Bias - variance tradeoff in time series

Caglar Altunel

12/24/2020

Bias-variance tradeoff in time series

Bias-variance tradeoff is an important concept that one must always pay regard when modelling for both prediction and forecasting purposes. There are several measures that provides us what the best fit is for our dataset while modelling, such as minimum mean squared error, AIC etc. However, while minimizing these values, we may create a model that captures almost the entire bias (low bias) in the data that use for modelling, but significantly fails in another data simply because that happens to be “so specific” for the dataset we used (high variance). This is called overfitting, refers to when a statistical model incorporates noise rather than real patterns in the data. Long story short, while making our model less biased, we increase variance. Therefore, we must build models that can be generalized so that we reach more or less the same accuracy when tested on a different dataset. Below codes provides an example of bias-variance tradeoff and overfitting in time series. Here the target is to forecast final consumption expenditure of Australia, for the period quarter one of 2013 - quarter four of 2018, by making use of the training time series quarter two of 1959 - quarter four of 2012. The dataset can be downloaded from: https://stats.oecd.org/Index.aspx?DataSetCode=QNA. I would like to thank so much to my thesis partner, Hallvard Holte for his exceptional contribution.

Let’s start with loading the essential packages.

```
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 3.6.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: forecast
```

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

```
## Loading required package: fma
```

```
## Warning: package 'fma' was built under R version 3.6.3
```

```
## Loading required package: expsmooth
```

```
## Warning: package 'expsmooth' was built under R version 3.6.3
```

```
library(seasonal)
```

```
## Warning: package 'seasonal' was built under R version 3.6.3
```

Loading data and decomposition

This is where the dataset is downloaded to, so I set this as working directory.

```
setwd("C:/Users/Caglar/Desktop/Github")
```

Below code loads the dataset to R environment as time series.

```
cons = ts(scan("cons.txt"),start=1959.5,frequency=4)
```

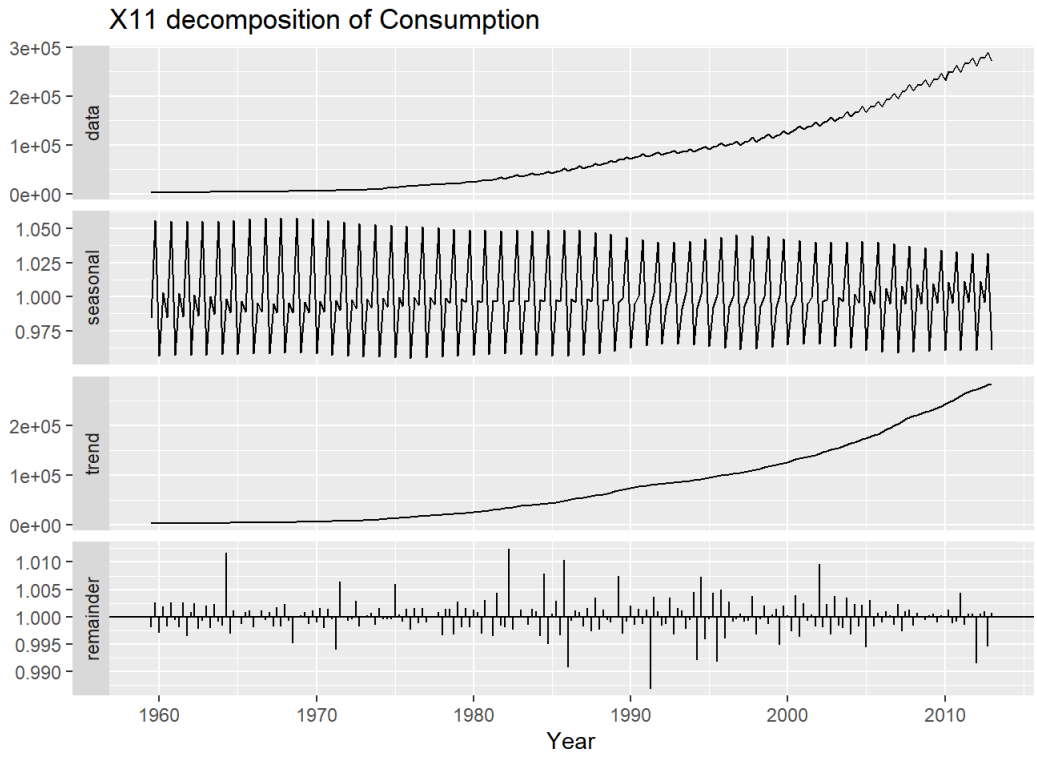
The dataset is divided into two, train and test, so that we will test the model on a new dataset to figure out whether overfitting occurs or not. The period, from Q1 2013 onwards, to be forecasted, therefore it is filtered out as the test data.

```
cons_train = window(cons,end=2013)
cons_test = window(cons,start=2013.25)
```

First, training data is decomposed to see trend and seasonality. It can be seen (the output of the below code) from the systematic variations

that seasonality is quite strong in the data. Time series has an obvious upward trend too. At the first place, data will be stationarized (The impact of trend and seasonality will be erased so the data we will use for modelling will have no or very limited temporal patterns.) in order to make it ready for time series modelling.

```
consumption_x11=seas(cons_train, x11="") #decomposition of consumption using X11
autoplot(consumption_x11)+
  ggtitle("X11 decomposition of Consumption")+xlab("Year")
```



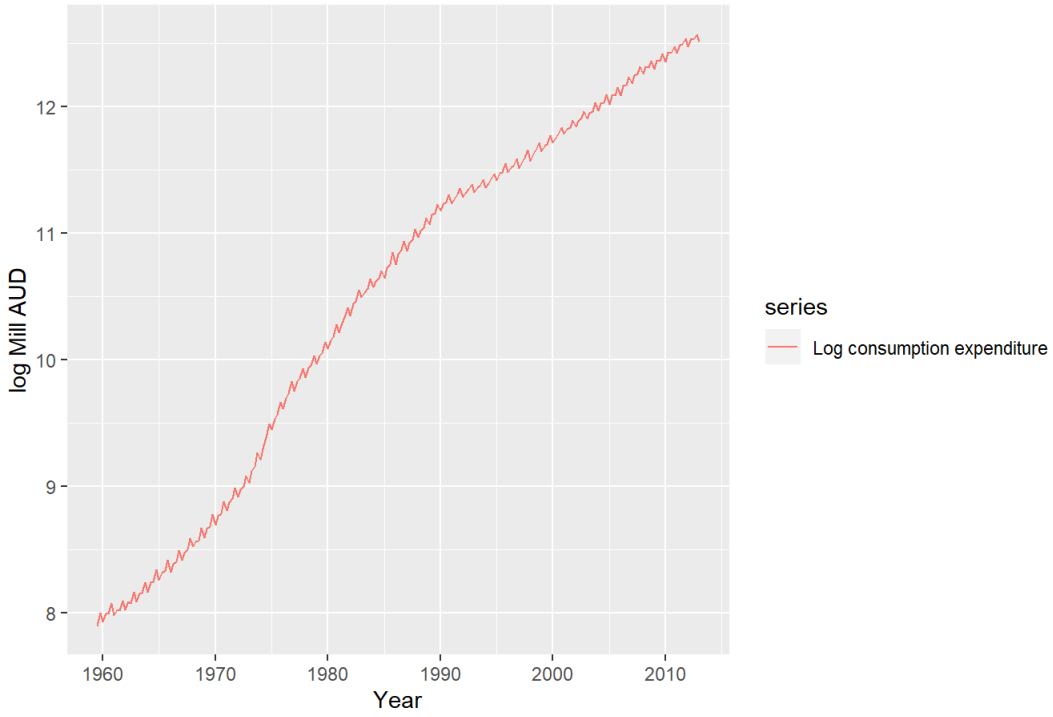
How to make the data stationary

In order to stabilize the trend, time series is log-scaled by the below code.

```
cons_log = c()

for (t in 1:215){
  cons_log=c(cons_log,log(cons_train[t]))
}

cons_log = ts(cons_log,start=1959.5,frequency=4)
```

It can be seen from the below plot that the trend is still there, but at least it is no longer exponential. Further modifications needed to erase the temporal pattern.

```
autoplot(cons_log,series="Log consumption expenditure",xlab="Year",ylab="log Mill AUD")
```

