ARIMA V/S FEED FORWARD NEURAL NETWORK-

PRESENTED BY- PUNEET PANDEY

Packages:

Quantmod: Quantmod is a Quantitative Financial Modelling and Trading Framework for R. It is a tool for data management and visualization. where quant traders can quickly and cleanly explore and build trading models.

Forecast: is a generic function for forecasting from time series or time series models. The function invokes particular methods which depend on the class of the first argument. Auto.arima() routine lives inside forecast.

Tseries: Time series Analysis and computational finance package contains some statistical routines like Augmented Dickey Fuller test, Ljung Box test etc.

Timeseries: Time Series Class, function to generate or identify time series data

Dplyr: Dplyr is a grammar of data manipulation, it contains routines like summarise (), groupby (), arrange () etc.

Data:

I will be using Yes Bank's Data for stock prices prediction. The data points start from 26th of July 2005 to 27th November 2020. Yes Bank was established in 2004 so approximately we are covering from the start of establishment till now. The data has 3774 rows (data points) and 7 columns namely dates, open, high, low, close, adjusted close, volumes. Date is obviously the date at which the data point was taken, open is the price at which the stock was opened in the day or that date, high is the highest the stock achieved on that date, similarly low, adjusted close price amends a stock closing price to reflect that stocks value after accounting for any corporate actions. It is more formally used than closing prices as it tends to give the right information about the stock. We'll also be using the Adj Closed prices in prediction of stock prices. We'll be using different algorithms and will be monitoring their efficiency.

Introduction:

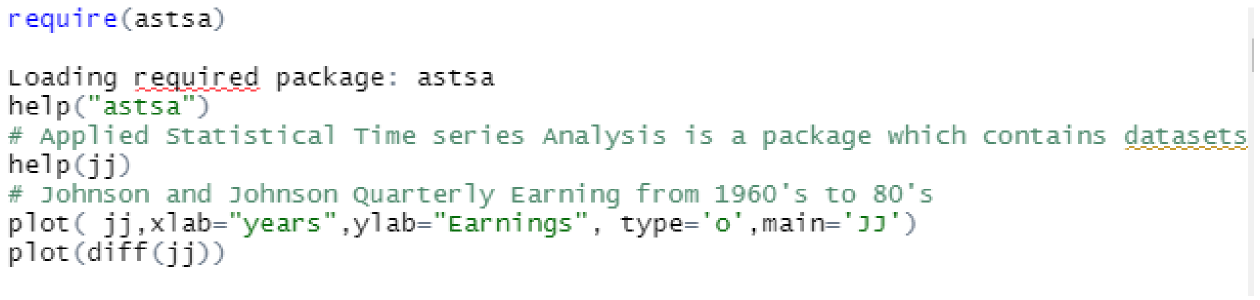
For future some future information about some data based on present and past given information of the data, forecasting algorithms are used. The present/past data are extracted and prepared for prediction of future values of given variable such as ADJ closed. In this project we would focus on quantitative forecasting techniques, would do some exploratory data analysis on data to find out about our data that is it standing on our assumptions of forecasting or not such as stationarity etc. and also behavior of our data points i.e. trends seasonality etc. The main motivation of this study comes from understanding what happens before it happens. Obviously, we would not be exactly predicting what the stock prices be at a certain point because if I could I wouldn't be writing this project as I would have been the richest person in this world, as data always comes with noise and our prediction would include some confidence interval in which our prediction would most probably lie. The main objective of this project is to show the number of models we are trying to fit in the data and compare them.

Pre-requisites:

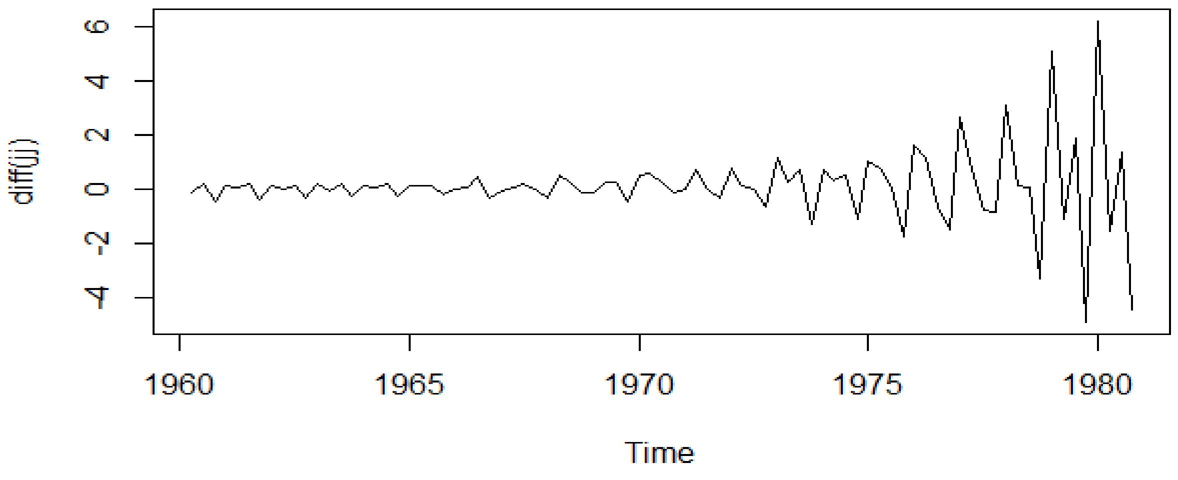
1.Stationarity:

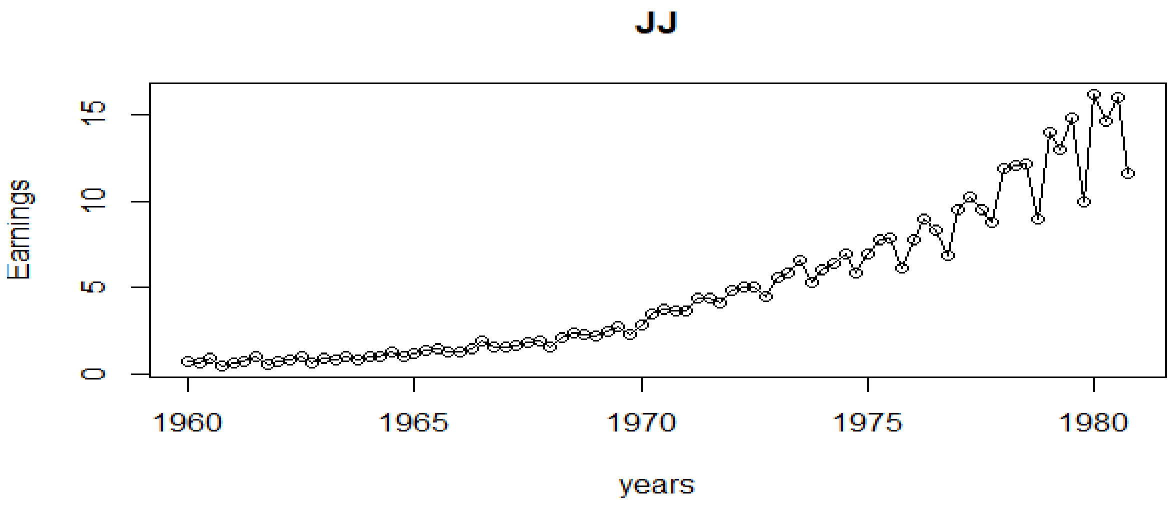
Stationarity is no systematic change in mean, variance and periodic variation of the data set. We always try to model a stationary time series and whenever we encounter a non-stationary time series, we try to change it to stationary one, stationarity basically means one section of data should be similar to other section. A time series/Stochastic process is said to be strictly Stationary if the joint distribution of $X(t_1)$, $X(t_2)$,....,$X(t_k)$ is same as $X(t_1+\tau)$,$X(t_2+\tau)$,....,$X(t_k+\tau)$ this is mathematically exactly what we said at the start. Remember, stationarity basically means one section of data should be similar to other section, τ here is lag.

While plotting Johnsons and Johnsons quarterly earnings on R we get a non-stationary plot but that can change it to stationary plot by difference operator which plots X_t-X_{t-1}, both the plots are given below



```
require(astsa)

Loading required package: astsa
help("astsa")
# Applied Statistical Time series Analysis is a package which contains datasets
help(jj)
# Johnson and Johnson Quarterly Earning from 1960's to 80's
plot( jj,xlab="years",ylab="Earnings", type='o',main='JJ')
plot(diff(jj))
```





2.ACF:

A random variable is a function from sample space to real numbers, it can be discrete as well as continuous. Covariance is the linear relationship between two random variables.

$$\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X) = E[(X-E(X))(Y-E(Y))]$$

where expectation of X is mean of X and Exp of Y is mean of Y so

$$\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X) = E[(X-\mu(X)(Y-\mu(Y))]$$

Stochastic processes are non-deterministic process that is the state of the system at time t doesn't depend upon last t-1 steps of the model. So, a set of random variables X1, X2, X3...... is stochastic process and the realization of X1, .. is time series. each of the Xi's would have a different distribution N~ (μ_i, σ^2_i)

$$\mathrm{Cov}(X_i, Xj) = \mathrm{Cov}(X_i, X_j) = E[(Xi-\mu(Xi))(X_j-\mu(X_j))]$$

$$\mathrm{Cov}(X_i, X_i) = E[(Xi-\mu(Xi)(Xi-\mu(Xi))] = E[(xi-\mu(xi))^2] = \text{Variance of } Xi$$

$$\gamma_k = \gamma(t, t+k) \sim c_k$$

This is Auto covariance function, we see that ACF depends only on the time difference k and not the time t but Why?, because we are working with stationary time series and remember we said that in Stationary TS, one section of data should be similar to other section, so it doesn't matter if you take X20,X30 or X40,X50, ACF remains same.

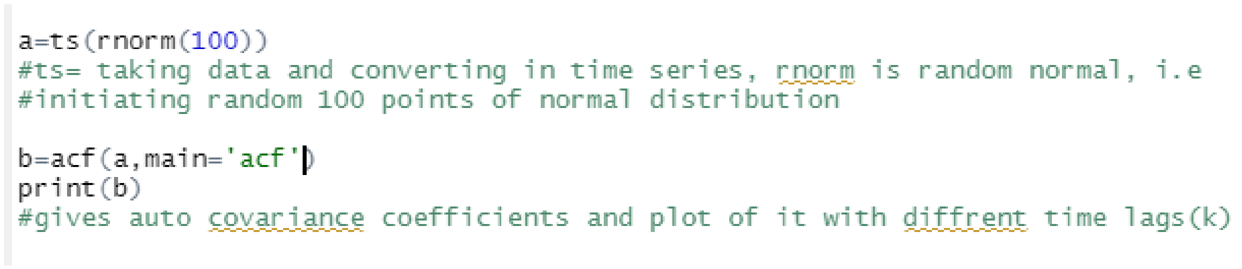
AUTOCORELATION FUNCTION: We assume weak stationarity, Auto Correlation Function (ACF) gives you the autocorrelation coefficient which is just the ratio of Autocovariance coefficient at k and time 0, So ACF between X_t and X_{t+k} would be say ρ_k

$\rho_k=\gamma_k/\gamma_0$

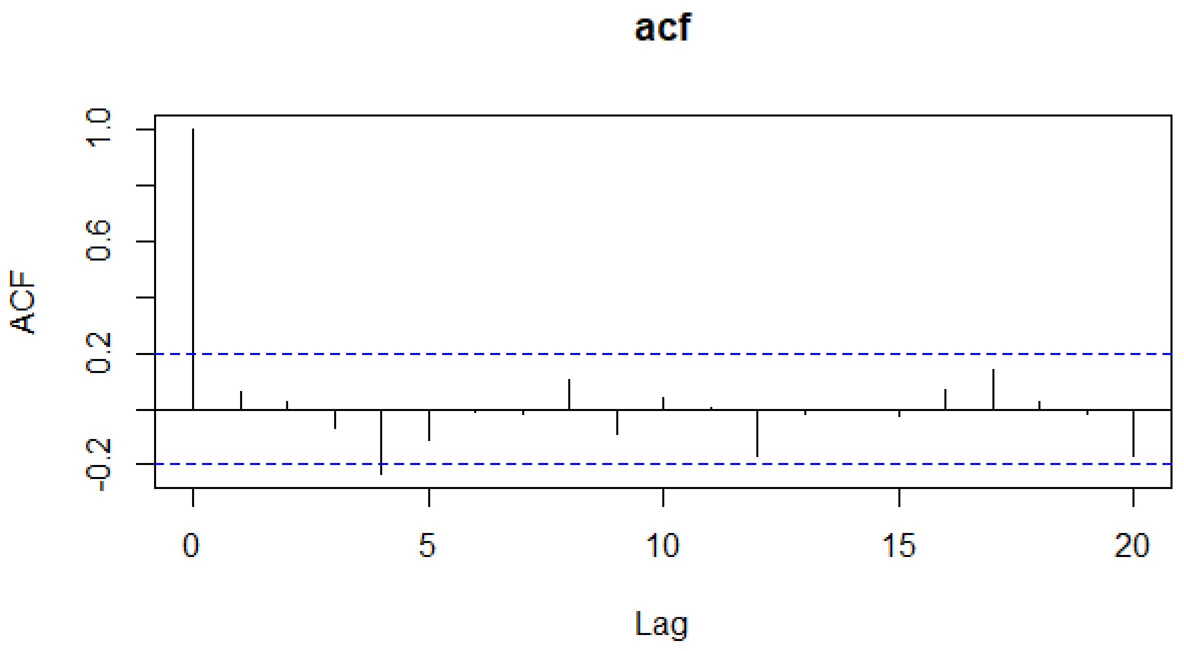
putting the value to γ_k and γ_0 we get:

$$\rho_k= [\Sigma_{\tau=1}^{N-\kappa} (x_t-\mu)\ (x_{t+k}-\mu)/\ \Sigma_{\tau=1}^{\kappa} (x_t-\mu)^2]$$

Let’s Generate ACF, following is a simulation in R



```
a=ts(rnorm(100))
#ts= taking data and converting in time series, rnorm is random normal, i.e
#initiating random 100 points of normal distribution

b=acf(a,main='acf')
print(b)
#gives auto covariance coefficients and plot of it with diffrent time lags(k)
```



Look above when you used ACF it gave you a graph called correlogram which always

starts with 1 at t=0 as at t=0 $\rho_k=\gamma_0/\gamma_0=1$.

