AI6123 Time Series Analysis

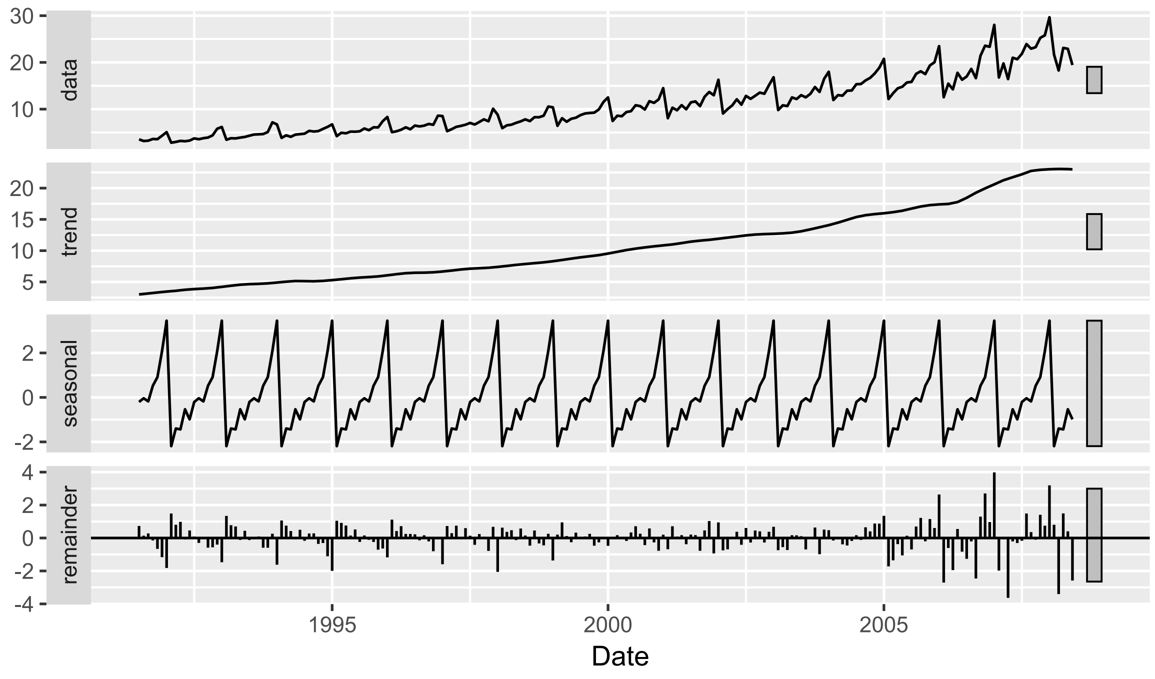
Chen Yongquan (G2002341D)

Nanyang Technological University

Assignment 2

Table of Contents

Preliminary Data Analysis



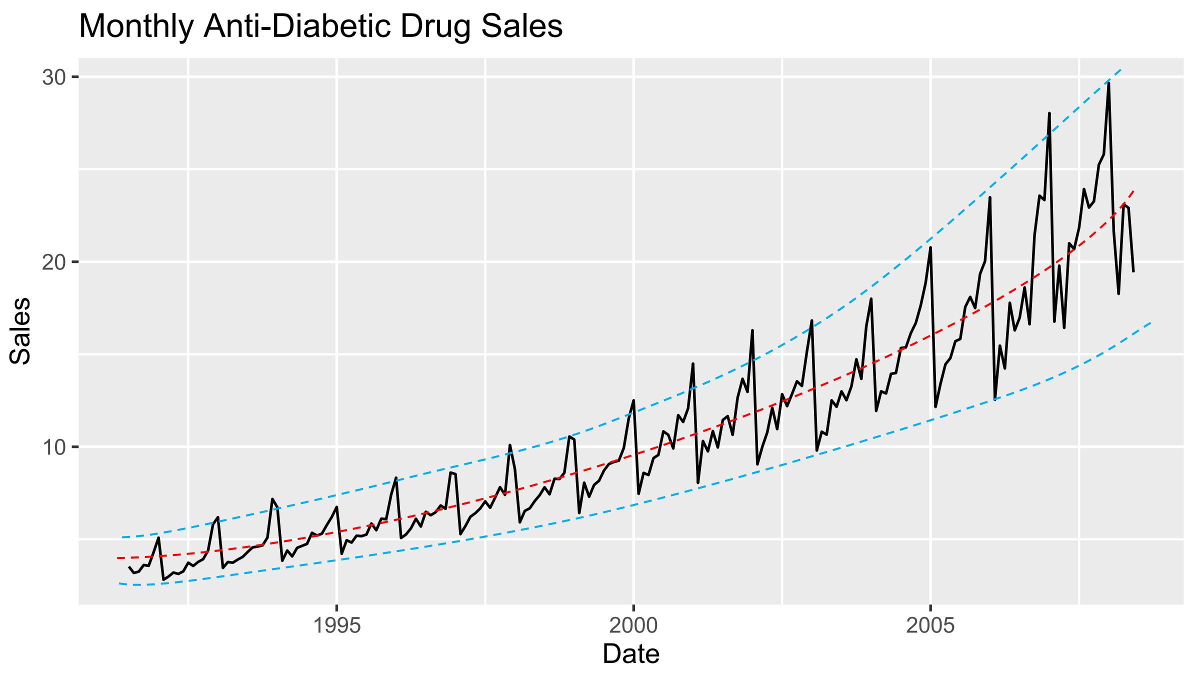


Figure 1

For this assignment we will be fitting a good model to predict the drug sales based on the monthly anti-diabetic drug sales in Australia from 1992 to 2008.