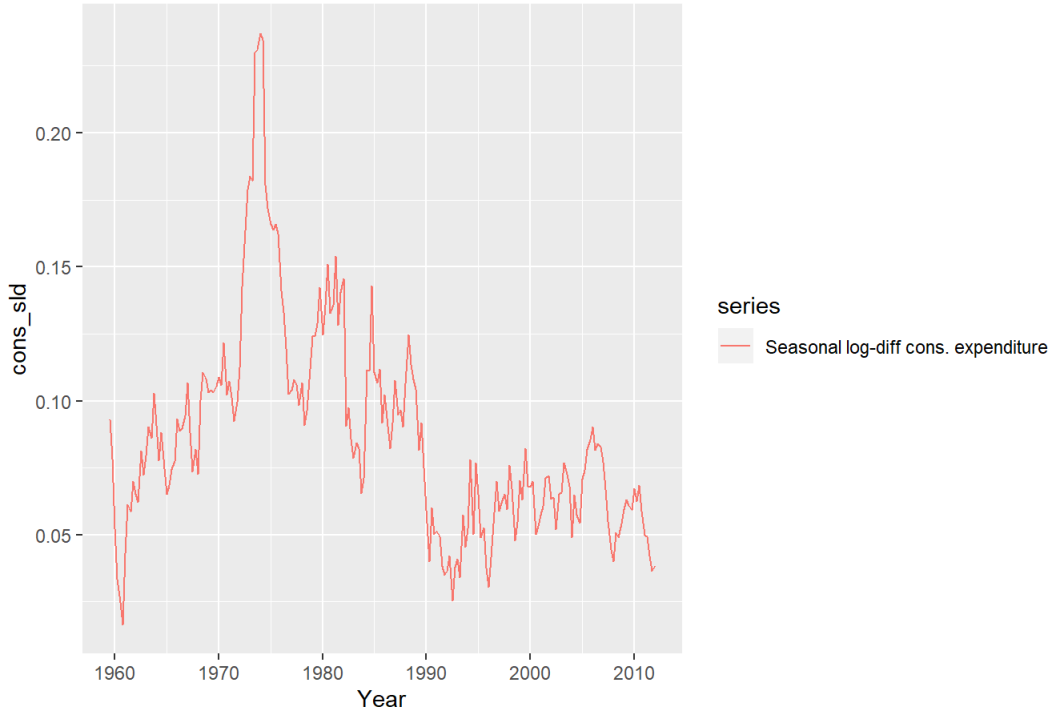


As a second step, a seasonal log-difference transformation is pursued to see whether we manage to reduce the temporal effect in the data or not.

```
cons_sld = c() #Seasonal log-diff transformation

for (t in 5:215){
  cons_sld=c(cons_sld,log(cons_train[t])-log(cons_train[t-4]))
}

cons_sld = ts(cons_sld,start=1959.5,frequency=4)
```

The output suggests the trend is managed to be erased, but still a noticeable seasonality left (There are systematic spikes in the data.).

```
autoplot(cons_sld,series="Seasonal log-diff cons. expenditure",xlab="Year")
```

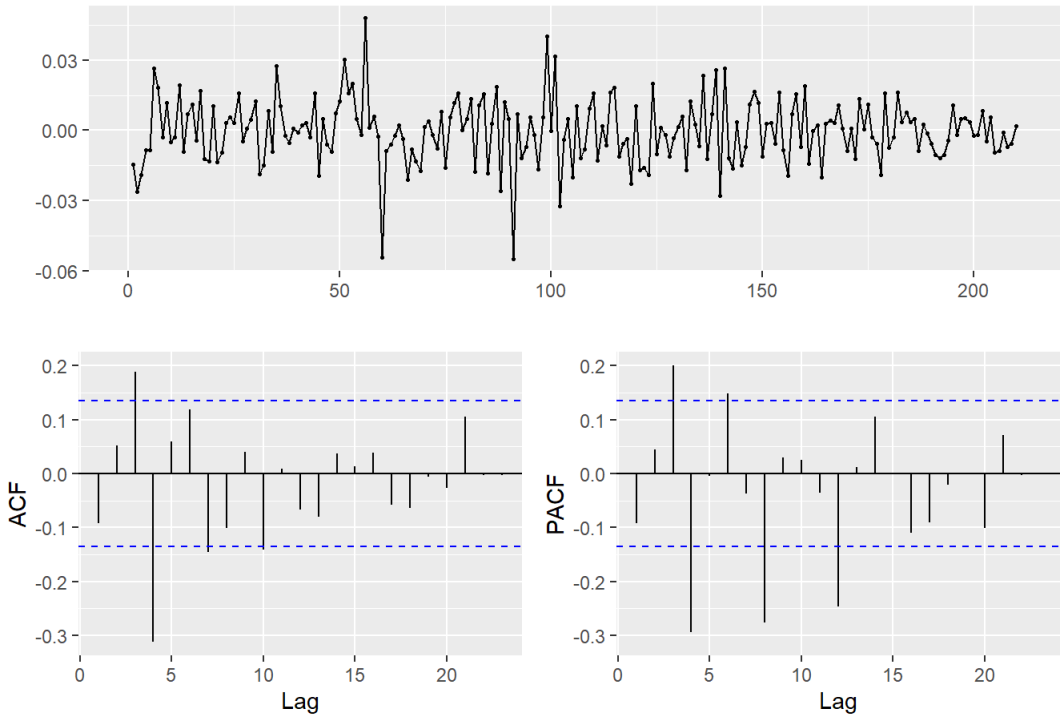


As a final step, we take the first difference of seasonally log-differenced dataset, to reduce the seasonality that we spotted above.

```
# First difference of the log-seasonal diff
cons_fsld = c()

for (t in 2:211){
  cons_fsld=c(cons_fsld,cons_sld[t]-cons_sld[t-1])
}

cons_fsld = ts(cons_fsld)
```

ACF, PACF (Please visit https://www.real-statistics.com/time-series-analysis/stochastic-processes/partial-autocorrelation-function/ for the detailed explanation of PACF and https://www.real-statistics.com/time-series-analysis/stochastic-processes/autocorrelation-function/ for ACF) and residuals show that the temporal pattern is deducted from the dataset at a certain extent. The significant spikes that can be seen in ACF and PACF will give us an idea at what degree we will use AR and MA components.

```
ggtsdisplay(cons_fsld)
```



Finding the best-fitting model

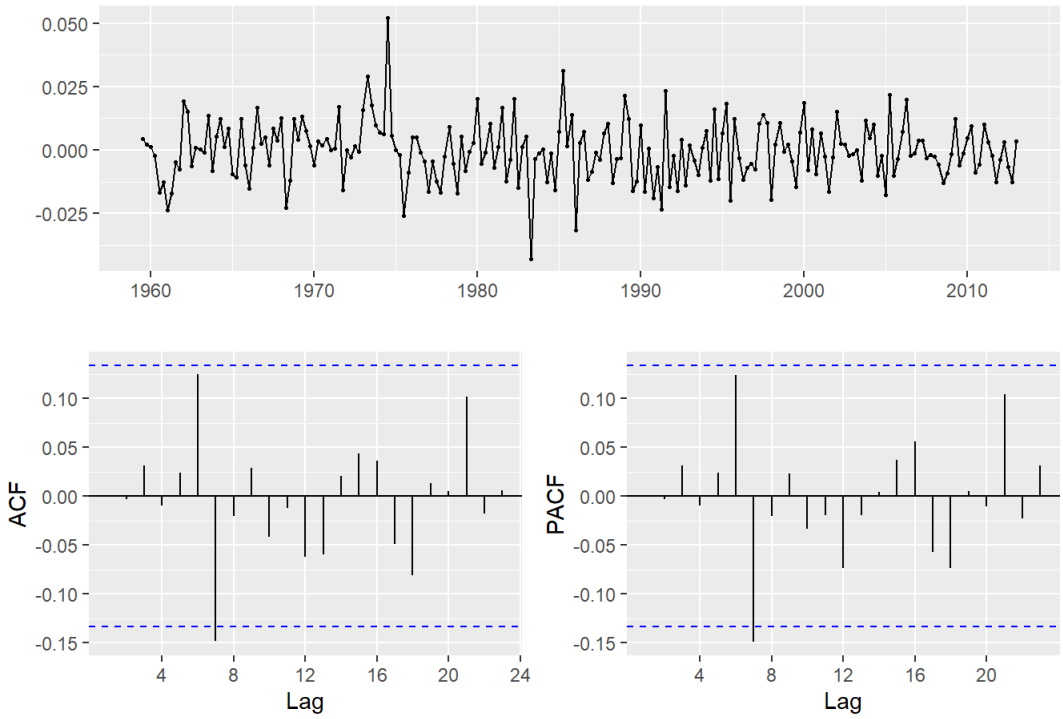
But beforehand, in order to exemplify how one can reach an overfitting model as the best-fit, let's run the auto.arima() function, which automatically gives the best-fit, with respect to the algorithm it follows. Please visit https://otexts.com/fpp2/ for the detailed explanation of the functions that are used in this article. Please notice that we used the log-scaled time series for modelling, as we made transformations in the above to figure out the level of the temporal pattern in the data. auto.arima() function gave us that the best model is arima(2,1,1)(1,1,2) (Please visit https://otexts.com/fpp2/arima.html for the detailed explanation of the concept of ARIMA models.).

```
arima_auto=auto.arima(cons_log) #auro.arima() function gives (2,1,1)(1,1,2)
summary(arima_auto)
```

