Chapter 5. ARIMA Modeling with R

When building forecasting models, we do not have to pretend that the model we fit is true. We have to be aware that the model we are building is approximating a complex reality. In this chapter, we will discuss four approximations: **moving average** (MA) models, **autoregressive** (AR) models, **autoregressive moving average** (ARMA) models, and **autoregressive integrated moving average** (ARIMA) models. These four models vary in their specifics and have different mechanisms in capturing different types of autocorrelation behavior.

1. Introduction

A time series is a sequence of measurements of the same variable collected over time. Thus, a time series is a list of observations where the ordering matters since there is dependency in ordering as well as since changing the order may indicate the change in substantial meaning implied in the data.

One of objectives of a time series analysis is to specify a model that describe the pattern of the time series you have. Therefore, in general such a model would be used to describe the important feature of the time series pattern, to figure out how the past affects the future, and to predict future values of the series.

Basically, autoregressive integrated moving average (ARIMA) models are basic types of time series models that relate past values and past prediction errors to the current value of a series. Moving average (MA), autoregressive (AR) models, and autoregressive moving average (ARMA) models are all special cases of ARIMA models. We will discuss about each model in more details in later sections.

When you look at a time series, you need to first consider if there is (are):

- a trend. As you have seen in chapter 4, a trend means that the sequence of measurements, on average, tends to increase or decrease over time.
- a seasonality, a regularly repeating pattern of highs and lows associated with calendar time, such as quarters, months, days of the week, and so on.
- outliers, which are several data points far away from most of other data.
- a constant variance over time for stationarity.
- any abrupt changes to either the level of the series or the variance.

2. Autoregressive (AR) models

In previous chapter, you have learned an important concept, stationarity. In order for an ACF to make sense, the series should be stationary, meaning that the autocorrelation for any particular lag is the same regardless of where we are in time. Here, we again review the stationarity we discussed in chapter 4.

A series X_t is called stationary if it satisfies the following properties:

- The mean of the series, $E[X_t]$, is the same for all t.
- The variance of X_t is the same for all t.
- The correlation between X_t and X_{t-k} is the same for all t.

The last property of a stationary time series indicates that a theoretical autocorrelation of particular lag is the same across the whole series. Therefore, a stationary time series has theoretically the same structure forwards as it does backwards. However, most time series we encounter in the real world would not be stationary. They may have a continual upward trend or downward trend, which is a violation of the requirement that the mean is the same for all t. Seasonal patterns also violate that requirement as well. For this reason, the first thing to do when you encounter a time series is to see if the series is stationary and to make the series stationary since the following models, including AR, MA, ARMA, and ARIMA, will be defined under the presumption that the series is stationary.

1) AR(1) process

The first order autoregressive process is referred to as AR(1). The AR(1) process is written by

$$X_t = \mu + \emptyset_1 X_{t-1} + \varepsilon_t$$

$$\varepsilon_t \sim wn(0, \sigma^2),$$

where the errors ε_t are independent of X. The AR(1) process represents that the value of X at time t is a linear function of the value of X at time t-1.

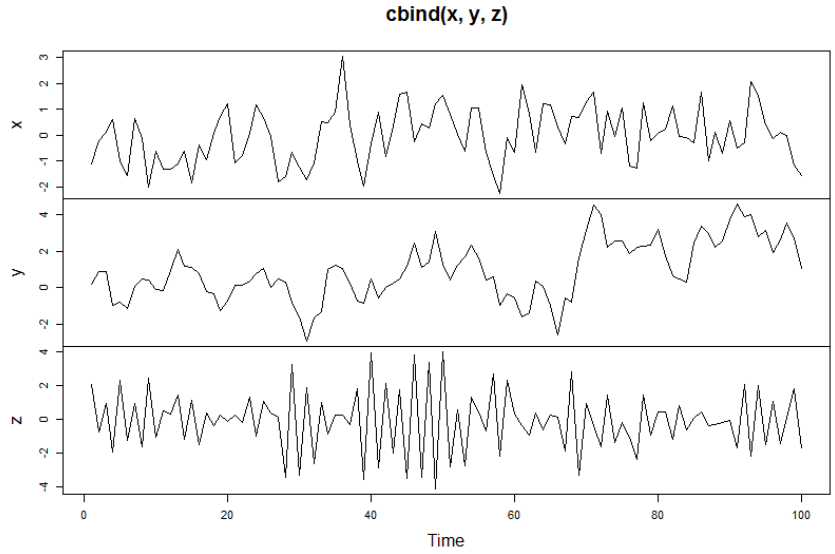
The ACF for AR(1) process shows a distinct pattern. For $\emptyset_1 > 0$, the ACF exponentially decreases to 0 as the lag k increases. For $\emptyset_1 < 0$, the ACF also exponentially decreases to 0 as the lag k increases, but the signs for autocorrelations alternate between positive and negative. Keep in mind that such patterns of the ACF are the case for theoretical ACF

The AR(1) process will be shown by the following simulations. The **arima.sim(…)** function can be used to simulate ARIMA model. For simplicity, we set the mean $\mu = 0$ and the length of output series n = 100 in our simulations.

****** The purpose of the simulations here is to show you how data following AR, MA, ARMA, or ARIMA models typically look like. Generally, when data are given, you would not know what model the data would follow before you fit the models to the data. We will discuss how to fit the data to the models in later section ******

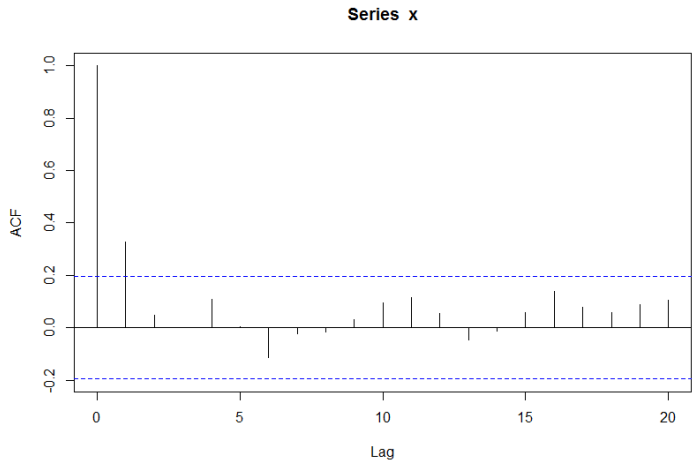
```
# Simualte AR(1) process with 0.1 slope
x <- arima.sim(model = list(ar = c(0.1)), mean = 0, n = 100)

# Simulate AR(1) process with 0.9 slope
y <- arima.sim(model = list(ar = c(0.9)), mean = 0, n = 100)
```

```
# Simualte AR(1) process with -0.8 slope
z <- arima.sim(model = list(ar = c(-0.8)), mean = 0, n = 100)

# Plot the simulated data
plot.ts(cbind(x, y, z))
```

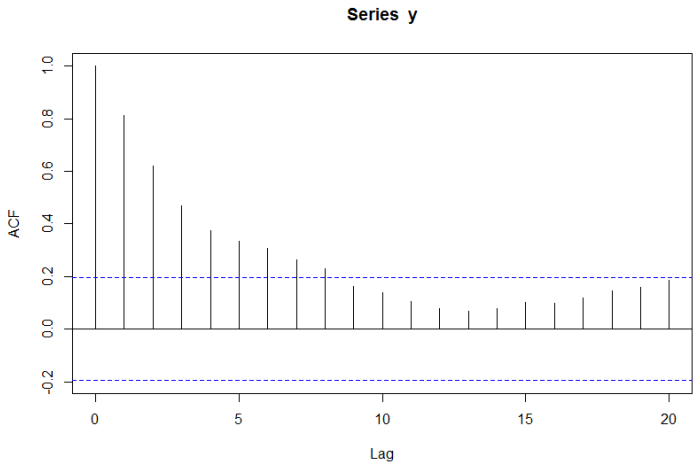


Note that you may have a bit different series from that I have because this is just a simulation. You must have different series whenever you run a simulation. As you can see, the AR(1) process with large value of slope $\emptyset_1$ shows strong persistence in level, which means each observation is close to the neighbors, because the large value of slope $\emptyset_1$ lead to greater autocorrelation. Also, the AR(1) process with a negative value of slope $\emptyset_1$ exhibits oscillatory pattern. Now, let us see the sample ACFs for the AR(1) processes we simulated above.