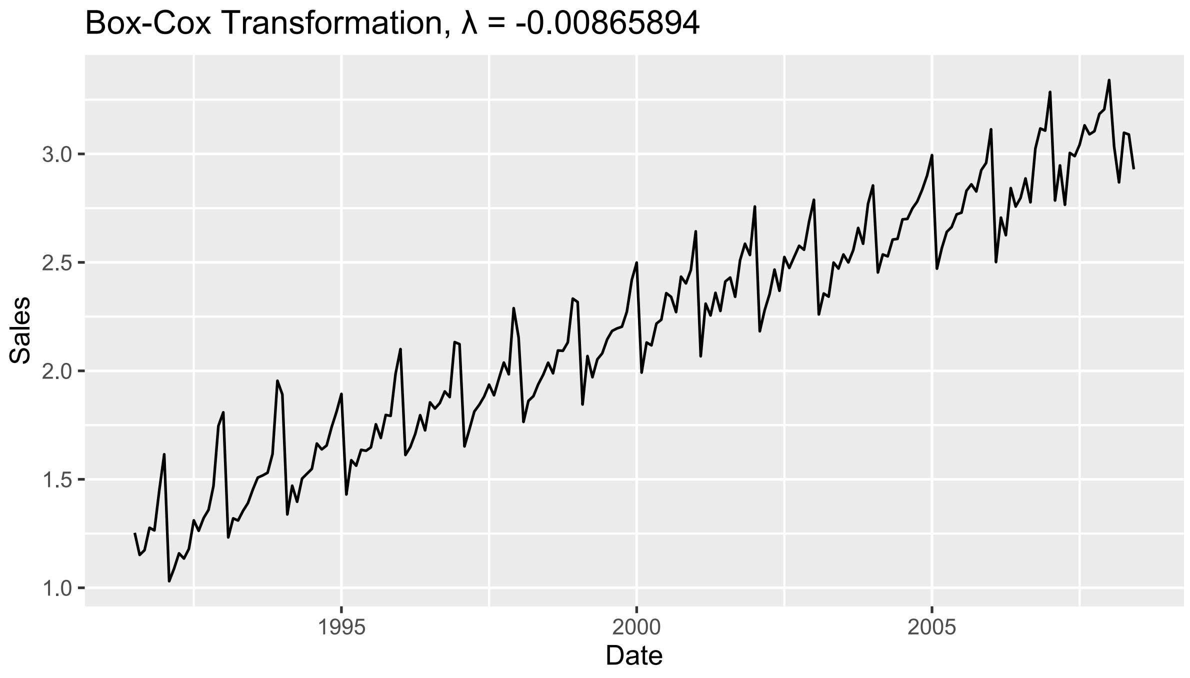
Figure 1 shows the plot of the original data, from which we can see an obvious trend component which may be linear with noise or non-linear. Either way we will need to perform differencing later to remove the component and make the data stationary before fitting a model. There is an obvious seasonality component, and between 1995 and 2000, we observe 5 peaks and troughs indicating that seasonality has an annual period i.e. 12 months. There is a definite increase in seasonality variance from 1991 to 2008, thus we will have to apply a Box-Cox transformation to make the sample variance relatively constant across the years before any differencing is done.

Box-Cox Transformation

We perform Box-Cox transformation using the code below:

```
lambda = BoxCox.lambda(x$value, 'guerrero')
x_bc = BoxCox(x$value, lambda)
```

The λ-value is automatically selected using Guerrero's method [1], and the value is shown in Figure 2.



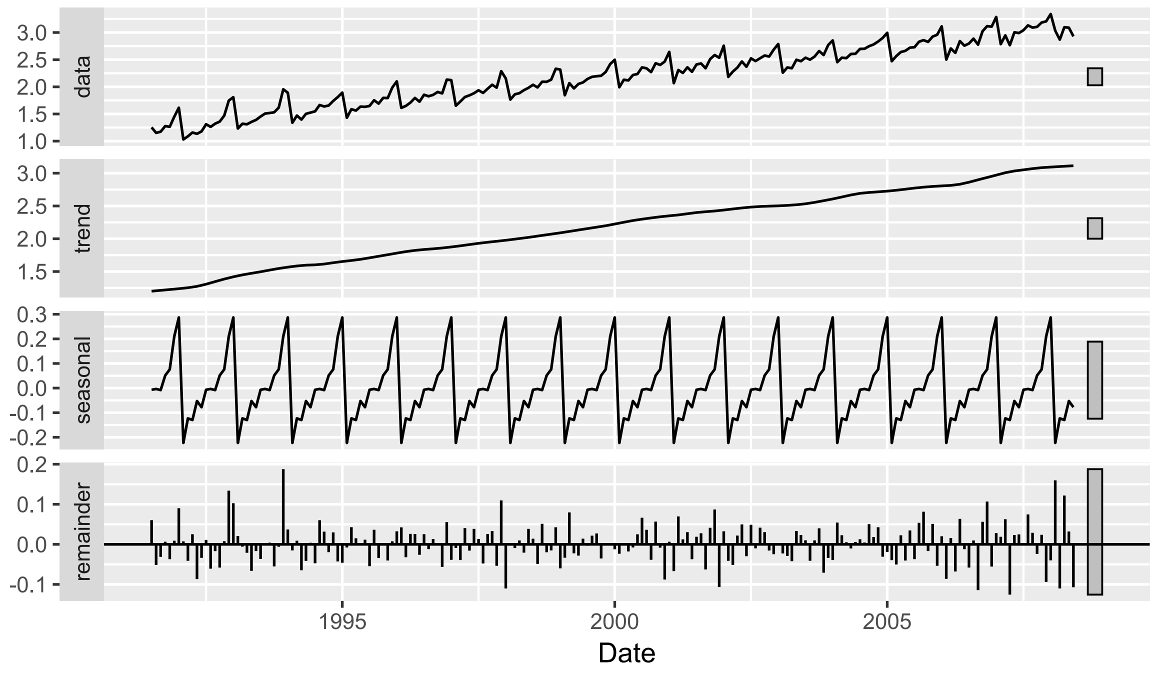


Figure 2

After taking the logarithmic transformation, we can see that sample variance across the years are much more constant. Seasonal decomposition shows that after the seasonal and trend components are removed the remainder does not exceed 0.2, as compared to a maximum remainder of 4 before transformation. There is still an obvious periodic seasonal trend and an increasing linear trend.