SUMMARY: ACF gives you correlation in the data between time lags!

3.Moving Average Processes [MA(q)]:
Suppose X_t is the price of stock at time t and Z_t be some announcement of the company that is affecting the stock price obviously time t, that is Z_t i.e. noise would affect the price of the stock for t days suppose it is affecting for 2 days

$$X_t = Z_t + a*Z_{(t-1)} + b*Z_{(t-2)} \text{ is MA(2) process}$$

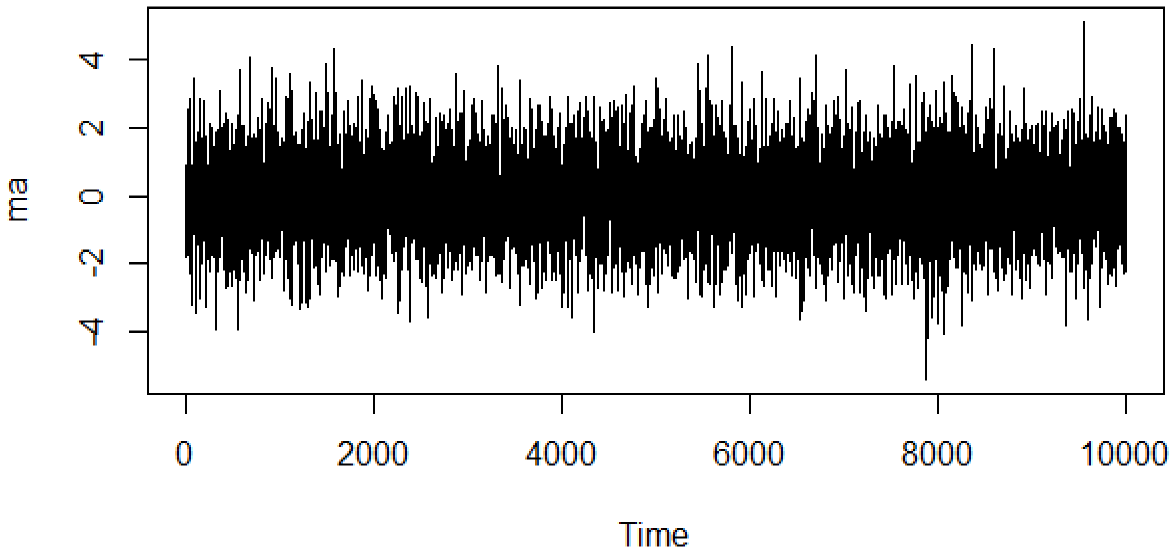
Similarly define for MA(q) process:

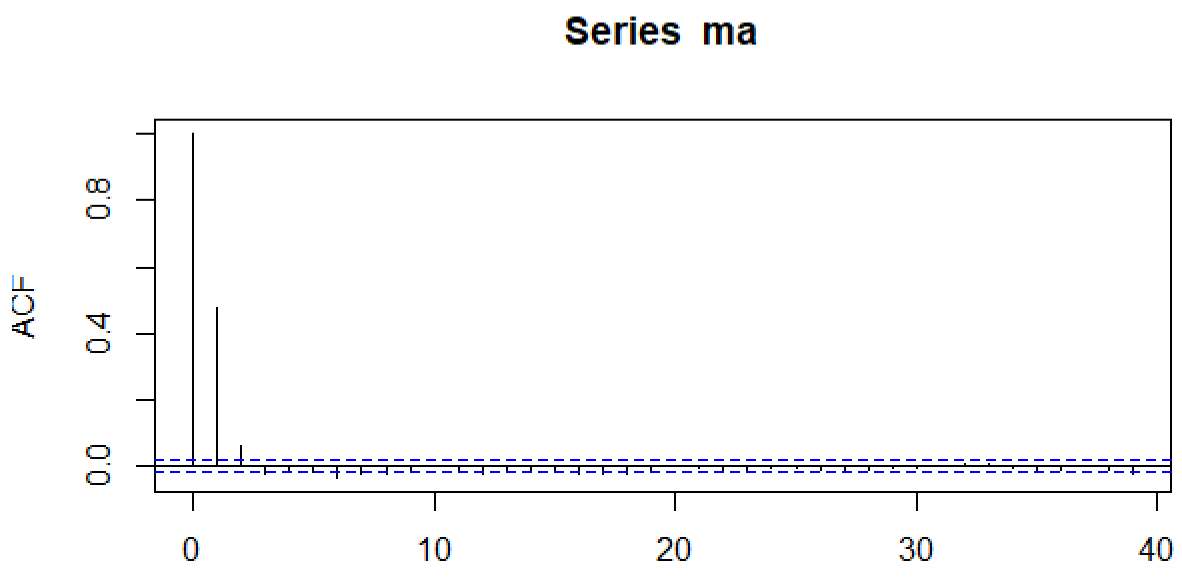
$$X_t = Z_t + a*Z_{(t-1)} + b*Z_{(t-2)} + \ldots\ldots + e*Z_{(t-q)} \text{ is MA(q) process.}$$

NOTE: Z_i's are i.i.d and N ~ (μ,σ^2). An MA(q) process is defined where the value of time series at t can be written as some weighted average of noise/experiment.

Simulation of a MA(q) process in R:

```
noise=rnorm(10000)

ma2= NULL

for (i in 3:10000){
  ma2[i]=noise[i]+0.6*noise[i-1]+0.1*noise[i-2]
  }
ma= ma2[3:10000]

ma=ts(ma)


plot(ma)
acf(ma)
```





Moving Average Processes are stationary! A computational proof using the equation of Covariance and Expectation and understanding the non-contribution of k>q in Expectation shows that the covariance of X_t, X_{t+q} doesn't depend on t.

Looking at the ACF we get a very keen observation about MA(q) processes that is after q lags it vanishes (of Course don't take into the consideration of 1st ACF its always going to be 1) which is obvious because innovation/noise/announcements are affecting the price of stock for 2 days only.

4.Autoregressive Processes [AR(p)]:

Just like a moving average process is Z_t in addition with weighted average of all previous noise/innovations an autoregressive process is Z_t in addition with weight average of history that is

$$X_t = \theta_0 Z_t + \theta_1 X_{(t-1)} + \ldots + \theta_p X_{(t-p)}$$

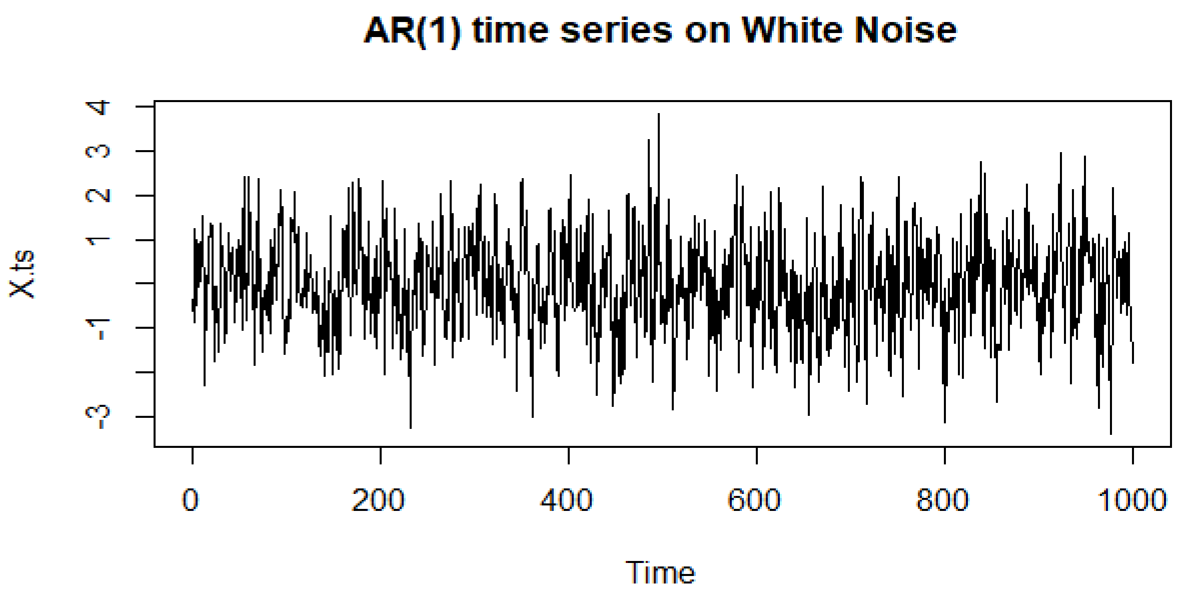
A random walk is an AR (1) process, as $\mu=0$, $p=1$, $\theta_1=1$:

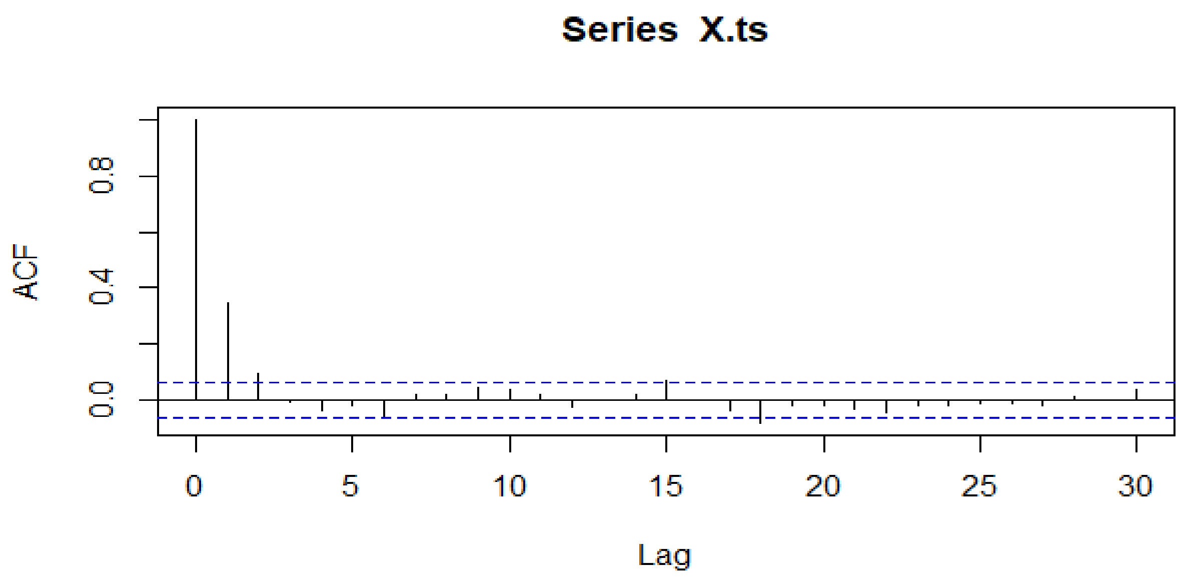
$$X_t = X_{(t-1)} + Z_t$$

An autoregressive process need not be stationary, as random walk is not a stationary process because if you find out the covariance of X_t ,$X_{(t+k)}$ it comes out to be t dependent.

