Time Series Forcasting report for service

Kevork Sulahian

2021-04-26

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

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

```
df20 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2020')
```

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

```
df19 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2019')
```

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

```
df18 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2018')
```

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

```r
df17 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2017')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df16 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2016')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df15 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2015')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df14 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2014')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df13 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2013')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df12 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2012')
```

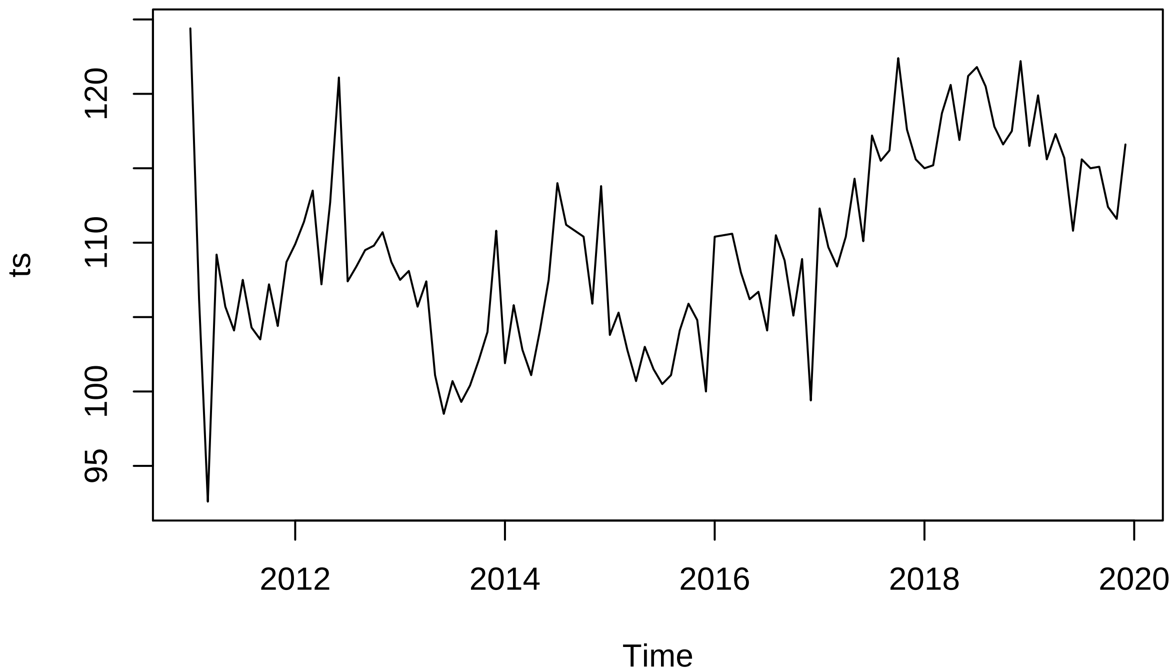
```
## New names:
## * `` -> ...2
## * `` -> ...3
```

```r
df11 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2011')
```

```
## New names:
## * `` -> ...2
## * `` -> ...3
```

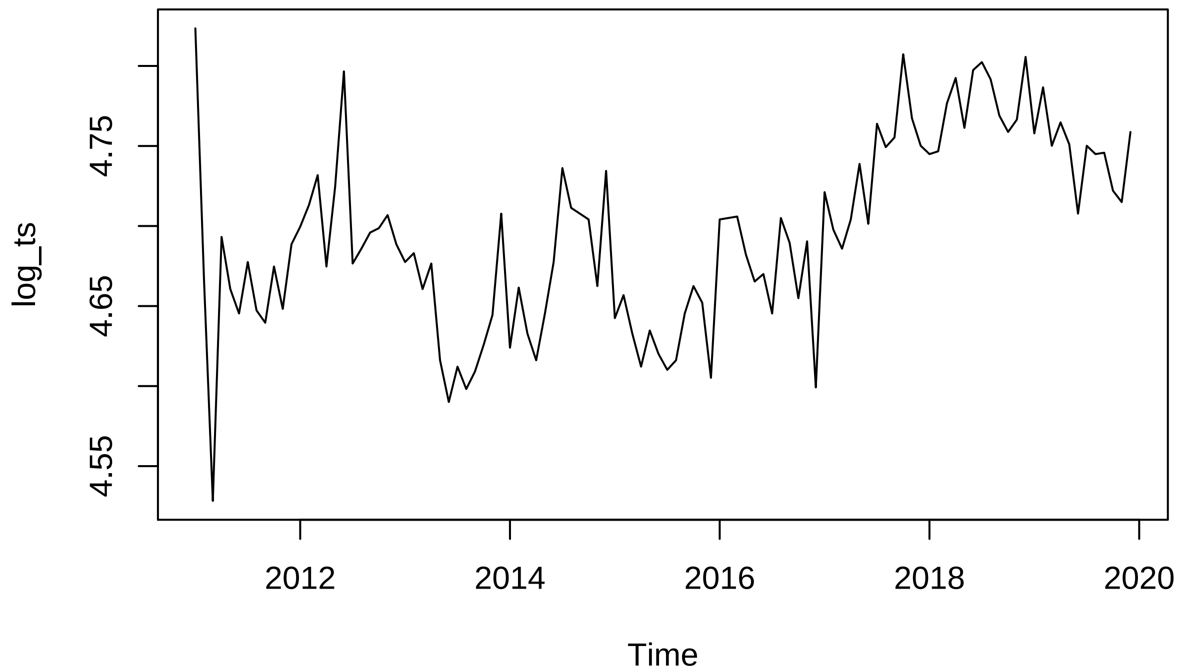
```r
# df10 <- read_xlsx('Servis-2000-2020.xlsx', sheet = '2010')

df20 <- df20[,13]
df19 <- df19[,13]
df18 <- df18[,13]
df17 <- df17[,3]
```

```r
df16 <- df16[,3]
df15 <- df15[,3]
df14 <- df14[,3]
df13 <- df13[,3]
df12 <- df12[,3]
df11 <- df11[,3]
# df10 <- df10[,3]

df20 = df20[-c(1:4),]
df19 = df19[-c(1:4),]
df19 = df19[-c(13:18),]
df18 = df18[-c(1:4),]
df17 = df17[-c(1:4),]
df16 = df16[-c(1:4),]
df15 = df15[-c(1:4),]
df14 = df14[-c(1:4),]
df13 = df13[-c(1:4),]
df12 = df12[-c(1:4),]
df11 = df11[-c(1:4),]

colnames(df20) = "data"
colnames(df19) = "data"
colnames(df18) = "data"
colnames(df17) = "data"
colnames(df16) = "data"
colnames(df15) = "data"
colnames(df14) = "data"
colnames(df13) = "data"
colnames(df12) = "data"
colnames(df11) = "data"



df = rbind(df11,df12,df13,df14,df15,df16,df17,df18,df19)
df = as.numeric(df$data)
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.

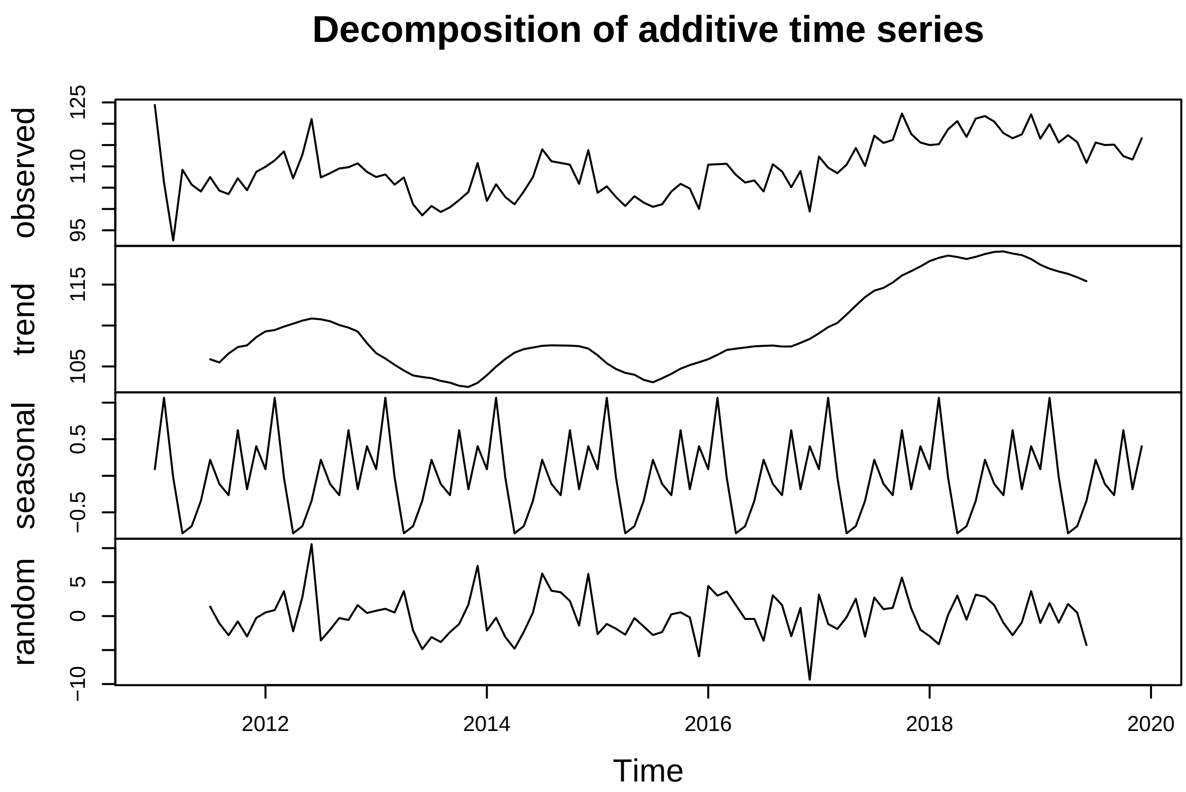
```
ts_components <- decompose(ts)
```

we can print out the estimated values of the seasonal component

