# Time Series Forcasting report for total industry

Kevork Sulahian

2021-04-22

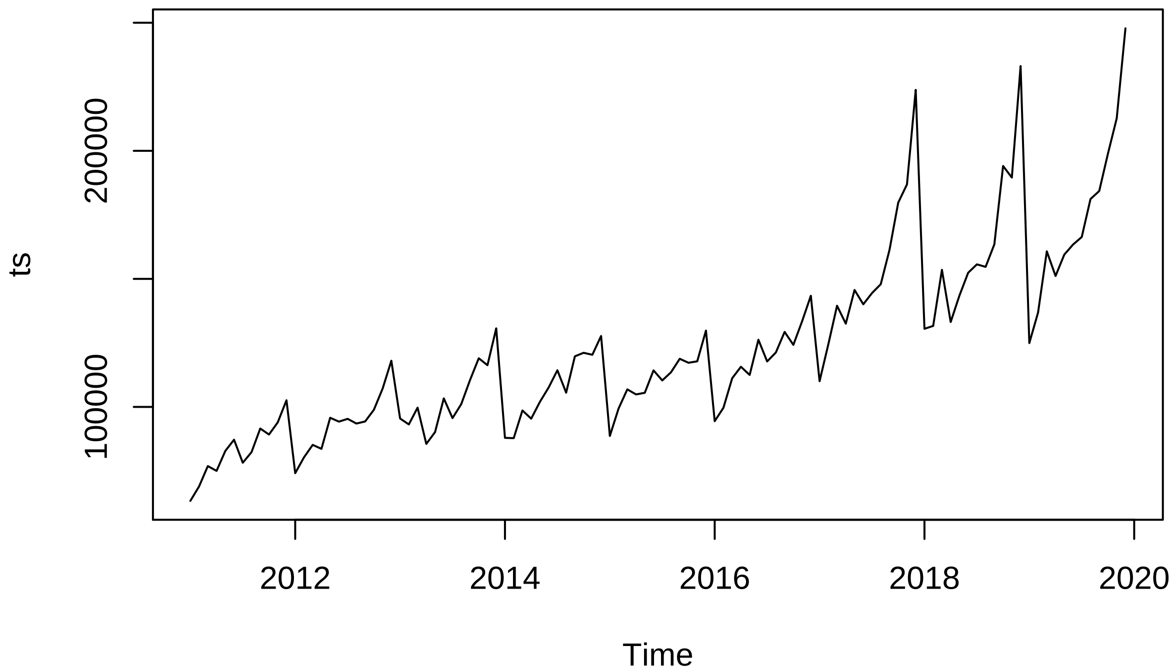
```r
library(readxl)
library(forecast)
```

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

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

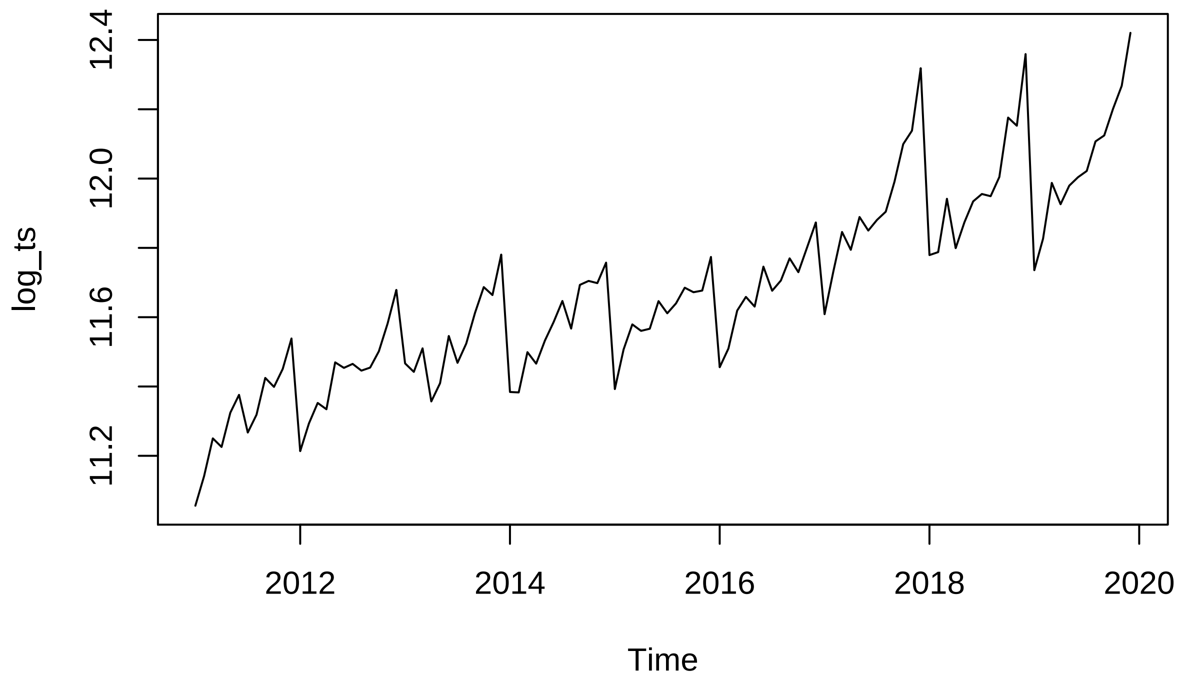
```
## New names:
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * `` -> ...6
## * ...
```

```r
df = df[4,]
df = df[-c(1,3)]
# rownames(df) = df[1]
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.

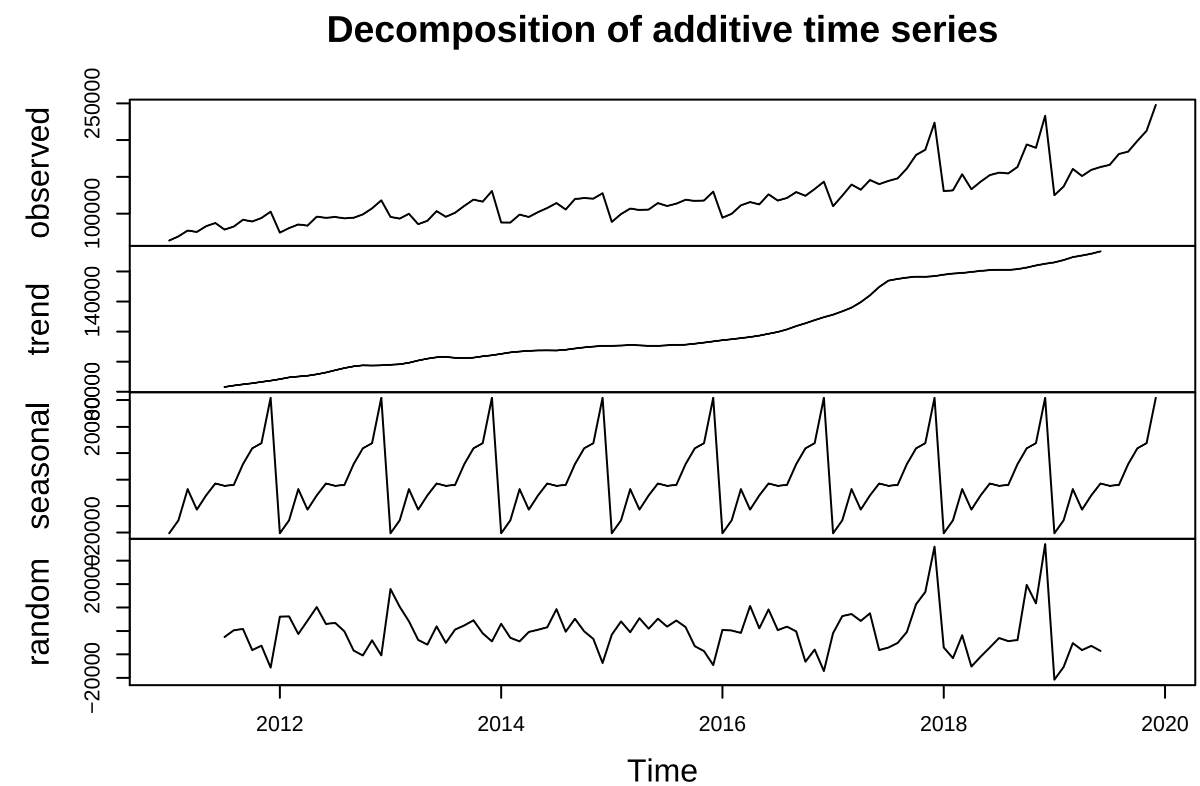
```
ts_components <- decompose(ts)
```

we can print out the estimated values of the seasonal component

