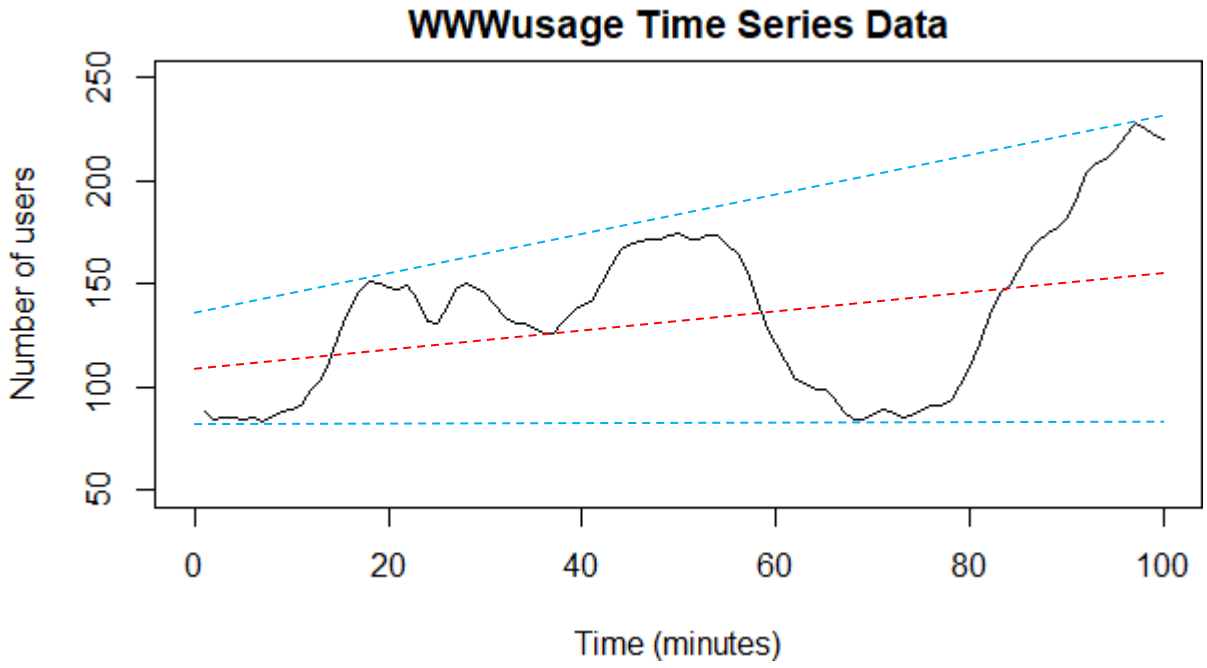
AI6123 Time Series Analysis

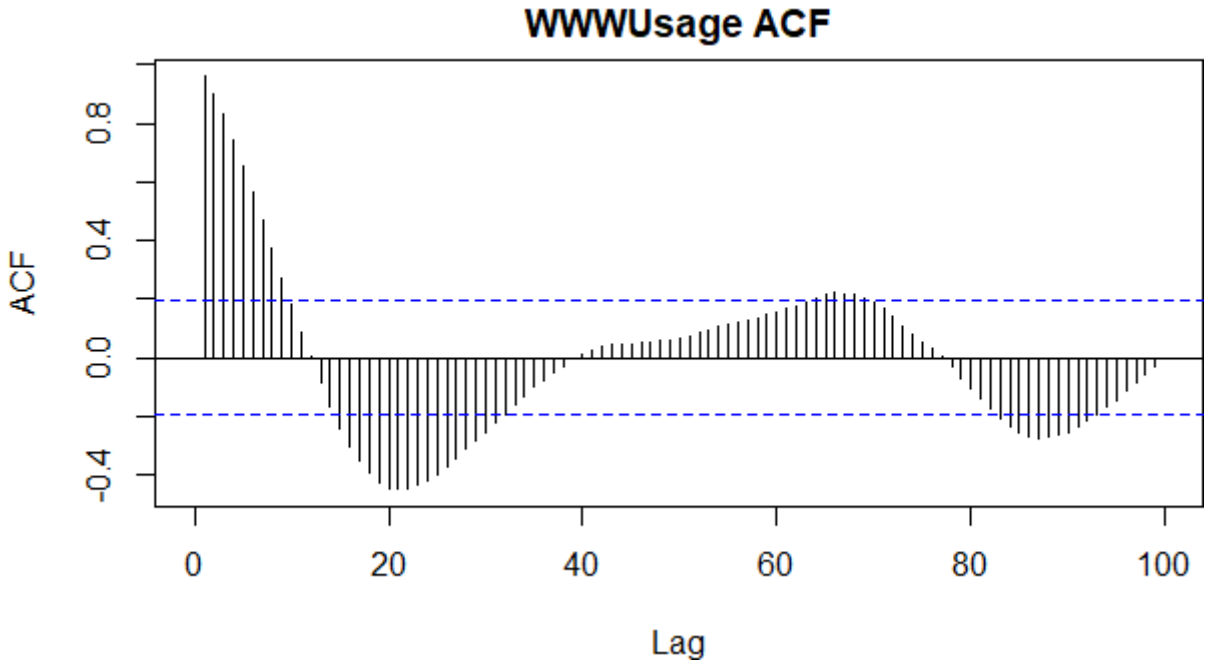
Chen Yongquan (G2002341D)
Nanyang Technological University

Assignment 1

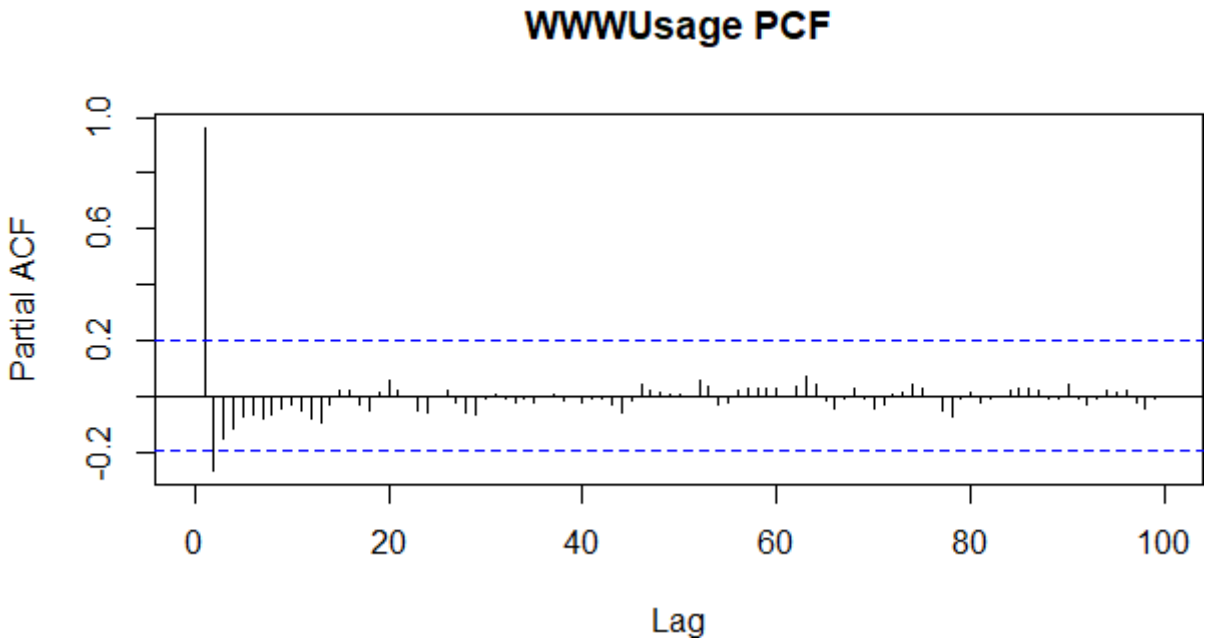
Contents



For this assignment we will be fitting an appropriate model for the WWWusage time series dataset [1]. Looking at the original plot of the data itself, we see a slightly increasing linear trend, so if the series does indeed have a trend component then it is not stationary. The variance seems to be increasing with time, but it doesn't seem to be due to a seasonality component as the periods are fluctuating. It might be due to a cyclical component, but we will keep these initial findings in mind and see if they remain after differencing.



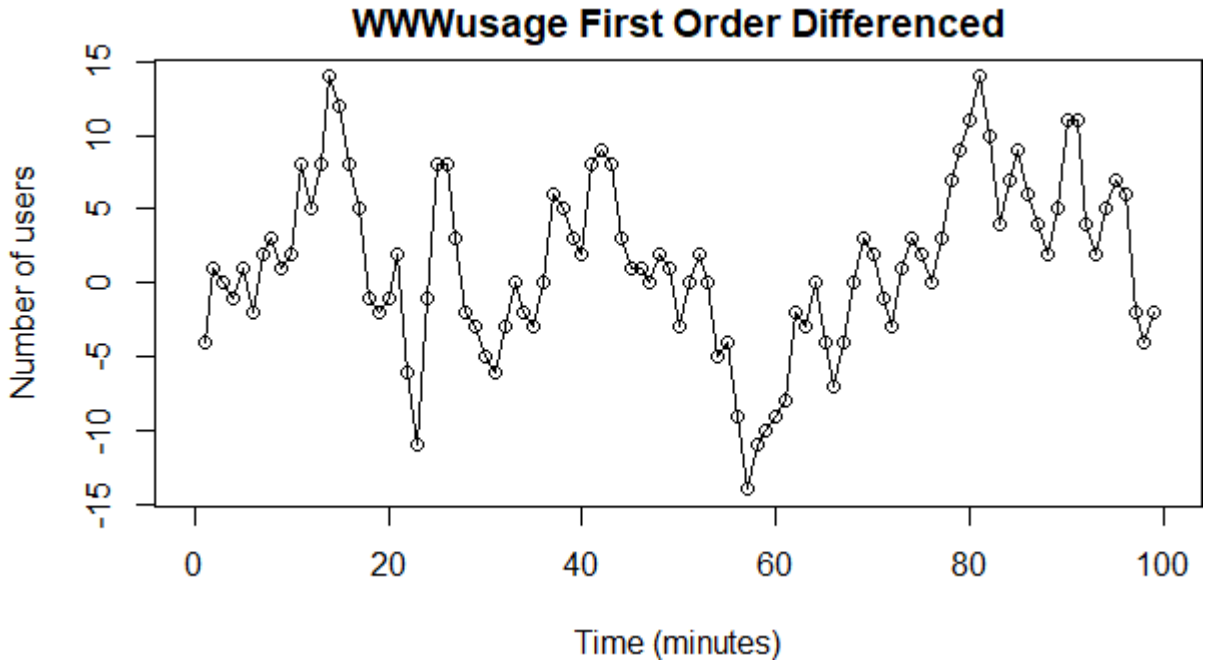
We can see that the sample ACF confirms our initial premise that the data is non-stationary as the ACF dies down slowly and does not cut off even after 100 lags.



