Time Series Forcasting report

Kevork Sulahian

September 26, 2019

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

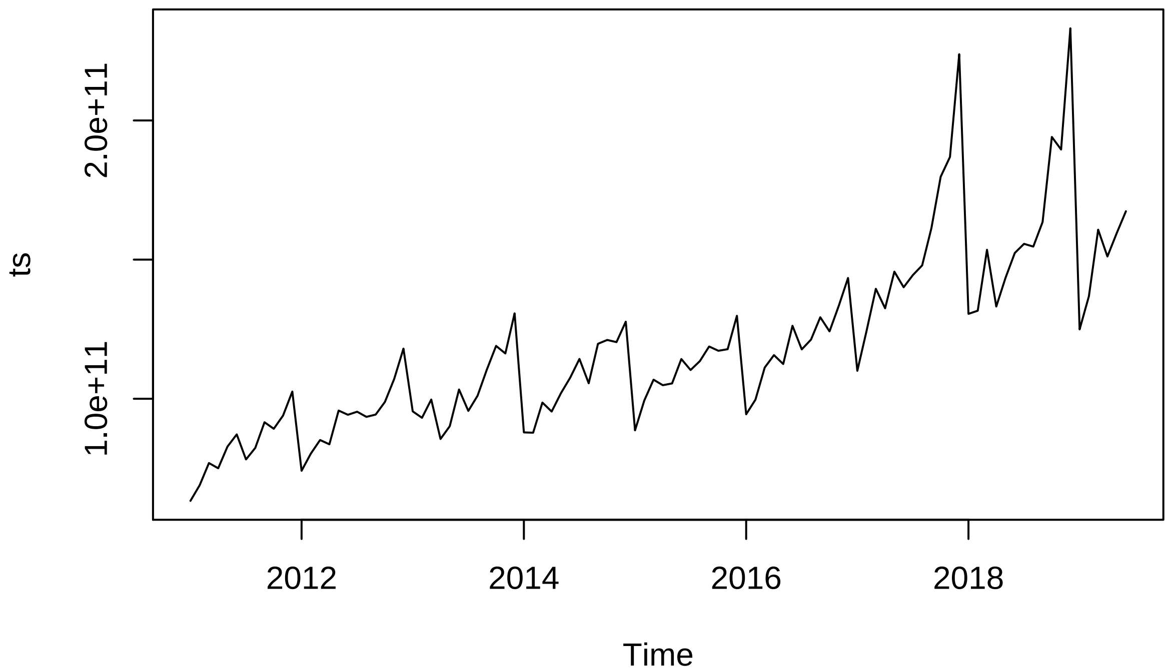
```r
df <- read_xls('economy.xls', sheet='2011-2019 NACE 2')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * `` -> ...6
## * ... and 99 more problems
```

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

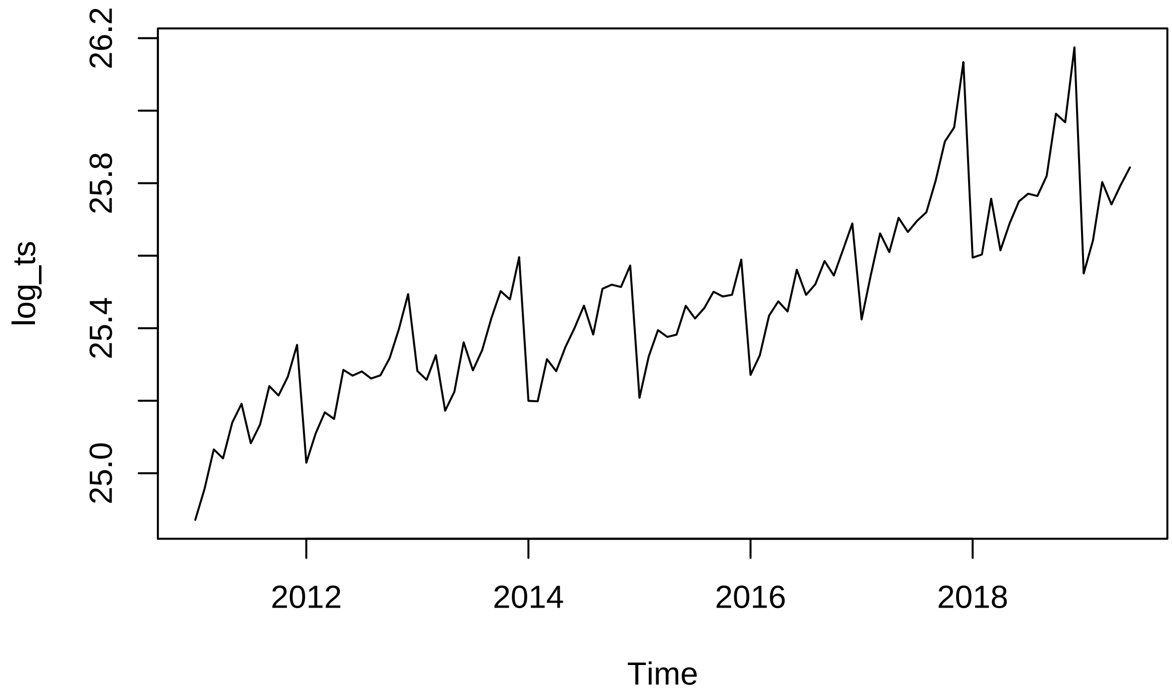
```
## Warning: Setting row names on a tibble is deprecated.
```

```r
df = df[-1]
df = t(df)
df[] <- sapply(df[],function(x) as.numeric(as.character(x)))
df = as.numeric(df)
df= df * 1000000
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.

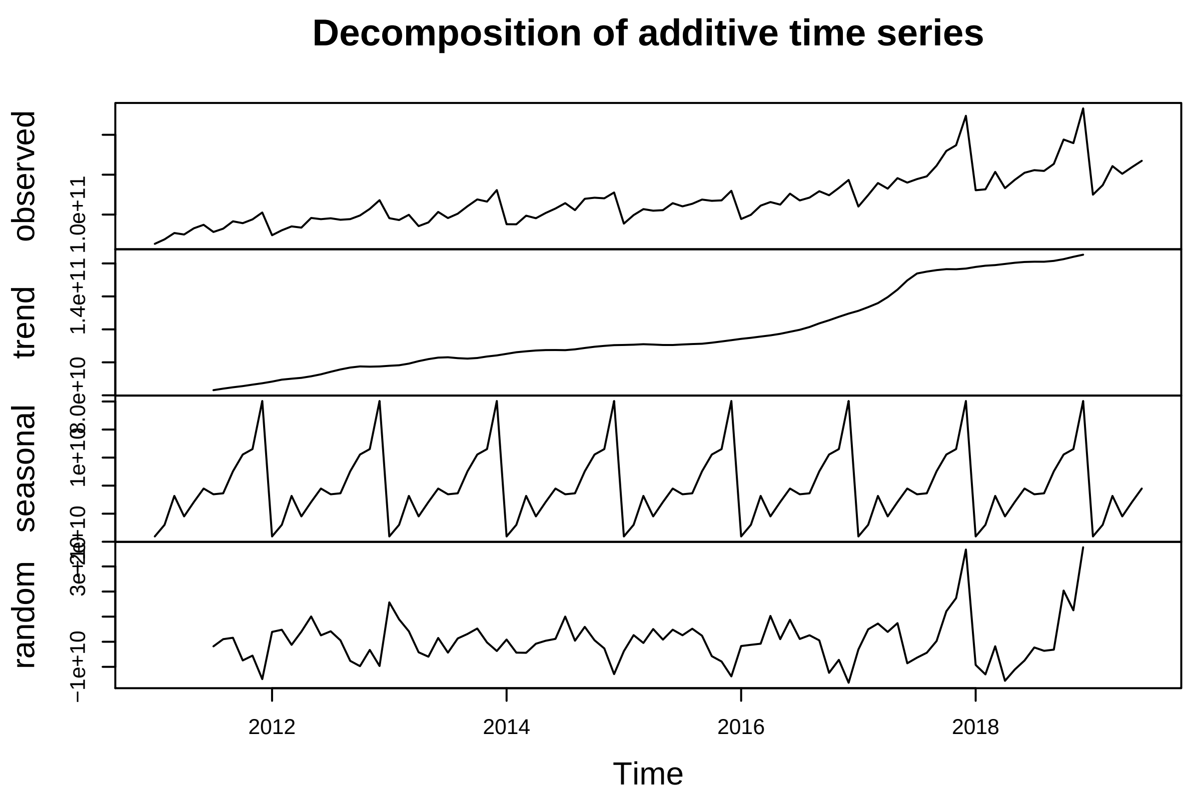
```
ts_components <- decompose(ts)
```

we can print out the estimated values of the seasonal component