```
ts_components$seasonal
```

```
##              Jan        Feb        Mar        Apr        May        Jun
## 2011 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2012 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2013 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2014 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2015 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2016 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2017 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2018 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
## 2019 -20296.500 -15377.583  -3605.059 -11343.268  -5960.194  -1457.614
##              Jul        Aug        Sep        Oct        Nov        Dec
```

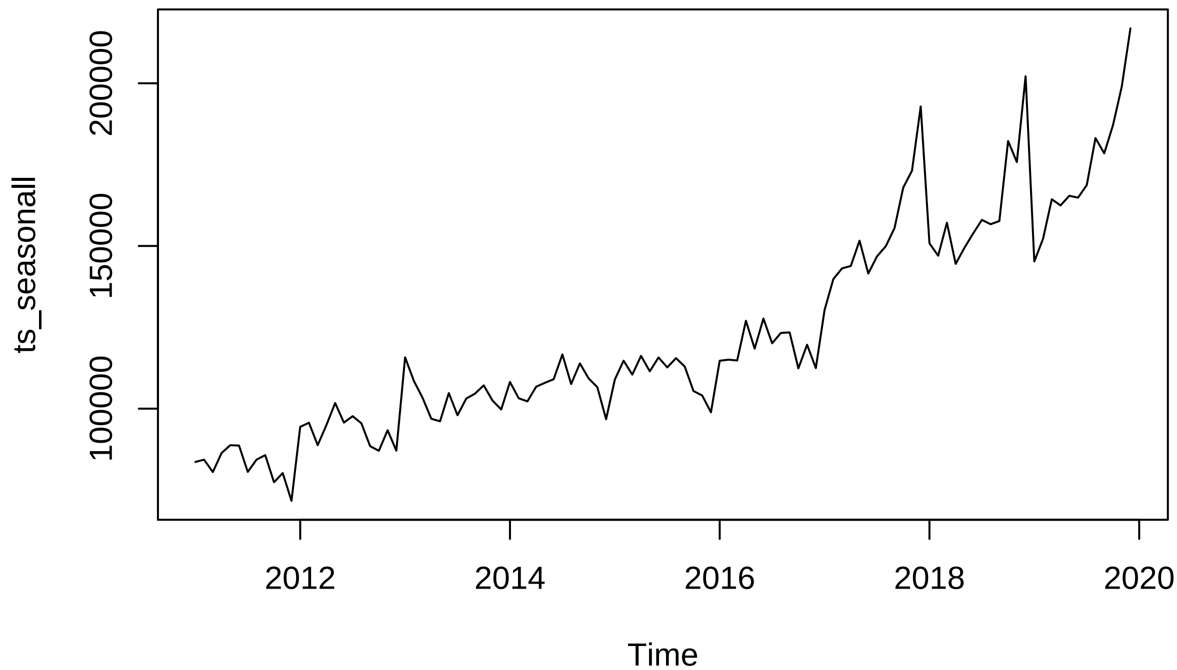
```
## 2011  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2012  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2013  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2014  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2015  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2016  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2017  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2018  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
## 2019  -2348.985  -1997.691   5847.761  11831.074  13768.751  30939.306
```




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

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

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts)
##
## Smoothing parameters:
##  alpha: 0.3444881
##  beta : 0
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   182005.7804
## b      859.0055
## s1  -46424.5490
## s2  -32968.9766
## s3   -7812.7574
## s4  -18184.3629
## s5   -8636.7940
## s6   -3044.6551
## s7     340.6217
```

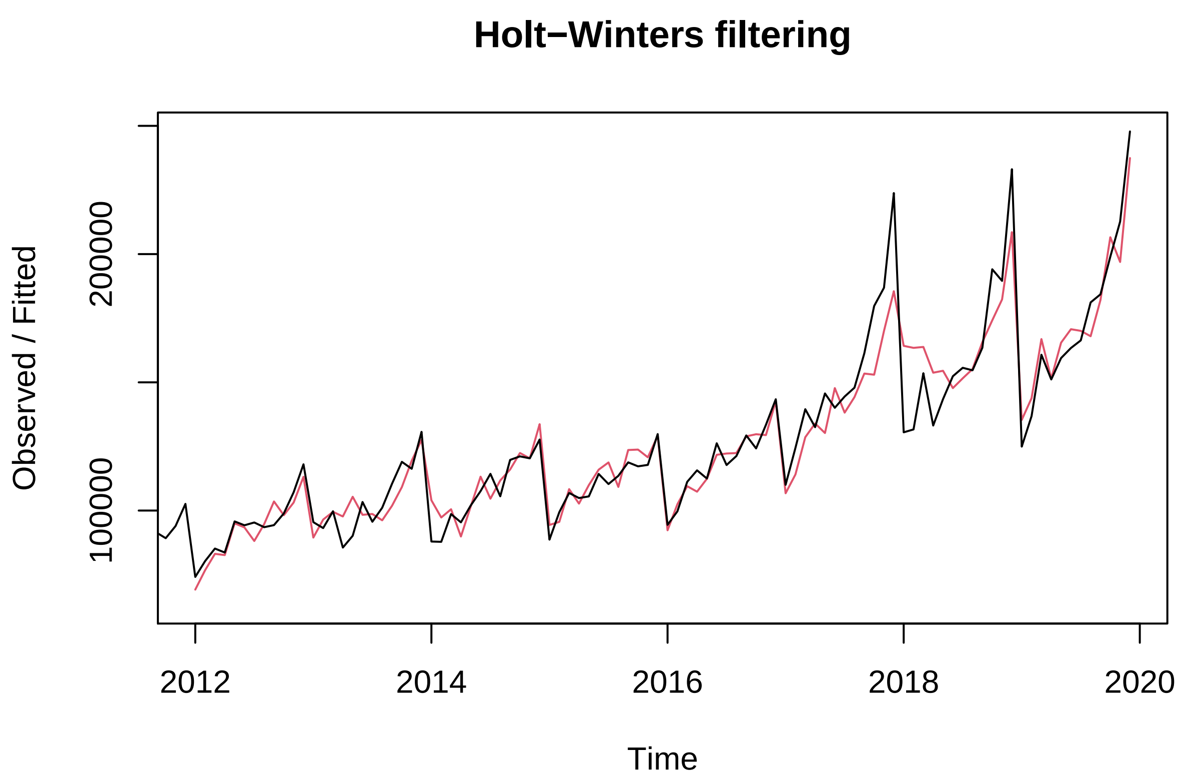
```
## s8    9742.9149
## s9   11264.5693
## s10  27672.3528
## s11  35116.2729
## s12  65820.9196
```

```
#
```

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
```

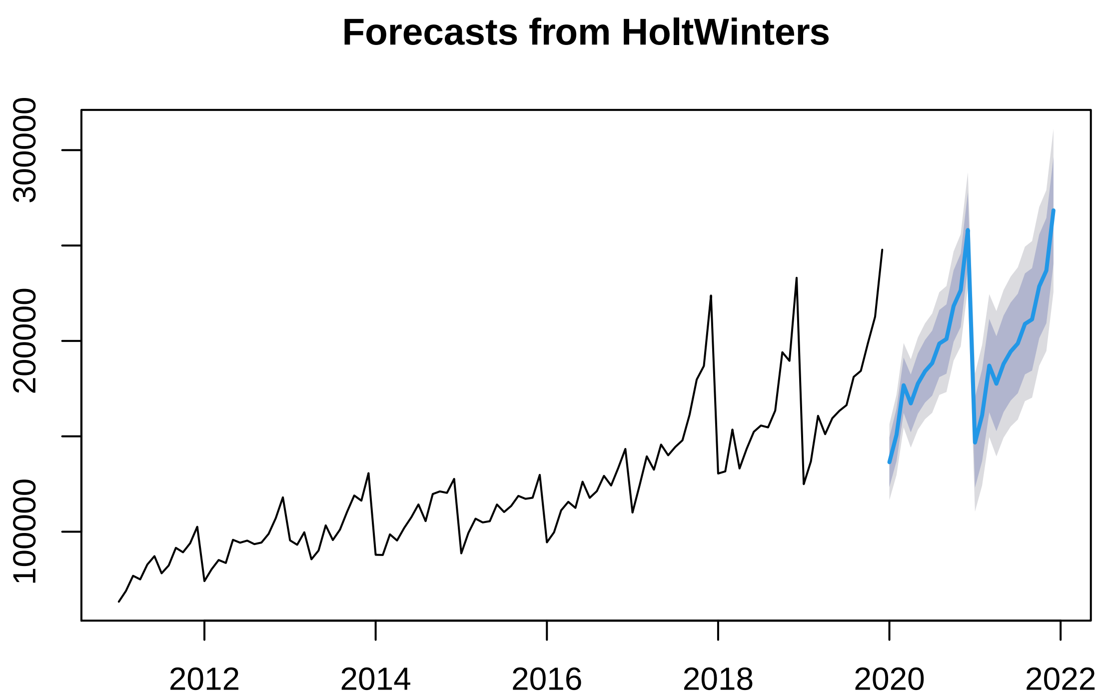
```
## [1] 9814489310
```




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

