6.1.2 - R: Time Series Case Study

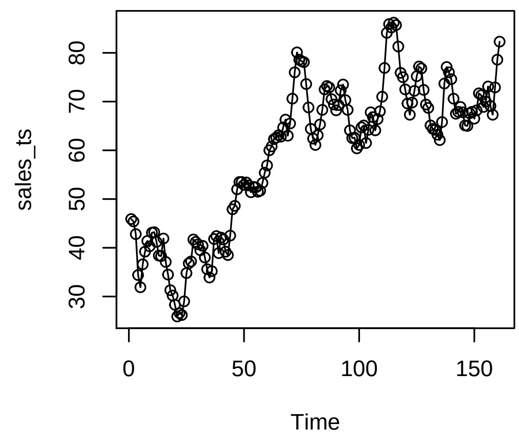
Stat 5100: Dr. Bean

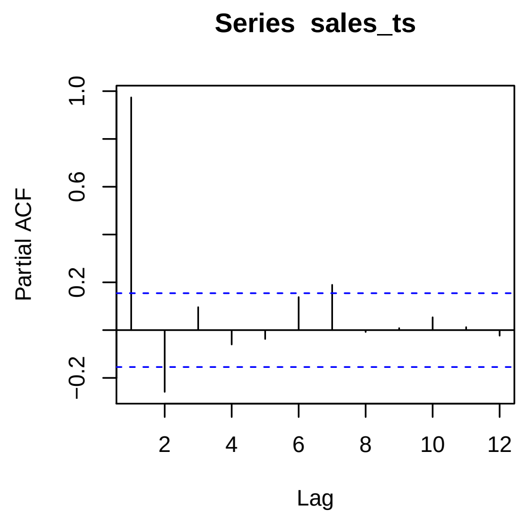
Data: Weekly sales (in thousands of units) of Super Tech Videocassette Tapes over 161 weeks [see Bowerman & O'Connell "Forecasting and Time Series: An Applied Approach", 3rd Edition, Section 10.4 Case Study. Goal: Want to forecast sales 25 weeks beyond end of data

```
library(stat5100)

# Manually read in the time series. Assume that the data are ordered in time
# and that there are no missing weeks in the time series.
sales <- c(45.9,  45.4,  42.8,  34.4, 31.9,  36.6,  39.2,  41.4,
           40.3,  43.1,  43.2,  41.2, 38.4,  38.3,  41.9,  37.1,
           34.5,  31.3,  30.2,  28.3, 25.9,  26.6,  26.2,  29,
           34.8,  36.8,  37.2,  41.7, 41.2,  40.7,  39.5,  40.4,
           38,    35.6,  33.9,  35.2, 41.8,  42.4,  38.9,  42.1,
           41.7,  39.2,  38.5,  42.5, 47.9,  48.6,  52,    53.5,
           53.5,  52.9,  53.4,  52.8, 51.4,  52.5,  52.4,  51.5,
           51.7,  53.3,  55.4,  56.9, 60,    60.8,  62.3,  62.6,
           63.1,  62.8,  64.7,  66.3, 63,    65.5,  70.6,  76,
           80.1,  78.6,  78.3,  78.1, 73.6,  68.8,  64.4,  62.4,
           61.1,  63.1,  65.3,  68.3, 72.5,  73.2,  72.9,  70.5,
           69.4,  68.2,  69.3,  72.3, 73.5,  70.3,  68.3,  64.1,
           62.5,  62.6,  60.4,  61.1, 64.7,  65.1,  61.5,  64.2,
           67.8,  66.8,  64.1,  66.4, 68,    71,    76.9,  84.1,
           85.9,  85.2,  86.2,  85.7, 81.3,  75.9,  75,    72.5,
           69.6,  67.3,  69.8,  72.2, 75.2,  77.2,  76.8,  72.4,
           69.4,  68.7,  65.1,  64.4, 64.2,  63.2,  62.1,  65.8,
           73.7,  77.1,  76,    74.6, 70.6,  67.5,  67.9,  68.9,
           67.8,  65.1,  65,    67.6, 67.9,  66.5,  68.2,  71.7,
           71.3,  68.9,  70,    73.1, 69.1,  67.3,  72.9,  78.6,  82.3)
```

Look at original data and check stationarity

```
# Create Time Series
sales_ts <- ts(sales)

# Sample Autocorrelation Plot (ACF) / Sample Partial Autocorrelation Plots (PACF)
par(mfrow = c(2, 2))
plot(sales_ts)
points(sales_ts)
acf(sales_ts, lag.max = 12)
pacf(sales_ts, lag.max = 12)
par(mfrow = c(1, 1))
```



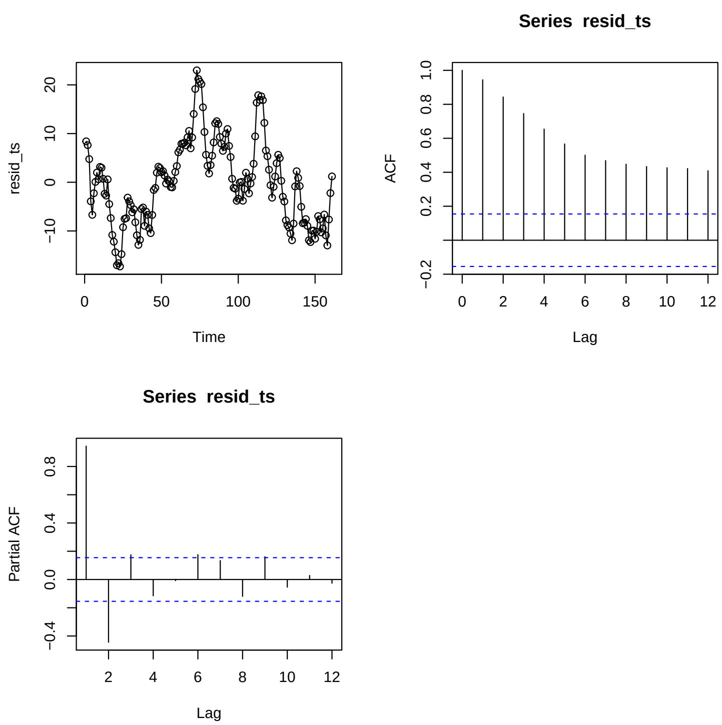



```r
# Autocorrelation check for white noise
Box.test(sales_ts, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_ts
## X-squared = 779.33, df = 6, p-value < 2.2e-16

Box.test(sales_ts, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_ts
## X-squared = 1366.7, df = 12, p-value < 2.2e-16
```

