MAS8382 ~ Time Series Data Project

Introduction

The data for this project is a time series that represent monthly sales of a product over ten years, from January 2011 to December 2020. The aim of the project is to find a time series model, that is suitable for the data and can be used for forecasting. In order to assess the predictive accuracy of each model fitted, the values for the tenth year will be forecasted, and the forecasts will be compared to the actual observed values during the tenth year. In order to get a better idea for the data it is essential to plot them. Before that, the data should be loaded and the time series needs to be created (Appendix 1.1). The plotted time series can be found below:

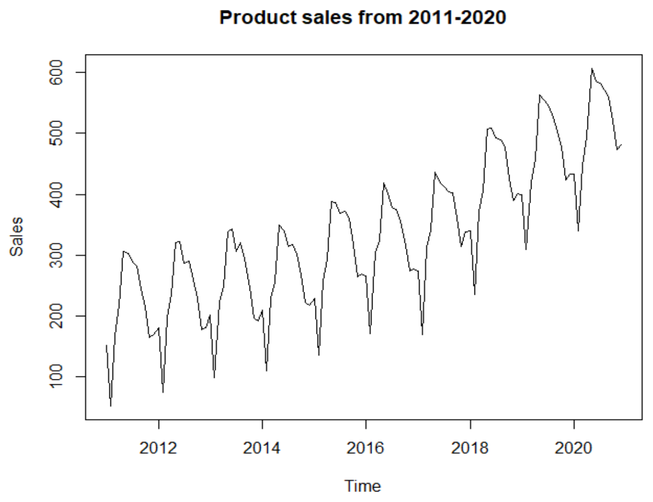


Figure 1: Plot of the time series

From the plot it is clear, that there is an evident seasonal effect. An upwards liner trend is also observed as it seems that the sales have increased during the years. So, the process does not appear to be stationary models appropriate for stationary processes cannot be used in this case. For this project, three models will be fitted and assessed.

1. SARIMA model

As mentioned above, there is a time varying mean, and the process is not stationary. However, the time series display some homogeneity, because it seems that the only thing changing with time, is the mean. Moreover, the seasonal effect is also evident. So, an appropriate model for this data could be a seasonal arima (SARIMA) model. The first thing to do now is choose values for p, d, q and p*, d*, and q*. In order to do that the autocorrelation and partial autocorrelation plots are plotted (Appendix 1.2):

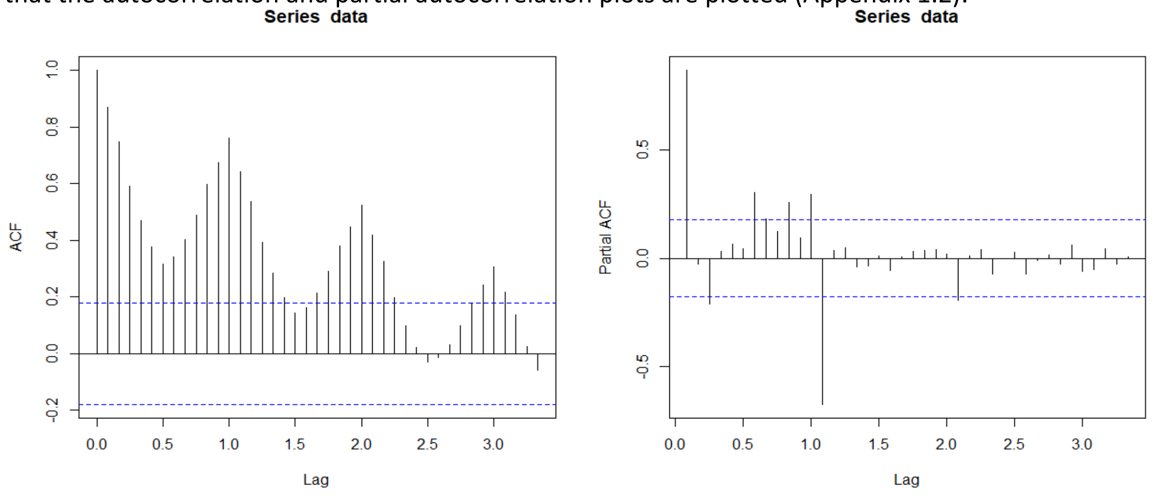


Figure 2: Acf and pacf plots before any differencing

From the autocorrelation plot it appears that the acf's decay to zero but not very fast, which confirms non stationarity. There are also some spikes at lag 1 (12 months) lag two (24 months) and so, confirming seasonality. To tackle the issue of non-stationarity, the seasonal differences are computed (Appendix 1.2). Then the data are plotted again along with the acf and pacf plots:

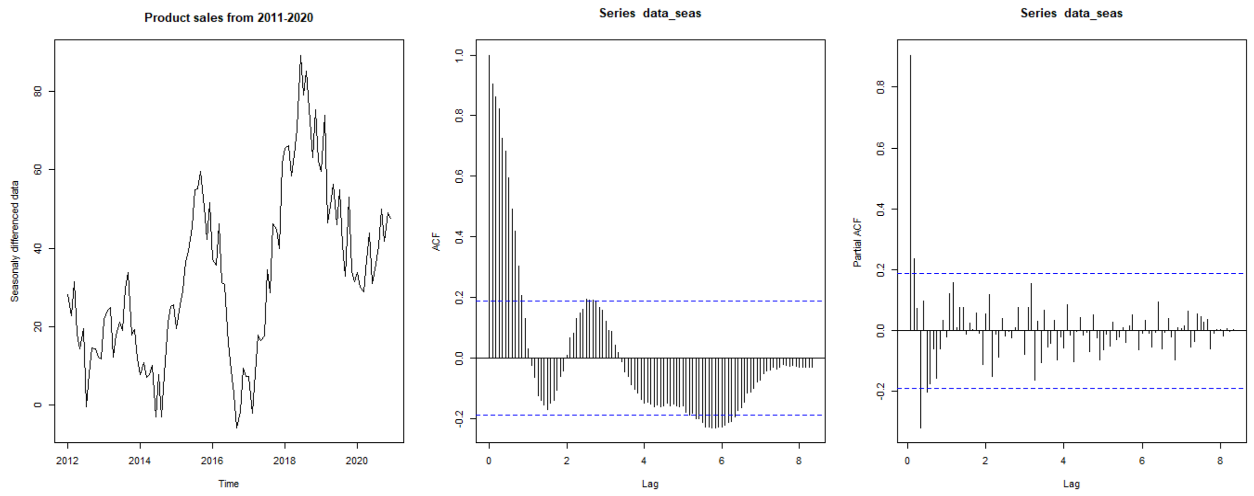


Figure 3: Time series, acf and pacf plots after differencing seasonal once

Again, there are clear evidence of non-stationarity, and the linear trend is still present. The acf plot decays very slowly to zero, giving more evidence of non-stationarity. The next step is to difference the data again, but this time non-seasonally (Appendix 1.2). The corresponding plots can be found below:

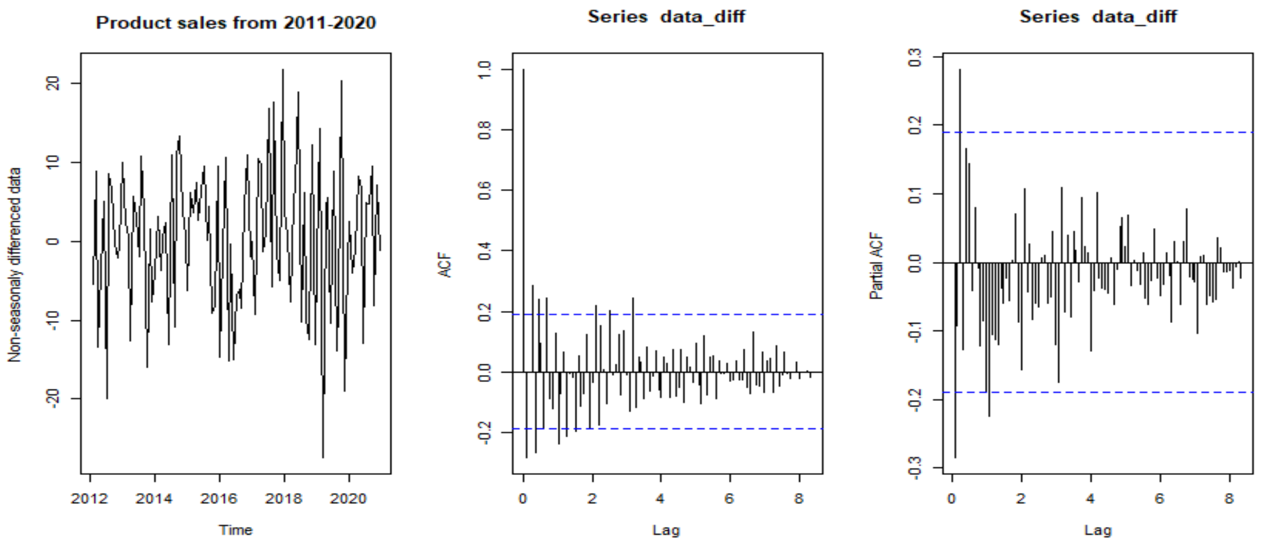


Figure 4: Time series, acf and pacf plots after seasonal and non-seasonal differencing

This time, there is no evidence of non-stationarity. Since there was one seasonal and one non seasonal differencing, the value for d and d^* is chosen to be 1. Now from the acf plot, looking at the non-seasonal acfs, they go to zero after lag 1, so q is chosen to be 1. Similarly, looking at the seasonal acfs, it seems that they go to zero after lag 1, so q^* is also chosen to be one. Now looking at the non-seasonal pacf's they go to zero after lag 0, so p is chosen to be 0. Similarly, the non-seasonal pacfs cut off after lag 0, so p^* is also chosen to be zero. So the model is ARIMA (0,1,1) x $(0,1,1)_{12}$. Now in order to check if the model can be improved underfitting and overfitting is performed (Appendix 1.3). The results are summarized below:

Model	Log L	P value (compared to mod)
Mod : $(0,1,1) \times (0,1,1)_{12}$	-381.45	.
$(1,1,1) \times (0,1,1)_{12}$	-381.24	0.518
$(0,1,1) \times (1,1,1)_{12}$	-380.4	0.147
$(0,1,2) \times (0,1,1)_{12}$	-381.02	0.352
$(0,1,1) \times (0,1,2)_{12}$	-380.82	0.262

When over fitting, none of the p values is significant, so the original simpler model is preferred. Next underfitting gives the following results:

Model	Log L	P value (compared to mod)
Mod: $(0,1,1) \times (0,1,1)_{12}$	-381.45	.
$(0,1,0) \times (0,1,1)_{12}$	-387.22	0.00068
$(0,1,1) \times (0,1,0)_{12}$	-387	0.00087

When under fitting all of the p values are very significant so the null hypothesis of the simpler model is rejected, and again model 1 is selected. Now in order to check that the chosen model is appropriate the following plots were generated (Appendix 1.4):

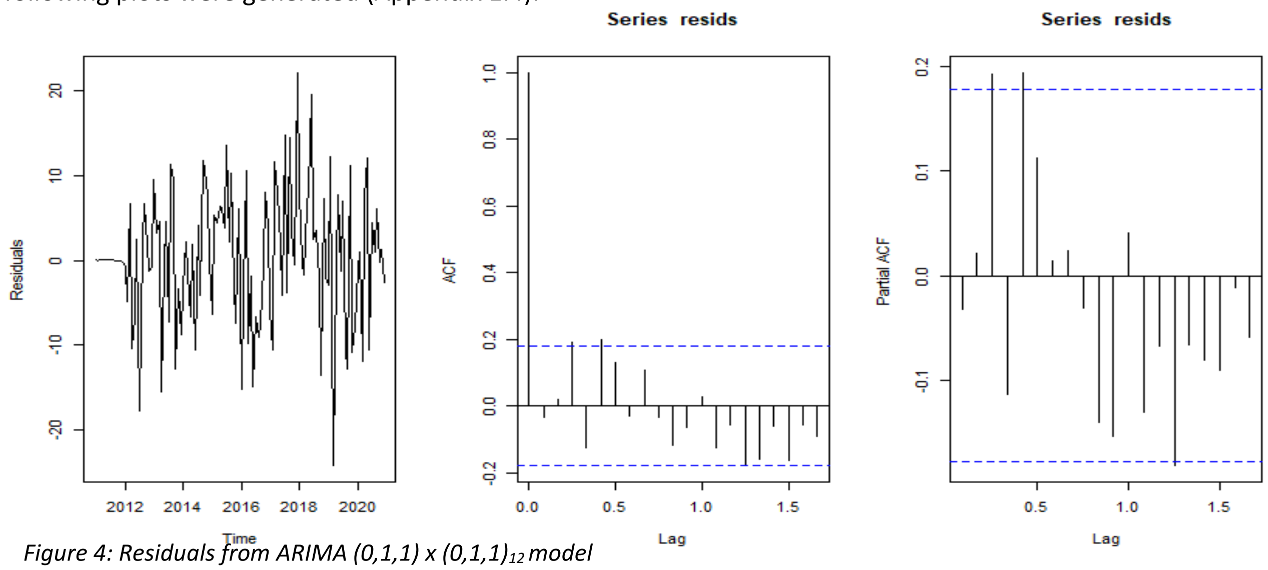


Figure 4: Residuals from ARIMA $(0,1,1) \times (0,1,1)_{12}$ model

The residual plots seem okay, so the chosen model can be used for forecasting. Forecasts for the next 6 months are generated (values in Appendix 1.4.1), along with 95% forecast prediction intervals, and in order to check the predictive performance of the model, the model is fitted to the first nine years, and then it used for forecasting the values for the 10nth year (Appendix 1.4):