Simulations in R:

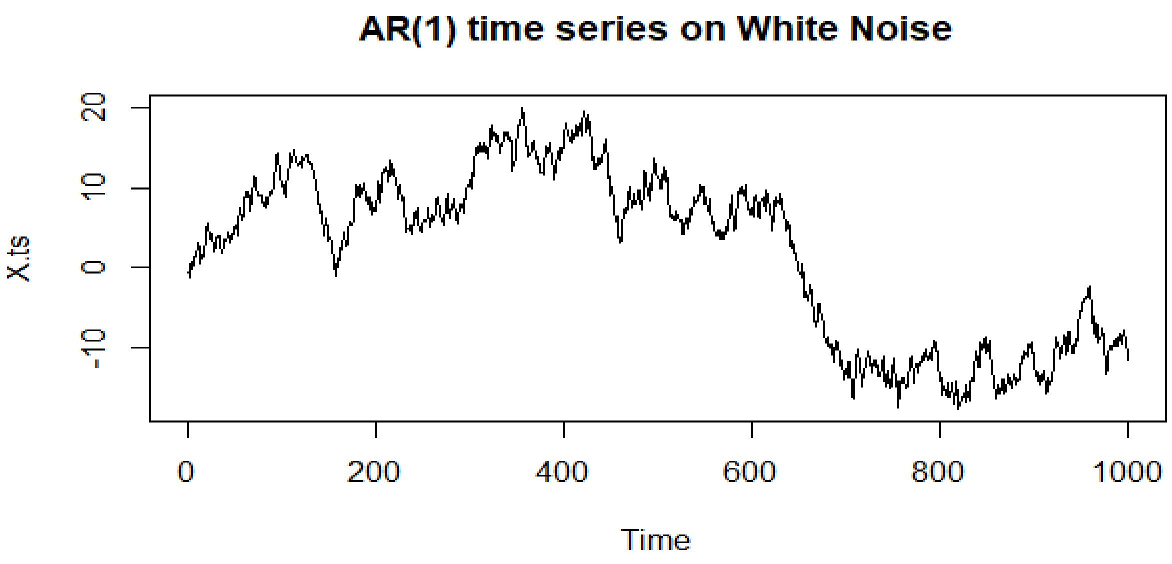
```r
#Simulating AR process

set.seed(1)
N=1000;
phi=0.4;
Z=rnorm(N,0,1);
X=NULL;
X[1]=Z[1];

for(i in 2:N){
  X[i]=phi*X[i-1]+Z[i]
}

X.ts=ts(X)
plot(X.ts,main='AR(1) time series on White Noise',phi=0.4)
X.acf=acf(X.ts)
```

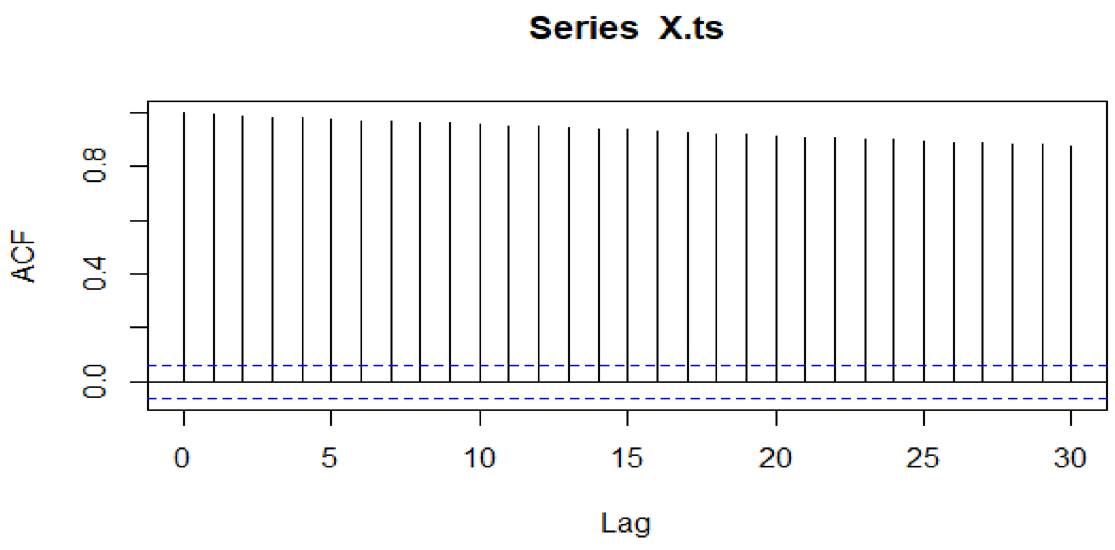




In the previous example change ϕ=1 and see how stationarity is long gone and ACF becomes highly corelated

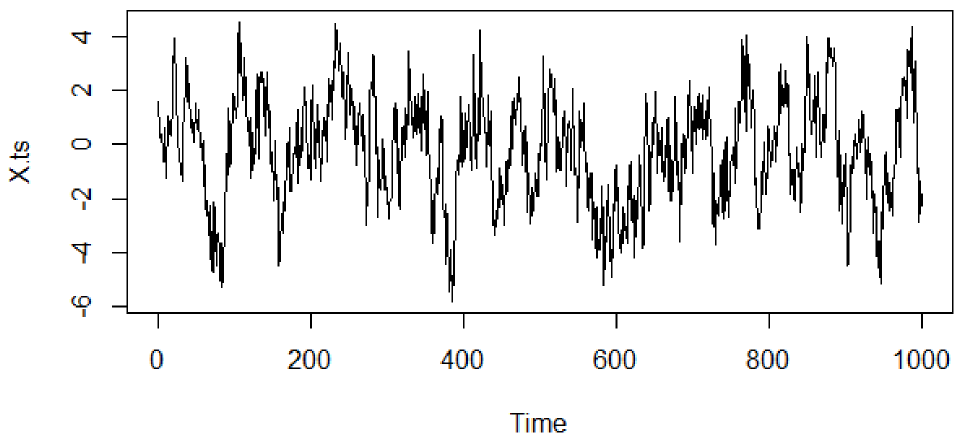
```
set.seed(1)
N=1000;
phi=1;
Z=rnorm(N,0,1);
X=NULL;
X[1]=Z[1];

for(i in 2:N){
  X[i]=phi*X[i-1]+Z[i]
}

X.ts=ts(X)
par(mfrow=c(1,1))
plot(X.ts,main='AR(1) time series on White Noise',phi=0.4)
X.acf=acf(X.ts)
```

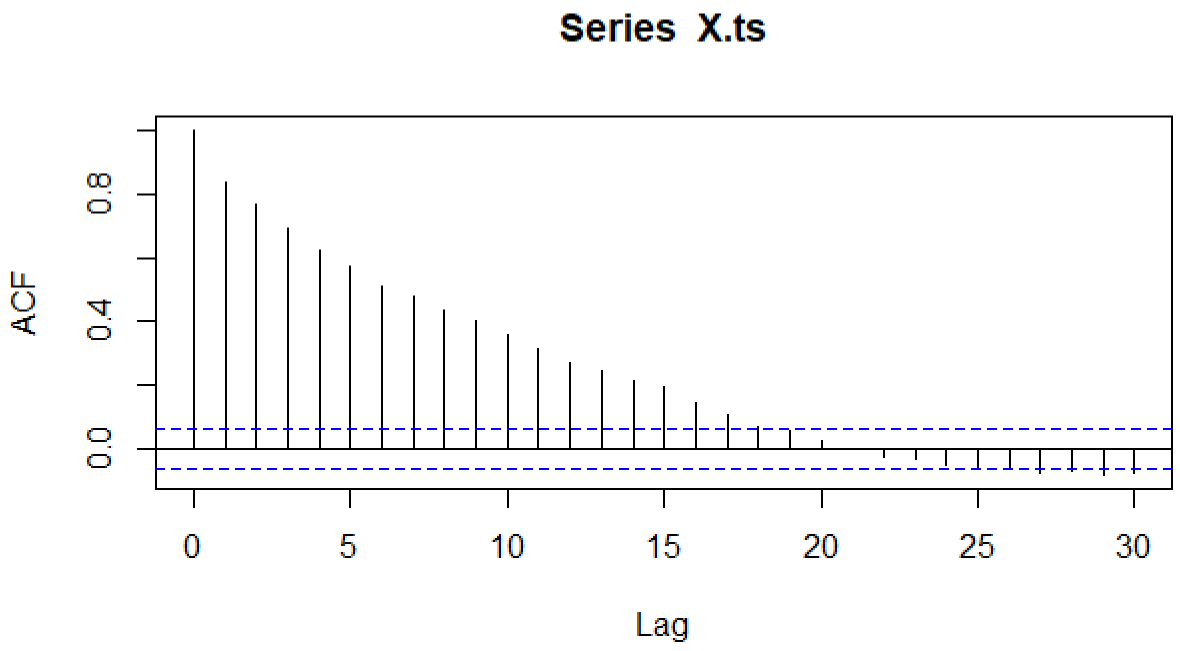




But why is it happening that for ϕ=1, the series lost its stationarity? This will be answered later, lets check some more simulations

```
#Simulating AR(2) process!
#We are going be a little more sophisticated and use arima.sim

set.seed(1)
X.ts<- arima.sim(list(ar=c(0.7,0.2)),n=1000)
plot(X.ts)
acf.ts=acf(X.ts)

#you can use negative phi's and it would introduce negative corelations!!
```



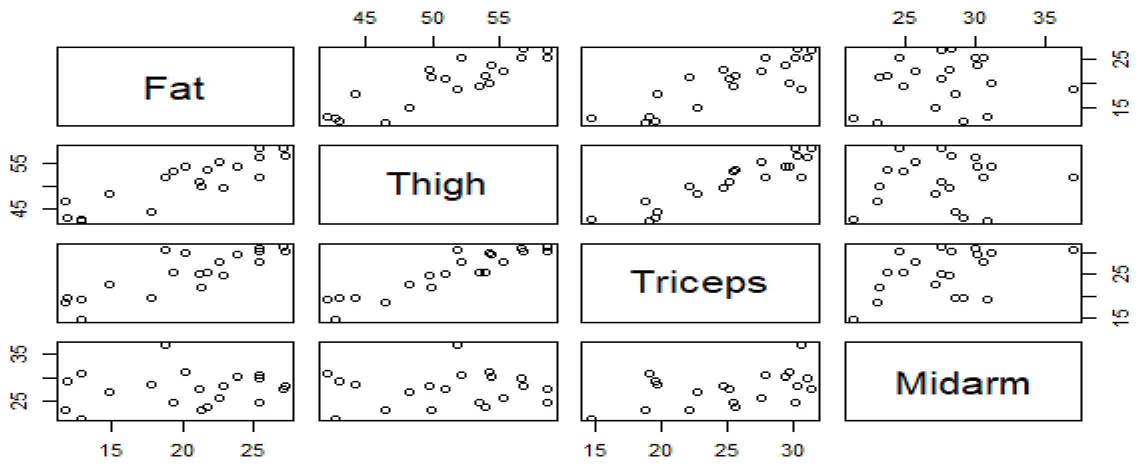


Looking at these simulations we are sure that ACF is not telling us about the order of Autoregressive processes just like MA(q) process (cuts after lag q), the way of finding out is by studying about Partial Auto correlation function (PACF)

5.PACF:

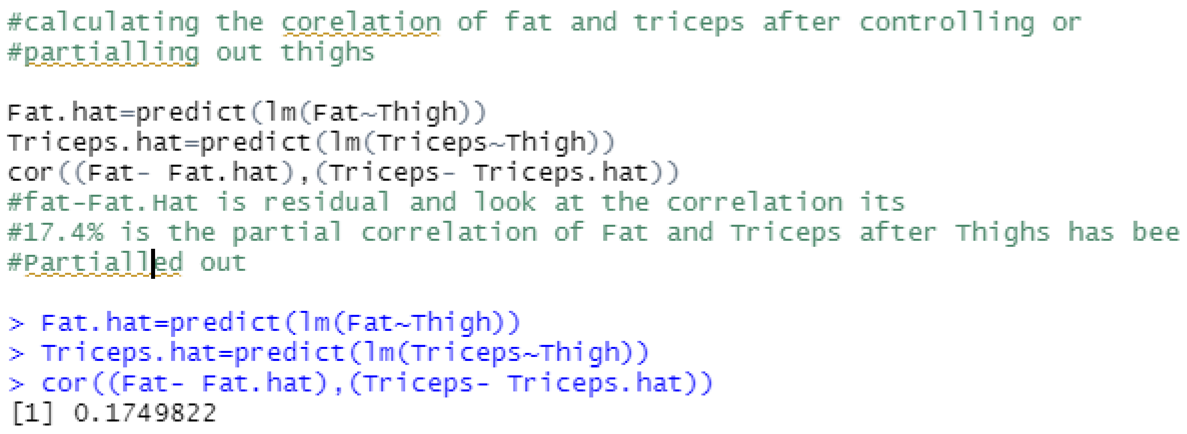
The ACF for a MA(q) process cuts off at lag q and pacf for an AR(p) process cuts off at lag p. We'll go deeper into the PACF concept by bodyfat dataset, treat it as a regression problem. Install and load isdals package. It is expensive and cumbersome to find the bodyfat in humans as it involved immersion of a person in water. This dataset provides information on bodyfat, triceps skinfold, thickness, thigh area etc. for 20 healthy females aged 20 to 34. It is desirable if a model could fit and provide reliable prediction of the amount of bodyfat.

```
install.packages("isdals")
library(isdals)
data(bodyfat)
attach(bodyfat)
pairs(cbind(Fat,Thigh,Triceps,Midarm))

#Lets look at the corelations

cor(cbind(Fat,Thigh,Triceps,Midarm))
```



```
              Fat     Thigh   Triceps    Midarm
Fat     1.0000000 0.8780896 0.8432654 0.1424440
Thigh   0.8780896 1.0000000 0.9238425 0.0846675
Triceps 0.8432654 0.9238425 1.0000000 0.4577772
Midarm  0.1424440 0.0846675 0.4577772 1.0000000
```

Fat and thigh, Fat and Triceps have a big correlation, but also thigh and triceps have a good correlation too, for a model to fit it would be better to have low correlation because if not we'll be living elusive statistical power, coefficient would be loosely estimated.



```
#calculating the corelation of fat and triceps after controlling or
#partialling out thighs

Fat.hat=predict(lm(Fat~Thigh))
Triceps.hat=predict(lm(Triceps~Thigh))
cor((Fat- Fat.hat),(Triceps- Triceps.hat))
#fat-Fat.Hat is residual and look at the correlation its
#17.4% is the partial correlation of Fat and Triceps after Thighs has bee
#Partialled out

> Fat.hat=predict(lm(Fat~Thigh))
> Triceps.hat=predict(lm(Triceps~Thigh))
> cor((Fat- Fat.hat),(Triceps- Triceps.hat))
[1] 0.1749822
```

Find all the partial correlation by using pcor () routine. The pcor is finding out the partial correlation between two variables. The partial correlation between two variables say Fat and Triceps is formed by deleting the relationship of other variables such as Thighs and midarm on Fat and Triceps and then finding the correlation!

But what is the relationship between this and AR(p) process?

Estimate by looking backward over the last several terms and denote by $\hat{X}_{(t+h)}$

$$\hat{X}_{(t+h)} = \beta_1 X_{(t+h-1)} + \beta_2 X_{(t+h-2)} + \ldots. + \beta_{(h-1)} X_{(t+1)}$$

The subscript of β is just telling how far you are from the thing you are predicting. But due to stationarity we can also do this

$$\hat{X}_t = \beta_1 X_{(t+1)} + \beta_2 X_{(t+2)} + \ldots. + \beta_{(h)} X_{(t+h)}$$

We have got a predictor of $X_{(t+h)}$ and X_t , partial out the intervening or the in between random variable just like we partial out thighs and midarm from fat and triceps.

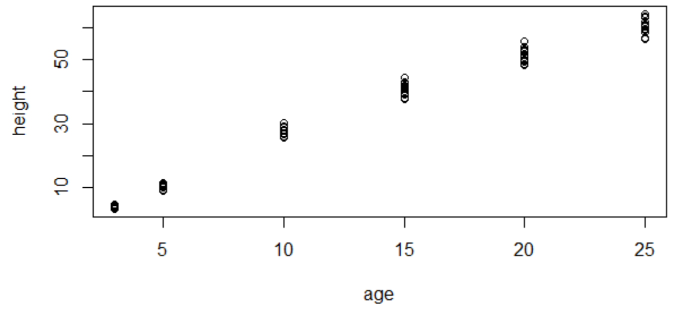
$$\text{Corr } [(X_{(t+h)} - \hat{X}_{(t+h)}), (X_t - \hat{X}_t)]$$

We remove the linear effects of all the terms in between $(X_{(t+1)}, X_{(t+2)} \ldots., X_{(t+h-1)})$. The excess correlation at lag =k not accounted for by a (k-1)st order model is the partial correlation at lag=k(PACF).

6.Akaike Information Criteria (AIC):

What is the order of the model? Is it AR(2)? Why cant be it AR(3)? AIC measures the quality of the model, numerically AIC could be think of equivalent to SSE for a regression problem, you know weather to use linear/polynomial/etc. regression based on the value of SSE. For example, in loblolly dataset just be seeing you can say like oh! Use quadratic regression, but to be sure you find out Multiple R^2 and Adjusted R^2 values.

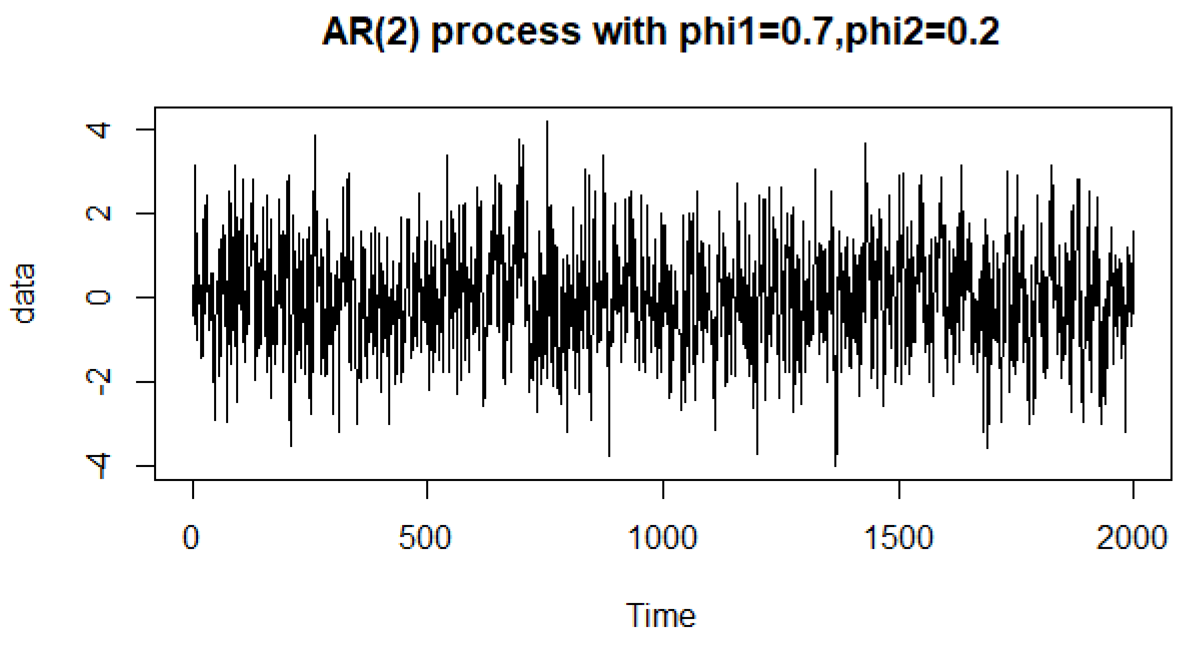
```
attach(Loblolly)
plot(age,height)
```

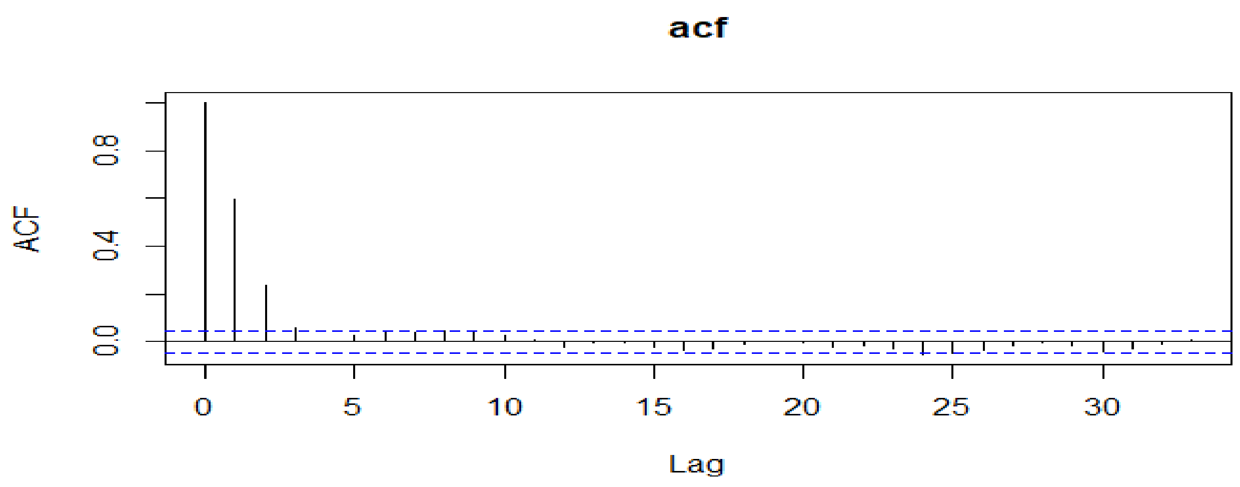


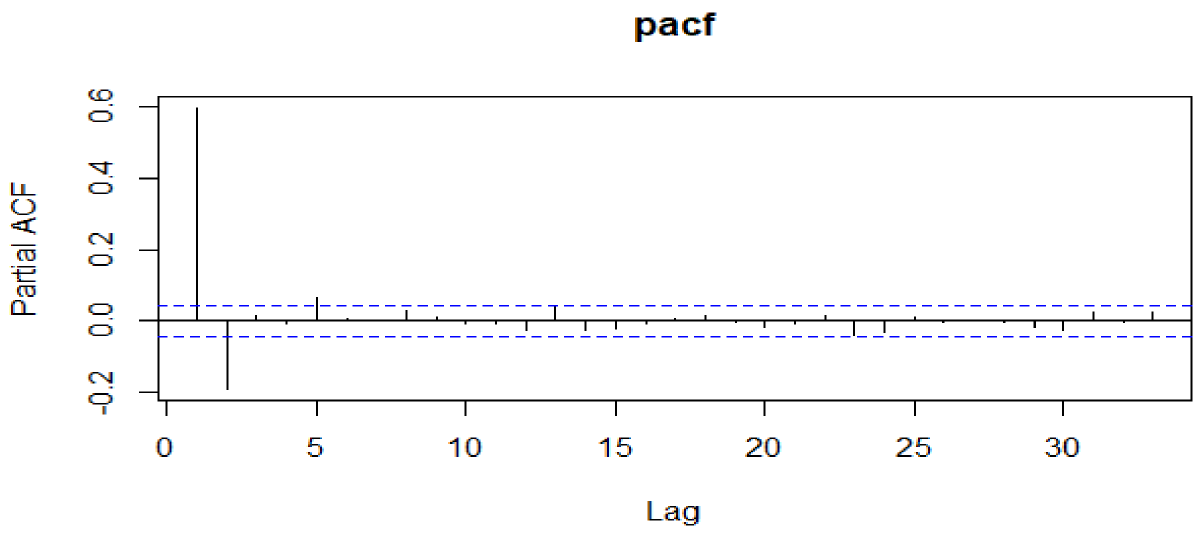
Adjusted R^2 is analogous to AIC in the sense that it makes you pay a penalty as you bring higher order terms into the model. The ACF or PACF may not be enough to know what model you have , you may just need a less subjective numerical evidence to support your claim which is AIC.

