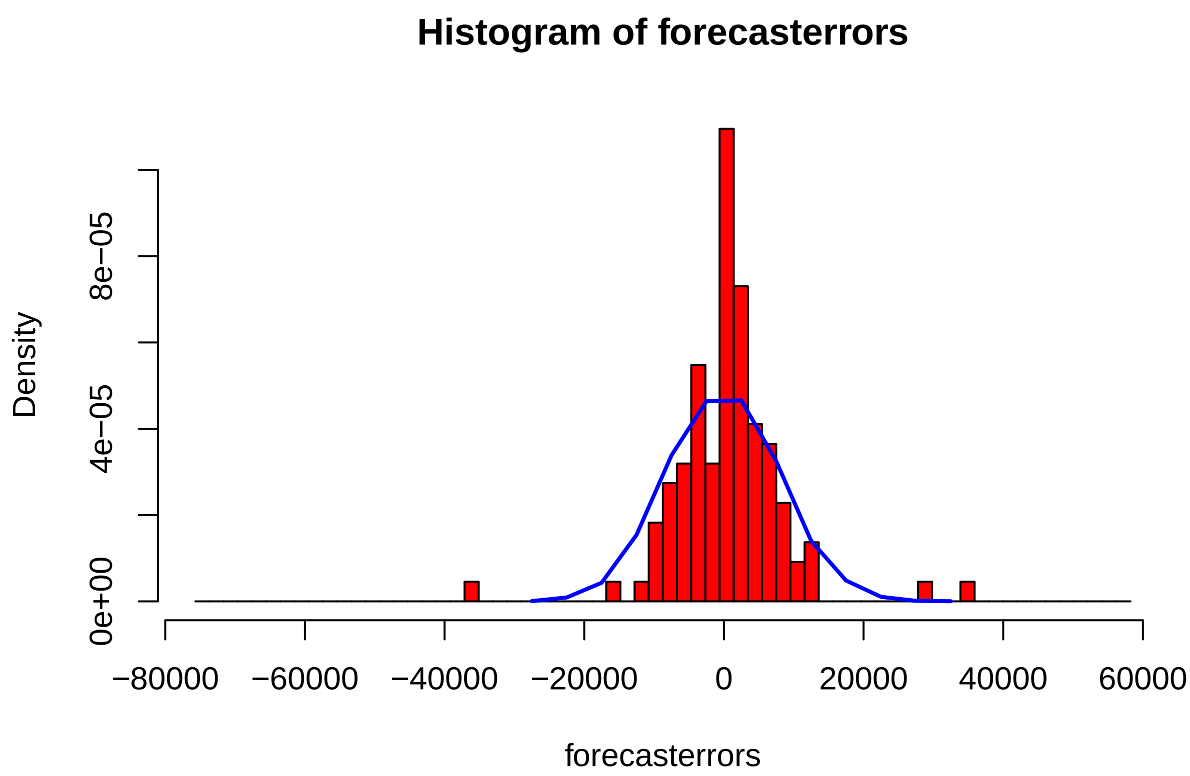
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 19.714, df = 20, p-value = 0.4759
```

we can reject the null hypothesis, it's rather similar to the HW

```
plot.ts(ts_arima_forecast$residuals)           # make time plot of forecast errors
```




```
plotForecastErrors(ts_arima_forecast$residuals)
```



Arima, 0,1,0 as given from the loop