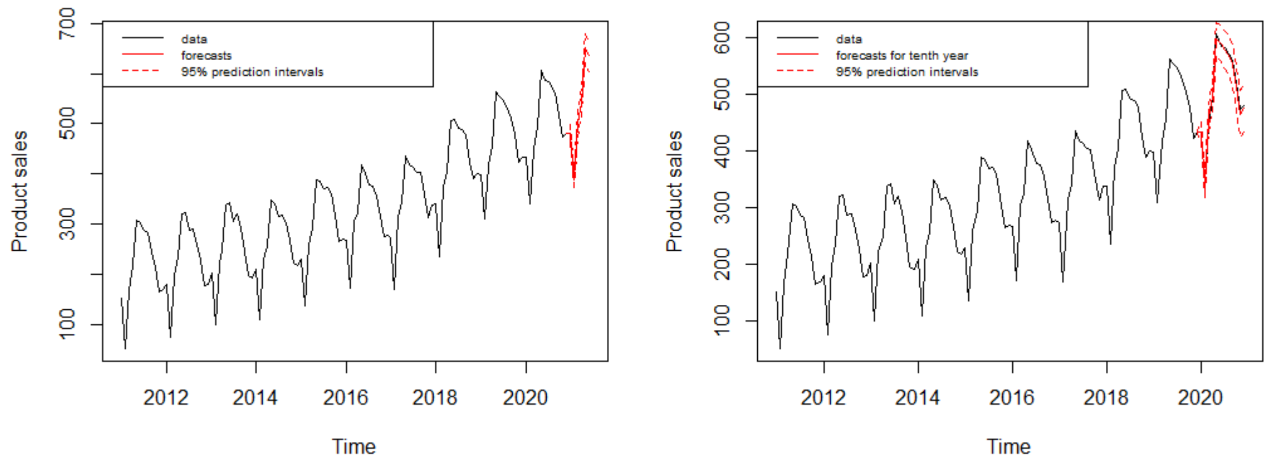


Figure 5: Six-month forecasts (left) and tenth year forecasts (right) with 95% prediction intervals

The intervals are quite narrow for the 6 months forecast and as expected the sales seem to follow a similar trend to the sales up to that point. Since the prediction is short term, the ARIMA model seems appropriate. The predictions for the tenth are very close to the actual data and the MSE is 38.89, so this model is quite appropriate for this specific time series.

2. Time Series Regression model

Another way, of handling a time varying mean, is to model it through a time series regression model. After fitting the model, the following plots are generated (Appendix 1.5):

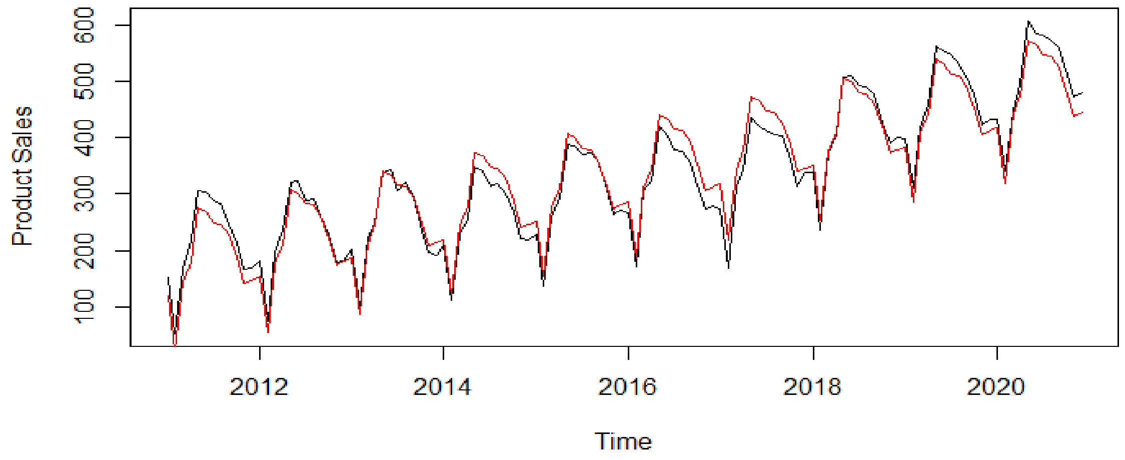


Figure 6: Plot of fitted values for the time regression model

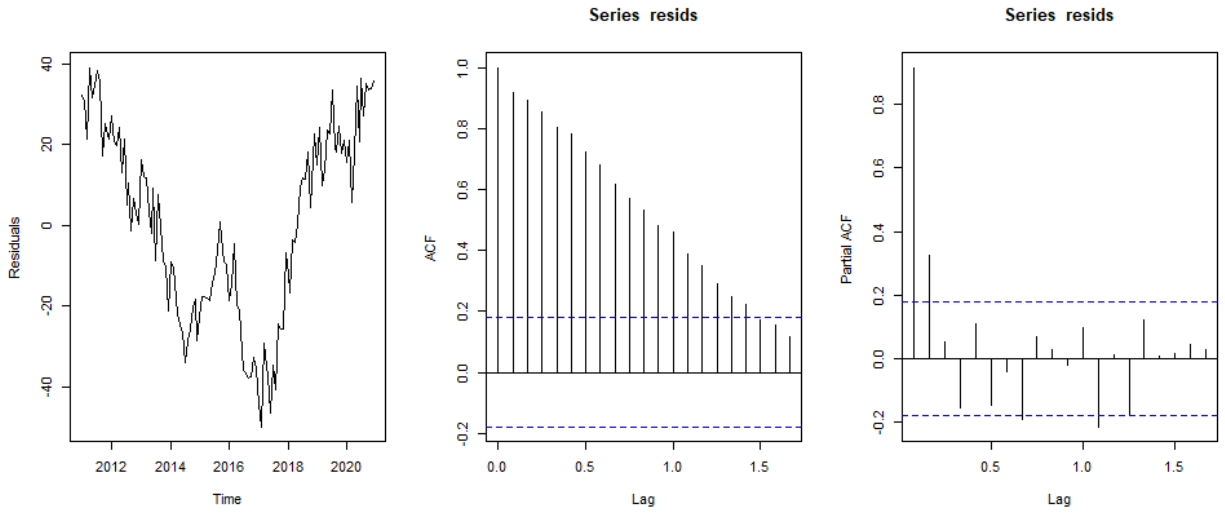


Figure 7: Plots for residuals the time regression model

From figure 6, it seems that the fitted values, are not super close to the actual values, so this is an indication that the predictions, generated might not be that accurate. Moreover, from figure 7 it is clear that the residuals seem to be autocorrelated and not independent. The acf seems to gradually tail of to 0, and pacf seems to cut off after lag 1. So, an AR (1) model can be fitted for the residuals (Appendix 1.6). Just like before overfitting and underfitting is performed:

Model	Log L	P value (when compared to mod_res)
Mod_res : (1,0,0)	-427.84	.
(2,0,0)	-416.62	$2.16*10^{-6}$

The p-value is very significant so the extra AR parameter is added in the model. Now again over fitting is performed:

Model	Log L	P value (when compared to mod_res_over)
Mod_res_over: (2,0,0)	-416.62	.
(3,0,0)	-416.2	0.361
(2,0,1)	-416.36	0.467

Now none of the p values are significant, so the model for the residuals is an AR (2) model. To check that the residuals behave as they should, the following plots are generated (Appendix 1.6):

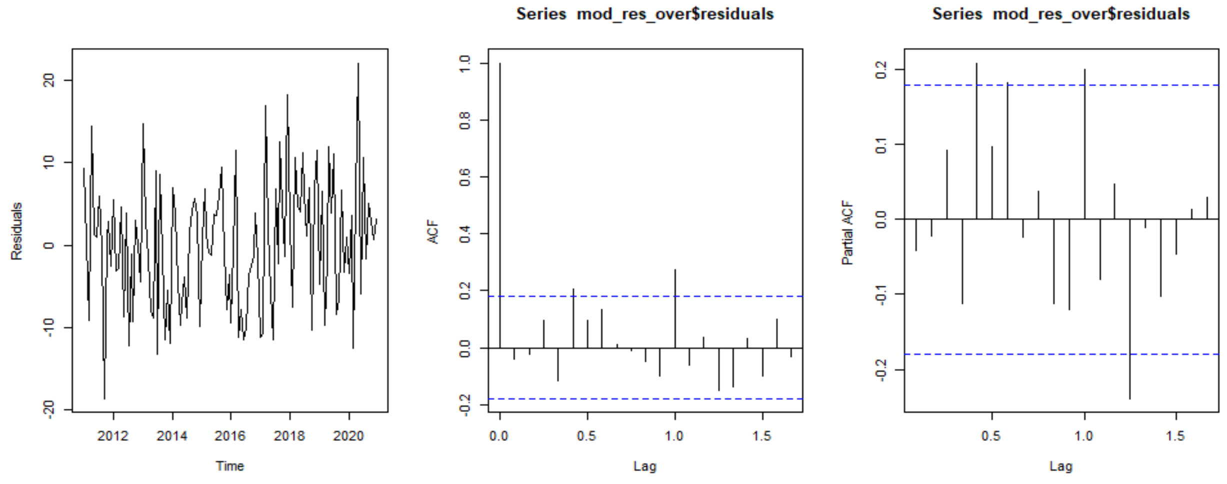


Figure 8: Plots of residuals, acfs and pacfs for the AR(2) residuals model

Now the residuals seem stationary, and the acf and pacf plots seem okay, so forecasts for the next six months (values in Appendix 1.7.1) and for the tenth year can be generated, like before (Appendix 1.7):

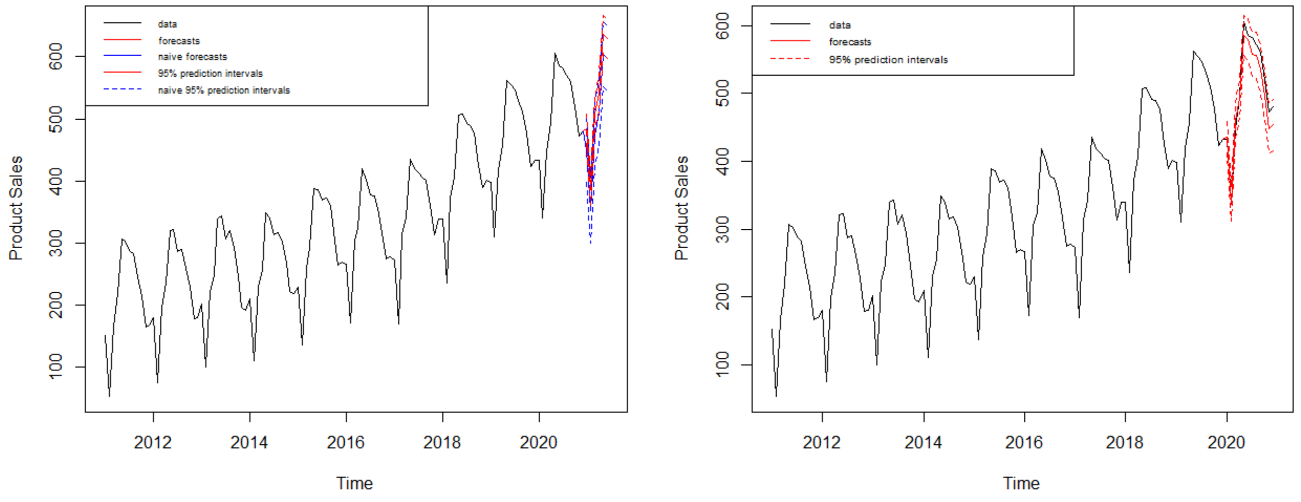


Figure 9: Six-month forecasts (left) and tenth year forecasts (right) with 95% prediction intervals

The left plot (bigger size image on Appendix 1.7.2) is a bit crowded, so it is not very easy to draw conclusions. However, form the right plot it is clear that the predictions for the tenth year are not that close to the actual data, and the MSE is 327.16 which is much larger when compared to the arima model. So, the time regression model might not be the most appropriate for these data.

