ARIMA modeling with R

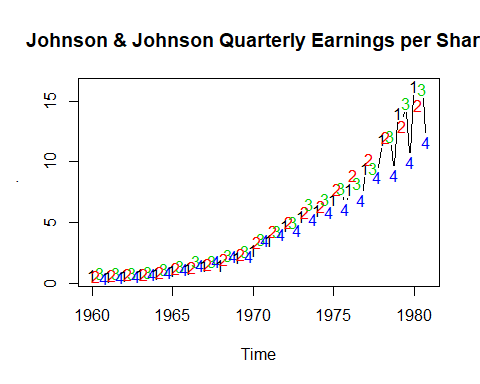
Koji Mizumura

Table of Contents

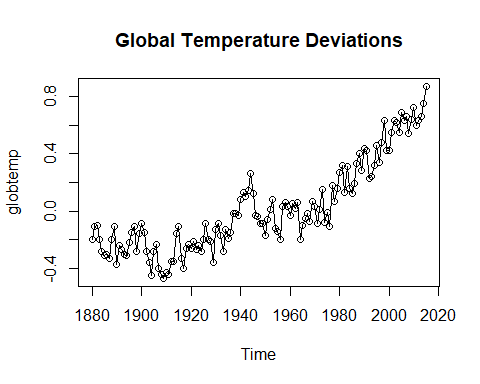
# Time series data and models

## First thing First

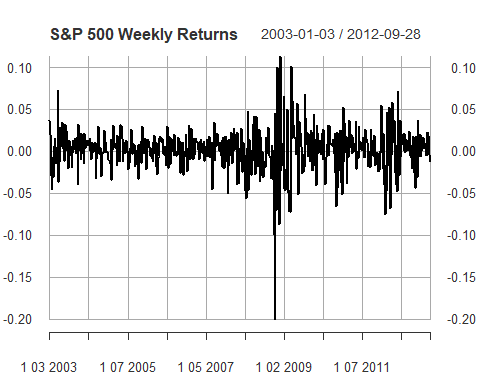
astsa::jj %>%   
 plot(main = "Johnson & Johnson Quarterly Earnings per Share",   
 type = "c")  
text(jj, labels=1:4, col=1:4)



library(astsa)  
plot(globtemp, main="Global Temperature Deviations", type="o")



plot(sp500w, main = "S&P 500 Weekly Returns")



Regression: , where is white noise

White Noise: - independent normals with common variance - is basic building block of time series

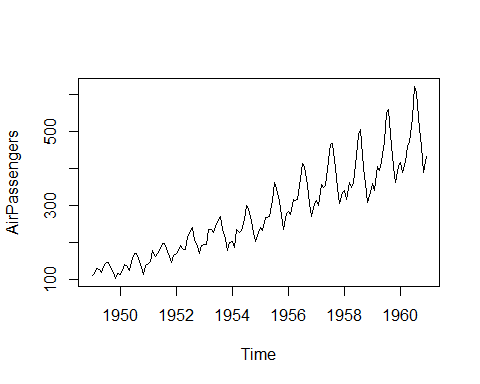
AutoRegression: $X\_t = \phiX\_{t-1} + \epsilon\_t$ ()

## Data Play

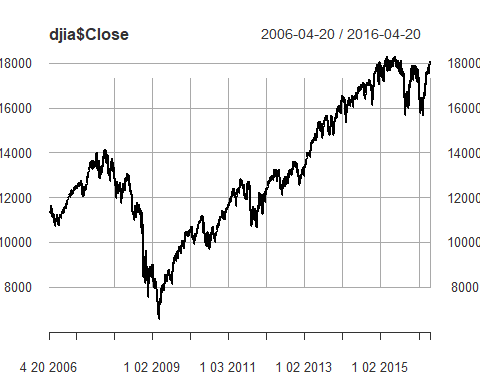
# View a detailed description of AirPassengers  
str(AirPassengers)

## Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...

