Week 7 Topics

1. Chapter 7 – Regression-Based Models

Autocorrelation

# **Introduction to Autocorrelation**

Auto correlation is a characteristic of data which shows the degree of similarity between the values of the same variables (e.g. *Y*t) over successive time intervals.

When you have a series of numbers, and there is a pattern such that values in the series can be predicted based on preceding values in the series, the series of numbers is said to exhibit autocorrelation. This is also known as serial correlation and serial dependence. The existence of autocorrelation in the residuals of a model is a sign that the model may be unsound. Autocorrelation is diagnosed using a correlogram (ACF plot) and can be tested using the Durbin-Watson test.

The auto part of autocorrelation is from the Greek word for self, and autocorrelation means data that is correlated with itself, as opposed to being correlated with some other data. Consider the nine values of Y in figure 1. The column to the right shows the last eight of these values, moved “up” one row (or moved down one row), with the first value deleted (i.e. 0.400). When we correlate these two columns of data, excluding the last observation that has missing values, the correlation is 0.64. This means that the data is correlated with itself (i.e., we have autocorrelation/serial correlation).

|  |  |
| --- | --- |
| t | Y |
| 1 | 0.400 |
| 2 | 0.397 |
| 3 | 0.157 |
| 4 | -0.083 |
| 5 | -0.243 |
| 6 | -0.323 |
| 7 | -0.243 |
| 8 | -0.083 |
| 9 | 0.077 |
| 10 | 0.347 |

Original lag 1 (up) lag-1 (down)

|  |  |  |
| --- | --- | --- |
| t | Y | lag 1 |
| 1 | 0.397 | 0.157 |
| 2 | 0.157 | -0.083 |
| 3 | -0.083 | -0.243 |
| 4 | -0.243 | -0.323 |
| 5 | -0.323 | -0.243 |
| 6 | -0.243 | -0.083 |
| 7 | -0.083 | 0.077 |
| 8 | 0.077 | 0.347 |
| 9 | 0.347 |  |

|  |  |  |
| --- | --- | --- |
| t | Y | lag 1 |
| 1 | 0.400 |  |
| 2 | 0.397 | 0.157 |
| 3 | 0.157 | -0.083 |
| 4 | -0.083 | -0.243 |
| 5 | -0.243 | -0.323 |
| 6 | -0.323 | -0.243 |
| 7 | -0.243 | -0.083 |
| 8 | -0.083 | 0.077 |
| 9 | 0.077 | 0.347 |
| 10 | 0.347 |  |

OR

Figure 1: Time dependent dataset with lag 1

In more academic term, we can explain autocorrelation in the following paragraphs

*Repetitive time series data (observations), have the characteristics of repetitive events, in which the fundamental component of the observation repeats over time. This phenomenal is called autocorrelation. Autocorrelation, also known as serial correlation, is the correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them. The analysis of autocorrelation is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies. It is often used in signal processing for analyzing functions or series of values, such as time domain signals.*

Different fields of study define *autocorrelation* differently, and not all these definitions are equivalent. In some fields, the term is used interchangeably with *autocovariance*.

Unit root processes, trend stationary processes, autoregressive processes, and moving average processes are specific forms of processes with autocorrelation.

The autocorrelation (Box and Jenkins, 1976) function can be used for the following two purposes:

1. To detect non-randomness in data.
2. To identify an appropriate time series model if the data is not random.

# **Autocorrelation of stochastic (Random) processes**

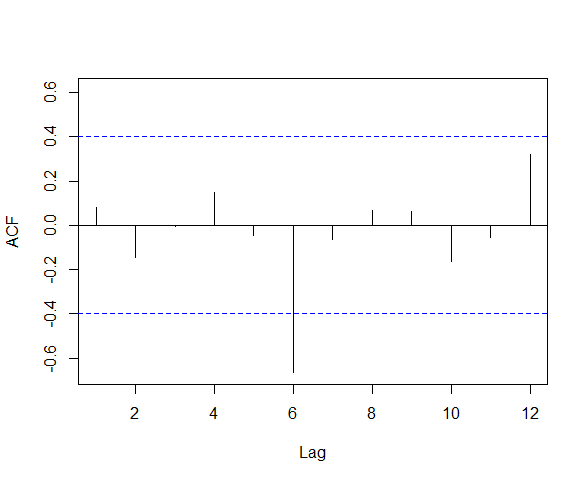
In statistics, the autocorrelation of a real or complex random process is the *Pearson* correlation between values of the process at different times, as a function of the two times or of the time lag. Let {Yt} be a random process, and *t* be any point in time (*t* may be an integer for a discrete-time process or a real number for a continuous-time process). Then, Yt is the value (or realization) produced by a given run of the process at time t. Suppose that the process has mean and variance at time t, for each t. Then the definition of the autocorrelation function between times is:

Where *E* is the expected value operator, and the bar represents complex conjugate.

# **Computing Autocorrelation**

Given the observations, the mean of *Yi* then, lag k autocorrelation function is defined as:

Figure 2 shows a sample of Amtrak ridership with lag 1 and lag 2 observations.