```
ts_components$seasonal
```

```
##              Jan         Feb         Mar         Apr         May         Jun
## 2011  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2012  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2013  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2014  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2015  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2016  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2017  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2018  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
## 2019  0.08971354  1.06679687 -0.02434896 -0.78684896 -0.68893229 -0.34257812
##              Jul         Aug         Sep         Oct         Nov         Dec
```

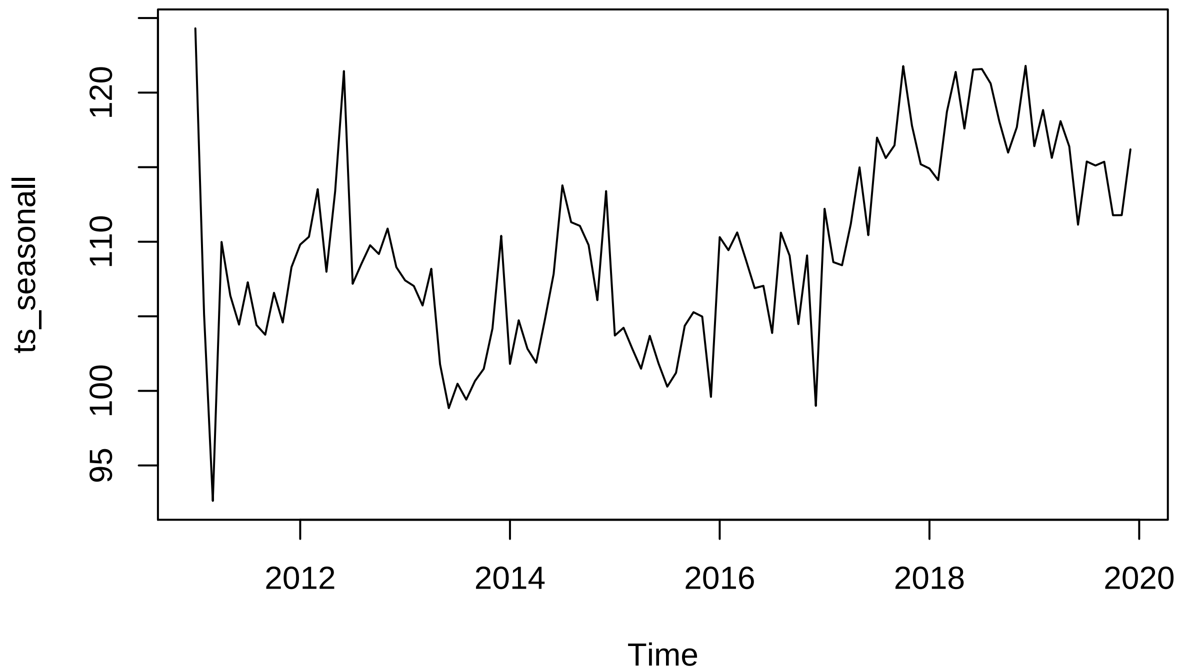
```
## 2011  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2012  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2013  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2014  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2015  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2016  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2017  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2018  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
## 2019  0.21888021 -0.11132812 -0.26497396  0.62304688 -0.18372396  0.40429687
```



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: 0.4464598
##  beta : 0.01143828
##  gamma: 0.5099043
##
## Coefficients:
##              [,1]
## a   116.1324529
## b     0.2405613
## s1   -2.2673524
## s2   -0.8330147
## s3   -1.8397982
## s4   -1.3729214
## s5   -2.4329696
## s6   -2.5330655
## s7    0.7304692
```

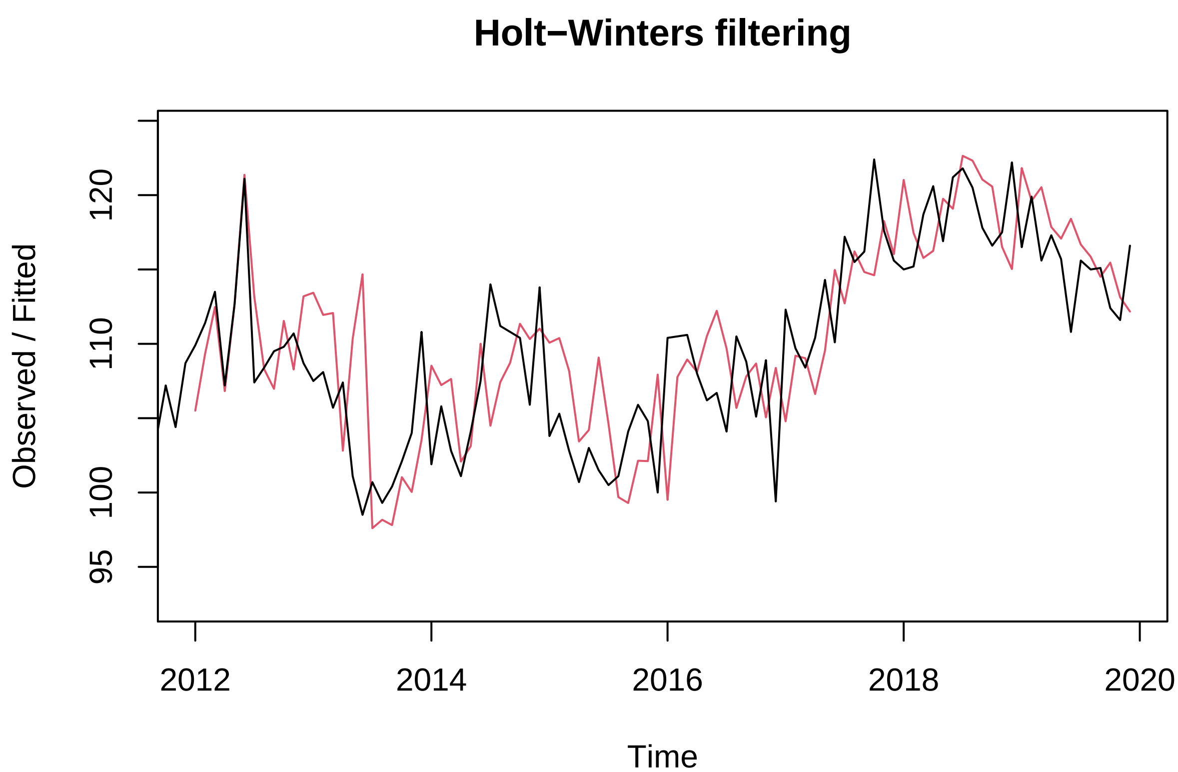
```
## s8    0.2157524
## s9   -0.5806335
## s10  -1.1650806
## s11  -1.9225320
## s12  -0.7335701
```

```
#
```

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

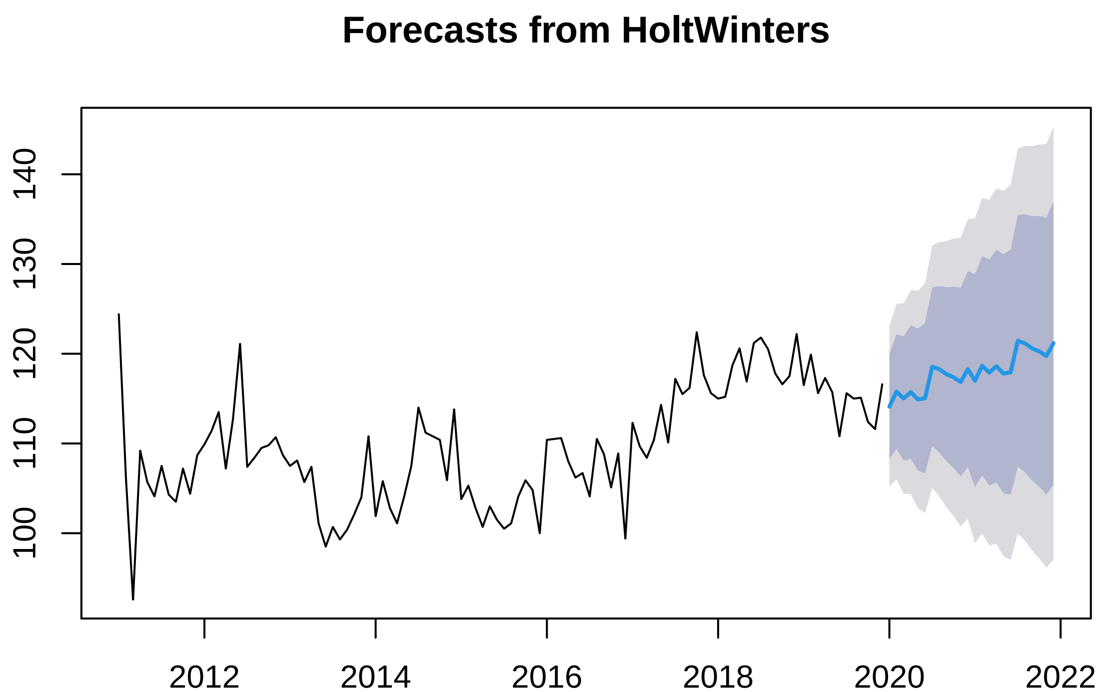
```
ts_forcaste$SSE
```

```
## [1] 1994.608
```