The ACF plot descends very slowly, which is evidence of nonstationarity. Also, there seems to be an obvious increase trend in sales over time. (Apparently, they haven't invented streaming services yet...)

Remove linear effect of time and recheck for stationarity.

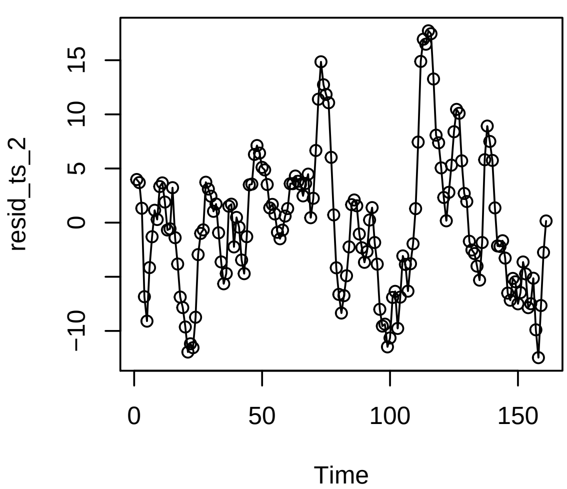
```r
sales_df <- data.frame(sales = sales, time = 1:length(sales))

sales_lm <- lm(sales ~ time, data = sales_df)

resid_ts <- ts(sales_lm$residuals)

# Sample Autocorrelation Plot (ACF) / Sample Partial Autocorrelation Plots (PACF)
par(mfrow = c(2, 2))
plot(resid_ts)
points(resid_ts)
acf(resid_ts, lag.max = 12)
pacf(resid_ts, lag.max = 12)
par(mfrow = c(1, 1))
```

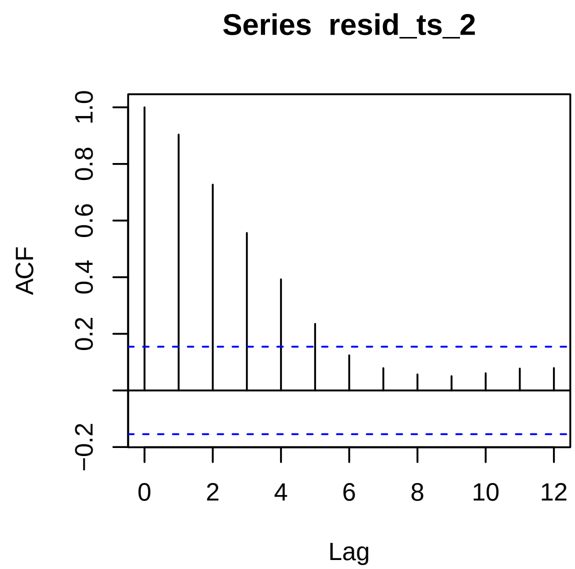


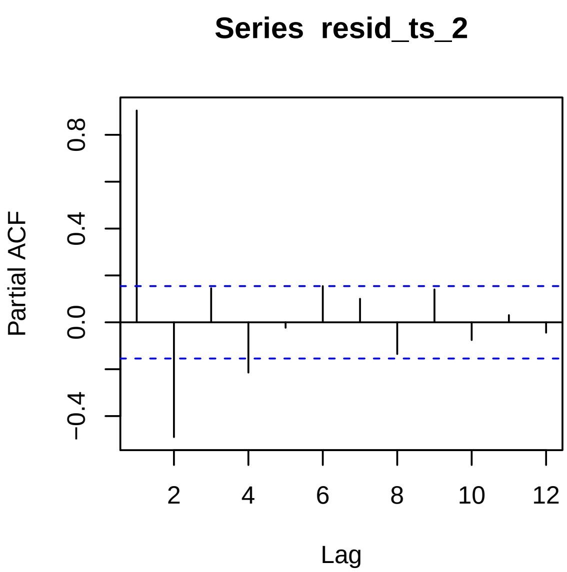

```r
# Autocorrelation check for white noise
Box.test(resid_ts, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  resid_ts
## X-squared = 523.89, df = 6, p-value < 2.2e-16

Box.test(resid_ts, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
## data:  resid_ts
## X-squared = 720.33, df = 12, p-value < 2.2e-16
```

ACF plot is still decreasing too slowly. It looks like this might be due to cyclic behavior in the residuals. We will try a new trend model that will remove what seems to be a two year (104 week) cycle in the residuals.

```r
sales_lm_2 <- lm(sales ~ time + sin(2*pi*time/104) + cos(2*pi*time/104), data = sales_df)

resid_ts_2 <- ts(sales_lm_2$residuals)

# Sample Autocorrelation Plot (ACF) / Sample Partial Autocorrelation Plots (PACF)
par(mfrow = c(2, 2))
plot(resid_ts_2)
points(resid_ts_2)
acf(resid_ts_2, lag.max = 12)
pacf(resid_ts_2, lag.max = 12)
par(mfrow = c(1, 1))
```







```r
# Autocorrelation check for white noise
Box.test(resid_ts_2, lag = 6, type = "Ljung")
```