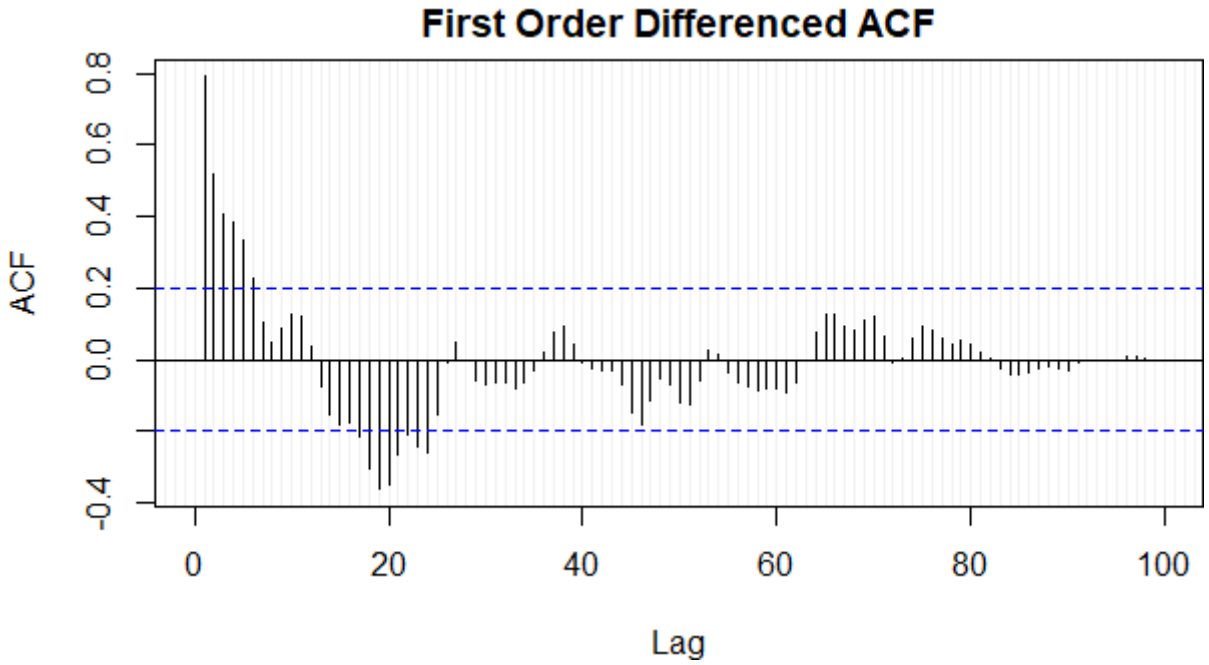
The sample PACF is useless for determining if our series is stationary or not. Also, because we have confirmed that the original data is non-stationary, stationary AR(p) or MA(q) models would not fit to it, so it is also useless for determining if an AR(p) model would be appropriate. However, we do see that the first partial autocorrelation is much higher than the others and close to 1.0, which again confirms that the data is non-stationary.

First Order Differencing

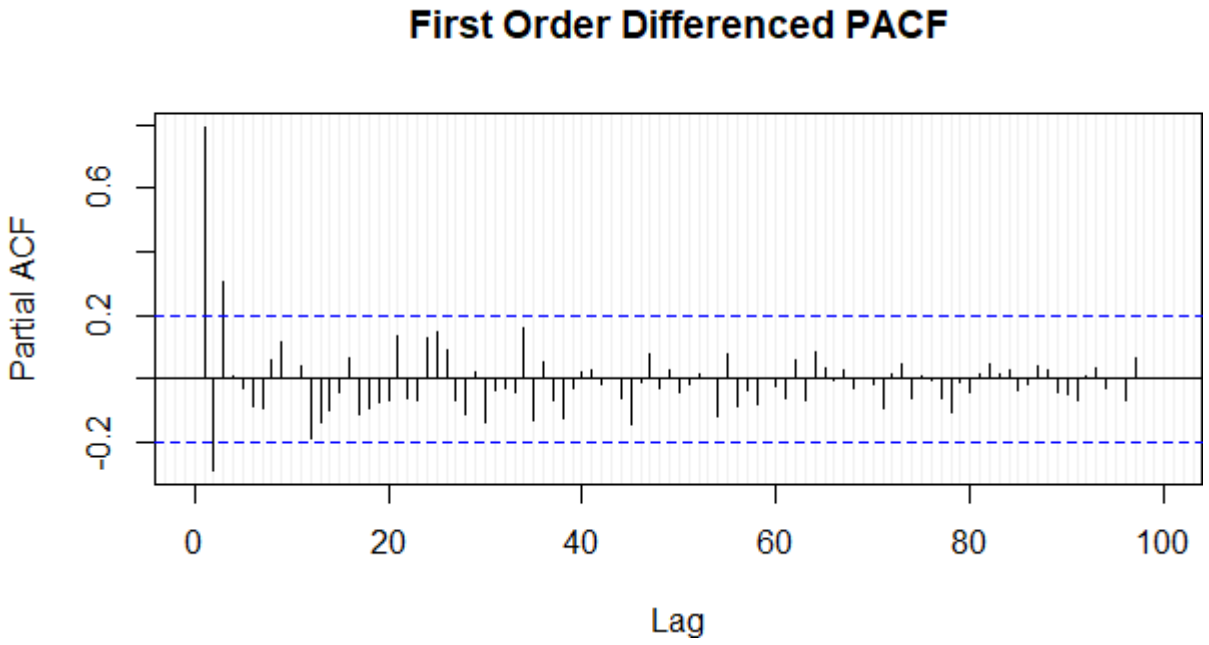
To make the data stationary, we will try to remove the slightly positive trend component by doing lag-1 differencing, and hopefully the cyclical pattern with increasing variance also disappears.



From the plot of the first order differenced data, we can see that the data appears more stationary now. The variance is more constant now, and the data more closely resembles white noise.



The sample ACF dies down quickly in a damped sine-wave pattern and cuts off after lag 24, indicating that the first order differenced data is now stationary.



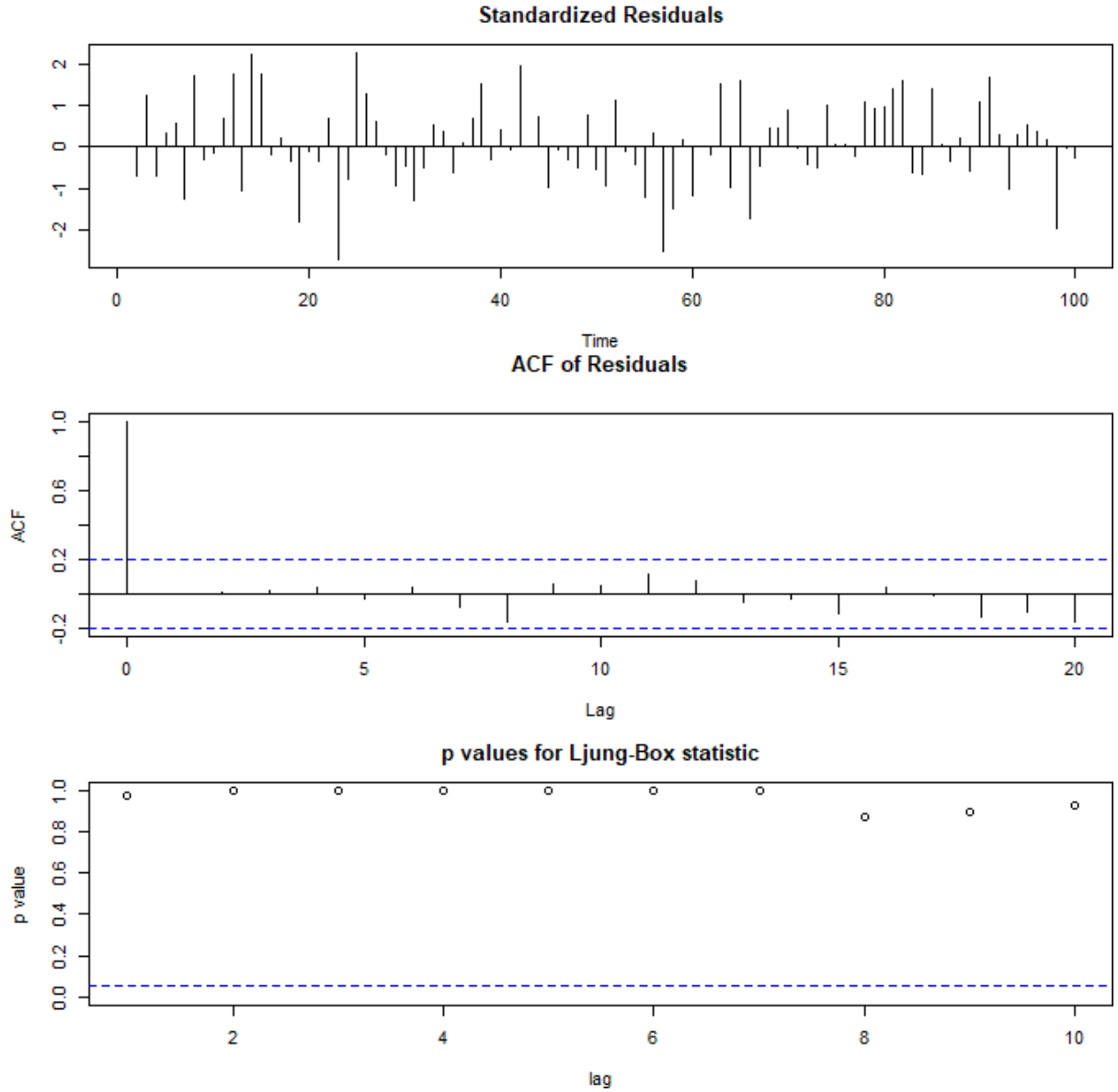
The sample PACF cuts off after lag 3. Between the SACF and SPACF, the SPACF cuts off more abruptly which suggests that an ARIMA(3,1,0) model would be more appropriate for the data.