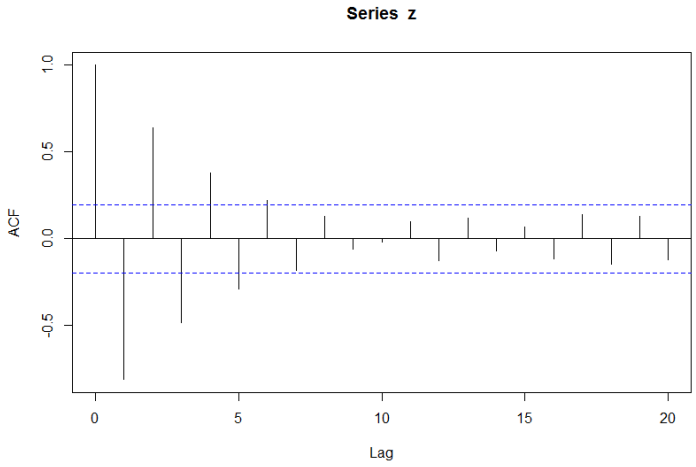
```
# Plot the ACF for x, y, and z
acf(x)
```



```
acf(y)
```



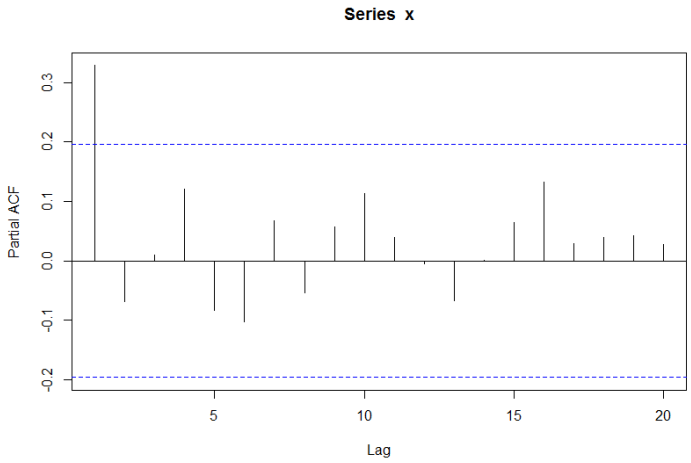

```
acf(z)
```



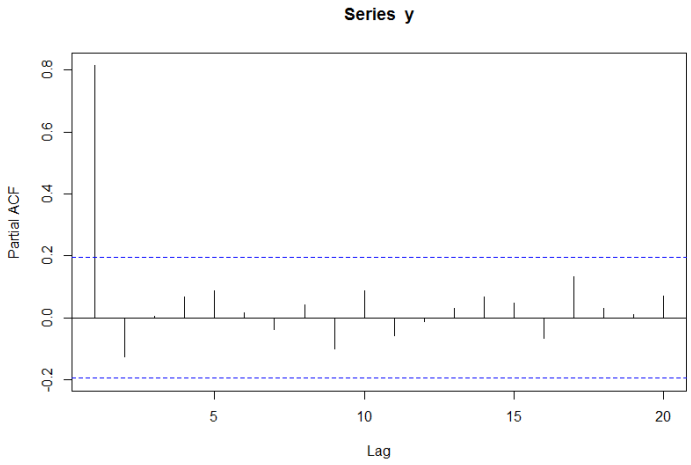
If the value of slope $\emptyset_1$ is positive and relatively small, the sample ACFs damp abruptly after lag 1. If the value of slope $\emptyset_1$ is positive and relatively large, the sample ACFs gradually damp. However, if the value of slope $\emptyset_1$ is negative, the decay involves back-and-forth oscillations. As we have seen at the plots of series, the sample ACFs also show that the persistence is much stronger when $\emptyset_1 = 0.9$.

Finally, the partial autocorrelation function for the AR(1) process cuts off abruptly. Why? The partial autocorrelations are just the last coefficients in a sequence of successively longer population autoregressions. If the true process is in fact an AR(1), the first autocorrelation is just the autoregressive coefficient, and coefficient on all longer lags are 0. Now, we show the partial autocorrelation functions (PACFs) for three AR(1) processes we had above from simulations.

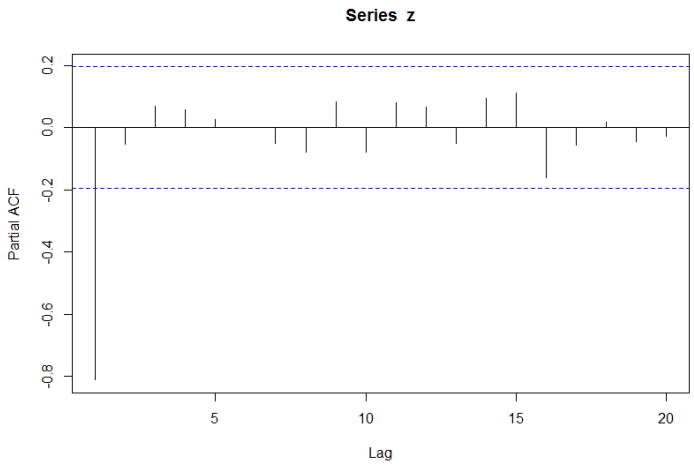
```
# Plot the PACF for x, y, and z
pacf(x)
```




```
pacf(y)
```




```
pacf(z)
```

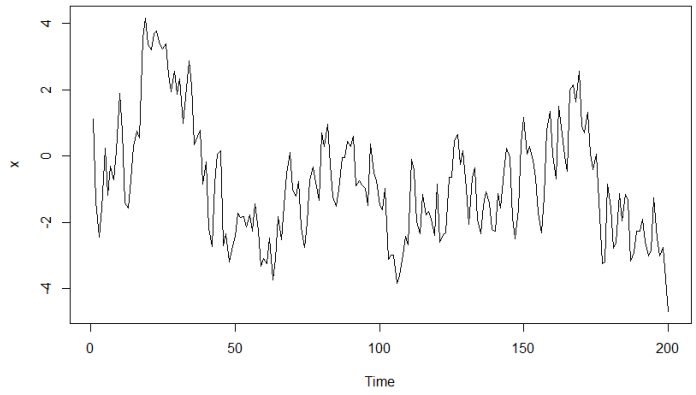


At lag 1, the partial autocorrelations are close to the parameters of the process (0.4, 0.9, and -0.8, respectively), and at longer lags, the partial autocorrelations approach 0.

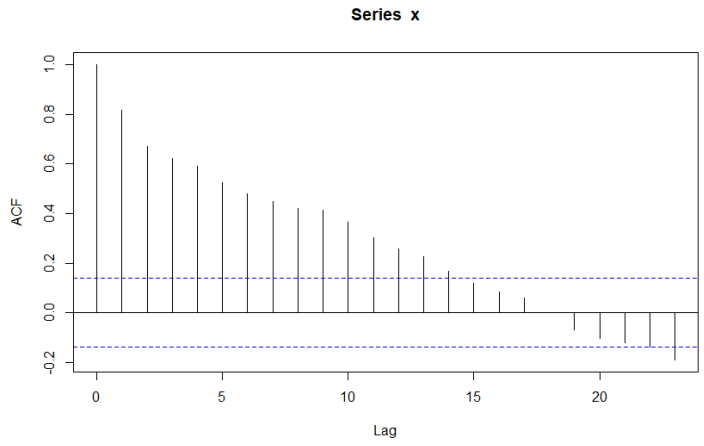
2) Comparison with random walk

We have discussed the random walk in chapter 4. The random walk process is a special case of the AR(1) process, where the value of slope is equal to 1 and the intercept is equal to 0. The random walk process is not stationary since its variance increases over time. The AR(1) process is stationary if and only if $-1 < \emptyset_1 < 1$. As you have seen above, the AR(1) process exhibits stronger persistence when the value slope $\emptyset_1$ is closer to 1. Another property of AR(1) process when $\emptyset_1$ is closer to 1 is that its sample ACF decays to zero at a slow rate, which means the values far in the past still impact on the future values of the process. Let me show you these properties by simulating some AR(1) processes as well as random walk process.

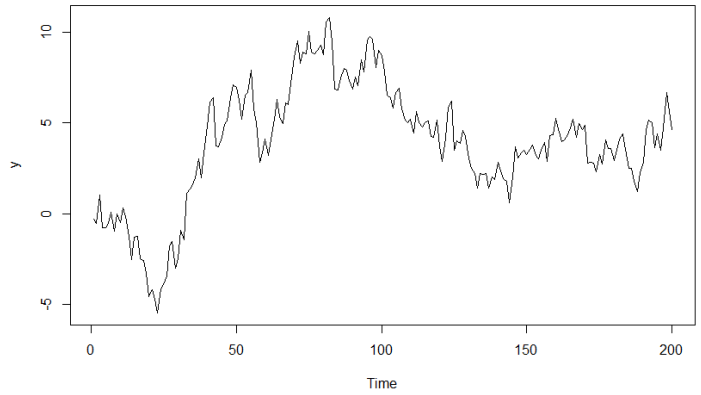
```
# Simulate AR(1) process with 0.9 slope
x <- arima.sim(model = list(ar = c(0.9)), mean = 0, n = 200 )
ts.plot(x)
```



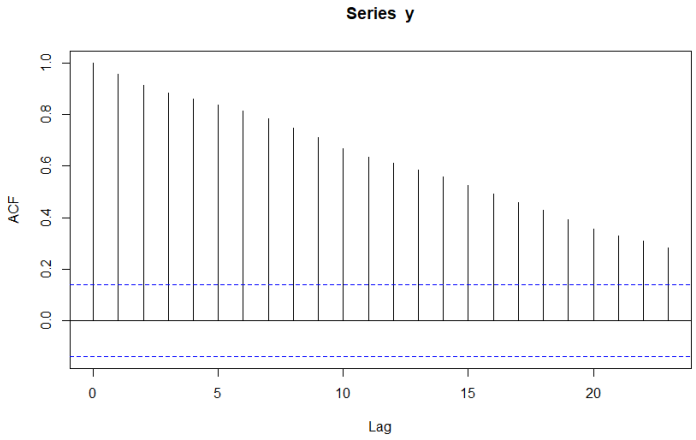

```
acf(x)
```



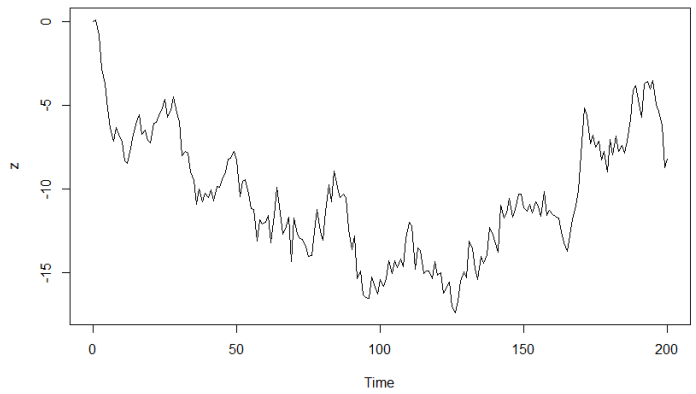
```
# Simulate AR(1) process with 0.98 slope
y <- arima.sim(model = list(ar = c(0.98)), mean = 0, n = 200 )
ts.plot(y)
```