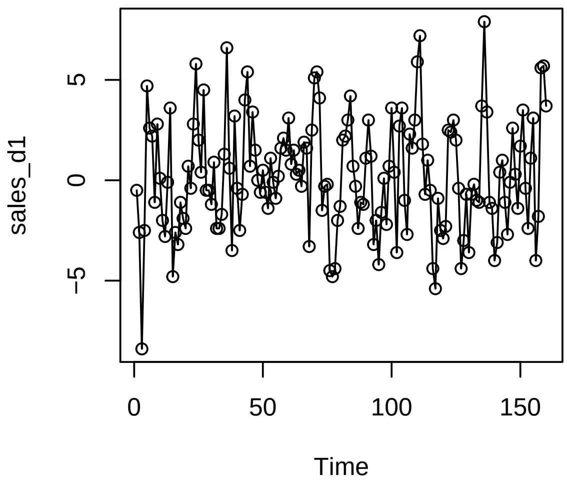
```
##
##  Box-Ljung test
##
## data:  resid_ts_2
## X-squared = 310.25, df = 6, p-value < 2.2e-16
```

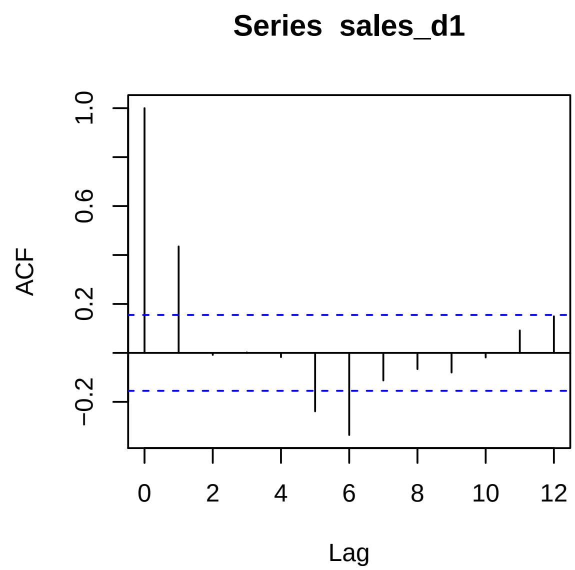
```r
Box.test(resid_ts_2, lag = 12, type = "Ljung")
```

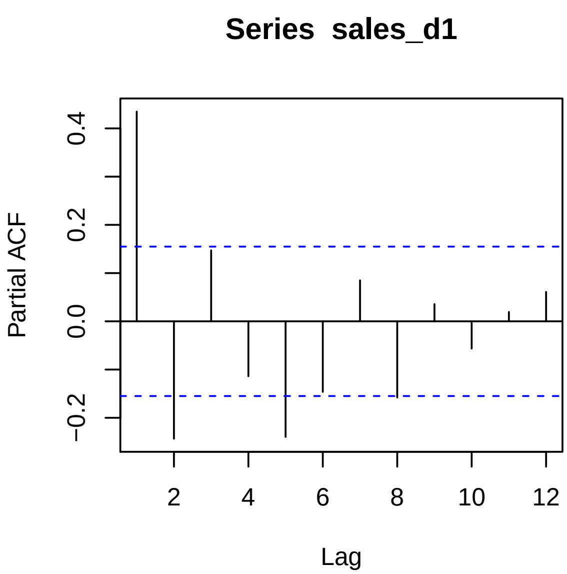
```
##
##  Box-Ljung test
##
## data:  resid_ts_2
## X-squared = 315.11, df = 12, p-value < 2.2e-16
```

This looks better, but still not a stationary process (ACF is still decreasing too slowly). We will now resort to differencing as a way to achieve stationarity.

```r
sales_d1 <- ts(diff(sales, lag = 1))

# Sample Autocorrelation Plot (ACF) / Sample Partial Autocorrelation Plots (PACF)
par(mfrow = c(2, 2))
plot(sales_d1)
points(sales_d1)
acf(sales_d1, lag.max = 12)
pacf(sales_d1, lag.max = 12)
par(mfrow = c(1, 1))
```







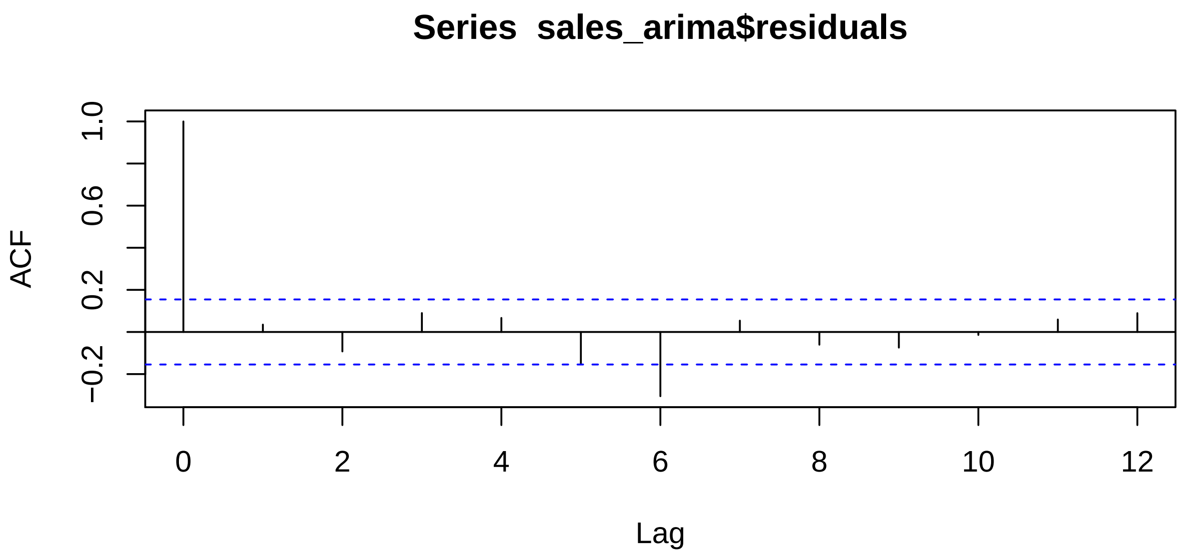

```
# Autocorrelation check for white noise
Box.test(sales_d1, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_d1
## X-squared = 59.37, df = 6, p-value = 6.044e-11

Box.test(sales_d1, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_d1
## X-squared = 68.799, df = 12, p-value = 5.369e-10
```