3. Quadratic Time Series Regression model

Looking back at figure 1, it seems like there could be a quadratic trend. So, the quadratic model is fitted (Appendix 1.8):

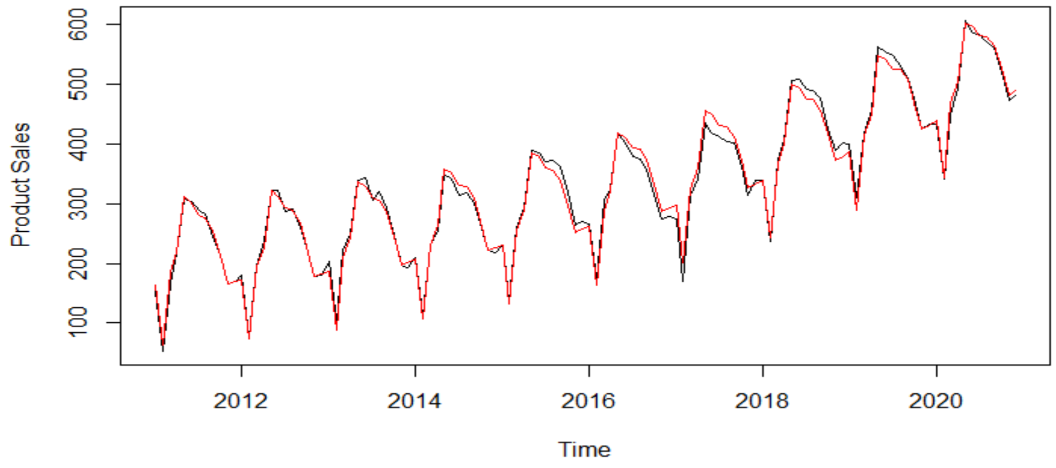


Figure 10: Plot of fitted values for the quadratic time regression model

The fitted values seem to be closer to the actual values now, when compared to figure 6. After following the same procedure as before an AR(2) model is chosen again for the residuals. The corresponding plots are generated to make sure everything is okay:

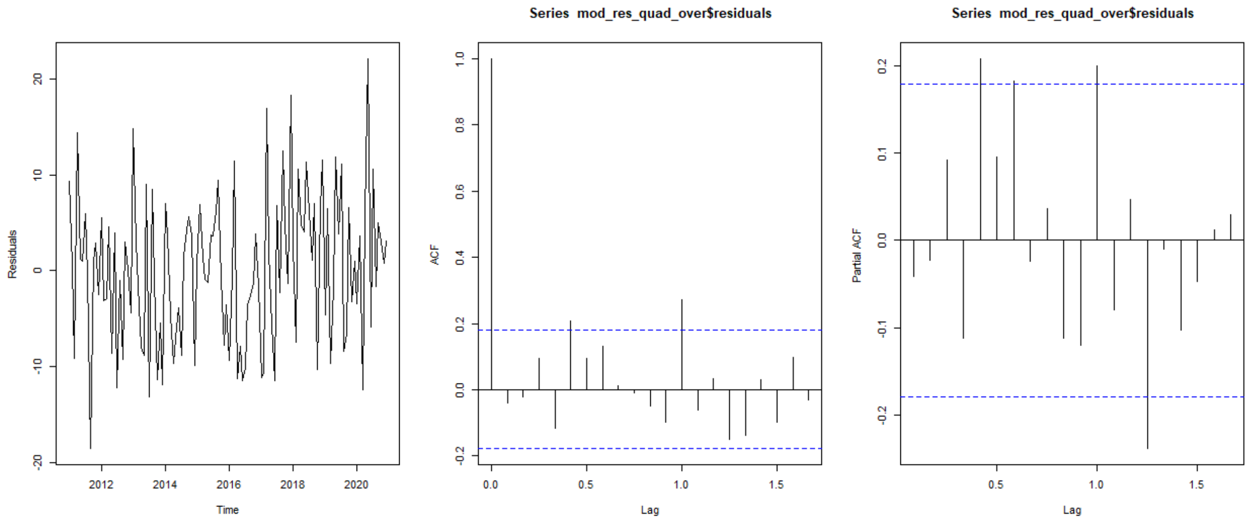


Figure 11: Plots for residuals for the quadratic time regression model

The plots for the residuals seem okay, and there is no evidence of non-stationarity. After that forecasts for the tenth year are generated and compared with the actual values trough the MSE (Appendix 1.8). The MSE is now 224.3. It is clear that the quadratic model is a better fit to the data. However, the MSE is still large when compared to the one generated for the arima model.

4. DLM model

When using a time regressions model, one of the drawbacks, is that the trend and any seasonal effects are predicted to continue unchanged into the future. However, this normally not very realistic. To get around the problem, is to allow the coefficients to vary over time and fit a dynamic linear model. The DLM model is built using maximum likelihood estimates. First the smoothed trend line is fitted to the data (Appendix 1.9):

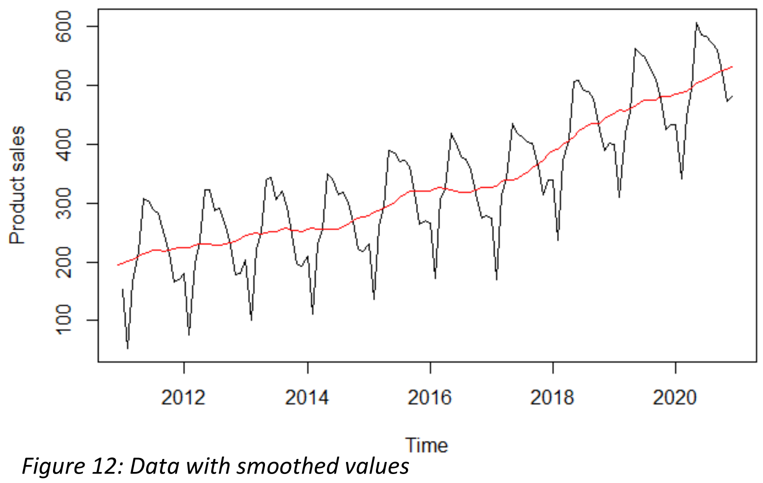


Figure 12: Data with smoothed values

Next forecasts for the next six months (values in Appendix 1.9.1) and for the tenth year can be generated (Appendix 1.9). The plots can be found below:

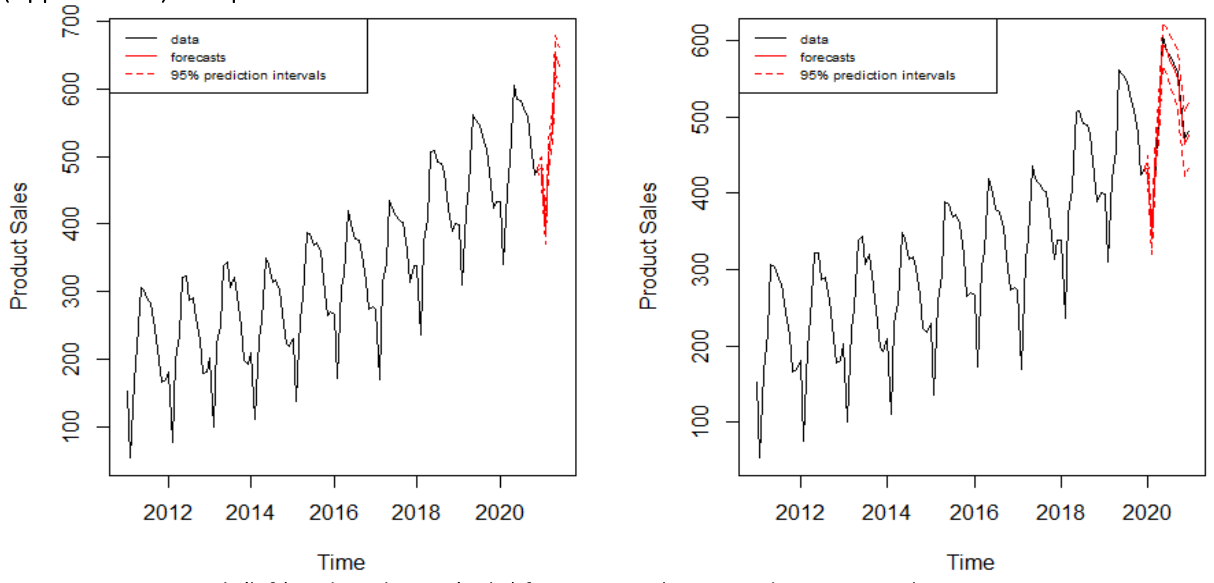


Figure 12: Six-month (left) and tenth year (right) forecasts, with 95% prediction intervals

From the left plot the prediction intervals seem close to the forecasted data. From the right plot, it is clear that the predicted values for the tenth year are quite close to the actual values. The MSE is 52.67. This is quite smaller when compared to the time regression model, but a bit larger when compared to the arima model.

Conclusion and Comparison

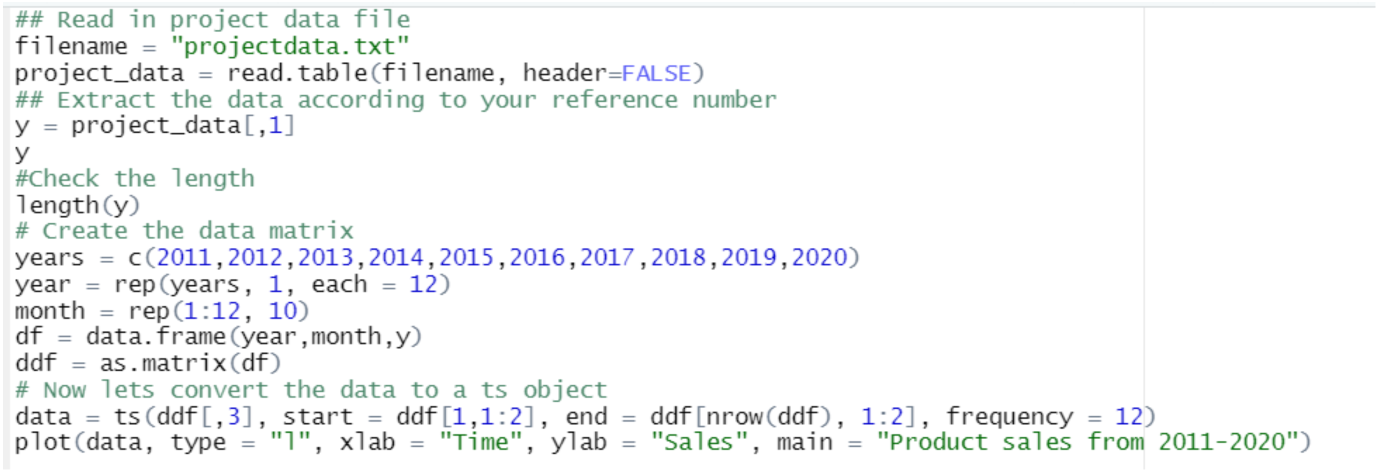
After having fitted the 4 different models the following table is generated:

Model	MSE
SARIMA (0,1,1) x (0,1,1)$_{12}$	**38.89**
Time Series Regression	327.16
Quadratic Time Series Regression	224.3
DLM	52.67

From the table above it is clear, that the arima model is the best fit for these data, followed by the DLM model. The quadratic model is preferred over the linear one, but the MSE is still quite large compared to the other two model. Moreover, the quadratic trend which extends into the future may not be a very realistic scenario for the product sales data, and so may be appropriate only for short term predictions.