Simulation in R:

```
#AKAIKE INFORMATION CRITERIA
#Lets simulate AR(2) process X_t= Zt+0.7Xt-1+0.2Xt-2

set.seed(43)
data=arima.sim(list(order=c(2,0,0),ar=c(0.7,-0.2)),n=2000)
plot(data)
acf(data)
pacf(data)
arima(data,order=c(2,0,0))
arima(data,order=c(3,0,0))
```







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

Coefficients:
         ar1      ar2  intercept
      0.7111  -0.1912     0.0057
s.e.  0.0219   0.0220     0.0465

sigma^2 estimated as 0.9985:  log likelihood = -2836.63,  aic = 5681.26
> arima(data,order=c(3,0,0))

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

Coefficients:
         ar1      ar2     ar3  intercept
      0.7136  -0.2003  0.0128     0.0057
s.e.  0.0224   0.0271  0.0224     0.0471

sigma^2 estimated as 0.9983:  log likelihood = -2836.47,  aic = 5682.93
```

Hence an AR (2) process is suitable because we started with AR (2). Now suppose you have data, so you can use auto.arima() routine to get the best AIC.

Formal Definition: Suppose we have a statistical model of some data. Let k bet the number of estimated parameters, let $\hat{L}$ be the maximum value of likelihood function for the model. Then AIC value is

$$AIC=2k-2\ln(\hat{L})$$

Caution : Sometimes say for AR(2) the AIC was 537 and for AR(3) its 536.6 but we would still choose AR(2) as for a small change in AIC we cannot add up more complexities in our model.

7.Backshift Operator(B):

Assume $X_1, X_2, \ldots$ be a time series then define Backshift Operator as :

$$BX_t=X_{(t-1)}$$

$$B^2X_t=BBX_t=BX_{(t-1)}=X_{(t-2)}$$

Utilizing Backshift Operator in MA2 process:

$$X_t=Z_t+0.2Z_{(t-1)}+0.4Z_{(t-2)}$$

$$X_t=Z_t+0.2BZ_t+0.4B^2Z_t$$

$$X_t=(1+0.2B+0.4B^2)\ Z_t$$

$$X_t=\psi(B)Z_t$$

Utilizing Backshift Operator in AR2 process:

$$X_t=0.2X_{(t-1)}+0.3X_{(t-2)}+Z_t$$

$$X_t=0.2BX_t+0.3B^2X_t+Z_t$$

$$(1-0.2B+0.3B^2)\ X_t = Z_t$$

$$Z_t=\tau(B)X_t$$

Similarly, you can vouch for AR(p) process as well as MA(q) process.

8.Duality:

Under Invertibility condition of MA(q) process, a MA(q) process could be converted/written as AR(∞) process and under Stationarity condition of AR(p) process , an AR(p) can be converted/written as MA(∞) process.

$\{X_t\}$ is a stochastic process, and $\{Z_t\}$ are innovations i.e. random disturbances/noise, the $\{X_t\}$ is called invertible if:

$$Z_t=\Sigma^{\infty}_{k=0}\,\pi_k\,X_{(t-k)}$$

Where $\Sigma^{\infty}_{k=0}\,\pi_k$ is convergent.

A MA(q) process is said to be invertible if the roots of the polynomial $\psi(B)$ described above lies outside the unit circle. The proof of this can be done by expanding $\psi(B)$ and using basic concept of series convergence from real analysis.

An AR(p) process is said to be stationary(weakly) if the roots of the polynomial $\tau(B)$ lies outside the unit circle.

9. ARMA Model: ARMA(p,q) model is defined by

X_t= Noise + Autoregressive Part + Moving Average Part

$$X_t= Z_t+ \phi_1X_{t-1}+ \phi_2X_{t-2}+ \dots + \phi_p X_{t-p} + \theta_1Z_{(t-1)} + \theta_2Z_{(t-2)} + \dots. + \theta_q Z_{(t-q)}$$

Using Backshift operator in both Autoregressive and moving average processes to denote it as

$$\theta(B)\ Z_t=\phi(B)\ X_t$$

$\theta(B)=1+\theta_1B+\theta_2B^2+\dots+\theta_pB^p$: Polynomial acting on noise

$\phi(B)=1-\phi_1B-\phi_2B^2-\dots-\phi_pB^p$: Polynomial acting on series values.

Goal: Convert ARMA process in AR(∞),MA(∞) process. How to do that? From above equation you can concur that.

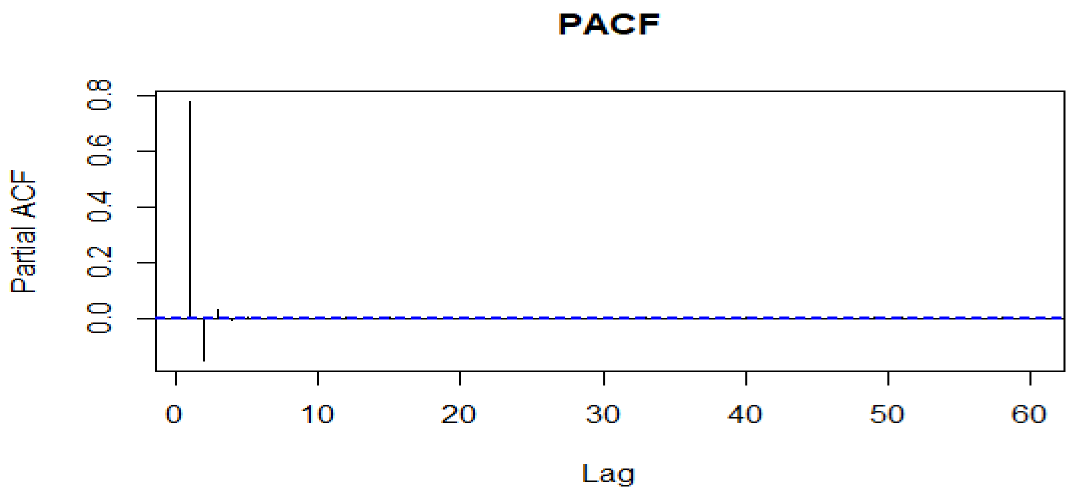
$$\theta(B)/\phi(B) * Z_t= X_t \quad AR(\infty)$$

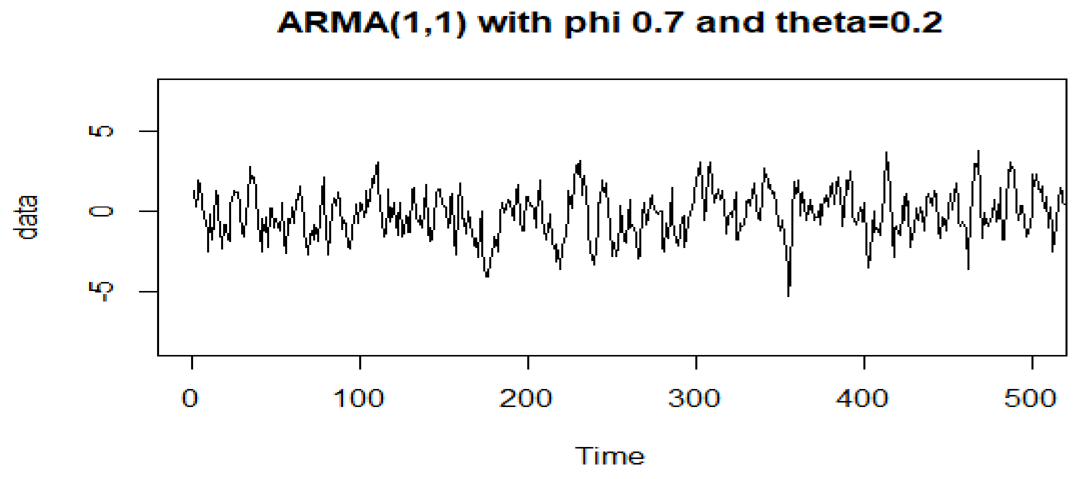
$$Z_t=\phi(B)/\theta(B)* X_t \ MA(\infty)$$

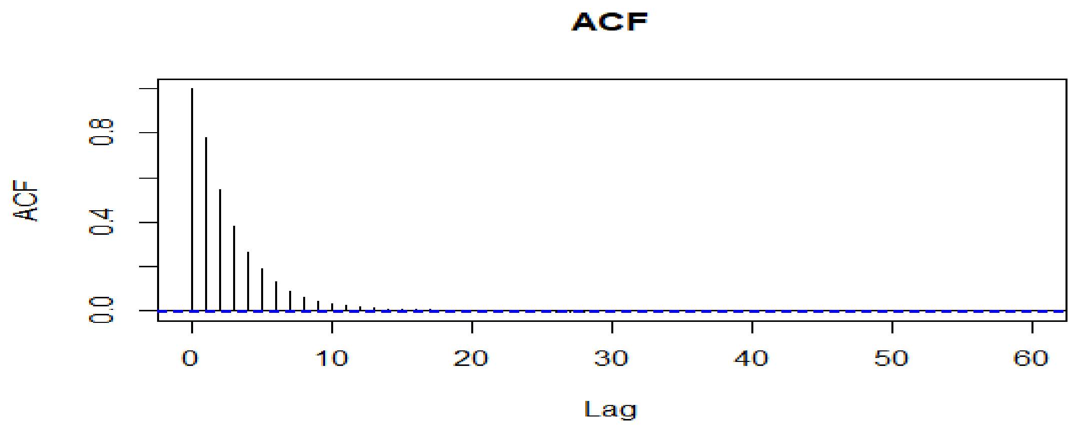
Compute those fractions and find the roots for which it converge to change process as talked in Duality.

Simulation of ARMA(1,1) Model :

```
#SIMULATION OF ARMA MODEL

# X_t=Z_t+0.7X_(t-1)+0.2Z_(t-1)

set.seed(500)
data=arima.sim(list(order=c(1,0,1),ar=0.7,ma=0.2),n=1000000)