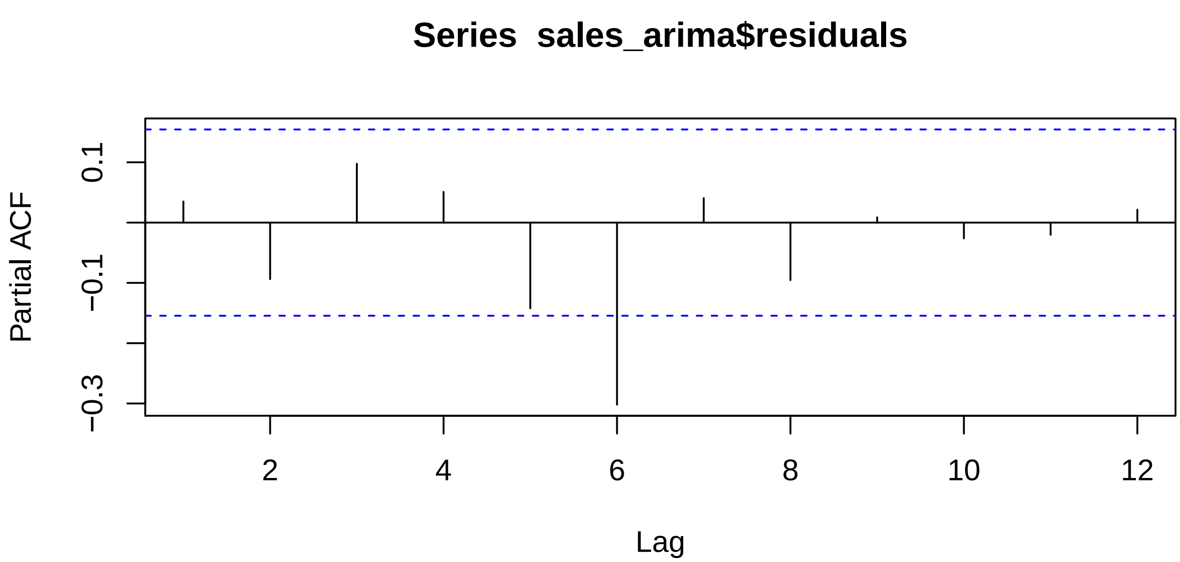
The Box-Ljung test indicates significant autocorrelation and the ACF/PACF plots show no evidence of non-stationarity. As such, we can now determine which dependence structures to fit to account for the autocorrelation.

Model 1: ARIMA(2, 1, 0)

The ACF plot has a damped exponential/sine pattern and the PACF has spikes at 1 and 2. Based on this, we will try fitting an AR(2) structure to the first-order difference time series.

```
# Note that I am using the original time series, not the differenced
# time series, and am including the differencing term within the arima statement.
sales_arima <- forecast::Arima(sales_ts, order = c(2, 1, 0),
                               include.drift = FALSE)

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

summary(sales_arima)

## Series: sales_ts
## ARIMA(2,1,0)
##
## Coefficients:
##          ar1      ar2
##       0.5425  -0.2393
## s.e.  0.0766   0.0775
##
## sigma^2 estimated as 6.134:  log likelihood=-371.3
## AIC=748.6   AICc=748.75   BIC=757.83
##
## Training set error measures:
##                     ME     RMSE     MAE       MPE     MAPE      MASE       ACF1
## Training set 0.1568165 2.453555 1.95076 0.1872017 3.635346 0.8655618 0.03509532

lmtest::coeftest(sales_arima)

##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.542516   0.076603  7.0822 1.419e-12 ***
## ar2 -0.239316   0.077529 -3.0868  0.002023 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Check to see if the residuals of the time series follow a
# "white noise" process.
par(mfrow = c(2, 1))
acf(sales_arima$residuals, lag.max = 12)
pacf(sales_arima$residuals, lag.max = 12)
```