```
ts_components$seasonal
```

```
##              Jan          Feb          Mar          Apr          May
## 2011 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2012 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2013 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2014 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2015 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
```

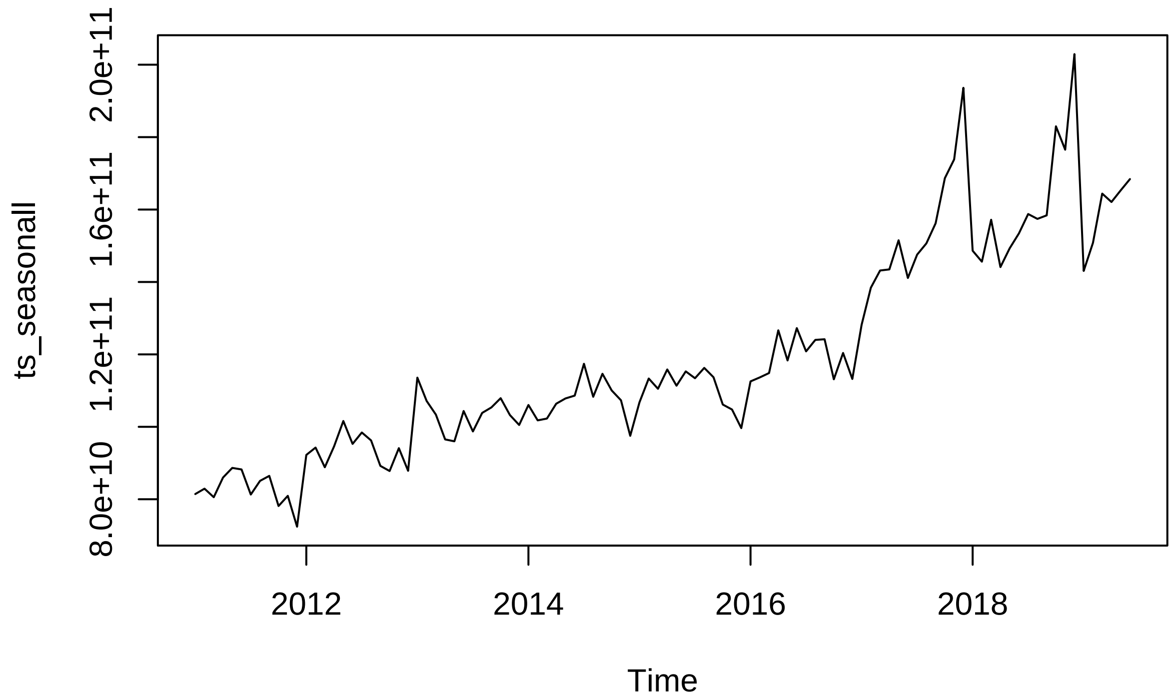
```
## 2016 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2017 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2018 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
## 2019 -18114721416 -13972761892  -3652881535 -10968033321  -5840660702
##              Jun          Jul          Aug          Sep          Oct
## 2011  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2012  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2013  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2014  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2015  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2016  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2017  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2018  -1030591654  -3088941580  -2737647830   5107804253  11091117274
## 2019  -1030591654
##              Nov          Dec
## 2011  13028793837  30178524566
## 2012  13028793837  30178524566
## 2013  13028793837  30178524566
## 2014  13028793837  30178524566
## 2015  13028793837  30178524566
## 2016  13028793837  30178524566
## 2017  13028793837  30178524566
## 2018  13028793837  30178524566
## 2019
```



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

```r
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: 0.3841348
##  beta : 0
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   166929650345
## b      859005492
## s1    2680812331
## s2    1004407035
```

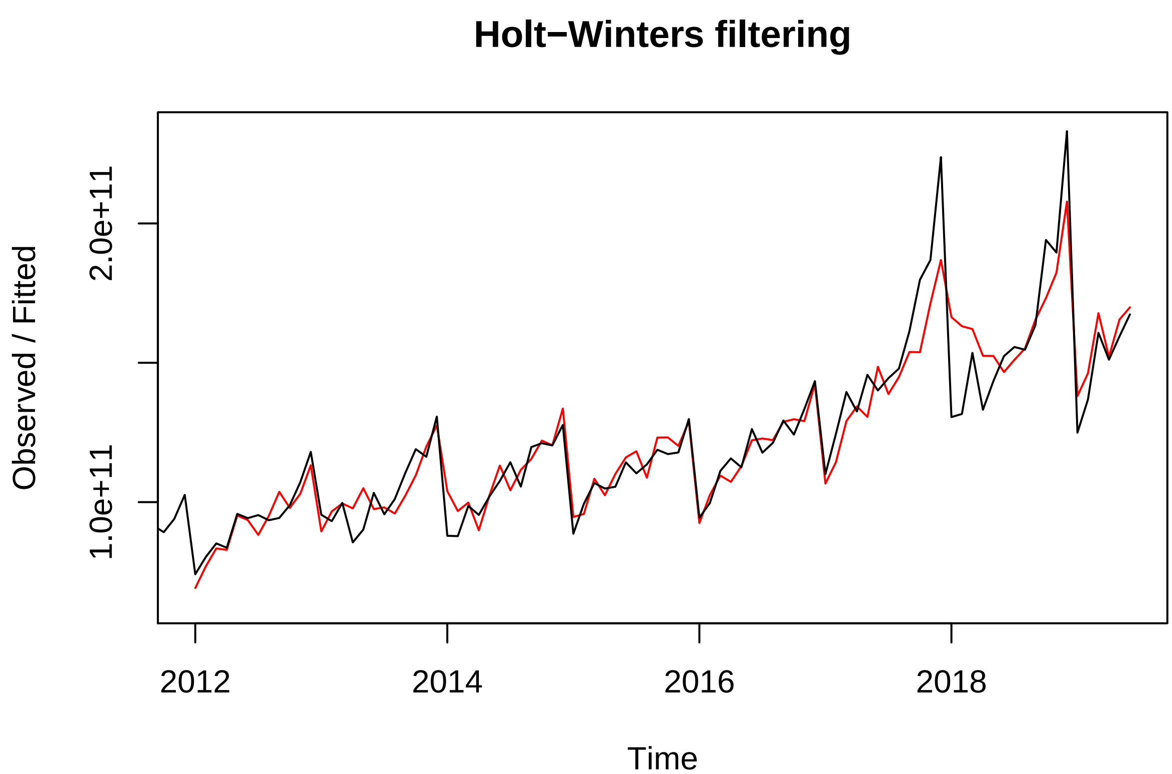
```
## s3     9728353964
## s4    31427608853
## s5    23274665015
## s6    56266579294
## s7   -47717485979
## s8   -33061264111
## s9    -7294279687
## s10 -17388238299
## s11   -7579809787
## s12     458749655
```

```
#
```

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

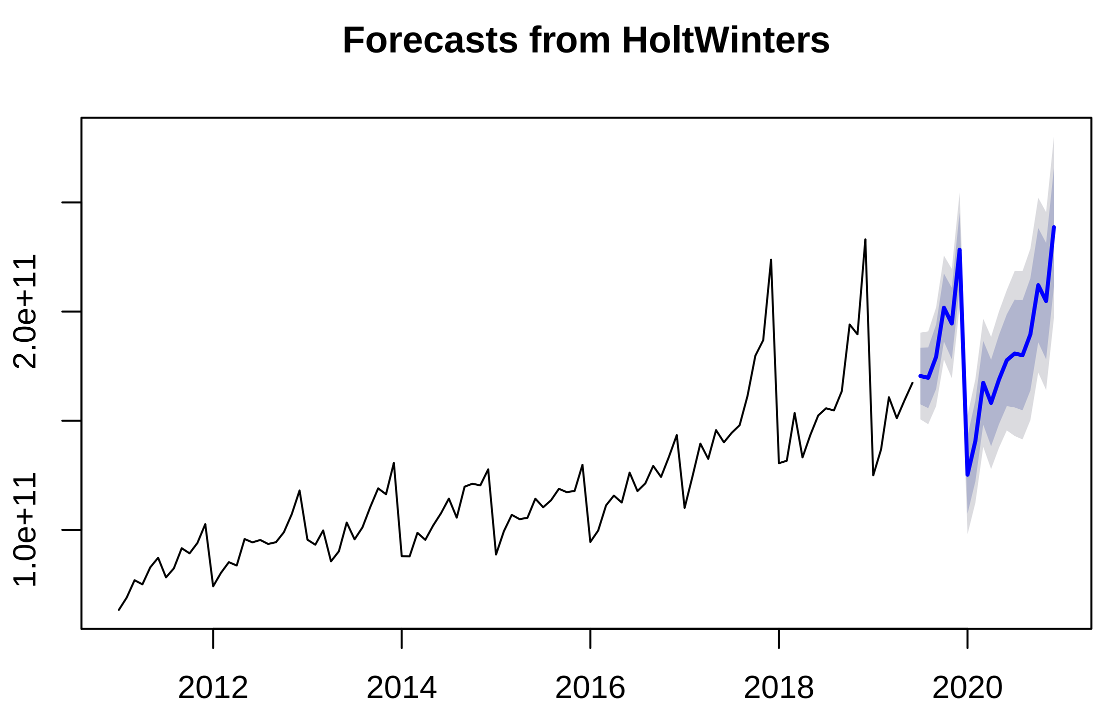
```
ts_forcaste$SSE
```