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

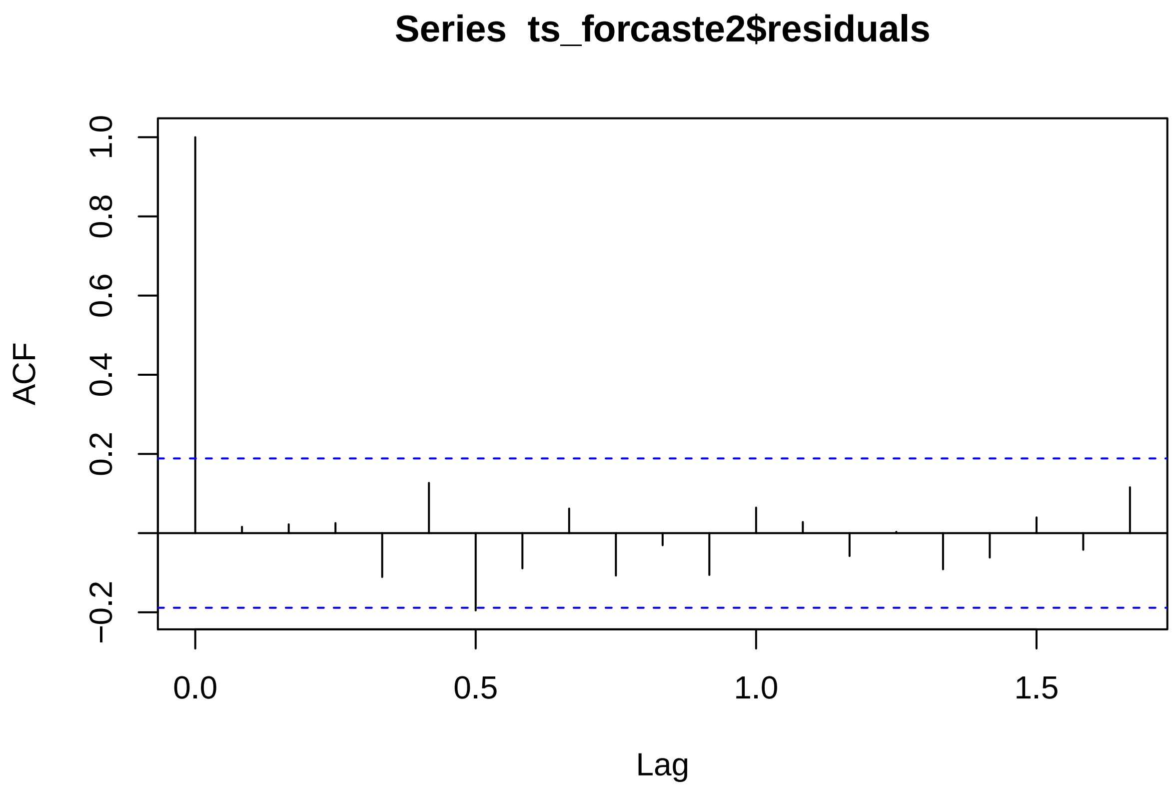
```
##          Point Forecast
## Jan 2020       114.1057
## Feb 2020       115.7806
## Mar 2020       115.0143
## Apr 2020       115.7218
## May 2020       114.9023
## Jun 2020       115.0428
## Jul 2020       118.5469
## Aug 2020       118.2727
## Sep 2020       117.7169
## Oct 2020       117.3730
## Nov 2020       116.8561
## Dec 2020       118.2856
## Jan 2021       116.9924
## Feb 2021       118.6673
## Mar 2021       117.9011
## Apr 2021       118.6085
## May 2021       117.7890
## Jun 2021       117.9295
## Jul 2021       121.4336
## Aug 2021       121.1594
## Sep 2021       120.6036
## Oct 2021       120.2597
## Nov 2021       119.7428
## Dec 2021       121.1724
```



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:

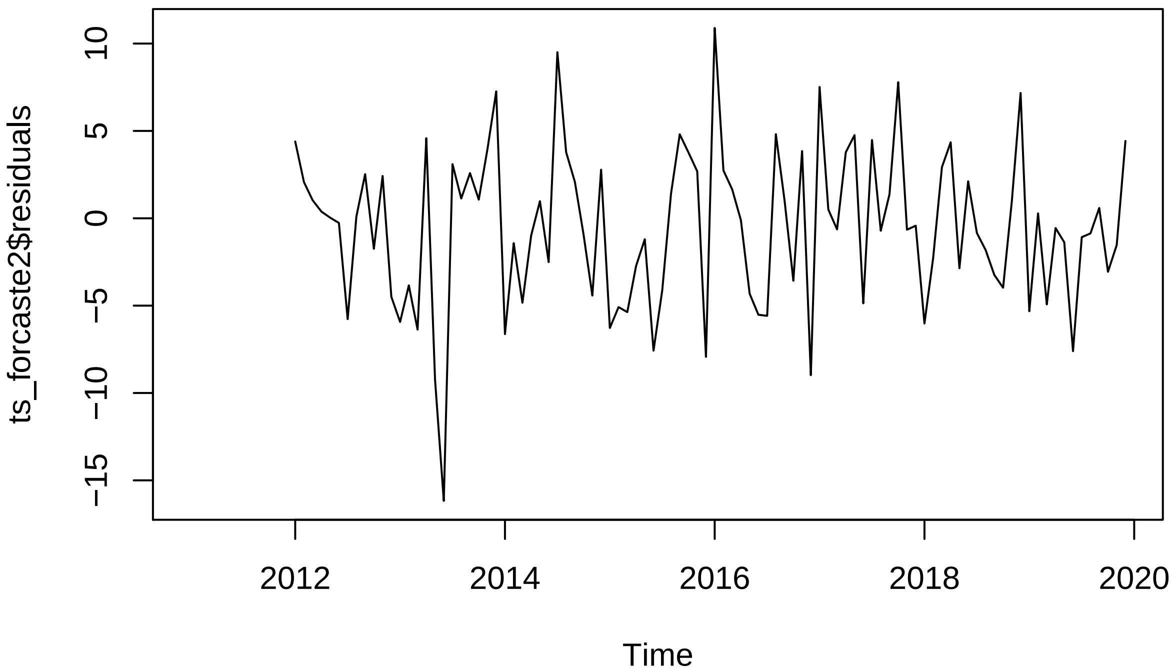



```
##
##  Box-Ljung test
##
## data:  ts_forcaste2$residuals
## X-squared = 15.324, df = 20, p-value = 0.7576
```

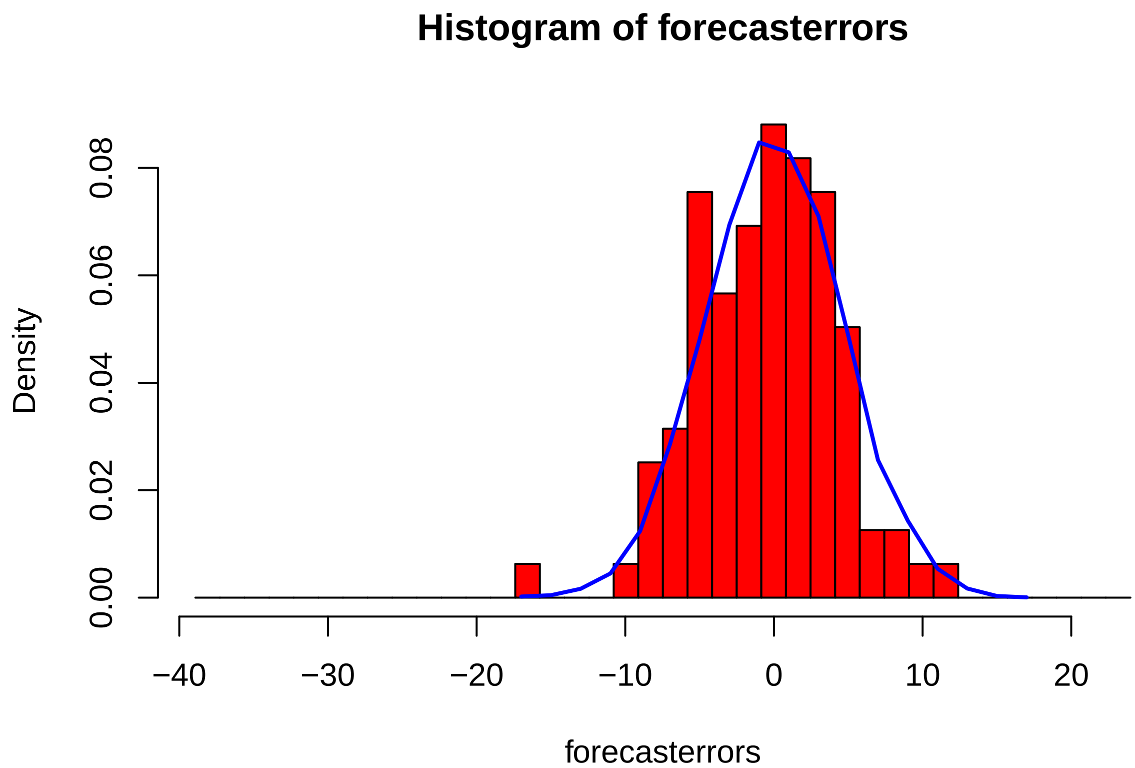
The correlogram shows that the autocorrelations for the in-sample forecast errors do not exceed the significance bounds for lags 1-20. Furthermore, the p-value for Ljung-Box test is 0.2, indicating that there is little evidence of non-zero autocorrelations at lags 1-20.