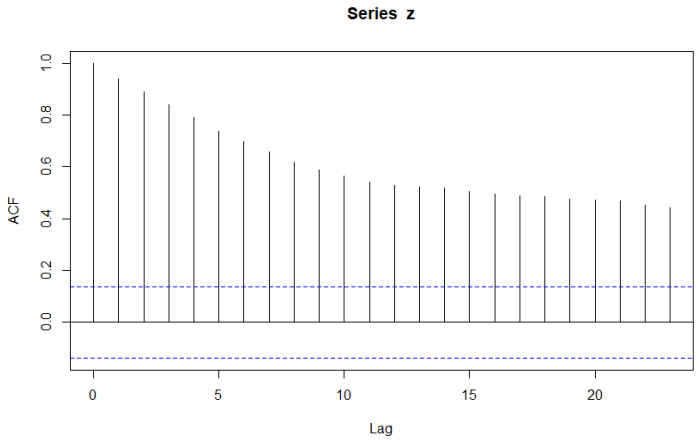

```
acf(y)
```




```
# Simulate random walk process
Z <- arima.sim(model = list(order = c(0, 1, 0)), n = 200)
ts.plot(z)
```



```
acf(z)
```



Notice that for the simulation of random walk process, **order = c(0, 1, 0)** is used. Each number in c(*p*, *d*, *q*) represents as follows.

- *p*: *p*-th order autoregressive (AR) component
- *d*: the degree of differencing in the integrated component
- *q*: *q*-th order moving average (MA) component

We will discuss both the integrated model and moving average model later. So, at this point, just note that c(0, 1, 0) refers to no AR component, one differencing of the series, and no MA component since the change, $X_t - X_{t-1}$, in the series is absolutely random in random walk process.

3) AR(*p*) process

The general *p*th order autoregressive process, or AR(*p*) is

$$X_t = \mu + \emptyset_1 X_{t-1} + \emptyset_2 X_{t-2} + \ \dots + \emptyset_p X_{t-p} + \varepsilon_t$$

$$\varepsilon_t \sim wn(0, \sigma^2)\ .$$

where $\emptyset_1$, $\emptyset_2$, ... , $\emptyset_p$ are parameters for the model. AR(*p*) process is referring to the use of past values in the regression equation for the series X. The autoregressive parameter *p* specifies the number of lags used in the model. For instance, AR(2) is represented as

$$X_t = \mu + \emptyset_1 X_{t-1} + \emptyset_2 X_{t-2} + \varepsilon_t\ .$$

As autocorrelation function (ACF) of the AR(1) process, that for the general AR(*p*) process decays gradually with displacement. The AR(*p*) partial autocorrelation function has a sharp cutoff at lag *p*, for the same reason that AR(1) PACF has a sharp cutoff at lag 1.

3. Moving average (MA) models

Moving average models or moving average processes are based on the fact that the variation in time series is driven by shocks of various sorts. Thus, in move average models, the output variable depends linearly on the current and the past shocks.

1) MA(1) process

The first-order moving average process, or MA(1) process, is represented by

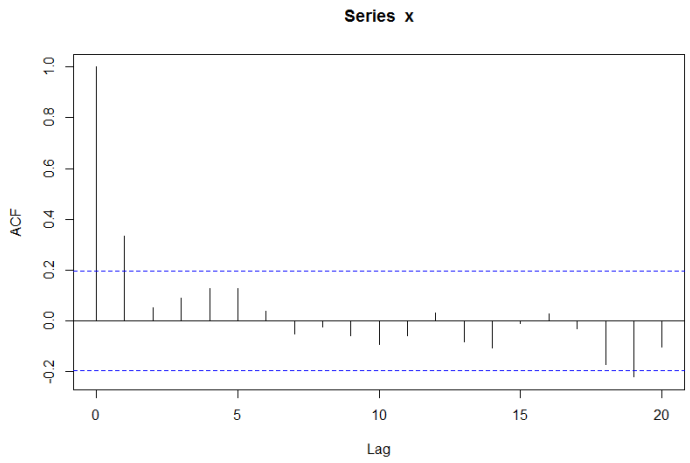
$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

$$\varepsilon_t \sim wn(0, \sigma^2).$$

MA(1) process is that the current observed value in series is expressed as a linear function of current and lagged unobservable shocks.

The sample ACF for MA(1) process shows a "spike" at lag 1 followed by non-significant values for lags past 1 in general. MA(1) process also can be simulated using **arima.sim(…)** function. You will see the plots of MA(1) processes with three different values of slope θ_1 from the following simulation. Again, we set the mean $\mu = 0$ in our simulations for simplicity.

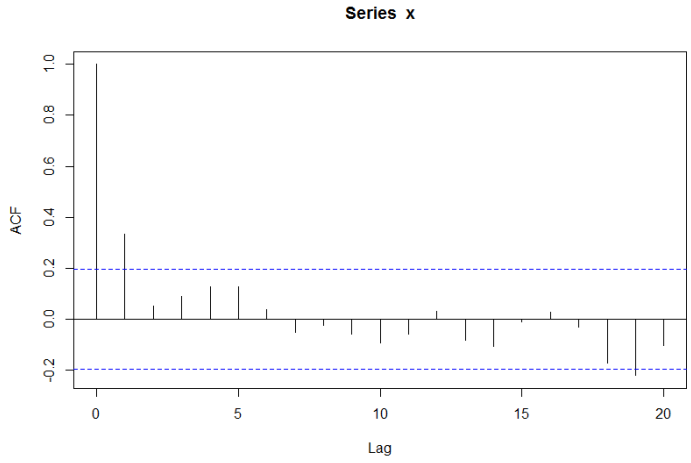
```
# Simulate MA(1) process with 0.4 slope
x <- arima.sim(model = list(ma = 0.4), n = 100)

# Simulate MA(1) process with 0.9 slope
y <- arima.sim(model = list(ma = 0.9), n = 100)

# Simulate MA(1) process with -0.6 slope
z <- arima.sim(model = list(ma = -0.6), n = 100)

# Plot simulated data
plot.ts(cbind(x, y, z))
```



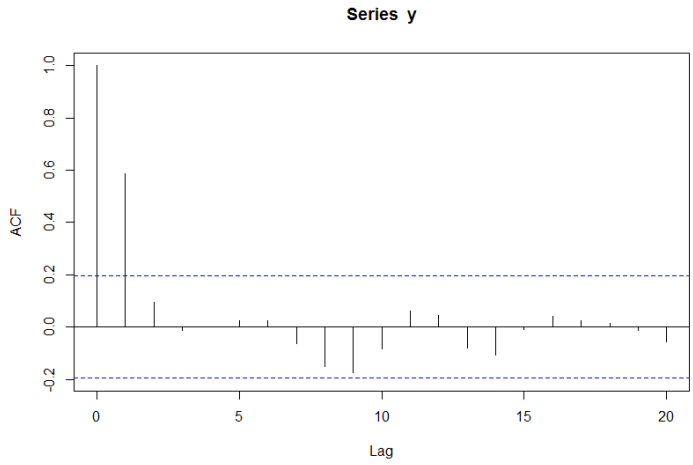
The only difference in the realizations in simulation above comes from different values of slope. Take a look at the first two simulations, whose slopes are 0.4 and 0.9. Past shocks feed positively into the current value of the series with a small weight of $\theta_1 = 0.4$ in one case and a large weight of $\theta_1 = 0.9$ in the other. You might think that $\theta_1 = 0.9$ would induce much more persistence than $\theta_1 = 0.4$, but it does not. The structure of the MA(1) process, in which only the first lag of the shock appears on the right, forces it to have a very short memory, and therefore weak dynamics, regardless of the value of slope.

Now we estimate the sample ACF for three simulated MA(1) processes.