Lag-12 Seasonal Differencing

We thus perform differencing to remove any trend and seasonal components before attempting to fit a model. We perform seasonal differencing first followed by order differencing if necessary. This is because seasonal differencing also removes linear trend components (or partially removes higher order trend components), while order differencing only removes trend components, but leaves the seasonal component untouched. Furthermore, each time we perform seasonal differencing, we lose S (seasonal period) number of samples while we lose 1 sample for each order differencing. Thus, performing differencing in this order would be more sample efficient, especially when sample size is small. Our code for lag-12 differencing is as shown below:

```
x_s12 = diff(x_bc, 12)
```

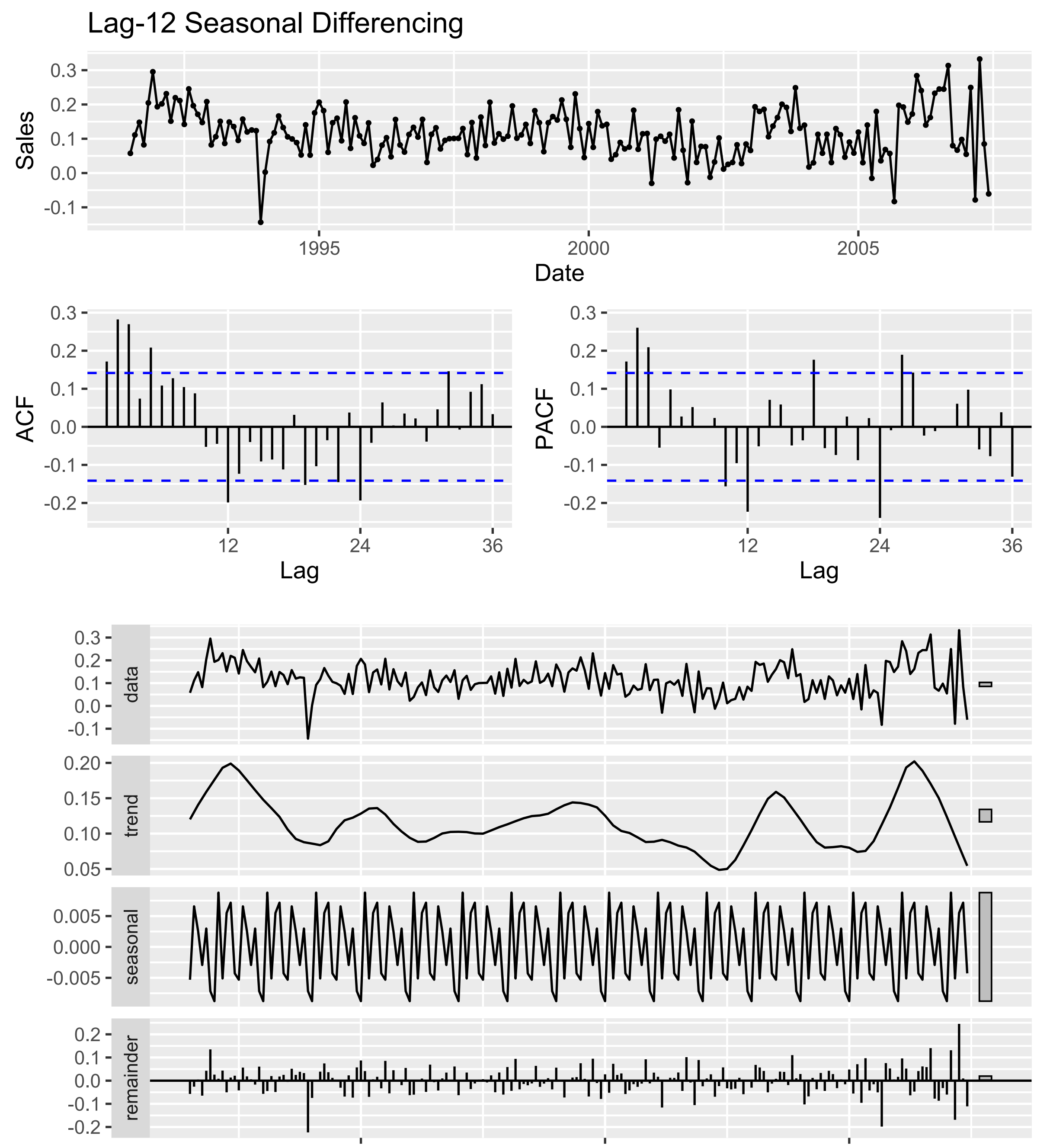


Figure 3

Based on Figure 3, data seems sufficiently stationary for us to start fitting a stationary model to the data. Seasonal decomposition shows no discernible trend component with the trend line lying within a range of 0.15. Seasonal component still seems to show some periodic pattern oscillating with an amplitude of 0.005. But it seems sufficiently small for us to ignore it for now:

$$\frac{0.005}{0.3-(-0.1)} \times 100\% = 1.25\%$$

Referring to the SACF and SPACF charts in Figure 3, SACF cuts off after lag 3 or 5, with spikes at lag 12 and 24 while SPACF cuts off after 3, with spikes at lag 12 and 24. Since both SPACF and SACF present spikes at lag 24, we can suggest either a seasonal P or Q of 2 respectively. Also, because both SPACF and SACF cut off after lag 3, we can suggest either a p or q of 3. Alternatively, since SACF also seem to cut off after lag 5, we can try a q of 5. This is the list of models we can try:

1. SARIMA(3,0,0,2,1,0,12)
2. SARIMA(0,0,3,2,1,0,12)
3. SARIMA(0,0,5,2,1,0,12)
4. SARIMA(3,0,0,0,1,2,12)
5. SARIMA(0,0,3,0,1,2,12)
6. SARIMA(0,0,5,0,1,2,12)