```
## [1] 9.131448e+21
```



```r
ts_forcaste2 = forecast:::forecast.HoltWinters(ts_forcaste, h= 18)
(as.data.frame(ts_forcaste2))[1]
```

```
##          Point Forecast
## Jul 2019   170469468169
## Aug 2019   169652068366
## Sep 2019   179235020787
## Oct 2019   201793281168
## Nov 2019   194499342823
## Dec 2019   228350262594
## Jan 2020   125225202813
## Feb 2020   140740430174
## Mar 2020   167366420090
## Apr 2020   158131466971
## May 2020   168798900975
## Jun 2020   177696465909
## Jul 2020   180777534078
## Aug 2020   179960134275
## Sep 2020   189543086696
## Oct 2020   212101347077
## Nov 2020   204807408732
## Dec 2020   238658328503
```

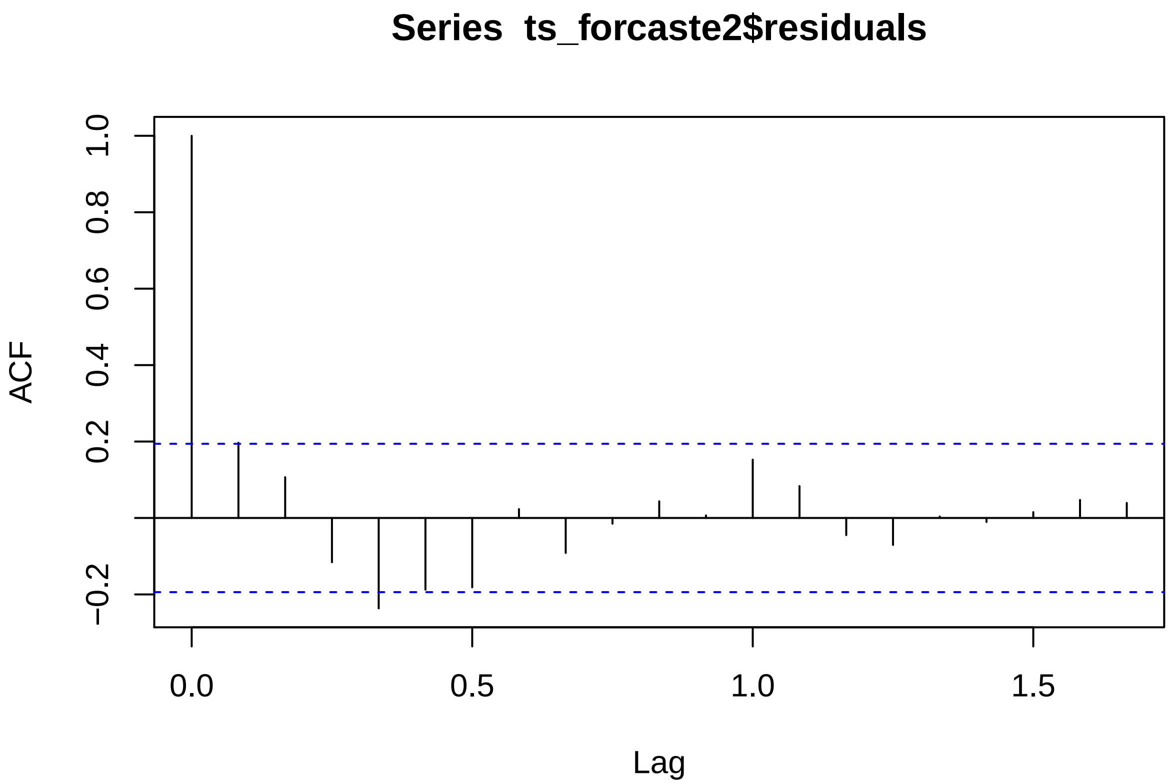


Growth

```
year_2019 <- window(ts, 2019)
year_2019_predict_HW <- (as.data.frame(ts_forcaste2))[1][c(1:6),]
sum_year_2019 = sum(c(year_2019,year_2019_predict_HW))
year_2020 = (as.data.frame(ts_forcaste2))[1][c(7:18),]
growth_HW <- growth(sum(year_2020),sum_year_2019)
growth_HW
```

```
## [1] 0.0485728
```

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



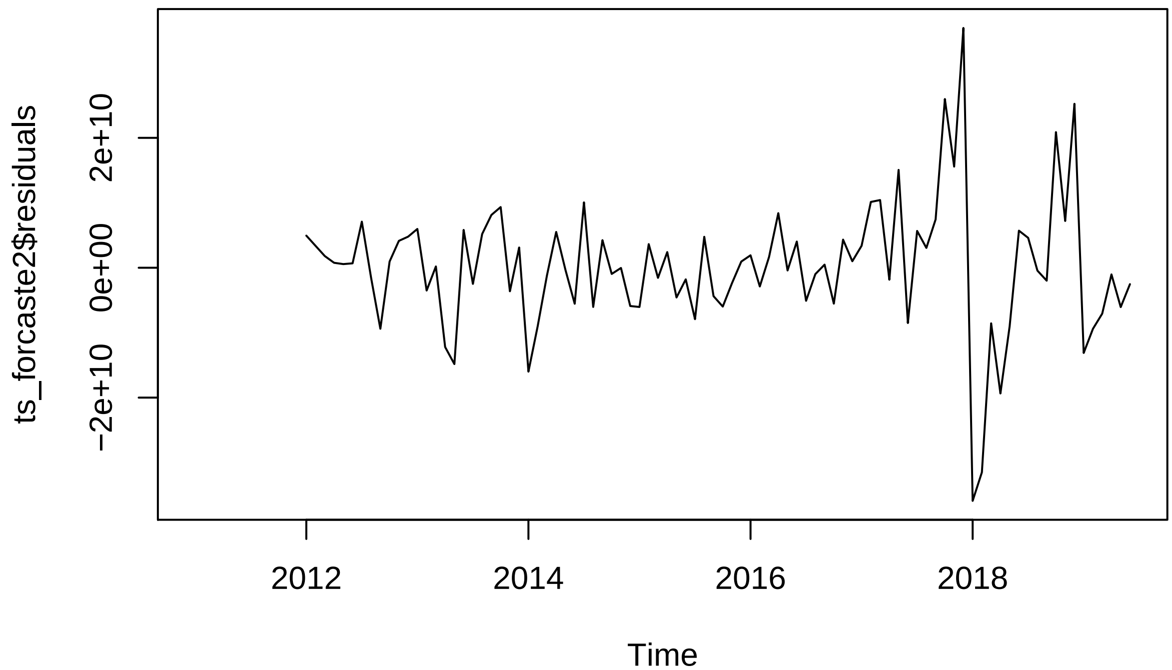
Ljung-Box test:

```
##
##  Box-Ljung test
##
## data:  ts_forcaste2$residuals
## X-squared = 23.626, df = 20, p-value = 0.2591
```

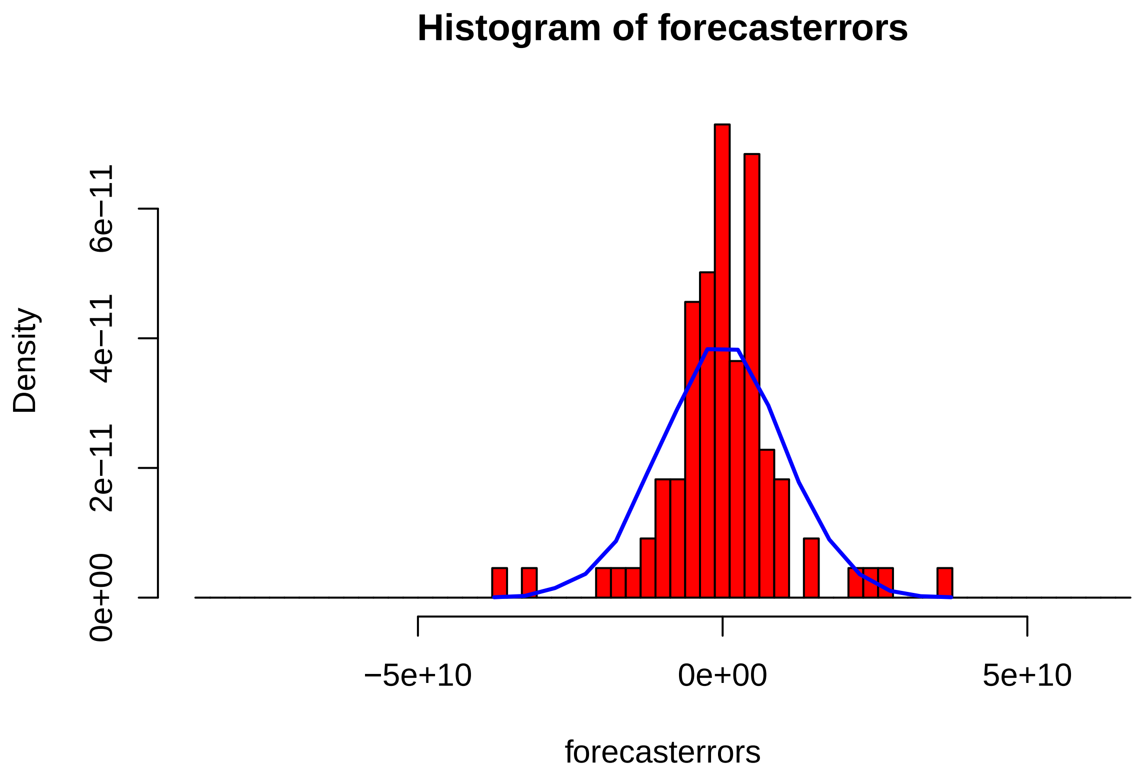
The correlogram shows that the autocorrelations for the in-sample forecast errors do not exceed the significance bounds for lags 1-20. Furthermore, the p-value for Ljung-Box test is 0.2, indicating that there is little evidence of non-zero autocorrelations at lags 1-20.

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)
```