Fit ARIMA(3,1,0) model

Using R's *stats* package, we fit a model using: `fit310 = arima(x, order = c(3,1,0))`

```
Call:
arima(x = x, order = c(3, 1, 0))

Coefficients:
          ar1      ar2     ar3
      1.1513  -0.6612  0.3407
s.e.  0.0950   0.1353  0.0941

sigma^2 estimated as 9.363:  log likelihood = -252,  aic = 509.99
[1] "AIC:  511.993984037159"
[1] "BIC:  522.374463437698"
```

Then we perform diagnostic testing on this model using: `ts.diag()`



The standardized residuals appear to be white noise, while the ACF of the residuals all lie within the blue boundaries indicating no significant correlation between serial observations. Finally, the p-values for the Ljung-Box test all lie significantly above the 0.05 threshold so we do not reject the null hypothesis that the residuals show no autocorrelation and accept that the ARIMA(3,1,0) model provides an adequate fit for our data.

It has been reported that the p-values plotted for the Ljung-Box statistic using `tsdiag()` from R's *stats* package are wrong when testing for autocorrelation between residuals of an estimated ARIMA model. When testing ARIMA models, `tsdiag()` uses the number of lags, h, for calculation of p-values where instead it should be $h - (p + q)$ to reflect the parameter estimation of an ARIMA(p, d, q) model.[1]

A possible workaround for this is to use the `sarima()` function from the *astsa* package, which outputs the plots for diagnostic testing too when fitting a model, using the correct degrees of freedom.

We thus fit another ARIMA(3,1,0) model using: `fit310 = sarima(x, 3, 1, 0)`

```
Call:
stats::arima(x = xdata, order = c(p, d, q),
             seasonal = list(order = c(P, D, Q), period = S),
             xreg = constant, transform.pars = trans, fixed = fixed,
             optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1      ar2     ar3  constant
      1.1460  -0.6593  0.3346    0.9799
s.e.  0.0954   0.1351  0.0947    1.6501

sigma^2 estimated as 9.336:  log likelihood = -251.83,  aic = 513.67

$degrees_of_freedom
[1] 95

$ttable
         Estimate     SE t.value p.value
ar1        1.1460 0.0954 12.0183  0.0000
ar2       -0.6593 0.1351 -4.8797  0.0000
ar3        0.3346 0.0947  3.5336  0.0006
constant   0.9799 1.6501  0.5938  0.5540

$AIC
[1] 5.188536

$AICc
[1] 5.192835

$BIC
[1] 5.319603
```

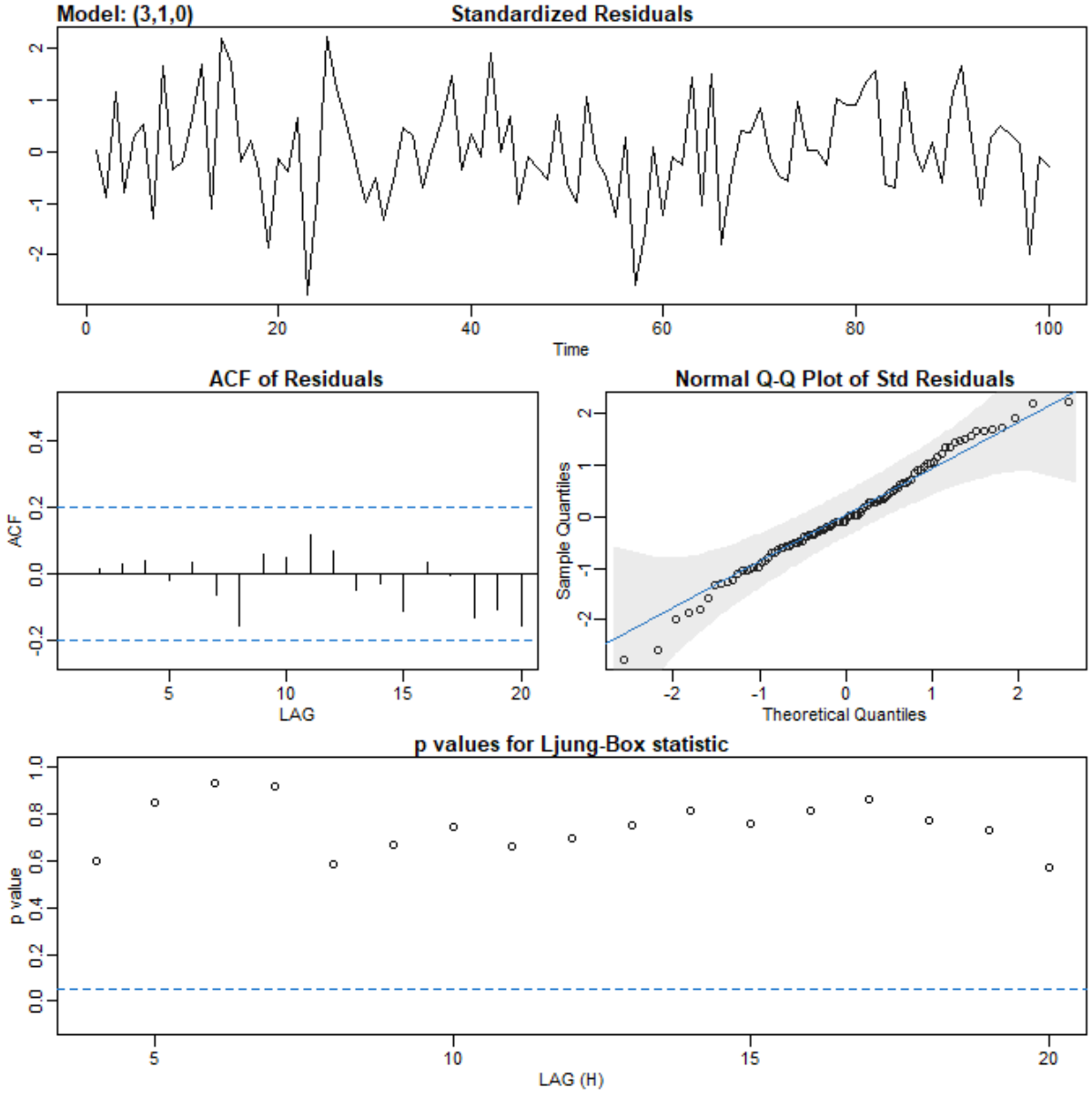
The `sarima()` function by default includes a constant term in the fitted model, which is equivalent to setting *include.constant* to *TRUE* for the `Arima()` function. We have verified that

[1] https://www.stat.pitt.edu/stoffer/tsa4/Rissues.htm#Issue4
https://stat.ethz.ch/pipermail/r-help/2008-August/170520.html
Shumway, R. H., & Stoffer, D. S. (2017). ARIMA Models in *Time series analysis and its applications: With R examples* (p. 140). Springer. Accessible at: https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf

doing so results in the same fitted model with the same coefficients. The constant term corresponds to the β_0 in our trend component and any ARIMA errors will also be rolled into the constant term when fitting. We will leave the setting as is because fitting a bias term here can provide a better fit.

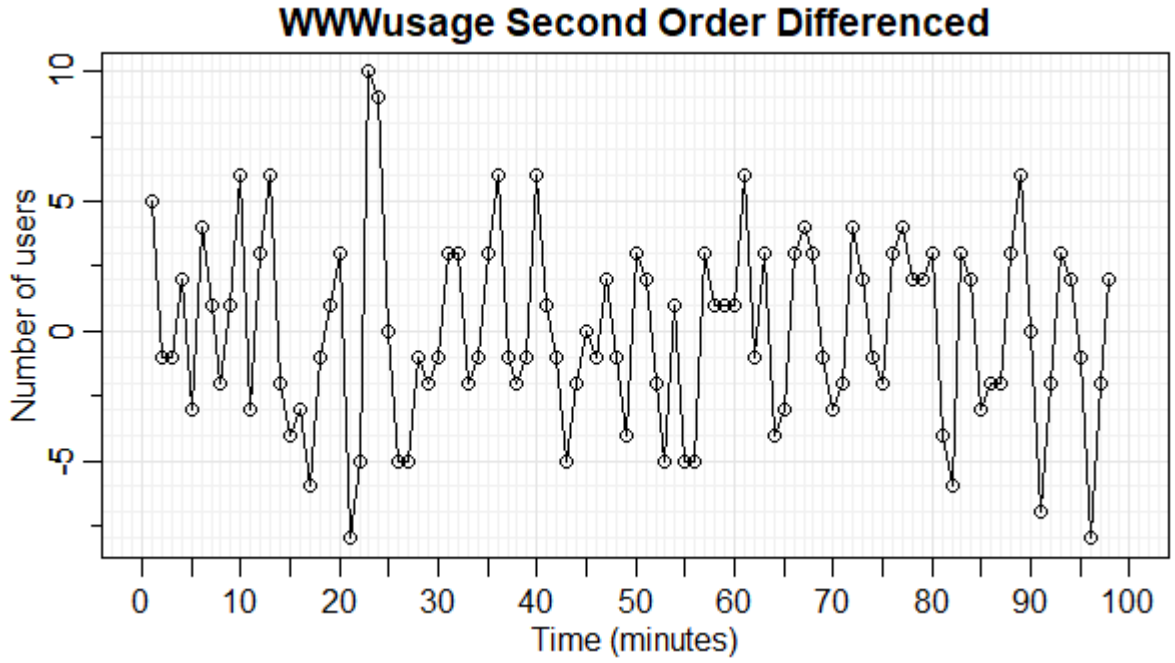
The final IC values that the `sarima()` function outputs are divided by the number of samples used to fit the model. This means that ARIMA models with higher differencing order will be penalized more as they lose one sample with each order of difference. By the parsimony principle, we should prefer the simplest solution possible, and these normalized IC values will be handy for comparing between ARIMA models of different order of differencing. We will thus use this value as our metric for choosing the best model of out all our adequately fitted models.