Instead of the original *arima()* function provided in the *stats* package, we will use the *sarima()* function provided by the *astsa* package, which provides sample count normalized AIC values, as explained in Assignment 1. Figure 4 show the diagnostics for the 6 models above, the parameters for each model are at the top-left corner of each diagram. Unfortunately, none of them passed the Ljung-Box diagnostic test as they have p-values that lie below the 5% threshold. The best models were SARIMA(3,0,0,0,1,2,12) and SARIMA(0,0,5,0,1,2,12) with normalized AIC values of -2.62496 and -2.603839 respectively. Apparently, the extended autocorrelation function (EACF) cannot be applied to a time series with a seasonal pattern, so we will resort to a brute-forcing approach to find a pair of p and q values for mixed SARIMA(p,0,q,2,1,0,12) and SARIMA(p,0,q,0,1,2,12) models. We fit models with a minimum of 0 and maximum of 5 for the p and q values using the code below:

```
df = data.frame()
for (p in 0:5) {
  for (q in 0:5) {
    tryCatch({
      fit = sarima(x_bc,p,0,q,2,1,0,12)
      df = rbind(df,
                 data.frame(p = fit$fit$arma[1],
                            q = fit$fit$arma[2],
                            P = fit$fit$arma[3],
                            Q = fit$fit$arma[4],
                            S = fit$fit$arma[5],
                            d = fit$fit$arma[6],
                            D = fit$fit$arma[7],
                            AIC = fit$AIC,
                            AICc = fit$AICc,
                            BIC = fit$BIC))
    },
    error=function(cond) {},
    warning=function(cond) {})
    tryCatch({
      fit = sarima(x_bc,p,0,q,0,1,2,12)
      df = rbind(df,
                 data.frame(p = fit$fit$arma[1],
                            q = fit$fit$arma[2],
                            P = fit$fit$arma[3],
                            Q = fit$fit$arma[4],
                            S = fit$fit$arma[5],
                            d = fit$fit$arma[6],
                            D = fit$fit$arma[7],
                            AIC = fit$AIC,
                            AICc = fit$AICc,
                            BIC = fit$BIC))
    },
    error=function(cond) {},
    warning=function(cond) {})
  }
}
```

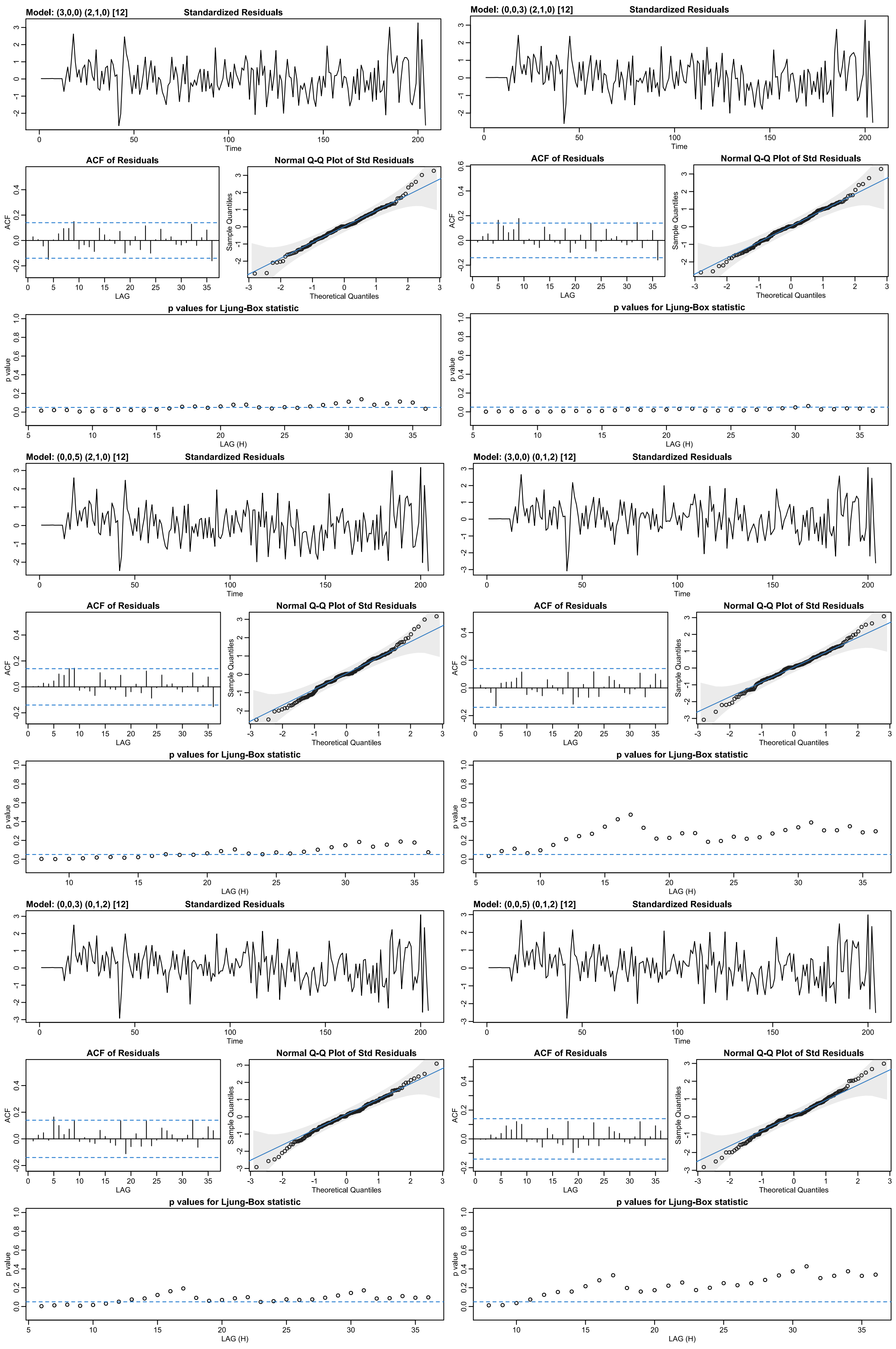

Model: (3,0,0) (2,1,0) [12]
Model: (0,0,3) (2,1,0) [12]
Model: (0,0,5) (2,1,0) [12]
Model: (3,0,0) (0,1,2) [12]
Model: (0,0,3) (0,1,2) [12]
Model: (0,0,5) (0,1,2) [12]
Standardized Residuals
ACF of Residuals
Normal Q-Q Plot of Std Residuals
p values for Ljung-Box statistic
Time
LAG
ACF
Theoretical Quantiles
Sample Quantiles
p value
LAG (H)

Figure 4

The AIC, AICc and BIC values for all successfully fitted models are stored into a data frame and sorted accordingly. The tables below show the models ranked by AIC, AICc and BIC respectively.

p	d	q	P	D	Q	S	AIC	AICc	BIC
3	0	3	0	1	2	12	-2.628263	-2.623669	-2.467796
3	0	1	0	1	2	12	-2.626388	-2.623559	-2.498014
3	0	0	0	1	2	12	-2.624960	-2.622849	-2.512633
4	0	0	0	1	2	12	-2.623539	-2.620710	-2.495165
5	0	0	0	1	2	12	-2.621276	-2.617620	-2.476855
5	0	1	0	1	2	12	-2.618715	-2.614121	-2.458247
3	0	4	0	1	2	12	-2.618422	-2.612777	-2.441907
4	0	3	0	1	2	12	-2.618422	-2.612777	-2.441907
5	0	4	0	1	2	12	-2.617339	-2.609250	-2.408731
5	0	5	0	1	2	12	-2.616909	-2.607422	-2.392254