plot(data,main="ARMA(1,1) with phi 0.7 and theta=0.2",xlim=c(0,500))
acf(data,main="ACF")
pacf(data,main='PACF')
```

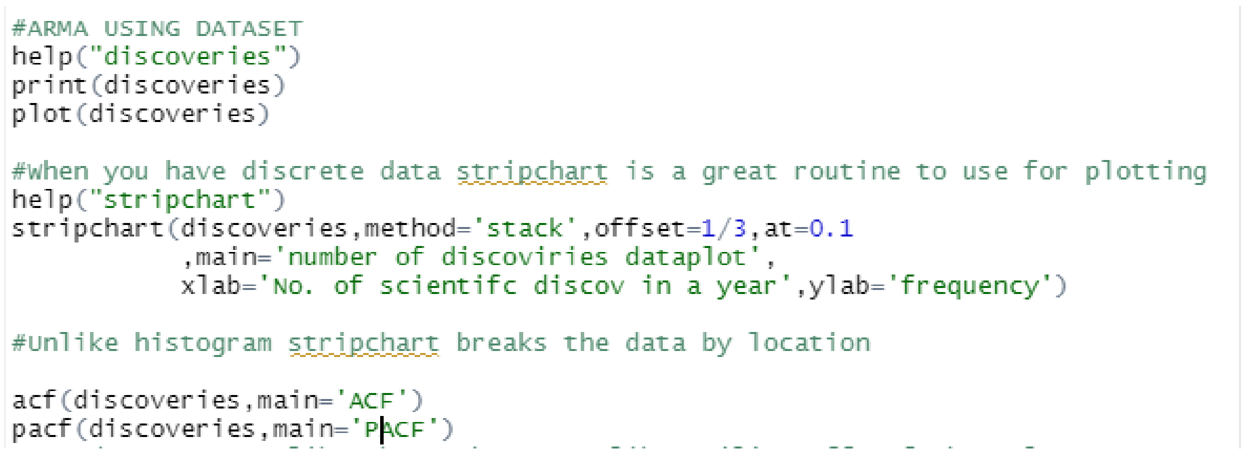




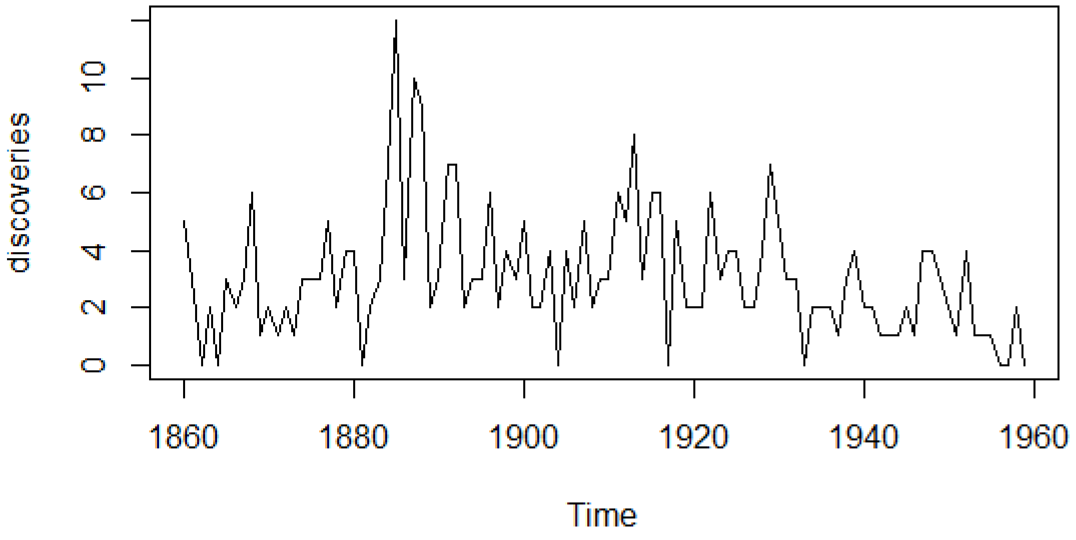


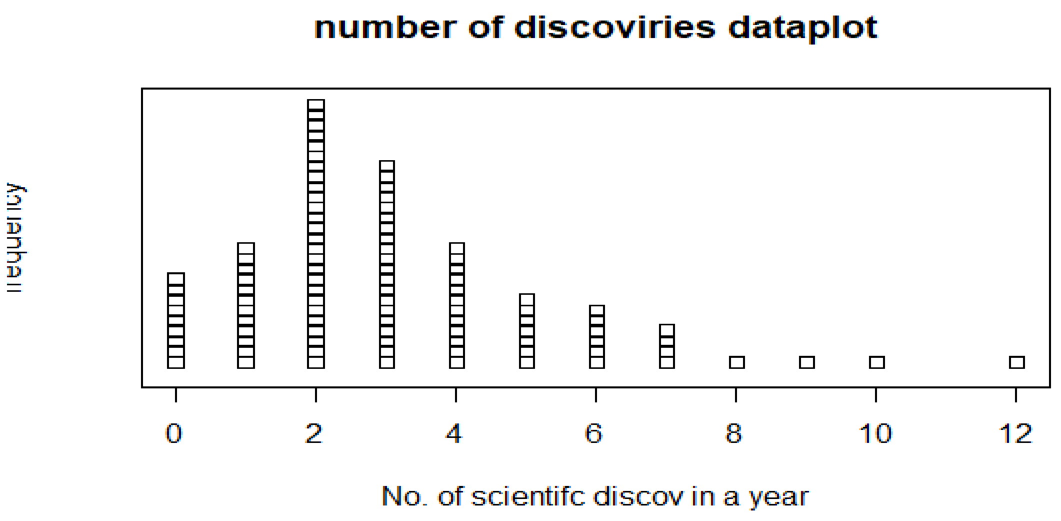
	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cuts after lag p	Tails off
PACF	Cuts after lag p	Tails off	Tails off

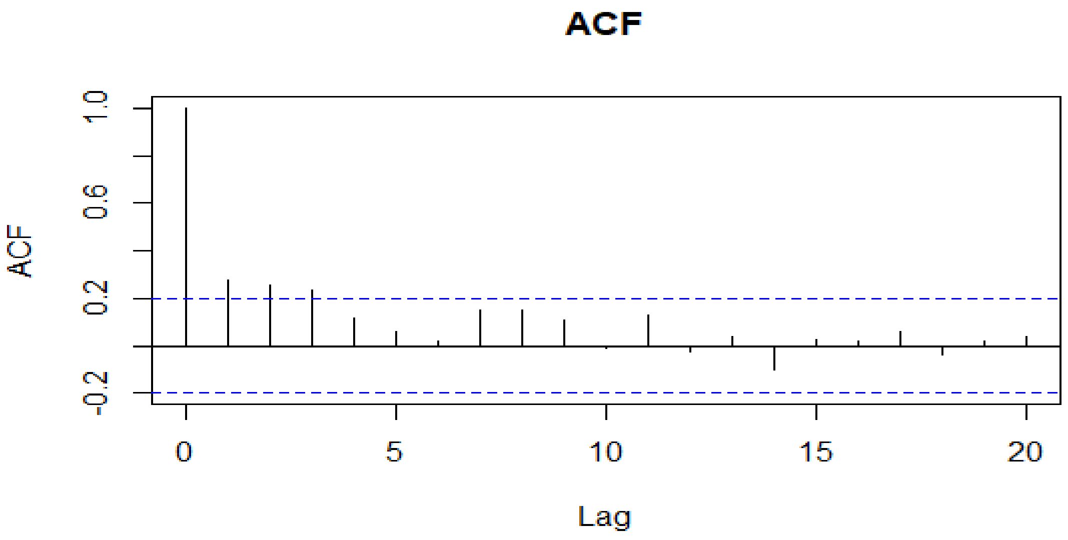
We believe that tails off means in the sense of exponential decay, and cuts off a lag p well you know abruptly cutting off. But if your system is noisy the difference between cutting off and tailing off may not be easy, so how do we concur if its AR, MA, ARMA or something else? Well look at the simulation of discoveries data set in R!

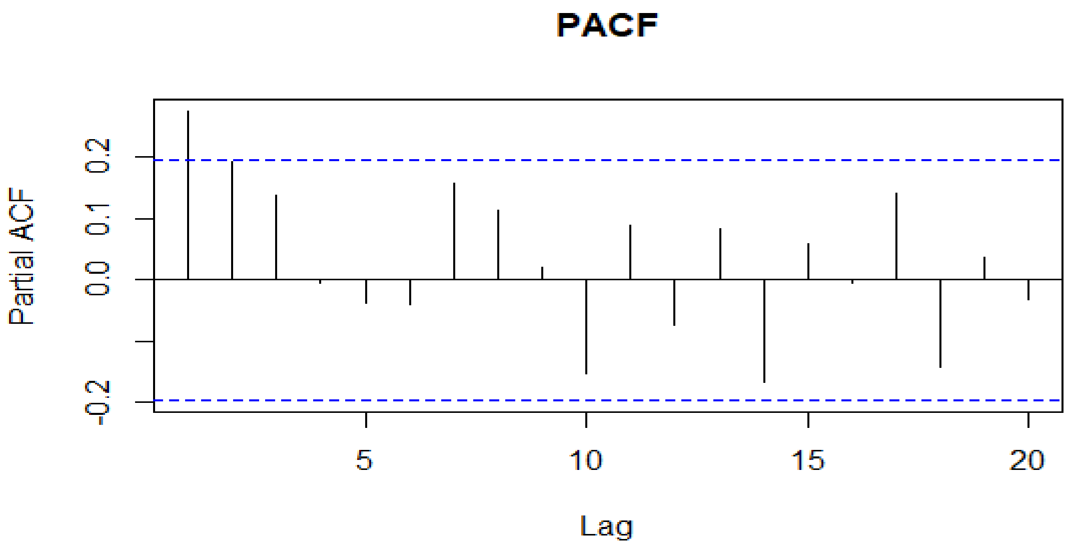


```
#ARMA USING DATASET
help("discoveries")
print(discoveries)
plot(discoveries)

#when you have discrete data stripchart is a great routine to use for plotting
help("stripchart")
stripchart(discoveries,method='stack',offset=1/3,at=0.1
           ,main='number of discoviries dataplot',
           xlab='No. of scientifc discov in a year',ylab='frequency')

#Unlike histogram stripchart breaks the data by location

acf(discoveries,main='ACF')
pacf(discoveries,main='PACF')
```




number of discoviries dataplot
frequency
0
2
4
6
8
10
12
No. of scientifc discov in a year


ACF
ACF
1.0
0.6
0.2
-0.2
0
5
10
15
20
Lag


PACF
Partial ACF
0.2
0.1
0.0
-0.2
5
10
15
20
Lag

ACF doesn't seems like abrupt but most like tailing off ACF, but also can be seen as cut off.

So we cannot say either ACF/PACF is cutting/tailing off then we take a range of values of p and q(in ARMA(p,q)) say p and q belongs to {0,1,2,3}, and check for the combination find the AIC value to find the best fit !! Either you can check by arima() routine each time or use auto.arima()routine to find best model.

Auto arima lives in forecast library so make sure to call that!

10.ARIMA: Auto regressive Integrated Moving Average Processes.

The difference operator: - Denoted by Δ and equal to 1-B where B is the backshift operator. For example in Random walk which is AR(1) process is given by

$$X_t = X_{(t-1)} + Z_t$$

$$X_t - X_{(t-1)} = Z_t$$

$$X_t - BX_t = Z_t$$

$$(1-B)\, X_t = Z_t$$

$$\Delta\, X_t = Z_t$$

Differencing can be Iterated i.e.

$$\Delta^2 X_t = \Delta(\Delta X_t) = \Delta(X_t - X_{(t-1)}) = (X_t - X_{(t-1)}) - (X_{(t-1)} - X_{(t-2)}) = X_t - 2X_{(t-1)} + X_{(t-2)}$$

Δ^k is the kth differencing operator.

Remember, ARMA(p,q) processes is the sum of both AR and MA processes

$$\theta(B)\, Z_t = \phi(B)\, X_t$$

If z is a complex number and roots of the complex polynomial $\theta(z)$ and $\phi(z)$ lies outside the unit circle then ARMA process would be stationary and invertible. But real life stock data are non-stationary and they might have systematic change in trend which we have to remove for which we need the differencing operator.

A process X_t is ARIMA of order (p,d,q) if $Y_t = \Delta^d X_t = (1-B)^d X_t$ is an ARMA(p,q) process.

11.Unit Root Test:

We have seen that it is difficult to tell whether a time series should be modelled as stationary or non-stationary. To help decide between these two possibilities it will be helpful to use hypothesis testing.

What do you mean by unit root? Remember that in AR(p) process the root of the polynomial should lie outside the unit circle for it to be stationary and in MA(q) the root of its polynomial should also lie outside unit circle for it to be invertible (and also we have seen non invertible MA processes are not stationary) so for an ARMA(p,q) model if there is a unit root a root whose absolute value is 1 then ARMA process is non stationary and behaves much like a random walk.

To curb this, we use Augmented Dickey fuller test as one of the tests which determines the occurrence of a unit root or not determined by the p value of Null hypothesis and Alternate hypothesis, which we will be using in our data set.

The null hypothesis is : Non stationary and alternate hypothesis is : Stationary, we set the p value level to 5% i.e. 0.05, a p-value of > 0.0.5 says not to reject null hypothesis meaning non stationarity and <0.05 says stationarity.

12. Ljung Box Test:

The null hypothesis of the Box Ljung Test, H0, is that our model does not show lack of fit (or in simple terms—the model is just fine). The alternate hypothesis, Ha, is just that the model does show a lack of fit.

If the p value is greater than 0.05 then the residuals are independent which we want for the model to be correct. If you simulate a white noise time series using the code below and use the same test for it then the p value will be greater than 0.05.

Note that it is applied to the residuals of a fitted ARIMA model, not the original series, and in such applications the hypothesis actually being tested is that the residuals from the ARIMA model have no autocorrelation.

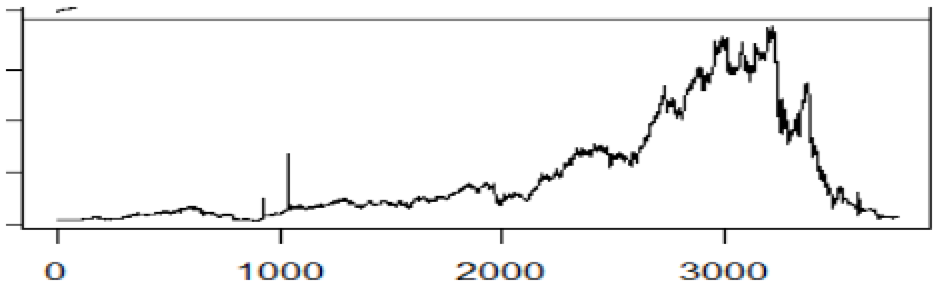
ARIMA ON YES BANK's DATA

Import the data set into R, delete the columns(variables) Open, High, Low, Close and volume from the dataset as we just need ADJ close for analysis

Plot the data set to understand the behavior of the time series.