```
##          Point Forecast
```

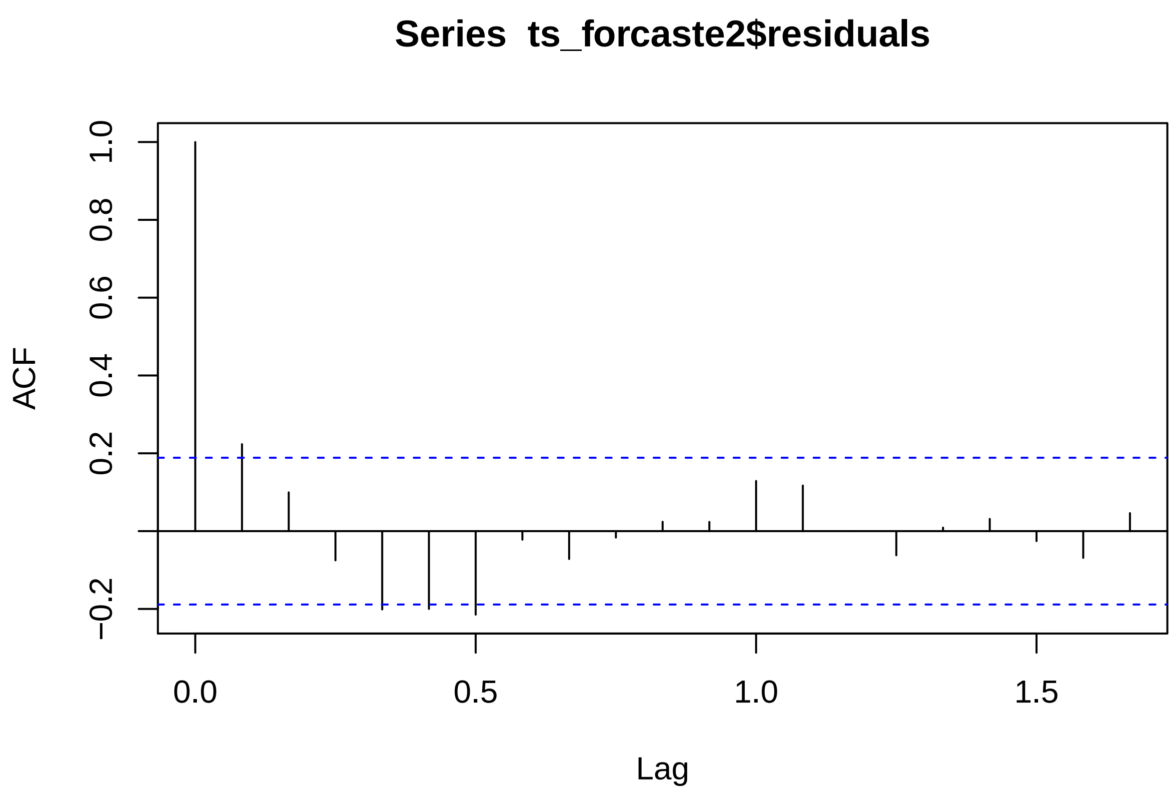
```
## Jan 2020       136440.2
## Feb 2020       150754.8
## Mar 2020       176770.0
## Apr 2020       167257.4
## May 2020       177664.0
## Jun 2020       184115.2
## Jul 2020       188359.4
## Aug 2020       198620.7
## Sep 2020       201001.4
## Oct 2020       218268.2
## Nov 2020       226571.1
## Dec 2020       258134.8
## Jan 2021       146748.3
## Feb 2021       161062.9
## Mar 2021       187078.1
## Apr 2021       177565.5
## May 2021       187972.1
## Jun 2021       194423.2
## Jul 2021       198667.5
## Aug 2021       208928.8
## Sep 2021       211309.5
## Oct 2021       228576.3
## Nov 2021       236879.2
## Dec 2021       268442.8
```



### Growth

```
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:

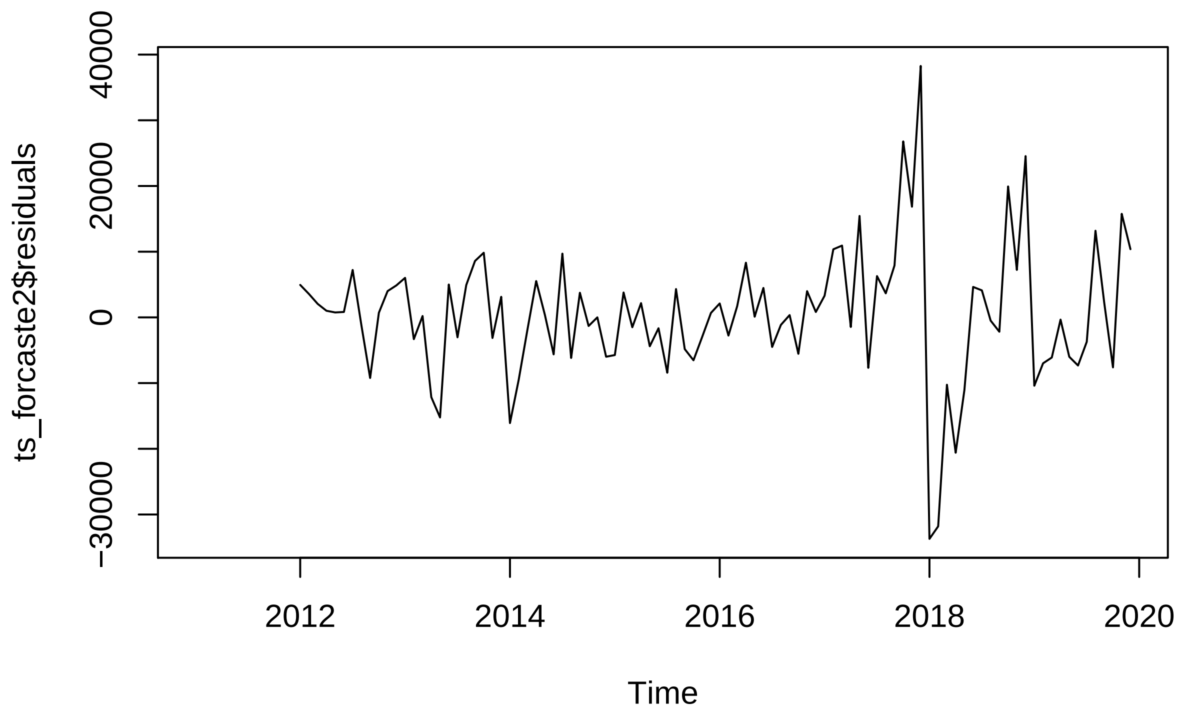



```
##
##  Box-Ljung test
##
## data:  ts_forcaste2$residuals
## X-squared = 25.284, df = 20, p-value = 0.1908
```

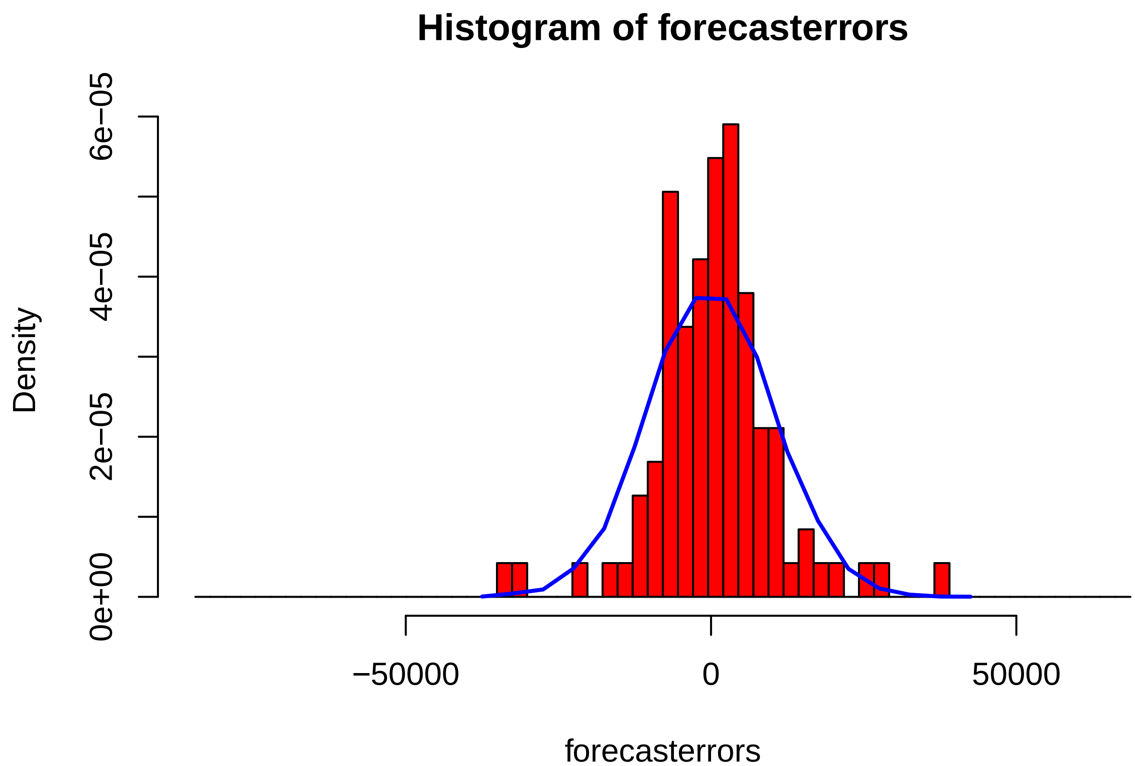
The correlogram shows that the autocorrelations for the in-sample forecast errors do not exceed the significance bounds for lags 1-20. Furthermore, the p-value for Ljung-Box test is 0.2, indicating that there is little evidence of non-zero autocorrelations at lags 1-20.