Figure 5: Brute-forced models ranked by AIC

p	d	q	P	D	Q	S	AIC	AICc	BIC
3	0	3	0	1	2	12	-2.628263	-2.623669	-2.467796
3	0	1	0	1	2	12	-2.626388	-2.623559	-2.498014
3	0	0	0	1	2	12	-2.624960	-2.622849	-2.512633
4	0	0	0	1	2	12	-2.623539	-2.620710	-2.495165
5	0	0	0	1	2	12	-2.621276	-2.617620	-2.476855
5	0	1	0	1	2	12	-2.618715	-2.614121	-2.458247
3	0	2	0	1	2	12	-2.616563	-2.612907	-2.472142
4	0	1	0	1	2	12	-2.616542	-2.612886	-2.472121
3	0	4	0	1	2	12	-2.618422	-2.612777	-2.441907
4	0	3	0	1	2	12	-2.618422	-2.612777	-2.441907

Figure 6: Brute-forced models ranked by AICc

p	d	q	P	D	Q	S	AIC	AICc	BIC
3	0	0	0	1	2	12	-2.624960	-2.622849	-2.512633
3	0	1	0	1	2	12	-2.626388	-2.623559	-2.498014
4	0	0	0	1	2	12	-2.623539	-2.620710	-2.495165
1	0	2	0	1	2	12	-2.598519	-2.596407	-2.486191
1	0	1	0	1	2	12	-2.582139	-2.580639	-2.485858
2	0	1	0	1	2	12	-2.597687	-2.595576	-2.485359
0	0	3	0	1	2	12	-2.596478	-2.594367	-2.484151
5	0	0	0	1	2	12	-2.621276	-2.617620	-2.476855
3	0	2	0	1	2	12	-2.616563	-2.612907	-2.472142
4	0	1	0	1	2	12	-2.616542	-2.612886	-2.472121

Figure 7: Brute-forced models ranked by BIC

Typically, we only use AIC as the criteria for model selection. Strictly speaking, we should first iterate through all the models and check whether they passed the Ljung-Box test before choosing the best model out of all the adequate ones. However, we note that the model SARIMA(3,0,1,0,1,2,12) ranks within the top two models for all three IC values and also passed the Ljung-Box test as shown in Figure 8. SARIMA(3,0,3,0,1,2,12) is the top model for AIC and AICc but is not within the top 10 for BIC. Furthermore, this latter model barely passed the Ljung-Box test compared to the former model as shown in Figure 9. Q-Q plot for standardized residuals of both models show a relatively straight line, indicating that the residuals follow a gaussian white noise distribution. ACF of residuals show no correlation between lags which indicates that the residuals are the white noise irregular terms. Based on all these factors, we choose SARIMA(3,0,1,0,1,2,12) as an adequately good model for forecasting.