```
library(readr)
data <- read_csv("YESBANK.csv", col_types = cols(Open = col_skip(),
                                  High = col_skip(), Low = col_skip(),
                               Close = col_skip(), Volume = col_skip()))

plot(ts(data))
```

This is how the plot looks:



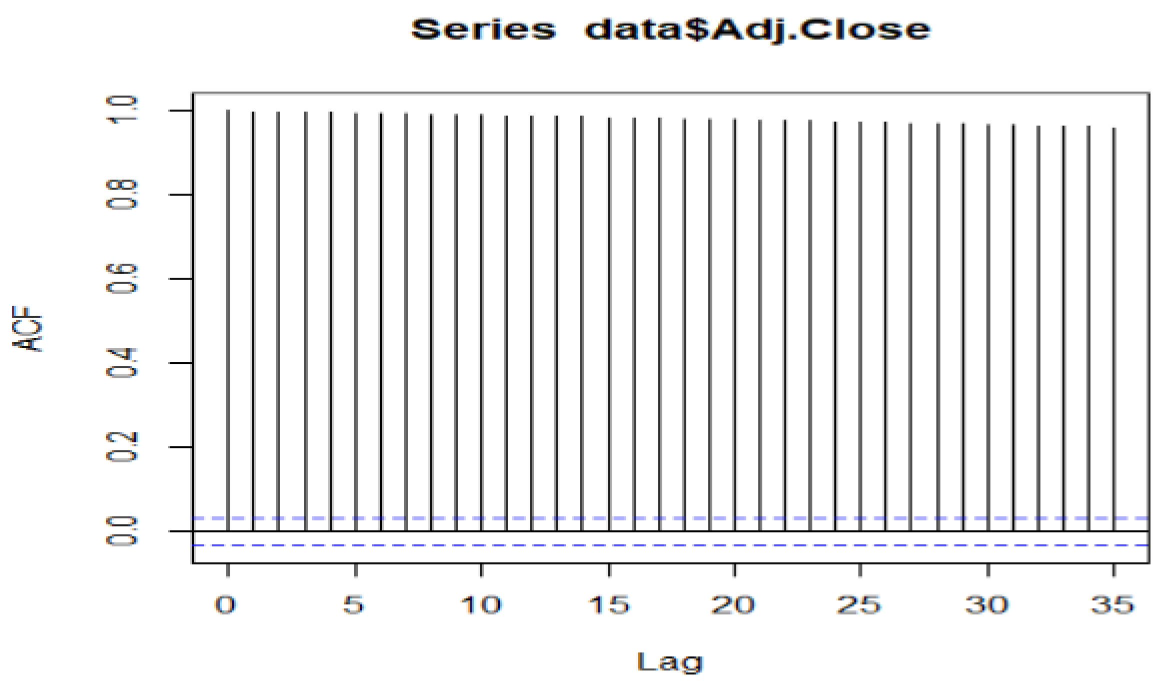
Now do Augmented Dickey fuller test:

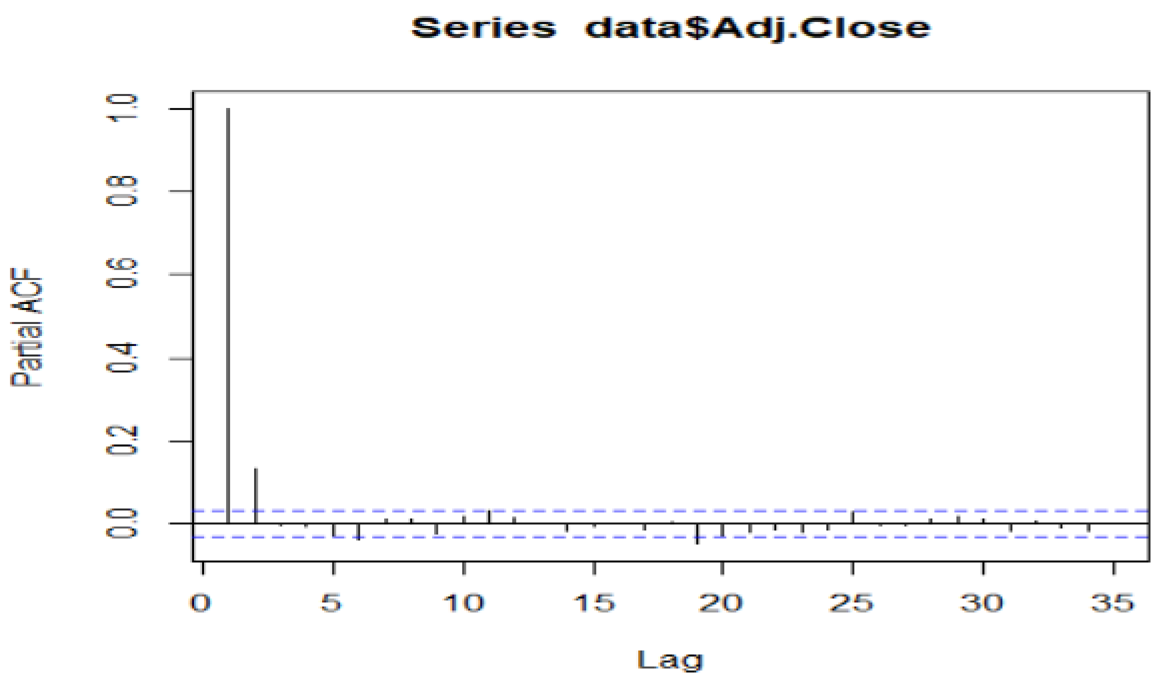
```
adf.test(data$Adj.Close)

        Augmented Dickey-Fuller Test

data:  data$Adj.Close
Dickey-Fuller = -0.67419, Lag order = 15, p-value = 0.9729
alternative hypothesis: stationary
```

ADF test says it's a non-stationary data and of course we can see that with trend in the data, now plot ACF and PACF to know the Autoregressive and Moving Average process order

Autocorrelation refers to how correlated a time series is with its past values. As we know in AR models, the ACF will dampen exponentially. The ACF is the plot used to see the correlation between the points, up to and including the lag unit. We can see that the autocorrelations are significant for a large number of lags, but perhaps the autocorrelations at posterior lags are merely due to the propagation of the autocorrelation at the first lags. For identifying the (p) order of the AR model we use the PACF plot. For MA models we will use ACF plot to identify the (q) order and the PACF will dampen exponentially. If we look the PACF plot, we can note that the it has a significant spike only at first lags, meaning that all the higher-order autocorrelations are effectively explained by the first lag autocorrelation. As we are using

AUTO-ARIMA function that gives us the better approach to the dataset, we will not deep the analysis on finding model parameters





Now let's see what model does Auto Arima gives us.

```
modelfit <- auto.arima(data$Adj.Close, lambda = "auto")
print(modelfit)
```

```
Series: data$Adj.Close
ARIMA(5,2,0)
Box Cox transformation: lambda= -0.2280452

Coefficients:
          ar1      ar2      ar3      ar4      ar5
      -1.1174  -0.9656  -0.6902  -0.3741  -0.1408
s.e.   0.0161   0.0235   0.0260   0.0235   0.0161

sigma^2 estimated as 0.0009131:  log likelihood=7846.89
AIC=-15681.78   AICc=-15681.75   BIC=-15644.37
```

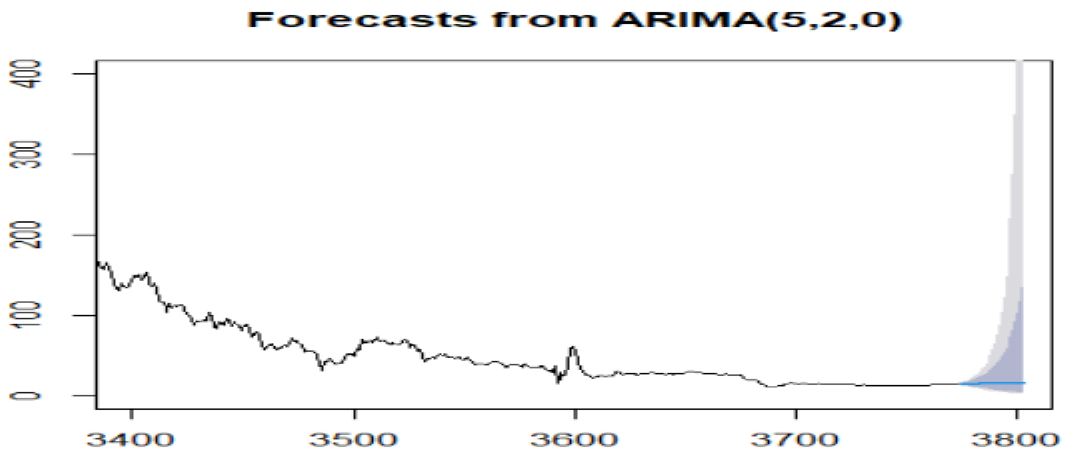
Auto Arima is giving us the best fit model for our data as ARIMA(5,2,0)

Now do Ljung's Box Test to check if ARIMA (5,2,0) does fit your data or not.

```
Box.test(modelfit$residuals, type="Ljung-Box")

        Box-Ljung test

data:  modelfit$residuals
X-squared = 1.9487, df = 1, p-value = 0.1627
```

So the model is a good fit to the data!

As we can see, the AUTO-ARIMA selects the best model parameters, giving us a very good estimation. We need to remind that this is an auto regressive model, so we are going to have very good past predictions. Now with the model fitted we can proceed to forecast our daily close price values to the future. We focus on forecasting the close stock price for the next 30 days or an average month.



```
price_forecast <- forecast(modelfit, h=30)
plot(price_forecast,ylim=c(0,400),xlim=c(3400,3800))
```

The blue line shows the mean of our prediction. With the blue line we can see the darker and lighter blue areas representing the 80 % and 95% confidence interval respectively.

```
print(price_forecast$mean)
```

```
> print(price_forecast$mean)
Time Series:
Start = 3774
End = 3803
Frequency = 1
 [1] 14.55661 14.51041 14.53643 14.58897 14.66397 14.70858 14.70654 14.75081
 [9] 14.79788 14.83813 14.87357 14.90873 14.94646 14.98763 15.02596 15.06365
[17] 15.10201 15.14107 15.18025 15.21932 15.25828 15.29756 15.33705 15.37663
[25] 15.41627 15.45604 15.49596 15.53602 15.57620 15.61650
>
```

Finalizing our AUTO-ARIMA model we do a quick test and train set approach dividing the close price data. We select our train set as the 70 percent of our dataset. The test set the 30 remaining percent of the dataset.

```
#Dividing in training and test set

N = length(data$Adj.Close)
n = 0.7*N
train = data[1:n, ]
test  = data[(n+1):N,  ]
trainarimafit <- auto.arima(train$Adj.Close, lambda = "auto")
predlen=length(test)
trainarimafit1 <- forecast(trainarimafit, h=predlen)
```

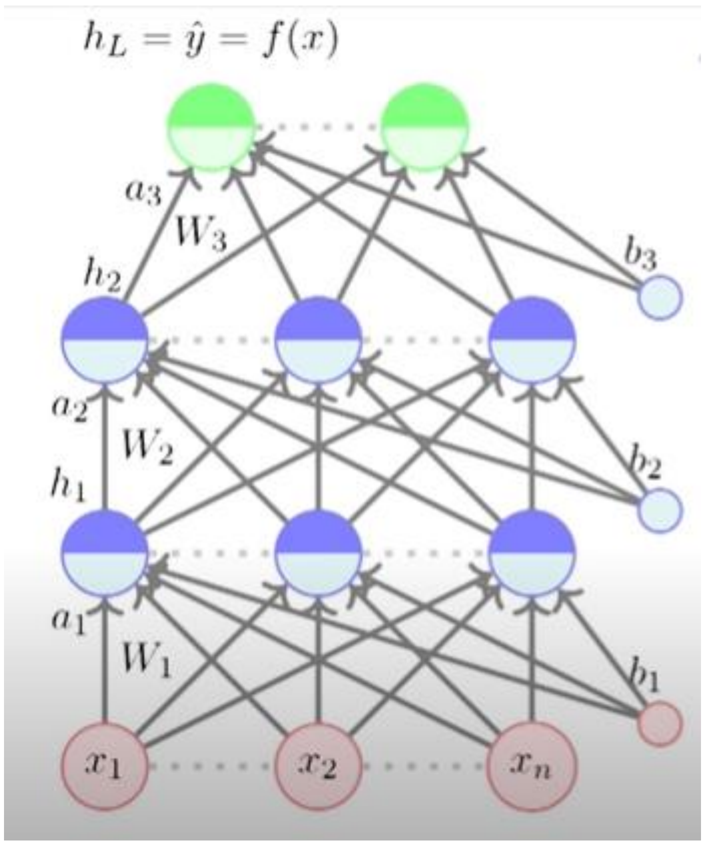
```
accuracy(trainarimafit1)
```

```
> accuracy(trainarimafit1)
                    ME     RMSE      MAE        MPE     MAPE     MASE
Training set 0.1238624 2.879412 1.075322 0.04853502 2.466509 1.029624
                   ACF1
Training set -0.1013896
>
```