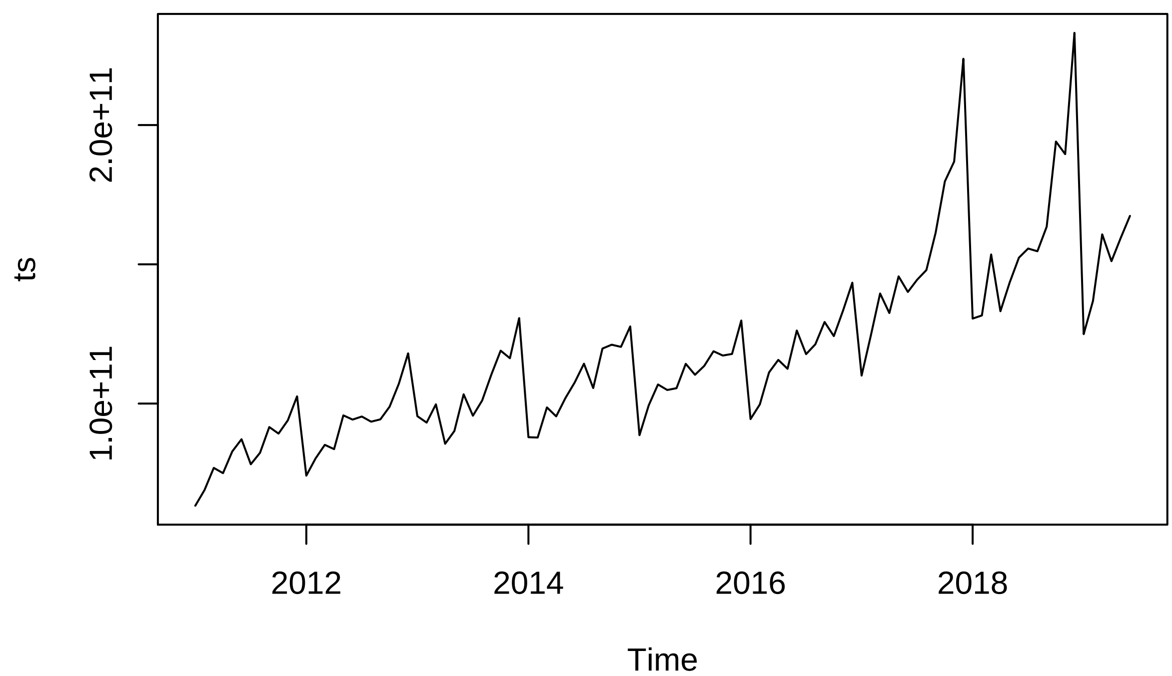

```r
plotForecastErrors(ts_forcaste2$residuals)
```



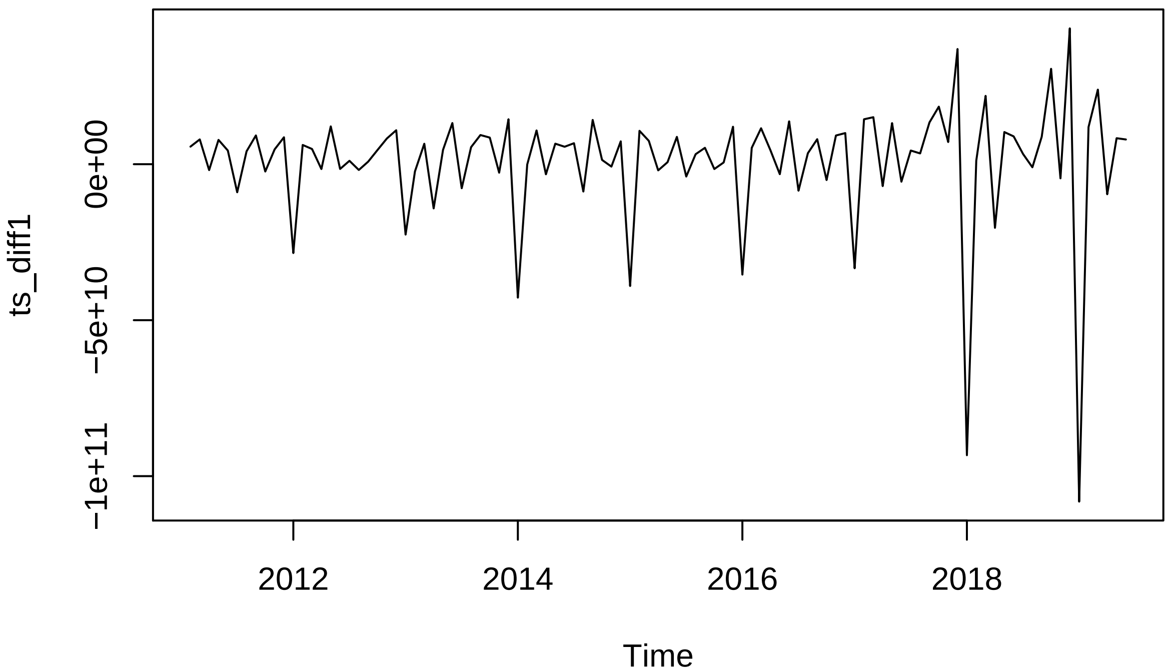
From the time plot, it appears plausible that the forecast errors have constant variance over time. From the histogram of forecast errors, it seems plausible that the forecast errors are normally distributed with mean zero.

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)
```

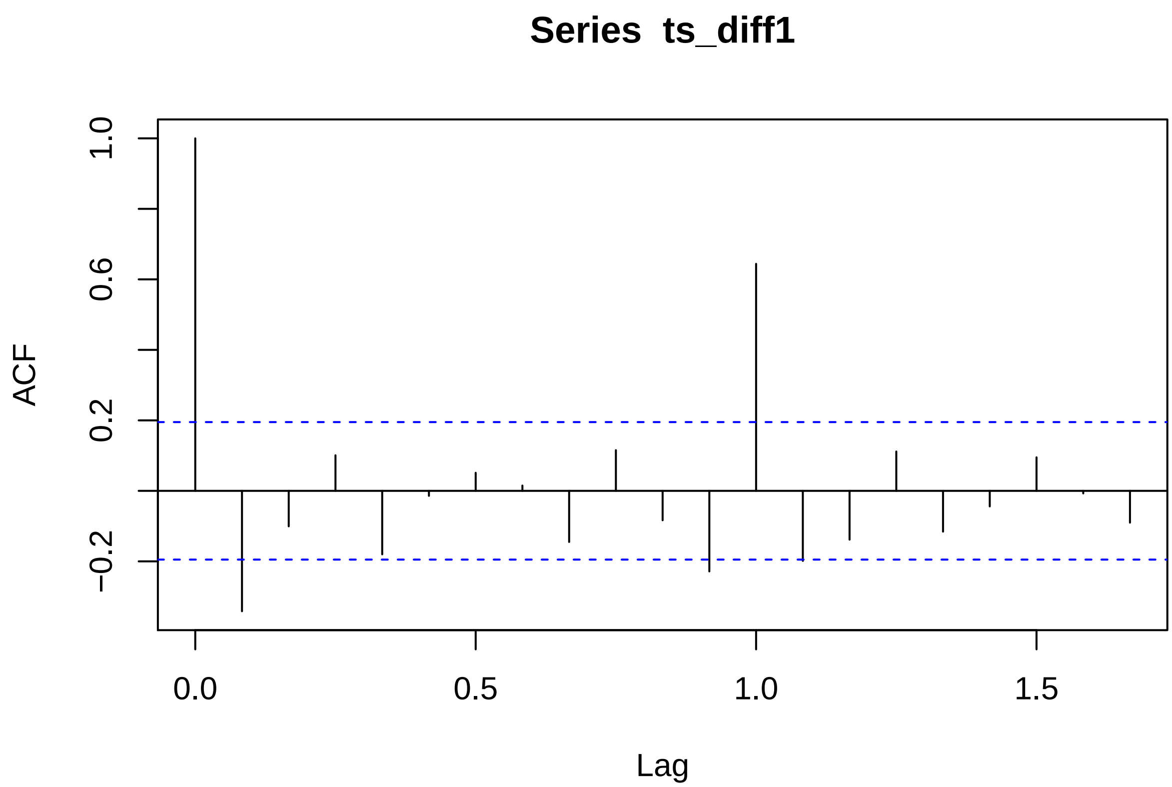