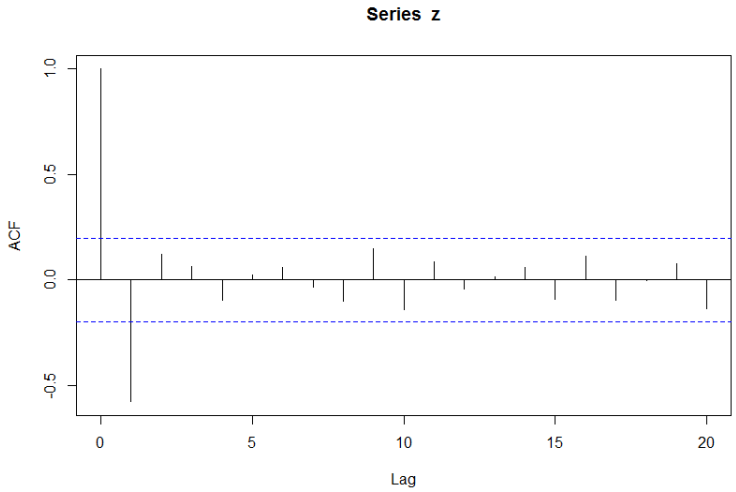
```
# Plot the ACF for x, y, and z
acf(x)
```




```
acf(y)
```



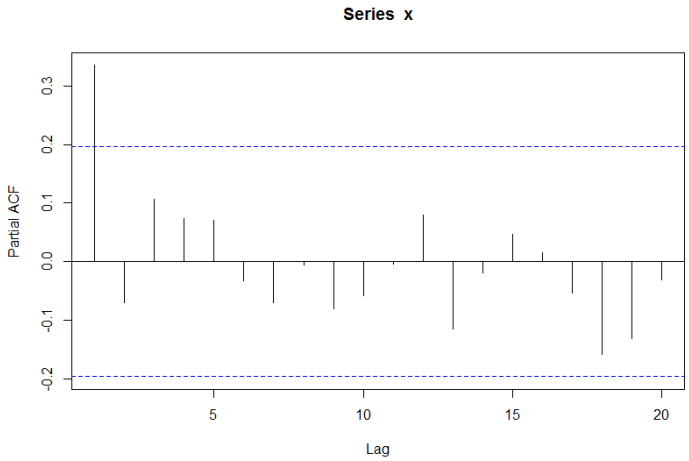
```
acf(z)
```



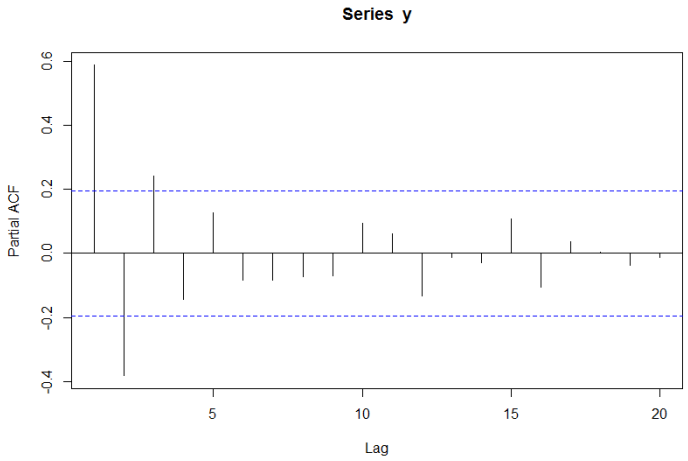
The key feature of the sample ACF for MA(1) process here is the sharp cutoff in the autocorrelation function for lags past 1. Thus, a different sample would have a slightly different sample ACF, but would likely have the same broad features.

Now, we consider the partial autocorrelation function (PACF) for MA(1) process. The PACF of the MA(1) will decay gradually to 0. As we have discussed so far, the PACFs are just the coefficients on the last included lag in a sequence of progressively higher-order autoregressive approximations. If $\theta_1 > 0$, the pattern of decay will be one of damped oscillation; otherwise, the decay will be one-sided.

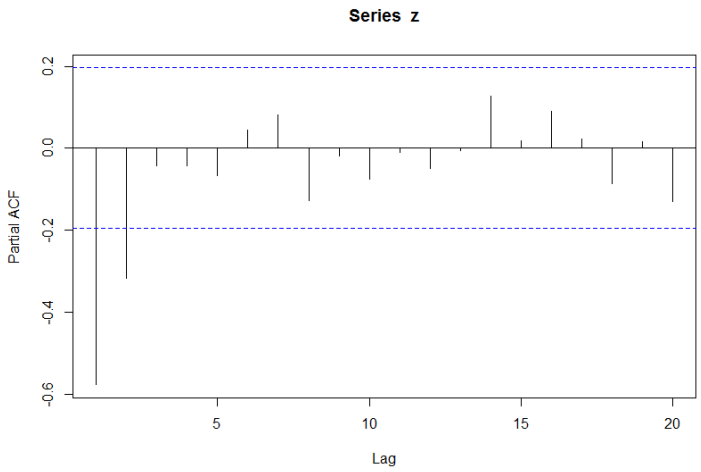
```
# Plot the PACF for x, y, and z
pacf(x)
```



```
pacf(y)
```




```
pacf(z)
```



The sample PACFs display a similar damped oscillation.

2) MA(q) process

We consider the general finite-order moving average process of order q, or MA(q), which is denoted by

$$X_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}$$

$$\varepsilon_t \sim wn(0, \sigma^2).$$

The MA(q) process is a natural generalization of the MA(1). By allowing for more lags of the shock, the MA(q) process can capture richer dynamic patterns. Of course, MA(1) process is a special case of the MA(q) when q = 1.

The MA(q) process has the potential for longer memory, which emerges clearly in its autocorrelation function. As all autocorrelations beyond lag 1 are 0 in the MA(1) process, in the MA(q) all autocorrelations beyond lag q are 0. This autocorrelation cutoff is a distinctive property of moving average process. In contrast, the partial autocorrelation function of the MA(q) process decays gradually in either an oscillating or a one-sided fashion, depending on the parameters of the process.

4. Autoregressive moving average (ARMA) model

Autoregressive (AR) and moving average (MA) models are often combined in order to obtain better and more parsimonious approximations to the series, which is referred to as the autoregressive moving average process, ARMA(p, q) for short.

The simplest ARMA process that is not a pure AR or pure MA is the ARMA(1, 1) given by

$$X_t = \mu + \emptyset_1 X_{t-1} + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

$$\varepsilon_t \sim wn(0, \sigma^2).$$

The ARMA(p, q) process is a natural generalization of the ARMA(1, 1) that allows for multiple AR and MA lags as follows:

$$X_t = \mu + \emptyset_1 X_{t-1} + \emptyset_2 X_{t-2} + \cdots + \emptyset_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

$$\varepsilon_t \sim wn(0, \sigma^2).$$

In contrast to pure AR or pure MA processes, neither the autocorrelation nor partial autocorrelation functions of ARMA processes cut off at any particular lag. Instead, each damps gradually with precise pattern depending on the process.

By allowing for both AR and MA components, ARMA models often provide accurate approximations to a time series that nevertheless have just a few parameters. For instance, in a certain situation, it might take an AR(6) to get the same approximation accuracy as could be obtained with an ARMA(2, 1), but AR(6) has six parameters to be estimated, whereas the ARMA(2, 1) has only three.

5. Fitting a model

Before we move on, let me review the ACF and PACF to help you identify the orders p and q of a model.

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

From now on, we will use **astsa** package to easily fit models to time series data. So, please install and import the package as follows.

```
# Install and import astsa package
install.packages("astsa")
library(astsa)
```

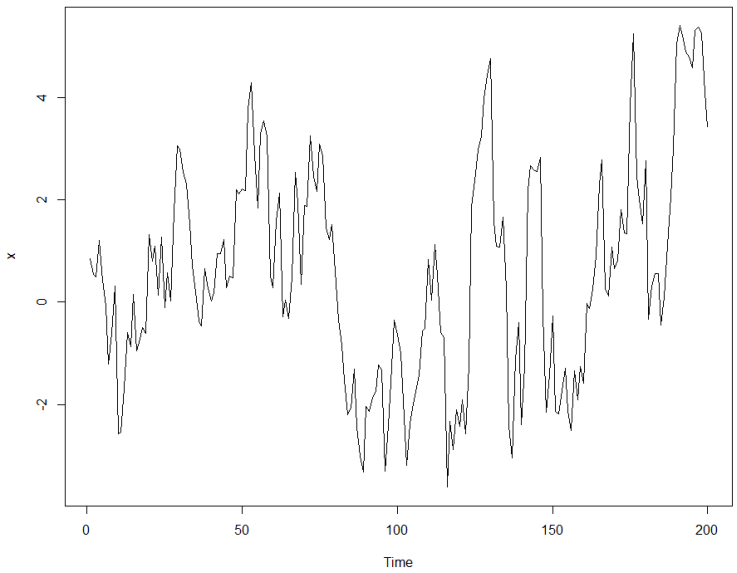
In this section, **you will generate data from each model in order to show how to fit models to data. In other words, you will use the data generated from simulations of each model instead of using a certain type of data. Then, you will fit the model to the data and see how it works. Please note that the reason why simulations are performed here is just to obtain the data following a particular type of model. Generally, you will encounter a certain type of data and then you will have to analyze what model the data would follow.**

1) Fitting an AR(1) model

First of all, you will generate 200 observations from simulation of the following AR(1) model.