```
ts_arima = Arima(ts, order=c(2,1,1),seasonal = list(order = c(2,1,0)))
ts_arima
```

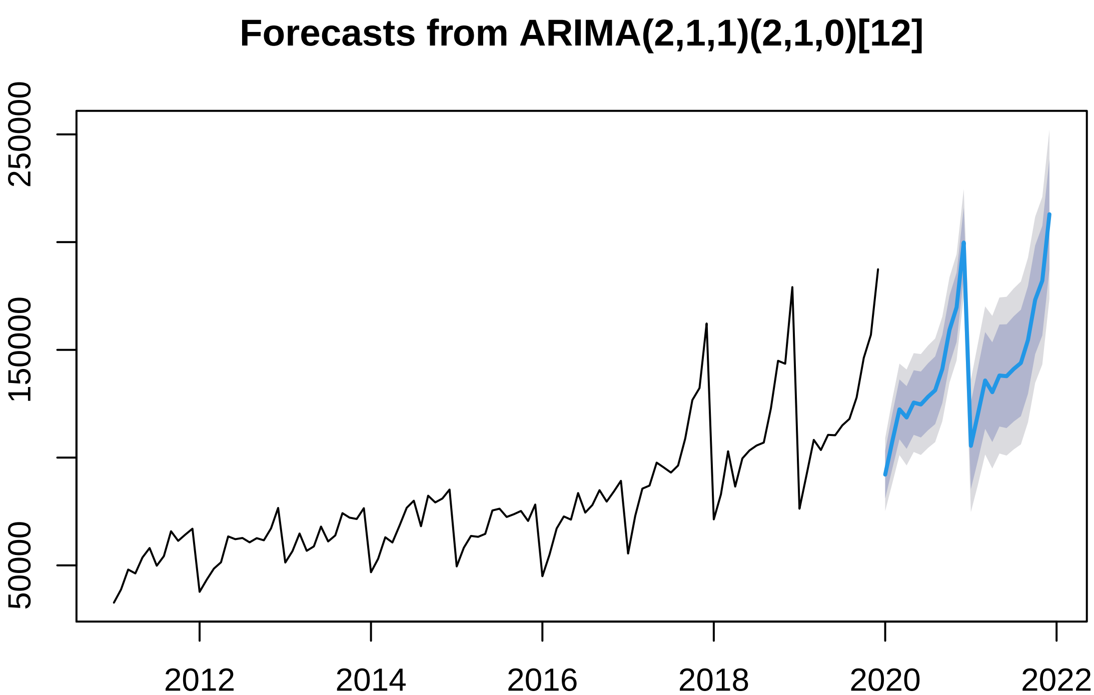
```
## Series: ts
## ARIMA(2,1,1)(2,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1     sar1     sar2
##       0.5262  0.1481  -0.9532  -0.0865  -0.1356
## s.e.  0.1144  0.1093   0.0558   0.1043   0.1196
##
## sigma^2 estimated as 74528880:  log likelihood=-994.04
## AIC=2000.08   AICc=2001.03   BIC=2015.4
```

```
ts_arima_forecast = forecast(ts_arima,h = 24)
ts_arima_forecast
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       92115.67  81051.95 103179.4  75195.18 109036.2
## Feb 2020      107593.74  94842.37 120345.1  88092.20 127095.3
## Mar 2020      122397.56 108513.50 136281.6 101163.71 143631.4
## Apr 2020      118630.34 104081.57 133179.1  96379.91 140880.8
## May 2020      125545.51 110553.33 140537.7 102616.95 148474.1
## Jun 2020      124649.43 109346.86 139952.0 101246.17 148052.7
```

```
## Jul 2020       128240.39 112707.02 143773.8 104484.15 151996.6
## Aug 2020       131240.95 115526.91 146955.0 107208.39 155273.5
## Sep 2020       141128.71 125266.42 156991.0 116869.43 165388.0
## Oct 2020       159176.98 143188.04 175165.9 134724.00 183630.0
## Nov 2020       169717.11 153616.21 185818.0 145092.90 194341.3
## Dec 2020       199814.65 183611.99 216017.3 175034.82 224594.5
## Jan 2021       105460.24  85380.97 125539.5  74751.65 136168.8
## Feb 2021       120373.79  98939.39 141808.2  87592.71 153154.9
## Mar 2021       135808.06 113352.50 158263.6 101465.24 170150.9
## Apr 2021       130363.30 107213.92 153512.7  94959.37 165767.2
## May 2021       138099.58 114428.36 161770.8 101897.58 174301.6
## Jun 2021       137787.89 113706.54 161869.2 100958.64 174617.1
## Jul 2021       141150.28 116730.36 165570.2 103803.24 178497.3
## Aug 2021       143923.52 119212.90 168634.1 106131.88 181715.1
## Sep 2021       154626.96 129658.33 179595.6 116440.73 192813.2
## Oct 2021       173197.54 147993.71 198401.4 134651.61 211743.5
## Nov 2021       182114.41 156691.65 207537.2 143233.65 220995.2
## Dec 2021       212935.48 187305.56 238565.4 173737.90 252133.1
```

