MSIS 5223: Tutorial 14 – Time Series Analysis: Advanced in R

## 1. Advanced Time Series

An important concept in time series is that of autocorrelation, sometimes referred to as serial dependency or serial correlation. The idea behind autocorrelation is that a previous data point may influence each subsequent data point. That is, the event is correlated with an event in the past. Exponential Smoothing models assume there is no correlation in the data. Often, however, data do exhibit autocorrelation.

When speaking about correlated time periods, the term "lag" is used. If the correlated time periods are successive, one right after the other, then the lag is 1. If the correlation occurs two time periods apart, then it is a lag of 2.

Many phenomena exhibit serial dependency. One of the best examples include stock prices. Stock prices are heavily influenced by previous evaluations. If the price was high a few days ago with an increasing trend, then the current price will more than likely be higher; if the price was decreasing, then the current price will be lower. Another example includes fly populations. The population of the past and the proliferation of breeding heavily influences the future population of the flies.

Autoregressive Integrated Moving Average (ARIMA) models are a class of models designed to take advantage of the autocorrelation. This class of models consist of three components, which include the following parameters:

- AR: autoregressive term
- I: integrated term, or the degree of differencing
- MA: moving average

These terms are typically denoted by ARIMA($p$, $d$, $q$) where $p$ is the autoregressive term, $d$ is the integrated term, and $q$ is the moving average term. Many of the models previously learned can be represented using this terminology. These are considered special cases of ARIMA.

- Mean Method = ARIMA(0, 0, 0) with constant
- Naïve Method = ARIMA(0, 1, 0)
- Drift Method = ARIMA(0, 1, 0) with constant
- Exponential Smoothing = ARIMA(0, 1, 1)

- Trend-Adjusted Exponential Smoothing = ARIMA(0, 2, 2)

The Trend-Adjusted Exponential Smoothing requires two values for the terms I and MA. This is because the method is estimating a smoothing parameter for leveling and for the trend. Look at the Exponential Smoothing method; notice it only requires a single value for the terms I and MA. The basic exponential smoothing method is only applying a smoothing parameter for leveling.

When the terms are zero values, the model is typically written with only the non-zero values. For example, for a Naïve Method, the model would be written as I(1); for exponential smoothing, it would be IMA(1, 1).

ARIMA models do not take into consideration seasonal effects. In order to do so, these models introduce three additional terms: ARIMA($p$, $d$, $q$)($P$, $D$, $Q$)$_m$ where $P$ is the autoregressive feature of the seasonal component, $D$ is the differencing feature of the seasonal component, and $Q$ is the moving average feature of the seasonal component. The value $m$ is the number of periods within each season. A familiar model previously learned can be expressed as follows:

- Holt-Winters Exponential Smoothing = ARIMA(0, 1, m+1)(0, 1, 0)m

The component, MA, will not be discussed in this tutorial because it was covered in the tutorial *Time Series Analysis Basics*. The focus of this tutorial will be on the components AR and I and the role they have in ARIMA models.

## 2. Stationary and Non-Stationary

The component I is the degree of differencing. Stationary (ARMA) and Non-Stationary (ARIMA) models differ from each other based on the presence or absence of this differencing component. Moving Average (MA) models are stationary models, those without the differencing component. Additionally, Autoregressive (AR) time series, autoregressive and moving average (ARMA) time series are also stationary models.

What exactly is a stationary model? *Stationary* data refers to a process that has a tendency to revert to the average value of a time series. This indicates that the mean and variance of the data remains constant over time. If a series of data has a trend component, the data will continually move away from the mean. This tendency increases the variance in addition to increasing the mean. *Non-stationary* time series are those that do not

exhibit constant means and variances. When determining whether to use MA, AR, or ARMA models, it is important to check data for stationarity.

To illustrate the concept of stationarity, look at Figure 1 below. This is the NYC birth data presented in the previous tutorial. The data spans a little over a decade, 1946 to 1959. This data shows the number of births per month in New York City collected for a research study. During the summer months, an increase in births occur. The data is indicative of seasonality because regular, predictable fluctuations appear every single year. Seasonal components do not have to be limited to annual fluctuations, but can occur every month, week, or even every second.

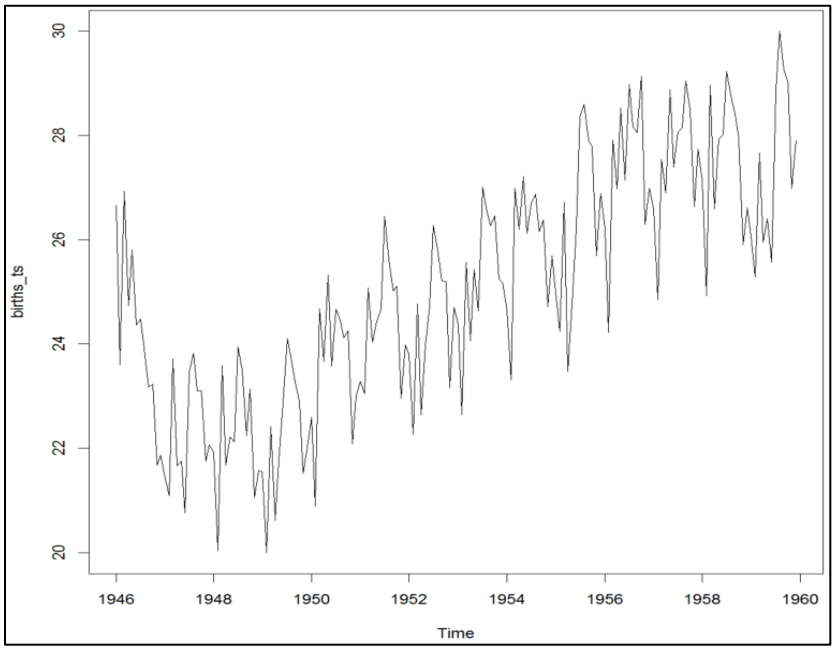


*Figure 1 NYC Births per Month*

Visually, a trend is apparent in the data. This is a strong indication the data is not stationary. To understand the effects of the trend the seasonal component is removed. Figure 2 presents the data without the seasonal component. The trend is just as apparent as before. A greater indication of a trend is to regress *births* onto *time*. First, convert the time series into a data frame and then use the function `lm()` to create the regression function. Figure 3 presents the results of the regression. The p-value for the variable *time* is highly significant, indicating the *trend* component heavily influences the time series.