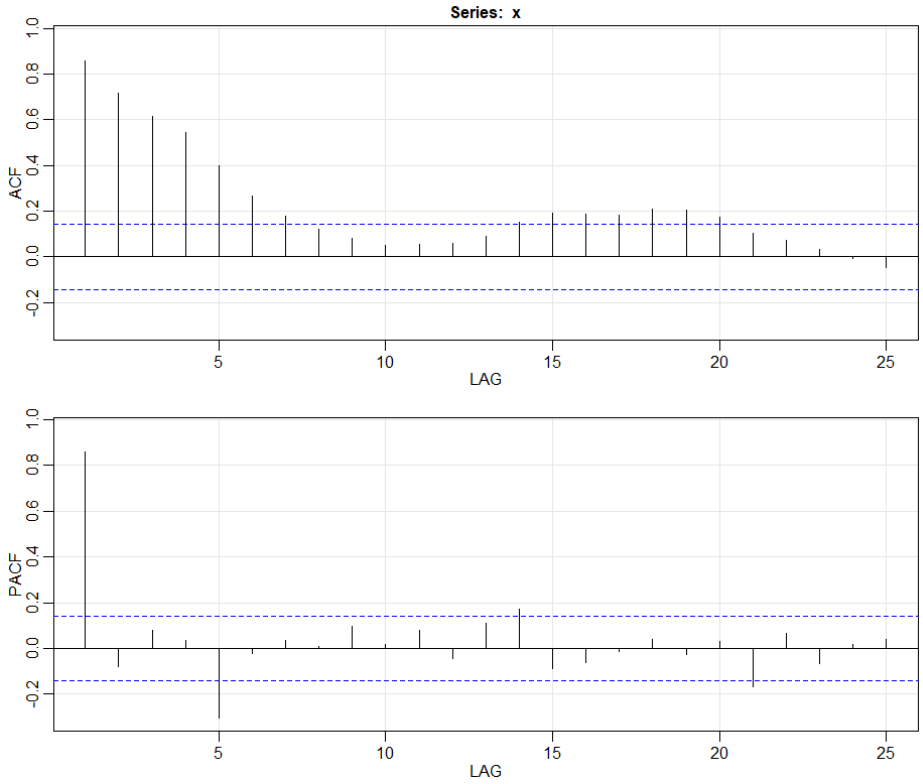
$$X_t = 0.87X_{t-1} + \varepsilon_t$$

```
# Generate the AR(1) process with 200 observations
x <- arima.sim(model = list(order = c(1, 0, 0), ar = 0.87), n = 200)

# Plot the generated series
plot(x)
```



Now, you have time series data whose underlying process is AR(1). Then, **suppose you encounter this time series (pretend the data not to be generated from the simulation) and you try to identify a model for the series (The true model will be AR(1) because this data is generated from the simulation of AR(1), but suppose you do not know what true model is.**). You will be able to guess what model fits to the data by examining ACF and PACF.

```
# Plot the sample ACF and PACF
acf2(x)
        ACF  PACF
 [1,]  0.86  0.86
 [2,]  0.72 -0.08
 [3,]  0.62  0.08
 [4,]  0.54  0.03
 [5,]  0.40 -0.30
 [6,]  0.27 -0.02
 [7,]  0.18  0.03
 [8,]  0.12  0.01
 [9,]  0.08  0.10
[10,]  0.05  0.02
[11,]  0.06  0.08
[12,]  0.06 -0.05
[13,]  0.09  0.11
[14,]  0.15  0.17
[15,]  0.19 -0.09
[16,]  0.19 -0.06
[17,]  0.18 -0.01
[18,]  0.21  0.04
[19,]  0.20 -0.03
[20,]  0.18  0.03
[21,]  0.10 -0.17
[22,]  0.07  0.07
[23,]  0.03 -0.06
[24,] -0.01  0.02
[25,] -0.05  0.04
```



The ACF tails off gradually, and the PACF cuts off after lag 1. So, you may guess this process would be AR(1). Then, let us see if this process is really AR(1) process as you guess.

```
# Fit an AR(1) to the series generated from simulation
x_fit <- sarima(x, p = 1, d = 0, q = 0)
x_fit

$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ar1   xmean
      0.8628  0.6535
s.e.  0.0353  0.5341

sigma^2 estimated as 1.139:  log likelihood = -297.51,  aic = 601.02

$degrees_of_freedom
[1] 198

$ttable
      Estimate     SE t.value p.value
ar1     0.8628 0.0353 24.4759  0.0000
xmean   0.6535 0.5341  1.2235  0.2226

$AIC
[1] 1.150388
```

```
$AICc
[1] 1.161

$BIC
[1] 0.1833714
```

The **sarima(…)** function produces so many results, including a residual diagnostic plot. At this time, ignoring other results, just pay attention to the t-table highlighted and compare the estimates to the true values. Remember that you created a times series data whose underlying process is AR(1) from the simulation. Therefore, if you fit AR(1) model (as you guessed) to the data, the estimates should be close to true coefficients ($\mu = 0$ and $\emptyset_1 = 0.87$). Take a look at an estimate for $\emptyset_1$ first. The estimate is 0.8628, which is very close to true value 0.87, and p.value is 0.0000. The p.value is interpreted in the following way:

- A small p.value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so you reject the null hypothesis.
- A large p.value (> 0.05) indicates weak against the null hypothesis, so you fail to reject the null hypothesis.
- The p.values very close to the cutoff (0.05) are considered to be marginal (could go either way).

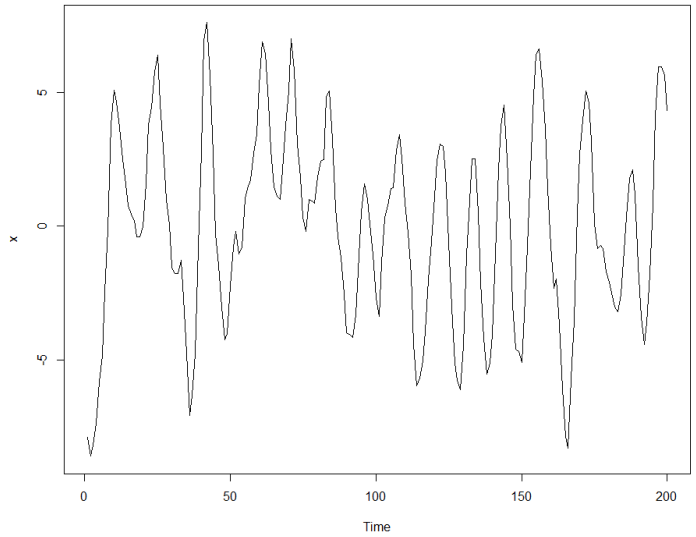
In this case, the null hypothesis for $\emptyset_1$is that $\emptyset_1 = 0$, the null hypothesis for μ is that $\mu = 0$. Thus, while you can reject the null hypothesis for $\emptyset_1$, which is $\emptyset_1 = 0$, you cannot reject the null hypothesis for μ, which is $\mu = 0$; you have a desired result.

2) Fitting an AR(2) model

Now, you will generate 200 observations from simulation of the following AR(2) model.

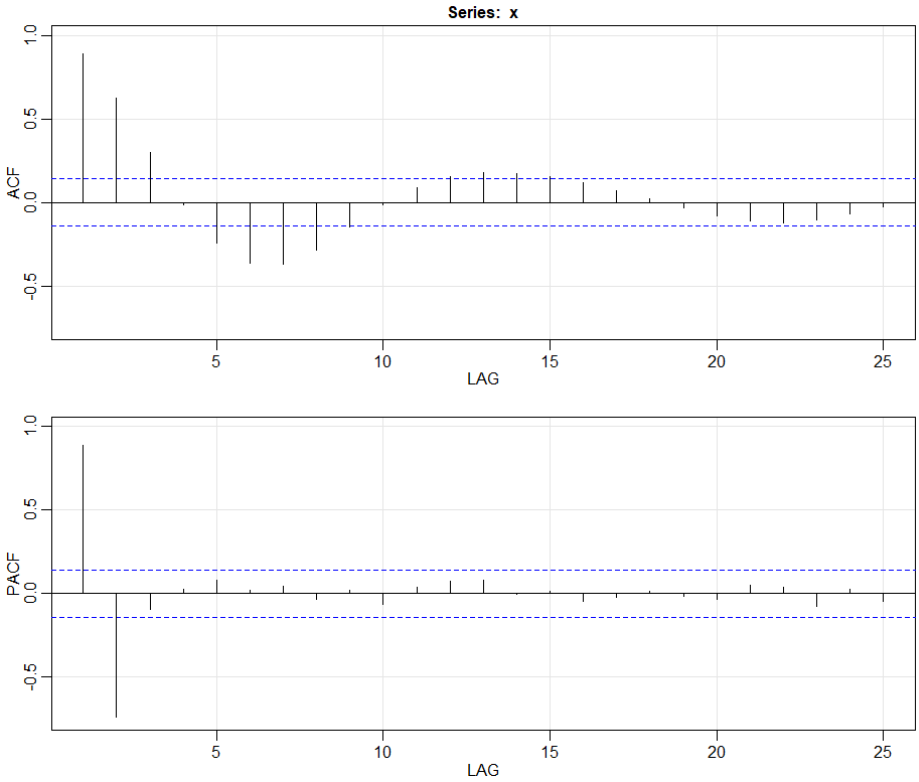
$$X_t = 1.65X_{t-1} - 0.8X_{t-2} + \varepsilon_t$$

```
# Generate the AR(2) process with 200 observations
x <- arima.sim(model = list(order = c(2, 0, 0), ar = c(1.65, -0.8)), n = 200)

# Plot the generated series
plot(x)
```



Thus, the time series x is AR(2) process. As you have seen in fitting AR(1) process, **suppose you encounter this time series whose underlying process is AR(2). Without knowing what this process is, you try to identify a model for it.** To guess what model would fit to the data, let us examine ACF and PACF.

```
# Plot the sample ACF and PACF
acf2(x)
         ACF   PACF
 [1,]  0.89  0.89
 [2,]  0.63 -0.74
 [3,]  0.30 -0.09
 [4,] -0.01  0.02
 [5,] -0.25  0.08
 [6,] -0.37  0.02
 [7,] -0.37  0.04
 [8,] -0.28 -0.04
 [9,] -0.15  0.02
[10,] -0.01 -0.07
[11,]  0.09  0.04
[12,]  0.15  0.07
[13,]  0.18  0.08
[14,]  0.17 -0.01
[15,]  0.15  0.01
[16,]  0.12 -0.05
[17,]  0.07 -0.03
[18,]  0.02  0.01
[19,] -0.03 -0.01
[20,] -0.08 -0.03
[21,] -0.11  0.05
[22,] -0.12  0.03
[23,] -0.11 -0.08
[24,] -0.07  0.03
[25,] -0.03 -0.05
```