```
forecast:::plot.forecast(ts_arima_forecast)
```



Growth

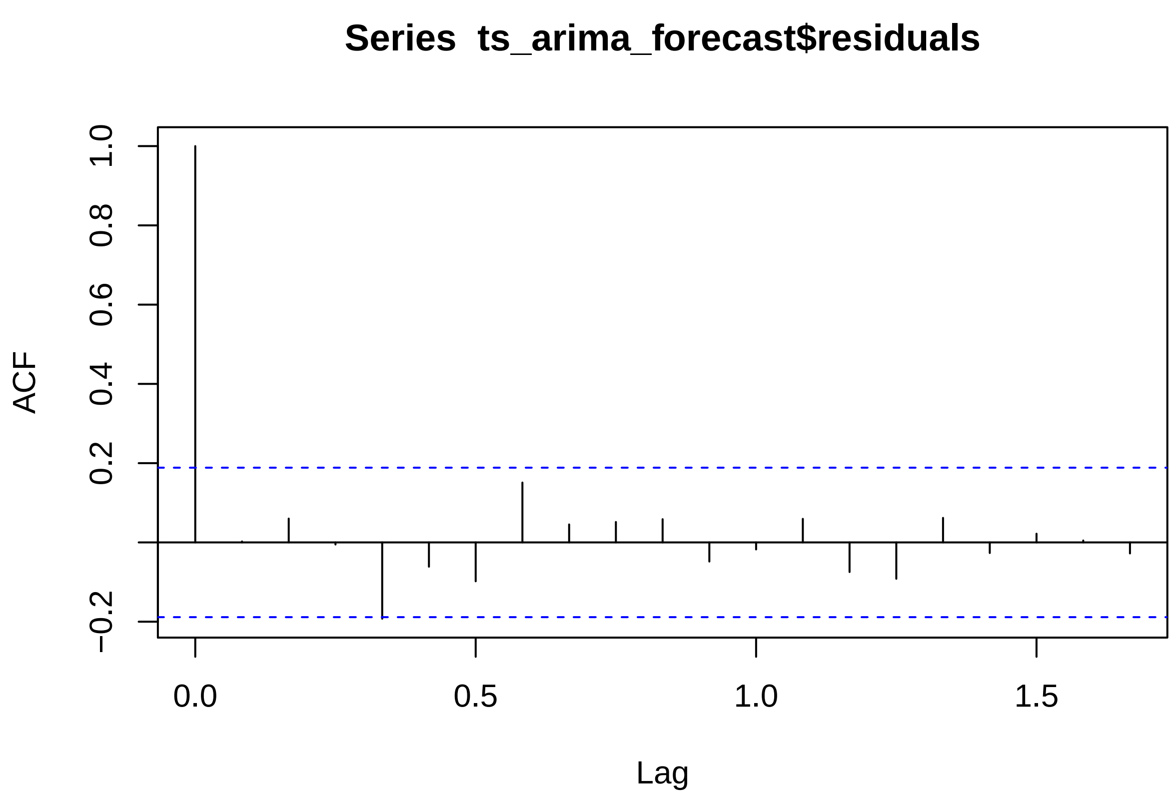
```
# this_year_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1]
#
# year_2019_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1][c(1:2),]
```