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)
```



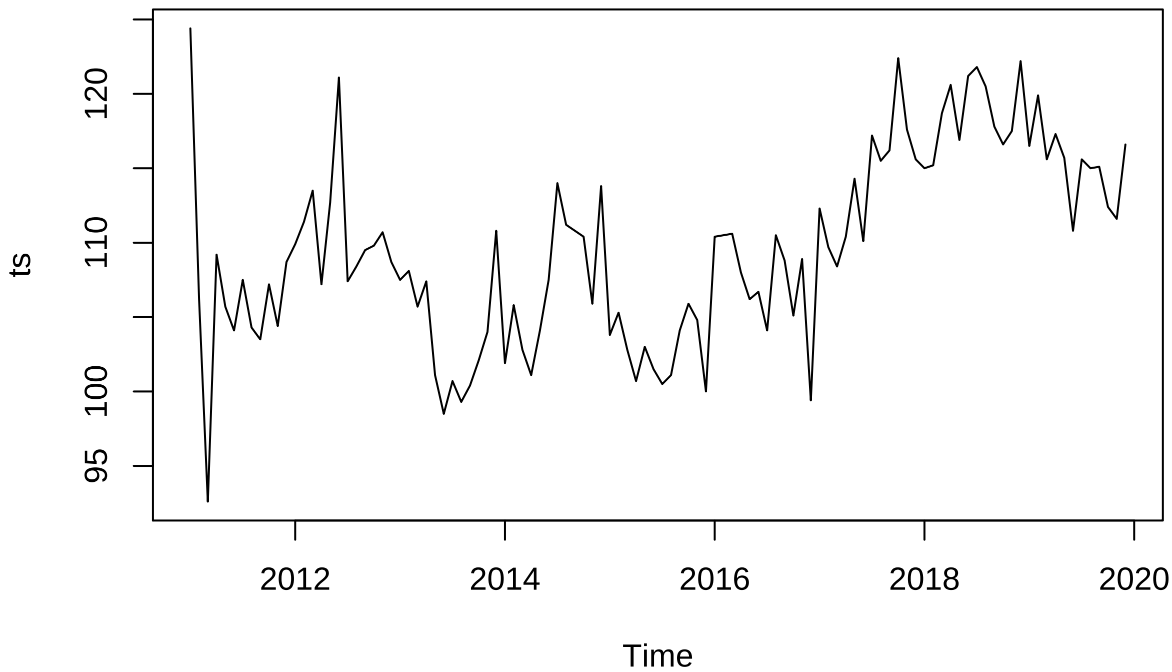

```r
plotForecastErrors(ts_forcaste2$residuals)
```



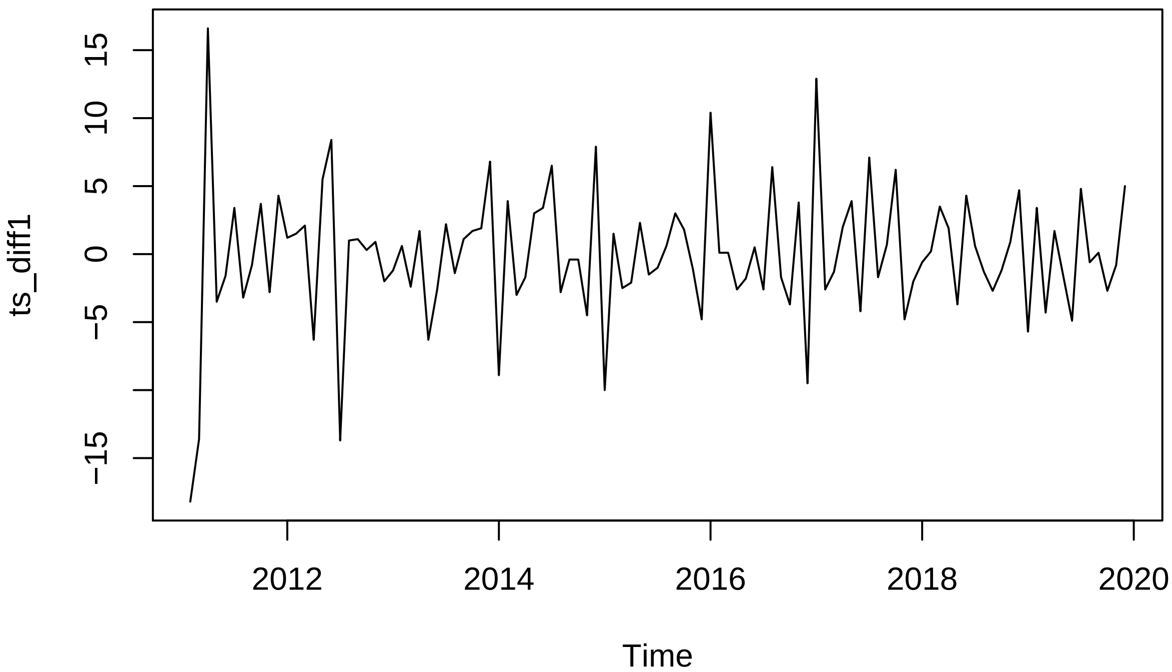
From the time plot, it appears plausible that the forecast errors have constant variance over time. From the histogram of forecast errors, it seems plausible that the forecast errors are normally distributed with mean zero.

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)
```

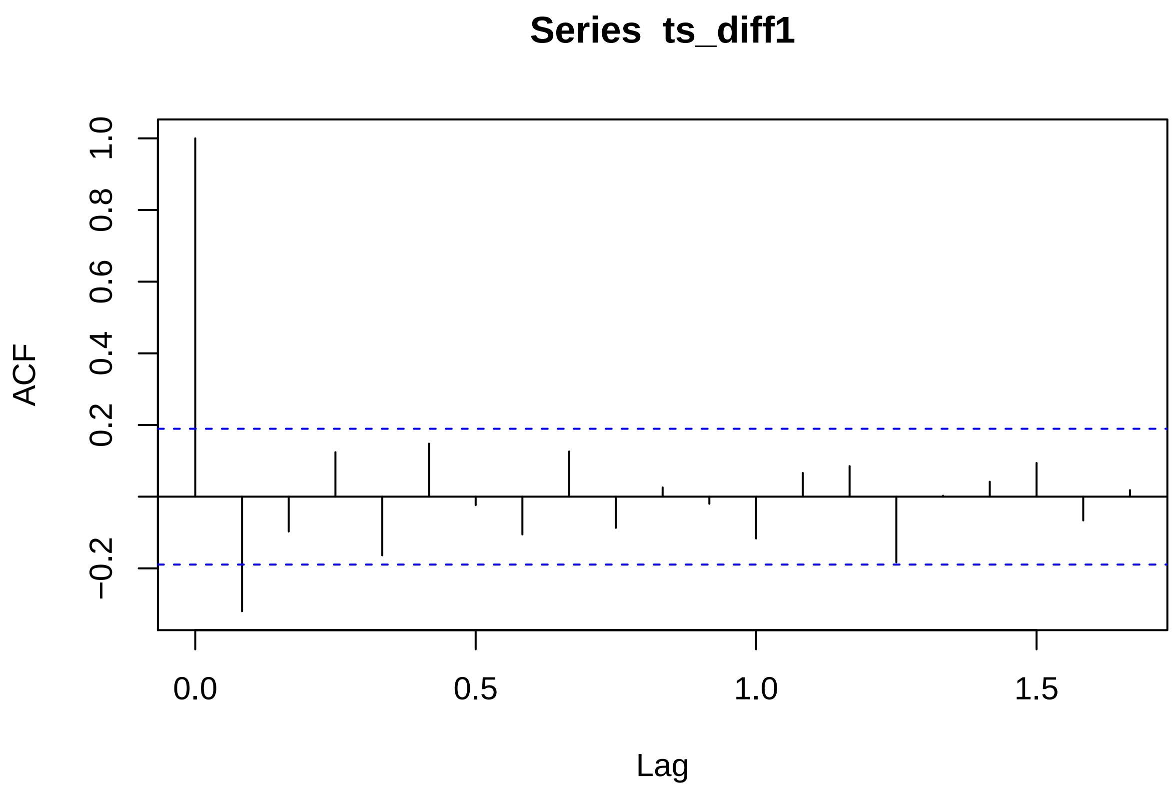



```
ts_diff1 <-  diff(ts, differences = 1)

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
```