From ACF and PACF, you may guess that the process would be AR(2) since the ACF tails off and the PACF cuts off after lag 2. Now, you try to fit AR(2) to the data.

```
# Fit an AR(2) to the series generated from simulation
x_fit <- sarima(x, p = 2, d = 0, q = 0)


$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ar1      ar2    xmean
      1.6388  -0.8266  -0.0080
s.e.  0.0398   0.0405   0.3465

sigma^2 estimated as 0.8457:  log likelihood = -268.99,  aic = 545.99

$degrees_of_freedom
[1] 197

$ttable
       Estimate     SE  t.value p.value
ar1      1.6388 0.0398  41.1300  0.0000
ar2     -0.8266 0.0405 -20.4269  0.0000
xmean   -0.0080 0.3465  -0.0230  0.9817

$AIC
[1] 0.8623843
```

```
$AICc
[1] 0.87341

$BIC
[1] -0.0881409
```

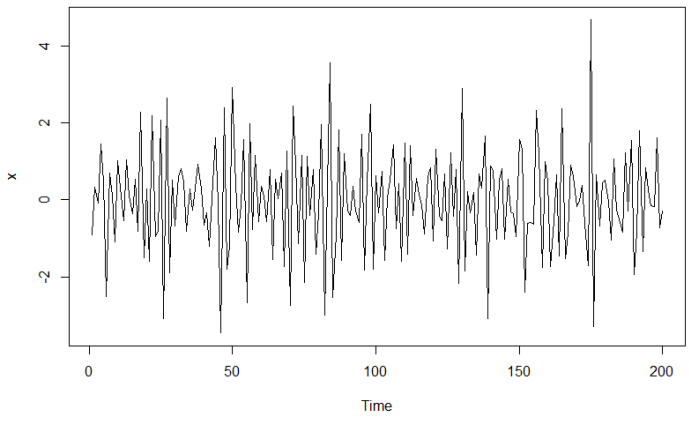
Again, ignore other results at this time. Look at t-table highlighted. You guessed this process would be AR(2) and fitted AR(2) to the series. Since the true process of this series was AR(2), the estimates for AR(2) you fitted should be close to true coefficients ($\mu = 0$, $\emptyset_1 = 1.65$, and $\emptyset_2 = -0.8$). The estimate for $\emptyset_1$ is 1.6388 (p.value is 0.0000, so that you can reject the null hypothesis, $\emptyset_1 = 0$), and the estimate for $\emptyset_2$ is -0.8266 (p.value is 0.0000. so that you can reject the null hypothesis, $\emptyset_2 = 0$). Therefore, those two estimates are statistically significant. The estimate for μ is -0.0080, which is not significant since p.value is large, so you cannot reject the null hypothesis, $\mu = 0$; you have a desired result.

3) Fitting MA(1) model

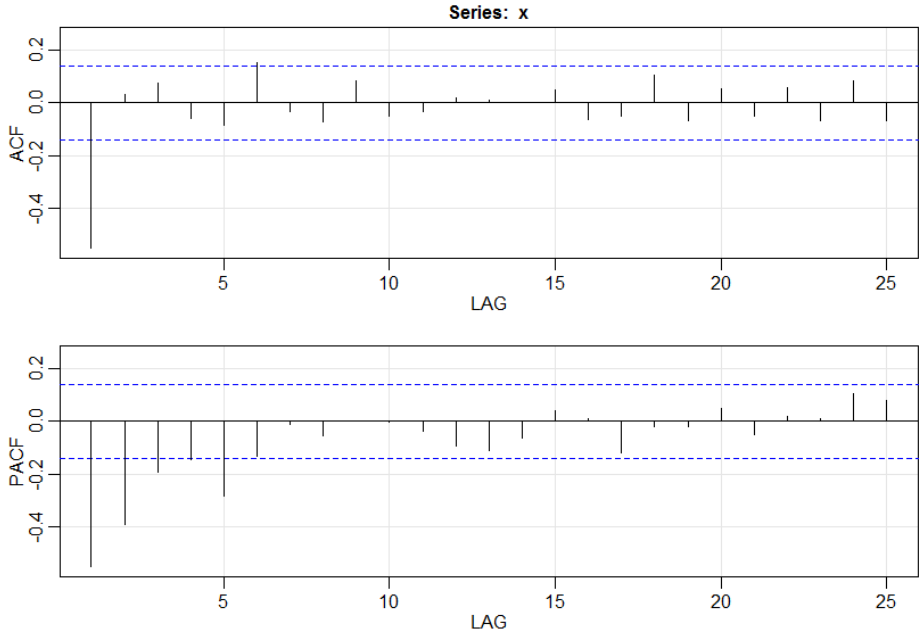
As you have seen in fitting AR model, in order to see how MA(1) model fits the data, you first generate data from an MA(1) model given by

$$X_t = \varepsilon_t - 0.85\varepsilon_{t-1}$$

```
# Generate the MA(1) process with 200 observations
x <- arima.sim(model = list(order = c(0, 0, 1), ma = -0.85), n = 200)

# Plot the generated series
plot(x)
```




```
# Plot the sample ACF and PACF
acf2(x)
```



Take a look sample ACF and sample PACF. The ACF cuts off after lag 1, and PACF tails off. So, it looks like MA(1); needless to say, **this process should be MA(1) because you created this series from the simulation of MA(1) model.** Now, you try to fit MA(1) to this series.

```
# Fit an MA(1) to the series generated from simulation
x_fit <- sarima(x, p = 0, d = 0, q = 1)


$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
          ma1    xmean
      -0.8610  -0.0069
s.e.   0.0335   0.0099

sigma^2 estimated as 0.9406:  log likelihood = -278.34,  aic = 562.68

$degrees_of_freedom
[1] 198

$ttable
      Estimate     SE  t.value p.value
ma1    -0.8610 0.0335 -25.6736  0.0000
xmean  -0.0069 0.0099  -0.6971  0.4866

$AIC
[1] 0.9587569

$AICc
[1] 0.9693691
```

```
$BIC
[1] -0.008259954
```

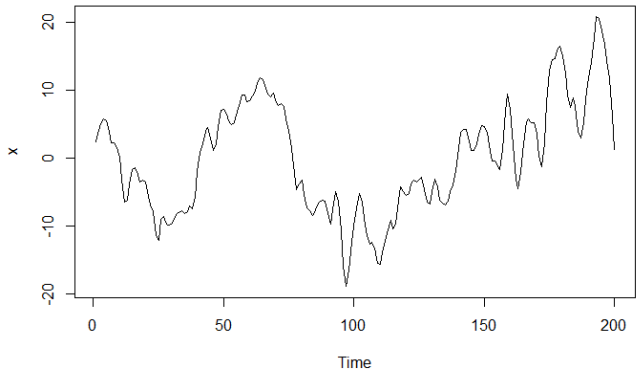
As discussed in fitting AR model, from the result above the estimate for θ_1 is significant and close to true coefficient -0.85, but the estimate for μ is not significant; desired result.

4) Fitting ARMA model

You are now fitting ARMA model by combining AR and MA model. No big differences from fitting AR and MA model. The only difference is that ARMA model has both components at the same time. You generate a series from the ARMA (1, 2) model given by

$$X_t = 0.9X_{t-1} + \varepsilon_t + 1.6\varepsilon_{t-1} + 0.8\varepsilon_{t-2}$$

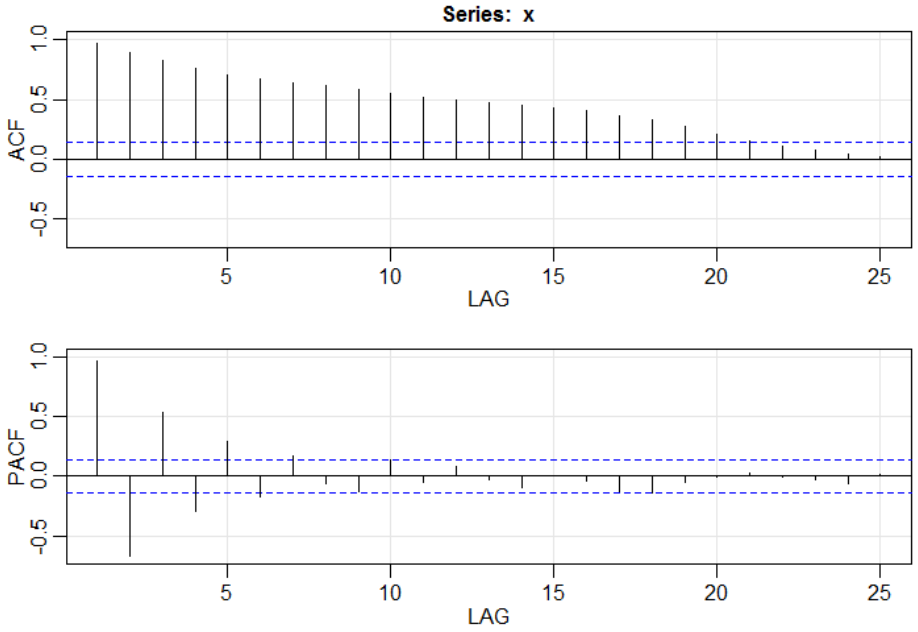
```
# Generate the ARMA(1, 2) process with 200 observations
x <- arima.sim(model = list(order = c(1, 0, 2), ar = 0.9, ma = c(1.6, 0.8)), n = 200)

# Plot the generated series
plot(x)
```




```
# Plot the sample ACF and PACF
acf2(x)
```

```
       ACF  PACF
 [1,] 0.97  0.97
 [2,] 0.90 -0.66
 [3,] 0.82  0.54
 [4,] 0.76 -0.30
 [5,] 0.71  0.29
 [6,] 0.67 -0.18
 [7,] 0.64  0.18
 [8,] 0.61 -0.06
 [9,] 0.59 -0.13
[10,] 0.55  0.14
[11,] 0.52 -0.05
[12,] 0.49  0.08
[13,] 0.47 -0.03
[14,] 0.45 -0.09
[15,] 0.43  0.00
[16,] 0.40 -0.04
[17,] 0.37 -0.13
[18,] 0.33 -0.13
[19,] 0.27 -0.06
```

```
[20,] 0.21 -0.01
[21,] 0.16  0.02
[22,] 0.11  0.00
[23,] 0.07 -0.03
[24,] 0.05 -0.06
[25,] 0.02  0.02
```