```
ts_diff1 <-  diff(ts, differences = 1)

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
```

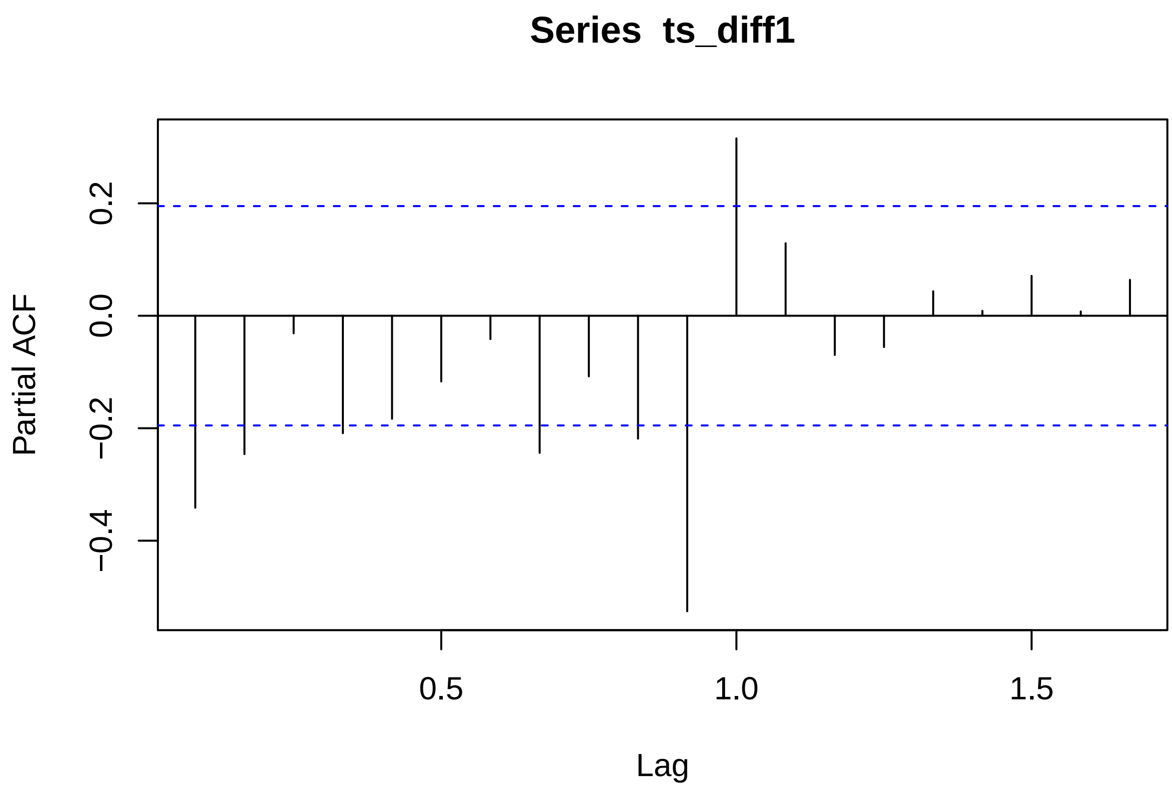


We see from the correlogram that the autocorrelation exceeds the significance bound 3 times but all the others do not exceed

```
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
##  1.000 -0.342 -0.101  0.101 -0.180 -0.014  0.051  0.015 -0.145  0.115
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
## -0.083 -0.228  0.644 -0.199 -0.138  0.112 -0.116 -0.044  0.095 -0.007
## 1.6667
## -0.090
```

```
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
## -0.342 -0.246 -0.031 -0.209 -0.183 -0.117 -0.041 -0.244 -0.108 -0.219
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.525  0.315  0.129 -0.070 -0.056  0.044  0.009  0.071  0.008  0.064
```

Arima, 1,1,1

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

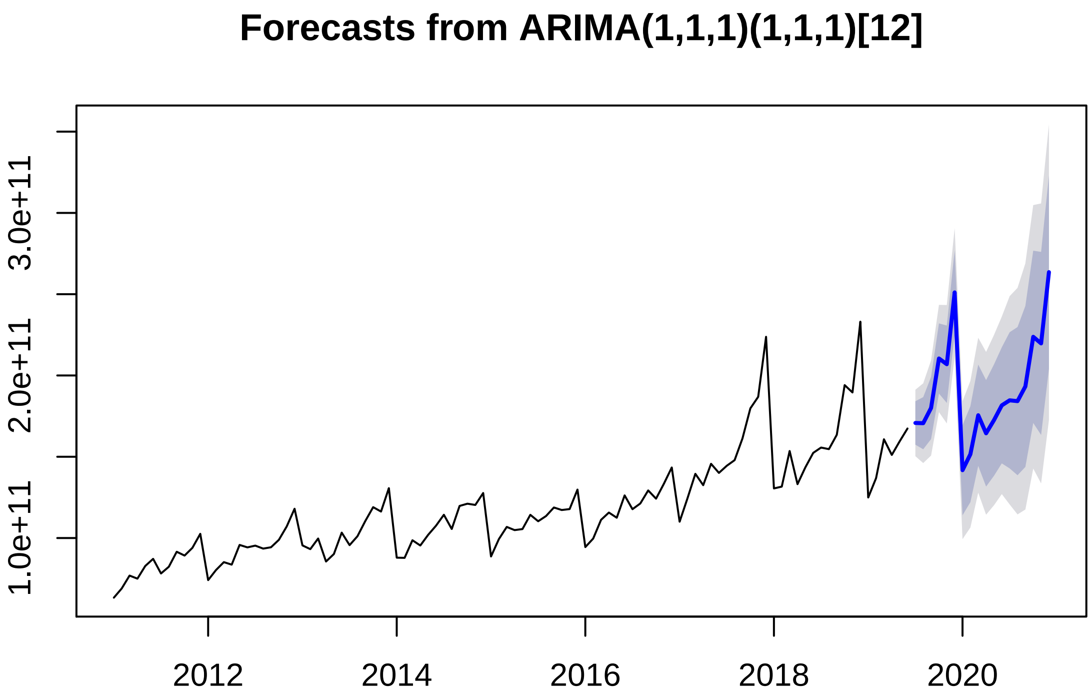
```
## Series: ts
## ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##           ar1     ma1     sar1    sma1
##       -0.3595  0.0227  -0.5334  0.4228
## s.e.   0.3126  0.3363   1.1785  1.2365
```

```
##
## sigma^2 estimated as 1.087e+20:  log likelihood=-2177.43
## AIC=4364.86   AICc=4365.58   BIC=4377.3
```