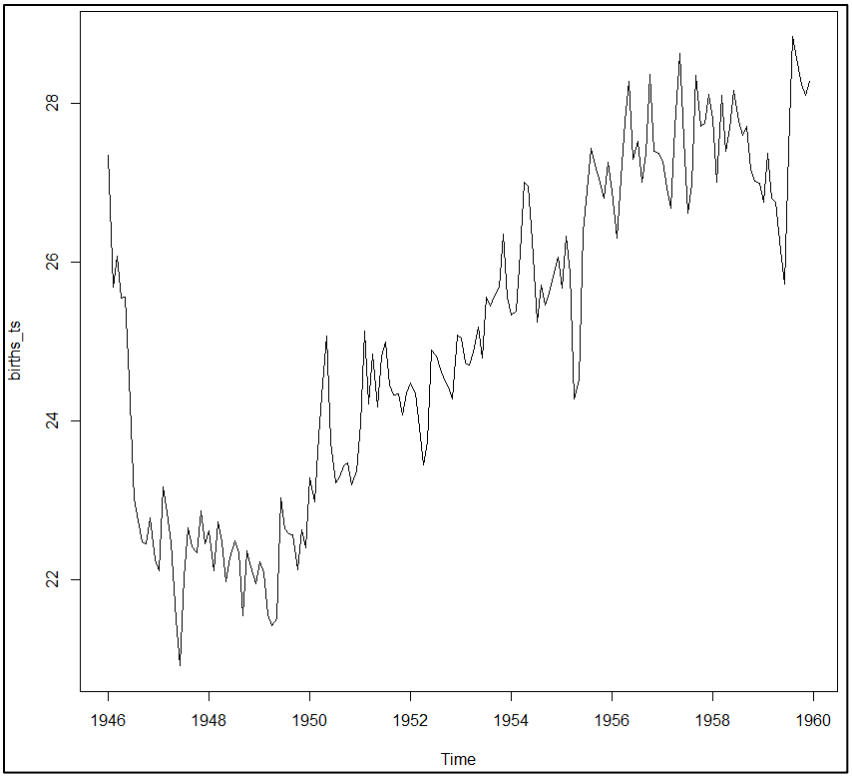


*Figure 2 NYC Data Without Seasonal Component*

```
> births_ts_dc = decompose(births_ts)
> births_trendcomp = births_ts_dc$trend
+
+ births_trend_data = data.frame(trend = c(births_trendcomp), time = c(time(births_trendcomp)))
+
+ births_trend_reg = lm(births_trend_data$trend ~ births_trend_data$time)
+ summary(births_trend_reg)

Call:
lm(formula = births_trend_data$trend ~ births_trend_data$time)

Residuals:
     Min       1Q   Median       3Q      Max
-1.01794 -0.26448  0.02224  0.27966  2.26983

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)           -954.14917   22.37207  -42.65   <2e-16 ***
births_trend_data$time    0.50134    0.01146   43.77   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

*Figure 3 Regression of Trend on Time*

To further illustrate the impact the trend has on the mean and variance, the mean of the data is calculated with and without the trend component. The mean with the trend component equals 25.06 with a variance of 4.32; the mean and variance without the trend component equals -0.04 and 0.31 respectively. Again, the definition of stationarity is the

mean and variance do not increase over time. The trend component does increase both the mean and variance of the NYC birth data.

In addition to assessing the trend to determine the existence of stationarity, several objective assessments are available to test the possibility of stationarity:

- Augmented Dickey–Fuller (ADF) t-test
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test

These tests assesses different aspects of the data, so do not rely on a single test to assess stationarity. The following figure illustrates the results of these three tests on the data. Keep in mind, the seasonal component is absent in the time series data because it was removed earlier.

```
> #Use of the Augmented Dickey–Fuller (ADF) t-test
+ adf.test(births_ts_trend, k = 20, alternative = "stationary")

    Augmented Dickey-Fuller Test

data:  births_ts_trend
Dickey-Fuller = -2.5244, Lag order = 20, p-value = 0.3574
alternative hypothesis: stationary

> #Use the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test
+ kpss.test(births_ts_trend)