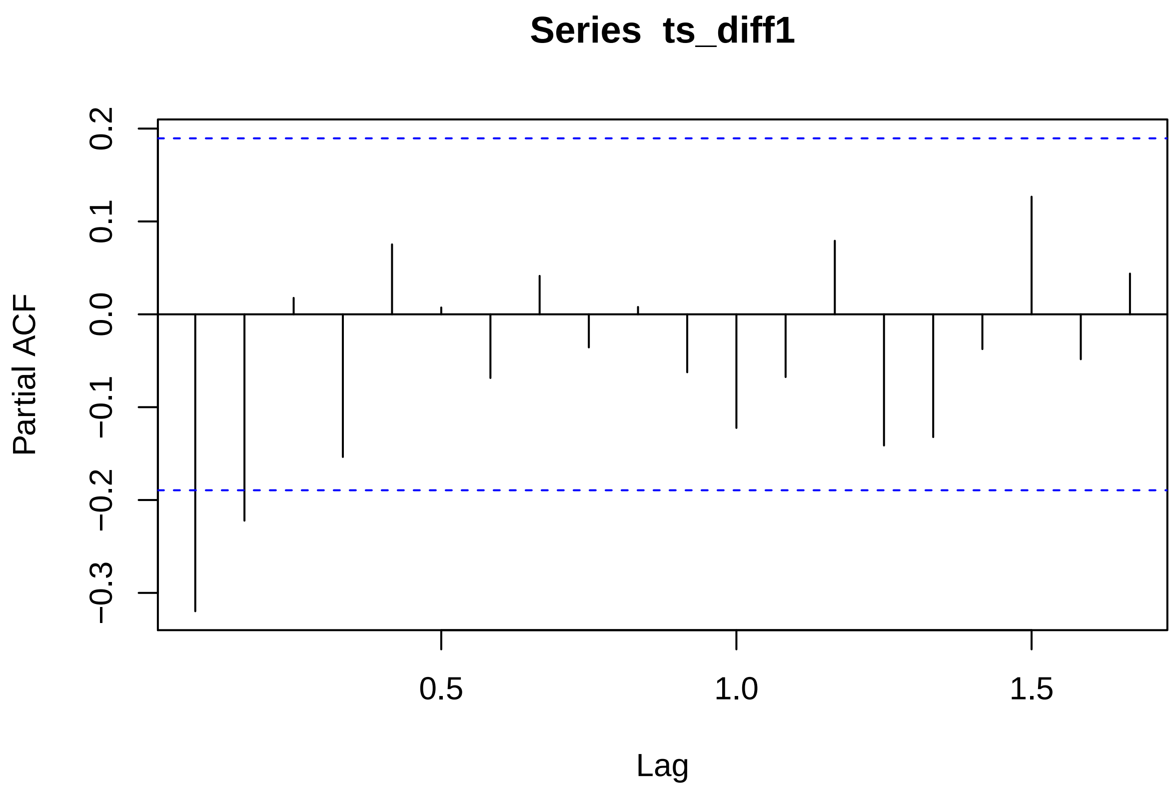


We see from the correlogram that the autocorrelation exceeds the significance bound 3 times but all the others do not exceed

```
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 -0.320 -0.097  0.124 -0.164  0.148 -0.024 -0.106  0.126 -0.087  0.026
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.020 -0.117  0.066  0.085 -0.183  0.003  0.042  0.094 -0.066  0.018
```

```
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
## -0.320 -0.222  0.018 -0.154  0.075  0.007 -0.069  0.041 -0.036  0.008 -0.062
## 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.122 -0.068  0.079 -0.141 -0.132 -0.037  0.127 -0.048  0.044
```

Arima, 1,1,1

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

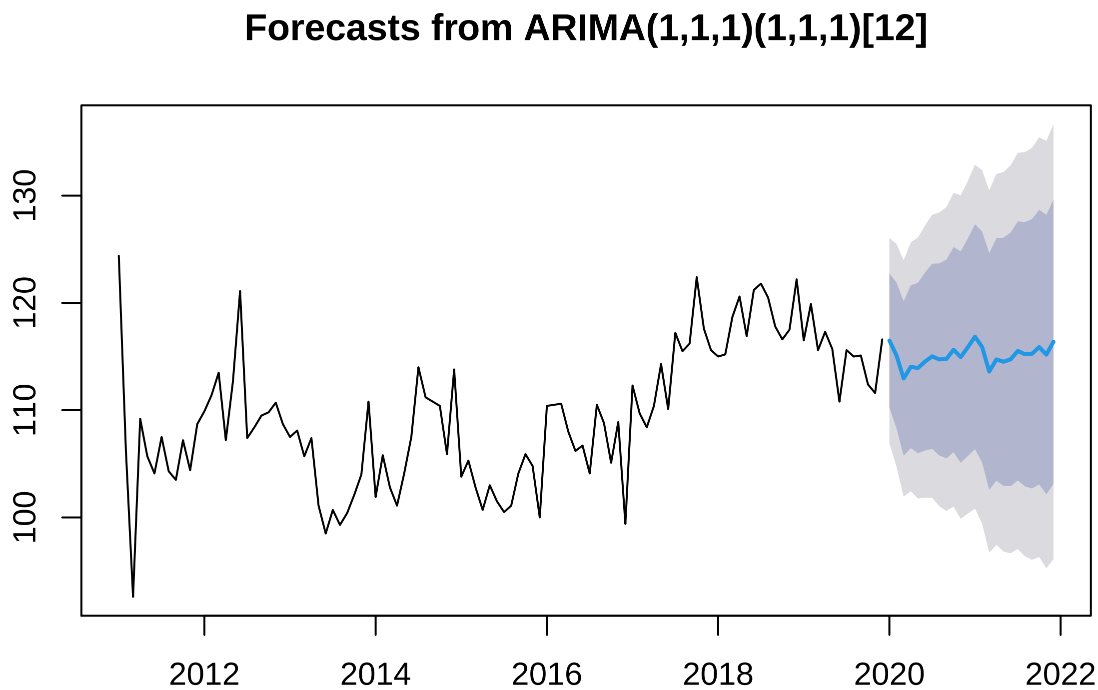
```
## Series: ts
## ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sar1     sma1
##       0.0689  -0.6445  -0.0687  -0.9998
## s.e.  0.1887   0.1423   0.1303   0.2939
```

```
##
## sigma^2 estimated as 21.24:  log likelihood=-292
## AIC=593.99   AICc=594.67   BIC=606.76
```

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

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       116.4954 110.2395 122.7514 106.92780 126.0631
## Feb 2020       115.1072 108.3156 121.8989 104.72034 125.4942
## Mar 2020       112.9556 105.7534 120.1577 101.94086 123.9703
## Apr 2020       114.0457 106.4606 121.6308 102.44534 125.6461
## May 2020       113.9301 105.9808 121.8794 101.77273 126.0875
## Jun 2020       114.5339 106.2365 122.8314 101.84405 127.2238
## Jul 2020       115.0201 106.3884 123.6517 101.81913 128.2210
## Aug 2020       114.7293 105.7760 123.6827 101.03641 128.4223
## Sep 2020       114.7761 105.5123 124.0400 100.60828 128.9440
## Oct 2020       115.6410 106.0766 125.2053 101.01359 130.2683
## Nov 2020       114.9466 105.0909 124.8023  99.87366 130.0195
## Dec 2020       115.8683 105.7290 126.0076 100.36154 131.3750
## Jan 2021       116.8502 106.3656 127.3348 100.81533 132.8851
## Feb 2021       115.8923 105.1245 126.6600  99.42444 132.3601
## Mar 2021       113.6000 102.5582 124.6418  96.71303 130.4870
## Apr 2021       114.7325 103.4235 126.0416  97.43681 132.0283
## May 2021       114.5150 102.9448 126.0851  96.81991 132.2100
## Jun 2021       114.7407 102.9152 126.5662  96.65511 132.8262
## Jul 2021       115.5232 103.4477 127.5986  97.05537 133.9910
## Aug 2021       115.2112 102.8909 127.5315  96.36890 134.0535
## Sep 2021       115.2616 102.7012 127.8221  96.05214 134.4711
## Oct 2021       115.8816 103.0855 128.6776  96.31173 135.4514
## Nov 2021       115.1799 102.1525 128.2074  95.25622 135.1037
## Dec 2021       116.3818 103.1261 129.6375  96.10898 136.6547
```

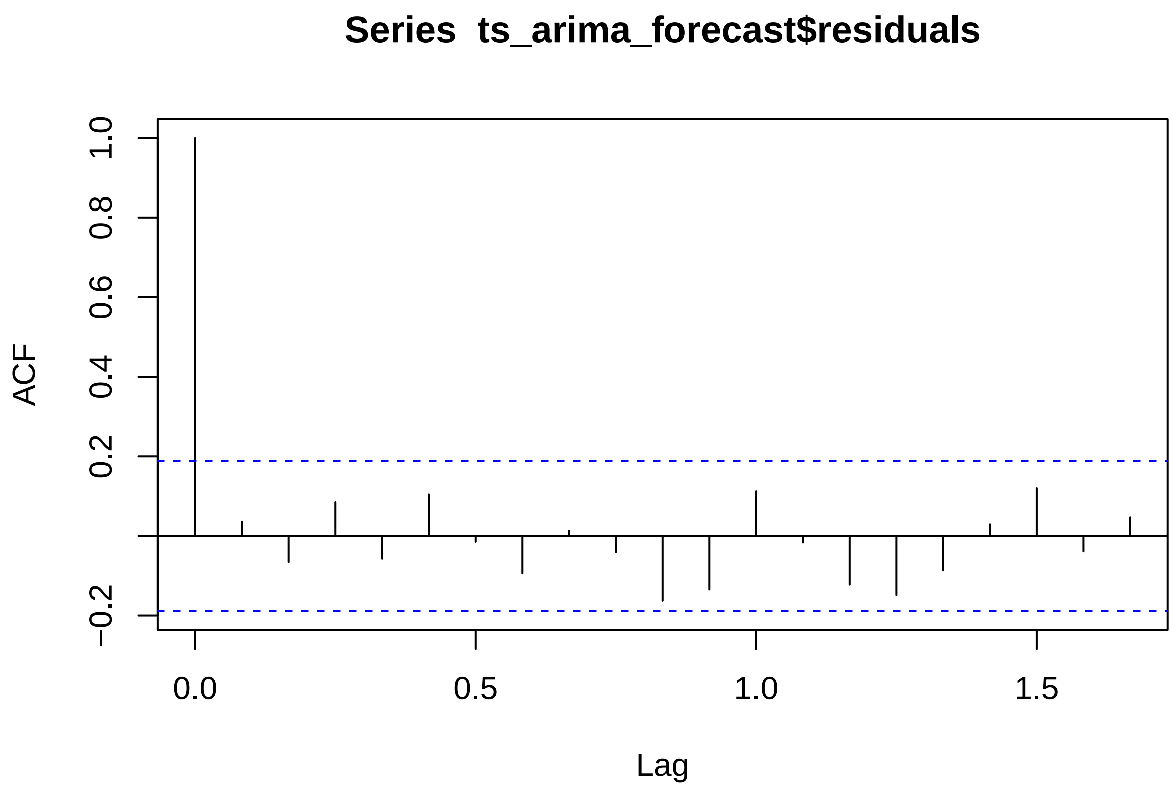
```r
forecast:::plot.forecast(ts_arima_forecast)
```



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)
```