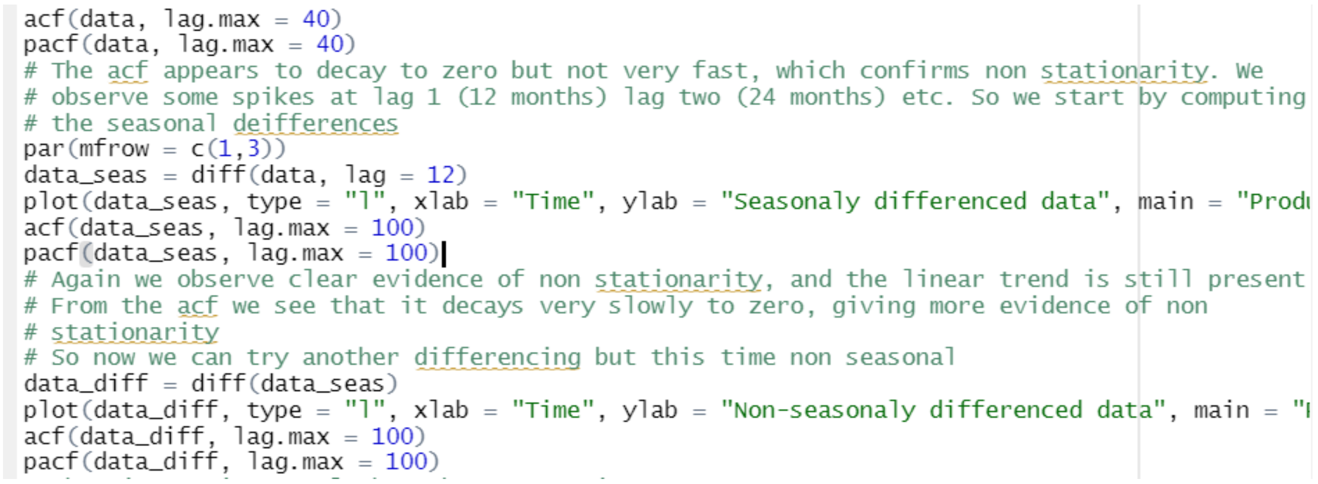
Appendix

1.1: Loading the data and creating the time series object



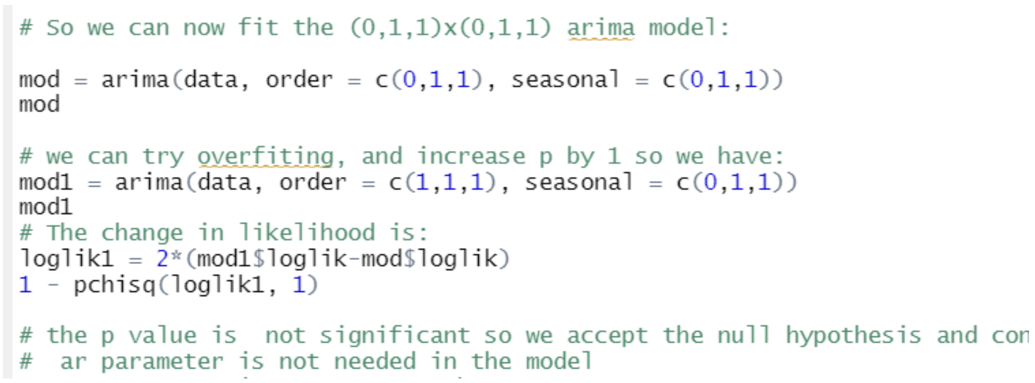
```
## Read in project data file
filename = "projectdata.txt"
project_data = read.table(filename, header=FALSE)
## Extract the data according to your reference number
y = project_data[,1]
y
#Check the length
length(y)
# Create the data matrix
years = c(2011,2012,2013,2014,2015,2016,2017,2018,2019,2020)
year = rep(years, 1, each = 12)
month = rep(1:12, 10)
df = data.frame(year,month,y)
ddf = as.matrix(df)
# Now lets convert the data to a ts object
data = ts(ddf[,3], start = ddf[1,1:2], end = ddf[nrow(ddf), 1:2], frequency = 12)
plot(data, type = "l", xlab = "Time", ylab = "Sales", main = "Product sales from 2011-2020")
```

1.2: Acf, pacf plots for data, and differencing



```
acf(data, lag.max = 40)
pacf(data, lag.max = 40)
# The acf appears to decay to zero but not very fast, which confirms non stationarity. We
# observe some spikes at lag 1 (12 months) lag two (24 months) etc. So we start by computing
# the seasonal deifferences
par(mfrow = c(1,3))
data_seas = diff(data, lag = 12)
plot(data_seas, type = "l", xlab = "Time", ylab = "Seasonaly differenced data", main = "Produ
acf(data_seas, lag.max = 100)
pacf(data_seas, lag.max = 100)
# Again we observe clear evidence of non stationarity, and the linear trend is still present
# From the acf we see that it decays very slowly to zero, giving more evidence of non
# stationarity
# So now we can try another differencing but this time non seasonal
data_diff = diff(data_seas)
plot(data_diff, type = "l", xlab = "Time", ylab = "Non-seasonaly differenced data", main = "
acf(data_diff, lag.max = 100)
pacf(data_diff, lag.max = 100)
```

1.3: Fitting the arima model, preforming overfitting and undefitting, and residual plots

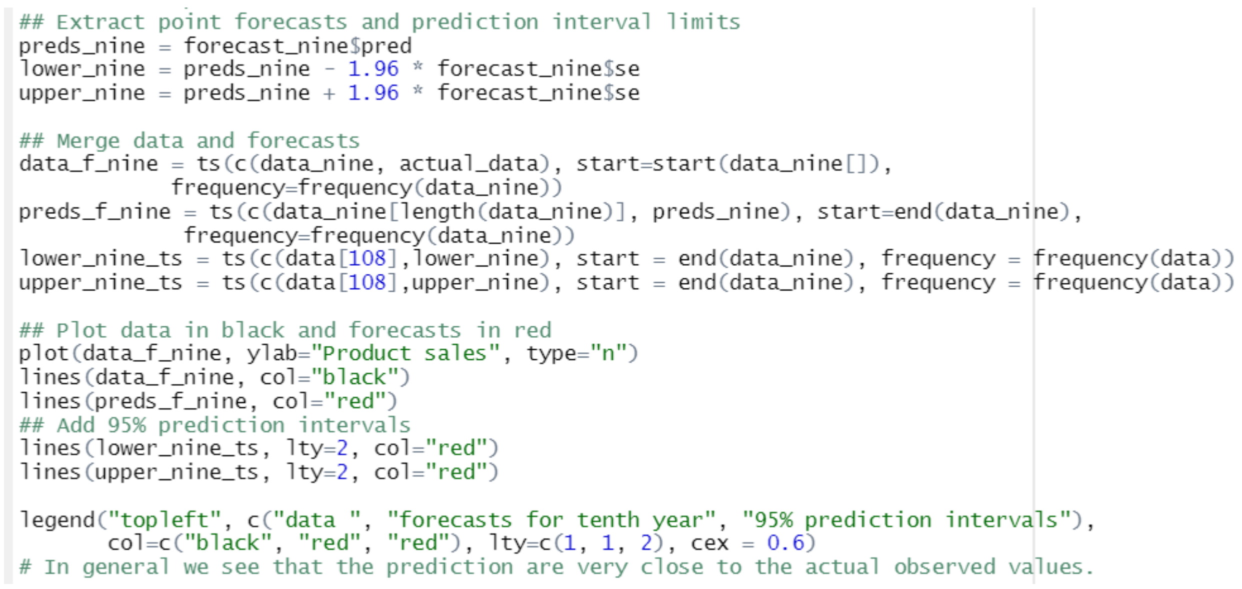


```
# So we can now fit the (0,1,1)x(0,1,1) arima model:

mod = arima(data, order = c(0,1,1), seasonal = c(0,1,1))
mod

# we can try overfiting, and increase p by 1 so we have:
mod1 = arima(data, order = c(1,1,1), seasonal = c(0,1,1))
mod1
# The change in likelihood is:
loglik1 = 2*(mod1$loglik-mod$loglik)
1 - pchisq(loglik1, 1)

# the p value is  not significant so we accept the null hypothesis and con
#  ar parameter is not needed in the model
# Now we try to increase p star by 1:
mod2 = arima(data, order = c(0,1,1), seasonal = c(1,1,1))
mod2

loglik2 = 2*(mod2$loglik-mod$loglik)
1 - pchisq(loglik2, 1)

# The p value is not significant so we reject the null hypothesis.The extr
# needed
# Now we increase q by 1:
mod3 = arima(data, order = c(0,1,2), seasonal = c(0,1,1))
mod3

loglik3 = 2*(mod3$loglik-mod$loglik)
1 - pchisq(loglik3, 1)
# The p value is not significant so the extra ma parameter is not needed
# Finally we increase q star by 1:
mod4 = arima(data, order = c(0,1,1), seasonal = c(0,1,2))
mod4

loglik4 = 2*(mod4$loglik-mod$loglik)
1 - pchisq(loglik4, 1)

# the p value is not significant
par(mfrow = c(1,3))

# Let's examine the residuals now
resids = mod$residuals
# plot residuals
plot(resids, ylab = "Residuals")
# Plot acf and pacf
acf(resids)
pacf(resids)
```

1.4: Forecasts for the chosen arima model

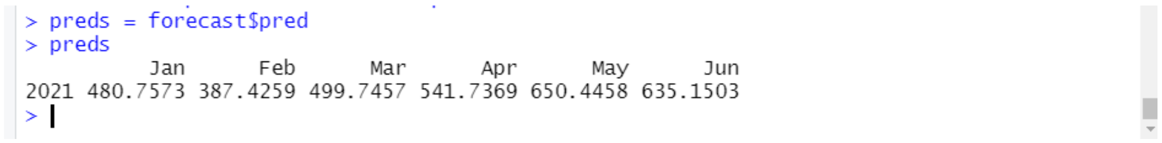
```
final_mod = arima(data, order = c(0,1,1), seasonal = c(0,1,1))

## Generate forecasts
forecast = predict(final_mod, n.ahead=6)
## Extract point forecasts and prediction interval limits
preds = forecast$pred
lower = preds - 1.96 * forecast$se
upper = preds + 1.96 * forecast$se

## Merge data and forecasts
data_f = ts(c(data, upper), start=start(data),
              frequency=frequency(data))
preds_f = ts(c(data[length(data)], preds), start=end(data),
             frequency=frequency(data))
## Plot data in black and forecasts in red
plot(data_f, ylab="Product sales", type="n")
lines(data, col="black")
lines(preds_f, col="red")
## Add 95% prediction intervals
lines(lower, lty=2, col="red")
lines(upper, lty=2, col="red")
legend("topleft", c("data", "forecasts", "95% prediction intervals"),
       col=c("black", "red", "red"), lty=c(1, 1, 2), cex = 0.6)
```

```
data_nine = ts(ddf[,3], start = ddf[1,1:2], end = ddf[108, 1:2], frequency = 12)

#Now fit our model to this data:
data_nine_mod = arima(data_nine,  order = c(0,1,1), seasonal = c(0,1,1))
forecast_nine = predict(data_nine_mod, n.ahead=12)
forecast_data = forecast_nine$pred
actual_data = tail(data,12)

# So now we can find the MSE
mse_nine = mean((actual_data - forecast_data)^2)
mse_nine
```



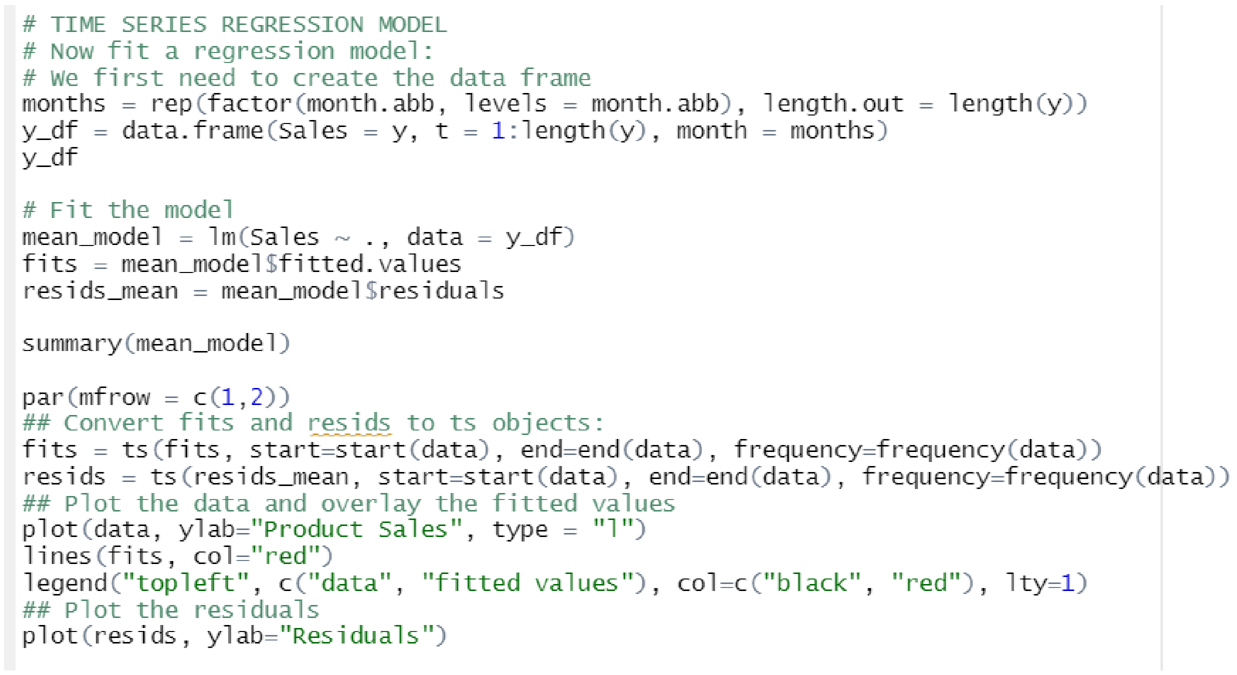
```
## Extract point forecasts and prediction interval limits
preds_nine = forecast_nine$pred
lower_nine = preds_nine - 1.96 * forecast_nine$se
upper_nine = preds_nine + 1.96 * forecast_nine$se

## Merge data and forecasts
data_f_nine = ts(c(data_nine, actual_data), start=start(data_nine[]),
             frequency=frequency(data_nine))
preds_f_nine = ts(c(data_nine[length(data_nine)], preds_nine), start=end(data_nine),
              frequency=frequency(data_nine))
lower_nine_ts = ts(c(data[108],lower_nine), start = end(data_nine), frequency = frequency(data))
upper_nine_ts = ts(c(data[108],upper_nine), start = end(data_nine), frequency = frequency(data))

## Plot data in black and forecasts in red
plot(data_f_nine, ylab="Product sales", type="n")
lines(data_f_nine, col="black")
lines(preds_f_nine, col="red")
## Add 95% prediction intervals
lines(lower_nine_ts, lty=2, col="red")
lines(upper_nine_ts, lty=2, col="red")

legend("topleft", c("data ", "forecasts for tenth year", "95% prediction intervals"),
       col=c("black", "red", "red"), lty=c(1, 1, 2), cex = 0.6)
# In general we see that the prediction are very close to the actual observed values.
```