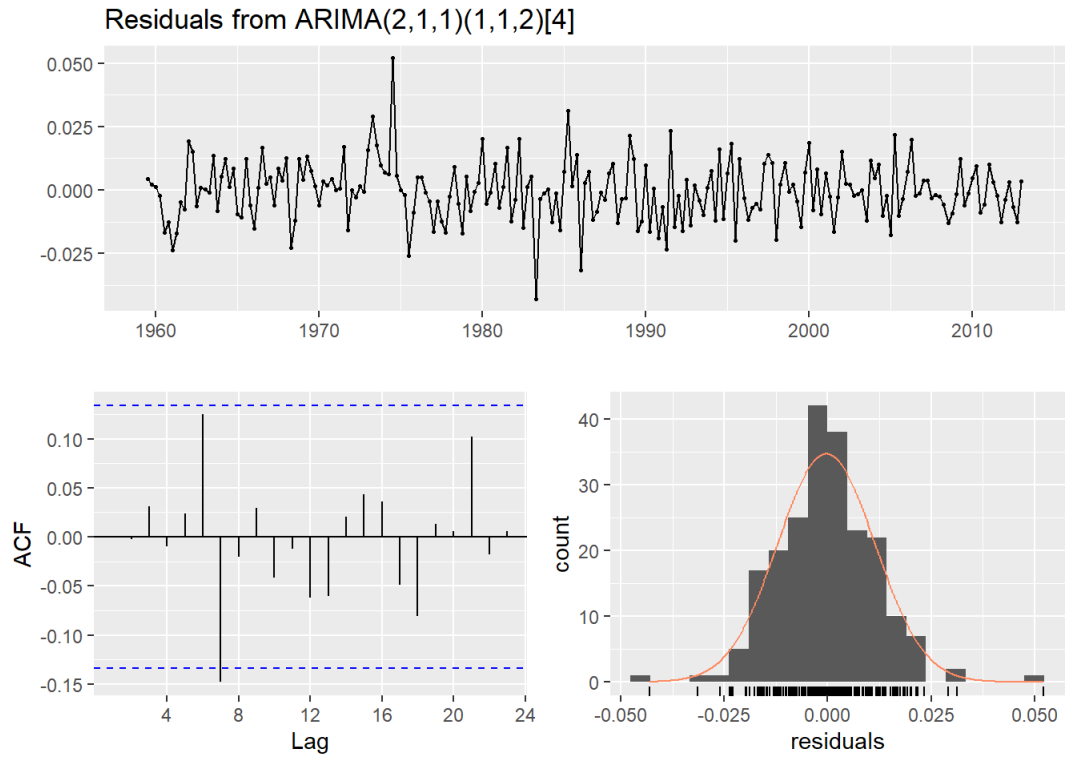
```
## Series: cons_log
## ARIMA(2,1,1)(1,1,2)[4]
##
## Coefficients:
##          ar1     ar2      ma1     sar1     sma1     sma2
##       0.5371  0.2429  -0.5773  -0.0797  -0.5241  -0.1834
## s.e.  0.1409  0.0747   0.1362   0.3783   0.3674   0.2568
##
## sigma^2 estimated as 0.0001456:  log likelihood=632.63
## AIC=-1251.26   AICc=-1250.71   BIC=-1227.83
##
## Training set error measures:
##                       ME        RMSE         MAE          MPE       MAPE
## Training set -0.0003805983 0.01175345 0.008933683 -0.00266028 0.08715793
##                   MASE          ACF1
## Training set 0.102754 -0.0006977315
```

ACF, PACF and the normal distribution of residuals show that the model can be deemed as unbiased, as there is also almost no temporal pattern left.

```
ggtsdisplay(residuals(arima_auto))
```




```
checkresiduals(arima_auto)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(1,1,2)[4]
## Q* = 9.0184, df = 3, p-value = 0.02905
##
## Model df: 6.   Total lags used: 9
```

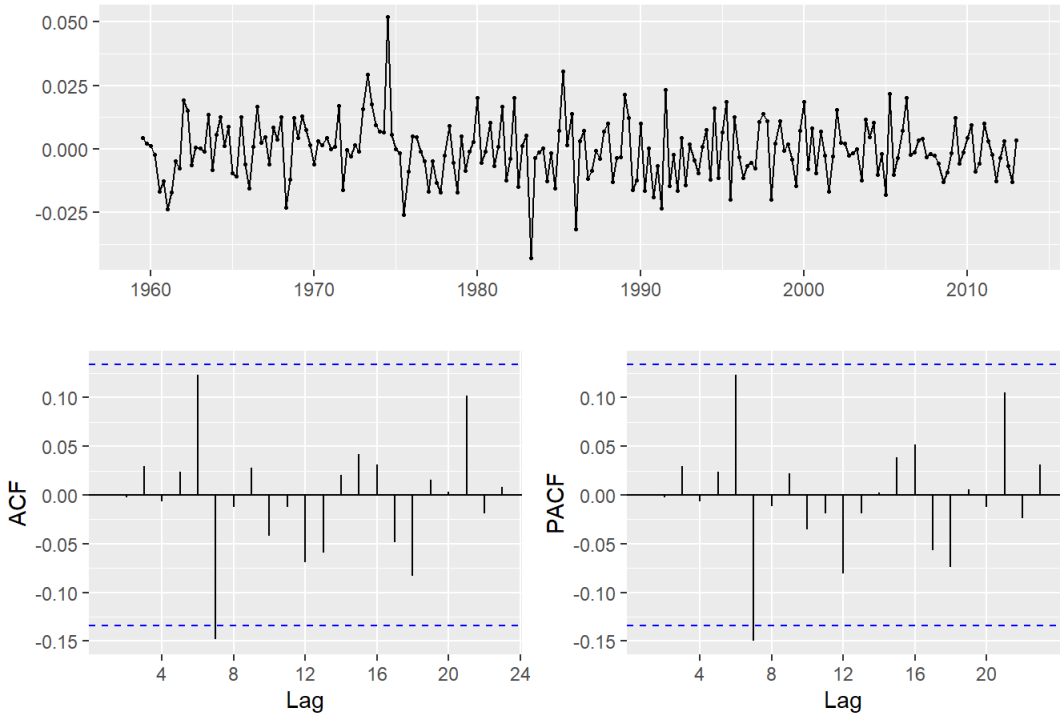
However, due to the algorithm that auto.arima() function pursues, we may be missing the real best-fitting one. In fact, the decaying spikes until lag 12 in PACF that we found out within our stationarized data (ggtsdisplay(cons_fsld)) may be pointing out the fact that a model with three seasonally differentiated error terms is a better-fit. We will give this a try by the below code:

```
arima_manual=Arima(cons_log, order=c(2,1,1), seasonal=c(0,1,3)) #However, ggtsdisplay(cons_fsld) shows seaso
nal MA(3).
summary(arima_manual) #In fact, it has a slightly better AICc, and RMSE.
```