    KPSS Test for Level Stationarity

data:  births_ts_trend
KPSS Level = 4.7618, Truncation lag parameter = 2, p-value = 0.01
```

*Figure 4 Assessing Stationarity*

The Augmented Dickey-Fuller (ADF) t-test uses a significant p-value to indicate stationarity. The p-value of the ADF test is not significant and indicates the time series is non-stationary; this agrees with the regression results.

The next assessment is the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, which requires a non-significant p-value to consider stationarity. The test here is significant indicating the time series data is non-stationary. The results of the three assessments—regression, ADF, KDSS—are heavily swayed toward a non-stationary model. That is, the mean and variance increase with respect to time. In order to use an ARMA model, the trend component must be removed; if the trend component remains in the time series, an ARIMA model is used.

## 3. Autoregression in Time Series

Stationary models are easier to work with because of their simplicity. As stated earlier, Moving Average and Autoregressive models require stationary data. The previous tutorial dealt with MA models, but we have yet to cover AR models. What exactly is autoregression?

Autoregression is a condition of time-ordered data in which the target variable $y_t$ is based on an explanatory variable that is a previous time value, such as $y_{t-1}$. The subscript *t* is the time period where *t* is the current time period and *t-1* is the previous time period; *t-3* represents three time periods ago, *t-4* is four time periods ago, etc.

| Period | Demand (*t*) | Demand (*t-1*) | Demand (*t-2*) |
|---|---|---|---|
| 1 | 15 | - | - |
| 2 | 28 | 15 | - |
| 3 | 30 | 28 | 15 |
| 4 | 33 | 30 | 28 |
| 5 | 35 | 33 | 30 |
| 6 | 41 | 35 | 33 |

*Table 1 Lagged Values for t*

Table 1 illustrates the differences of the lag values. The first column presents the actual demand over time. The second column presents the lag value of a single period at *t-1*. The last column gives a lag value at *t-2*. As an example, look at Period 4, the highlighted row. The current value is 33 for demand. The demand at *t-1* is 30 or the value for Period 3. For *t-2*, the value is 28 or the value for Period 2. In a time series model, if the lag of a target variable is included as an explanatory variable, the explanatory variable is a *lagged dependent variable*. As an equation, this is written as follows:

$$y_t = \beta_1 y_{t-1} + \varepsilon_t$$

where $\beta_1$ is some slope and $\varepsilon_t$ is some random error. Simply put, the current time value is based on the previous time value. The above equation is the most basic version of an autoregressive model, or AR(1). Again, the value of the current time period is based on the previous time period allowing for some random error.

A concept related to autoregression is autocorrelation or serial dependency.[1] That is, the value at $y_t$ is correlated with the value at $y_{t-1}$. This is a first-order autocorrelation.

[1] Note, that serial dependency is not the same as serial correlation. Serial correlation is a condition in which the error terms, $\varepsilon_t$ and $\varepsilon_{t-1}$, exhibit autocorrelation. Serial correlation is not covered in this text.

When $y_t$ is correlated with $y_{t-2}$, this is a second-order autocorrelation. Time series data can contain higher-order autocorrelation beyond first- and second-order, such as lags of 20 or even 30!

For data in which the previous two terms contribute to the current time period, the model is written as AR(2). This type of model exhibits first-order autocorrelation and second-order autocorrelation. It should be noted that it is possible to have a weak first-order autocorrelation and still have AR(2). This type of model is written as follows:

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \varepsilon_t$$

Another possibility for AR(2) models is the existence of a second-order autocorrelation without a first-order autocorrelation. This means the first component of the equation is 0 and removed from the equation like so:

$$y_t = \beta_2 y_{t-2} + \varepsilon_t$$

This presents a problem in that AR(2) can represent both versions of autocorrelation, one with a first-order autocorrelation and one without. Recall that an ARMA model can be written as ARMA($p$, $q$). Using this notation, a time series with a first-order autocorrelation and second-order autocorrelation can be written as ARMA($p$=1,2) (or AR($p$=1,2) for short) while a time series without a first-order autocorrelation and with second-order autocorrelation can be written as ARMA($p$=2). An extended example of this is a time series with only a $11^{th}$-order and $20^{th}$-order lag effect can be written as ARMA($p$=11,20).

At this point you may be wondering how you determine which order of autocorrelations exist in a time series or whether the time series has a moving average component. The process used in this course is Box-Jenkins approach, a three-step process:

1. Identification: Identify the appropriate autoregressive and moving average components within the time series
2. Estimation: Estimate the parameters of the AR and MA components within the time series
3. Diagnostic Checking: Assess the model for quality using fit measures or error tests

*Identification of Components*

Identifying the correct autoregressive and moving average components within a time series is perhaps the most time intensive part of the Box-Jenkins approach. This methodology requires the data scientist to assess alternatives prior to selecting this best fit model. This process utilizes the autocorrelation function (ACF) and the partial correlation function (PACF).

Throughout the process, the principal of *parsimony* guides the selection of parameters. Parsimony is the idea that the model with the fewest number of parameters is preferred. For example, assume you discover these two alternatives during this process:

- AR($p$=1,3,4)
- ARMA($p$=1, $q$=2)

The first model contains three parameters: a first-, third-, and fourth-order autocorrelation. The second model only contains two parameters: a first-order autocorrelation and a second-order moving average. Adhering to the principal of parsimony, you would select the second model because it has the fewest number of parameters.

What happens when you discover two models with the same number of parameters, but those parameters are different types? For example, consider the following two models:

- ARMA($p$=2, $q$=1)
- AR($p$=2,4)

Both models contain exactly the same number of parameters: two. At this point, either model would work equally well. Assuming no problems are encountered while estimating the components, the diagnostic checking may reveal which of the two models fits the data better. If one model has an overall better fit, then select that model for forecasting.

The ACF produces a correlogram that displays the correlations of previous time periods with the current time period $y_t$. That is, the correlations $Corr_1(y_t, y_{t-1})$, $Corr_2(y_t, y_{t-2})$, $Corr_3(y_t, y_{t-3})$, $Corr_4(y_t, y_{t-4})$, etc. are shown where $Corr_1(y_t, y_{t-1})$ is the correlation between the current time period and the previous time period, $Corr_2(y_t, y_{t-2})$ is the correlation between the current time period and 2-time periods ago, $Corr_3(y_t, y_{t-3})$ is the correlation between the current time period and 3-time periods ago, and so forth.

The following assessment is done using the unemployment data from Maine that was used in the previous tutorial. To illustrate the concept of stationarity, the data is assessed without removing any seasonal or trend effects. Figure 5 presents the plot of the ACF. Each vertical line represents a lag effect. The horizontal, blue, dashed line represents the 95% confidence interval. Any lag effect that extends beyond either boundary needs consideration.

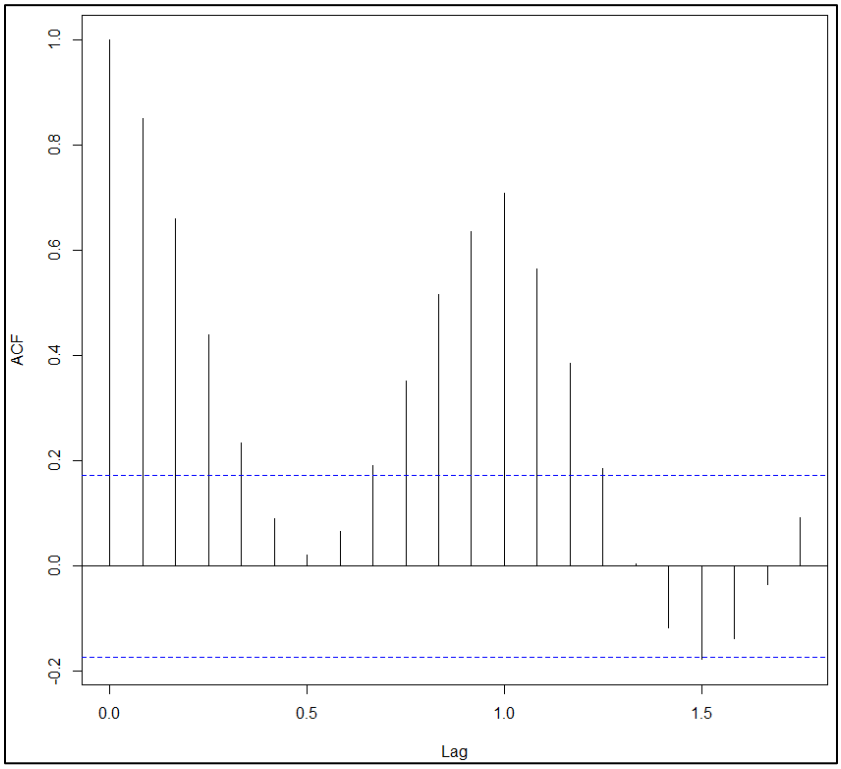


*Figure 5 ACF of Unemployement Data*

Notice the pattern presented in Figure 5 is seasonal. The plot starts out at the top and curves down toward the midline at 0.5, curves back up at 1.0, then down again. These oscillations are indicative of seasonal effects. This would indicate the time series is non-stationary. Performing a ADF test results in a p-value of 0.5, which means the model is not stationary; the KPSS results in a p-value of 0.04, also indicating the model is not stationary.

The seasonal effect is removed from the model and the autocorrelation function is plotted in Figure 6. The difference between Figure 5 and Figure 6 is apparent: Figure 6 reveals no oscillations and is free from a seasonal effect. With the seasonal component removed, the trend effect is readily apparent. A trend effect is characterized by

decreasing (or increasing) lag effects that start out high and decrease, not dramatically, but steadily toward zero without reaching zero until the end. This is another indication that the data is not stationary.

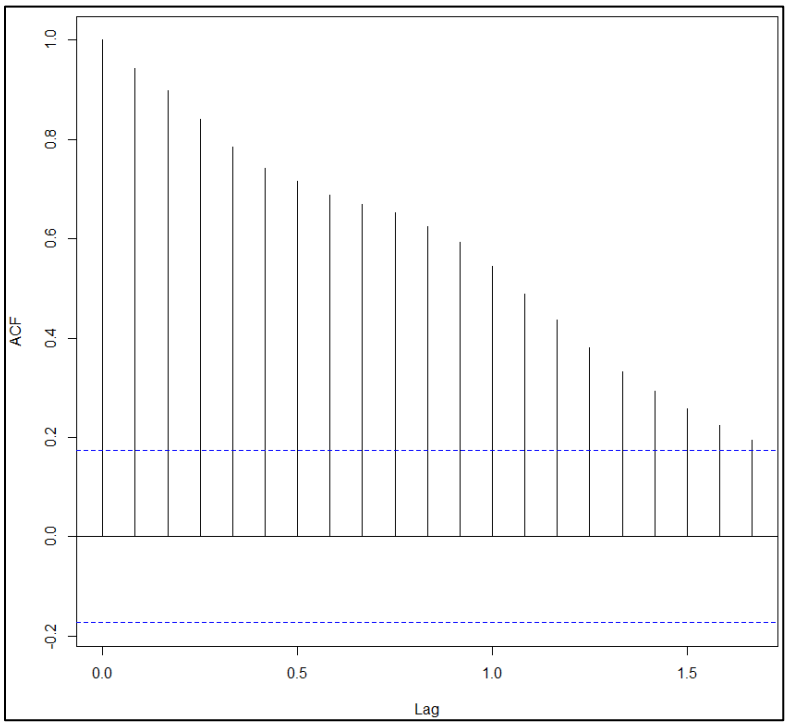


*Figure 6 ACF of Unemployment Data: Without Seasonal Effect*

Below is a plot of the time series without the seasonal component. The data exhibits two trends, one decreasing from 1996 to about 2001, and the other increasing from 2001 to 2006. Simply removing the trend component from the data like removing the seasonal component will not be enough. The removal of this trend effect requires differencing, or the I in ARIMA. For now, we will not discuss what differencing is; the discussion will come after this section. To difference a time series, use the diff() function like so:

```
maine_ts_diff1 = diff(maine_ts_trend, differences=1)
```

where *maine_ts_trend* is the time series object and *differences* is the argument specifying the number of differencing components to apply.

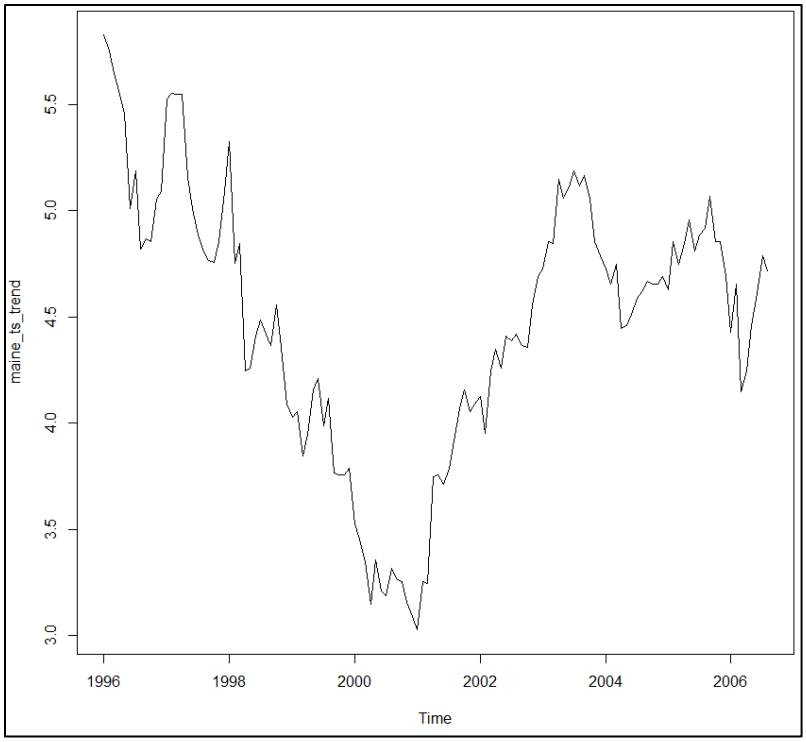


*Figure 7 Plot of Unemployment: Without Seasonal Effect*

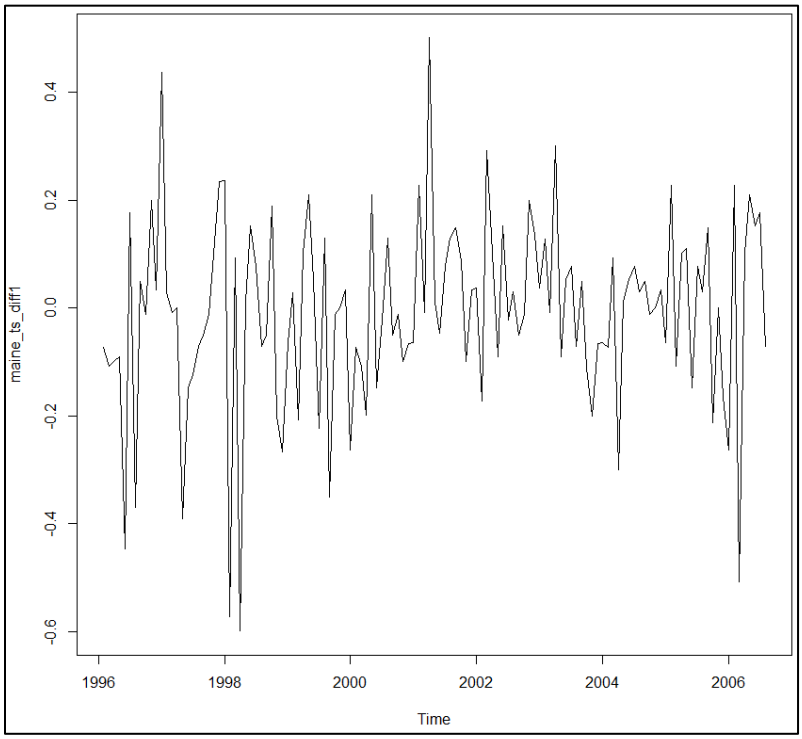


*Figure 8 Unemployment without Trend via Differencing*

Compare Figure 7 with Figure 8. It is already apparent that the trend effects of 1996-2001 and 2001-2006 are missing. This is a good indication that the time series is

approaching stationarity. The Augmented Dickey-Fuller test results in a non-significant value, so it indicates it is still not stationary. The KPSS results in a non-significant p-value, so it indicates the model is stationary. These conflicting results may indicate that the model still has some residual trend effects.

The differencing function is applied again, but this time with two differencing components instead of one. The results of the ADF test and the KPSS test both indicate the model is stationary. Now the autocorrelation function can be viewed using the autocorrelation function `acf()` within R. Compare Figure 9 below with those of Figure 5 and Figure 6. It is apparent that Figure 9 does not possess a seasonal effect or a trend effect. As stated earlier, if only the first 2 or 3 lag effects extend beyond the 95% confidence interval, then that is an indication of stationarity.

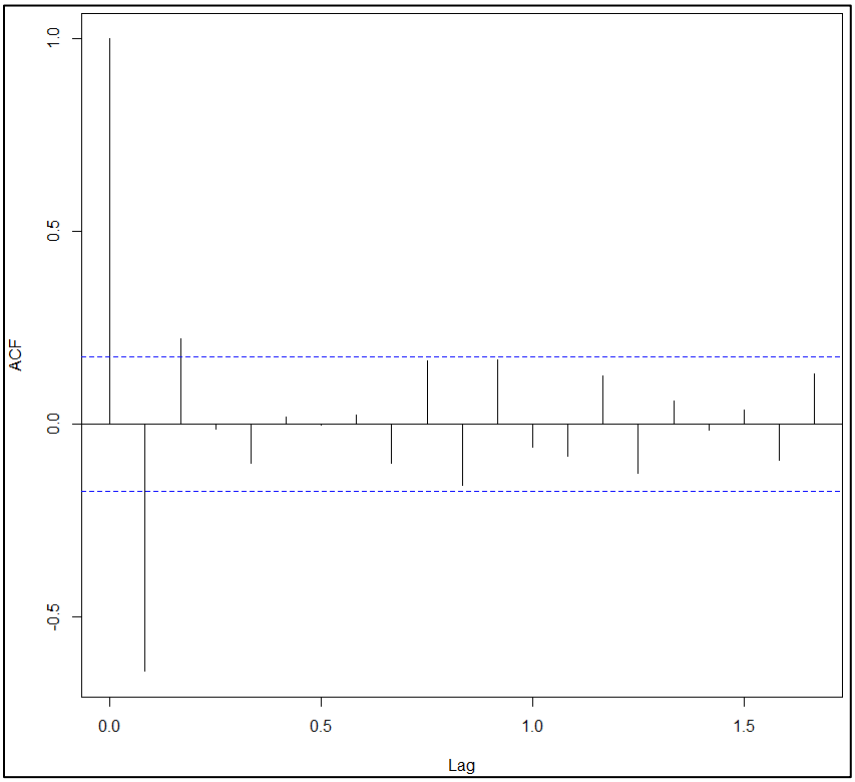


*Figure 9 ACF of Unemployement: No Seasonal or Trend Effects*

With all this talk about stationarity, we have yet to discuss autocorrelation using the autocorrelation function. Again, the main purpose in using the ACF is to determine the number of components for AR and MA. The following criteria are a general guideline when using a correlogram:

- If the correlogram dies down gradually, then that indicates an AR component.
- If the correlogram cuts off dramatically, then that indicates a MA component.
- If the correlogram dies down gradually, then it may indicate an ARMA model.

Figure 9 presents an abrupt end to the lag effects greater than the 95% confidence interval, which may indicate the model has at least a single moving average component. It is possible, however, that the model has an autocorrelation component because the first three lag effects extend beyond the confidence interval. Performing a PACF will further clarify the potential effects.

The PACF is used to assess partial correlations. What is a partial correlation? Partial correlations are indirect correlations. In a time series, you can assume that $y_t$ and $y_{t-2}$ are correlated because $y_t$ and $y_{t-1}$ are correlated, and $y_{t-1}$ and $y_{t-2}$ are correlated. The PACF plots these partial correlations. The function in R is `pacf()`.

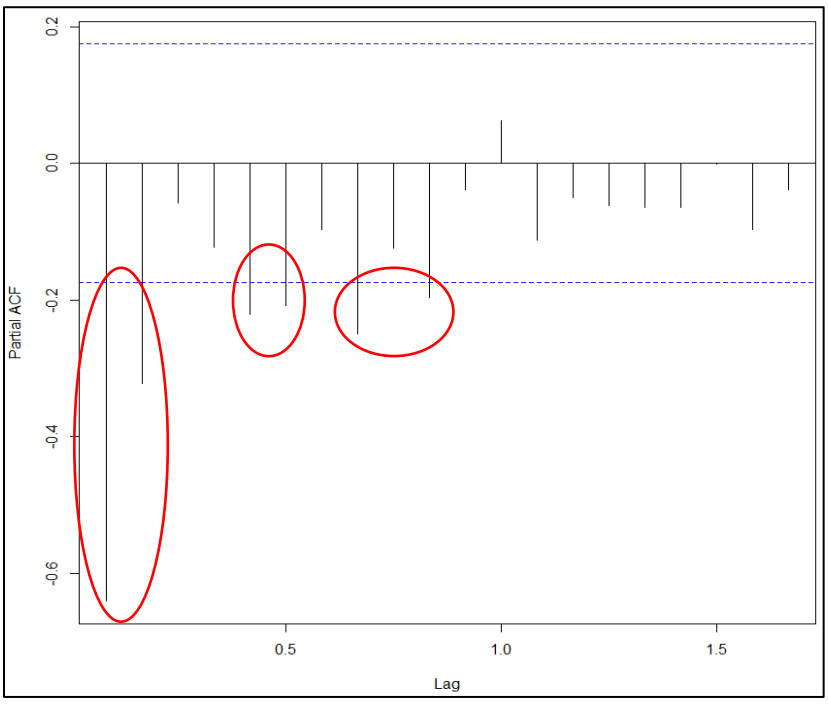


*Figure 10 Partial Correlation Function Plot*

Figure 10 presents the plot of the partial correlation function for the unemployment time series. Notice several of the lag effects extend beyond the 95%

confidence interval. The red circles on the PACF indicate possible effects due to autocorrelation or moving average. The following criteria are a general guideline when using a partial correlogram:

- If the partial correlogram dies down gradually, then that may indicate a MA component.
- If the partial correlogram cuts off dramatically, then that may indicate an AR component.
- If the partial correlogram dies down gradually, then it may indicate an ARMA model.

When determining the types and numbers of components, it is important to take into consideration both the ACF and PACF plots at the same time. The following alternative models are possibilities:

- ARMA(1, 1): The correlogram “dies down” while the partial correlogram also dies down with multiple effects.
- ARMA(2, 0): Since the partial correlogram dies off dramatically after the first two lags, then the effects can be attributed to autocorrelation effects.
- ARMA(2, 1): A combination of the two previous models, which may also include an ARMA(3, 1) model.

The principal of parsimony would dictate that the ARMA(1, 1) and ARMA(2, 0) models should be selected because they have the fewest number of parameters. This doesn’t necessarily mean those two perform better over the ARMA(2, 1) model. In order to decide on a model, the components must be estimated and a diagnostic routine must be performed assessing model fit and error.

An important note should be made here. As the model underwent differencing twice, the model would have the component I(2). Thus, if the ARMA(2, 1) model is chosen, the final model would be ARIMA(2, 1, 1).

To help illustrate the differences inherent within the correlogram due to the effects of autocorrelation or moving average components, the following simulated ARIMA models are run:

- ARIMA(2, 0, 0)
- ARIMA(0, 0, 2)

- ARIMA(2, 0, 2)

These simulations are run using the arima.sim() function for R. This function contains many arguments and the two most important are model and n where the former specifies the type of ARIMA constructed and the latter is the number of time points to simulate. For example, for the first simulation of ARIMA(2, 0, 0), the code would appear as follows for 50 time periods and autocorrelation components of 0.9 and -0.2:

```
sim_ts1 = arima.sim(model = list(ar = c(0.9, -0.2)), n = 50)
```

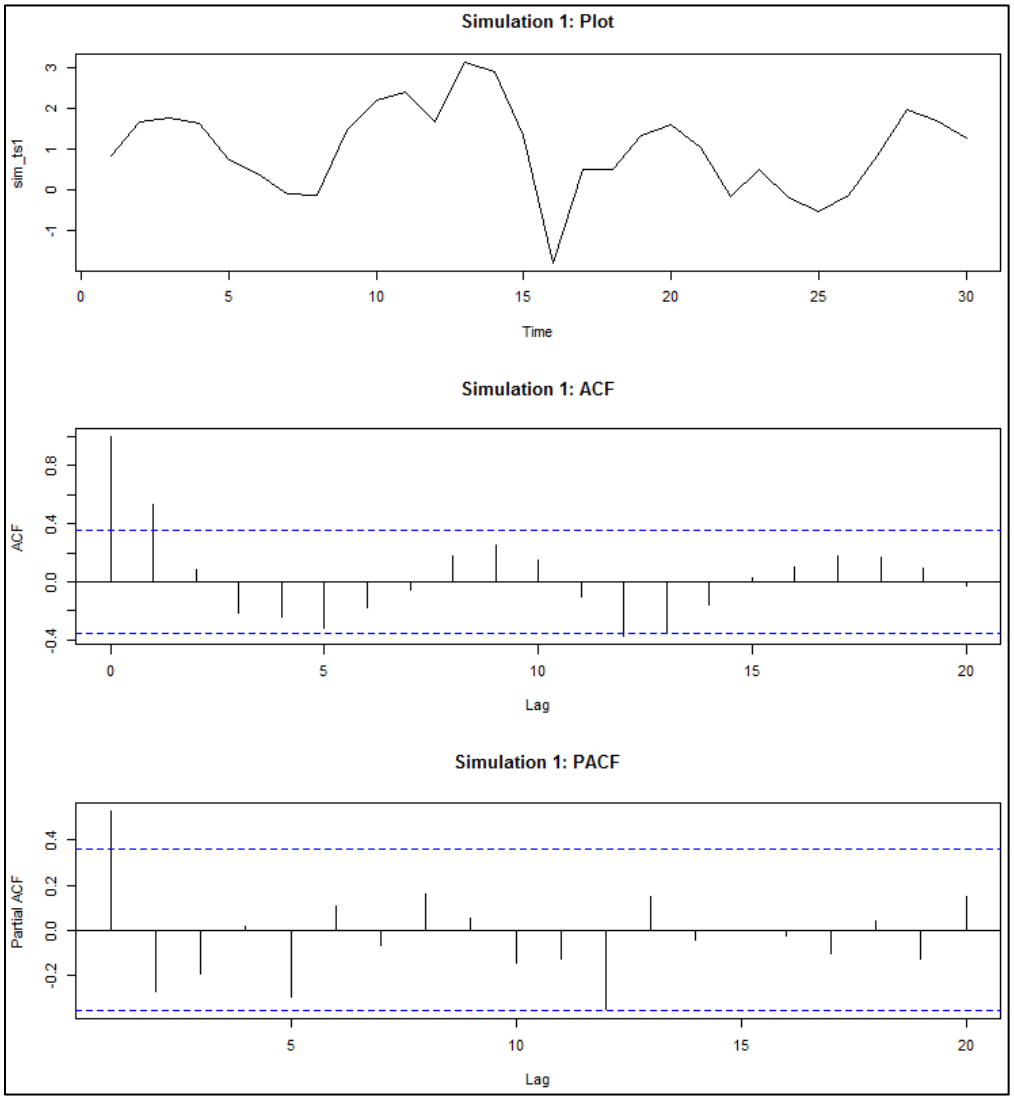


*Figure 11 Simulation 1 Plot, ACF, PACF*

Figure 11 presents the plot of Simulation 1, the autocorrelation function, and the output of the partial correlation function. Notice that the ACF only display 2 lags above the 95% confidence interval while the PACF displays a single element extending above

the confidence interval. While many possibilities exist to illustrate an AR(2) model, this provides an excellent example. The next two figures present the results of the second and third simulation models. For Simulation 2, notice that two lags extend over the significance line, but not consecutively one right after the other. For Simulation 3, the PACF reveals 4 lags extending beyond the line, but they are not entirely consecutive.

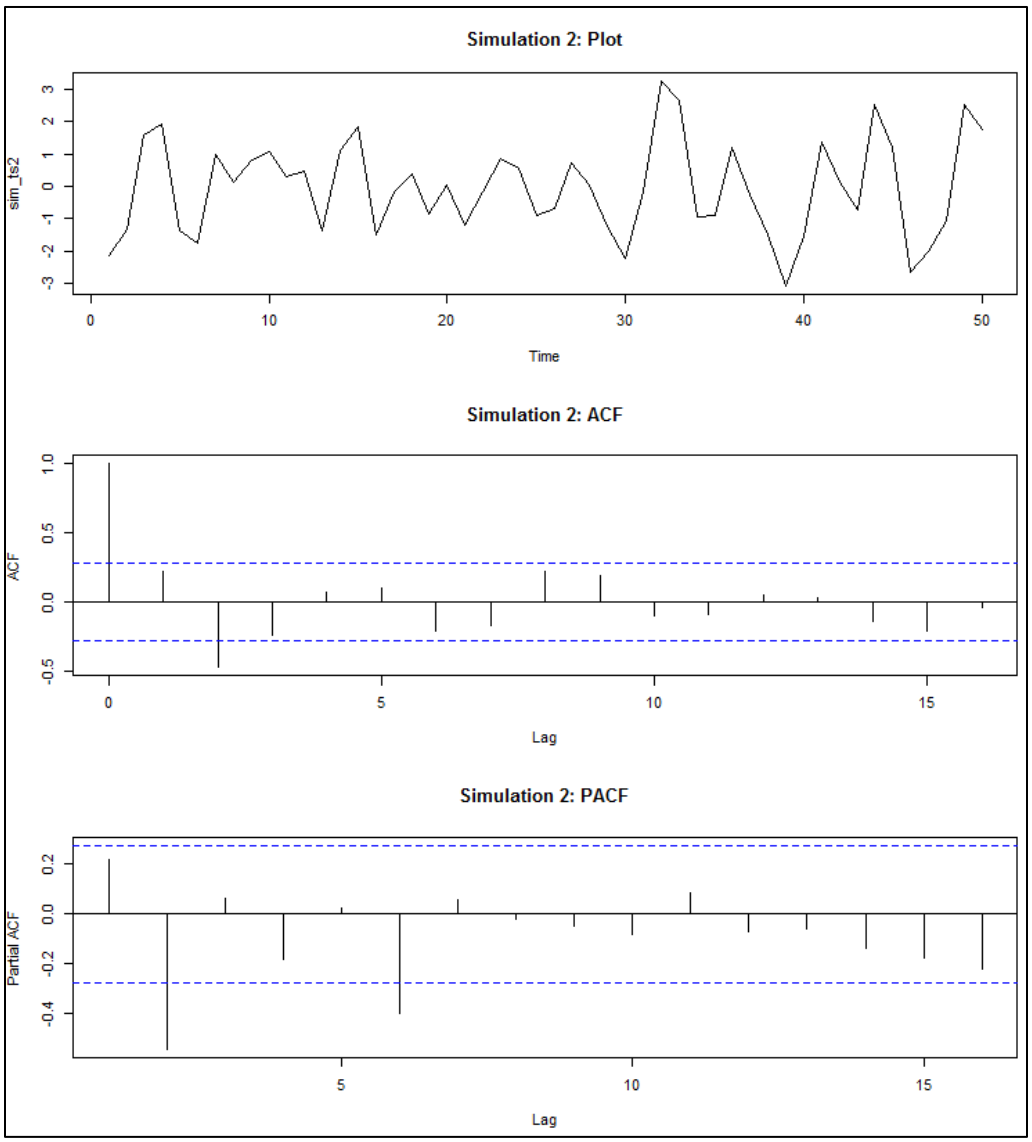


*Figure 12 Simulation 2 Plot, ACF, PACF*

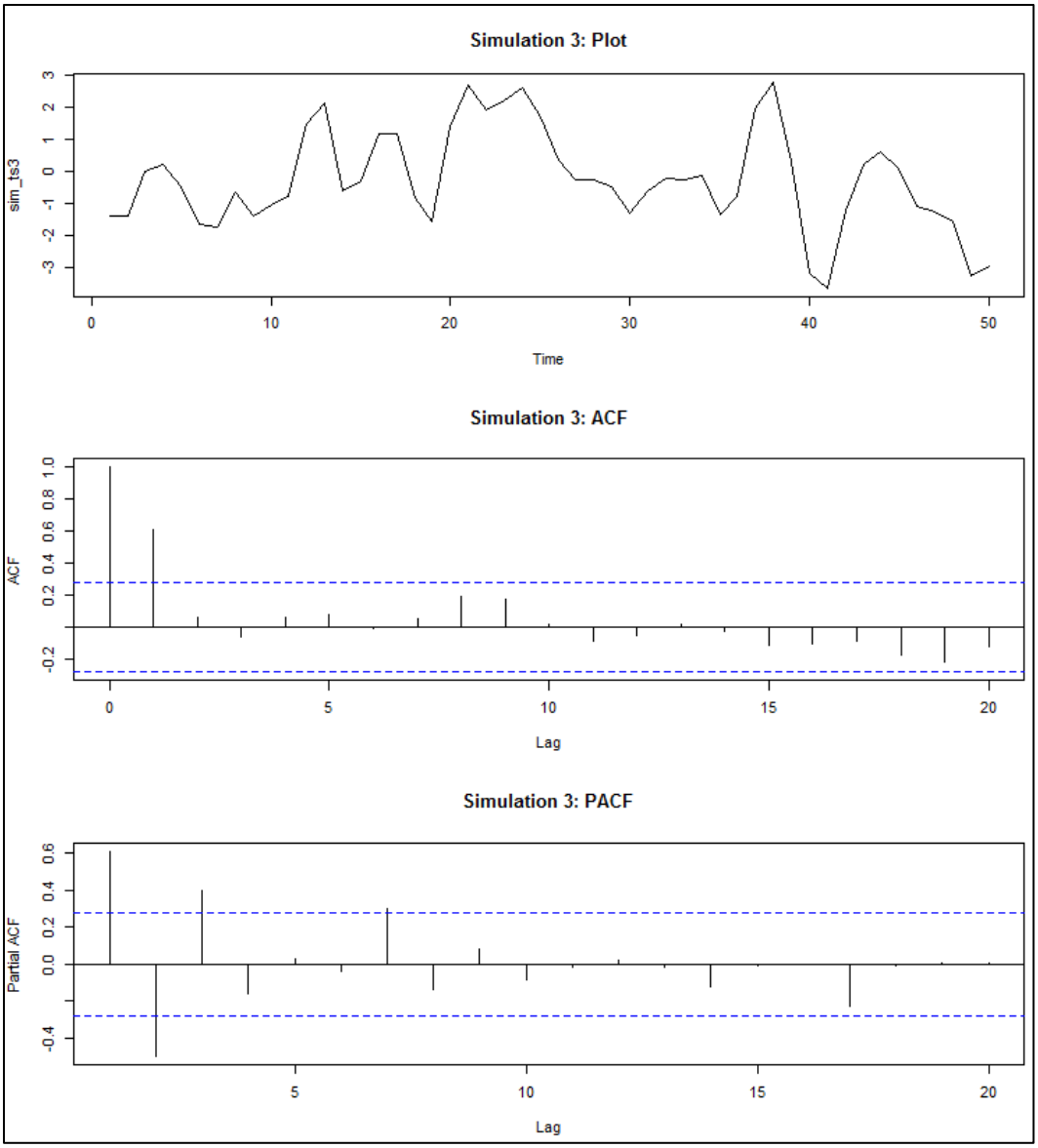


*Figure 13 Simulation 3 Plot, ACF, PACF*

### *Estimation of Components*

The purpose of estimating the model is to obtain the coefficients for the various parameters being estimated. Much like regression, the coefficients provide the necessary elements to build a model. In the case of time series, the parameters are used to forecast future values. In addition to parameter estimation, the proposed models are tested for goodness of fit to determine which alternative is the best. Typical estimators include AIC, BIC, MSE, MFE, MAD, and many others.

To obtain an estimation of parameters within R, simply use the arima() function. In the previous section, the result of analyzing the correlograms for the unemployment data indicated four possible models: ARMA(1, 1), ARMA(2, 0), ARMA(2, 1), and ARMA(3, 1). The next step is to estimate each model and determine the best fit. The figure on the right presents the four alternatives and their output from R.