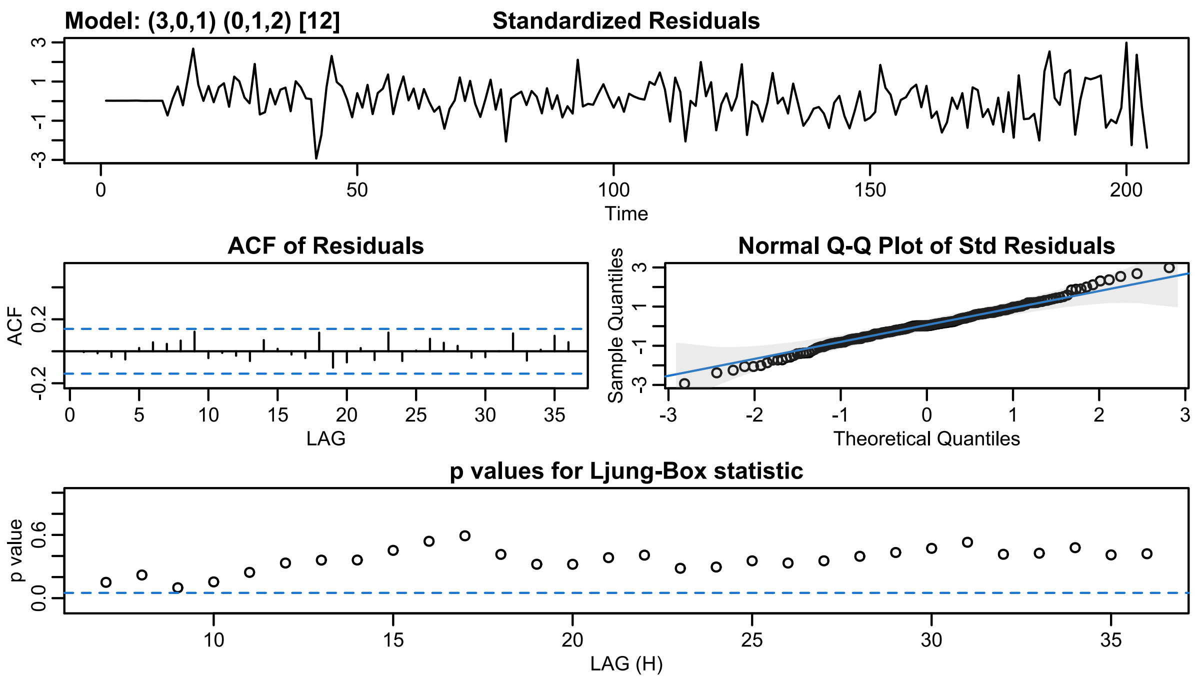


Figure 8

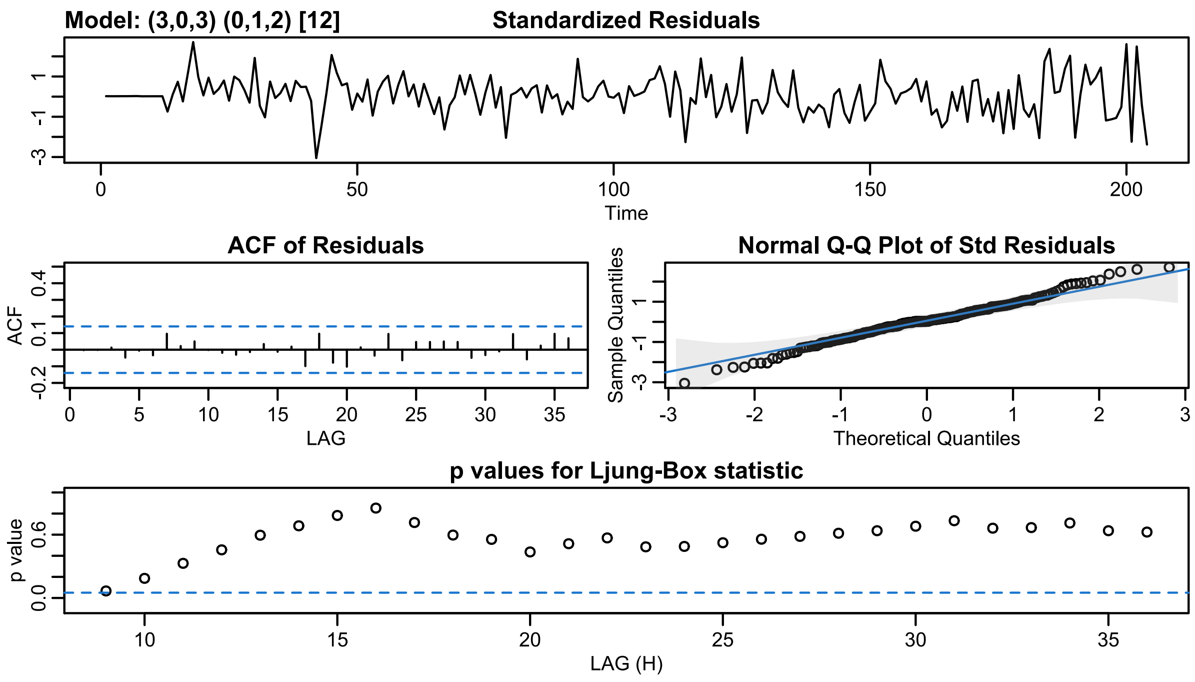


Figure 9

One-Step Forecasting using 1:9 Train-Test Split

To evaluate the performance of our model, we split the dataset into a $1:9$ ratio and use the first 180 samples for training. Then, we use the remaining ~ 20 samples as a test set for one-step forecasting, which we then use to calculate a root mean square error (RMSE) value. The code for these steps are shown on the following page:

```r
train = x_bc[1:180]
test = x_bc[181:204]
fit301012_train = Arima(train,
                        order = c(3,0,1),
                        seasonal = list(order = c(0,1,2),
                                        period = 12))
fit301012_test = Arima(test, model = fit301012_train)
fit301012_pred = forecast(fit301012_test)
plot(train, # One-step forecasting using ground truth test values
     main = "SARIMA(3,0,1,0,1,2,12)",
     ylab = "Sales",
     xlab = "Date",
     type = "l",
     xaxt = "n",
     xlim = c(1,204),
     ylim = c(1,3.5),
     panel.first = abline(h = seq(1,3,0.25),
                          v = seq(7,204,12), col = "gray95")
    )
axis(1,at=seq(7,204,12), labels = seq(1992,2008))
lines(180:204, x_bc[180:204], col="blue")
lines(180:204, c(x_bc[180],fit301012_pred$fitted), col="red")
accuracy(fit301012_pred$fitted, x_bc[181:204])
```

Figure 10 shows the forecasted values as the red line and the original ground-truth values as the blue line. We can see that the model predicted the sales values extremely closely for a year or so from the middle of 2006 up till the first quarter of 2007 until which there is slight deviations from the targets.

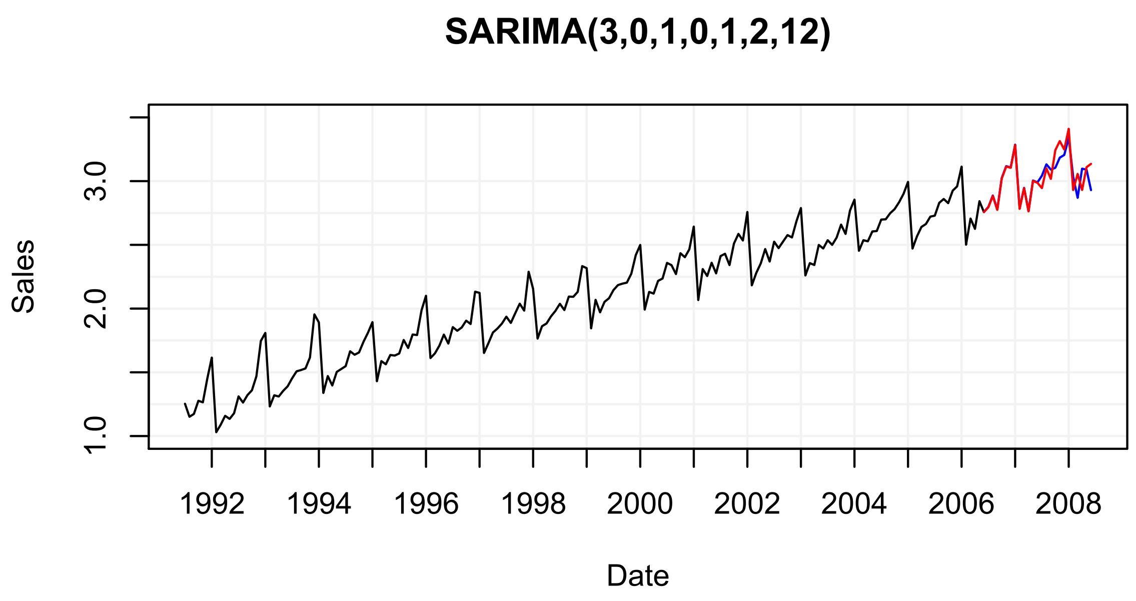


Figure 10

The errors calculated using the predicted values and the target values are as shown below:

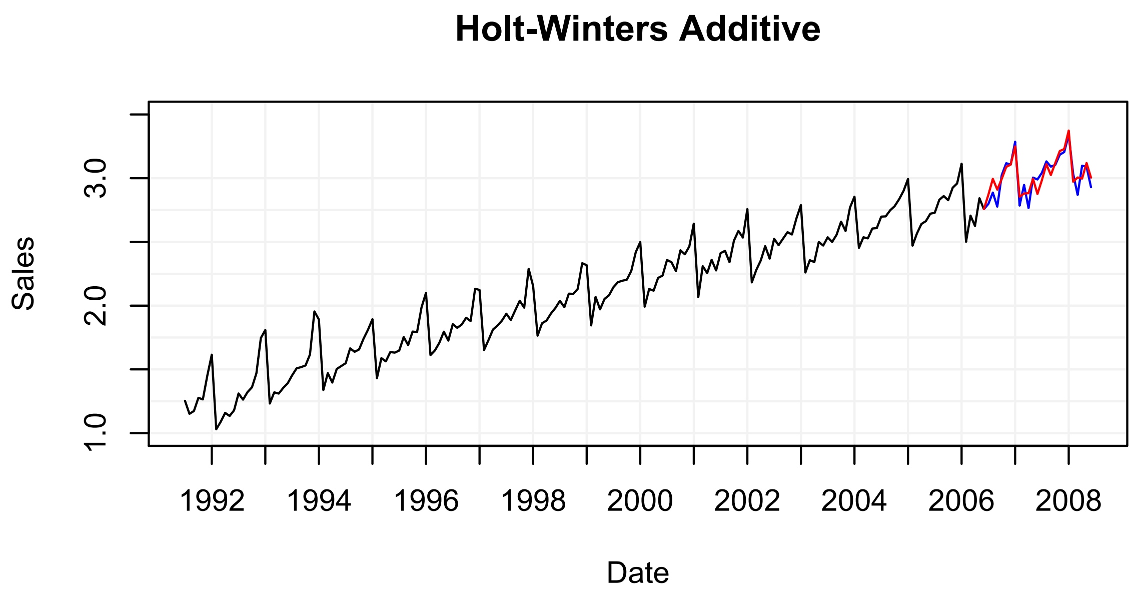
ME	RMSE	MAE	MPE	MAPE
-0.01187059	0.08560343	0.05432624	-0.3981788	1.783402

Holt-Winters Method

We also try using triple exponential smoothing methods for fitting a model. We fit two models, one using additive seasonality the other using multiplicative seasonality. The former is preferred when seasonal variations are roughly constant across the series, the latter when seasonal variations are changing proportional to the season. For brevity, we show only the code for the additive method below (the multiplicative method only differs by a single argument to $hw()$):

```
train_ts = ts(train,
              get_year_month(x$date[1]),
              get_year_month(x$date[180]),
              frequency = 12)
test_ts = ts(x_bc[181:204],
             get_year_month(x$date[181]),
             get_year_month(x$date[204]),
             frequency = 12)
hw_add = hw(train_ts,
            seasonal = 'additive')
hw_add_test = hw(test_ts,
                 seasonal = 'additive',
                 model = hw_add)
hw_add_pred = forecast(hw_add_test)
plot(train, # One-step forecasting using ground truth test values
     main = "Holt-Winters Additive",
     ylab = "Sales",
     xlab = "Date",
     type = "l",
     xaxt = "n",
     xlim = c(1,204),
     ylim = c(1,3.5),
     panel.first = abline(h = seq(1,3,0.25)
                          v = seq(7,204,12), col = "gray95")
    )
axis(1,at=seq(7,204,12), labels = seq(1992,2008))
lines(180:204, x_bc[180:204], col="blue")
lines(180:204, c(x_bc[180],hw_add_pred$fitted), col="red")
accuracy(hw_add_pred$fitted, x_bc[185:204])
```

The test set predictions are shown below in Figure 11:



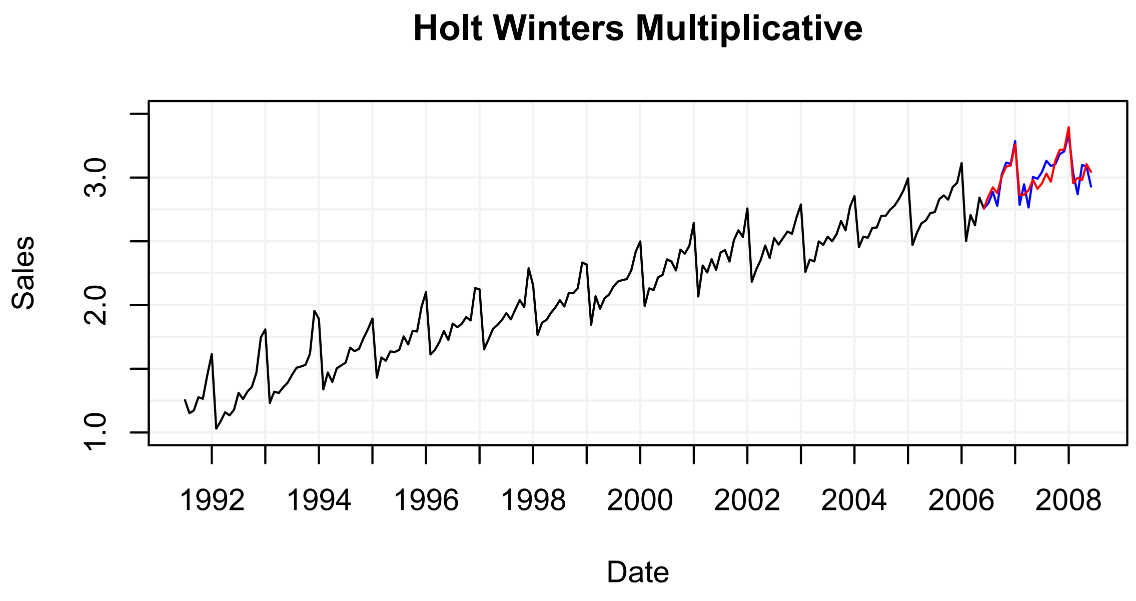


Figure 11

The errors are shown in the table below:

Model	ME	RMSE	MAE	MPE	MAPE
Additive	0.01980874	0.223418	0.202177	0.3780362	6.626005
Multiplicative	0.03122102	0.2372935	0.2159982	0.731032	7.056982