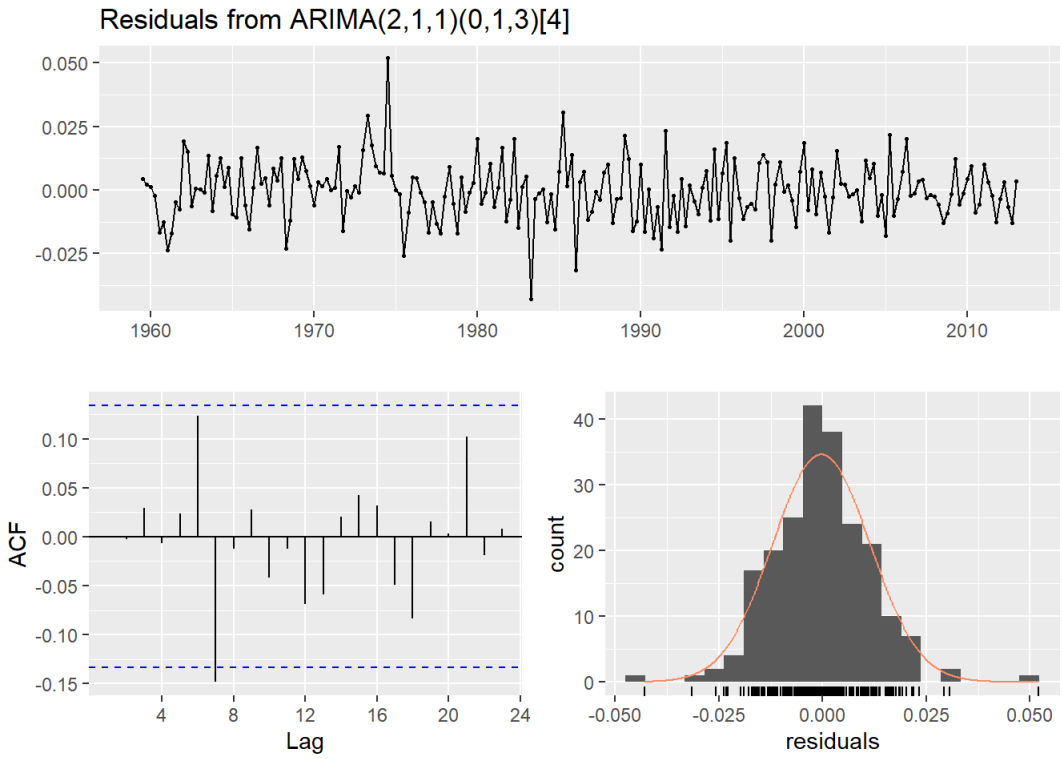
```
## Series: cons_log
## ARIMA(2,1,1)(0,1,3)[4]
##
## Coefficients:
##          ar1     ar2      ma1     sma1     sma2    sma3
##       0.5373  0.2456  -0.5783  -0.6090  -0.1415  0.0244
## s.e.  0.1391  0.0758   0.1342   0.0782   0.0770  0.0765
##
## sigma^2 estimated as 0.0001455:  log likelihood=632.66
## AIC=-1251.32   AICc=-1250.77   BIC=-1227.89
##
## Training set error measures:
##                       ME       RMSE         MAE           MPE       MAPE
## Training set -0.000376725 0.01175148 0.008933554 -0.002638319 0.08716457
##                    MASE          ACF1
## Training set 0.1027526 -0.0004030778
```

The summary shows indeed it has slightly better AICc and RMSE compared to ARIMA (2,1,1)(1,1,2). ACF, PACF and the distribution of residuals point out that the model can be deemed as unbiased.

```
ggtsdisplay(residuals(arima_manual))
```




```
checkresiduals(arima_manual)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(0,1,3)[4]
## Q* = 8.8722, df = 3, p-value = 0.03104
##
## Model df: 6.   Total lags used: 9
```

Theoretical ACF and PACF, and proposal of a simplified model

However, we need to know how our model's theoretical ACF and PACF would look, which reflects ACF and PACF plots theoretically if all of the assumptions of the linear regression model hold. The equation for ARIMA (2,1,1)(0,1,3) must be solved initially. Which is:

$$(1-\phi_1 B-\phi_2 B^2)(1-B)(1-B^4)y_t=(1+\theta_1 B)(1+\Theta_1 B^4+\Theta_2 B^8+\Theta_3 B^{12})\varepsilon_t$$

unchanged image

When it is solved for y, the outcome is:

$$y_t=(1-\phi_1)y_{t-1}+(\phi_2-\phi_1)y_{t-2}-\phi_2 y_{t-3}+y_{t-4}-(1+\phi_1)y_{t-5}+(\phi_1-\phi_2)y_{t-6}+\phi_2 y_{t-7}$$

$$+\varepsilon_t+\theta_1\varepsilon_{t-1}+\Theta_1\varepsilon_{t-4}+\theta_1\Theta_1\varepsilon_{t-5}+\Theta_2\varepsilon_{t-8}+\theta_1\Theta_2\varepsilon_{t-9}+\Theta_3\varepsilon_{t-12}+\theta_1\Theta_3\varepsilon_{t-13}$$

Where:

$\phi_1 = 0.5373$

$\phi_2 = 0.2456$

$\theta_1 = -0.5783$

$\Theta_1 = -0.6090$

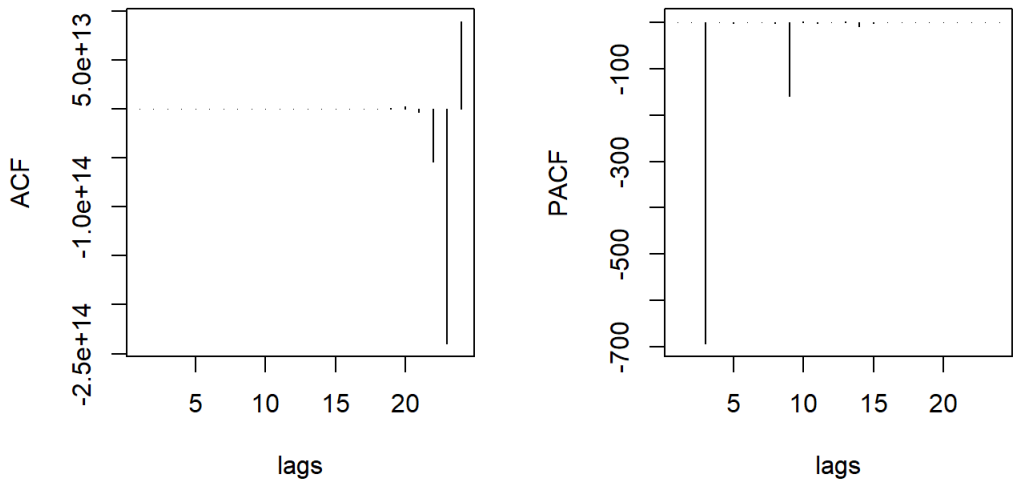
$\Theta_2 = -0.1415$

$\Theta_3 = 0.0244$

unchanged image

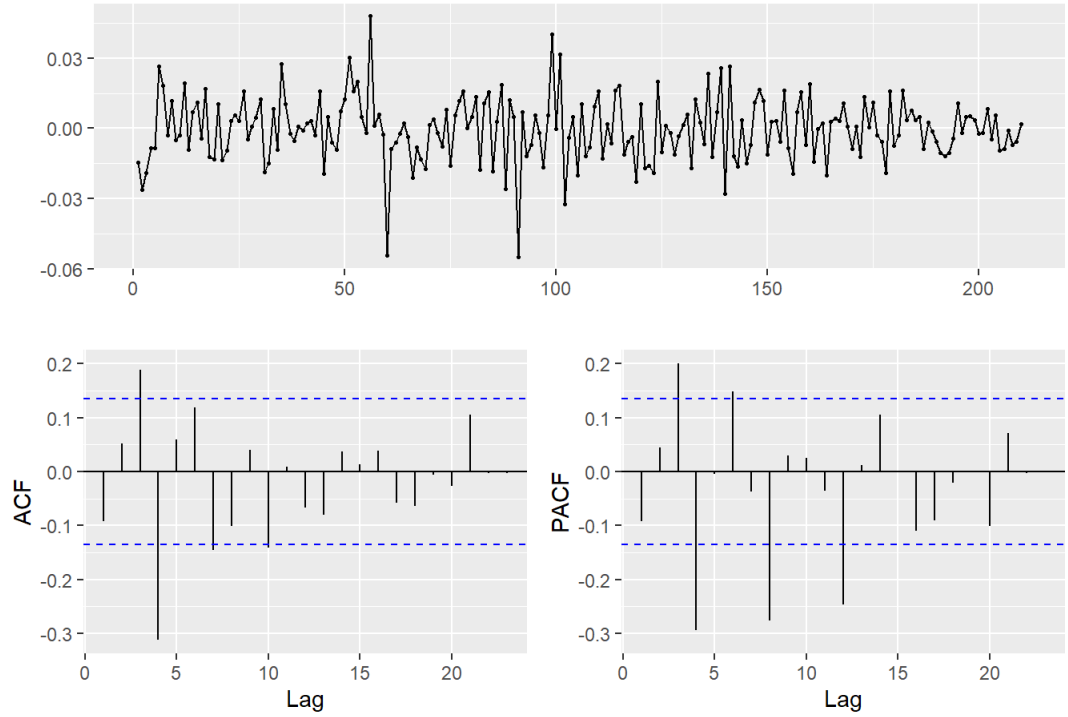
Theoretical ACF and PACF plots for the above model can be created by the below code:

```
## Another way of identifying a model is having a look at theoretical ACF and PACF.
arma_acf=ARMAacf(ar=c(1.5373, -0.2917, -0.2456, 1, -1,5373, 0.2917, 0.2456), ma=c(-0.5783, 0, 0, -0,6090, 0
.3521847, 0, 0, -0.1415, 0.08182945, 0, 0, 0.0244, -0.01411052), lag.max=24)
arma_pacf=ARMAacf(ar=c(1.5373, -0.2917, -0.2456, 1, -1,5373, 0.2917, 0.2456), ma=c(-0.5783, 0, 0, -0,6090,
0.3521847, 0, 0, -0.1415, 0.08182945, 0, 0, 0.0244, -0.01411052), lag.max=24, pacf=T)
par(mfrow=c(1,2), pty="s")
plot(arma_acf[-1],type="h", xlab="lags", ylab="ACF")
plot(arma_pacf,type="h", xlab="lags", ylab="PACF")
```



It can be seen that it is far from what is obtained for our stationarized data. Let's recall it (in the below) and see how ACF and PACF looks like. As discussed above, now it is time to use it to build a time series model that is very unlikely to be overfitting. Relatively large one-time spike at lag 4 and decaying spikes from lag 4, 8, 12 and so on point out that building a typical MA(q) model where q is the seasonal difference.

```
ggtsdisplay(cons_fsld) #Recall ACF and PACF of our stationerized data.
```




```
# Significant spike at lag 4 in ACF along with a decaying partial correlations of lag 4, 8 and 12.
# This points out that an MA(q) model where q is the seasonal difference would be appropriate.
```

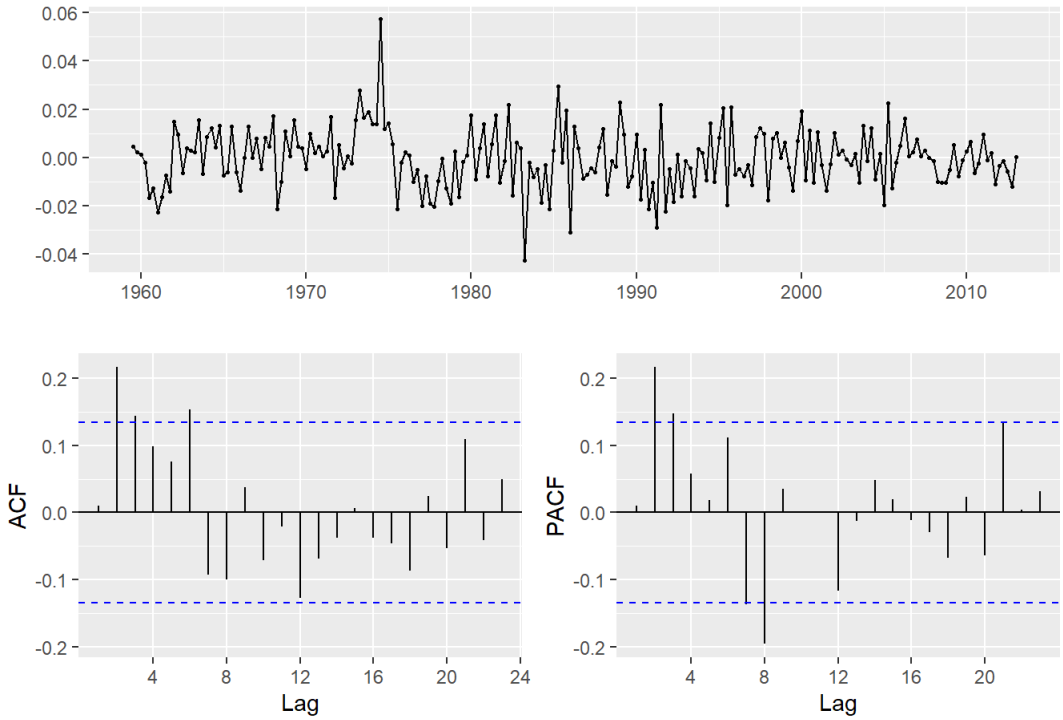
The below code is to build ARIMA(0,1,0)(0,1,1)

```
# Accordingly, lets construct an ARIMA model of (0,1,0)(0,1,1).
arima_manual2=Arima(cons_log, order=c(0,1,0), seasonal=c(0,1,1))
summary(arima_manual2) # Its fit is not better than ARIMA (2,1,1)(0,1,3) model.
```

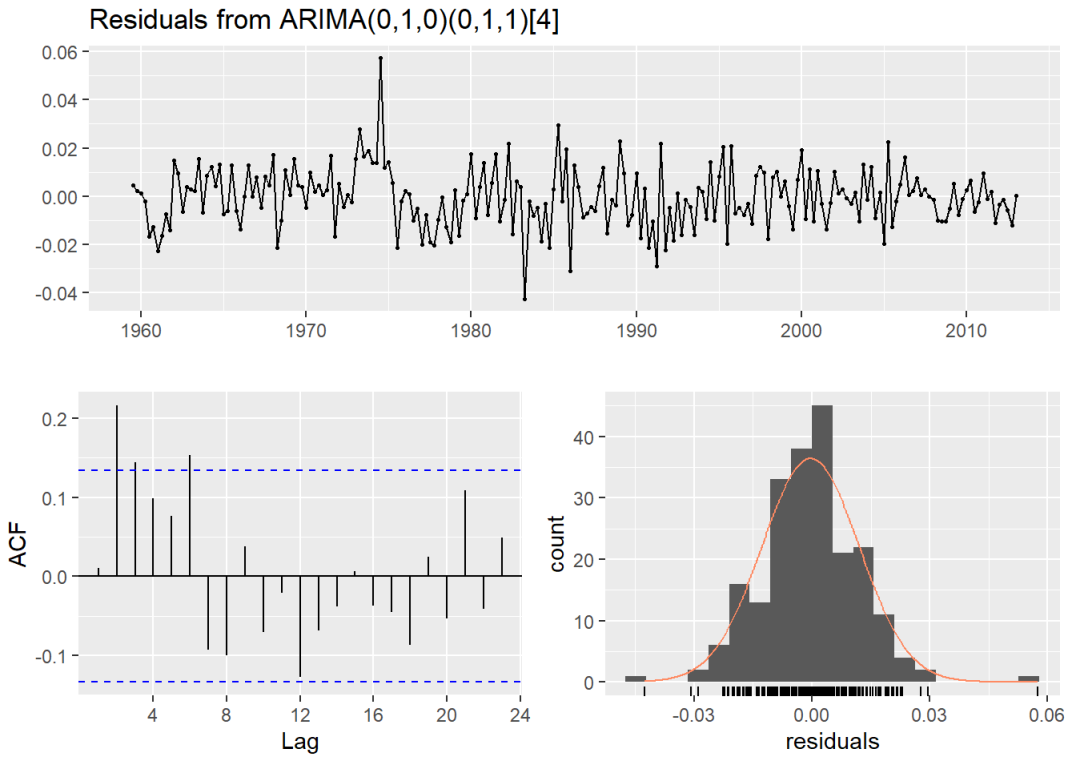
```
## Series: cons_log
## ARIMA(0,1,0)(0,1,1)[4]
##
## Coefficients:
##          sma1
##       -0.5700
## s.e.   0.0712
##
## sigma^2 estimated as 0.0001575:  log likelihood=622.11
## AIC=-1240.21   AICc=-1240.16   BIC=-1233.52
##
## Training set error measures:
##                         ME        RMSE         MAE           MPE       MAPE
## Training set -0.0003863259 0.01237402 0.009530037 -0.002257481 0.09339329
##                    MASE       ACF1
## Training set 0.1096132 0.01026202
```