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)
```



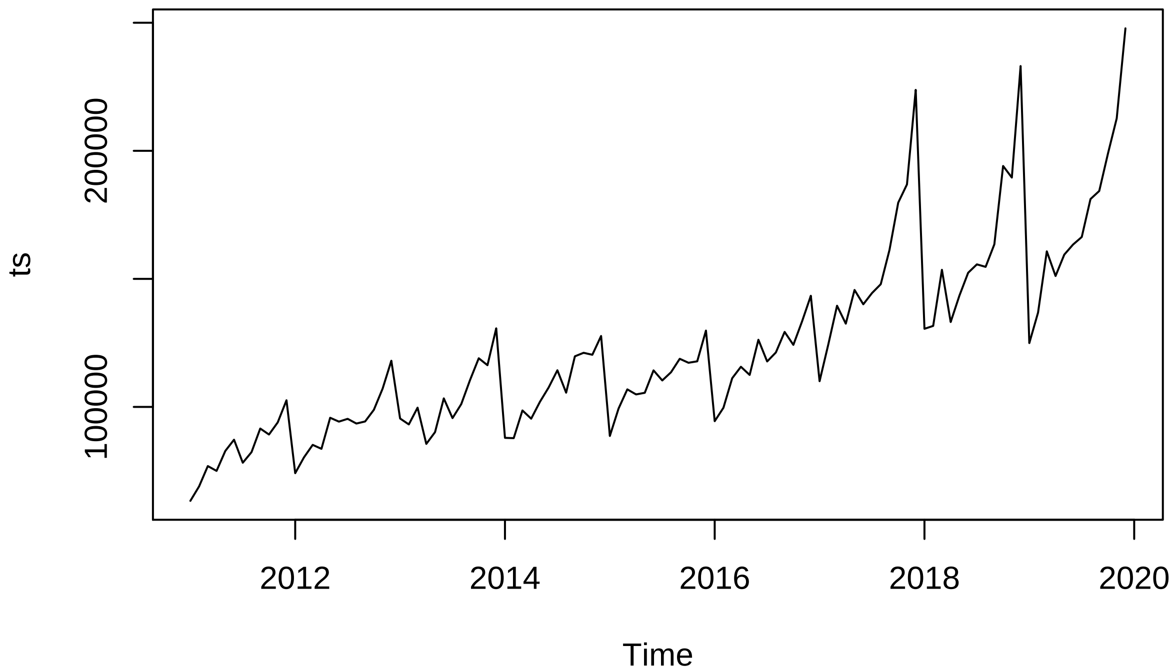

```r
plotForecastErrors(ts_forcaste2$residuals)
```



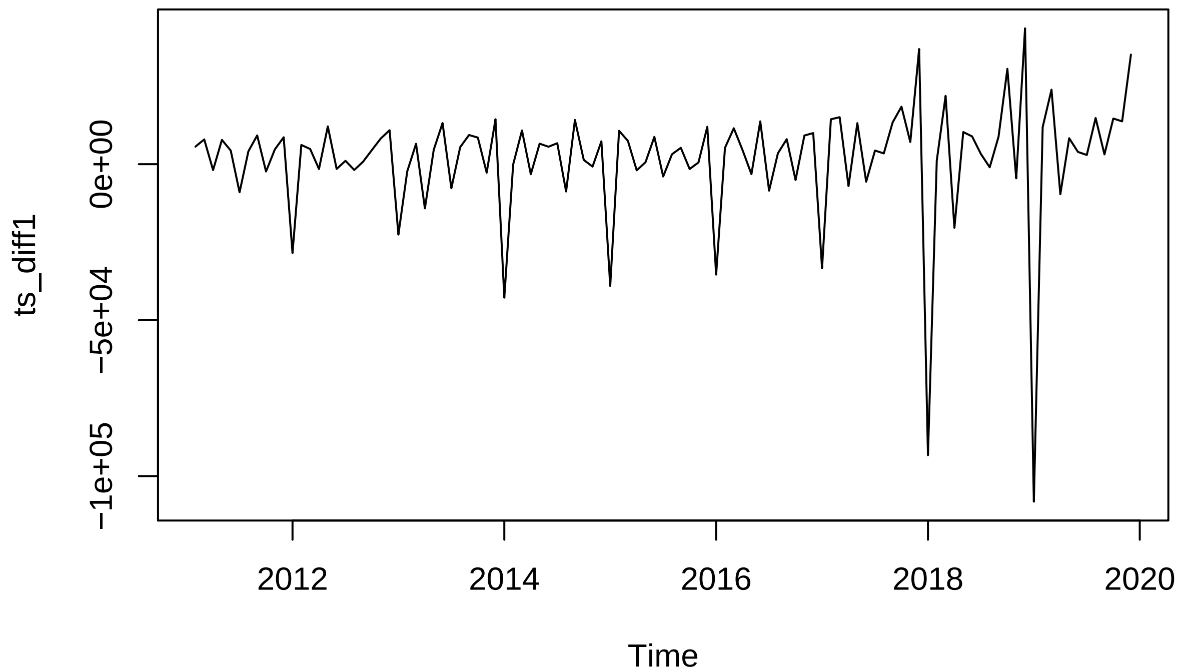

From the time plot, it appears plausible that the forecast errors have constant variance over time. From the histogram of forecast errors, it seems plausible that the forecast errors are normally distributed with mean zero.

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)
```

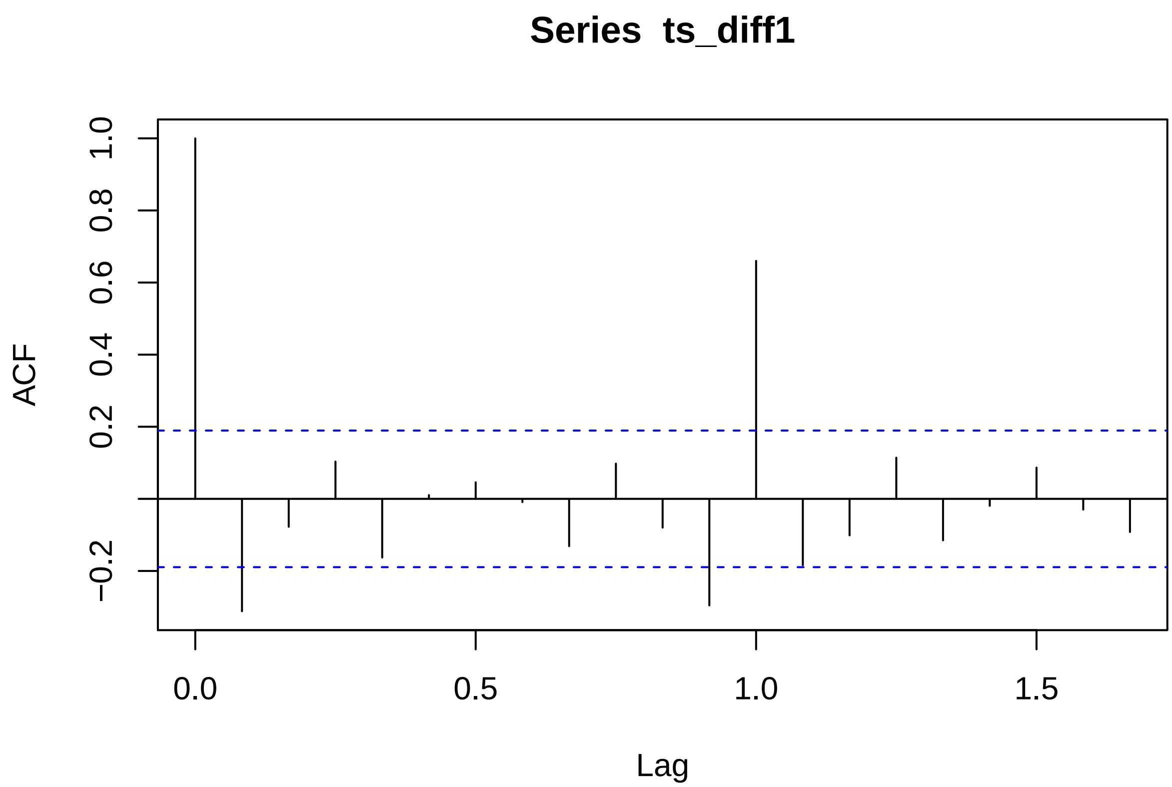



```
ts_diff1 <-  diff(ts, differences = 1)

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
```