1.4.1: Values for the six-month forecast



```
> preds = forecast$pred
> preds
          Jan      Feb      Mar      Apr      May      Jun
2021 480.7573 387.4259 499.7457 541.7369 650.4458 635.1503
> 
```

1.5: Time series regression model fitting and corresponding plots



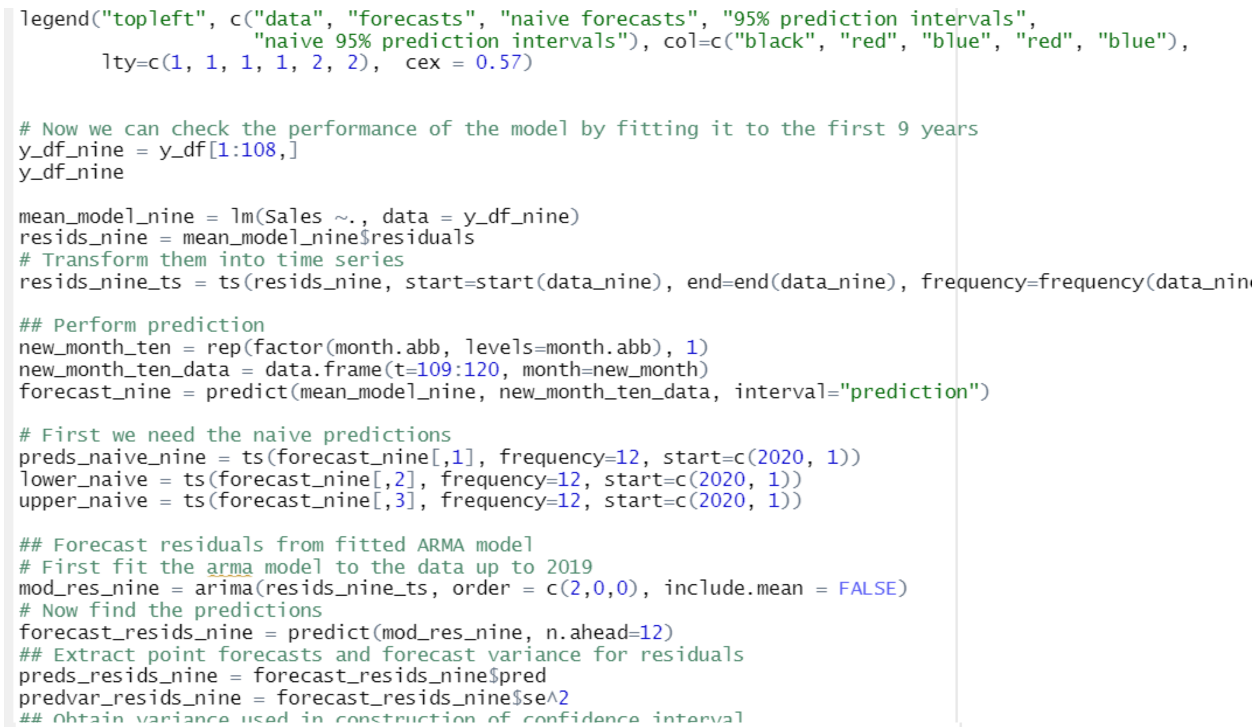
```
# TIME SERIES REGRESSION MODEL
# Now fit a regression model:
# We first need to create the data frame
months = rep(factor(month.abb, levels = month.abb), length.out = length(y))
y_df = data.frame(Sales = y, t = 1:length(y), month = months)
y_df

# Fit the model
mean_model = lm(Sales ~ ., data = y_df)
fits = mean_model$fitted.values
resids_mean = mean_model$residuals

summary(mean_model)

par(mfrow = c(1,2))
## Convert fits and resids to ts objects:
fits = ts(fits, start=start(data), end=end(data), frequency=frequency(data))
resids = ts(resids_mean, start=start(data), end=end(data), frequency=frequency(data))
## Plot the data and overlay the fitted values
plot(data, ylab="Product Sales", type = "l")
lines(fits, col="red")
legend("topleft", c("data", "fitted values"), col=c("black", "red"), lty=1)
## Plot the residuals
plot(resids, ylab="Residuals")
```

1.6: Residual plots, overfitting and underfitting for time regression model

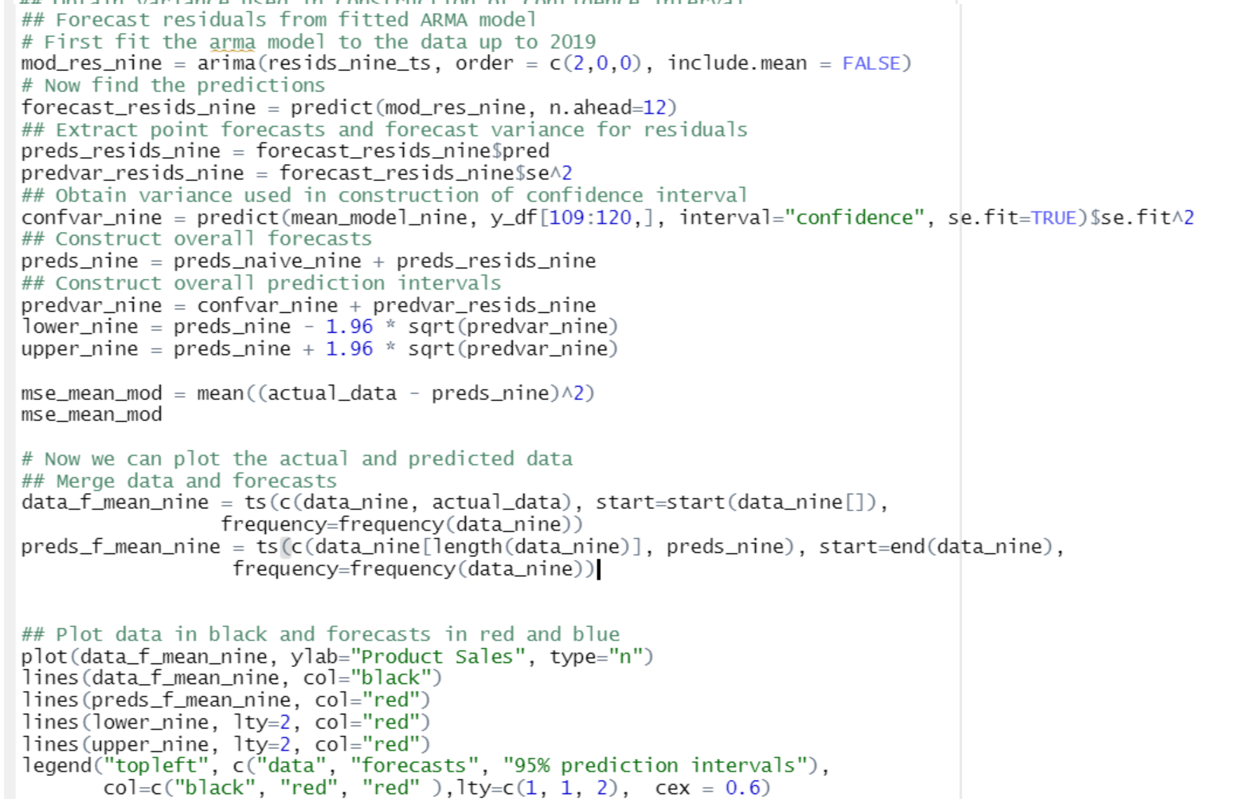
```r
# The acf seems to gradually tail of to 0, and pacf seems to cut off after lag 1. So we can
# fit an AR(1) model as follows:
mod_res = arima(resids, order = c(1,0,0), include.mean = FALSE)
mod_res
# Now lets examine the residuals on this fit
plot(mod_res$residuals, ylab = "Residuals")
acf(mod_res$residuals)
pacf(mod_res$residuals)
# Now from these plots we do not observe any obvious problem

# Next we try overfitting
mod_res_over = arima(resids, order = c(2,0,0), include.mean = FALSE)
mod_res_over

loglik_mean = 2*(mod_res_over$loglik - mod_res$loglik)
1 - pchisq(loglik_mean,1)
# The p value is very significant so we add this parameter. Now we overfit again:
mod_res_over1 = arima(resids, order = c(3,0,0), include.mean = FALSE)
mod_res_over1

loglik_mean1 = 2*(mod_res_over1$loglik - mod_res_over$loglik)
1 - pchisq(loglik_mean1,1)
# Now the p value is not significant so we do not add the third ar parameter

# Also we can try an arma(2,1):
mod_res_arma = arima(resids, order = c(2,0,1), include.mean = FALSE)
mod_res_arma

loglik_mean_arma = 2*(mod_res_arma$loglik - mod_res_over$loglik)
1 - pchisq(loglik_mean_arma,1)
# The p value is not significant so the ma term is nit needed.
# So our final model for the residuals is:
mod_res_over = arima(resids, order = c(2,0,0), include.mean = FALSE)
# We can plot the acf and pacf again:
plot(mod_res_over$residuals, ylab = "Residuals")
acf(mod_res_over$residuals)
pacf(mod_res_over$residuals)
```

1.7: Six moths forecast and forecasts for tenth year

```r
# Now we can generate forecasts
## Construct predictors over the forecast period
new_month = rep(factor(month.abb, levels=month.abb), 1)
new_y_data = data.frame(t=121:126, month=new_month[1:6])
## Perform prediction
forecast = predict(mean_model, new_y_data, interval="prediction")
## Extract forecasts and prediction interval limits
preds_naive = ts(forecast[,1], frequency=12, start=c(2021, 1))
lower_naive = ts(forecast[,2], frequency=12, start=c(2021, 1))
upper_naive = ts(forecast[,3], frequency=12, start=c(2021, 1))

## Forecast residuals from fitted ARMA model
forecast_resids = predict(mod_res_over, n.ahead=6)
## Extract point forecasts and forecast variance for residuals
preds_resids = forecast_resids$pred
predvar_resids = forecast_resids$se^2
## Obtain variance used in construction of confidence interval
confvar = predict(mean_model, new_y_data, interval="confidence", se.fit=TRUE)$se.fit^2
## Construct overall forecasts
preds = preds_naive + preds_resids
## Construct overall prediction intervals
predvar = confvar + predvar_resids
lower = preds - 1.96 * sqrt(predvar)
upper = preds + 1.96 * sqrt(predvar)


## Merge data and forecasts
y_f = ts(c(data, upper_naive), start=start(data), frequency=frequency(data))
preds_naive_f = ts(c(data[length(data)], preds_naive), start=end(data),
                   frequency=frequency(data))
lower_naive_f= ts(c(data[length(data)],lower_naive), start = end(data), frequency = frequency(data))
upper_naive_f= ts(c(data[length(data)],upper_naive), start = end(data), frequency = frequency(data))
preds_f = ts(c(data[length(data)], preds), start=end(data), frequency=frequency(data))

## Plot data in black and forecasts in red and blue
plot(y_f, ylab="Product Sales", type="n")
lines(data, col="black")
lines(preds_naive_f, col="blue")
lines(preds_f, col="red")
lines(lower_naive, lty=2, col="blue")
lines(upper_naive, lty=2, col="blue")
lines(lower, lty=2, col="red")
```