The model's AICc is fair enough. Let's see how its ACF, PACF and residuals look like:

```
ggtsdisplay(residuals(arima_manual2))
```



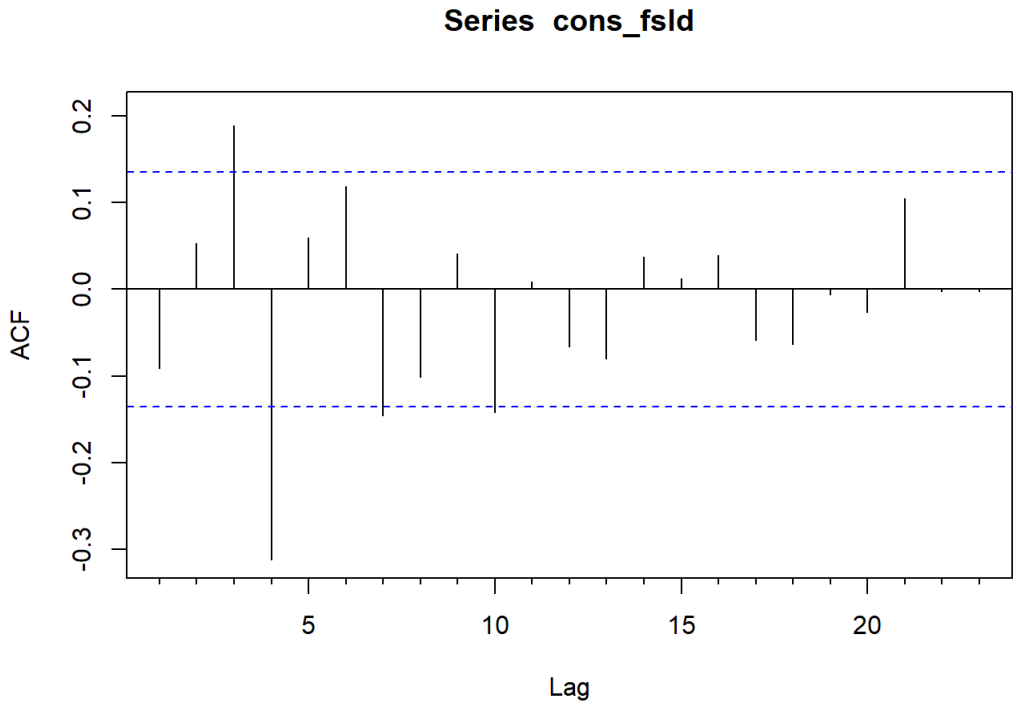

```
checkresiduals(arima_manual2)
```



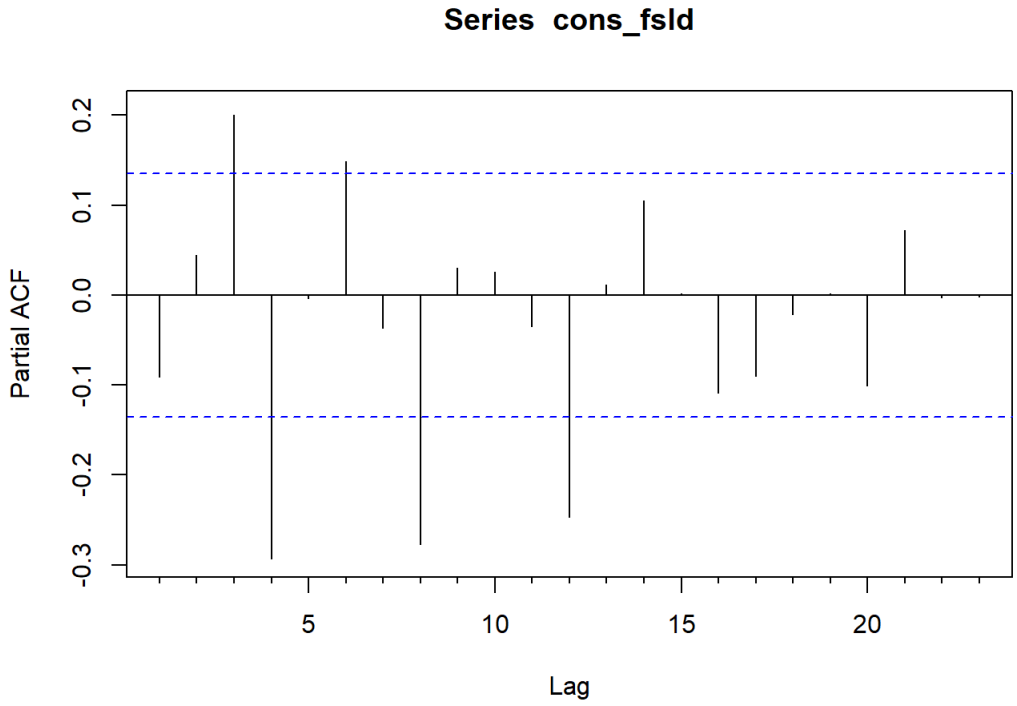

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)(0,1,1)[4]
## Q* = 27.665, df = 7, p-value = 0.0002529
##
## Model df: 1.   Total lags used: 8
```

Compared to our findings for the model ARIMA(2,1,1)(0,1,3), we can see that there is more temporal pattern left in this one. In other words, ARIMA(0,1,0)(0,1,1) captures less bias compared to ARIMA(2,1,1)(0,1,3). That being said, ARIMA(0,1,0)(0,1,1) most likely has less variance, therefore practically more useful for forecasting purposes. Beforehand, let's see how theoretical ACF and PACF of ARIMA(0,1,0)(0,1,1) look like:

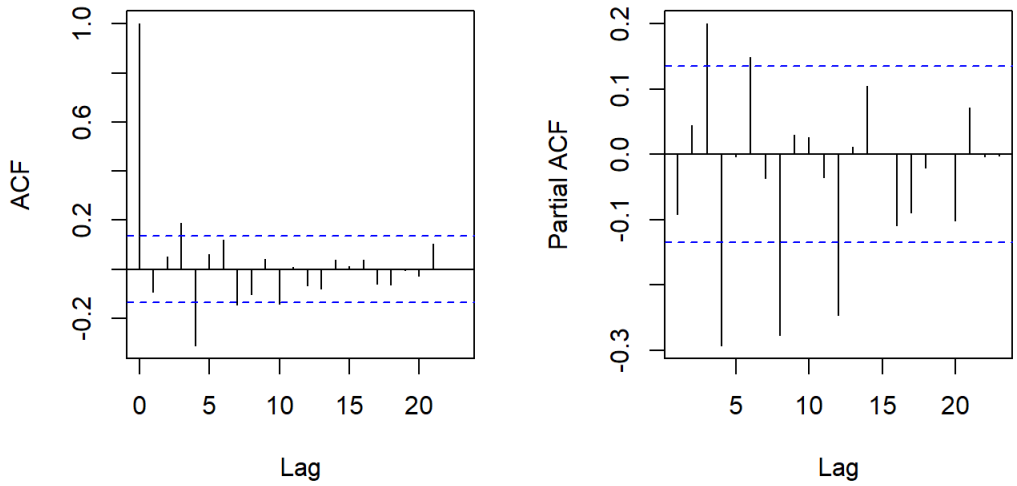
```
# Let's see the theoretical ACF and PACF of the model implied.
arma_acf2=ARMAacf(ma=c(0,0,0,-0.57), lag.max=24)
arma_pacf2=ARMAacf(ma=c(0,0,0,-0.57), lag.max=24, pacf=T)