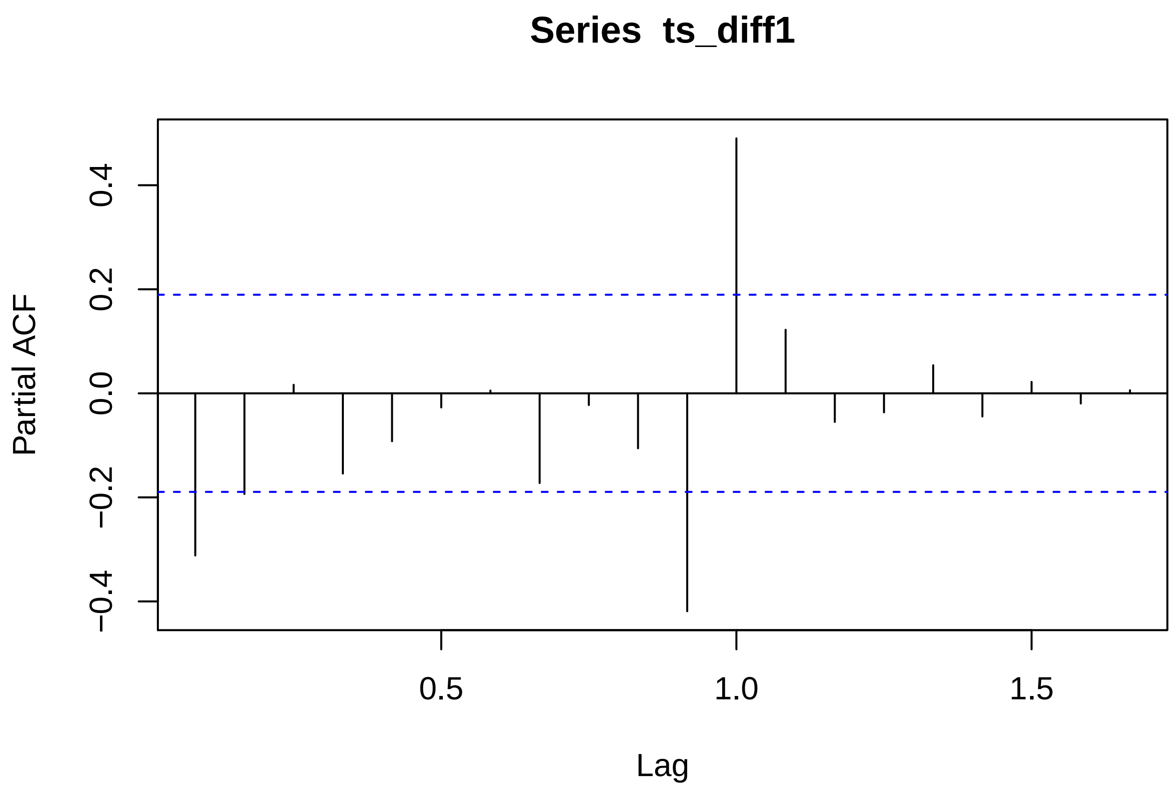



We see from the correlogram that the autocorrelation exceeds the significance bound 3 times but all the others do not exceed

```
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 -0.312 -0.078  0.103 -0.163  0.010  0.046 -0.009 -0.131  0.098 -0.080
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.296  0.660 -0.184 -0.101  0.114 -0.115 -0.019  0.087 -0.030 -0.092
```

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




```r
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
## -0.312 -0.193  0.016 -0.154 -0.092 -0.027  0.005 -0.172 -0.023 -0.106 -0.419
## 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
##  0.490  0.122 -0.055 -0.037  0.054 -0.045  0.022 -0.020  0.006
```

## Arima, 1,1,1

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

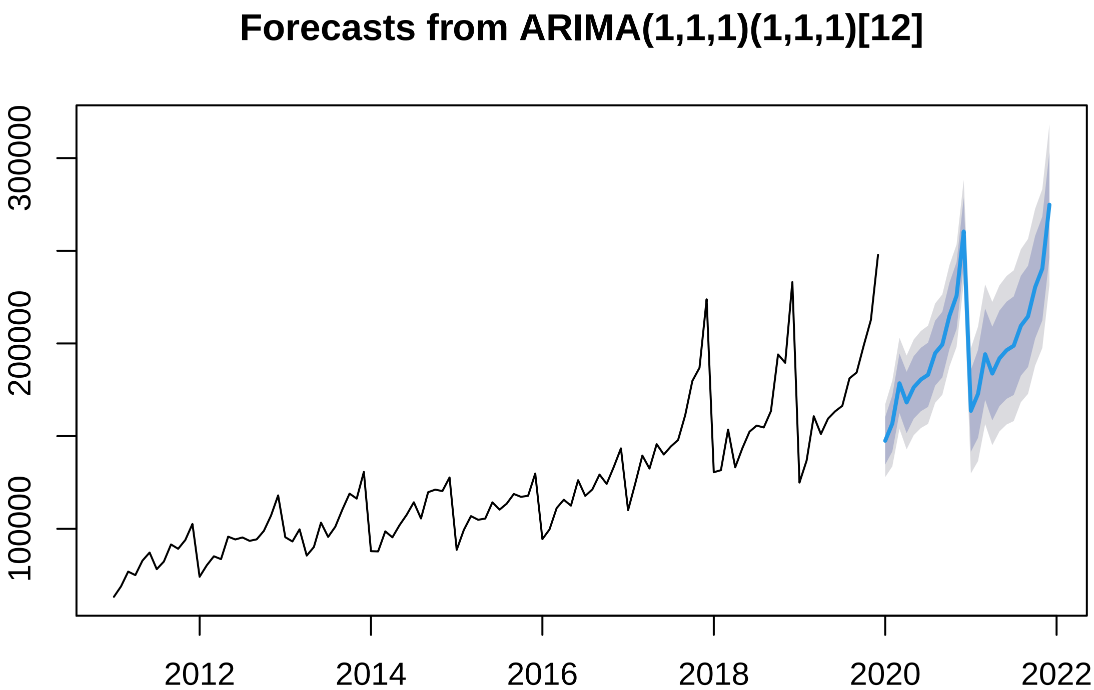
```
## Series: ts
## ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.5303  -0.9098  0.3019  -0.5237
## s.e.  0.1710   0.1144  0.3137   0.2882
```

```
##
## sigma^2 estimated as 100477930:  log likelihood=-1008.86
## AIC=2027.73   AICc=2028.4   BIC=2040.5
```