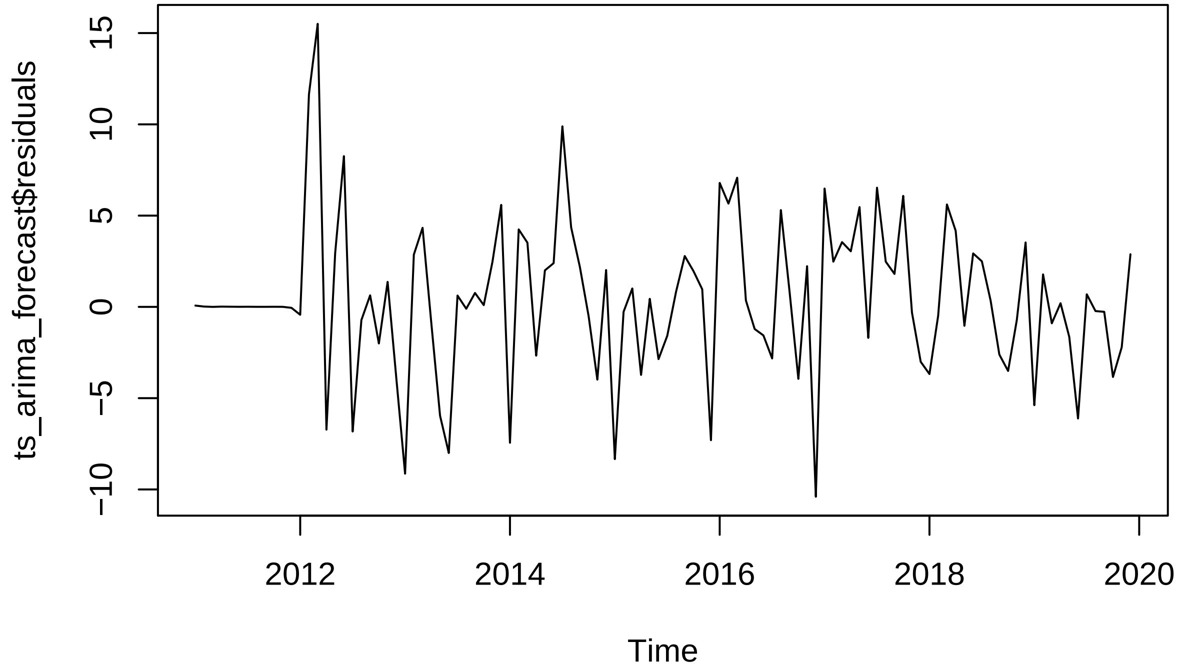



```
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

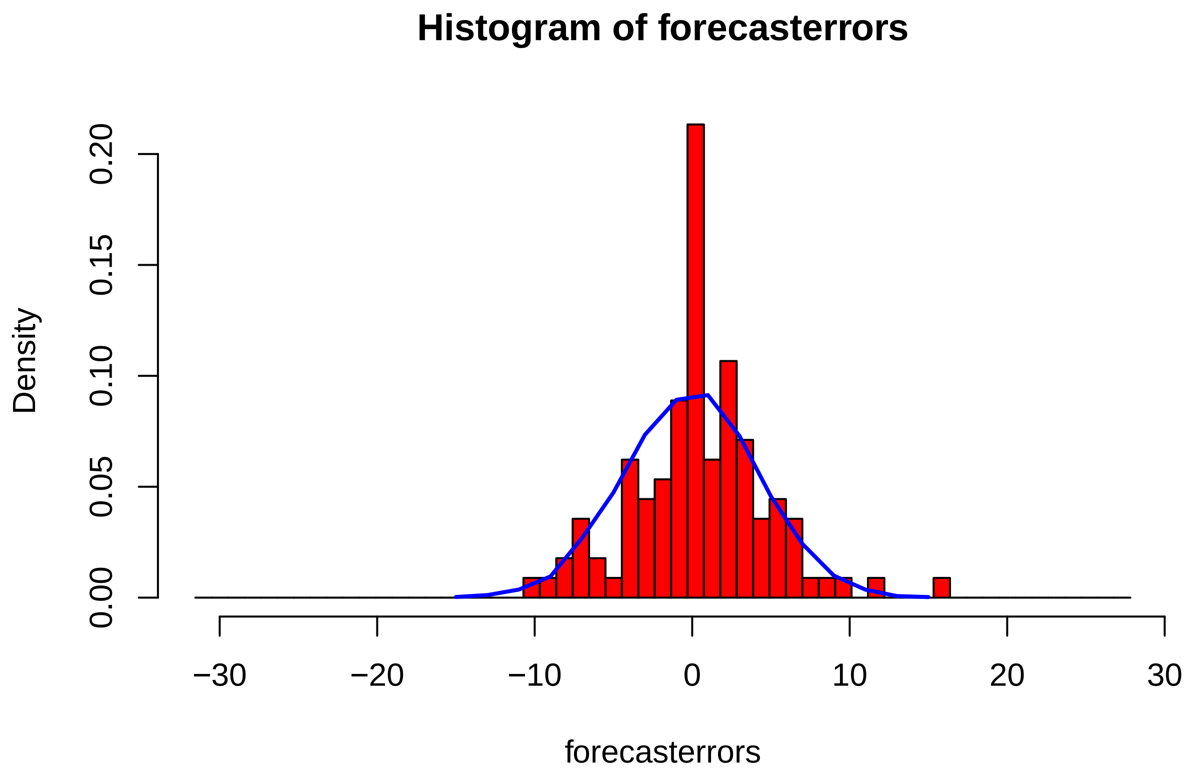
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 19.569, df = 20, p-value = 0.4852
```

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)
```

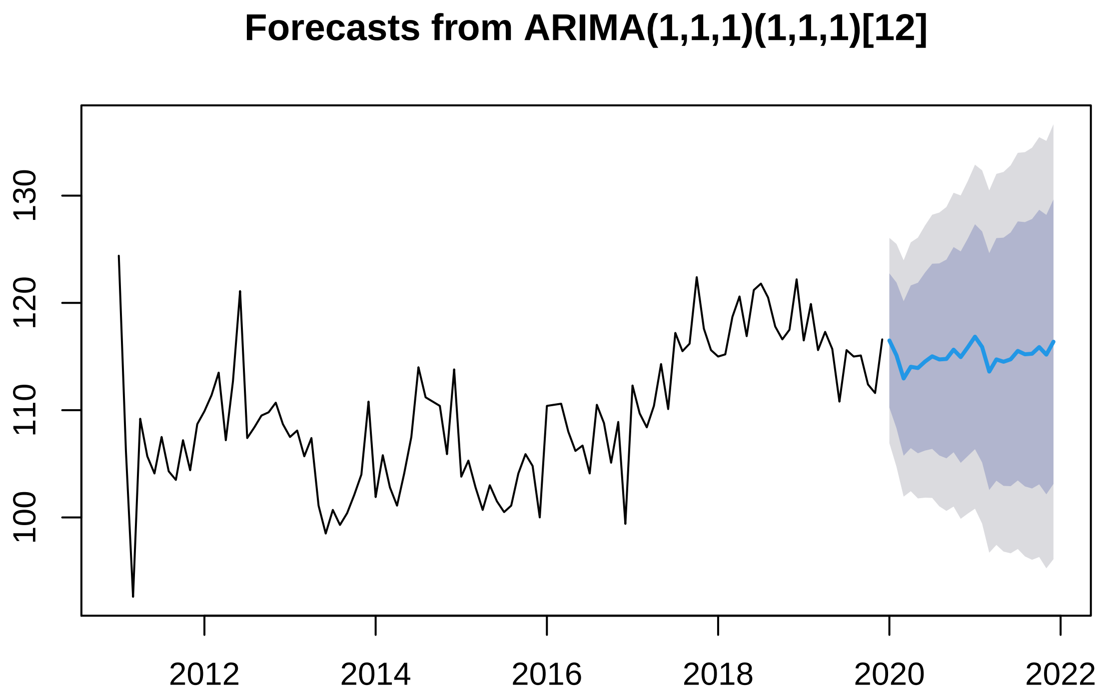


Arima, 0,1,0 as given from the loop