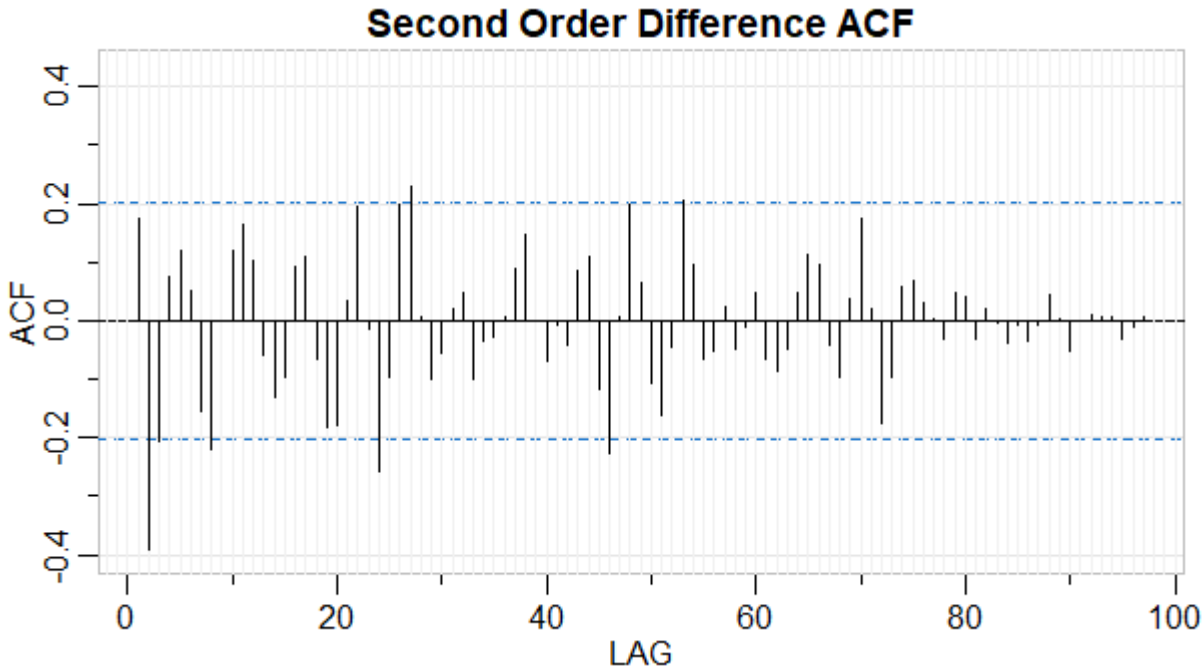
The `sarima()` diagnostic plots for the ARIMA(3,1,0) model are shown below. The conclusions remain the same, though the p-values demonstrate greater variance while there is an additional normal Q-Q plot for the standardized residuals. The relatively straight line indicates that the residuals of the fitted model follow a normal distribution and are thus likely white noise.



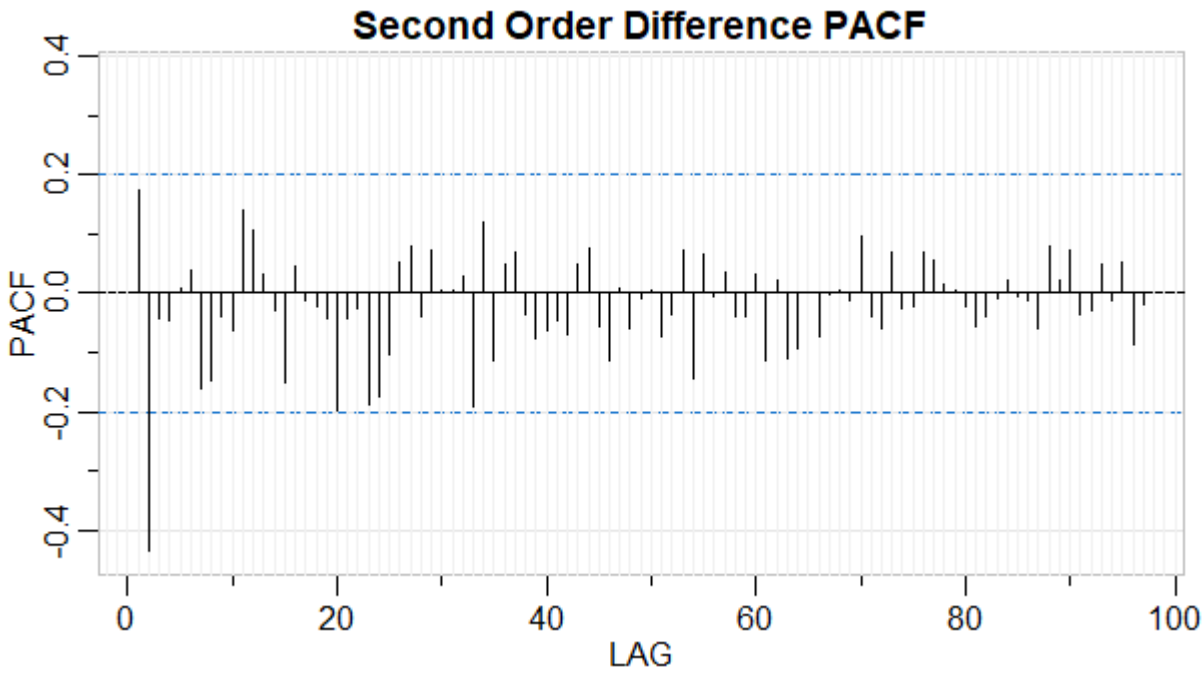
Second Order Differencing

Even though first order differencing already gave us a stationary series, we can try second order differencing just for the sake of exploration.





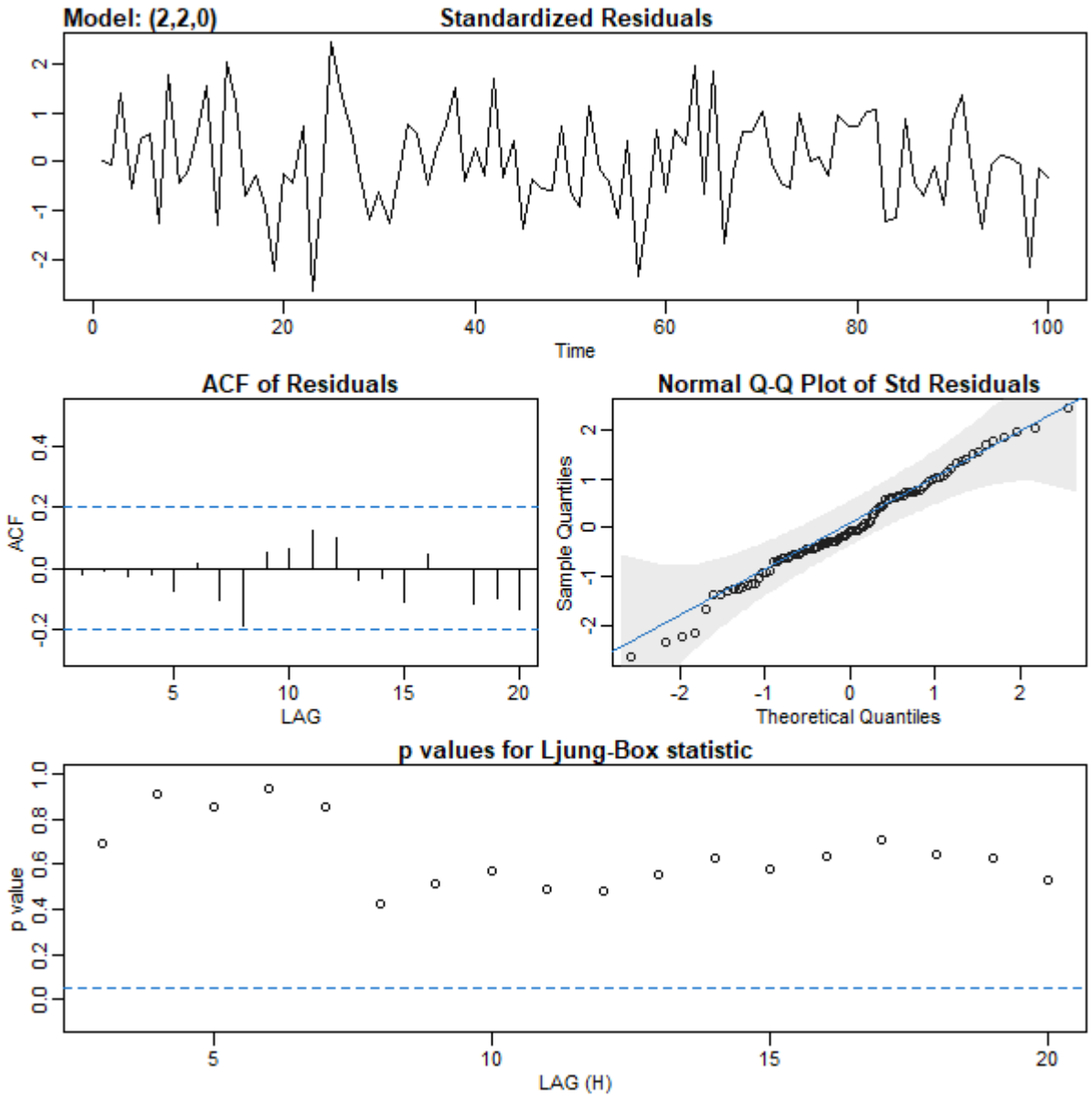
If we accept the presence of some outliers within the threshold of $100 \times 5\% = 5$, we can see that sample ACF cuts off after lag 3. Then, we see that there are outlying spikes up to lag 53. The sample ACF plot indicates that there is more stationarity in the twice differenced data.