In R we can use Acf from “forecast” package to compute the autocorrelation of ***lag k*** of a dataset. Figure 3 show the plot of autocorrelation lags 1-12 of 23 months’ observation and their values.

|  |  |  |  |
| --- | --- | --- | --- |
| Month | Ridership | Lag 1 Series | Lag 2 Series |
| 1/1/91 | 1708.917 |  |  |
| 2/1/91 | 1620.586 | 1708.917 |  |
| 3/1/91 | 1972.715 | 1620.586 | 1708.917 |
| 4/1/91 | 1811.665 | 1972.715 | 1620.586 |
| 5/1/91 | 1974.964 | 1811.665 | 1972.715 |
| 6/1/91 | 1862.356 | 1974.964 | 1811.665 |
| 7/1/91 | 1939.86 | 1862.356 | 1974.964 |
| 8/1/91 | 2013.264 | 1939.86 | 1862.356 |
| 9/1/91 | 1595.657 | 2013.264 | 1939.86 |
| 10/1/91 | 1724.924 | 1595.657 | 2013.264 |
| 11/1/91 | 1675.667 | 1724.924 | 1595.657 |
| 12/1/91 | 1813.863 | 1675.667 | 1724.924 |
| 1/1/92 | 1614.827 | 1813.863 | 1675.667 |
| 2/1/92 | 1557.088 | 1614.827 | 1813.863 |
| 3/1/92 | 1891.223 | 1557.088 | 1614.827 |
| 4/1/92 | 1955.981 | 1891.223 | 1557.088 |
| 5/1/92 | 1884.714 | 1955.981 | 1891.223 |
| 6/1/92 | 1623.042 | 1884.714 | 1955.981 |
| 7/1/92 | 1903.309 | 1623.042 | 1884.714 |
| 8/1/92 | 1996.712 | 1903.309 | 1623.042 |
| 9/1/92 | 1703.897 | 1996.712 | 1903.309 |
| 10/1/92 | 1810 | 1703.897 | 1996.712 |
| 11/1/92 | 1861.601 | 1810 | 1703.897 |
| 12/1/92 | 1875.122 | 1861.601 | 1810 |

Figure 3: Autocorrelation function output for lag 1-12

Figure 2: Lag 1 and 2 autocorrelations for Amtrak ridership sample dataset

Strong positive or negative (high absolute values) autocorrelation at multiple lags larger than 1, clearly reflect a cyclical (repetitive) pattern. Therefore, strong positive autocorrelation at lag 12, 24, 36,… in monthly dataset, will reflect an annual seasonality.

# **Positive and negative autocorrelation**

The example in figure 1 shows positive first-order autocorrelation, where lag 1 (or first order) indicates that observations that are one apart are correlated, and positive means that the correlation between the observations is positive. When data exhibiting positive first-order correlation is plotted, the points appear in a smooth snake-like curve, as on the graph in figure 4. With negative first-order correlation, the points form a zigzag pattern if connected

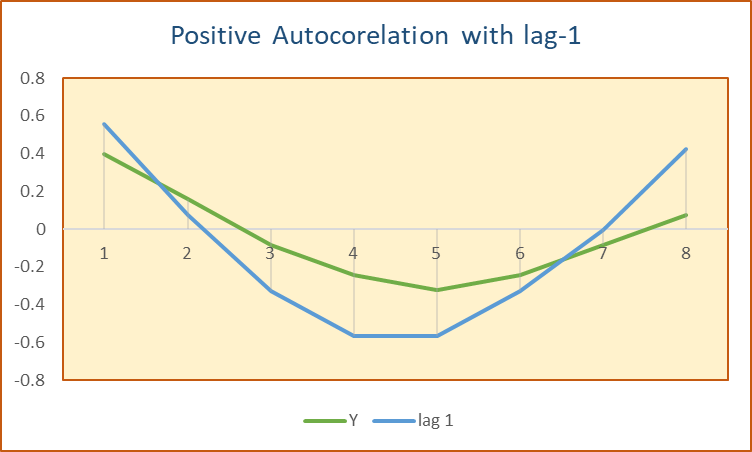


Figure 4: plot of figure 1 dataset with positive lag 1

Textbook defines lag 1 positive autocorrelation as when in a series, consecutive values move generally in the same direction. Therefore, a strong linear trend must have strong lag 1 autocorrelation.

# **The implications of autocorrelation**

When autocorrelation is detected in the residuals from a model, it suggests that the model is mis-specified (i.e., in some sense wrong). A cause is that some key variable or variables are missing from the model. Where the data has been collected across space or time, and the model does not explicitly account for this, autocorrelation is likely. For example, if a weather model is wrong in one suburb, it will likely be wrong in the same way in a neighboring suburb. The fix is to either include the missing variables, or explicitly model the autocorrelation (e.g., using an ARIMA model). See textbook page 146 (lower paragraphs)

The existence of autocorrelation in residuals means that computed standard errors, and consequently p-­values, are misleading.

Autoregressive Model (AR)

Autoregression is a time series model that uses observations from previous time steps as input

to a regression equation to predict the value at the next time step.

It is a very simple idea that can result in accurate forecasts on a range of time series problems.

In this section, I will show you how to implement an autoregressive model for time series.

Previous values -> Timeseries Mode [regression equation] -> next value

### Autoregression

A regression model, such as linear regression, generates an output value based on a linear combination of input values.

For example:

(1)

Where is the prediction (estimation in statistics), *b0* and *b1* are coefficients found by optimizing the model on training data, and *X* is an input value.

This technique can be used on time series where input variables are taken as observations at previous time steps, we call it lag (check the autocorrelation) variables.

For example, we can predict the value for the next time step (t) given the observations at the last two time steps (t-1 and t-2). As a regression model, this would look as follows:

(2)

(3)

Because the regression model uses data from the same input variable at previous time steps, it is referred to as an autoregression (regression of self).

## Step1:

Now, we are going back to autocorrelation. We know an autoregression model assumes that the observations at previous time steps are useful to predict the value at the next time step. Interestingly, if all lag variables show low or no correlation with the output variable, then we can conclude the time series problem may not be predictable. This information can be very useful when getting started on a new dataset. Therefore, we plot autocorrelation

## Step 2:

We plot the correlation coefficient for each lagged variable. This can very quickly give an idea of which lag variables may be good candidates for use in a predictive model and how the relationship between the observation and its historic values changes over time.

## Step 3

After selecting the number of observations from the past which hold all dataset components, we perform the following

1. Building the model on training dataset
2. Test the model on the validation dataset
3. And forecasting for one or more period.