ACFmanual2=Acf(cons_fsld)
```



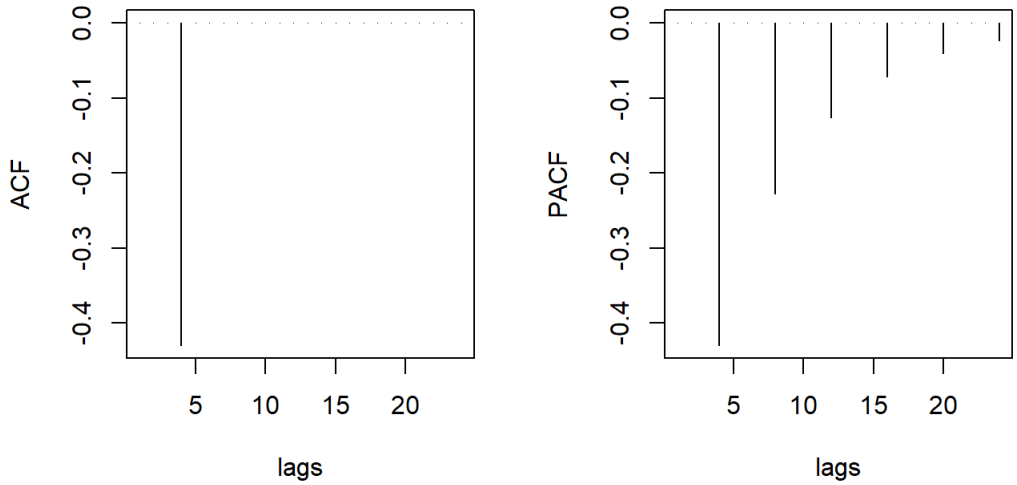

```
PACFmanual2=Acf(cons_fsld, type=c("partial"))
```




```
par(mfrow=c(1,2), pty="s")
plot(ACFmanual2, main="")
plot(PACFmanual2, main="")
```




```
par(mfrow=c(1,2), pty="s")
plot(arma_acf2[-1],type="h", xlab="lags", ylab="ACF")
plot(arma_pacf2,type="h", xlab="lags", ylab="PACF")
```

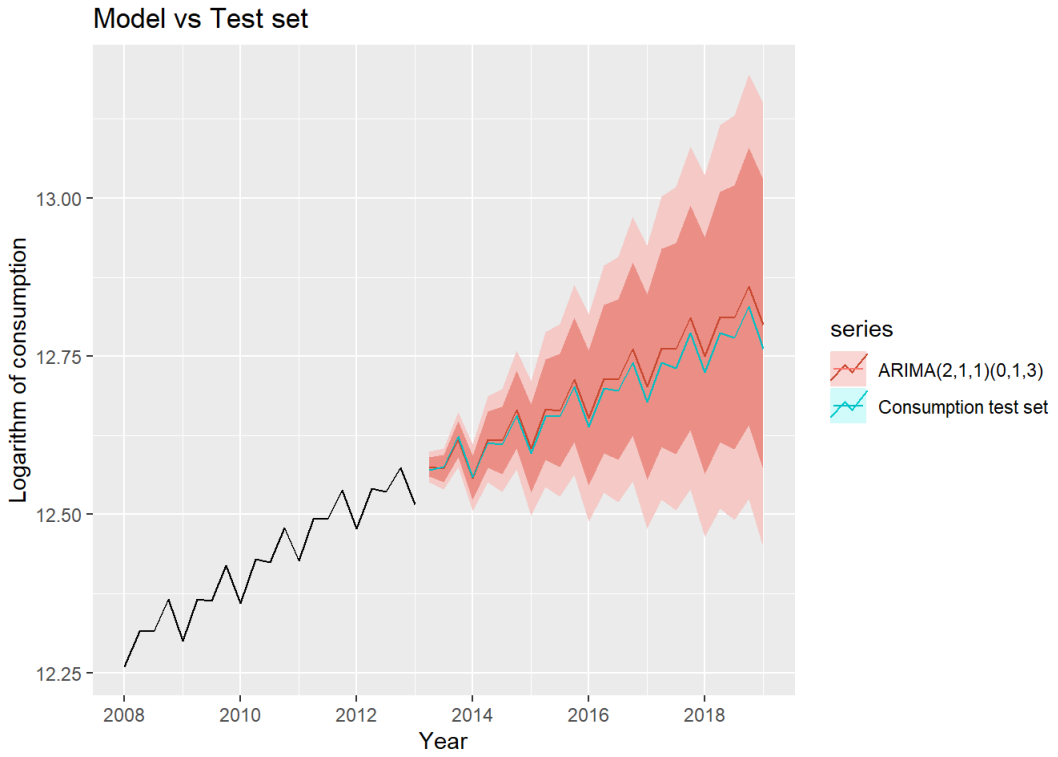



```
## Indeed variable's ACF and PACF trends looks similar with the model implied.
```

Comparing forecast performances

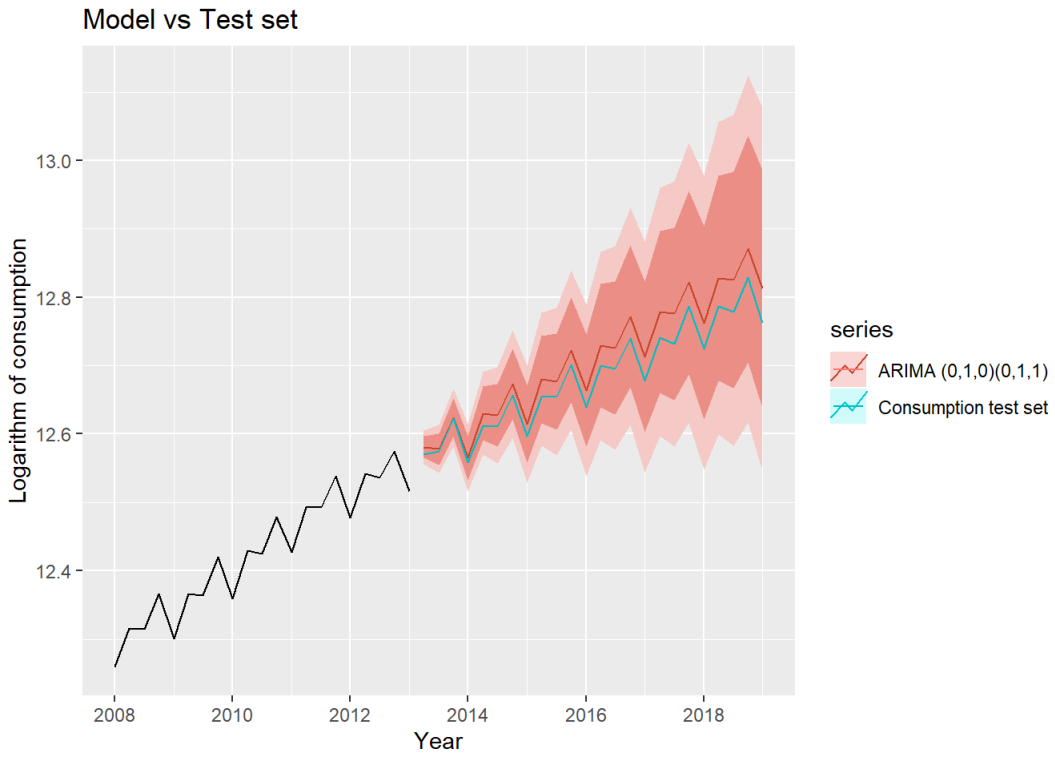
Theoretical ACF and PACF of ARIMA(0,1,0)(0,1,1) looks almost exactly like how we previously described the ACF and PACF plots of the stationerized data (ggtsdisplay(cons_fsld)). This gives us an idea that ARIMA(0,1,0)(0,1,1) is more practical for forecasting purposes. Let's compare forecasts of it with ARIMA(2,1,1)(0,1,3).