Sample PACF cut off sharply after lag 2, which is earlier than sample ACF, so we can suggest a possible ARIMA(2,2,0) model.

Fit ARIMA(2,2,0) model

```
Call:
stats::arima(x = xdata, order = c(p, d, q),
             seasonal = list(order = c(P, D, Q), period = S),
             include.mean = !no.constant,
             transform.pars = trans, fixed = fixed,
             optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1      ar2
      0.2579  -0.4407
s.e.  0.0915   0.0906

sigma^2 estimated as 10.13:  log likelihood = -252.73,  aic = 511.46

$degrees_of_freedom
[1] 96

$ttable
    Estimate     SE t.value p.value
ar1   0.2579 0.0915  2.8173  0.0059
ar2  -0.4407 0.0906 -4.8637  0.0000

$AIC
[1] 5.219026

$AICc
[1] 5.220315

$BIC
[1] 5.298157
```



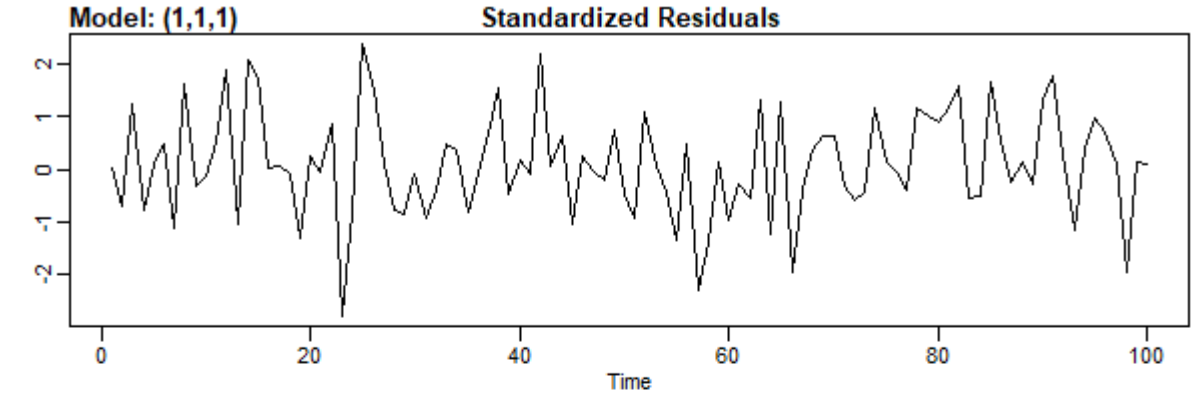
Diagnostic testing shows that model is adequate however its AIC = 5.219026 is higher than that of ARIMA(3,1,0), AIC = 5.188536, so we still prefer our first model.

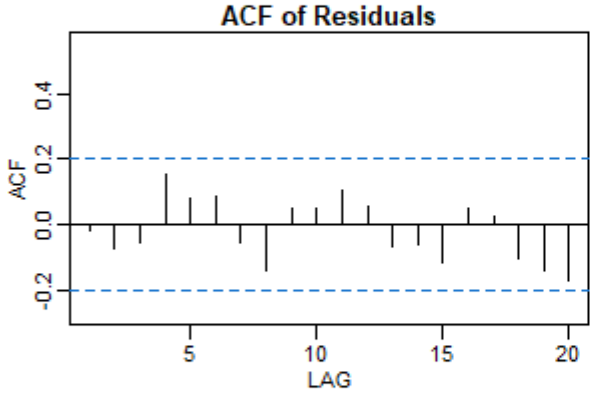
Auto ARIMA

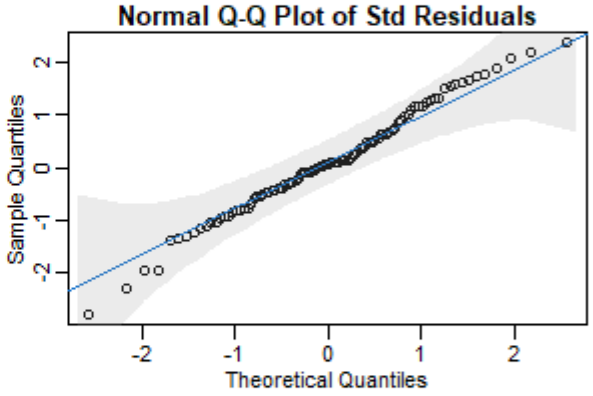
We can also try using the auto.arima() from the forecast package to try and return a best model according to AIC, AICc or BIC value. Auto ARIMA suggests an ARIMA(1,1,1) model.

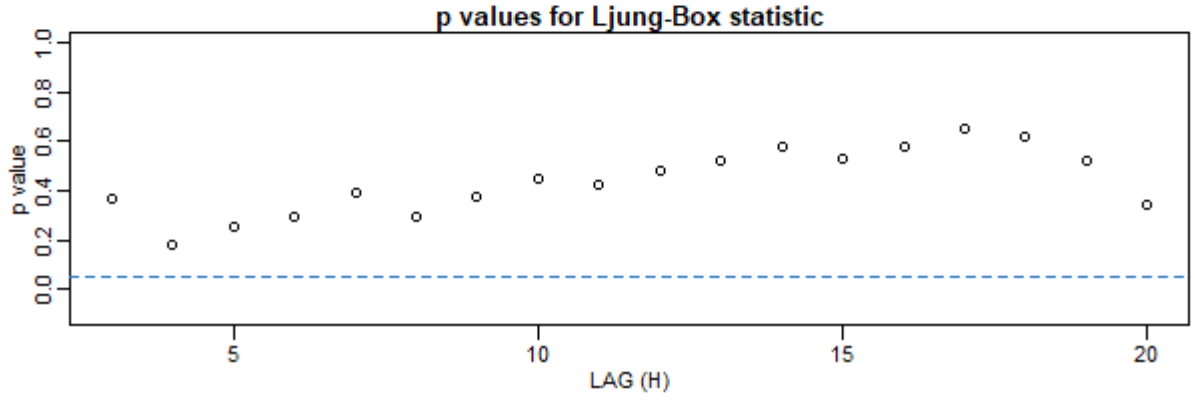
```
Call:
stats::arima(x = xdata, order = c(p, d, q),
             seasonal = list(order = c(P, D, Q), period = S),
             include.mean = !no.constant,
             transform.pars = trans, fixed = fixed,
             optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ar1      ma1
       0.6504   0.5256
s.e.   0.0842   0.0896
```

```
sigma^2 estimated as 9.793:  log likelihood = -254.15,  aic = 514.3

$degrees_of_freedom
[1] 97

$ttable
    Estimate     SE t.value p.value
ar1   0.6504 0.0842  7.7204       0
ma1   0.5256 0.0896  5.8690       0

$AIC
[1] 5.194944

$AICc
[1] 5.196207

$BIC
[1] 5.273584
```









Model is adequate but its AIC = 5.194944 is slightly higher than that of ARIMA(3,1,0), AIC = 5.188536, while p-values for Ljung-Box statistic are closer to the threshold than that of ARIMA(3,1,0). So ARIMA(3,1,0) remains our best model.

Extended ACF for mixed ARMA(p, q) models

We can also try using an Extended ACF matrix [2] to choose both p and q for mixed ARIMA models.