```
> maine_arima1 = arima(maine_ts_diff2, order = c(1, 0, 1))
> maine_arima1

Call:
arima(x = maine_ts_diff2, order = c(1, 0, 1))

Coefficients:
         ar1    ma1  intercept
      -0.131  -1.00      0.001
s.e.   0.088   0.03      0.000

sigma^2 estimated as 0.0303:  log likelihood = 38.95,  aic = -69.9
> maine_arima2 = arima(maine_ts_diff2, order = c(2, 0, 0))
> maine_arima2

Call:
arima(x = maine_ts_diff2, order = c(2, 0, 0))

Coefficients:
         ar1     ar2  intercept
      -0.853  -0.326      0.001
s.e.   0.084   0.084      0.008

sigma^2 estimated as 0.0367:  log likelihood = 29.02,  aic = -50.04
> maine_arima3 = arima(maine_ts_diff2, order = c(2, 0, 1))
> maine_arima3

Call:
arima(x = maine_ts_diff2, order = c(2, 0, 1))

Coefficients:
         ar1    ar2     ma1  intercept
      -0.105  0.179  -1.000      0.001
s.e.   0.088  0.088   0.025      0.000

sigma^2 estimated as 0.0294:  log likelihood = 41,  aic = -72
> maine_arima4 = arima(maine_ts_diff2, order = c(3, 0, 1))
> maine_arima4

Call:
arima(x = maine_ts_diff2, order = c(3, 0, 1))

Coefficients:
         ar1    ar2    ar3     ma1  intercept
      -0.116  0.188  0.066  -1.000      0.001
s.e.   0.089  0.089  0.089   0.024      0.000

sigma^2 estimated as 0.0293:  log likelihood = 41.27,  aic = -70.55
```

*Figure 14 Estimation of Four Models for Unemployment*

Each of the four models is represented in the output. Within the code, the argument used to specify the parameters is order=c(*p*,*d*,*q*). Look at the first model, the ARMA(1,1). The code indicates the model specifications by order=c(1,0,1). Notice the differencing parameter is included as zero in the code; a zero value must be included for any of the three parameter types if no parameter exists.

To select the best model, select the Akaike Information Criterion (AIC) with the smallest value is desirable. The first two models have the largest values or those closest to a positive value while the last two have the smallest values. Note, in the fourth model an additional autoregressive parameter is added compared to the third model; this results in a smaller AIC value. Keep in mind that the principle of parsimony demands a simple model and adding complexity won't necessarily increase model fit.

Based on the results of this estimation, the third model with an autoregressive parameter of 2 and moving average parameter of 1 is the best fitting model for this given data. Remember, though, that the model was differenced twice. While the results of this model indicate an ARMA(2, 1), the reality is this data is truly an ARIMA(2, 2, 1). If you difference a model, always be sure to indicate the necessary differencing parameters.

In addition to checking the model fit, it is important to check the significance of the estimated coefficients. Using the coeftest() function from the library lmtest, the coefficients from the arima object are assessed (see the figure below). The first autoregressive component is not significant while the second one is; the moving average parameter is significant. These results may indicate an additional component is missing. A return to the ACF and PACF can help shed further light on what may be missing. Again, while it is possible to add an additional parameter (for example, the PACF indicates a lag effect of 8 and 10), keep in mind the principle of parsimony.

```
> coeftest(maine_arima3)

z test of coefficients:

           Estimate Std. Error z value Pr(>|z|)
ar1       -0.105458   0.087869   -1.20    0.230
ar2        0.179359   0.087886    2.04    0.041 *
ma1       -0.999999   0.025185  -39.71   <2e-16 ***
intercept  0.000616   0.000455    1.35    0.176
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.'
```

*Figure 15 P-Values for Coefficients*

### *Diagnostic Checking*

The last part of the Box-Jenkins approach is to perform diagnostic checks to determine the goodness of fit. This process involves further assessment of the ACF and PACF correlograms and comparing fit measures. Luckily, the AIC was already provided during the previous estimation exercise. To further assess fit, Bayesian Information Criterion can be used.

The following code in R is used to obtain the values of BIC for each of the proposed models:

```
> maine_arima1_bic = AIC(maine_arima1, k = log(length(maine_ts_diff2)))
+ maine_arima2_bic = AIC(maine_arima2, k = log(length(maine_ts_diff2)))
+ maine_arima3_bic = AIC(maine_arima3, k = log(length(maine_ts_diff2)))
+ maine_arima4_bic = AIC(maine_arima4, k = log(length(maine_ts_diff2)))
+
> maine_arima1_bic
+ maine_arima2_bic
+ maine_arima3_bic
+ maine_arima4_bic
+
[1] -58.55023
[1] -38.69245
[1] -57.82316
[1] -53.52838
```

*Figure 16 BIC Calculation in R*

Similarly to AIC, a small value of BIC is preferable in order to fit the best model to the data. The first and third models present the smallest values of BIC, -58.55 and -57.82 respectively. Despite the first model having the best fit based on BIC, it should be remembered that all the previous work on estimation should be considered at this point; thus, model three is still the best option.

While AIC or BIC are used to assess goodness of fit, other assessments are available. Additionally, AIC does not check for errors based on forecasts. The goal of reducing errors in forecasting is to obtain an accurate and unbiased forecast. This helps reduce the cost created by forecasting over or under actual results. In the next major section below, forecasting accuracy will be covered.

### *Automating the Process*

This entire section is dedicated to developing ARIMA models. One of the difficulties in such modeling is the subjective process. Without a lot of domain knowledge and experience, determining the best model may be extremely difficult. Luckily, R has a built-in function that provides a way to automate the entire process.

The forecast library contains the function auto.arima(). The below example takes in the differenced data and outputs the suggested parameters for the ARIMA components. The results of the output indicate that the data suggests an ARMA(2, 1) model. More specifically, the model suggests ARMA($p$=1,2, $q$=1). The AIC and BIC fit measures are provided and are both low.