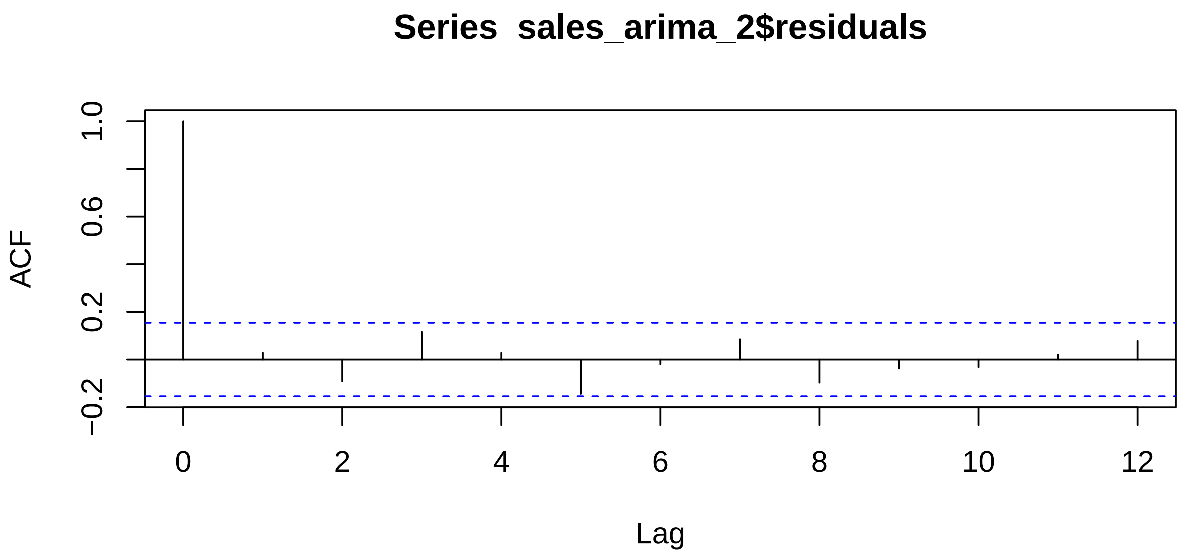

```r
par(mfrow = c(1, 1))

# Autocorrelation check of residuals
Box.test(sales_arima$residuals, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_arima$residuals
## X-squared = 23.329, df = 6, p-value = 0.0006935

Box.test(sales_arima$residuals, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
```

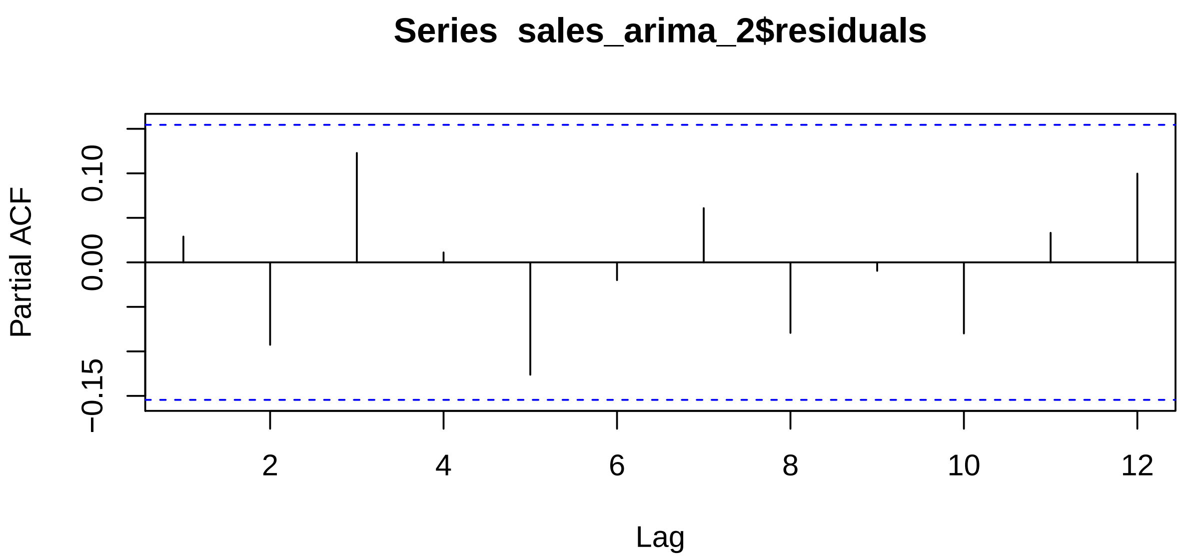
```
## data:  sales_arima$residuals
## X-squared = 27.461, df = 12, p-value = 0.006628
```

The ACF and PACF both have significant spikes at lag 6 and the Box-Ljung test indicates that there is significant evidence of autocorrelation among the residuals of the time series. Based on this, we will try adding an moving average (MA) term at lag 6 only (not lags 1-6).

Model 2: ARIMA(2, 1, (6))

```
# The "fixed = " term makes it so that the first 5 MA terms are set exactly equal
# to zero and not evaluated. The default behavior is to fit all MA terms.
sales_arima_2 <- forecast::Arima(sales_ts, order = c(2, 1, 6),
                                include.drift = FALSE,
                                fixed = c(NA, NA, 0, 0, 0, 0, 0, NA))
summary(sales_arima_2)

## Series: sales_ts
## ARIMA(2,1,6)
##
## Coefficients:
##          ar1      ar2  ma1  ma2  ma3  ma4  ma5      ma6
##       0.5339  -0.2514    0    0    0    0    0  -0.3189
## s.e.  0.0766   0.0779    0    0    0    0    0   0.0765
##
## sigma^2 estimated as 5.573:  log likelihood=-363.42
## AIC=734.84   AICc=735.1   BIC=747.14
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE     MASE      ACF1
## Training set 0.2158168 2.331114 1.840845 0.256667 3.423964 0.816792 0.0289731

lmtest::coeftest(sales_arima_2)

##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.533905   0.076557  6.9739 3.082e-12 ***
## ar2 -0.251374   0.077869 -3.2282  0.001246 **
## ma6 -0.318865   0.076519 -4.1672 3.084e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Check to see if the residuals of the time series follow a
# "white noise" process.
par(mfrow = c(2, 1))
acf(sales_arima_2$residuals, lag.max = 12)
pacf(sales_arima_2$residuals, lag.max = 12)
```






```
par(mfrow = c(1, 1))

# Autocorrelation check of residuals
Box.test(sales_arima_2$residuals, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_arima_2$residuals
## X-squared = 7.4574, df = 6, p-value = 0.2806

Box.test(sales_arima_2$residuals, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
```