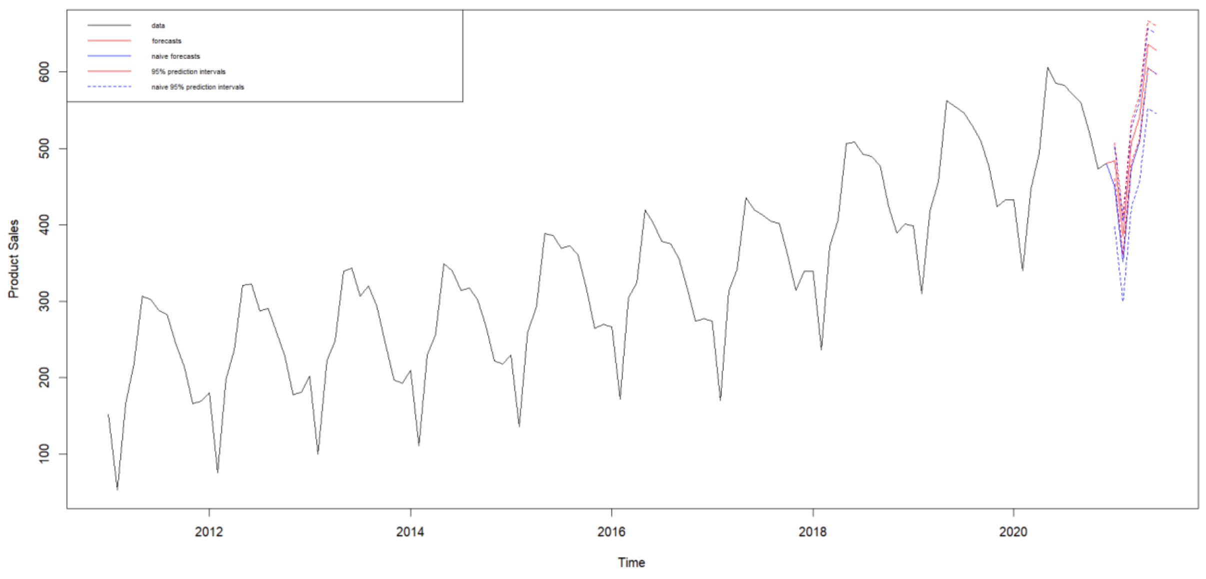
```
legend("topleft", c("data", "forecasts", "naive forecasts", "95% prediction intervals",
                    "naive 95% prediction intervals"), col=c("black", "red", "blue", "red", "blue"),
       lty=c(1, 1, 1, 1, 2, 2),  cex = 0.57)


# Now we can check the performance of the model by fitting it to the first 9 years
y_df_nine = y_df[1:108,]
y_df_nine

mean_model_nine = lm(Sales ~., data = y_df_nine)
resids_nine = mean_model_nine$residuals
# Transform them into time series
resids_nine_ts = ts(resids_nine, start=start(data_nine), end=end(data_nine), frequency=frequency(data_nin

## Perform prediction
new_month_ten = rep(factor(month.abb, levels=month.abb), 1)
new_month_ten_data = data.frame(t=109:120, month=new_month)
forecast_nine = predict(mean_model_nine, new_month_ten_data, interval="prediction")

# First we need the naive predictions
preds_naive_nine = ts(forecast_nine[,1], frequency=12, start=c(2020, 1))
lower_naive = ts(forecast_nine[,2], frequency=12, start=c(2020, 1))
upper_naive = ts(forecast_nine[,3], frequency=12, start=c(2020, 1))

## Forecast residuals from fitted ARMA model
# First fit the arma model to the data up to 2019
mod_res_nine = arima(resids_nine_ts, order = c(2,0,0), include.mean = FALSE)
# Now find the predictions
forecast_resids_nine = predict(mod_res_nine, n.ahead=12)
## Extract point forecasts and forecast variance for residuals
preds_resids_nine = forecast_resids_nine$pred
predvar_resids_nine = forecast_resids_nine$se^2
## Obtain variance used in construction of confidence interval
```



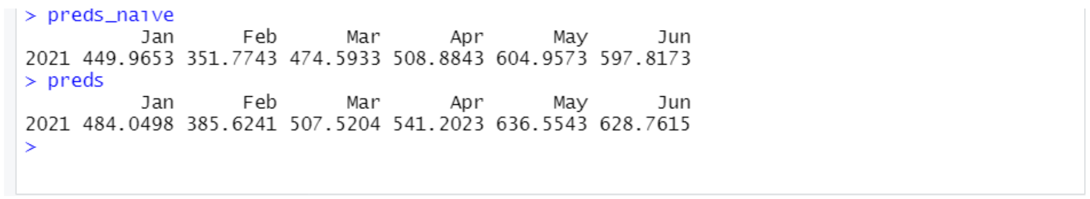
```
## Forecast residuals from fitted ARMA model
# First fit the arma model to the data up to 2019
mod_res_nine = arima(resids_nine_ts, order = c(2,0,0), include.mean = FALSE)
# Now find the predictions
forecast_resids_nine = predict(mod_res_nine, n.ahead=12)
## Extract point forecasts and forecast variance for residuals
preds_resids_nine = forecast_resids_nine$pred
predvar_resids_nine = forecast_resids_nine$se^2
```

```
## Obtain variance used in construction of confidence interval
confvar_nine = predict(mean_model_nine, y_df[109:120,], interval="confidence", se.fit=TRUE)$se.fit^2
## Construct overall forecasts
preds_nine = preds_naive_nine + preds_resids_nine
## Construct overall prediction intervals
predvar_nine = confvar_nine + predvar_resids_nine
lower_nine = preds_nine - 1.96 * sqrt(predvar_nine)
upper_nine = preds_nine + 1.96 * sqrt(predvar_nine)

mse_mean_mod = mean((actual_data - preds_nine)^2)
mse_mean_mod

# Now we can plot the actual and predicted data
## Merge data and forecasts
data_f_mean_nine = ts(c(data_nine, actual_data), start=start(data_nine[]),
                  frequency=frequency(data_nine))
preds_f_mean_nine = ts(c(data_nine[length(data_nine)], preds_nine), start=end(data_nine),
                   frequency=frequency(data_nine))


## Plot data in black and forecasts in red and blue
plot(data_f_mean_nine, ylab="Product Sales", type="n")
lines(data_f_mean_nine, col="black")
lines(preds_f_mean_nine, col="red")
lines(lower_nine, lty=2, col="red")
lines(upper_nine, lty=2, col="red")
legend("topleft", c("data", "forecasts", "95% prediction intervals"),
       col=c("black", "red", "red" ),lty=c(1, 1, 2),  cex = 0.6)
```

1.7.1 : Zoomed plot for figure 9

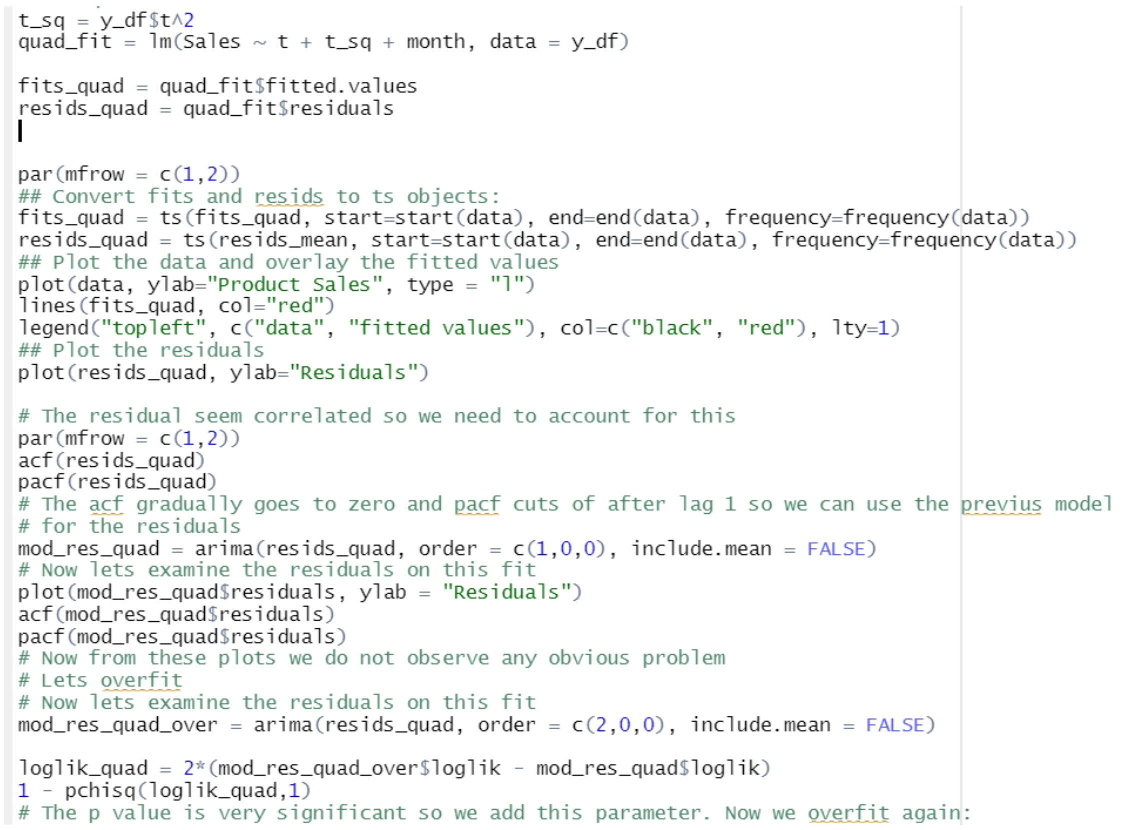


1.7.2: Naïve and not naïve predictions for the six months

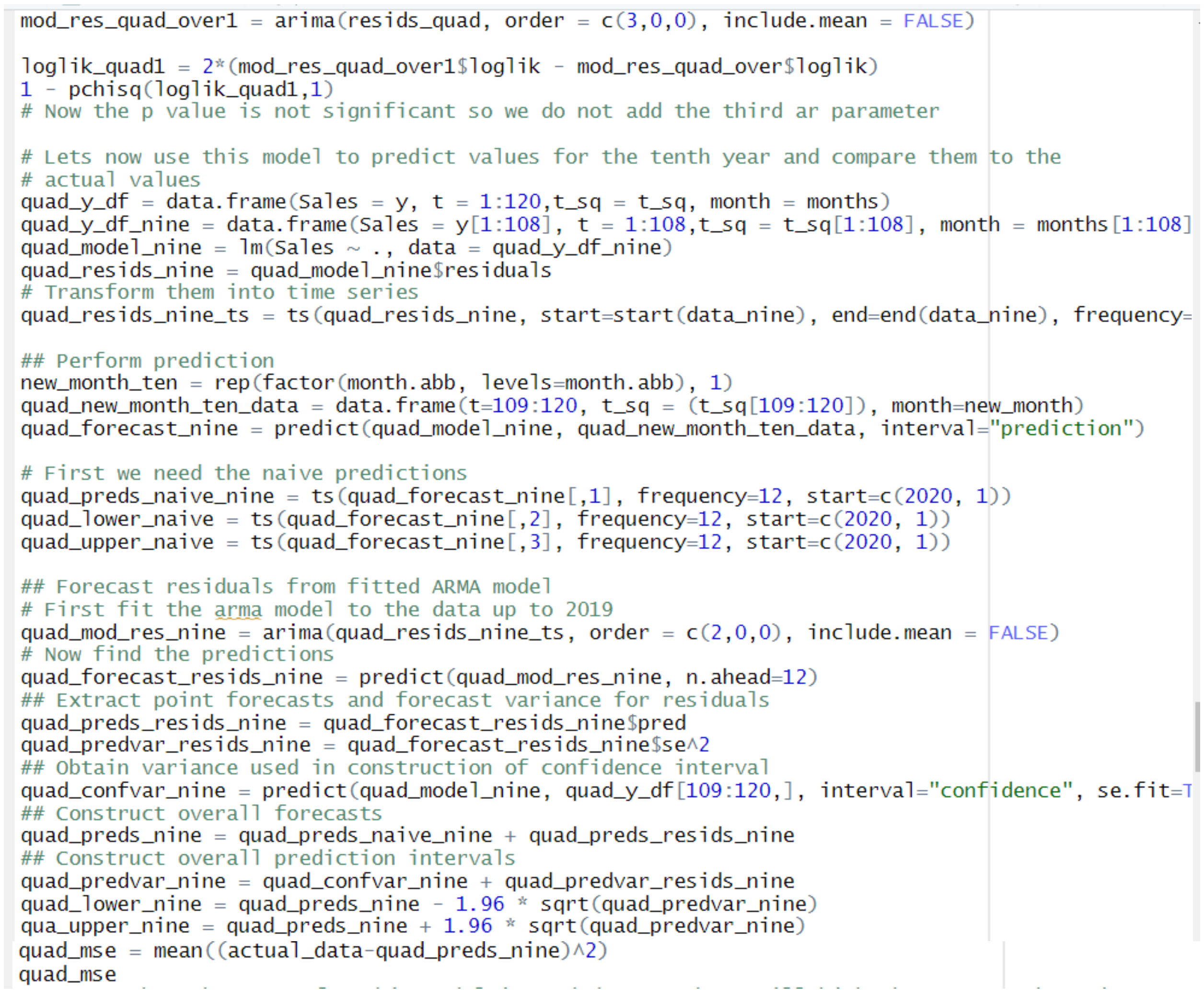


```
> preds_naive
          Jan      Feb      Mar      Apr      May      Jun
2021 449.9653 351.7743 474.5933 508.8843 604.9573 597.8173
> preds
          Jan      Feb      Mar      Apr      May      Jun
2021 484.0498 385.6241 507.5204 541.2023 636.5543 628.7615
>
```