Both sample ACF and PACF tails off. There is no abrupt dampness. We cannot see typical patterns of sample ACF and PACF for both AR and MA. We may guess the series would have both components, but the orders are difficult to discern from the series as well as sample ACF and PACF. In our case, we know the true model, which is ARMA(1, 2). Thus, we just fit ARMA(1, 2) to the data, and the further discussion on how to select a best model fitting to the data will be made in later section.

```
# Fit an ARMA(1, 2) to the series generated from simulation
x_fit <- sarima(x, p = 1, d = 0, q = 2)
```

```
$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ar1     ma1     ma2   xmean
      0.9009  1.5830  0.7966  0.0908
s.e.  0.0297  0.0486  0.0477  2.4366

sigma^2 estimated as 1.123:  log likelihood = -299.12,  aic = 608.23

$degrees_of_freedom
[1] 196

$ttable
```

```
      Estimate     SE t.value p.value
ar1     0.9009 0.0297 30.3269  0.0000
ma1     1.5830 0.0486 32.5904  0.0000
ma2     0.7966 0.0477 16.6955  0.0000
xmean   0.0908 2.4366  0.0373  0.9703

$AIC
[1] 1.15615

$AICc
[1] 1.167696

$BIC
[1] 0.2221159
```

From the t-table above, the estimates for AR and MA components are all significant, and the estimate of μ is not significant: desired result.

6. Selection of a model

We have seen that it is difficult to identify a best model fitting well to the data, especially in case that a series has both AR and MA component, which is ARMA process. Thus, when you have a series that is hard to identify a certain model, the best approach to discern the model is to start with a low order model, and then add more parameters at a time to see the changes in the results. Then, how do we compare the results and select a best model? The very important criteria for model selection is "information criterion," such as Akaike information criterion (AIC) and Bayes information criterion (BIC). These information criteria are designed explicitly for model selection. Model selection criteria generally involve information criteria function calculations for each of the models (the details about information criteria are beyond the level of this course). The R calculates the AIC and BIC results based on their standard definitions, which include the constant term from the log likelihood function. Note that you retain the model with the ***smallest AIC and BIC value***.

Let us generate the data from simulation for ARMA(1, 1) process given by

$$X_t = 0.9X_{t-1} + \varepsilon_t + 1.6\varepsilon_{t-1}$$

Then, we fit the data to AR(1), MA(1), and ARMA(1, 1) at a time and see if there is an improvement based on AIC and BIC.

```
# Generate the ARMA(1, 1) process with 200 observations
x <- arima.sim(model = list(order = c(1, 0, 1), ar = 0.9, ma = 1.6), n = 200)
```

We fit the data to AR(1) model first.

```
# Fit an AR(1) to the series
ar1_fit <- sarima(x, p = 1, d = 0, q = 0)
ar1_fit
```

```
$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ar1   xmean
      0.9511  0.3370
s.e.  0.0201  2.4893

sigma^2 estimated as 3.532:  log likelihood = -411.14,  aic = 828.28

$degrees_of_freedom
[1] 198

$ttable
      Estimate     SE t.value p.value
ar1     0.9511 0.0201 47.2792  0.0000
xmean   0.3370 2.4893  0.1354  0.8924

$AIC
[1] 2.281785

$AICc
[1] 2.292397

$BIC
[1] 1.314768
```

When the data are fitted to AR(1) model, the AIC and BIC are 2.281785 and 1.314768, respectively. Now, we try to fit the data to MA(1) model and see if there is an improvement in terms of AIC and BIC.

```
# Fit a MA(1) to the series
ma1_fit <- sarima(x, p = 0, d = 0, q = 1)
ma1_fit

$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ma1    xmean
      0.9298  -0.0317
s.e.  0.0218   0.4729

sigma^2 estimated as 12.07:  log likelihood = -533.84,  aic = 1073.69

$degrees_of_freedom
[1] 198

$ttable
      Estimate     SE t.value p.value
ma1     0.9298 0.0218 42.6279  0.0000
xmean  -0.0317 0.4729 -0.0670  0.9467

$AIC
[1] 3.510567

$AICc
```

```
[1] 3.521179

$BIC
[1] 2.54355
```

The AIC and BIC increase when we fit the data to MA(1) model: no improvement in model fit. Let us try to fit the data to ARMA(1, 1).

```
# Fit a ARMA(1, 1) to the series
arma11_fit <- sarima(x, p = 1, d = 0, q = 1)
arma11_fit

$`fit`

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
    REPORT = 1, reltol = tol))

Coefficients:
         ar1     ma1   xmean
      0.9103  0.5689  0.0544
s.e.  0.0286  0.0608  1.9522

sigma^2 estimated as 2.752:  log likelihood = -386.5,  aic = 781

$degrees_of_freedom
[1] 197

$ttable
      Estimate     SE t.value p.value
ar1     0.9103 0.0286 31.8467  0.0000
ma1     0.5689 0.0608  9.3573  0.0000
xmean   0.0544 1.9522  0.0278  0.9778

$AIC
[1] 2.042195

$AICc
[1] 2.053221

$BIC
[1] 1.09167
```

Now, we have smaller AIC and BIC in ARMA(1, 1) than those in AR(1). If you fit the data to ARMA(1, 2) and ARMA(2, 1), you would be able to find there is no significant improvement rather than ARMA(1, 1). In most cases, we prefer the model that has fewest parameters to estimate, provided that each one of the candidate models is correctly specified. This is called the most ***parsimonious*** model of the set. Thus, ARMA(1, 1) would be a good candidate for the data in this case.

The AICs and BICs from each run are summarized as below:

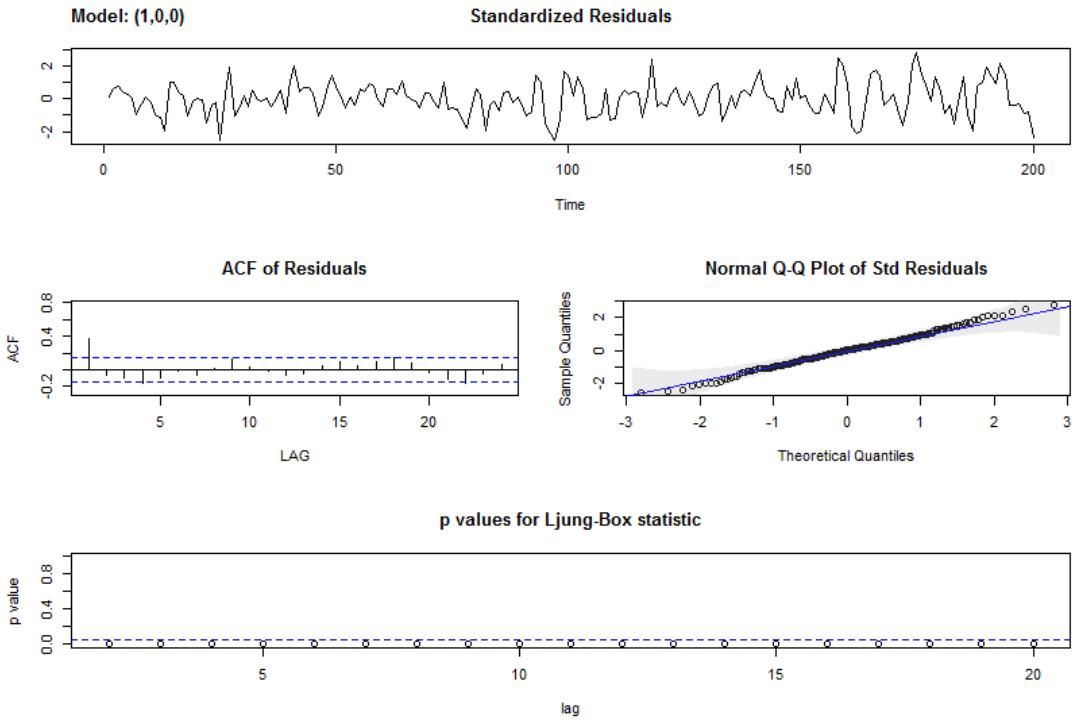
Model	AIC	BIC
AR(1)	2.281785	1.314768
MA(1)	3.510567	2.54355
ARMA(1, 1)	**2.042195**	**1.09167**

If we fit a model to data using **sarima(…)**, we have the outputs including a residual analysis, such as ***the standardized residuals***, ***the sample ACF of the residuals***, ***a normal Q-Q plot***, and ***the p.values corresponding to the Ljung-Box Q-statistic***. You always have to check these four residual plots when you fit a model.