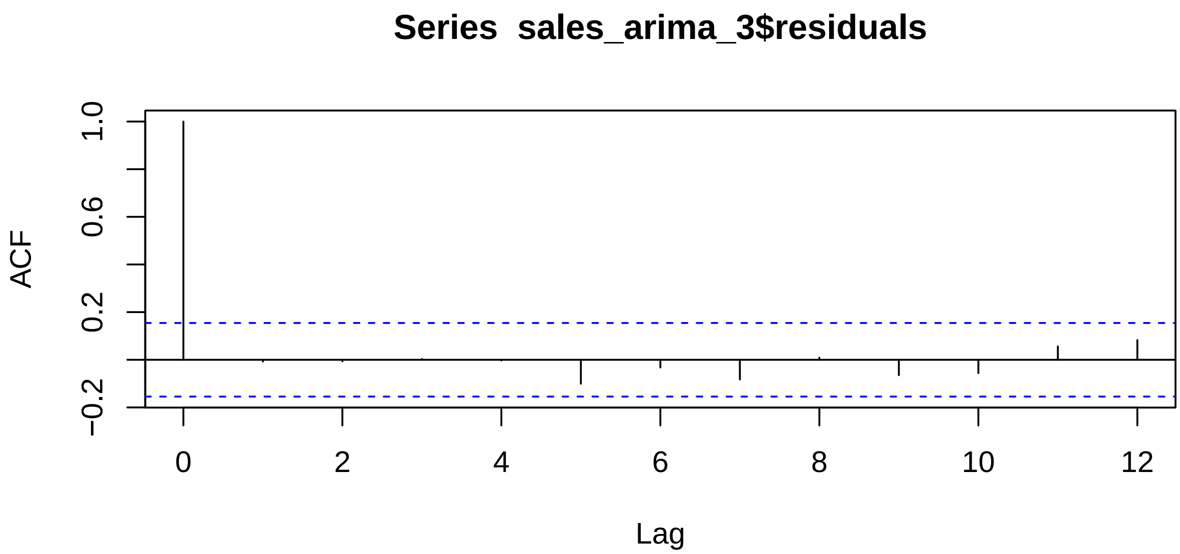
```
## data:  sales_arima_2$residuals
## X-squared = 11.88, df = 12, p-value = 0.4554
```

The addition of the MA term seems to eliminate the autocorrelation observed among the residuals.

Model 3: ARIMA(0, 1, (1, 6))

An alternative reading of the ACF/PACF plots may have suggested an MA(1, 6) model (not discussed in class). We explore that fit here and compare it to model 2.

```
sales_arima_3 <- forecast::Arima(sales_ts, order = c(0, 1, 6),
                                 include.drift = FALSE,
                                 fixed = c(NA, 0, 0, 0, 0, NA))
summary(sales_arima_3)

## Series: sales_ts
## ARIMA(0,1,6)
##
## Coefficients:
##          ma1  ma2  ma3  ma4  ma5      ma6
##       0.6471    0    0    0    0  -0.3522
## s.e.  0.0643    0    0    0    0   0.0513
##
## sigma^2 estimated as 5.029:  log likelihood=-357.33
## AIC=720.66   AICc=720.81   BIC=729.88
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.1694932 2.221447 1.749323 0.2101374 3.262424 0.7761834
##                      ACF1
## Training set -0.007535187

lmtest::coeftest(sales_arima_3)

##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1  0.647142   0.064347 10.0571 < 2.2e-16 ***
## ma6 -0.352219   0.051263 -6.8708 6.386e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Check to see if the residuals of the time series follow a
# "white noise" process.
par(mfrow = c(2, 1))
acf(sales_arima_3$residuals, lag.max = 12)
pacf(sales_arima_3$residuals, lag.max = 12)
```