# **Autoregressive (AR) as a Second-Layer Model**

Although ARIMA model is better model compared to AR, but AR can be used as a supporting model for short-term forecasting. We can capture the autocorrelation by constructing a second-layer (SL) forecasting model for the residual. Here are steps according to your textbook.

1. Generate k-step ahead forecast of the series (Ft+k), using a forecasting method (e.g. linear regression)
2. Generate k-step- ahead forecast of the forecast error (et+k)
3. Improve the initial k-step-ahead forecast of the time series dataset by adjusting it according to forecast error. Then, we have the *Improved* \*Ft+k = Ft+k + et+k

You can read more about this approach on page 148, 149 of the your textbook

Below, I provide codes for the figure 7-3 and 7-4 which is the application of the AR as a Second-Layer Model. Follow the code to see how this model is built and used. The data set is “Amtrak Data.csv”.

By following the above steps, we come up with the residual function of:

### From our work with Amtrak dataset, we have the and for t = April 2001

Calculating the error for April 2001 using the following:

* ridership = 2023,792 actual value
* ridership = 2004,271 predicted value.
* Error for the t-1 = March 2001

### (AR) as a Second-Layer Model: R-Code for figure 7-3 and AR(1) Second-Layer:

library(forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

ridership.ts <- ts(Amtrak.data$Ridership, start = c(1991, 1), end = c(2004, 3), freq = 12)

nValid <- 36

nTrain <- length(ridership.ts) - nValid

train.ts <- window(ridership.ts, start = c(1991, 1), end = c(1991, nTrain))

valid.ts <- window(ridership.ts, start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid))

train.lm.trend.season <- tslm(train.ts ~ trend + I(trend^2) + season)

# Figure 7-3

Acf(train.lm.trend.season$residuals, lag.max = 12, main = "")

Acf(train.lm.trend.season$residuals, lag.max = 12, main = "")$acf

[,1]

[1,] 1.00000000

[2,] 0.60405883

[3,] 0.44983171

. . .

# At lag-1 the acf is 0.60405883.

train.lm.residual<-(train.lm.trend.season$residuals)

plot(train.lm.residual)

ar.burg(x=train.lm.residual, aic = FALSE, order.max = 1)

Call:

ar.burg.default(x = train.lm.residual, aic = FALSE, order.max = 1)

Coefficients:

1

0.6043

Order selected 1 sigma^2 estimated as 2829

train.lm.trend.season.pred <- forecast(train.lm.trend.season, h = nValid, level = 0)

# the predicted value for April 2001 is 2004.271

# **Autoregressive Model summary**

An autoregression model is a linear regression model that uses lagged variables as input variables.

We could calculate the linear regression model manually using MS Excel, R, or other statistical tools.

Once fit, we can use the model to make a prediction by calling the predict() function for a number of observations in the future. For example a 7-day period forecast

Visit the <https://astrostatistics.psu.edu/su07/R/html/stats/html/ar.html> to learn how autoregressive model is built. Once the model is built you can use multiple linear regression approach to predict a future observation’s value. I prepared an assignment for week 7 which is based on AR model.

However, there is a more powerful forecasting model based on autocorrelation. The ARIMA model which is an improvement to AR. Next, we are going to learn about ARIMA.

1. Chapter 7 – ARIMA Models

ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

Before we introduce ARIMA models, we must first discuss the concept of stationarity and the technique of differencing time series.

Stationarity and differencing

# **Stationarity**

### A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time.

### Some cases can be confusing — a time series with cyclic behavior (but with no trend or seasonality) is stationary. This is because the cycles are not of a fixed length, so before we observe the series, we cannot be sure where the peaks and lows of the cycles will be.

### In general, a stationary time series will have no predictable patterns in the long-term. Time plots will show the series to be roughly horizontal (although some cyclic behavior is possible), with constant variance. Figure 1 shows graphs of 9 time series datasets.

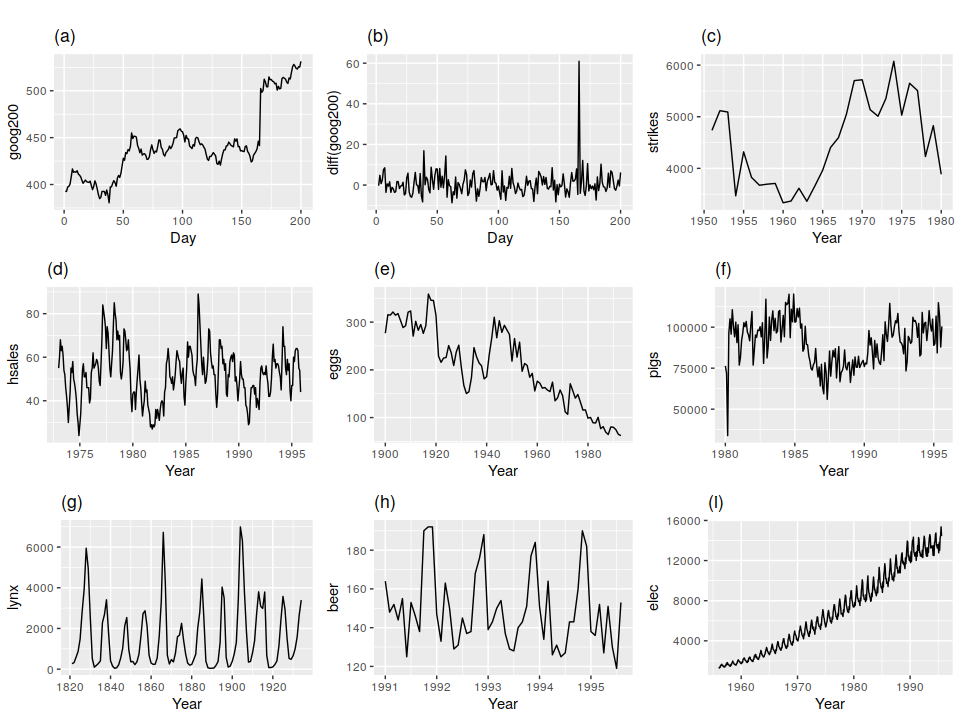


Figure 1: Graphs of 9 Time Series Datasets

We observe, that time series

* *d, h, and i* are not stationary since there is an obvious seasonality
* *a, c, e, f, and i* are not stationary since there is an obvious trend
* *b and g* are stationary

# **Differencing**

### One way to make a non-stationary time series stationary is to compute the differences between consecutive observations. This process is called *differencing*,

### Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

### As well as looking at the time plot of the data, the *ACF* plot is also useful for identifying non-stationary time series. For a stationary time series, the *ACF* will drop to zero relatively quickly, while the *ACF* of non-stationary data decreases slowly.