1.8: Fitting the quadratic regression model and generating forecasts and plots

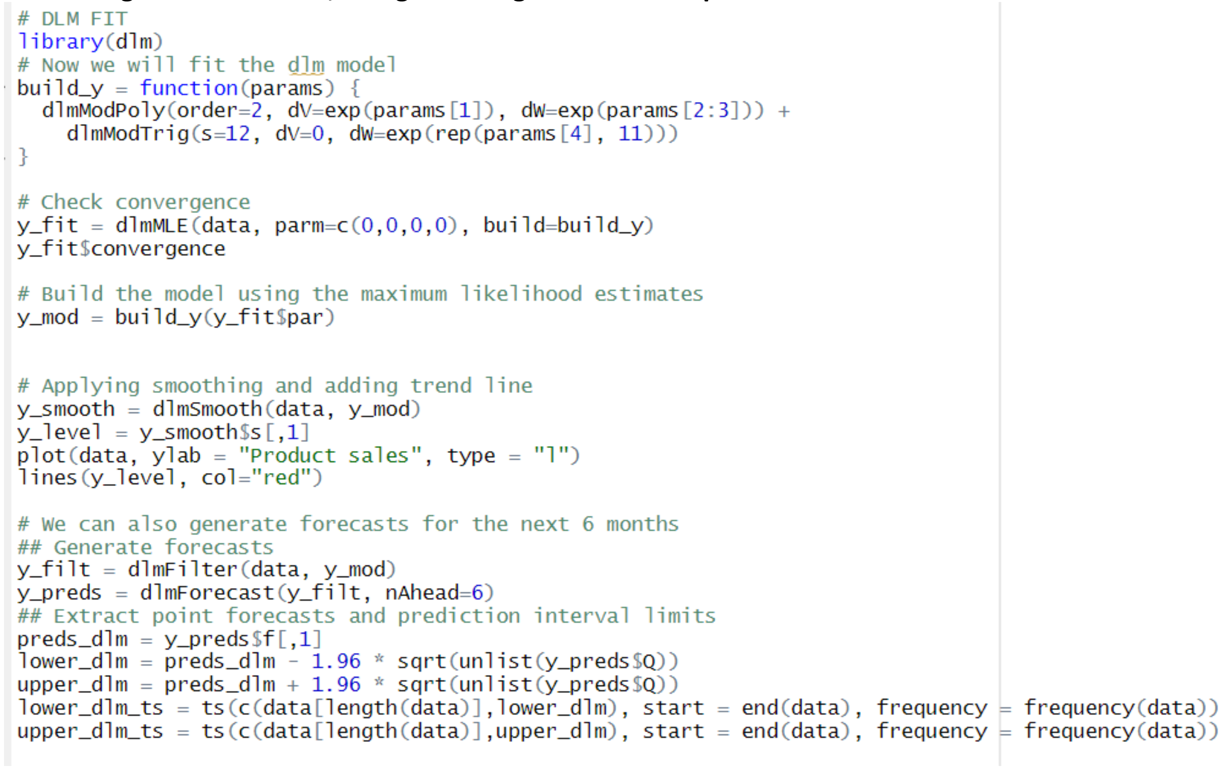


```
t_sq = y_df$t^2
quad_fit = lm(Sales ~ t + t_sq + month, data = y_df)

fits_quad = quad_fit$fitted.values
resids_quad = quad_fit$residuals


par(mfrow = c(1,2))
## Convert fits and resids to ts objects:
fits_quad = ts(fits_quad, start=start(data), end=end(data), frequency=frequency(data))
resids_quad = ts(resids_mean, start=start(data), end=end(data), frequency=frequency(data))
## Plot the data and overlay the fitted values
plot(data, ylab="Product Sales", type = "l")
lines(fits_quad, col="red")
legend("topleft", c("data", "fitted values"), col=c("black", "red"), lty=1)
## Plot the residuals
plot(resids_quad, ylab="Residuals")

# The residual seem correlated so we need to account for this
par(mfrow = c(1,2))
acf(resids_quad)
pacf(resids_quad)
# The acf gradually goes to zero and pacf cuts of after lag 1 so we can use the previus model
# for the residuals
mod_res_quad = arima(resids_quad, order = c(1,0,0), include.mean = FALSE)
# Now lets examine the residuals on this fit
plot(mod_res_quad$residuals, ylab = "Residuals")
acf(mod_res_quad$residuals)
pacf(mod_res_quad$residuals)
# Now from these plots we do not observe any obvious problem
# Lets overfit
# Now lets examine the residuals on this fit
mod_res_quad_over = arima(resids_quad, order = c(2,0,0), include.mean = FALSE)

loglik_quad = 2*(mod_res_quad_over$loglik - mod_res_quad$loglik)
1 - pchisq(loglik_quad,1)
# The p value is very significant so we add this parameter. Now we overfit again:
```

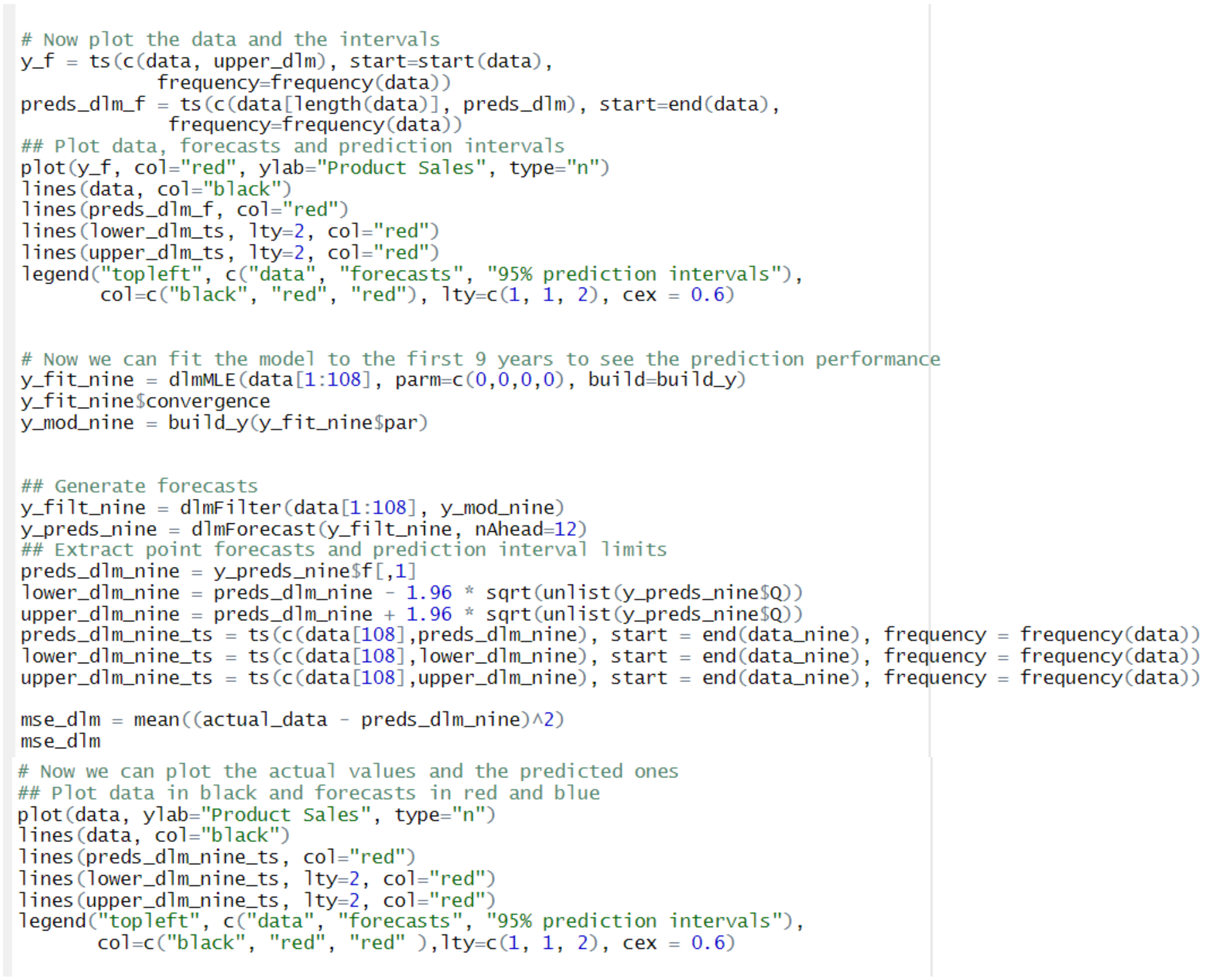


```
mod_res_quad_over1 = arima(resids_quad, order = c(3,0,0), include.mean = FALSE)

loglik_quad1 = 2*(mod_res_quad_over1$loglik - mod_res_quad_over$loglik)
1 - pchisq(loglik_quad1,1)
# Now the p value is not significant so we do not add the third ar parameter

# Lets now use this model to predict values for the tenth year and compare them to the
# actual values
quad_y_df = data.frame(Sales = y, t = 1:120,t_sq = t_sq, month = months)
quad_y_df_nine = data.frame(Sales = y[1:108], t = 1:108,t_sq = t_sq[1:108], month = months[1:108]
quad_model_nine = lm(Sales ~ ., data = quad_y_df_nine)
quad_resids_nine = quad_model_nine$residuals
# Transform them into time series
quad_resids_nine_ts = ts(quad_resids_nine, start=start(data_nine), end=end(data_nine), frequency=

## Perform prediction
new_month_ten = rep(factor(month.abb, levels=month.abb), 1)
quad_new_month_ten_data = data.frame(t=109:120, t_sq = (t_sq[109:120]), month=new_month)
quad_forecast_nine = predict(quad_model_nine, quad_new_month_ten_data, interval="prediction")

# First we need the naive predictions
quad_preds_naive_nine = ts(quad_forecast_nine[,1], frequency=12, start=c(2020, 1))
quad_lower_naive = ts(quad_forecast_nine[,2], frequency=12, start=c(2020, 1))
quad_upper_naive = ts(quad_forecast_nine[,3], frequency=12, start=c(2020, 1))

## Forecast residuals from fitted ARMA model
# First fit the arma model to the data up to 2019
quad_mod_res_nine = arima(quad_resids_nine_ts, order = c(2,0,0), include.mean = FALSE)
# Now find the predictions
quad_forecast_resids_nine = predict(quad_mod_res_nine, n.ahead=12)
## Extract point forecasts and forecast variance for residuals
quad_preds_resids_nine = quad_forecast_resids_nine$pred
quad_predvar_resids_nine = quad_forecast_resids_nine$se^2
## Obtain variance used in construction of confidence interval
quad_confvar_nine = predict(quad_model_nine, quad_y_df[109:120,], interval="confidence", se.fit=T
## Construct overall forecasts
quad_preds_nine = quad_preds_naive_nine + quad_preds_resids_nine
## Construct overall prediction intervals
quad_predvar_nine = quad_confvar_nine + quad_predvar_resids_nine
quad_lower_nine = quad_preds_nine - 1.96 * sqrt(quad_predvar_nine)
qua_upper_nine = quad_preds_nine + 1.96 * sqrt(quad_predvar_nine)
quad_mse = mean((actual_data-quad_preds_nine)^2)
quad_mse
```