```
#Here in this part, we will compare ARIMA(2,1,1)(0,1,3) and (0,1,0)(0,1,1).
arima_manual_fc=forecast(arima_manual, h=24)
autoplot(window(cons_log,start=2008))+
  autolayer(arima_manual_fc,series="ARIMA(2,1,1)(0,1,3)")+
  autolayer(log(cons_test),series="Consumption test set")+
  ggtitle("Model vs Test set")+
  xlab("Year")+ylab("Logarithm of consumption")
```



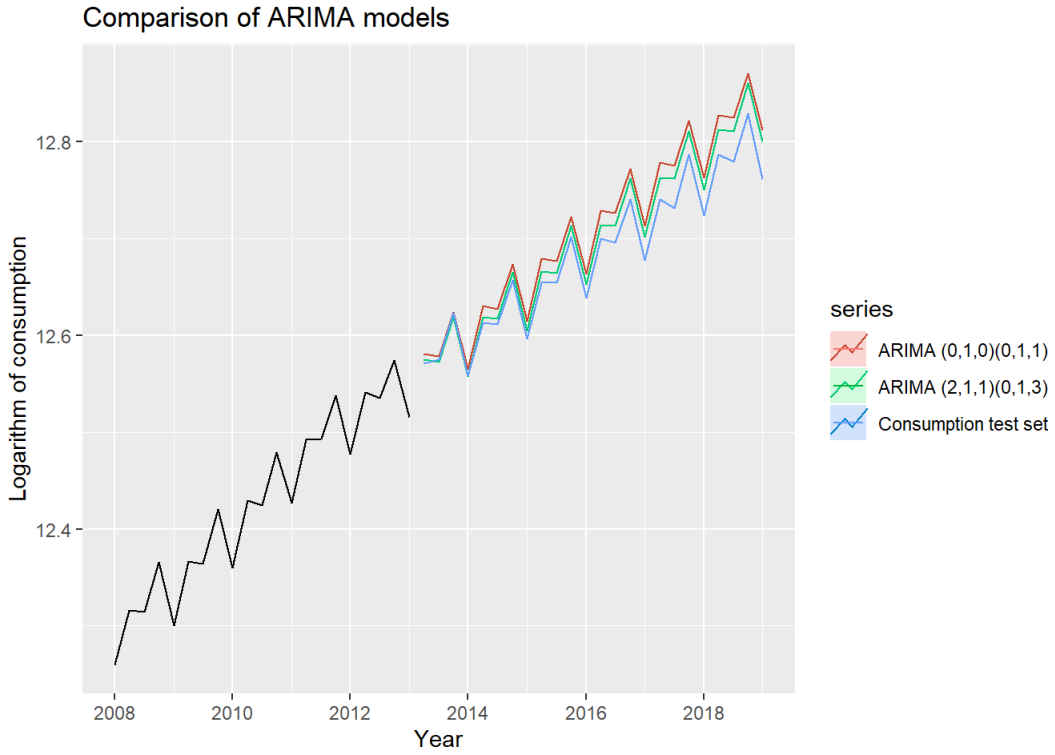
We can see that point forecast of ARIMA(2,1,1)(0,1,3) is very successful, but its forecast interval is too wide due to the fact that it has very high variance, so that it implies a range of stagnant to declining trend (lower bound) to an exponential growth in consumption (upper bound), which in practice does not make sense. Let's see how ARIMA(0,1,0)(0,1,1) will forecast:

```
arima_manual2_fc=forecast(arima_manual2, h=24)
autoplot(window(cons_log,start=2008))+
  autolayer(arima_manual2_fc,series="ARIMA (0,1,0)(0,1,1)")+
  autolayer(log(cons_test),series="Consumption test set")+
  ggtitle("Model vs Test set")+
  xlab("Year")+ylab("Logarithm of consumption")
```



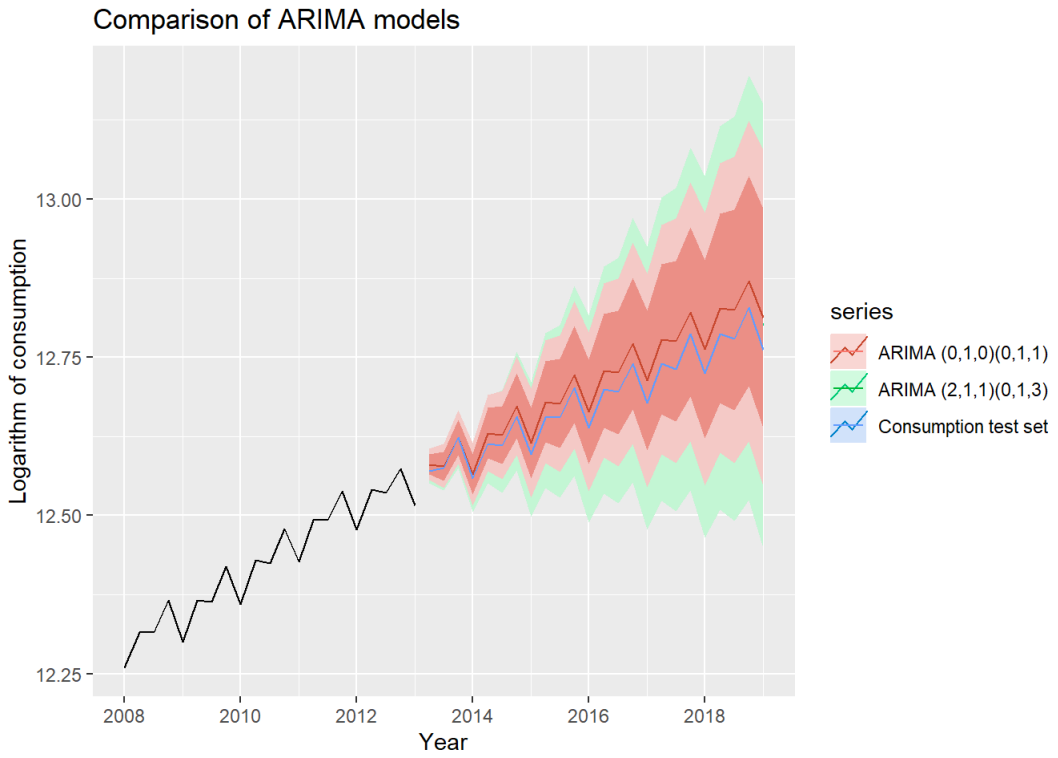
As expected, point forecast is slightly worse compared to ARIMA(2,1,1)(0,1,3), but, most importantly, its variance is fair, so that its forecast interval is much narrower. This model is statistically confident that consumption will continue to grow. Point forecast comparison can be seen by the below code:

```
autoplot(window(cons_log,start=2008))+
  autolayer(arima_manual_fc,series="ARIMA (2,1,1)(0,1,3)", PI=FALSE)+
  autolayer(arima_manual2_fc,series="ARIMA (0,1,0)(0,1,1)", PI=FALSE)+
  autolayer(log(cons_test),series="Consumption test set")+
  ggtitle("Comparison of ARIMA models")+
  xlab("Year")+ylab("Logarithm of consumption")
```



And the below code shows how better the forecast interval of ARIMA(0,1,0)(0,1,1) is, compared to ARIMA(2,1,1)(0,1,3).

```
autoplot(window(cons_log,start=2008))+
  autolayer(arima_manual_fc,series="ARIMA (2,1,1)(0,1,3)")+
  autolayer(arima_manual2_fc,series="ARIMA (0,1,0)(0,1,1)")+
  autolayer(log(cons_test),series="Consumption test set")+
  ggtitle("Comparison of ARIMA models")+
  xlab("Year")+ylab("Logarithm of consumption")
```




```
## - Conclusion: ARIMA(0,1,0)(0,1,1) has much narrower forecast interval. Therefore, we opt for that one.
```

Conclusion

By visually inspecting ACF and PACF plots of the stationerized data and comparing it with theoretical ACF and PACF plots, a more simplified model with less variance is built. This proves the fact that the best-fitting model that is found by the use of train data does not necessarily mean that it is also the most accurate one for forecasting. In most of the cases, due to bias-variance tradeoff, a more simplified, and therefore a more generalized model gives much more practical outcome.