# **Random walk model**

### The differenced series is the change between consecutive observations in the original series, and can be written as

(1)

### The differenced series will have only t−1 values, since it is not possible to calculate a difference for the first observation.

### When the differenced series is white noise, the model for the original series can be written as

, where denotes white noise, rearranging this leads to the “random walk” model . (2)

**White Noise**

Time series that show no autocorrelation are white noise. For white noise series, we expect each autocorrelation to be close to zero. Obviously, they will not be exactly equal to zero as there is some random variation. For a white noise series, we expect 95% of the spikes in the *ACF* to stay within where T is the length of time series (i.e. number of time periods or observations). It is common to plot these bounds on a graph of the ACF (the blue dashed lines above). If one or more large spikes are outside these bounds, or if substantially more than 5% of spikes are outside these bounds, then the series is probably not white noise.

*In Summary, a time series may be white noise. A time series is white noise if the variables are independent and identically distributed with a mean of zero. This means that all variables have the same variance (sigma^2) and each value has a zero correlation with all other values in the series.*

Random walk models are widely used for non-stationary data, particularly financial and economic data. Random walks typically have:

* long periods of apparent trends up or down
* sudden and unpredictable changes in direction

The forecasts from a random walk model are equal to the last observation, as future movements are unpredictable, and are equally likely to be up or down. Thus, the random walk model reinforces Naïve forecasts, which was introduced as the benchmark for our forecasting process.

A closely related model allows the differences to have a non-zero mean. Then equation (2) becomes:

or. as you see here instead of , since this is the notation used by ARIMA documents.

The value of is the average of the changes between consecutive observations. If

is positive, then the average change is an increase in the value of . Therefore, will tend to drift upward and if negative then, will tend to drift downward.

# **Second Order Differencing**

Occasionally the differenced data will not appear to be stationary and it may be necessary to difference the data a second time to obtain a stationary series. Let’s assume

Then, the second order differencing is:

In this case, the will have T-2 values (see white noise section for T). Then, we would model the “change in the changes” of the original data. In practice, it is almost never necessary to go beyond second-order differences.

# **Seasonal Differencing**

A seasonal difference is the difference between an observation and the previous observation from the same season. Therefore:

Where *m* is the number of periods in a season. It is also called “lag m differences”, as we subtract the observations after a lag m periods. For example for monthly seasonality, there are 12 periods season and *m* = 12.

If seasonally differenced data appear to be white noise, then an appropriate model for the original data is

Forecasts from this model are equal to the last observation from the relevant period in previous season. That is, this model gives seasonal Naïve forecasts, introduced already.

Autoregressive models: Revisited

In a multi-linear regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression, as we said before, indicates that it is a regression of the variable against itself.

Thus, an autoregressive model of order *p* can be written as:

Where  is white noise. This is like a multiple regression but with *lagged values* of  as predictors. We refer to this as an AR(p)model, an autoregressive model of order *P*.

For a AR(1) model (i.e. ) when:

* is equivalent to white noise.
* is equivalent to random walk.
* is equivalent to random walk with drift and
* tends to oscillate around the mean

The autoregressive models are normally restricted to stationary data, in which case some constraints on the values of the parameters are required. These constraints are:

* For an AR(1) model:
* For an AR(2) model:

When restrictions are much more complicated. R takes care of these restrictions when estimating a model.

Moving Average models (AR context)

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model. Therefore, a moving average model of order *q* can be written as:

(1)

Where is white noise. We refer to equation (1) as an MA(q) model, a moving average model of order *q* of course. We do not observe the values of , so it is not really a regression in the usual sense. In addition, some texts drop the from the equation (1).

Notice that each value of can be thought of as a weighted moving average of the past few forecast errors. However, moving average models should not be confused with the *moving average smoothing* we discussed in Chapter 5! A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

Changing the parameters results in different time series patterns. As with autoregressive models, the variance of the error term will only change the scale of the series, not the patterns.

It is possible to write any stationary AR(p) model as model which is:

(2)

Here I explain how this happens. I apply the repeat substitution on AR(1).

then we can write

same as

Considering -1<b1<1, then the value of will get smaller as the k value gets larger. So eventually we obtain

which is a

If you are interested to know how I got this equation, please let me know ☺

The invertibility constraints for other models are like the stationarity constraints.

* For an MA(1) model:
* For an MA(2) model:

Again, when restrictions are much more complicated. R takes care of these restrictions when estimating a model.

Non-Seasonal ARIMA Models

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average (in this context, “integration” is the reverse of differencing). The full model can be written as:

(1)

Where is the differenced series (it may have been differenced more than once). The “predictors” on the right-hand side of equation (1) include both lagged values of and lagged errors. We call this an ARIMA(p,d,q), where:

|  |  |
| --- | --- |
| p= | **order of the autoregressive part;** |
| d= | degree of first differencing involved; |
| q= | order of the moving average part. |

The same stationarity and invertibility conditions that are used for autoregressive and moving average models also apply to an ARIMA model.