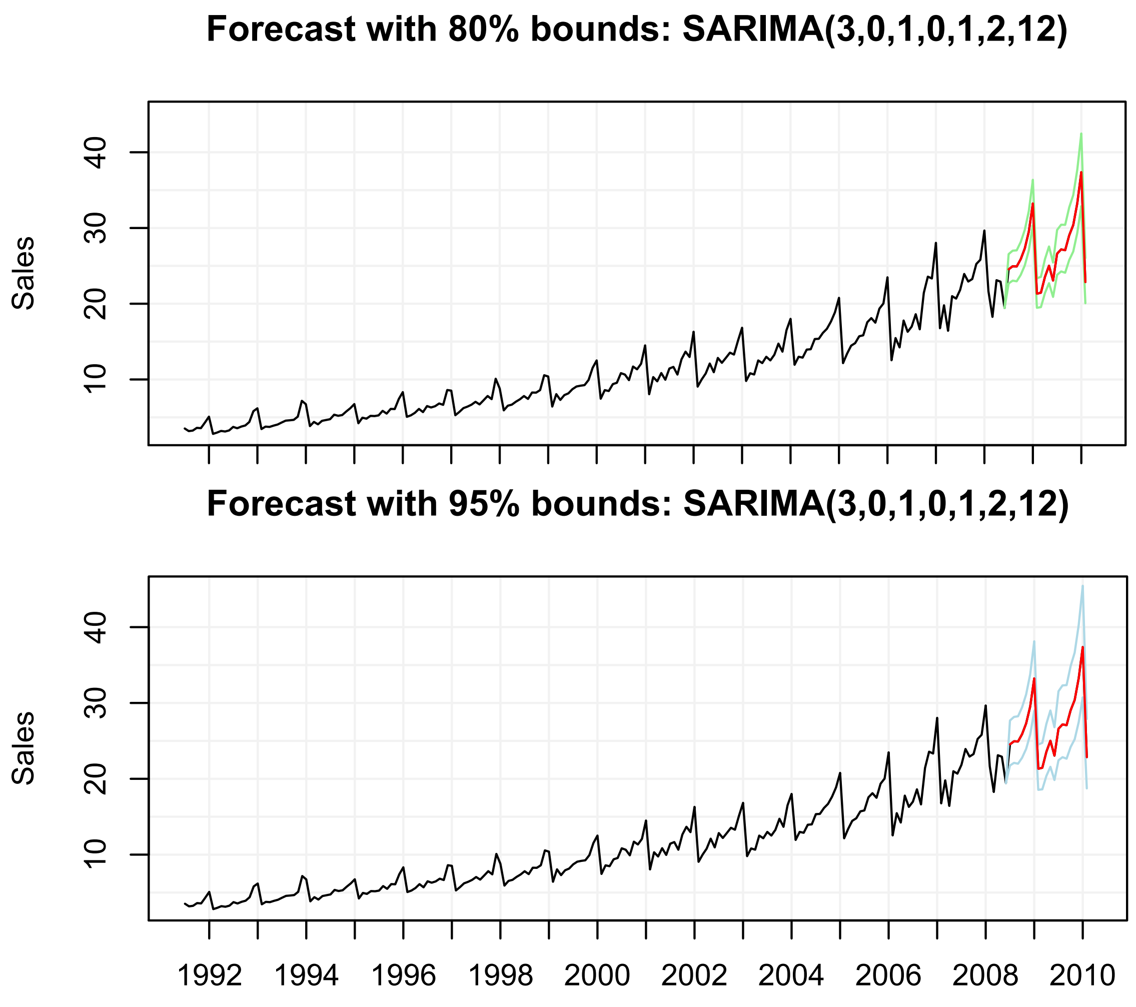
Though the plots show that the models fit the ground-truth values pretty well, RMSE for both are higher than that of our SARIMA model which has a RMSE of 0.08560343. We do note that the additive model did perform slightly better than the multiplicative model which fits our explanation above as after Box-Cox transformation, the data exhibit constant variance across seasons.

Forecasting

To use the SARIMA model for forecasting, we train using the entire dataset and predict 20 out-of-sample monthly values which we then apply a inverse Box-Cox transformation on to get the final prediction. The code for this is shown below:

```
fit301012 = Arima(x_bc,
                  order = c(3,0,1),
                  seasonal = list(order = c(0,1,2),
                                  period = 12))
pred = forecast(fit301012, h = 20)
plot(c(x$value, InvBoxCox(pred$mean, lambda)),
     main = "Forecast with 80% bounds: SARIMA(3,0,1,0,1,2,12)",
     ylab = "Sales",
     xlab = "Date",
     type = "l",
     xaxt = "n",
     xlim = c(1,225),
     ylim = c(3,45),
     panel.first = abline(h = seq(0,40,5),
                          v = seq(7,225,12), col = "gray95")
    )
lines(204:224,
      c(x$value[204],InvBoxCox(pred$upper[,1], lambda)),
      col="lightgreen") # 80% Upper Bound
lines(204:224,
      c(x$value[204],InvBoxCox(pred$lower[,1], lambda)),
      col="lightgreen") # 80% Lower Bound
axis(1,at=seq(7,225,12), labels = seq(1992,2010))
lines(205:224, InvBoxCox(pred$mean, lambda), col="red")
plot(c(x$value, InvBoxCox(pred$mean, lambda)),
     main = "Forecast with 95% bounds: SARIMA(3,0,1,0,1,2,12)",
     ylab = "Sales",
     xlab = "Date",
     type = "l",
     xaxt = "n",
     xlim = c(1,225),
     ylim = c(3,45),
     panel.first = abline(h = seq(0,40,5),
                          v = seq(7,225,12), col = "gray95")
    )
lines(204:224,
      c(x$value[204],InvBoxCox(pred$upper[,2], lambda)),
      col="lightblue") # 95% Upper Bound
lines(204:224,
      c(x$value[204],InvBoxCox(pred$lower[,2], lambda)),
      col="lightblue") # 95% Lower Bound
axis(1,at=seq(7,225,12), labels = seq(1992,2010))
lines(205:224, InvBoxCox(pred$mean, lambda), col="red")
```

The forecasted values with upper and lower bounds are shown in the figures below:



References

[1] V. Guerrero, "Time-series analysis supported by power transformations.," *Journal of Forecasting,* vol. 12, p. 37–48, 1993.