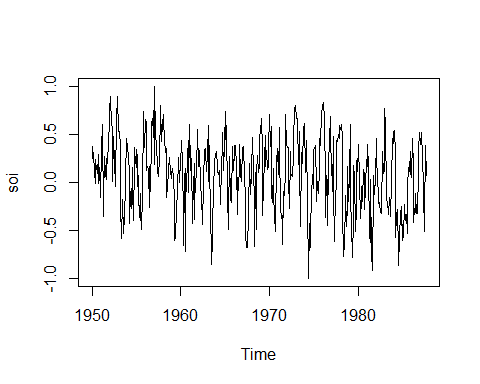
# Plot AirPassengers  
plot(AirPassengers)



# Plot the DJIA daily closings  
plot(djia$Close)



# Plot the Southern Oscillation Index  
plot(soi)



## Stationarity and non-stationarity

### Stationarity

A time series is stationary when it is “stable”, meaning - the mean is constant over time (no trend) - the correlation structure remains constant over time

Given data, we can estimate by averaging

For example, if the mean is constant, we can estimate it by sample average . Pairs can be used to estimate **correlation** on different lags. E.g.,

* (),(,), (, )… for lag 1
* (),(,), (, )… for lag 2

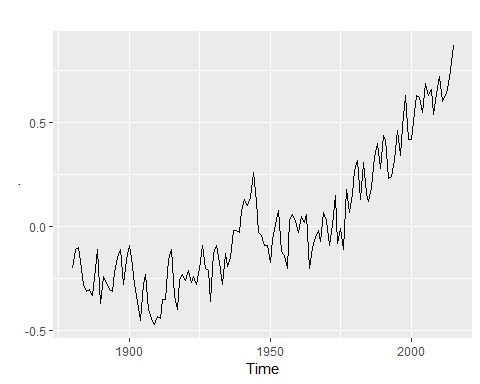
Southern oscillation index is reasonable to assume stationarity but perhaps some slight trend.

To estimate autoccrleation, compute the correlation coefficient between the time series and itself at various lags.

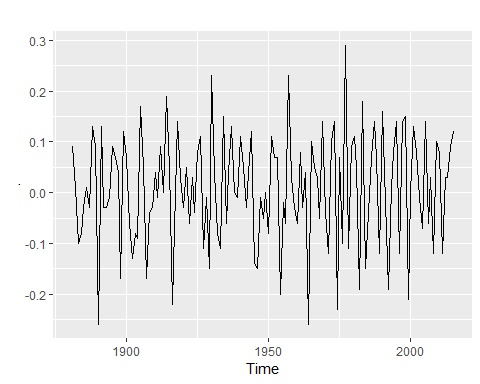
#### Random walk trend

Not stationary, but differenced data are stationary

globtemp %>% autoplot()

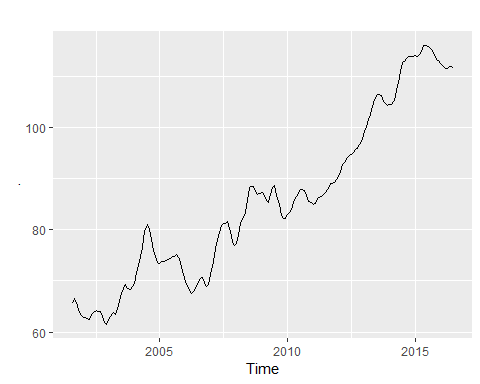


globtemp %>% diff() %>% autoplot()

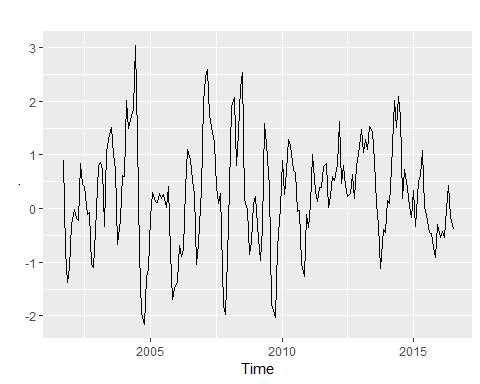


trend stationary … stationary around a trend, differencing still works!

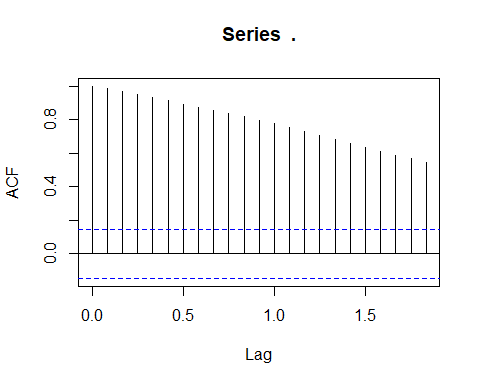
# plot  
chicken %>% autoplot()



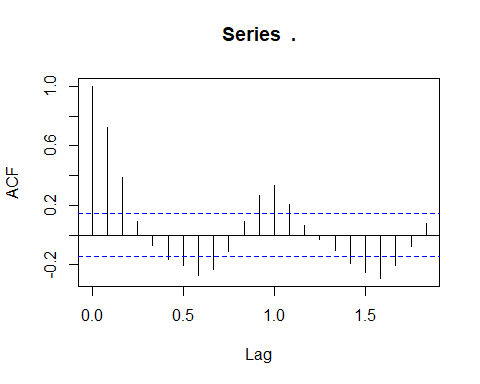
chicken %>% diff() %>% autoplot()



chicken %>% acf()



chicken %>% diff() %>% acf()



### Non stationarity in trend and variability

First log, and then difference.