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

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020       147503.2 134657.0 160349.3 127856.7 167149.7
## Feb 2020       156812.5 141694.0 171930.9 133690.8 179934.1
## Mar 2020       178520.7 162471.5 194570.0 153975.5 203066.0
## Apr 2020       168153.4 151609.4 184697.3 142851.6 193455.2
## May 2020       176316.3 159448.7 193183.9 150519.6 202113.0
## Jun 2020       180548.0 163432.5 197663.4 154372.1 206723.8
## Jul 2020       183095.0 165768.1 200421.8 156595.8 209594.1
## Aug 2020       194784.4 177265.1 212303.8 167990.9 221578.0
## Sep 2020       199379.3 181677.9 217080.7 172307.4 226451.3
## Oct 2020       214902.8 197025.5 232780.1 187561.8 242243.8
## Nov 2020       225786.7 207737.4 243836.0 198182.7 253390.7
## Dec 2020       260407.7 242189.2 278626.2 232544.9 288270.4
## Jan 2021       163636.7 141560.0 185713.4 129873.4 197400.1
## Feb 2021       172790.7 149071.5 196509.9 136515.3 209066.0
## Mar 2021       194167.2 169530.8 218803.7 156489.0 231845.5
## Apr 2021       183753.9 158490.7 209017.2 145117.1 222390.8
## May 2021       191957.1 166198.7 217715.6 152563.0 231351.3
## Jun 2021       196330.2 170141.8 222518.7 156278.5 236382.0
## Jul 2021       198777.1 172193.5 225360.8 158120.9 239433.4
## Aug 2021       209540.0 182581.0 236498.9 168309.8 250770.2
## Sep 2021       214579.2 187257.4 241901.0 172794.1 256364.2
## Oct 2021       230371.7 202695.6 258047.7 188044.8 272698.5
## Nov 2021       240397.6 212373.7 268421.4 197538.8 283256.4
## Dec 2021       274863.2 246496.8 303229.6 231480.6 318245.8
```

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



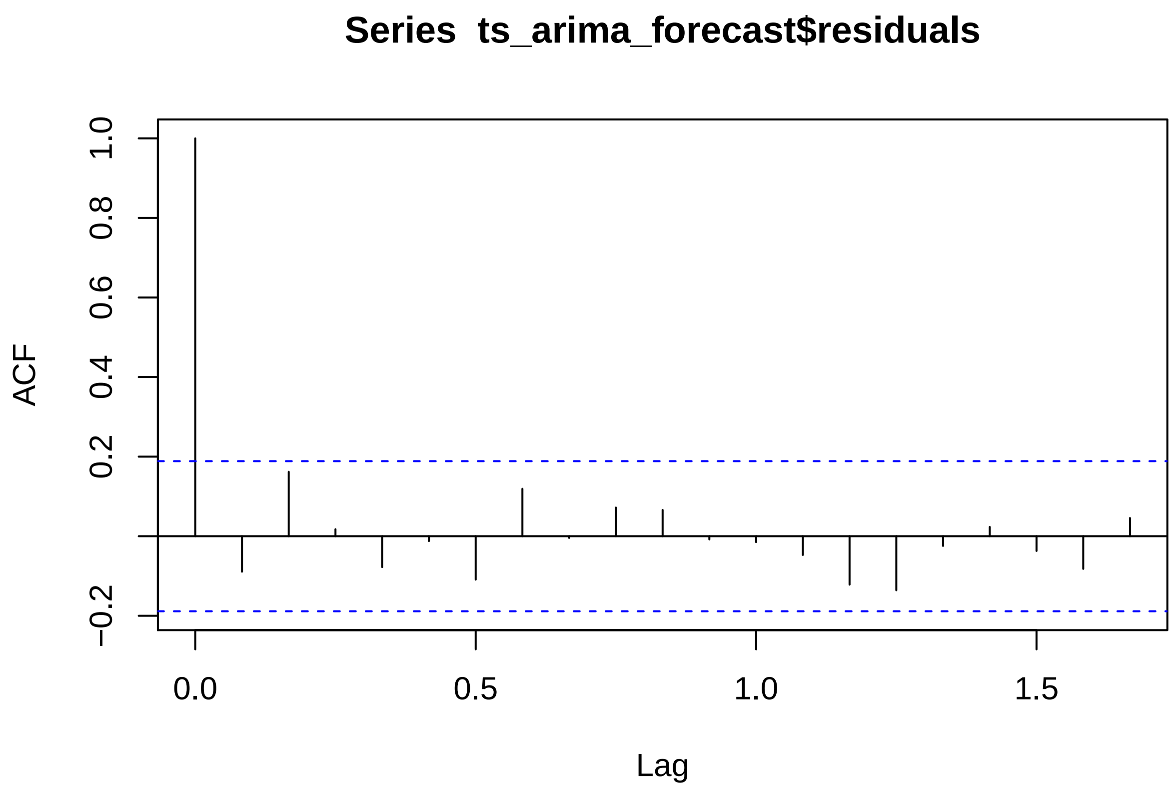

## Growth

```
# 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),]
# 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_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)
```

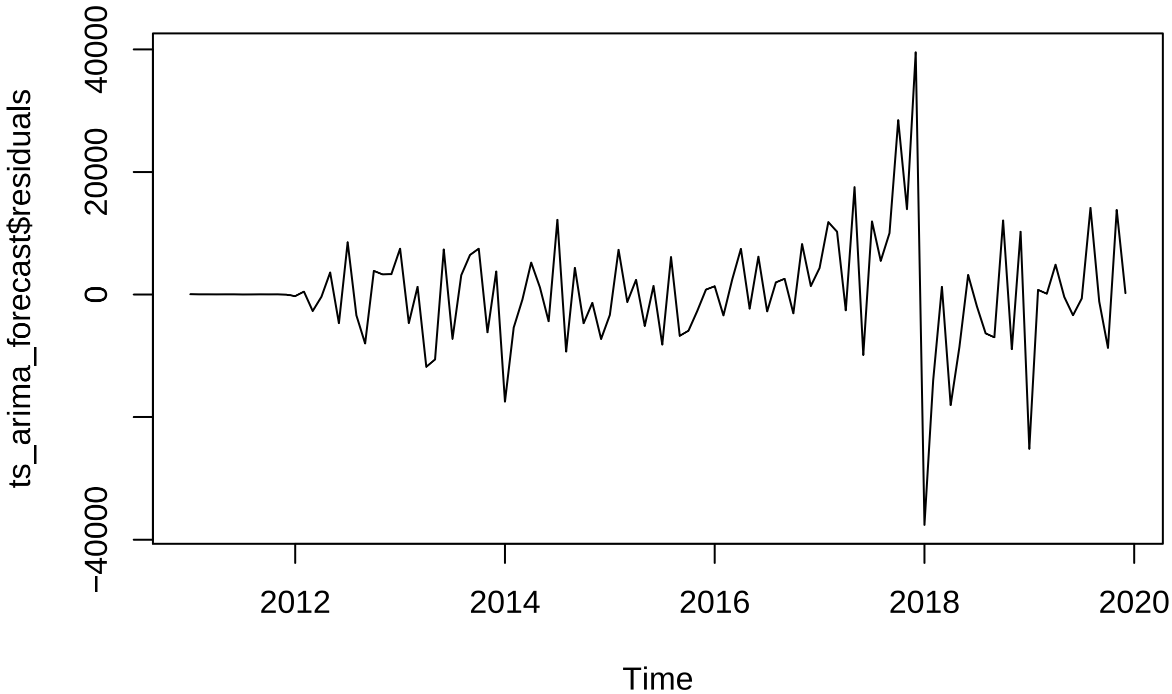



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

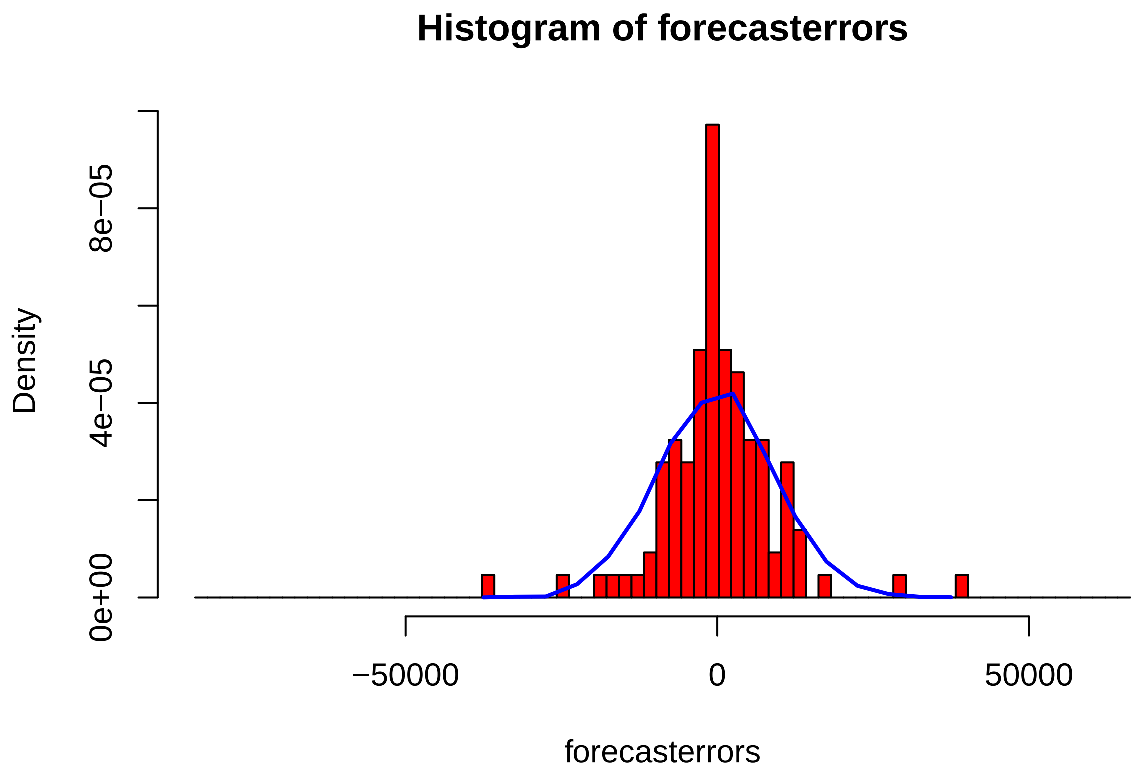
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 14.835, df = 20, p-value = 0.7858
```

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

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




```r
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
```