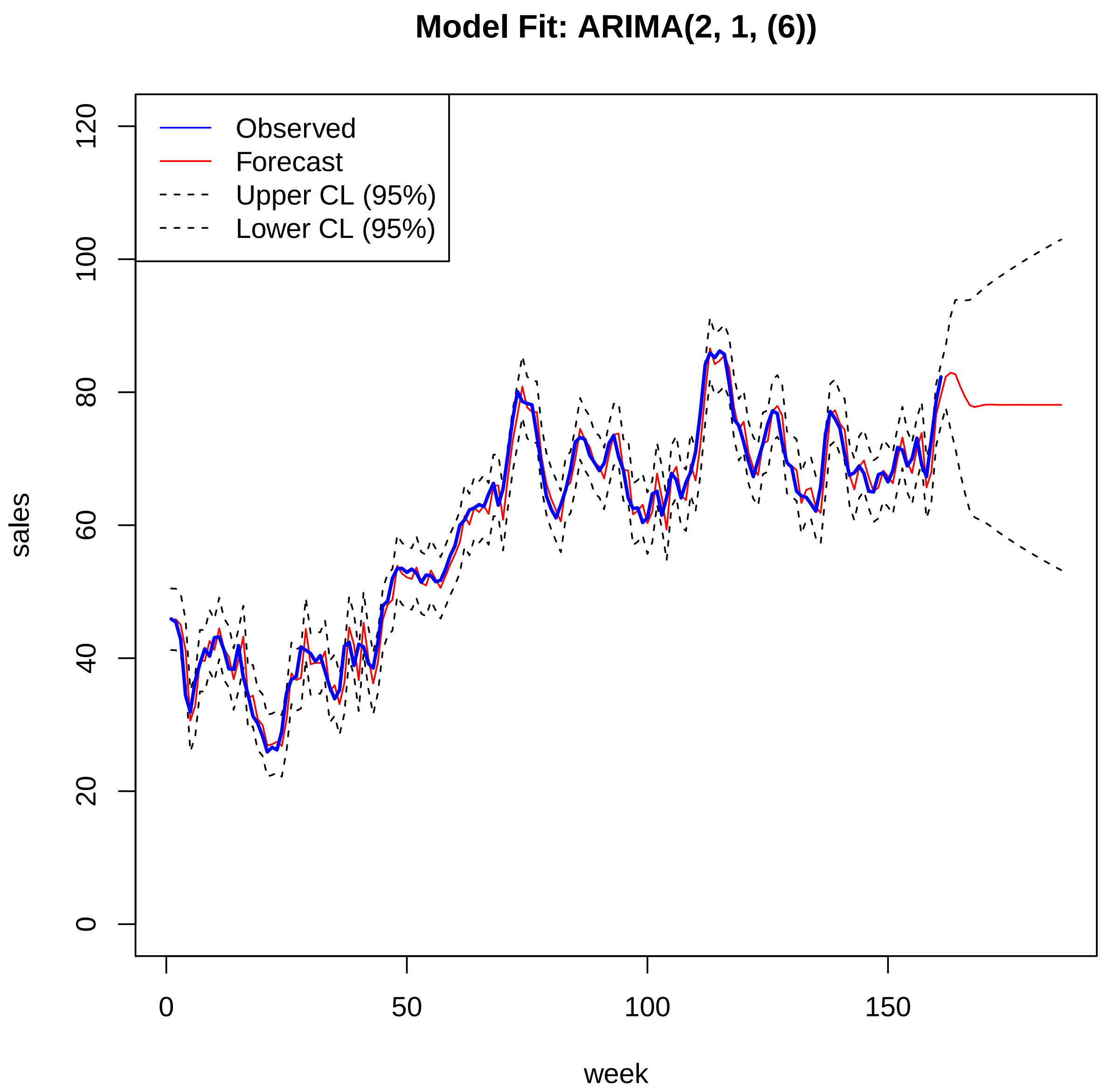

```
par(mfrow = c(1, 1))

# Autocorrelation check of residuals
Box.test(sales_arima_3$residuals, lag = 6, type = "Ljung")

##
##  Box-Ljung test
##
## data:  sales_arima_3$residuals
## X-squared = 1.9065, df = 6, p-value = 0.9281

Box.test(sales_arima_3$residuals, lag = 12, type = "Ljung")

##
##  Box-Ljung test
##
```

```
## data:  sales_arima_3$residuals
## X-squared = 6.1106, df = 12, p-value = 0.9104
```

The variance of the residuals in model 3 is slightly less than model 2. Both models 2 and 3 fully account for the spatial autocorrelation in the time series.