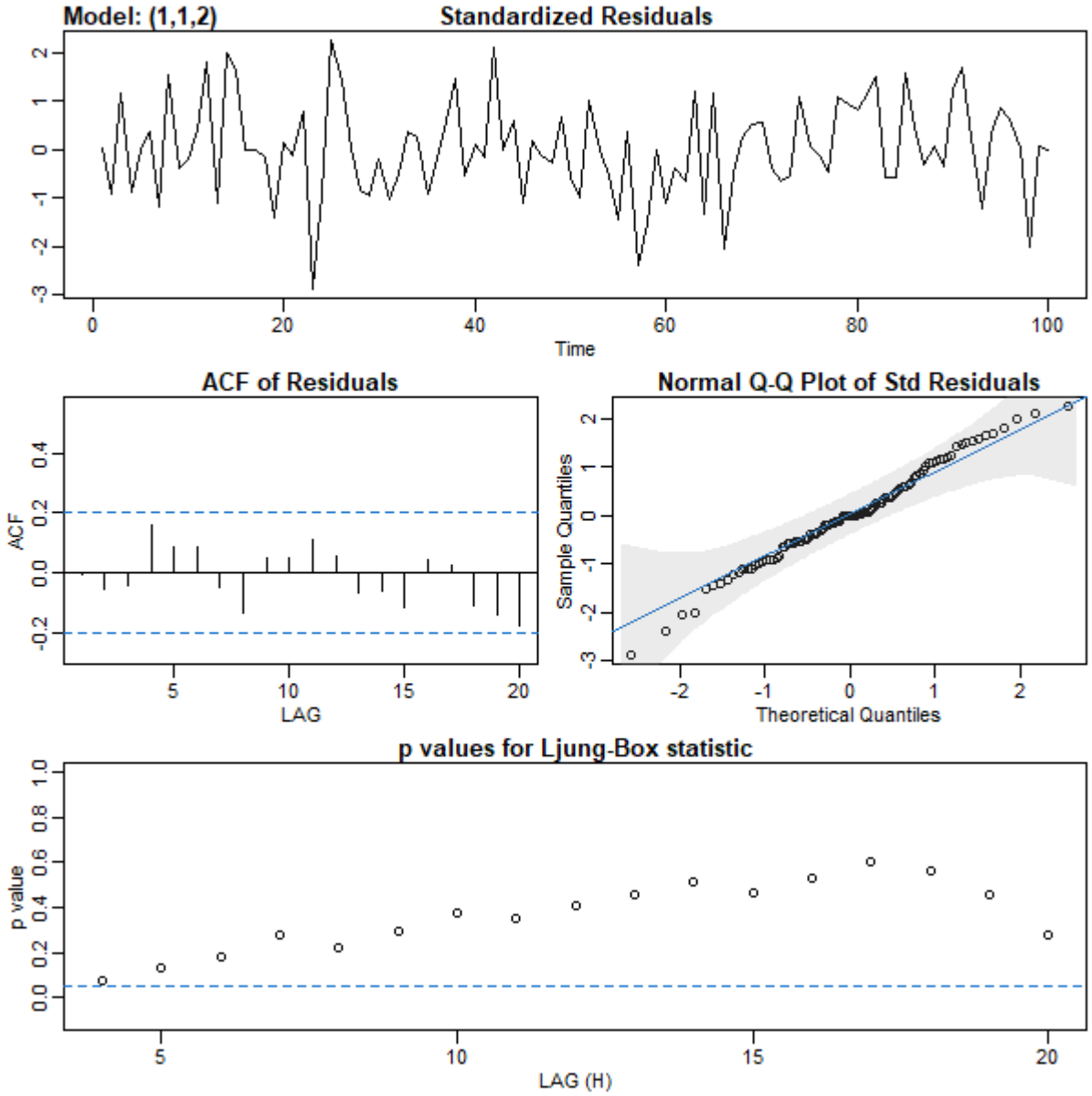
```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x o o o o o  o  o  o
1 x x o o o o o o o o o  o  o  o
2 x x x o o o o o o o o  o  o  o
3 o o o o o o o o o o o  o  o  o
4 x o o x o o o o o o o  o  o  o
5 x o o x o o o o o o o  o  o  o
6 x o o x o o o o o o o  o  o  o
7 x x x x o o o o o o o  o  o  o
```

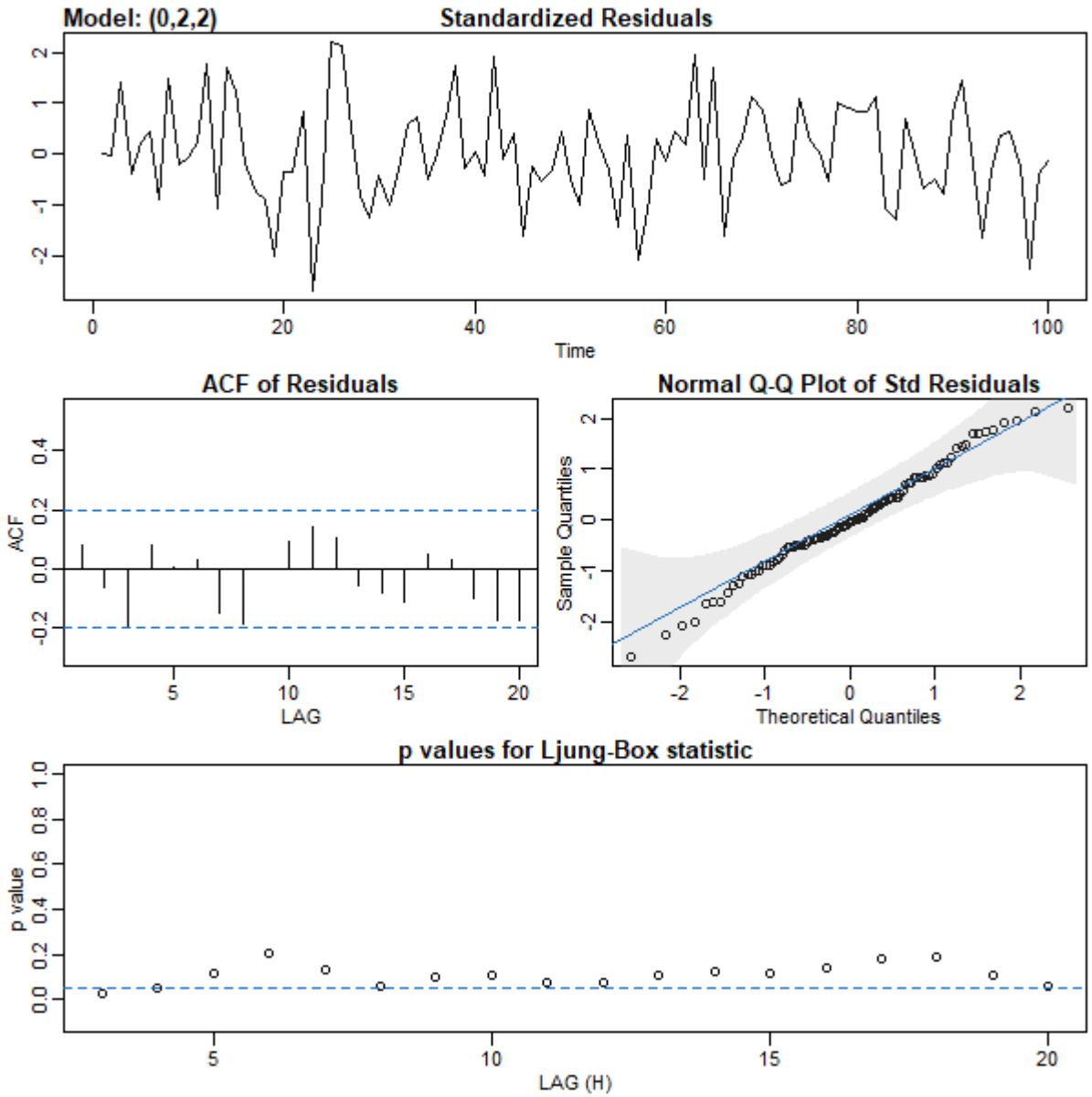
The above is the EACF matrix for the first order differenced data. Starting from the bottom-right and moving towards the upper-left, we try to find the corners of triangular patterns in the EACF matrix. EACF on first order differenced data suggests ARIMA(1,1,2) as a possible model.