```r
# sum_year_2019 = sum(c(year_2019,year_2019_predict_ARIMA))
# year_2020 = (as.data.frame(ts_arima_forecast))[1][c(3:14),]

year_2020 <- (as.data.frame(ts_arima_forecast))[1][c(1:12),]
year_2021 <- (as.data.frame(ts_arima_forecast))[1][c(13:24),]

growth_ARIMA2_21 <- growth(sum(year_2021), sum(year_2020))
growth_ARIMA2_20 <- growth(sum(year_2020), sum(year_2019))
```

As in the case of exponential smoothing models, it is a good idea to investigate whether the forecast errors of an ARIMA model are normally distributed with mean zero and constant variance, and whether the are correlations between successive forecast errors.

For example, we can make a correlogram of the forecast errors for our ARIMA(0,1,1) model, and perform the Ljung-Box test for lags 1-20, by typing:

```r
acf(ts_arima_forecast$residuals, lag.max=20)
```

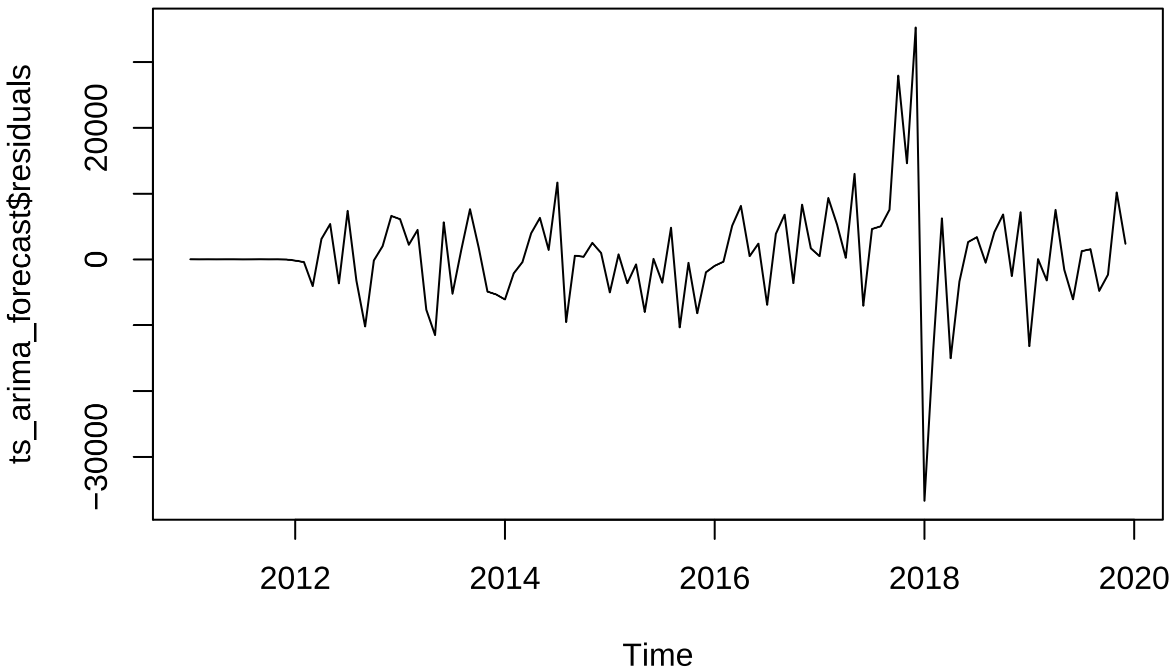



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

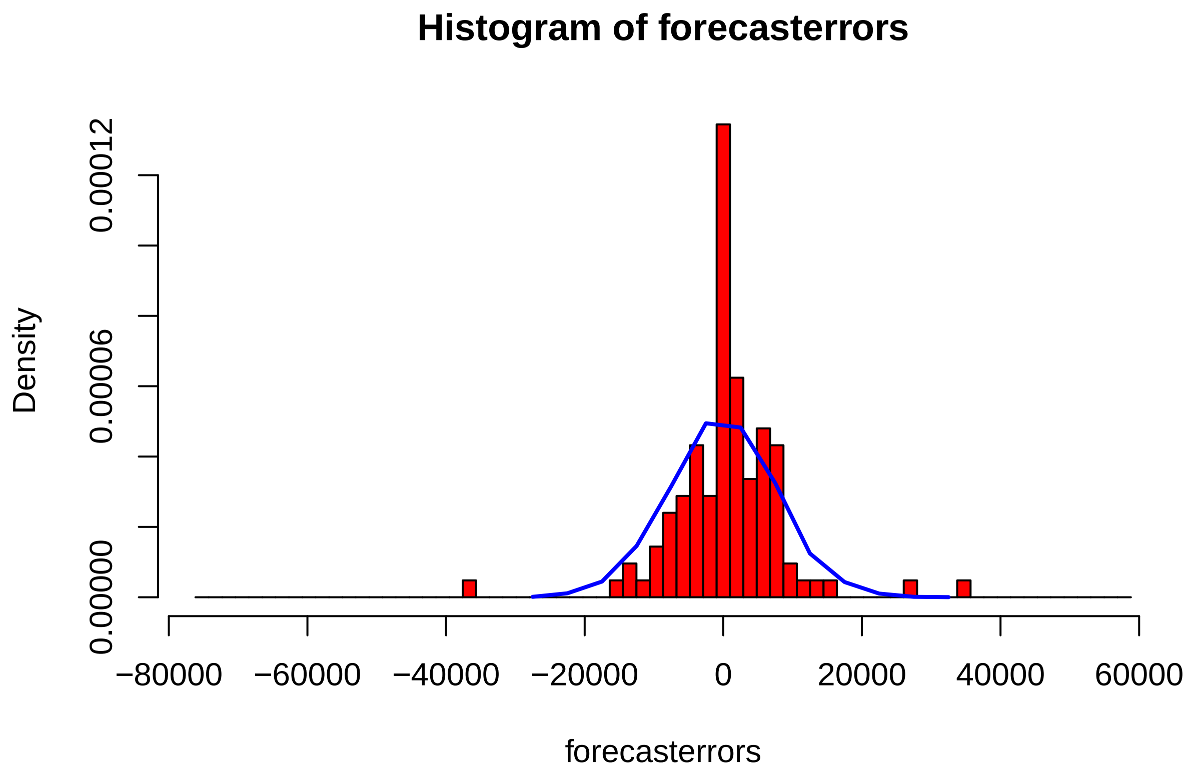
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 13.167, df = 20, p-value = 0.8701
```