## Differencing

As seen in the video, when a time series is trend stationary, it will have stationary behavior around a trend. A simple example is Yt=α+βt+Xt where Xt is stationary.

A different type of model for trend is random walk, which has the form Xt=Xt−1+Wt, where Wt is white noise. It is called a random walk because at time t the process is where it was at time t−1 plus a completely random movement. For a random walk with drift, a constant is added to the model and will cause the random walk to drift in the direction (positive or negative) of the drift.

We simulated and plotted data from these models. Note the difference in the behavior of the two models.

In both cases, simple differencing can remove the trend and coerce the data to stationarity. Differencing looks at the difference between the value of a time series at a certain point in time and its preceding value. That is, Xt−Xt−1 is computed.

To check that it works, you will difference each generated time series and plot the detrended series. If a time series is in x, then diff(x) will have the detrended series obtained by differencing the data. To plot the detrended series, simply use plot(diff(x)).

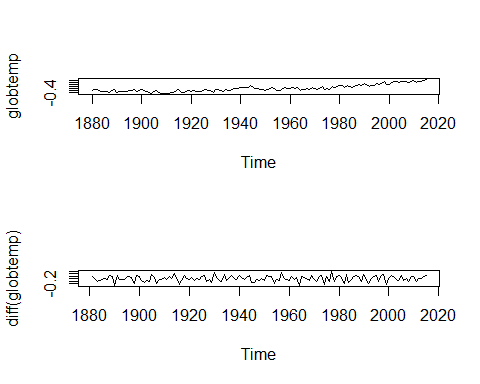
# Plot detrended y (trend stationary)  
plot(diff(y))  
  
# Plot detrended x (random walk)  
plot(diff(x))

## Detrending data

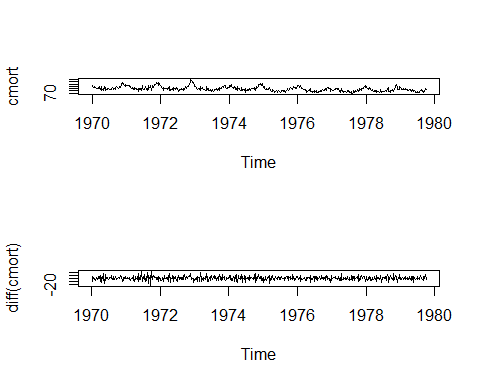
As you have seen in the previous exercise, differencing is generally good for removing trend from time series data. Recall that differencing looks at the difference between the value of a time series at a certain point in time and its preceding value.

In this exercise, you will use differencing diff() to detrend and plot real time series data.

# Plot globtemp and detrended globtemp  
par(mfrow = c(2,1))  
plot(globtemp)  
  
plot(diff(globtemp))



# Plot cmort and detrended cmort  
par(mfrow = c(2,1))  
plot(cmort)  
  
plot(diff(cmort))



## Dealing with trend and heteroskedasticity

Here, we will coerce nonstationary data to stationarrity by calculating the return or growth rate as follows.

Often time series are generated as meaning that the value of the time series observed at time t equals the value observed at time t−1 and a small percent change at time t.

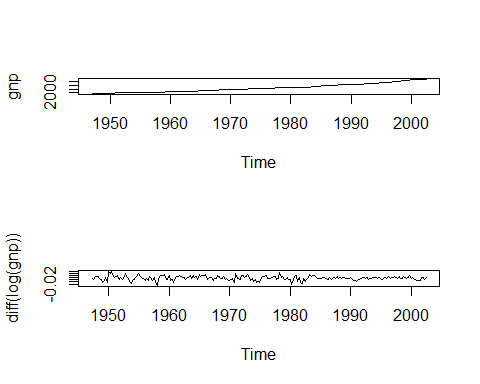
A simple deterministic example is putting money into a bank with a fixed interest p. In this case, Xt is the value of the account at time period t with an initial deposit of X0.

Typically, pt is referred to as the return or growth rate of a time series, and this process is often stable.