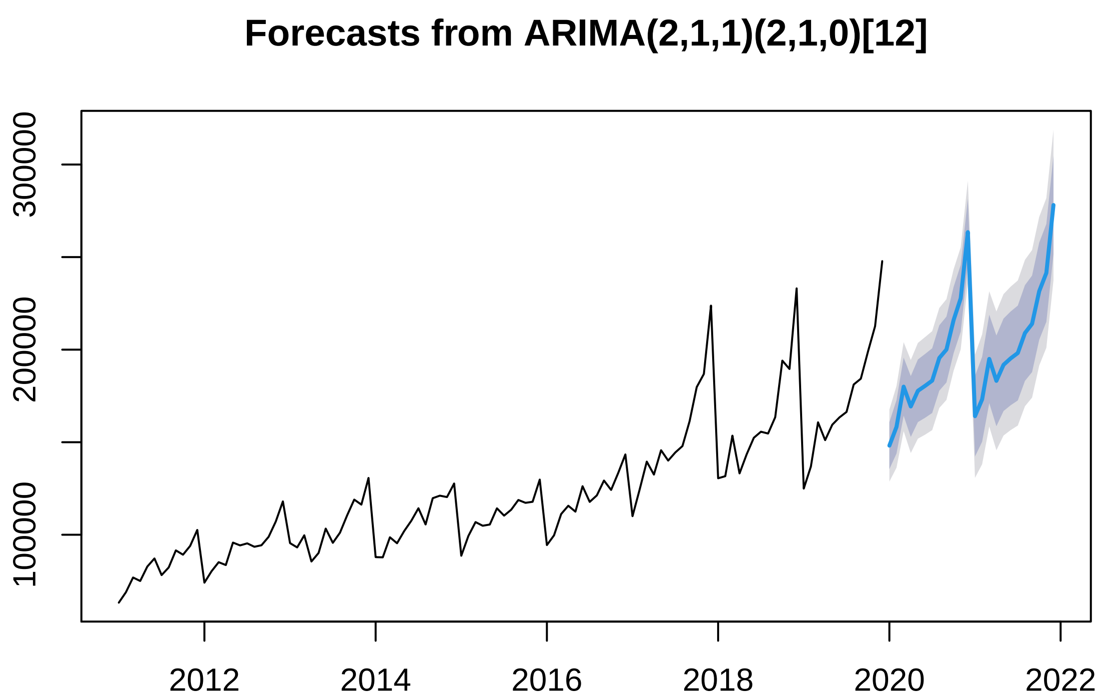
```
## Series: ts
## ARIMA(2,1,1)(2,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1     sar1     sar2
##       0.5014  0.1847  -0.9647  -0.1801  -0.1210
## s.e.  0.1104  0.1095   0.0545   0.1072   0.1275
##
## sigma^2 estimated as 98212454:  log likelihood=-1007.38
## AIC=2026.76   AICc=2027.71   BIC=2042.08
```

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

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020       148158.0 135457.0 160858.9 128733.5 167582.4
## Feb 2020       158278.8 143863.5 172694.1 136232.5 180325.1
## Mar 2020       180009.1 164311.4 195706.7 156001.6 204016.6
## Apr 2020       169295.4 152873.1 185717.8 144179.6 194411.3
## May 2020       177769.1 160862.8 194675.5 151913.1 203625.2
## Jun 2020       180412.5 163176.7 197648.4 154052.5 206772.6
```

```
## Jul 2020       183251.2 165777.6 200724.9 156527.6 209974.9
## Aug 2020       195512.8 177859.8 213165.7 168514.9 222510.6
## Sep 2020       200057.6 182263.4 217851.8 172843.7 227271.5
## Oct 2020       215970.9 198060.8 233881.1 188579.7 243362.1
## Nov 2020       227724.5 209715.6 245733.3 200182.4 255266.6
## Dec 2020       263501.5 245406.0 281596.9 235826.9 291176.1
## Jan 2021       164047.6 142279.7 185815.4 130756.4 197338.7
## Feb 2021       173142.9 150211.5 196074.4 138072.3 208213.5
## Mar 2021       194991.7 171110.3 218873.2 158468.3 231515.2
## Apr 2021       183147.4 158642.3 207652.5 145670.1 220624.7
## May 2021       191818.4 166846.9 216789.9 153627.8 230009.0
## Jun 2021       195284.0 169955.0 220613.0 156546.7 234021.4
## Jul 2021       198175.2 172558.1 223792.2 158997.3 237353.0
## Aug 2021       208976.0 183118.1 234833.9 169429.8 248522.2
## Sep 2021       213949.9 187883.6 240016.3 174084.8 253815.0
## Oct 2021       231559.6 205307.5 257811.7 191410.5 271708.7
## Nov 2021       241456.2 215034.7 267877.8 201048.0 281864.5
## Dec 2021       278133.8 251554.5 304713.0 237484.4 318783.2
```

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




## Growth

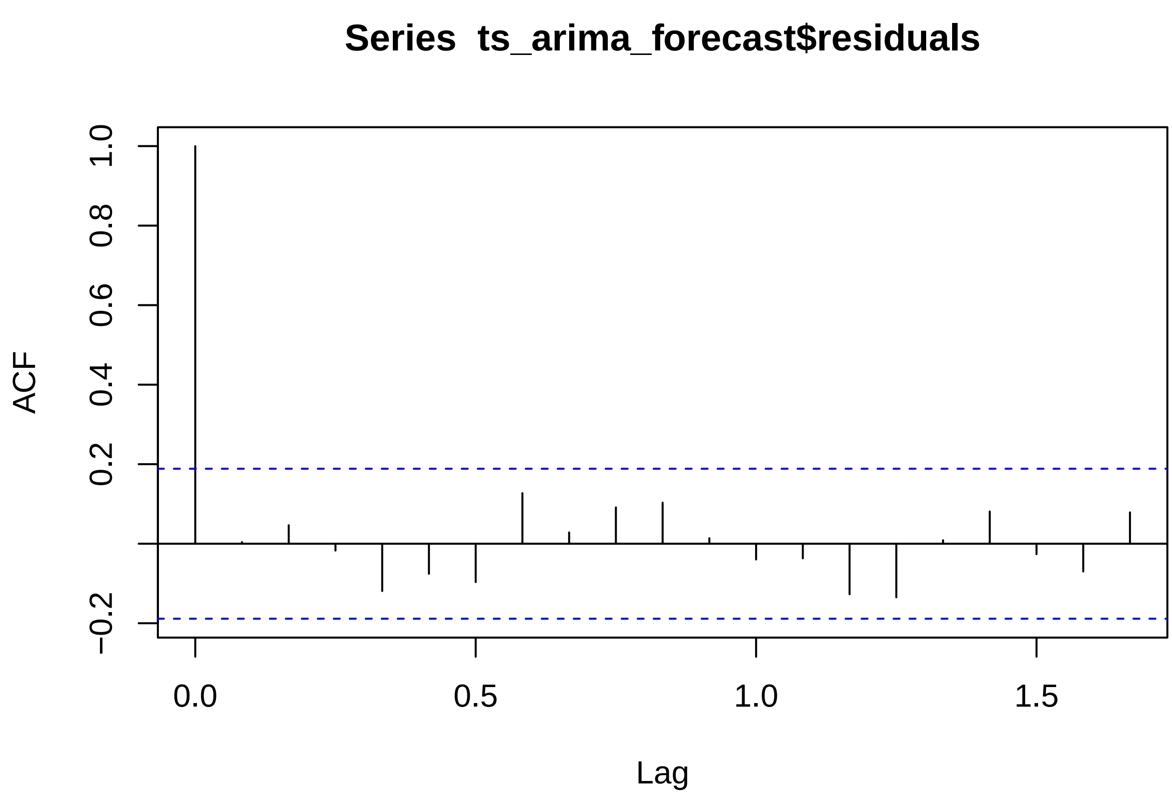
```
# this_year_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1]
#
# year_2019_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1][c(1:2),]
```