```
> auto.arima(maine_ts_diff2)
Series: maine_ts_diff2
ARIMA(2,0,1) with zero mean

Coefficients:
          ar1     ar2      ma1
      -0.0995  0.1841  -0.9789
s.e.   0.0909  0.0907   0.0309

sigma^2 estimated as 0.03092:  log likelihood=40.19
AIC=-72.39   AICc=-72.06   BIC=-61.04
```

*Figure 17 Automated ARIMA Modeling*

While this function is very handy and can save time, I suggest you still assess the correlogram and partial correlogram much like you would in a manual process. Like many black-box types of solutions, having software dictate an answer to you doesn't always provide you the best solution for your real-world problem.

## 4. Forecasting Assessment

Forecast error, which is the difference between actual demand and the forecast, evaluates the accuracy of a forecasting model. For all measures of forecast accuracy, the closer the measure is to zero, the better the forecast. The equation for forecast error is given below:

$$Error(E_t) = Actual(A_t) - Forecast(F_t)$$

Two classes of measures exist for forecast error: absolute measures of forecast error and relative measures of forecast error. Absolute measures, or scaled measures, are expressed in the same unit of measurement as the data; relative measures provide a perspective of the magnitude of the forecast error relative to actual demand. As an example, consider two time series where one has a demand of 100 and the other has a demand of 1000. If both have an absolute forecast error of 10, then the forecasted amounts are 90 and 990 respectively:

Time Series 1: 100 – 90 = 10 (90%)

Time Series 2: 1000 – 990 = 10 (99%)

An absolute forecast error of 10 is better when the actual demand is 1000 than when the actual demand is 100; that is, 99% accuracy is better than 90%.

Many versions of exist for absolute and relative measures. The following are common absolute measures used to evaluate forecast accuracy:

- Running Sum of Forecast Error (RSFE) (measures bias)
- Mean Forecast Error (MFE) (measures bias)
- Mean Absolute Error (MAE) (measures accuracy)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

A common relative measure for evaluating forecasting accuracy is Mean Absolute Percentage Error (MAPE).

### *Running Sum of Forecast Error and Mean Forecast Error*

Both of these measure the average magnitude or size of the forecast errors (i.e., central tendency) or bias. *Bias* represents the tendency of a forecast to be consistently higher or lower than the actual demand. A positive RSFE (and MFE) indicates that the forecasts generally are low—the forecasts underestimate demand and stock-outs may occur, for example. A negative RSFE (and MFE) indicates that the forecasts generally are high—the forecasts overestimate demand. A zero RSFE (and MFE) indicates that the forecast is unbiased; however, a zero RSFE (and MFE) value does not imply that the forecast was necessarily accurate, since a forecast could have very large negative and positive errors and still have a zero bias. It should be noted that Mean Forecast Error is sometimes referred to as simply Mean Error.

The Running Sum of Forecast Error is expressed as follows:

$$RSFE = \sum (A_t - F_t) = \sum E_t$$

The Mean Forecast Error is based on RSFE where

$$MFE = {}^{RSFE}/_{n} = \sum {}^{E_t}/_{n}$$

### *Mean Absolute Error*

The Mean Absolute Error measures the average magnitude of the forecast errors without regard to direction of error. This measure is sometimes referred to as the Mean Absolute Deviation or MAD. While this is a common usage, it is not correct. A MAE value greater than zero indicates the forecast either overestimates or underestimates demand. A zero MAE indicates that the forecast exactly predicted demand over the entire

evaluation period. In other words, positive and negative errors do not cancel out (as with MFE), thus MAE is accurate.

The Mean Absolute Error is expressed as

$$MAE = \sum {|E_t|}/{n}$$

### *Mean Squared Error and Root Mean Squared Error*

These measures are sensitive to large errors. In general, models that yield forecasts with many small errors and a few very large ones are not desirable. They measure the squared forecast error or the error variance and recognizes that large errors are disproportionately more "expensive" than small errors. They are used to compare errors across forecasting techniques that use the same sample.

Why are larger errors more "expensive" than smaller ones? Take the number 2 for example. When squared, the result is 4. This is a relatively small jump; thus, if the error between actual demand and forecasted demand, the resulting error would not be as detrimental as larger error. A larger amount of error, say 34, when squared, results in error of 1,156. See the table below for an illustration of how increasing error size results in greater squared values.

| Error | Error$^2$ |
|---|---|
| 2 | 4 |
| 4 | 16 |
| 6 | 36 |
| 8 | 64 |
| 10 | 100 |
| 12 | 144 |

*Table 2 Illustration of Squared Errors*

The equation for Mean Squared Error is

$$MSE = \sum E_t^2/n$$

The Mean Squared Error is considered the variance of the error estimation because it uses the same units of measurement as the square of the data being estimated. Taking the square root of this value yields the standard deviation of the estimated error, or the Root Mean Squared Error.

The RMSE is expressed as follows:

$$RMSE = \sqrt{MSE}$$

*Mean Absolute Percentage Error*

The Mean Absolute Percentage Error is a relative measure for evaluating forecast accuracy. It standardizes each forecast error with respect to the actual demand and measures the average relative magnitude of the forecast errors, expressed as a percentage. It is similar to MAE, except measures deviation as a percentage of actual data. It is expressed as

$$MAPE = \left(\sum |E_t/A_t|\right)(100/n)$$

*Example of Using Forecast Errors*

Consider the following data using a two-period simple moving average forecast. The following forecasting errors are assessed: RSFE, MAE, MSE, and MAPE.

| **Period** | **Demand** | **Forecast** | **$E_t$** | **$\|E_t\|$** | **$E_t^2$** | **$\|E_t/A_t\|$** |
|---|---|---|---|---|---|---|
| 1 | 100 | - | | | | |
| 2 | 80 | - | | | | |
| 3 | 70 | 90 | -20 | 20 | 400 | 0.29 |
| 4 | 90 | 75 | 15 | 15 | 225 | 0.17 |
| 5 | 80 | 80 | 0 | 0 | 0 | 0 |
| | | SUM= | **-5** | **35** | **625** | **15.33%** |

Look at RSFE which results in a value of -5. A negative value indicates a bias that, on average, overestimates the forecast. The MAE has a value of 35, which is a strong indication that forecasting error exists in the model. MSE results in a value of 625, and when specific values are assessed, the first forecasted value is very large at 400; an indication that the forecast is too large for the given demand.

The last value, MAPE, is not very useful for a forecast model in isolation. Again, it is a relative measure and allows you to compare across different samples of data. Still, it can provide some guidance on the idea of the relative magnitude of error present.

*Using R to Assess Forecasting Error*

Within the library forecast, R provides assessments for accuracy and bias. The function accuracy() provides the following measures of error: Mean Error, Root Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error, Mean Absolute Scaled Error, and Autocorrelation of Errors at Lag 1 (ACF1).

Recall that the model decided upon for the unemployment data is the ARIMA(2, 2, 1), or for the differenced data it is ARMA(2, 1). Three other models were proposed as alternatives to the final selection: ARMA(1, 1), AR(2), and ARMA(3, 1). The figure below presents the forecasted models along with the error measures from R.

```
> maine_arima1_fore = forecast(maine_arima1, h = 20)
+ maine_arima2_fore = forecast(maine_arima2, h = 20)
+ maine_arima3_fore = forecast(maine_arima3, h = 20)
+ maine_arima4_fore = forecast(maine_arima4, h = 20)
+
> accuracy(maine_arima1_fore)
+ accuracy(maine_arima2_fore)
+ accuracy(maine_arima3_fore)
+ accuracy(maine_arima4_fore)
                     ME   RMSE    MAE   MPE  MAPE   MASE   ACF1
Training set 0.007181 0.1741 0.1282 133.6 262.8 0.4404 0.0145
                      ME   RMSE    MAE   MPE  MAPE   MASE     ACF1
Training set -0.0001751 0.1916 0.1458 154.7 243.8 0.5005 -0.02847
                     ME   RMSE   MAE   MPE  MAPE   MASE     ACF1
Training set 0.005828 0.1715 0.128 153.8 245.2 0.4394 -0.02046
                     ME   RMSE    MAE   MPE  MAPE   MASE     ACF1
Training set 0.005394 0.1712 0.1273 148.7 241.1 0.4372 0.002079
```

*Figure 18 Error Measures for Unemployment Data*

The bias in the models is minimal as indicated by the small values of Mean Error and Root Mean Squared Error. Both are important to check because while MFE provides the degree of bias, the RMSE highlights (i.e. squared values) any major errors inherent in the forecast. The values for MAE are also low, indicating minimal error.

An interesting observation of these results shows the fourth model, ARMA(3, 1), as having less error than the chosen ARMA(2, 1). Most likely this is due to the fourth model having an additional autoregressive component which allows it to cover more variance. This does not mean the fourth model is better; remember, in the previous section it was determined that the third model performed the best based on estimation, diagnostics, and parsimony.