```r
ts_arima_forecast = forecast(ts_arima,h = 18)
ts_arima_forecast
```

```
##          Point Forecast        Lo 80        Hi 80        Lo 95
## Jul 2019   170788319965 157428067448 184148572483 150355576982
## Aug 2019   170645005267 154613295424 186676715109 146126620875
## Sep 2019   180021974811 160869582624 199174366999 150730918630
## Oct 2019   210457121225 188898689473 232015552977 177486345071
## Nov 2019   206924998308 183115365432 230734631185 170511307141
## Dec 2019   251110539040 225274163882 276946914198 211597213035
## Jan 2020   141729506303 114004417826 169454594780  99327642544
## Feb 2020   151566190295 122076402987 181055977603 106465452572
## Mar 2020   175554800329 144398986497 206710614162 127906094914
## Apr 2020   164410562793 131673783887 197147341700 114343979948
## May 2020   172509839910 138264873602 206754806217 120136683480
## Jun 2020   181614978374 145925547412 217304409335 127032703892
## Jul 2020   184727206520 142836852777 226617560262 120661439022
## Aug 2020   184216559042 138649586711 229783531373 114527888641
## Sep 2020   193265167425 143752001790 242778333059 117541315171
## Oct 2020   223788192073 170807130446 276769253701 142760650580
## Nov 2020   219721294255 163423195686 276019392823 133620782532
## Dec 2020   263567611400 204158698957 322976523843 172709520411
##                 Hi 95
## Jul 2019 191221062948
## Aug 2019 195163389659
## Sep 2019 209313030992
## Oct 2019 243427897379
## Nov 2019 243338689476
## Dec 2019 290623865044
## Jan 2020 184131370062
## Feb 2020 196666928018
## Mar 2020 223203505744
## Apr 2020 214477145638
## May 2020 224882996339
## Jun 2020 236197252855
## Jul 2020 248792974017
## Aug 2020 253905229443
## Sep 2020 268989019678
## Oct 2020 304815733566
## Nov 2020 305821805978
## Dec 2020 354425702389
```

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



Growth

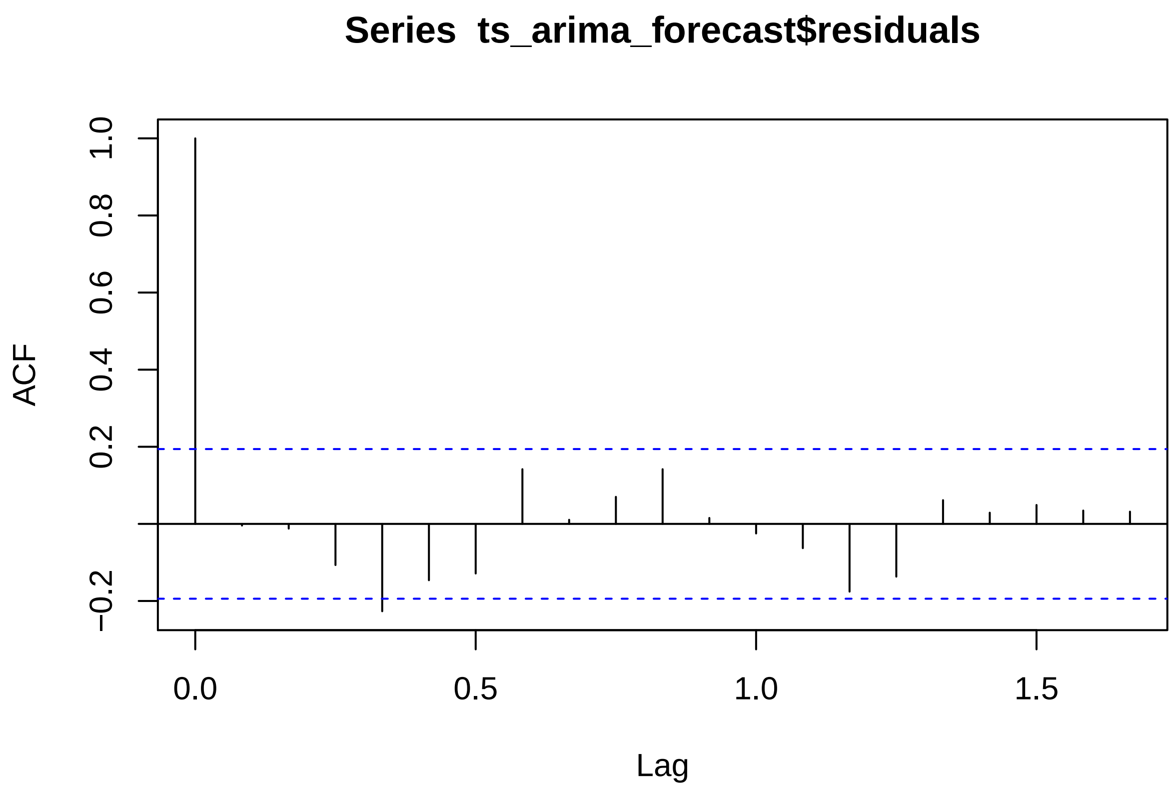
```
this_year_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1]

# growth_ARIMA <- growth(sum(c(this_year,as.numeric(this_year_predict_ARIMA$`Point Forecast`))), sum(la
# growth_ARIMA


year_2019_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1][c(1:6),]
sum_year_2019 = sum(c(year_2019,year_2019_predict_ARIMA))
year_2020 = (as.data.frame(ts_arima_forecast))[1][c(7:18),]
growth_ARIMA <- growth(sum_year_2019, sum(year_2020))
-growth_ARIMA
```

```
## [1] 0.0736588
```

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)
```

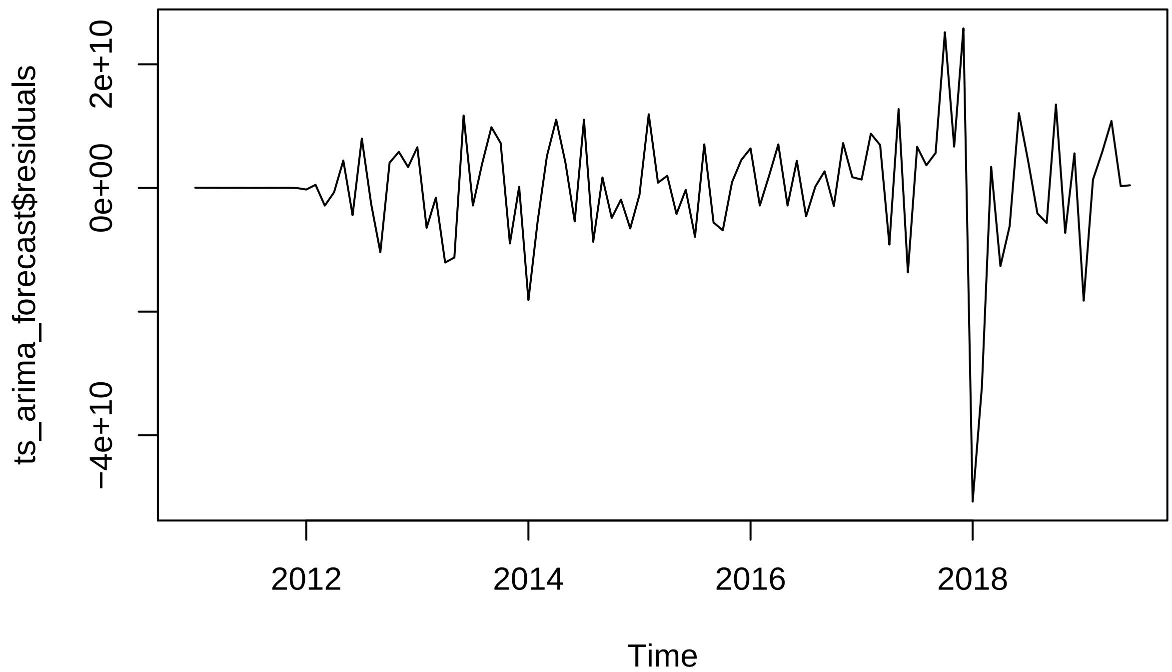



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

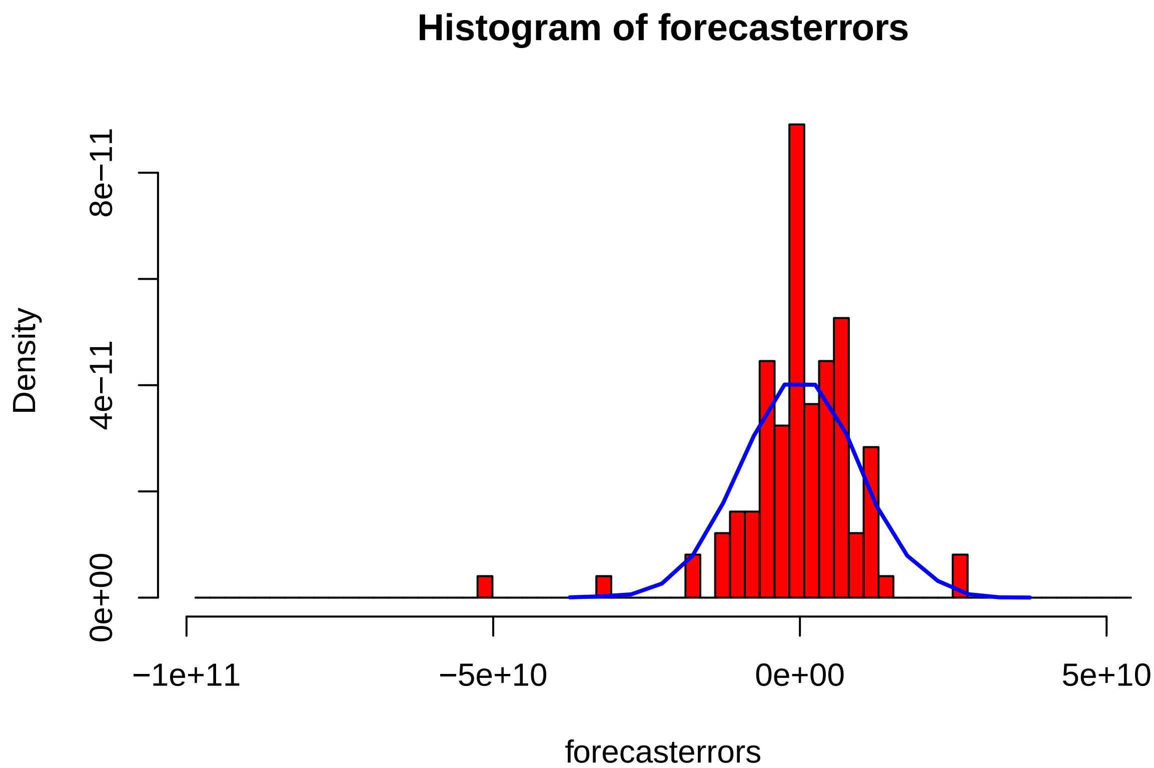
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 23.833, df = 20, p-value = 0.2498
```

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
```




```
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
```