```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)
```

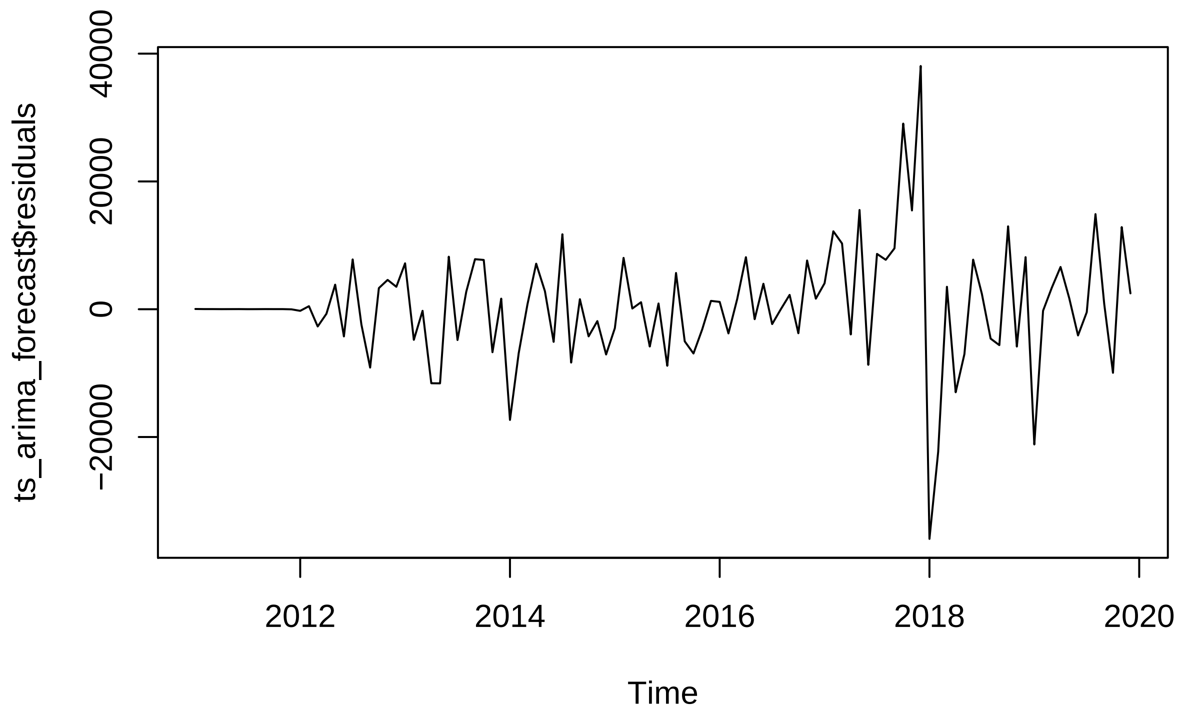



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

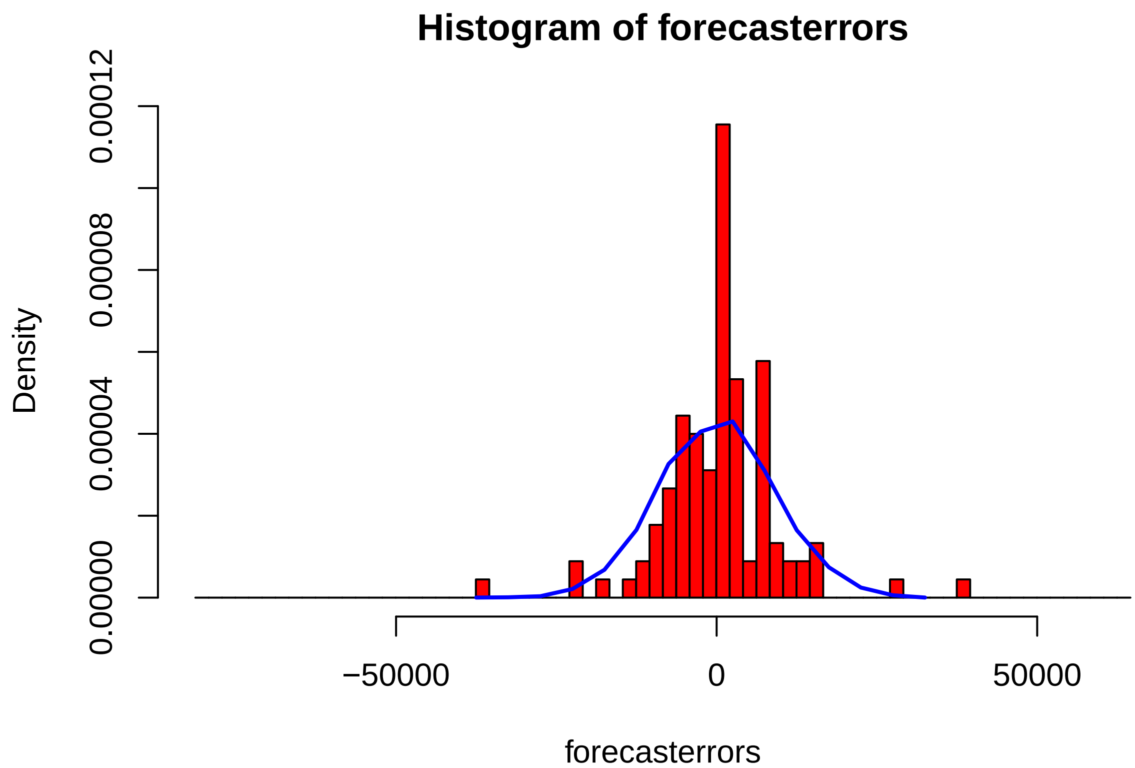
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 15.147, df = 20, p-value = 0.768
```

**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)
```

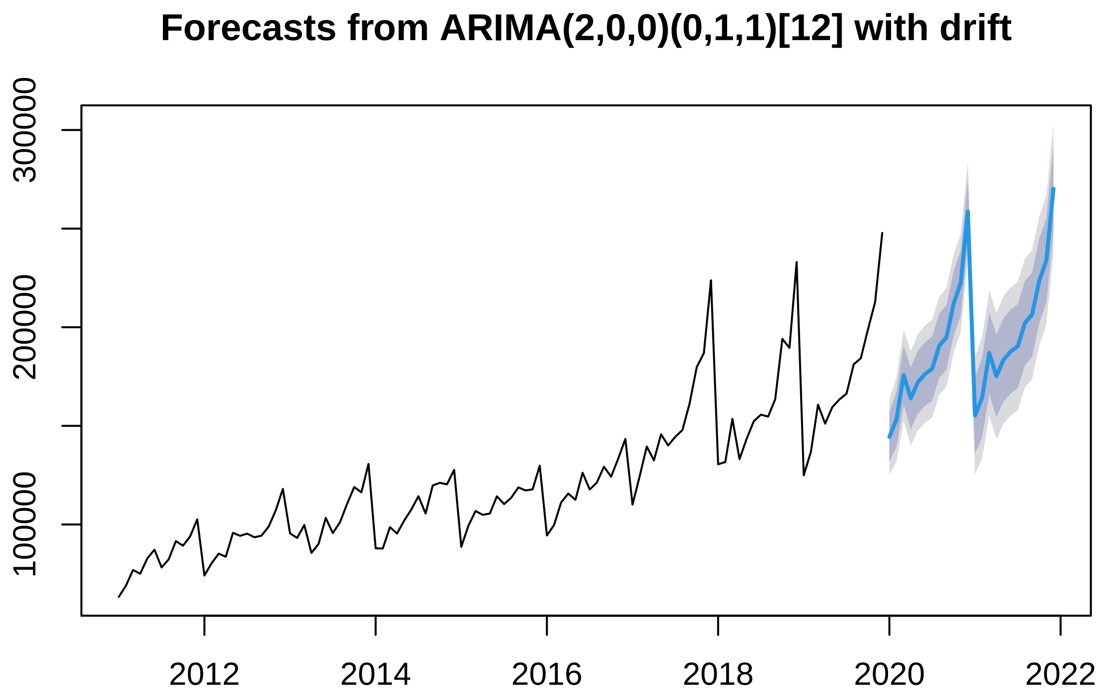



## A model chosen automatically

```
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(2,0,0)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1     ar2     sma1     drift
##       0.5027  0.1991  -0.1932  966.1516
## s.e.  0.0991  0.0995   0.1129  219.5238
##
## sigma^2 estimated as 95095303:  log likelihood=-1016.47
## AIC=2042.93   AICc=2043.6   BIC=2055.75
```

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




```
# str(fit)
```

### Growth

```
# year_2021_predict_auto.arima <- (as.data.frame(fit_forecast))[1]
# year_2021_predict_auto.arima_95_low <- (as.data.frame(fit_forecast))[4]
# year_2021_predict_auto.arima_95_high <- (as.data.frame(fit_forecast))[5]
#
# growth_auto.arima <- growth(sum(year_2021_predict_auto.arima),sum(year_2020))
# growth_auto.arima_95_low <- growth(sum(year_2021_predict_auto.arima_95_low),sum(year_2020))
# growth_auto.arima_95_high <- growth(sum(year_2021_predict_auto.arima_95_high),sum(year_2020))
#
# growth_auto.arima
# growth_auto.arima_95_low
# growth_auto.arima_95_high


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 =  -growth_ARIMA
#
# growth_ARIMA2 = -growth_ARIMA2
#
# growth_auto.arima
#
# growth_HW

growth_ARIMA_20
```

```
## [1] 0.09503449
```

```r
growth_ARIMA_21
```

```
## [1] 0.08090016
```

```r
growth_ARIMA2_20
```

```
## [1] 0.1016115
```

```r
growth_ARIMA2_21
```

```
## [1] 0.07597644
```

```r
growth_auto.arima_20
```

```
## [1] 0.07442554
```

```r
growth_auto.arima_21
```

```
## [1] 0.06070137
```