Many of the models we have already discussed are special cases of the ARIMA model, as shown in table below

|  |  |
| --- | --- |
| White noise | ARIMA(0,0,0) |
| Random walk | ARIMA(0,1,0) with no constant |
| Random walk with drift | ARIMA(0,1,0) with a constant |
| Autoregression | ARIMA(p,0,0) |
| Moving average | ARIMA(0,0,q) |

# **Understanding ARIMA models**

The R “*auto.arima()”* function is for building ARIMA models, but automated procedure can be a little risky, and it is worth understanding something of the behavior of the models even when you rely on an automatic procedure to choose the model for you.

The constant *C* has an important effect on the long-term forecasts obtained from these models. Let’s look again at the non-seasonal ARIMA model (equation (1) in the previous section). The constant has an important effect on the long-term forecasts obtained from these models.

* If  , the long-term forecasts will go to zero.
* If , the long-term forecasts will go to a non-zero constant.
* If , the long-term forecasts will follow a straight line.
* If , the long-term forecasts will go to the mean of the data.
* If ,, the long-term forecasts will follow a straight line.
* If ,, the long-term forecasts will follow a quadratic trend.

d is the degree of differencing (ARIMA(p, d, q))

The value of *d* also has an effect on the prediction intervals — the higher the value of *d*, the more rapidly the prediction intervals increase in size. For *d = 0*, the long-term forecast standard deviation will go to the standard deviation of the historical data, so the prediction intervals will all be essentially the same.

The value of *p* (the lag value) is important if the data show cycles. To obtain cyclic forecasts, it is necessary to have As well with some additional conditions on the parameters. For an AR(2) model, cyclic behavior occurs if .

# ***ACF* and *PACF* R Plots**

It is usually not possible to tell, simply from a time plot of time series dataset, what values of *p* and *q* are appropriate for the data. However, it is sometimes possible to use the *ACF* plot, and the closely related *PACF* plot, to determine appropriate values for *p* and *q.*

Let me refresh your memory about the *ACF* and *PACF* plots.

1. The *ACF* plot shows the autocorrelations which measure the relationship between and for different values of *k.*
2. Now if and are correlated, then and must also be correlated
3. So can we say, and might be correlated, simply because they are both correlated to , rather than because of any new information contained in that could be used in forecasting !!!!

So, I hope with this little reasoning based one *ACF* plot, I explained the problem. To overcome this problem, we use partial autocorrelations plot (*PACF*). *PACF* measures the relationship between and after removing the effects of lags 1, 2, …, k-1. So the first *partial autocorrelation* is identical to the *first autocorrelation*, because there is nothing between them to remove. Each partial autocorrelation can be estimated as the last coefficient in an autoregressive model. There are more depth into this topic but I rather stop here. Just we should know that *PACF* gives a better understanding of correlation among dataset values and consequently. Selecting values of *p* and *q*.

# **Maximum likelihood estimation**

Once the model order has been identified (i.e., the values of p, d, and q), we need to estimate the parameters . When R estimates the ARIMA model, it uses maximum likelihood estimation (MLE). This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed.

In practice, R will report the value of the *log likelihood* of the data; that is, the logarithm of the probability of the observed data coming from the estimated model. For given values of p, d, and q, R will try to maximize the *log likelihood* when finding parameter estimates.

ARIMA Modeling in R

R has two major functions for ARIMA. The “arima()” and “auto.arima()”.

# **How does auto.arima() work?**

The “auto.arima()” function in R uses a variation of the *Hyndman-Khandakar* algorithm. The arguments to “auto.arima()” provide for many variations on the algorithm. What is described here is the default behavior. The following is the automated ARIMA setting for ARIMA modeling:

|  |
| --- |
| ***Hyndman-Khandakar* algorithm for automatic ARIMA modelling** |
| 1. The number of differences 0 ≤ d ≤ 2 is determined using repeated KPSS tests. Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests are used for testing a null hypothesis that an observable time series is stationary around a deterministic trend (i.e. trend-stationary) against the alternative of a unit root. (wikipedia.org) |
| 1. The values of *p* and *q* are then chosen by minimizing the AICc after differencing the data *d* times. Rather than considering every possible combination of *p*and *q*, the algorithm uses a stepwise search to traverse the model space. |
| 1. Four initial models are fitted:    * ARIMA(0,d,0),    * ARIMA(2,d,2),    * ARIMA(1,d,0),    * ARIMA(0,d,1).   A constant is included unless d=2. If d ≤ 1, an additional model is also fitted:   * + ARIMA(0,d,0) without a constant. |
| 1. The best model (with the smallest AICc value) fitted in step (a) is set to be the “current model”. |
| 1. Variations on the current model are considered:    * vary pp and/or q from the current model by ±1;    * include/exclude   from the current model.   The best model considered so far (either the current model or one of these variations) becomes the new current model. |
| 1. Repeat Step 2(c) until no lower AICc can be found. |

Akaike’s Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. However, for ARIMA model the AICc or the corrected AIC is used. Good models are obtained by minimizing the AICc .

# **Choosing your own ARIMA model: the non-automated**

If you want to choose the model yourself, use the “Arima()” function in R. There is another function “arima()” in R which also fits an ARIMA model. However, it does not allow for the constant *C* unless d = 0, and it does not return everything required for other functions in the **forecast** package to work. Finally, it does not allow the estimated model to be applied to new data (which is useful for checking forecast accuracy). Consequently, it is recommended that Arima() be used instead.