1.9: Fitting the DLM model, and generating forecasts and plots

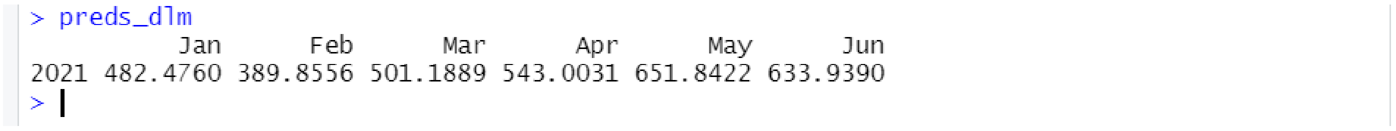


```
# DLM FIT
library(dlm)
# Now we will fit the dlm model
build_y = function(params) {
  dlmModPoly(order=2, dV=exp(params[1]), dW=exp(params[2:3])) +
    dlmModTrig(s=12, dV=0, dW=exp(rep(params[4], 11)))
}

# Check convergence
y_fit = dlmMLE(data, parm=c(0,0,0,0), build=build_y)
y_fit$convergence

# Build the model using the maximum likelihood estimates
y_mod = build_y(y_fit$par)


# Applying smoothing and adding trend line
y_smooth = dlmSmooth(data, y_mod)
y_level = y_smooth$s[,1]
plot(data, ylab = "Product sales", type = "l")
lines(y_level, col="red")

# We can also generate forecasts for the next 6 months
## Generate forecasts
y_filt = dlmFilter(data, y_mod)
y_preds = dlmForecast(y_filt, nAhead=6)
## Extract point forecasts and prediction interval limits
preds_dlm = y_preds$f[,1]
lower_dlm = preds_dlm - 1.96 * sqrt(unlist(y_preds$Q))
upper_dlm = preds_dlm + 1.96 * sqrt(unlist(y_preds$Q))
lower_dlm_ts = ts(c(data[length(data)],lower_dlm), start = end(data), frequency = frequency(data))
upper_dlm_ts = ts(c(data[length(data)],upper_dlm), start = end(data), frequency = frequency(data))
```



```
# Now plot the data and the intervals
y_f = ts(c(data, upper_dlm), start=start(data),
            frequency=frequency(data))
preds_dlm_f = ts(c(data[length(data)], preds_dlm), start=end(data),
             frequency=frequency(data))
## Plot data, forecasts and prediction intervals
plot(y_f, col="red", ylab="Product Sales", type="n")
lines(data, col="black")
lines(preds_dlm_f, col="red")
lines(lower_dlm_ts, lty=2, col="red")
lines(upper_dlm_ts, lty=2, col="red")
legend("topleft", c("data", "forecasts", "95% prediction intervals"),
       col=c("black", "red", "red"), lty=c(1, 1, 2), cex = 0.6)


# Now we can fit the model to the first 9 years to see the prediction performance
y_fit_nine = dlmMLE(data[1:108], parm=c(0,0,0,0), build=build_y)
y_fit_nine$convergence
y_mod_nine = build_y(y_fit_nine$par)


## Generate forecasts
y_filt_nine = dlmFilter(data[1:108], y_mod_nine)
y_preds_nine = dlmForecast(y_filt_nine, nAhead=12)
## Extract point forecasts and prediction interval limits
preds_dlm_nine = y_preds_nine$f[,1]
lower_dlm_nine = preds_dlm_nine - 1.96 * sqrt(unlist(y_preds_nine$Q))
upper_dlm_nine = preds_dlm_nine + 1.96 * sqrt(unlist(y_preds_nine$Q))
preds_dlm_nine_ts = ts(c(data[108],preds_dlm_nine), start = end(data_nine), frequency = frequency(data))
lower_dlm_nine_ts = ts(c(data[108],lower_dlm_nine), start = end(data_nine), frequency = frequency(data))
upper_dlm_nine_ts = ts(c(data[108],upper_dlm_nine), start = end(data_nine), frequency = frequency(data))

mse_dlm = mean((actual_data - preds_dlm_nine)^2)
mse_dlm
# Now we can plot the actual values and the predicted ones
## Plot data in black and forecasts in red and blue
plot(data, ylab="Product Sales", type="n")
lines(data, col="black")
lines(preds_dlm_nine_ts, col="red")
lines(lower_dlm_nine_ts, lty=2, col="red")
lines(upper_dlm_nine_ts, lty=2, col="red")
legend("topleft", c("data", "forecasts", "95% prediction intervals"),
       col=c("black", "red", "red" ),lty=c(1, 1, 2), cex = 0.6)
```

1.9.1: Predictions for the six months



```
> preds_dlm
          Jan      Feb      Mar      Apr      May      Jun
2021 482.4760 389.8556 501.1889 543.0031 651.8422 633.9390
> 
```

Project MAS8382

Question 1

$X_t = \varepsilon_t + \theta^* \varepsilon_{t-2}$ where $\varepsilon_t \sim N(0, \sigma^2)$

a) $d = 0$, $p = 0$, $q = 0$

Now for the seasonal parameters:

$d^* = 0$, $p^* = 0$ and $q^* = 1$; $s = 2$

So we have an ARIMA $(0,0,0) \times (0,0,1)_2$ model

b) We have that:

$\Theta^*(B^s) = 1 + \theta^* B^2$

The characteristic equation is therefore:

$1 + \theta^* x^2 = 0$

$\Leftrightarrow \theta^* x^2 = -1$

$\Leftrightarrow x^2 = -\frac{1}{\theta^*} = \pm\sqrt{\frac{-1}{\theta^*}} = \pm i\, \theta^{*-1/2}$

Clearly if $|\theta^*| < 1$ the roots lie outside the unit circle.

Also since $q = 0$ we, know that the process is invertible.

Also since d and d^* are 0 the process is invertible

c) We have that $X_t = \varepsilon_t + \theta^* \varepsilon_{t-2}$. If we work with the infinite order autoregression representation we have that:

$$\varepsilon_t = X_t - \theta^* \varepsilon_{t-2}$$

$$\begin{aligned}
X_t &= \varepsilon_t + \theta^* (X_{t-2} - \theta^* \varepsilon_{t-4}) \\
&= \varepsilon_t - \theta^{*2} \varepsilon_{t-4} + \theta^* X_{t-2} \\
&= \varepsilon_t - \theta^{*2} (X_{t-4} - \theta^* \varepsilon_{t-6}) + \theta^* X_{t-2} \\
&= \varepsilon_t - \theta^{*2} X_{t-4} + \theta^{*3} \varepsilon_{t-6} + \theta^* X_{t-2} \\
&\ \vdots \\
&= \varepsilon_t + \sum_{k=1}^{\infty} (-1)^{k+1} \theta^{*k} X_{t-k}
\end{aligned}$$

so $\boxed{\Pi_k = (-1)^{k+1} \theta^{*k}}$

d) We have that:

$$\begin{aligned}
X_{n+1} = \varepsilon_{n+1} + \theta^* \varepsilon_{n-1} &= \varepsilon_{n+1} + \theta^* (X_{n-1} - \theta^* \varepsilon_{n-3}) \\
&= \theta^* X_{n-1} + \varepsilon_{n+1} - \theta^2 \varepsilon_{n-3}
\end{aligned}$$

0.029

Question 2

$$s_t = \sum_{k=1}^{k} \left\{ a_k \cos\left(\frac{2\pi k t}{p}\right) + b_k \left(\frac{2\pi k t}{p}\right) \right\}$$

Now: $$s_i = \sum_{k=1}^{k} \left\{ a_k \cos\left(\frac{2 i \pi k}{p}\right) + b_k \sin\left(\frac{2 i \pi k}{p}\right) \right\}$$

$$s_{i+p} = \sum_{k=1}^{k} \left\{ a_k \cos\left(\frac{2(i+p)\pi k}{p}\right) + b_k \sin\left(\frac{2(i+p)\pi k}{p}\right) \right\}$$

$$s_{i+2p} = \sum_{k=1}^{k} \left\{ a_k \cos\left(\frac{2(i+2p)\pi k}{p}\right) + b_k \sin\left(\frac{2(i+2p)\pi k}{p}\right) \right.$$

$\vdots$

We know that:

$$\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$$
$$\sin(A+B) = \sin(A)\cos(B) + \cos(A)\sin(B)$$

So: $$s_{i+p} = \sum_{k=1}^{k} \left\{ a_k \left[\cos\left(\frac{2 i \pi k}{p}\right)\cos\left(\frac{2pk\pi}{p}\right) - \sin\left(\frac{2 i \pi k}{p}\right)\sin\left(\frac{2pk\pi}{p}\right) \right] \right.$$
$$\left. + b_k \left[\sin\left(\frac{2 i \pi k}{p}\right)\cos\left(\frac{2pk\pi}{p}\right) + \cos\left(\frac{2 i \pi k}{p}\right)\sin\left(\frac{2pk\pi}{p}\right) \right] \right\}$$

$$= \sum_{k=1}^{k} \left\{ a_k \left[\cos\left(\frac{2 i \pi k}{p}\right)\overset{1}{\cancel{\cos(2\pi k)}} + \sin\left(\frac{2 i \pi k}{p}\right)\overset{0}{\cancel{\sin(2\pi k)}} \right] \right.$$
$$\left. + b_k \left[\sin\left(\frac{2 i \pi k}{p}\right)\overset{1}{\cancel{\cos(2k\pi)}} + \cos\left(\frac{2 i \pi k}{p}\right)\overset{0}{\cancel{\sin(2\pi k)}} \right] \right\}$$

$$s_{i+p} = \sum_{k=1}^{k} \left\{ a_k \cos\left(\frac{2 i \pi k}{p}\right) + b_k \sin\left(\frac{2 i \pi k}{p}\right) \right\} = s_i$$

Similarly:

$$S_{i+2p} = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2(i+2p)\pi k}{p}\right) + b_k \sin\left(\frac{2(i+2p)\pi k}{p}\right) \right\}$$

$$= \sum_{k=1}^{K} \left\{ a_k \left[\cos\left(\frac{2i\pi k}{p}\right) \cancelto{1}{\cos\left(\frac{4p\pi k}{p}\right)} - \sin\left(\frac{2i\pi k}{p}\right) \cancelto{0}{\sin\left(\frac{4p\pi k}{p}\right)} \right] \right.$$

$$\left. + b_k \left[\sin\left(\frac{2i\pi k}{p}\right) \cancelto{1}{\cos\left(\frac{4p\pi k}{p}\right)} - \cos\left(\frac{2i\pi k}{p}\right) \cancelto{0}{\sin\left(\frac{4p\pi k}{p}\right)} \right] \right\}$$

$$= \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2i\pi k}{p}\right) + b_k \sin\left(\frac{2i\pi k}{p}\right) \right\}$$

$$= S_i = S_{i+p}$$

which is true for S_{i+3p}, S_{i+4p}, ...
as required ■

Now we also need to show that:

$$\sum_{i=1}^{p} S_i = 0$$

We have that:

$$S_1 = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2\pi k}{p}\right) + b_k \sin\left(\frac{2\pi k}{p}\right) \right\}$$

$$S_2 = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{4\pi k}{p}\right) + b_k \sin\left(\frac{4\pi k}{p}\right) \right\}$$

$$S_3 = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{6\pi k}{p}\right) + b_k \sin\left(\frac{6\pi k}{p}\right) \right\}$$

$$\vdots$$

$$S_p = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2\pi k p}{p}\right) + b_k \sin\left(\frac{2\pi k p}{p}\right) \right\}$$

So:

$$\sum_{i=1}^{p} s_i = \sum_{k=1}^{K}\left\{a_k\cos\left(\frac{2\pi k}{p}\right)+b_k\sin\left(\frac{2\pi k}{p}\right)\right\}$$
$$+\sum_{k=1}^{K}\left\{a_k\cos\left(\frac{4\pi k}{p}\right)+b_k\sin\left(\frac{4\pi k}{p}\right)\right\}$$
$$+ \cdots \sum_{k=1}^{K}\left\{a_k\cos\left(\frac{2\pi kp}{p}\right)+b_k\sin\left(\frac{2\pi kp}{p}\right)\right\}$$
$$=\sum_{k=1}^{K} a_k\left\{\cos\left(\frac{2\pi k}{p}\right)+\cos\left(\frac{4\pi k}{p}\right)+\cos\left(\frac{6\pi k}{p}\right)+\dots+\cos\left(\frac{2\pi kp}{p}\right)\right\}$$
$$+\sum_{k=1}^{K} b_k\left\{\sin\left(\frac{2\pi k}{p}\right)+\sin\left(\frac{4\pi k}{p}\right)+\sin\left(\frac{6\pi k}{p}\right)+\dots\sin\left(\frac{2\pi kp}{p}\right)\right\}$$

We know that:

$$\underbrace{\sum_{k=1}^{N}\cos\left(\frac{2\pi k}{N}\right)=0 \quad \text{and} \quad \sum_{k=1}^{N}\sin\left(\frac{2\pi k}{N}\right)=0}$$

P. Knapp, M., n.d, Sines and Cosines of Angles in Arithmetic progression. [online] Math dartmouth.edu

So now we have that

$$\sum_{i=1}^{p} s_i = 0 \quad \text{as required} \ \blacksquare$$