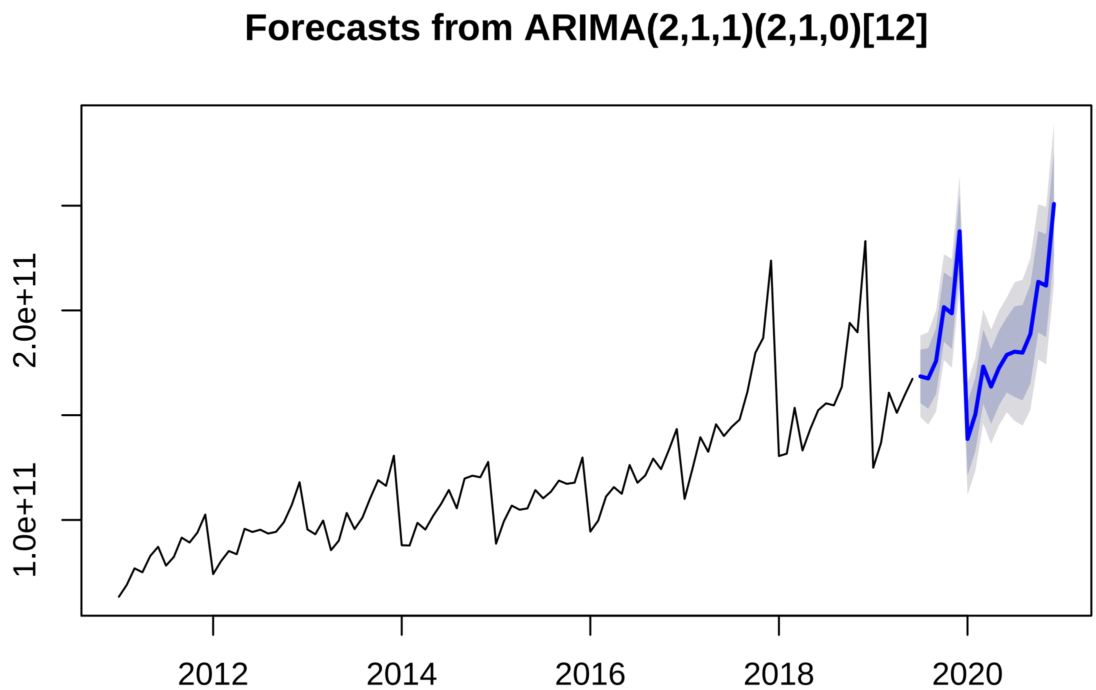
```
## Series: ts
## ARIMA(2,1,1)(2,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1     sar1     sar2
##       0.5222  0.1982  -0.9857  -0.1363  -0.1190
## s.e.  0.1135  0.1114   0.0861   0.1112   0.1648
##
## sigma^2 estimated as 9.836e+19:  log likelihood=-2173.54
## AIC=4359.08   AICc=4360.11   BIC=4374.01
```

```
ts_arima_forecast = forecast(ts_arima,h = 18)
ts_arima_forecast
```

```
##          Point Forecast        Lo 80        Hi 80        Lo 95
## Jul 2019   168553527898 155828774248 181278281548 149092696561
## Aug 2019   167531660835 153077999389 181985322282 145426693199
## Sep 2019   175847002176 160075741268 191618263083 151726940057
## Oct 2019   201591574585 185088058560 218095090611 176351624904
## Nov 2019   198603614313 181623400634 215583827992 172634618484
## Dec 2019   237810566298 220521444450 255099688146 211369136193
```

```
## Jan 2020   138627062656 121129531745 156124593567 111866898395
## Feb 2020   150504321916 132862690003 168145953828 123523774190
## Mar 2020   173220725922 155476469987 190964981857 146083228298
## Apr 2020   163632514180 145813038373 181451989987 136379977689
## May 2020   172515909415 154639634745 190392184085 145176506559
## Jun 2020   178798587286 160878126902 196719047669 151391608213
## Jul 2020   180344264986 158682056138 202006473835 147214775460
## Aug 2020   179825794236 157062135945 202589452527 145011783153
## Sep 2020   188730868045 165072010340 212389725751 152547767603
## Oct 2020   213672268657 189463832448 237880704866 176648660426
## Nov 2020   211845497910 187253149128 236437846692 174234746065
## Dec 2020   250844476239 225983538992 275705413486 212822953784
##                 Hi 95
## Jul 2019 188014359235
## Aug 2019 189636628471
## Sep 2019 199967064295
## Oct 2019 226831524266
## Nov 2019 224572610142
## Dec 2019 264251996403
## Jan 2020 165387226917
## Feb 2020 177484869641
## Mar 2020 200358223546
## Apr 2020 190885050671
## May 2020 199855312271
## Jun 2020 206205566359
## Jul 2020 213473754513
## Aug 2020 214639805319
## Sep 2020 224913968488
## Oct 2020 250695876889
## Nov 2020 249456249755
## Dec 2020 288865998693
```

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



Growth

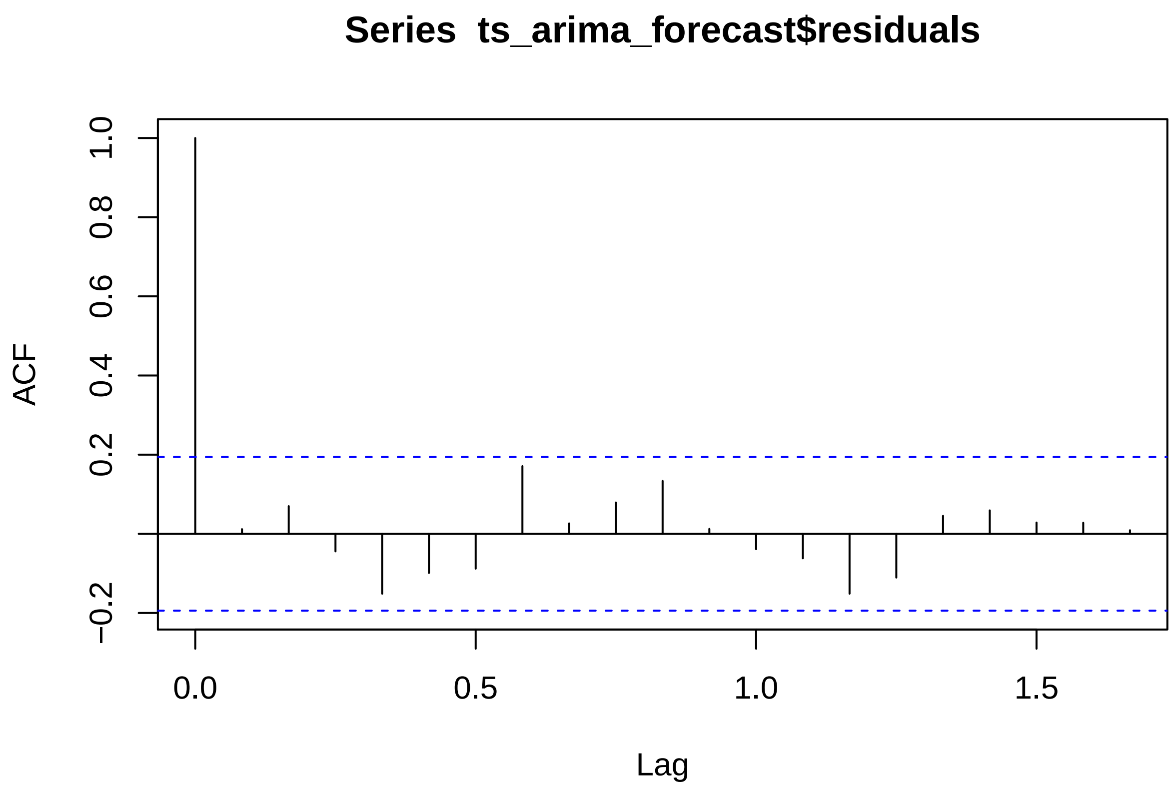
```
this_year_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1]

year_2019_predict_ARIMA <- (as.data.frame(ts_arima_forecast))[1][c(1:6),]
sum_year_2019 = sum(c(year_2019,year_2019_predict_ARIMA))
year_2020 = (as.data.frame(ts_arima_forecast))[1][c(7:18),]
growth_ARIMA2 <- growth(sum_year_2019, sum(year_2020))
-growth_ARIMA2
```

```
## [1] 0.0690669
```