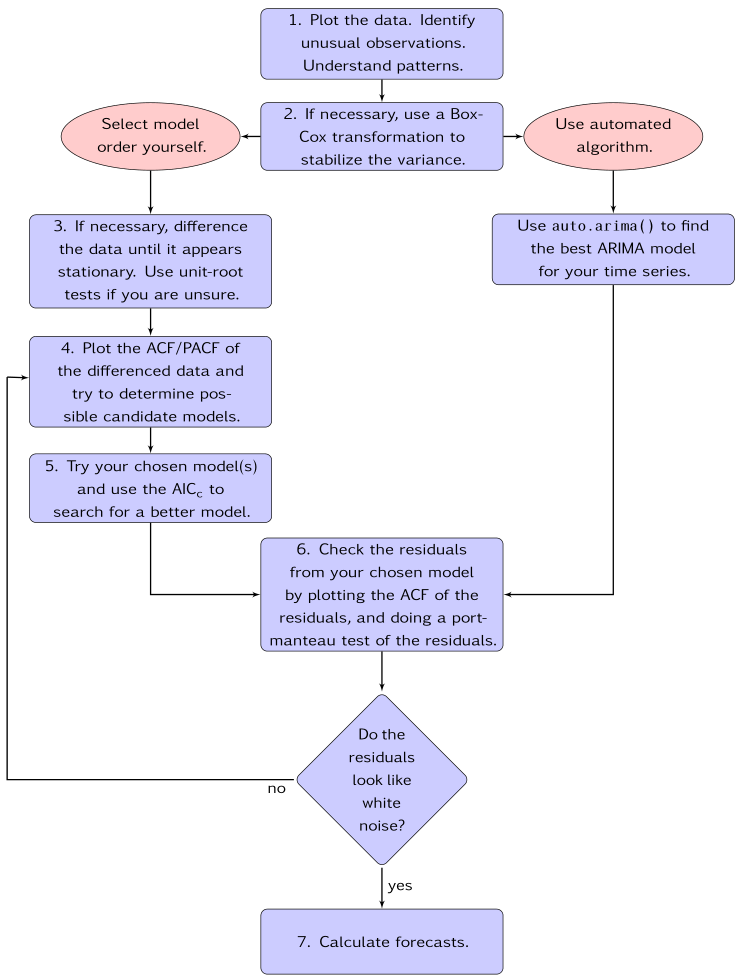
Note: Arima() and arima() are different R functions?

# **Modeling Procedure**

When fitting an ARIMA model to a set of (non-seasonal) time series data, the following procedure provides a useful general approach.

1. Plot the data and identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary, take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an ARIMA(p,d,0) or ARIMA(0,d,q) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

The *Hyndman-Khandakar* algorithm only takes care of steps 3–5. So even if you use it, you will still need to take care of the other steps yourself.

The process is summarized in the following flowchart:

Seasonal ARIMA Modeling in R

So far, we have restricted our attention to non-seasonal data and non-seasonal ARIMA models. However, ARIMA models are also capable of modelling a wide range of seasonal data.

A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

|  |  |  |
| --- | --- | --- |
| ARIMA | (p,d,q) | (P,D,Q)m |
|  | ↑ | ↑ |
| Non-seasonal part | Seasonal part of |
| of the model | of the model |

where *m* = number of observations per year. We use uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model.

The seasonal part of the model consists of terms that are similar to the non-seasonal components of the model, but involve backshifts of the seasonal period. For example, an ARIMA(1,1,1)(1,1,1)4 model (without a constant) is for quarterly data (m=4). The additional seasonal terms are simply multiplied by the non-seasonal terms. Here I intentionally omitted the seasonal terms because it is beyond the context of this course.

ARIMA vs ETS

It is a commonly held myth that ARIMA models are more general than exponential smoothing. While linear exponential smoothing models are all special cases of ARIMA models, the non-linear exponential smoothing models have no equivalent ARIMA counterparts. On the other hand, there are also many ARIMA models that have no exponential smoothing counterparts. In particular, all ETS models are non-stationary, while some ARIMA models are stationary.

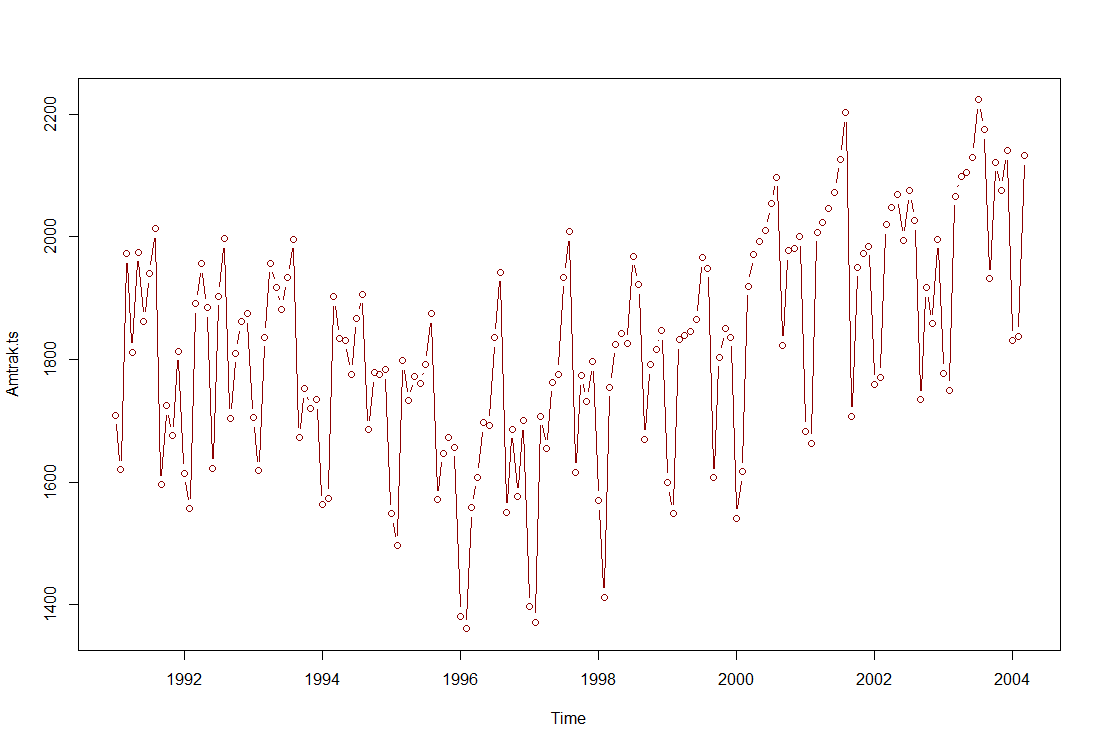
The *ETS* models with seasonality or non-damped trend or both have two unit roots (i.e., they need two levels of differencing to make them stationary). All other ETS models have one unit root (they need one level of differencing to make them stationary).