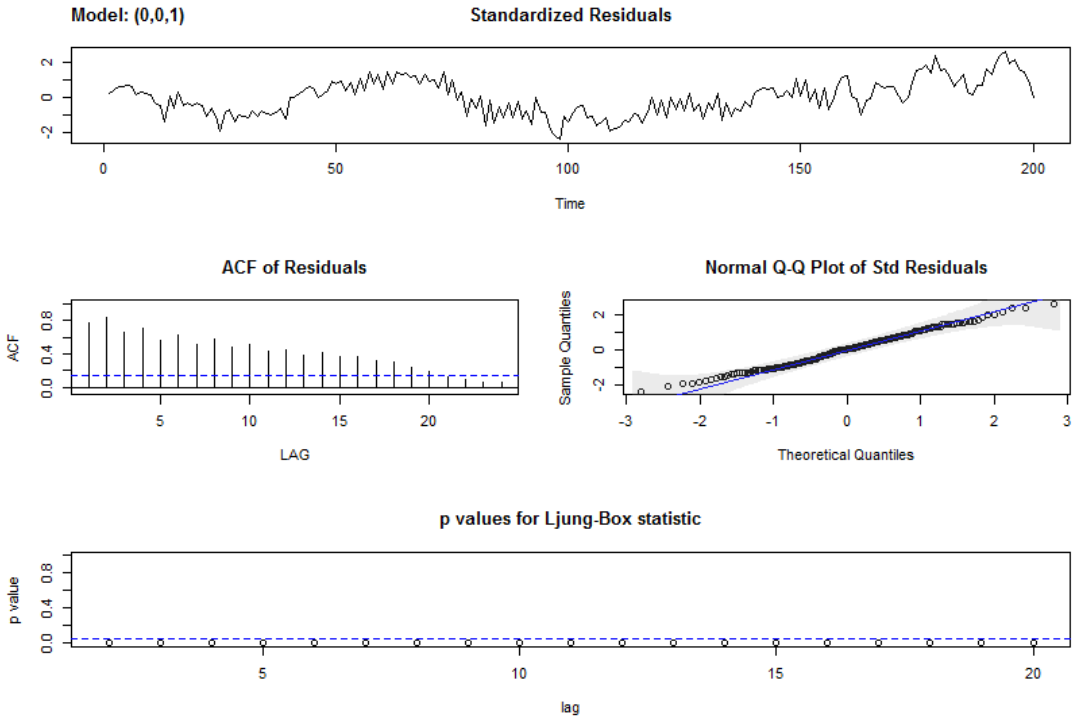
- The standardized residuals should look like a white noise process with mean 0 and variance 1. Otherwise, the model you fit is not a good candidate.
- The sample ACF of the residuals should be like that of a white noise, which means ACFs of all lags are close to 0 (within blue dotted line).
- Q-Q plot (or quantile-quantile plot) should show normal distributions. A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution (blue line indicates normal distribution), we should see the points forming a line that is roughly straight.
- The p.values for the Ljung-Box Q-statistics of the residuals indicate whether or not the ACFs of residuals are significantly different from zero.

We fitted three different ARMA models, which are AR(1), MA(1), ARMA(1, 1), to the data we created from the simulation of ARMA(1, 1). Each run of the models produces the following residual analysis.

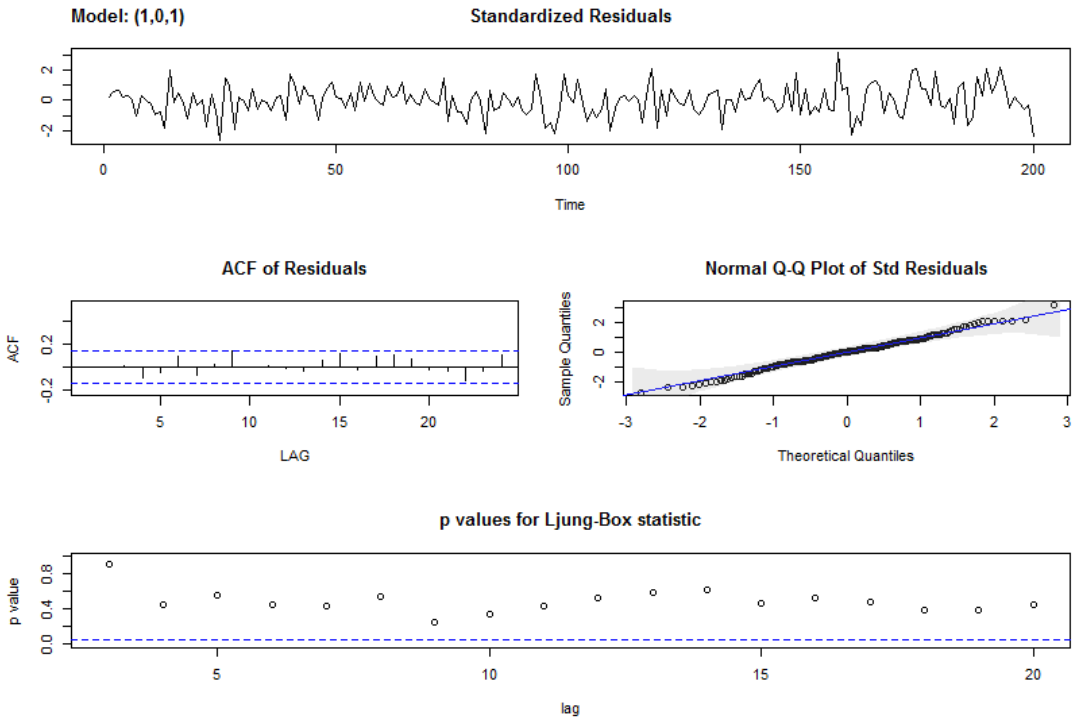
```
# Residual plots for AR(1)
ar1_fit
```



```
# Residual plots for MA(1)
ma1_fit
```




```
# Residual plots for ARMA(1, 1)
arma11_fit
```



Now, we look at the outputs above one by one.

	AR(1)	**MA(1)**	**ARMA(1, 1)**
Standardized residuals	Looks like white noise	Doesn't look like white noise	Looks like white noise
ACF of residuals	Some autocorrelations exist	Many autocorrelations exist	No autocorrelations exist
Normal Q-Q plot	Close to normal distribution	Close to normal distribution (but looks like not symmetric)	Close to normal distribution
p.values for Ljung-Box statistics	Autocorrelations of residuals are significantly different from 0	Autocorrelations of residuals are significantly different from 0	Autocorrelations of residuals are NOT significantly different from 0

7. Autoregressive integrated moving average (ARIMA) model

Autoregressive integrated moving average (ARIMA) model is also known as Box-Jenkins approach. Box and Jenkins claimed that non-stationary data can be made stationary by differencing the series. Thus, the ARIMA model is nothing special. A time series is called ARIMA(p, d, q) if the differenced series of order d is ARMA(p, q). For example, a process X_t is of ARIMA(1, 1, 0) if the following process Y_t is of ARMA(1, 0).

$$Y_t = \emptyset_1 Y_{t-1} + \varepsilon_t \,,$$

where $Y_t = X_t - X_{t-1}$. In this case, the model for the process X_t is an ARIMA(1, 1, 0) since the differenced process Y_t is an ARMA(1, 0) or an AR(1).

The general model for ARIMA(p, d, q) process is written as

$$Y_t = \emptyset_1 Y_{t-1} + \emptyset_2 Y_{t-2} + \cdots + \emptyset_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \,,$$

where Y_t is the differenced series of order d, $\emptyset's$ and $\theta's$ are unknown parameters (coefficients) and ε's are independent identically distributed error terms with zero mean. Here, Y_t is expressed in terms of its past values and the current and past values of error terms.

8. Put it all together: three steps to find a best model

Step 1: Ensuring stationarity

To model a time series with the ARIMA model (or Box-Jenkins approach), the series should be stationary. A stationary time series means that the series have no trends, no seasonality, and a constant mean and variance over time.

To convert a non-stationary process to a stationary process, we apply the differencing method, which finds the differences between consecutive values of a times series data. Those differenced values exhibit a new time series dataset which can be tested to see new correlations or other interesting statistical properties. We can apply the differencing method consecutively more than once, such as the first difference, the second difference, and so on, but more than second difference is unlikely. Most of non-stationary process becomes stationary process with the first difference.

We apply the appropriate differencing order *d* to make a time series stationary before we proceed to the next step.

Step 2: Identifying *p* and *q*

We identify the appropriate order of AR and MA processes by looking at the ACF and PACF.

For AR models, the ACF will dampen exponentially and the PACF will be used to identify the order *p* of the AR model. For example, if we have one significant spike at lag 1 on the PACF, we have an AR(1). If we have significant spikes at lag 1, 2, and 3 on the PACF, we have an AR(3).

For MA models, the PACF will dampen exponentially and the ACF will be used to identify the order *q* of the MA model. For example, if we have one significant spike at lag 1 on the ACF, we have an MA(1). If we have significant spikes at lag 1, 2, and 3 on the ACF, we have an MA(3).

In some cases (probably in most cases), we may not be able to identify *p* and *q* via ACF and PACF. In those cases, we can use AIC and BIC to do that (Remember that the smaller AIC and BIC, the better the model). Also, the coefficients of the model should be statistically significant.

Step 3: Check the residual ACF

After identifying the model, we also have to check the goodness of fit by plotting the ACF and residuals of the fitted model. If most of the sample autocorrelation coefficients of the residuals are close to 0, then the residuals are white noise indicating that the model fit is appropriate.