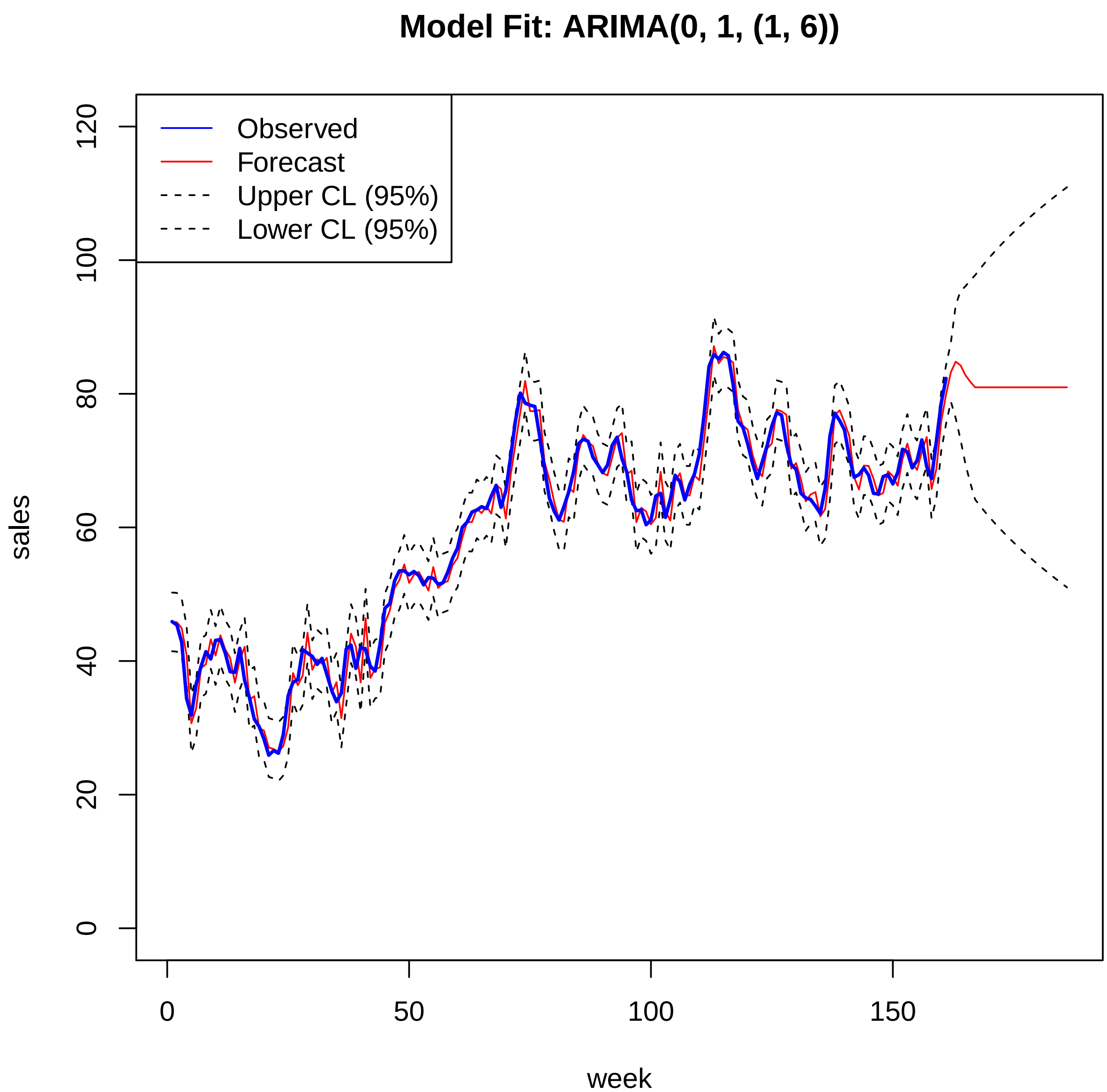
Forecast for Model 2

```
ahead_2 <- forecast::forecast(sales_arima_2, h = 25, level = 95)

# 1.96 is the z-score associated with a 95 percent confidence interval
current <- data.frame(fit = as.numeric(sales_arima_2$fitted),
                      lower = as.numeric(sales_arima_2$fitted -
                                         1.96*sqrt(sales_arima_2$sigma2)),
                      upper = as.numeric(sales_arima_2$fitted +
                                         1.96*sqrt(sales_arima_2$sigma2)),
                      week = sales_df$time)

ahead <- data.frame(fit = as.numeric(ahead_2$mean),
                    lower = as.numeric(ahead_2$lower[, 1]),
                    upper = as.numeric(ahead_2$upper[ ,1]),
                    week = seq(max(sales_df$time) + 1,
                               max(sales_df$time) + length(ahead_2$mean)))

final <- rbind(current, ahead)


plot(final$week, final$fit, col = "red", type = "l",
     xlab = "week", ylab = "sales",
     ylim = c(0, 120),
     main = "Model Fit: ARIMA(2, 1, (6))")
lines(final$week, final$lower, lty = 2)
lines(final$week, final$upper, lty = 2)
lines(sales_df$time, sales_df$sales, lwd = 2, col = "blue")
legend("topleft", legend = c("Observed", "Forecast",
                             "Upper CL (95%)",
                             "Lower CL (95%)"),
       lty = c(1, 1, 2, 2),
       col = c("blue", "red", "black", "black"))
```




```r
ahead_3 <- forecast::forecast(sales_arima_3, h = 25, level = 95)

# 1.96 is the z-score associated with a 95 percent confidence interval
current <- data.frame(fit = as.numeric(sales_arima_3$fitted),
                      lower = as.numeric(sales_arima_3$fitted -
                                    1.96*sqrt(sales_arima_3$sigma2)),
                      upper = as.numeric(sales_arima_3$fitted +
                                    1.96*sqrt(sales_arima_3$sigma2)),
                      week = sales_df$time)

ahead <- data.frame(fit = as.numeric(ahead_3$mean),
                    lower = as.numeric(ahead_3$lower[, 1]),
                    upper = as.numeric(ahead_3$upper[ ,1]),
                    week = seq(max(sales_df$time) + 1,
                               max(sales_df$time) + length(ahead_3$mean)))
```

```r
final <- rbind(current, ahead)


plot(final$week, final$fit, col = "red", type = "l",
     xlab = "week", ylab = "sales",
     ylim = c(0, 120),
     main = "Model Fit: ARIMA(0, 1, (1, 6))")
lines(final$week, final$lower, lty = 2)
lines(final$week, final$upper, lty = 2)
lines(sales_df$time, sales_df$sales, lwd = 2, col = "blue")
legend("topleft", legend = c("Observed", "Forecast",
                             "Upper CL (95%)",
                             "Lower CL (95%)"),
       lty = c(1, 1, 2, 2),
       col = c("blue", "red", "black", "black"))
```



Rough Script summarizing these results:

1. Introduce data and express desire to forecast 25 weeks.

2.
   - See need for stationarity based on time plot and SAC (p. 2).
   - Try linear trend, see remaining 2 year cycle (p. 3).
   - Try linear + trigonometric trends (p. 4). – But still see problems with 1st-order stationarity.

3. See stubbornness of time trends (p. 4), and need for differencing; first diff. appears sufficient (pp. 5-6).

4. See need for dependence structure after white noise check in first difference (p. 7).

5. Model 1: ARIMA(2,1,0), based on mixture of damped exp. decay and sine waves in SAC, and SPAC cuts off after lag 2 – note may have additional spikes at lags 5 and 6 (pp. 8-9). Goodness of fit checks: parameters significant, but model is inadequate (p. 8).

6. Model 2: ARIMA(2,1,(6)), based on spike in RSAC of Model 1 (pp. 10-11). Goodness of fit checks: no evidence of model inadequacy (pp. 8-9) (? – note Ljung-Box p-value).

7. Model 3: ARIMA(0,1,(1,6)), based on alternative reading of SAC and SPAC of first difference – on page 7, SAC spikes at lags 1 and 6, SPAC dies down in oscillating fashion. (pp. 12-13). Goodness of fit checks: no evidence of model inadequacy (pp. 12-13).

8. Compare forecasts from two 'adequate' models (pp 14-17):
   - Model 3 better only for short-term (2 week) forecasts, based on tighter confidence intervals (smaller STD for forecasts only for weeks 162-163).
   - Model 2 has tighter confidence intervals (smaller STD) for longer-term forecasts (weeks 164-186).

Conclude:

- Model 1 inadequate,
- Model 2 best for longer-term forecasts,
- Model 3 best for short-term forecasts

NOTE: One major difference between the R and SAS version of the results is that the R version does not fit an intercept term by default when dealing with differenced data. This can be specified using fit.mean = TRUE (which is the default for un-differenced data).