ARIMA R Native Codes

The following is an example for building an ARIMA model using seasonal ARIMA R code.

Assume Amtrak.ts is a time series dataset. We know the seasonal period is 12. First we create a ACF for up to lag 48 (4 period) differences AR chart.

library(astsa) #for generating colored plot

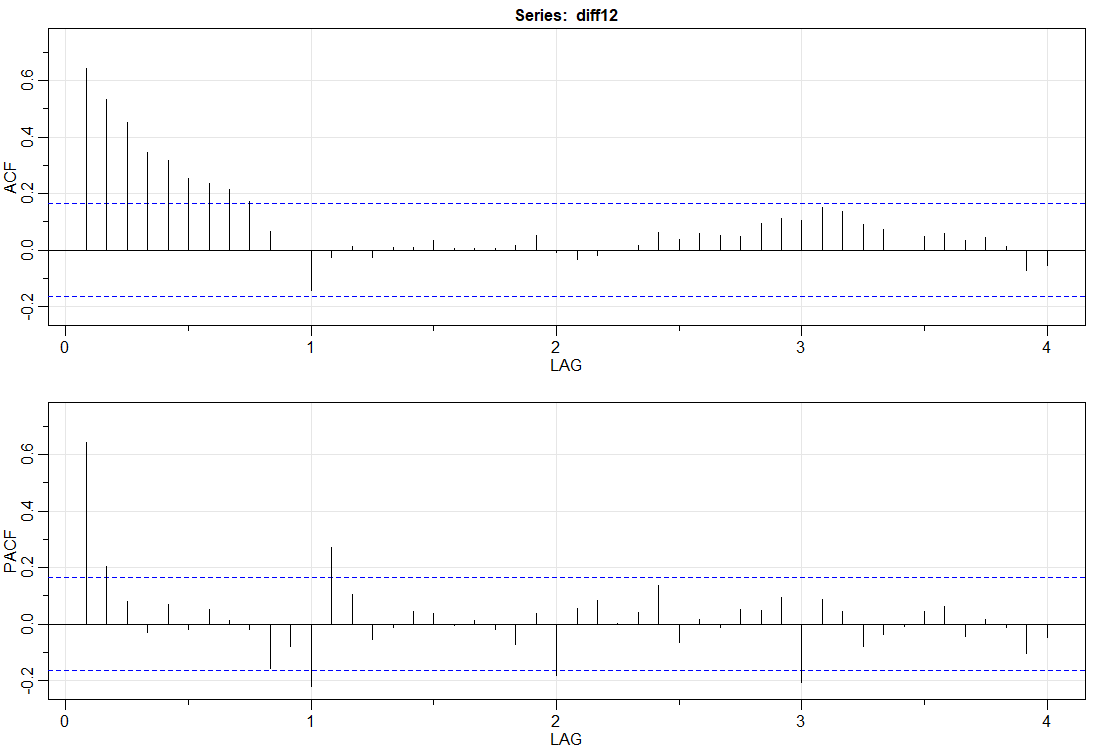
Amtrak.ts <- ts(Amtrak.df$Ridership, start = c(1991, 1), end = c(2004, 3), freq = 12)

plot(Amtrak.ts, type="b") #to see clearly the seasonality and trend

assuming the plot showed clear monthly effects and no obvious trend, so we examined the ACF and PACF of the 12th differences (seasonal differencing). Commands are:

diff12 = diff(Amtrak.ts, lag = 12) # get the difference up to 12: the size of the seasonality

acf2(diff12, 48) # plot Acf and Pacf together



The acf2 command asks for information about 48 lags. On the basis of the ACF and PACF of the 12th differences, we identified an ARIMA(1,0,0)(0,1,1)12  model as a possibility (match the numbers with the explanation given in previous sections ). The command for fitting this model is

Amtrak.ts.sariam<-sarima(Amtrak.ts, 1,0,0,0,1,1,12)

The parameters of the command just given are the data series, the non-seasonal specification of AR, differencing, and MA (colored red), and then the seasonal specification of seasonal AR, seasonal differencing, seasonal MA, and period or span for the seasonality (colored green).

The parameters of the command just given are the data series, the non-seasonal specification of AR, differencing, and MA, and then the seasonal specification of seasonal AR, seasonal differencing, seasonal MA, and period or span for the seasonality.

Output from the sarima command is:

Final Estimates of Parameters

kable(Amtrak.ts.sarima$ttable)