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 o x o o o o o x o o o  o  o  o
1 x x x o o o o x o o o  o  o  o
2 o o o o o o o o o o o  o  o  o
3 x o o o x o o o o o o  o  o  o
4 o o o o x o o o o o o  o  o  o
5 o o o o x o o o o o o  o  o  o
6 x o o o o o o o o o o  o  o  o
7 x x x x o o o o o o o  o  o  o
```

EACF on second order differenced data suggests possible ARIMA(0,2,2) and ARIMA(2,2,0) models, the latter of which matches our suggestion based on the sample PACF.

We have already tried fitting an ARIMA(2,2,0) model previously, so we will try the remaining models. For brevity, we will only be showing the diagnostic testing plots for the ARIMA(1,1,2) and ARIMA(0,2,2) models.

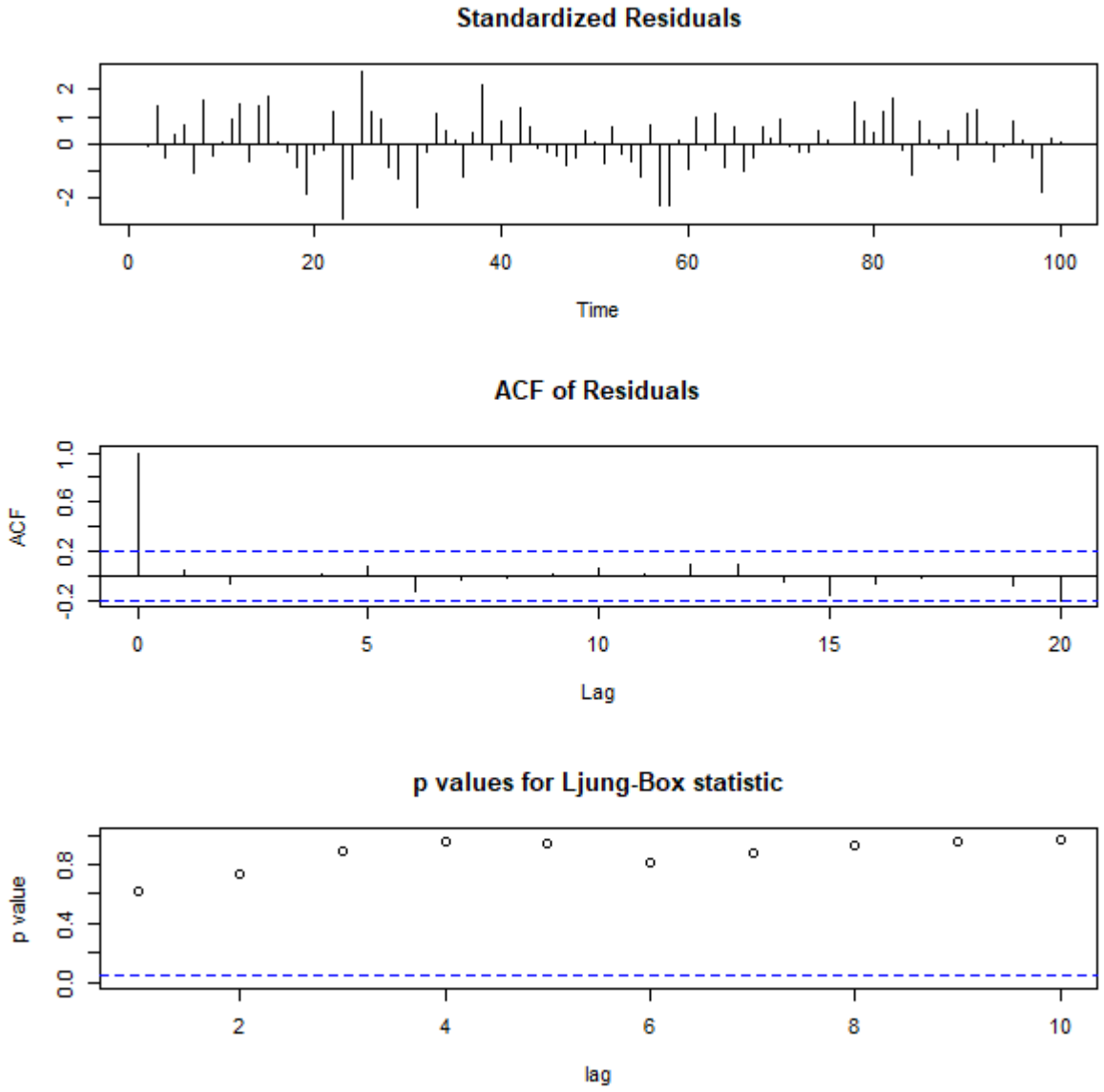

Model: (1,1,2)
Standardized Residuals
Time
ACF of Residuals
ACF
LAG
Normal Q-Q Plot of Std Residuals
Sample Quantiles
Theoretical Quantiles
p values for Ljung-Box statistic
p value
LAG (H)



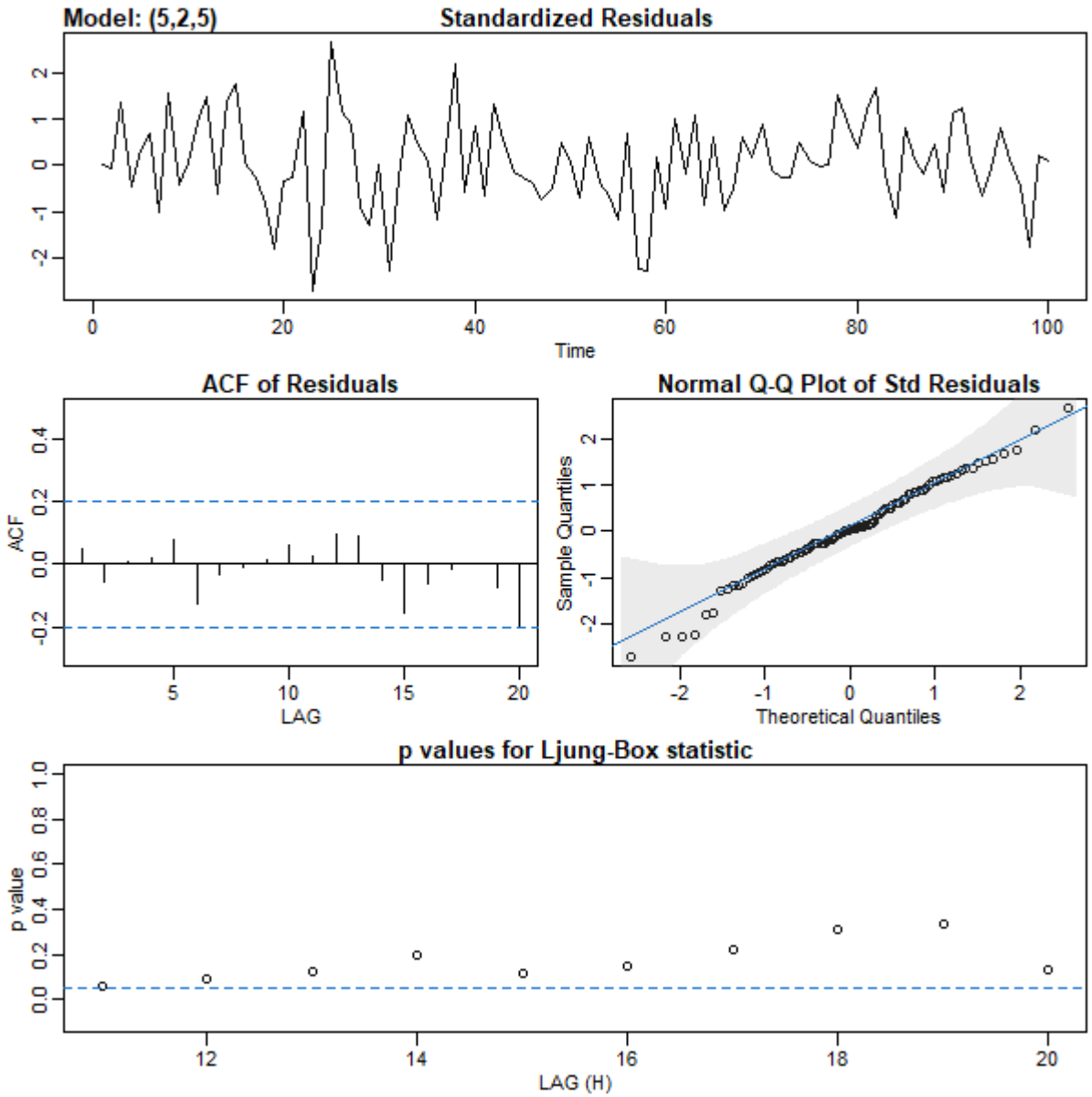
Both models have p-values for Ljung-Box statistic falling below the 0.05 threshold, which means that the residuals show some autocorrelation, so we reject the null hypothesis that the models are adequate for our data and ignore the AIC values, though the values are still higher than our best model.

ARIMA(5,2,5), p-value, normalized AIC

Using arima() and tsdiag(), the ARIMA(5,2,5) model actually gives the lowest unnormalized AIC out of all our models and the Ljung-Box test also deems it an adequate model.



The unnormalized AIC of this model is 509.8135, while the unnormalized AIC of the ARIMA(3,1,0) model is 513.6651.



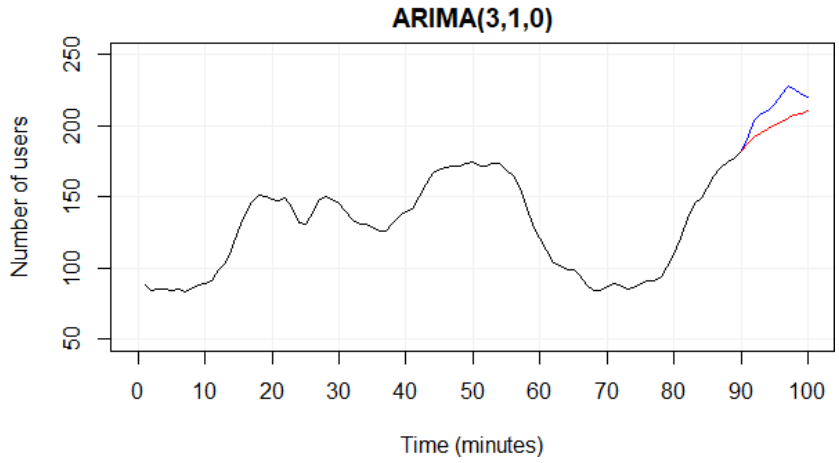
However, the normalized AIC from sarima() using the smaller sample count for second order differenced data is actually higher than the normalized AIC for ARIMA(3,1,0). This fits the parsimony principle that more complex differenced model should be less preferred. Furthermore, the p-value for Ljung-Box test using the adjusted degrees of freedom also deems it inadequate so we can actually ignore the low AIC even before. This again shows why we should use the normalized AIC value using sample counts and also to adjust the degrees of freedom for estimated ARIMA(p, d, q) models in the Ljung-Box statistic.

Forecasting Error

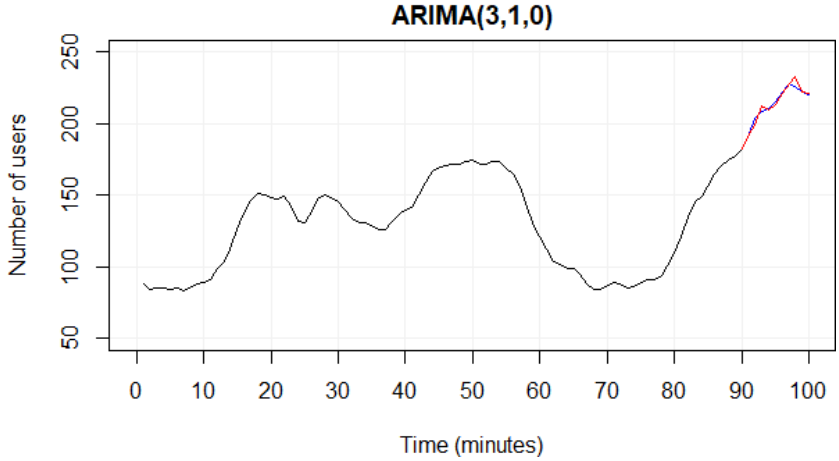
To measure the performance of our model, we can try and measure its forecasting error using a test set. We demonstrate two methods of testing for forecasting error. For brevity sake, we only measure it for ARIMA(3,1,0) since we have determined it as our best model and there are no close contenders. Else, if there are other similar models, we would compare the mean absolute errors and mean square errors and choose the one with the smallest errors. MSE penalizes residuals more than MAE.

Train-Test Split

For the first method, we can split the data into a 90:10 ratio to check for the forecasting error of our model.



The blue line shows the ground-truth values for 90 to 100 minutes. The red line shows our forecasted values after training on data from 0 to 90 minutes. The root mean square error, RMSE, is 14.85175 while MAE is 14.03889. Error is fairly high here as sequentially forecasted values are based off of prior forecasted values, and any errors in earlier forecasting will snowball into later values.

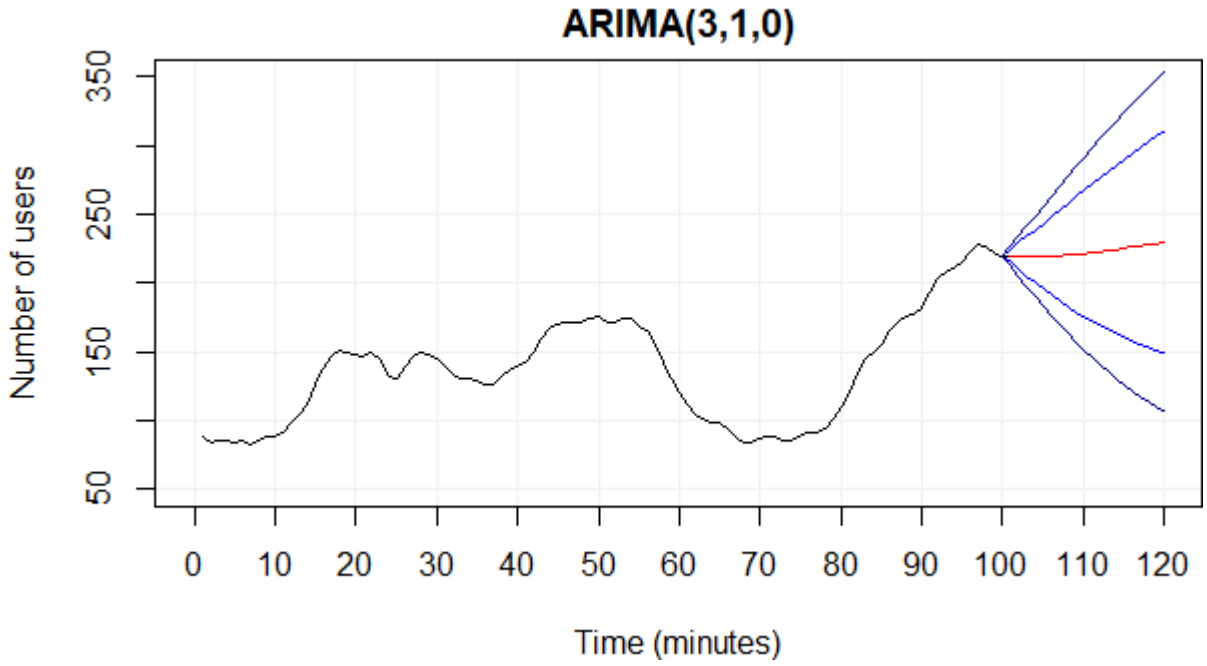


The above plot shows one-step forecasted values using ground-truth values at each step from 90 to 100 minutes. Errors are lesser here because we are using the ground truth test values for one-step forecasting. This is done by fitting a new model to the test data specifying the previous model trained on training data. In so doing, the coefficients are frozen and the Arima() function instead calculates the fitted values using the test data and frozen parameters.

Rolling cross-validation

Because the time series is chronologically dependent, we have to make a modification to the K-fold cross-validation process and discard subsequent samples after the test sets for some folds. The result is a rolling cross-validation process where the training set size gradually increases with each fold and our forecasting window rolls forward. For brevity, we will not be putting the full forecast error matrix here, but the root mean square error obtained is 3.213733 using a rolling window of 1.

Forecasting



This plot shows the forecasted values for 100 to 120 minutes using a model trained on data from the 0 to 100 minutes. Red line indicates the forecasted data after 100 mins. Dark blue lines indicate the 95% upper and lower bound while blue lines indicate the 80% upper and lower bound.

R Code