```
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
##       -0.7830  -0.4890  0.2163  -0.5424  -0.3254
## s.e.   0.3833   0.1673  0.4475   0.1220   0.1162
##
## sigma^2 estimated as 29.55:  log likelihood=-295.8
## AIC=603.61   AICc=604.56   BIC=618.93
```

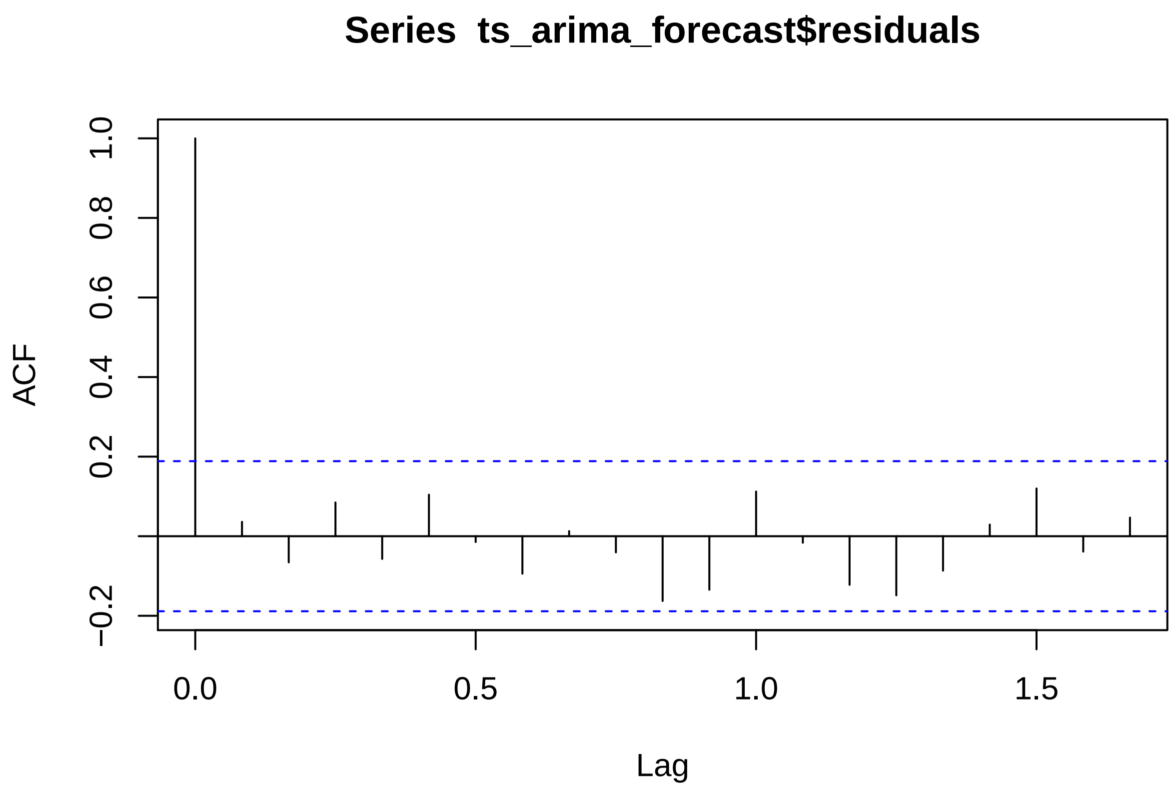
```
ts_arima_forecast2 = forecast(ts_arima,h = 24)
# ts_arima_forecast

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



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)
```

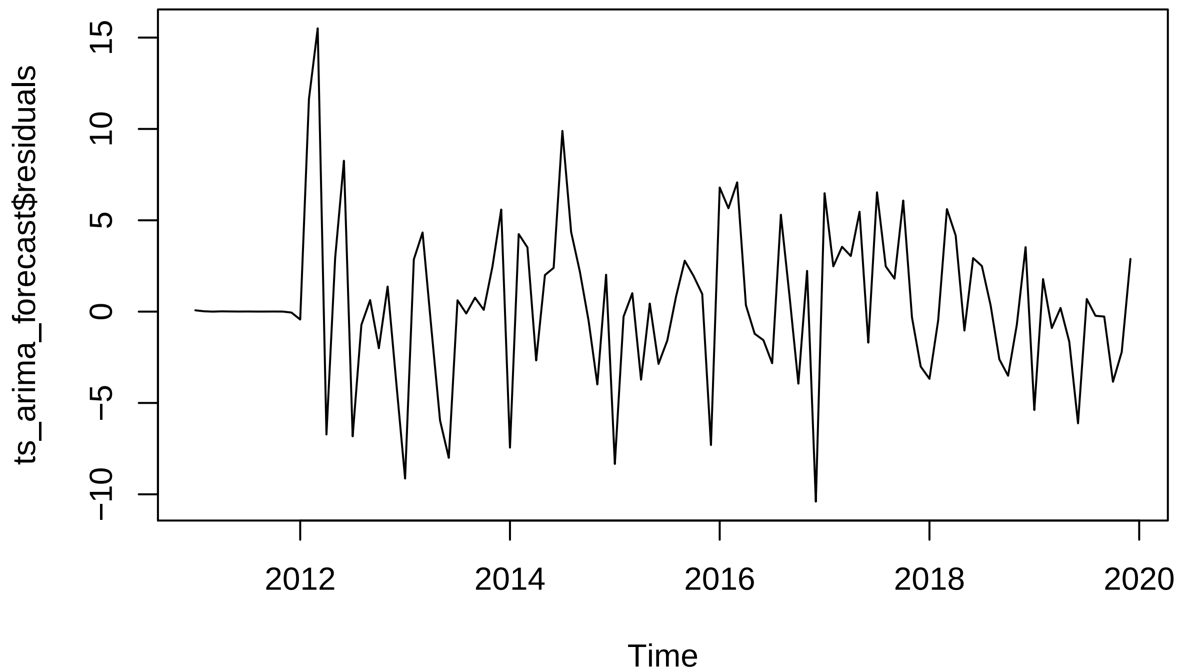



```r
Box.test(ts_arima_forecast$residuals, lag=20, type="Ljung-Box")
```

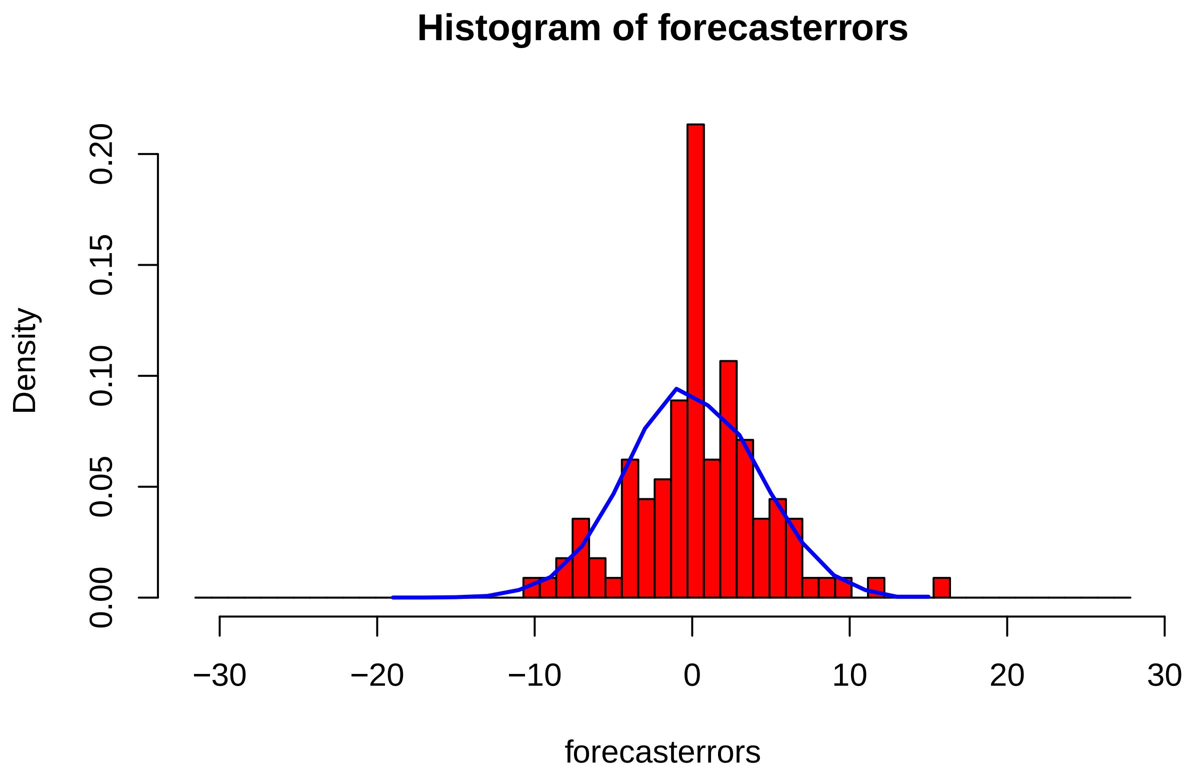
```
##
##  Box-Ljung test
##
## data:  ts_arima_forecast$residuals
## X-squared = 19.569, df = 20, p-value = 0.4852
```

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)
```