we can reject the null hypothesis, it's rather similar to the HW

```
plot.ts(ts_arima_forecast$residuals)            # make time plot of forecast errors
```




```
plotForecastErrors(ts_arima_forecast$residuals)
```

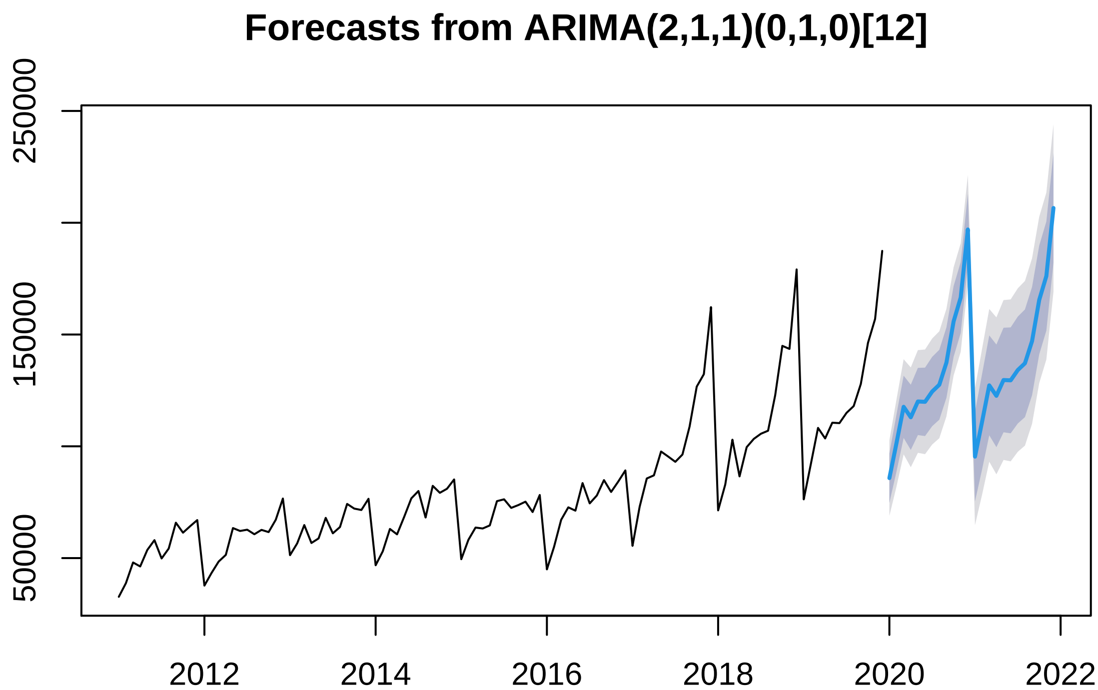


A model chosen automatically

```
fit <- auto.arima(ts,max.p = 5,max.q = 5,max.P = 5,max.Q = 5,max.d = 3,seasonal = TRUE)
fit
```

```
## Series: ts
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1
##       0.5543  0.1695  -0.9819
## s.e.  0.1080  0.1060   0.0549
##
## sigma^2 estimated as 74285214:  log likelihood=-994.88
## AIC=1997.77   AICc=1998.21   BIC=2007.98
```

```
fit_forecast = forecast(fit,h=24)
plot(fit_forecast)
```




```
# str(fit)
```

Growth

```
#
# year_2019_predict_auto.arima <- (as.data.frame(fit_forecast))[1][c(1:2),]
# year_2019_predict_auto.arima_95_low <- (as.data.frame(fit_forecast))[4][c(1:2),]
# year_2019_predict_auto.arima_95_high <- (as.data.frame(fit_forecast))[5][c(1:2),]
#
# sum_year_2019 = sum(c(year_2019,year_2019_predict_auto.arima))
# sum_year_2019_low = sum(c(year_2019,year_2019_predict_auto.arima_95_low))
# sum_year_2019_high = sum(c(year_2019,year_2019_predict_auto.arima_95_high))
#
#
# year_2020_predict_auto.arima <- (as.data.frame(fit_forecast))[1][c(3:14),]
# year_2020_predict_auto.arima_95_low <- (as.data.frame(fit_forecast))[4][c(3:14),]
# year_2020_predict_auto.arima_95_high <- (as.data.frame(fit_forecast))[5][c(3:14),]


#
# growth_auto.arima <- growth(sum(year_2020_predict_auto.arima),sum_year_2019)
# growth_auto.arima_95_low <- growth(sum(year_2020_predict_auto.arima_95_low),sum_year_2019_low)
# growth_auto.arima_95_high <- growth(sum(year_2020_predict_auto.arima_95_high),sum_year_2019_high)
```

```r
year_2020 <- (as.data.frame(fit_forecast))[1][c(1:12),]
year_2021 <- (as.data.frame(fit_forecast))[1][c(13:24),]

growth_auto.arima_21 <- growth(sum(year_2021), sum(year_2020))
growth_auto.arima_20 <- growth(sum(year_2020), sum(year_2019))
```

all the growths

```r
growth_ARIMA_20
```

```
## [1] 0.1004553
```

```r
growth_ARIMA_21
```

```
## [1] 0.08205767
```

```r
growth_ARIMA2_20
```

```
## [1] 0.1154102
```

```r
growth_ARIMA2_21
```

```
## [1] 0.09602832
```

```r
growth_auto.arima_20
```

```
## [1] 0.0785966
```

```r
growth_auto.arima_21
```

```
## [1] 0.07352846
```

```r
growth_HW_20
```

```
## [1] 0.03842402
```

```r
growth_HW_21
```

```
## [1] 0.03934687
```