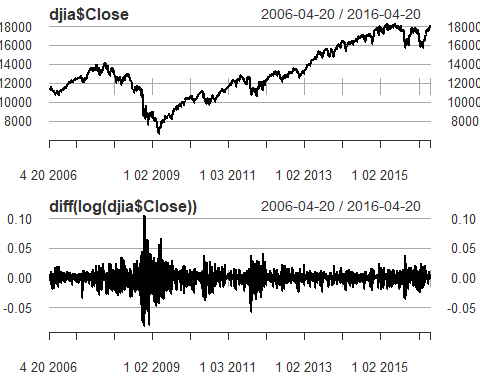
For reasons that are outside the scope of this course, it can be shown that the growth rate pt can be approximated by .

In R, pt is often calculated as diff(log(x)) and plotting it can be done in one line plot(diff(log(x))).

# astsa and xts are preloaded   
  
# Plot GNP series (gnp) and its growth rate  
par(mfrow = c(2,1))  
plot(gnp)  
plot(diff(log(gnp)))



# Plot DJIA closings (djia$Close) and its returns  
par(mfrow = c(2,1))  
plot(djia$Close)  
plot(diff(log(djia$Close)))



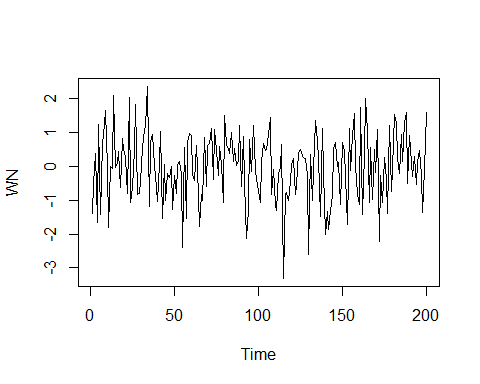
## Simulating ARMA Models

As we saw in the video, any stationary time series can be written as a linear combination of white noise. In addition, any ARMA model has this form, so it is a good choice for modeling stationary time series.

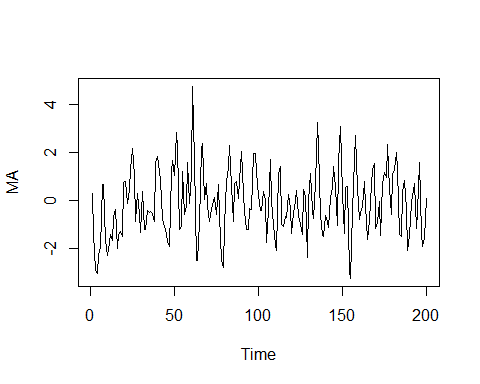
R provides a simple function called arima.sim() to generate data from an ARMA model. For example, the syntax for generating 100 observations from an MA(1) with parameter .9 is arima.sim(model = list(order = c(0, 0, 1), ma = .9 ), n = 100). You can also use order = c(0, 0, 0) to generate white noise.

In this exercise, you will generate data from various ARMA models. For each command, generate 200 observations and plot the result.

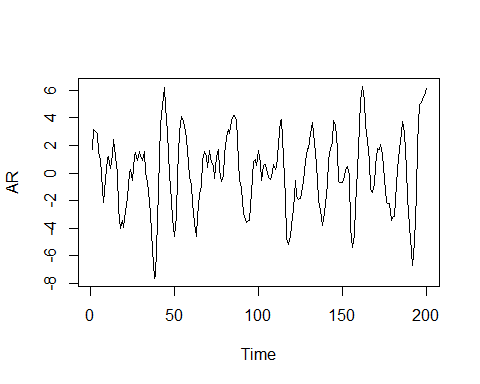
# Generate and plot white noise  
WN <- arima.sim(model = list(order = c(0, 0, 0)), n = 200)  
plot(WN)



# Generate and plot an MA(1) with parameter .9 by filtering the noise  
MA <- arima.sim(model = list(order = c(0, 0, 1), ma = .9), n = 200)   
plot(MA)



# Generate and plot an AR(1) with parameters 1.5 and -.75  
AR <- arima.sim(model = list(order = c(2, 0, 0), ar = c(1.5, -.75)), n = 200)   
  
plot(AR)



# Fitting ARMA models

## AR vs MA models

How to identify Ar and MA models - not visuallly, but by autocorrelation function (ACF, PACF)

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **AR(p)** | **MA(q)** | **ARMA(p,q)** |
| ACF | Tails off | Cuts off lag q | Tails off |
| PACF | Cuts off lag p | Tails off | Tails off |