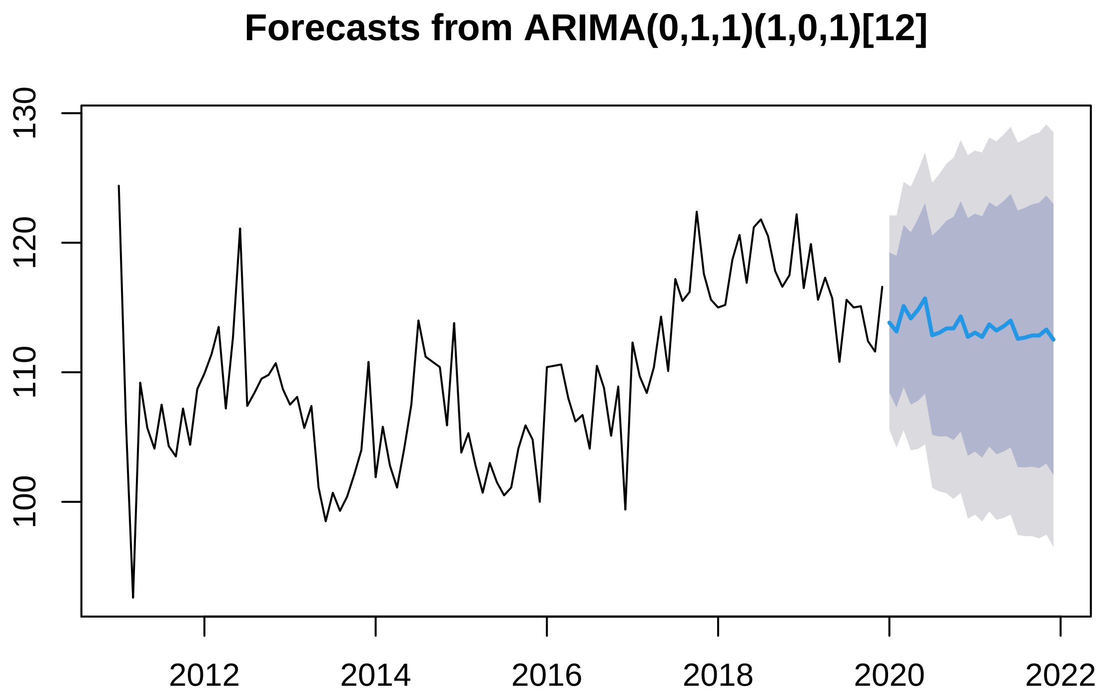


A model chosen automatically

```
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(0,1,1)(1,0,1)[12]
##
## Coefficients:
##           ma1    sar1     sma1
##       -0.5869  0.4943  -0.8211
## s.e.   0.0956  0.2134   0.2037
##
## sigma^2 estimated as 17.76:  log likelihood=-306.55
## AIC=621.1   AICc=621.49   BIC=631.79
```

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




```
# str(fit)
```

Growth

all the growths

```
hw_forecaste
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       114.1057 108.2754 119.9360 105.18900 123.0223
## Feb 2020       115.7806 109.3834 122.1777 105.99694 125.5642
## Mar 2020       115.0143 108.0852 121.9435 104.41715 125.6115
## Apr 2020       115.7218 108.2880 123.1556 104.35276 127.0908
## May 2020       114.9023 106.9858 122.8188 102.79503 127.0096
## Jun 2020       115.0428 106.6616 123.4239 102.22487 127.8606
## Jul 2020       118.5469 109.7161 127.3776 105.04145 132.0523
## Aug 2020       118.2727 109.0053 127.5401 104.09937 132.4460
## Sep 2020       117.7169 108.0237 127.4100 102.89243 132.5413
## Oct 2020       117.3730 107.2636 127.4824 101.91198 132.8340
## Nov 2020       116.8561 106.3388 127.3734 100.77122 132.9410
## Dec 2020       118.2856 107.3676 129.2036 101.58802 134.9832
## Jan 2021       116.9924 105.1427 128.8421  98.86978 135.1150
```

```
## Feb 2021       118.6673 106.4461 130.8885  99.97664 137.3579
## Mar 2021       117.9011 105.3123 130.4898  98.64827 137.1539
## Apr 2021       118.6085 105.6557 131.5614  98.79885 138.4182
## May 2021       117.7890 104.4752 131.1029  97.42724 138.1508
## Jun 2021       117.9295 104.2574 131.6016  97.01986 138.8391
## Jul 2021       121.4336 107.4058 135.4614  99.97997 142.8872
## Aug 2021       121.1594 106.7782 135.5406  99.16528 143.1536
## Sep 2021       120.6036 105.8710 135.3362  98.07206 143.1352
## Oct 2021       120.2597 105.1776 135.3418  97.19359 143.3258
## Nov 2021       119.7428 104.3128 135.1728  96.14466 143.3410
## Dec 2021       121.1724 105.3960 136.9488  97.04443 145.3003
```

```
ts_arima_forecast
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       116.4954 110.2395 122.7514 106.92780 126.0631
## Feb 2020       115.1072 108.3156 121.8989 104.72034 125.4942
## Mar 2020       112.9556 105.7534 120.1577 101.94086 123.9703
## Apr 2020       114.0457 106.4606 121.6308 102.44534 125.6461
## May 2020       113.9301 105.9808 121.8794 101.77273 126.0875
## Jun 2020       114.5339 106.2365 122.8314 101.84405 127.2238
## Jul 2020       115.0201 106.3884 123.6517 101.81913 128.2210
## Aug 2020       114.7293 105.7760 123.6827 101.03641 128.4223
## Sep 2020       114.7761 105.5123 124.0400 100.60828 128.9440
## Oct 2020       115.6410 106.0766 125.2053 101.01359 130.2683
## Nov 2020       114.9466 105.0909 124.8023  99.87366 130.0195
## Dec 2020       115.8683 105.7290 126.0076 100.36154 131.3750
## Jan 2021       116.8502 106.3656 127.3348 100.81533 132.8851
## Feb 2021       115.8923 105.1245 126.6600  99.42444 132.3601
## Mar 2021       113.6000 102.5582 124.6418  96.71303 130.4870
## Apr 2021       114.7325 103.4235 126.0416  97.43681 132.0283
## May 2021       114.5150 102.9448 126.0851  96.81991 132.2100
## Jun 2021       114.7407 102.9152 126.5662  96.65511 132.8262
## Jul 2021       115.5232 103.4477 127.5986  97.05537 133.9910
## Aug 2021       115.2112 102.8909 127.5315  96.36890 134.0535
## Sep 2021       115.2616 102.7012 127.8221  96.05214 134.4711
## Oct 2021       115.8816 103.0855 128.6776  96.31173 135.4514
## Nov 2021       115.1799 102.1525 128.2074  95.25622 135.1037
## Dec 2021       116.3818 103.1261 129.6375  96.10898 136.6547
```

```
ts_arima_forecast2
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       114.7401 107.77331 121.7068 104.08533 125.3948
## Feb 2020       114.9747 107.38190 122.5674 103.36255 126.5868
## Mar 2020       115.3724 107.31287 123.4320 103.04640 127.6985
## Apr 2020       115.8784 106.45606 125.3008 101.46817 130.2886
## May 2020       115.6653 105.67174 125.6588 100.38148 130.9491
## Jun 2020       113.6008 103.01886 124.1827  97.41713 129.7844
## Jul 2020       117.7331 106.37710 129.0891 100.36558 135.1007
## Aug 2020       116.7195 104.82278 128.6161  98.52506 134.9138
## Sep 2020       116.5786 104.12503 129.0321  97.53253 135.6246
## Oct 2020       116.9186 103.88153 129.9556  96.98014 136.8570
```

```
## Nov 2020       115.2441 101.70482 128.7833  94.53758 135.9506
## Dec 2020       117.9446 103.90162 131.9875  96.46774 139.4214
## Jan 2021       115.5991  99.92731 131.2708  91.63117 139.5670
## Feb 2021       116.5369 100.06182 133.0120  91.34044 141.7333
## Mar 2021       116.9334  99.72487 134.1420  90.61520 143.2517
## Apr 2021       118.1317  99.91544 136.3479  90.27234 145.9910
## May 2021       116.4948  97.53904 135.4505  87.50449 145.4850
## Jun 2021       115.8866  96.19720 135.5760  85.77425 145.9990
## Jul 2021       119.0082  98.53278 139.4835  87.69377 150.3225
## Aug 2021       117.9957  96.83326 139.1581  85.63056 150.3608
## Sep 2021       117.0739  95.22939 138.9185  83.66560 150.4823
## Oct 2021       116.2513  93.72618 138.7764  81.80210 150.7005
## Nov 2021       115.6059  92.44251 138.7694  80.18053 151.0313
## Dec 2021       119.4556  95.66284 143.2483  83.06772 155.8434
```

```
fit_forecast
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       113.8335 108.4221 119.2449 105.55753 122.1095
## Feb 2020       113.1457 107.2912 119.0003 104.19202 122.0995
## Mar 2020       115.1130 108.8466 121.3794 105.52934 124.6966
## Apr 2020       114.1551 107.5022 120.8079 103.98044 124.3297
## May 2020       114.8118 107.7938 121.8298 104.07868 125.5450
## Jun 2020       115.7010 108.3359 123.0662 104.43707 126.9650
## Jul 2020       112.8580 105.1614 120.5546 101.08709 124.6289
## Aug 2020       113.0507 105.0363 121.0650 100.79380 125.3076
## Sep 2020       113.3830 105.0630 121.7030 100.65866 126.1073
## Oct 2020       113.3904 104.7756 122.0052 100.21520 126.5656
## Nov 2020       114.3110 105.4111 123.2108 100.69985 127.9221
## Dec 2020       112.7321 103.5561 121.9082  98.69863 126.7656
## Jan 2021       113.0631 103.8743 122.2519  99.01002 127.1162
## Feb 2021       112.7232 103.4114 122.0349  98.48213 126.9642
## Mar 2021       113.6955 104.2626 123.1285  99.26904 128.1221
## Apr 2021       113.2221 103.6693 122.7748  98.61243 127.8317
## May 2021       113.5467 103.8757 123.2177  98.75619 128.3372
## Jun 2021       113.9862 104.1984 123.7740  99.01706 128.9554
## Jul 2021       112.5809 102.6777 122.4842  97.43519 127.7266
## Aug 2021       112.6762 102.6588 122.6935  97.35591 127.9964
## Sep 2021       112.8404 102.7102 122.9706  97.34760 128.3332
## Oct 2021       112.8441 102.6023 123.0859  97.18059 128.5076
## Nov 2021       113.2991 102.9469 123.6513  97.46680 129.1314
## Dec 2021       112.5187 102.0573 122.9801  96.51936 128.5181
```