```
library(TSA)
library(astsa)
library(forecast)

x = scan("wwwusage.txt", skip = 1, quiet = T)
par(mar=c(4,4,2,0))
plot(x,
     main = "WWWusage Time Series Data",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "l",
     ylim = c(50,250),
     xlim = c(0,100)
    )

par(mar=c(4,4,2,0))
acf(x,
    lag.max = 100,
    xlim = c(0,100),
    main = "WWWUsage ACF")

par(mar=c(4,4,4,0))
pacf(x,
     lag.max = 100,
     xlim = c(0,100),
     main = "WWWUsage PCF")

x_d1 = diff(x, lag = 1)
par(mar=c(4,4,2,0))
plot(x_d1,
     main = "WWWusage First Order Differenced",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "o",
     xlim = c(0,100),
    )

par(mar=c(4,4,2,0))
acf(x_d1,
    lag.max = 100,
    xlim = c(0,100),
    main = "First Order Differenced ACF",
    panel.first = abline(v = seq(-10,110,1), col = "grey95")
    )

par(mar=c(4,4,4,0))
pacf(x_d1,
    lag.max = 100,
    xlim = c(0,100),
    main = "First Order Differenced PACF",
    panel.first = abline(v = seq(-10,110,1), col = "grey95")
    )

fit310 = arima(x, order = c(3,1,0))
fit310;paste("AIC: ", AIC(fit310));paste("BIC: ", BIC(fit310))

par(mar=c(4,4,3,0))
tsdiag(fit310)

fit310 = sarima(x, 3, 1, 0)
fit310

fit310 = Arima(x, order = c(3,1,0), include.constant = T)
fit310;paste("AIC: ", AIC(fit310));paste("BIC: ", BIC(fit310))

x_d2 = diff(x_d1, lag = 1)
par(mar=c(4,4,2,0))
tsplot(x_d2,
     main = "WWWusage Second Order Differenced",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "o",
     xaxt = "n",
     xlim = c(0,100),
     panel.first = abline(h = seq(-20,20,1), v = seq(-150,150,1), col = "gray95")
    )
axis(1, at=seq(-10,110,5), labels=c(NA,NA,0,NA,10,NA,20,NA,30,NA,40,NA,50,NA,60,NA,70,NA,80,NA,90,NA,100,NA,NA))
axis(2, at=seq(-20,20,5), labels=seq(-20,20,5))
axis(3, at=seq(-10,110,120))
axis(4, at=seq(-20,20,40))

par(mar=c(4,4,0,0))
acf1(x_d2,
    max.lag = 97,
    xlim = c(0,100),
    main = "Second Order Difference ACF",
    panel.first = abline(v = seq(-10,110,1), col = "grey95")
```

```
    )

par(mar=c(4,4,0,1))
acf1(x_d2,
    pacf = T,
    max.lag = 97,
    xlim = c(0,100),
    main = "Second Order Difference PACF",
    panel.first = abline(v = seq(-10,110,1), col = "grey95")
    )

fit220 = sarima(x, 2, 2, 0)
fit220

fitauto = auto.arima(x)
fitauto;paste("AIC: ", AIC(fitauto));paste("BIC: ", BIC(fitauto))
fitauto = sarima(x, 1, 1, 1, no.constant = T)
fitauto

eacf(x_d1)
eacf(x_d2)

fit112 = sarima(x, 1, 1, 2)
fit112

fit022 = sarima(x, 0, 2, 2)
fit022

fit525 = sarima(x, 5, 2, 5)
fit525
AIC(fit525$fit)
par(mar=c(4,4,3,0))
tsdiag(fit525$fit)

par(mar = c(4,4,2,0))
ar310_train = Arima(x[1:90], order = c(3,1,0), include.constant = T)
pred_1 = forecast(ar310_train)
plot(c(x[1:90]), # 10-ahead forecasting with only historical data up to 90 mins
     main = "ARIMA(3,1,0)",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "l",
     xaxt = "n",
     xlim = c(0,100),
     ylim = c(50,250),
     panel.first = abline(h = seq(0,250,50), v = seq(0,120,10), col = "gray95")
    )
axis(1, at=seq(0,120,10), labels=seq(0,120,10)); lines(90:100, x[90:100], col="blue"); lines(90:100, c(x[90], pred_1$mean), col="red")
accuracy(pred_1$mean, x[91:100])

par(mar = c(4,4,2,0))
ar310_test = Arima(x[91:100], model = ar310_train)
pred_2 = forecast(ar310_test)
plot(c(x[1:90]), # One-step forecasting using ground truth test values
     main = "ARIMA(3,1,0)",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "l",
     xaxt = "n",
     xlim = c(0,100),
     ylim = c(50,250),
     panel.first = abline(h = seq(0,250,50), v = seq(0,120,10), col = "gray95")
    )
axis(1, at=seq(0,120,10), labels=seq(0,120,10)); lines(90:100, x[90:100], col="blue"); lines(90:100, c(x[90],pred_2$fitted), col="red")
#sqrt(mean(pred_2$residuals^2, na.rm=TRUE))
accuracy(pred_2$fitted, x[91:100])

ar310 = function(x, h) {forecast(Arima(x, order=c(3,1,0)), h = h)}
e = tsCV(x, ar310, h = 1)
e
sqrt(mean(e^2, na.rm=TRUE))

par(mar = c(4,4,2,0))
ar310 = Arima(x, order = c(3,1,0), include.constant = T)
pred_1 = forecast(ar310, h = 20)
plot(c(x, pred_1$mean),
     main = "ARIMA(3,1,0)",
     ylab = "Number of users",
     xlab = "Time (minutes)",
     type = "l",
     xaxt = "n",
     xlim = c(0,120),
     ylim = c(50,350),
     panel.first = abline(h = seq(0,250,50), v = seq(0,120,10), col = "gray95")
    )
axis(1, at=seq(0,120,10), labels=seq(0,120,10));lines(101:120, pred_1$mean, col="red")
lines(100:120, c(x[100],pred_1$upper[,1]), col="blue") # 80% Upper Bound
lines(100:120, c(x[100],pred_1$upper[,2]), col="darkblue") # 95% Upper Bound
lines(100:120, c(x[100],pred_1$lower[,1]), col="blue") # 80% Lower Bound
lines(100:120, c(x[100],pred_1$lower[,2]), col="darkblue") # 95% Lower Bound
```

References

1 Makridakis, S.G., Wheelwright, S.C., and Hyndman, R.J.: ‘Forecasting : Methods and Applications’ (Wiley, 1998, 3rd edn. 1998)

2 Tsay, R., and Tiao, G.: ‘Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation Function for Stationary and Nonstationary ARMA Models’, Journal of the American Statistical Association, 1984, 79, (385), pp. 84-96