|Type | Estimate| SE| t.value| p.value|

|:--------|--------:|------:|-------:|-------:|

|ar1 | 0.8292| 0.0513| 16.1525| 0.0000|

|sma1 | -0.7430| 0.0909| -8.1698| 0.0000|

|constant | 1.5529| 0.8311| 1.8685| 0.0637|

And to see model strength

Amtrak.ts.sarima$AIC

$AIC

[1] 10.46097

Amtrak.ts.sarima$AICc

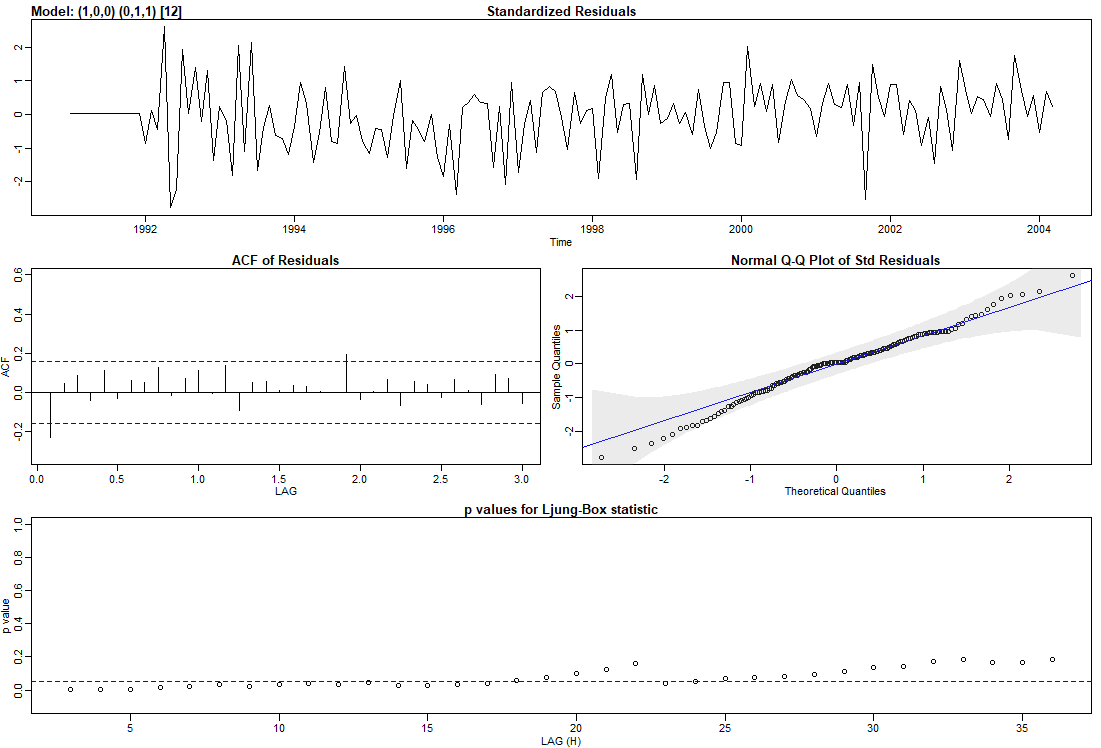
$AICc

[1] 10.46196

Amtrak.ts.sarima$BIC

$BIC

[1] 10.53668

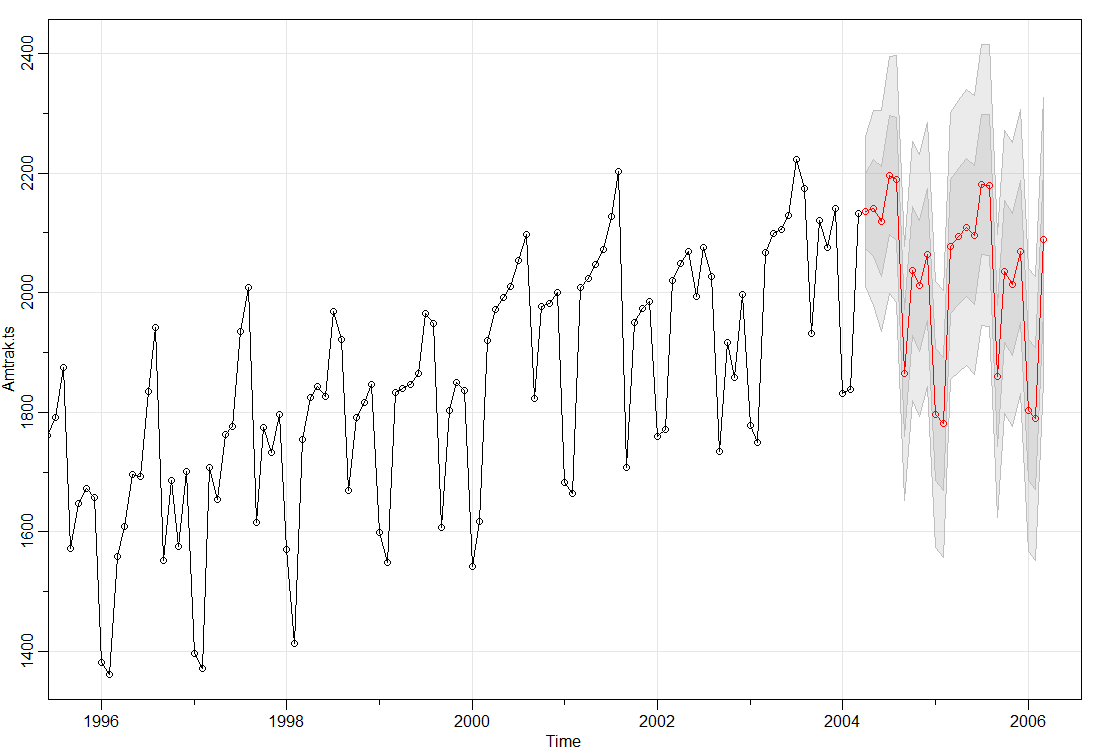


The output included these residual plots. The only difficulty we see is in the normal probability plot. The extreme standardized sample residuals (on both ends) are larger than they would be for normally distributed data.

* Standard residual
* ACF of residuals
* Normal Q-Q plot of std residuals
* P-values for Ljung-Box1 statistics

if we were to use R to generate forecasts for the next 24 months in R, one command that could be used is:

df.ts.sarima.fc<-sarima.for(Amtrak.ts, 24, 1,0,0,0,1,1,12)



we can get a partial forecast command as follow

$pred

1.

The Ljung–Box test (named for Greta M. Ljung and George E. P. Box) is a type of statistical test of whether any of a group of autocorrelations of a time series are different from zero. Instead of testing randomness at each distinct lag, it tests the "overall" randomness based on a number of lags and is therefore a portmanteau (hanger) test002E