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)
```

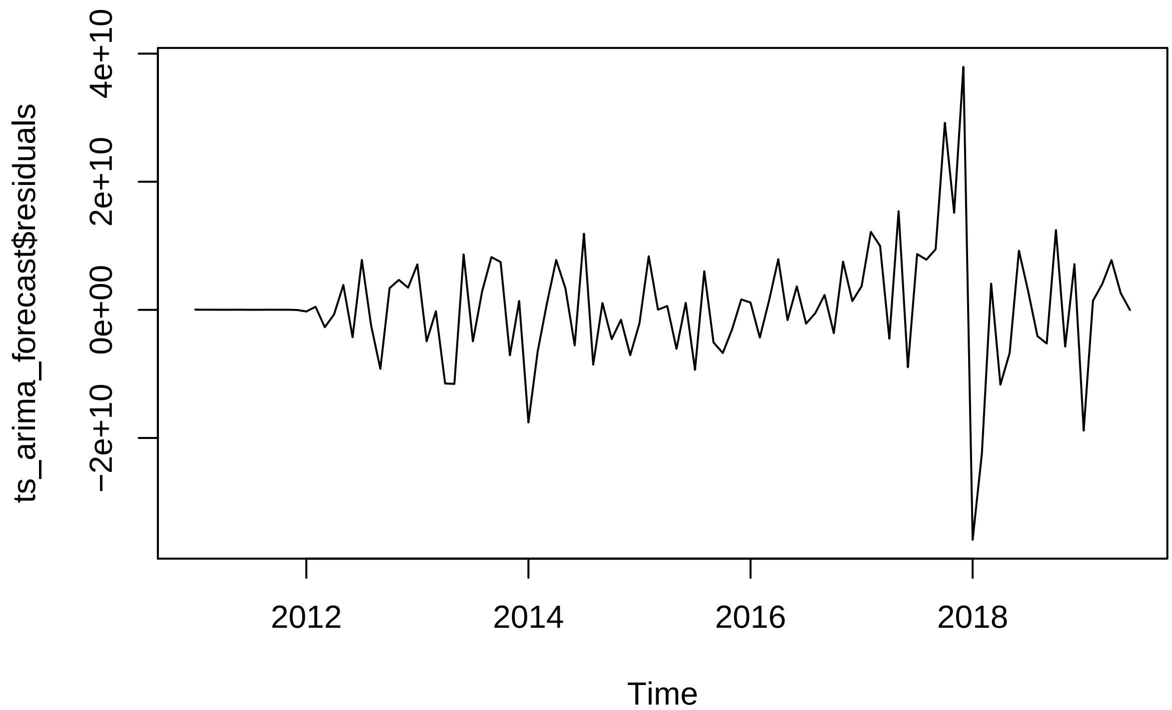



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

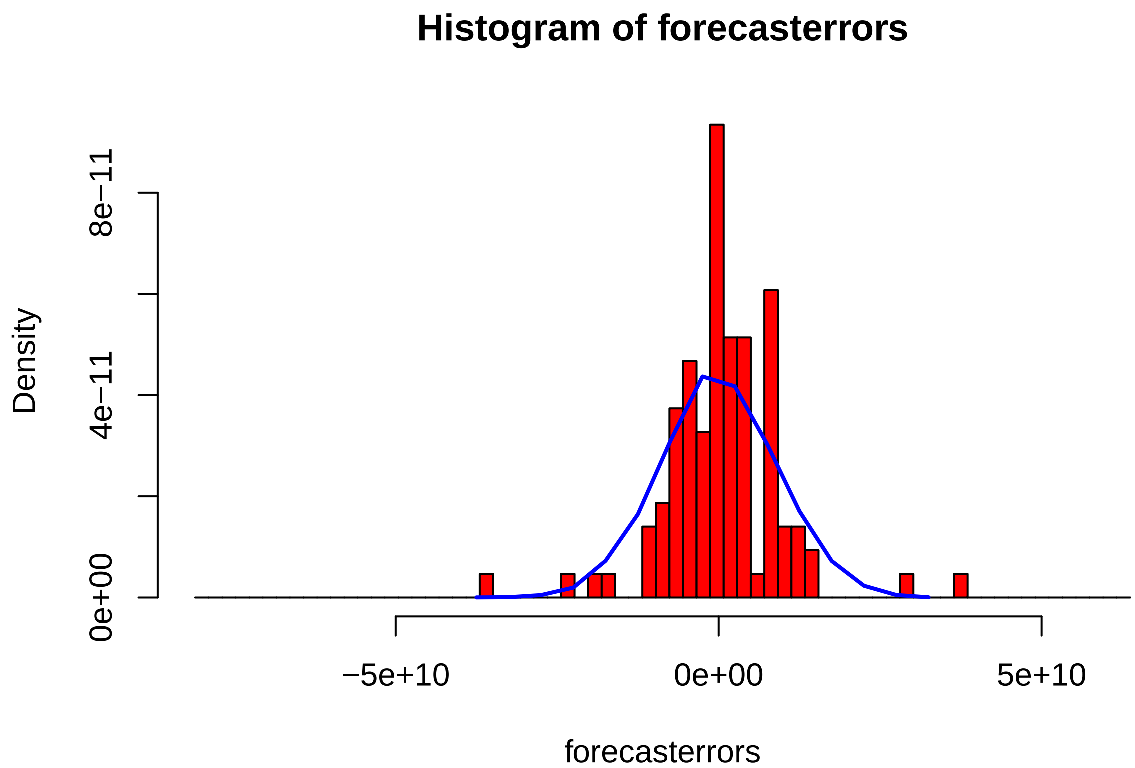
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 17.07, df = 20, p-value = 0.6484
```

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)
```

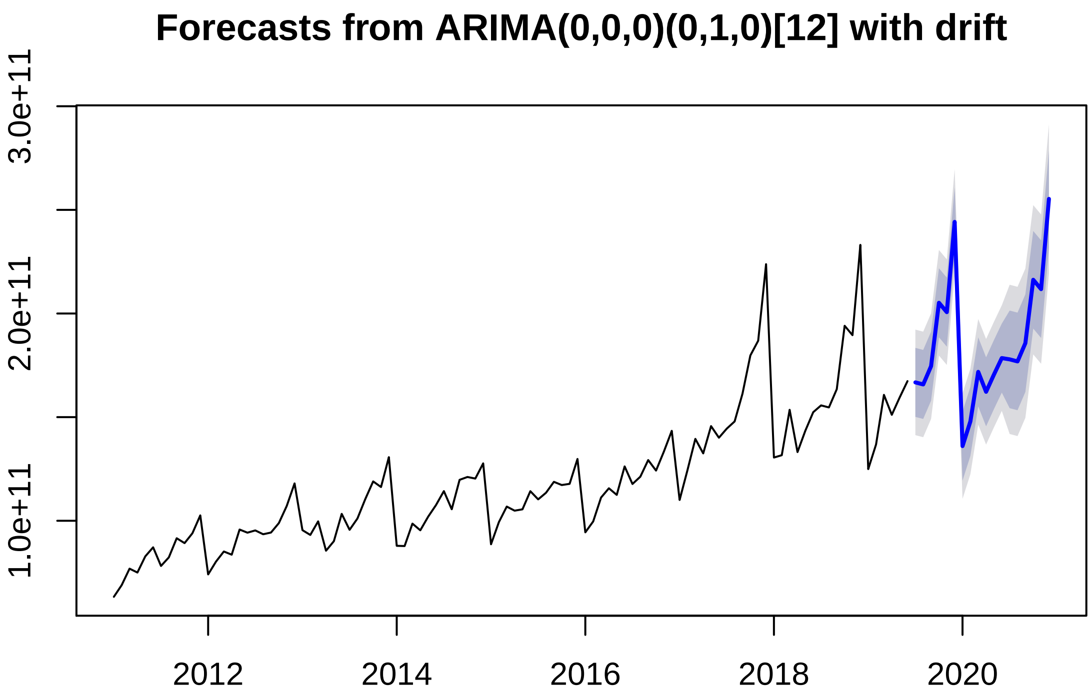


A model chosen automatically

```
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(0,0,0)(0,1,0)[12] with drift
##
## Coefficients:
##           drift
##       924981389
## s.e.   84479299
##
## sigma^2 estimated as 1.687e+20:  log likelihood=-2223.06
## AIC=4450.13   AICc=4450.27   BIC=4455.13
```

```
fit_forecast = forecast(fit,h=18)
plot(fit_forecast)
```




```
# str(fit)
```

Growth

```
year_2019_predict_auto.arima <- (as.data.frame(fit_forecast))[1][c(1:6),]
year_2019_predict_auto.arima_95_low <- (as.data.frame(fit_forecast))[4][c(1:6),]
year_2019_predict_auto.arima_95_high <- (as.data.frame(fit_forecast))[5][c(1:6),]

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(7:18),]
year_2020_predict_auto.arima_95_low <- (as.data.frame(fit_forecast))[4][c(7:18),]
year_2020_predict_auto.arima_95_high <- (as.data.frame(fit_forecast))[5][c(7:18),]

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)

growth_auto.arima
```

```
## [1] 0.06473158
```

```
growth_auto.arima_95_low
```

```
## [1] -0.04347715
```

```
growth_auto.arima_95_high
```

```
## [1] 0.1579852
```

all the growths

```
growth_ARIMA
```

```
## [1] -0.0736588
```

```
growth_ARIMA2
```

```
## [1] -0.0690669
```

```
growth_auto.arima
```

```
## [1] 0.06473158
```

```
growth_HW
```

```
## [1] 0.0485728
```