```
        ME     RMSE      MAE        MPE     MAPE      MASE
-0.1343804 6.267007 3.603912 -0.3635569 2.639844 0.9991601
      ACF1
).01473939
```

FEED FORWARD NEURAL NETWORKS:

1. The input of the network is a n-dimensional vector, [x_1, x_2, …, x_n] $\in R^n$
2. The network contains L-1 hidden layers (2 in the picture) having n neurons each.
3. Finally, one output layer containing k neurons
4. Each neuron in the hidden layer and the output layer can be split into 2 parts called pre-activation and activation neuron, the work of pre-activation neuron is to aggregate the information and the activation neuron is used for non-linearity (a_i's and h_i's are vectors $\in R^n$)
5. The input layer is called 0th layer (h_0) and output layer is called Lth layer(h_L)
6. The weights going to ith layer is denoted $W_i \in R^{n*n}$ and the bias at the ith layer $\in R^n$ where i is between 0 to L
7. While in the last layer $W_L \in R^{n*k}$ (because there are k neurons in output layer) and the bias $b_L \in R^k$.



8. The pre-activation at layer i is given by:

$$A_i(x) = b_i + W_i h_{(i-1)}\, x$$

Computing the first layer

$$A_1(x) = b_1 + W_1 h_0\, x$$

$$\begin{bmatrix} a11 \\ a12 \\ a13 \end{bmatrix} = \begin{bmatrix} b11 \\ b12 \\ b13 \end{bmatrix} + \begin{bmatrix} W111 & W112 & W113 \\ W121 & W122 & W123 \\ W131 & W132 & W133 \end{bmatrix} \begin{bmatrix} h01 = x1 \\ h02 = x2 \\ h03 = x3 \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma W11i\ xi + b11 \\ \Sigma W12i\ xi + b12 \\ \Sigma W13i\ xi + b13 \end{bmatrix}$$

So, its just weighted aggregation of all the input and that's what pre activation function does. Every entry in a_1 is just weighted aggregate of all the inputs.

The activation at layer I denoted as h_i is given by

$$h_i = g(a_i)$$

here g act on the vector element wise is sigmoidal function (logistic).

The output layer of 1 section or hidden layer of this feed forwards neural network is input layer of next layer of hidden layer and so on, so the transition of pre-activation to activation neuron is done by sigmoidal function and is an element wise operation its acting on each element of vector a_i

g is called activation function it can be linear, tanh, logistic.

We use sigmoidal function in the hidden layer because it gives the output between 0 to 1 but in the output layer, we do not need the output to be between 0 and 1 so we assume an output function say O at the last layer.

The activation at last layer is given by

$$f(x) = h_l(x) = O(a_L(x))$$

O is for example SoftMax, linear etc.

To simplify notations lets refer $a_i(x)$ as a_i and $h_i(x)$ as h_i
The output layer has different dimension as opposed to everywhere else as it has k classes and everywhere its n.

Now, I need to put this information into the paradigm of supervised machines learning that are
1. Data
2. Model
3. Parameters
4. Learning Algorithm
5. Objective function

1. Data: $\{x_i, y_i\}$ from i=1 to n, so lets see what exactly is our model assumption here right? So the question is given some data we don't really know the true relation between y and x (price of a stock and time) we make an assumption that y is related to x by using a function f and it has some parameters and we would like to learn those parameters (just like the parameters/coefficient in AR or MA models). So what is the function here? What is your Model?

2. Model: Multiplication with W gives a linear transformation and g a sigmoidal function gives a non-linear transformation, that is what a function does actually transformations., also do consider the dimension of each of the elements here.

Model:

$$\hat{y}_i = f(x_i) = O(W^3 g(W^2 \underbrace{g(\underbrace{\underbrace{\overset{\mathbb{R}^{n\times n}}{W^1}\overset{\mathbb{R}^n}{x}}_{\mathbb{R}^n} + \underbrace{b_1}_{\mathbb{R}^n}}_{\mathbb{R}^n})}_{\mathbb{R}^n} + b_2) + b_3)$$

3. As you can see by the model its continuous transformation of input vector x to output vector linear-non-linear-linear-non-linear, so this model has a great complexity and with great complexity comes great power (not responsibility)

4. The Parameters are $W_1, W_2, \dots, W_L$ and $b_1, b_2, \dots, b_L$

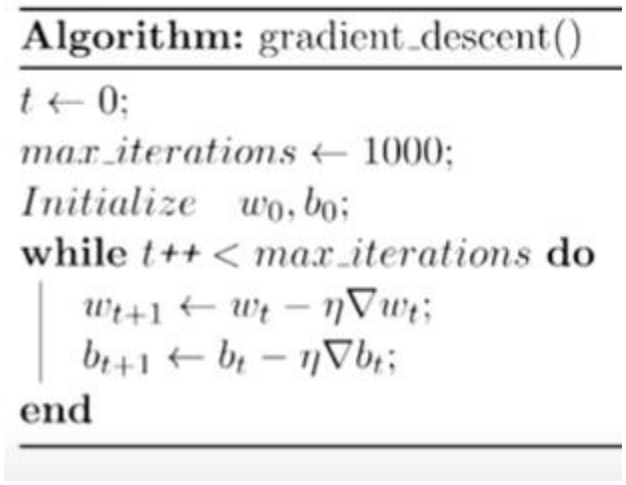
5. Algorithm used is Gradient Descent with Back propagation.

6. The objective/Loss function would me

$$\text{Min } (1/N * \Sigma^{N}_{i=1} \Sigma^{N}_{j=1} (ybar_{ij} - y_{ij}))$$

Parameters Finding Intuition:

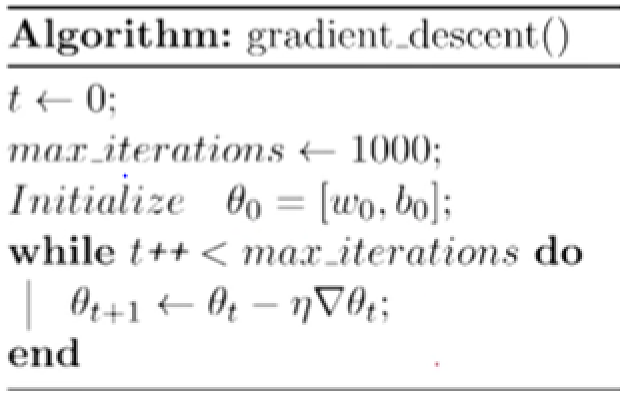
Recall your gradient Descent Algorithm



```
Algorithm: gradient_descent()
$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
$Initialize \quad w_0, b_0$;
while $t{++} < max\_iterations$ do
    $w_{t+1} \leftarrow w_t - \eta \nabla w_t$;
    $b_{t+1} \leftarrow b_t - \eta \nabla b_t$;
end
```

This is how it looks, I had initialized w_0 and b_o those two parameters, and then what I was doing iteratively is that I was moving at the direction opposite to the gradient at that point.

We can write it more compactly by using vectors $\theta_0 = [\ w_0 \ \ b_o]$



```
Algorithm: gradient_descent()
$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
$Initialize \quad \theta_0 = [w_0, b_0]$;
while $t{++} < max\_iterations$ do
    $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$;
end
```

Where $\nabla\theta_t = [(dL(theta))/dw, (dL(theta))/db]^T$

That is the collection of all the partial derivatives with all the parameters.

Now analogous to this you might have gotten the idea what I will be trying to do here now, now instead of $\theta_0 = [\ w_0\ \ b_o]$ now θ_0 is $W_1, W_2, \ldots, W_L$ and $b_1, b_2, \ldots, b_L$.

So, now when we find $\nabla\theta_t$ we would be differentiating the loss function with all the parameters W's and b's. but the loss function is a function of θ and W is a matrix, so how can we find the gradient of loss function with W_i's, see one of the W_i resides in θ , so it will be derivative with respect to every element of the matrix.

```
Algorithm: gradient_descent()
$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
Initialize  $\theta_0 = [W_1^0, ..., W_L^0, b_1^0, ..., b_L^0]$;
while $t{+}{+} < max\_iterations$ do
    $\theta_{t+1} \leftarrow \theta_t - \eta\nabla\theta_t$;
end
```

Except now our ∇θ looks more nasty

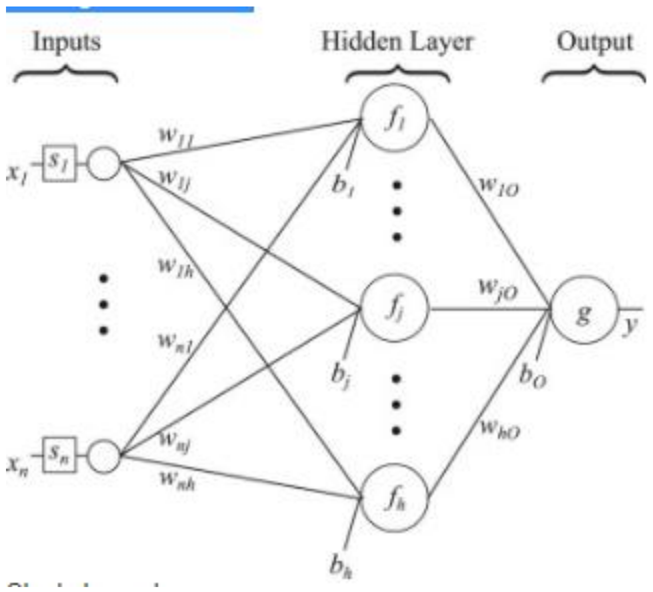
$$\begin{bmatrix}
\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{11n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{211}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{21n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L1}} \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{121}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{12n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{221}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{22n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,21}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{12}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L2}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{1n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{1nn}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2nn}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{1n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L,n}}
\end{bmatrix}$$

So assume that one oracle gave you all the quantities and you know the loss function then by gradient descent you could find all the parameters.

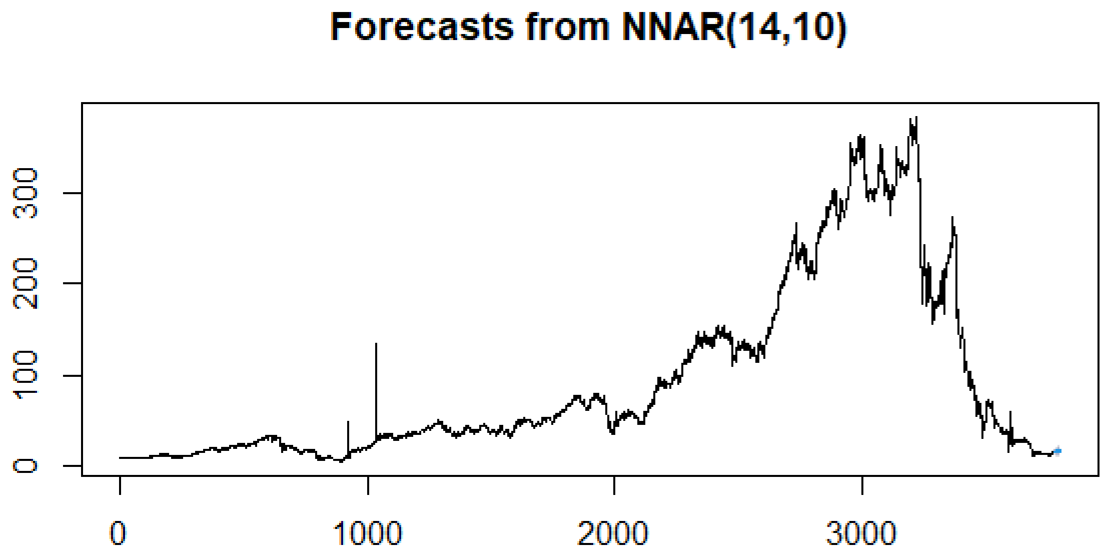
Simulation in R:

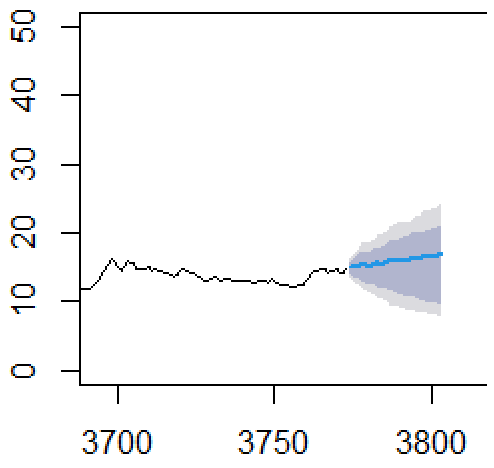
Researching deeply in new machine learning models we have reached some new neural network function in the forecast package called **nnetar**. A single hidden layer neural network is the simplest neural networks form. In this single hidden layer form, there is only one layer of input nodes that send weighted inputs to a subsequent layer of receiving nodes. This nnetar function in the forecast package fits a single hidden layer neural network model to a timeseries. The function model approach is to use lagged values of the time series as input data, reaching to a non-linear autoregressive model.

So we are using single layer FFNN



```
hn <- 10

lambda <- BoxCox.lambda(data$Adj.Close)
dnn_pred <- nnetar(data$Adj.Close, size= hn, lambda = lambda)
dnn_forecast <- forecast(dnn_pred, h= 30, PI = TRUE)
plot(dnn_forecast)
head(dnn_forecast)
accuracy(dnn_forecast)
```





```
> accuracy(dnn_forecast)
                    ME     RMSE      MAE         MPE     MAPE     MASE       ACF1
Training set 0.1072568 4.096656 1.787756 -0.07586246 2.281327 0.986734 0.01240648
```

CONCLUSION: According to the results of both models that are ARIMA and Feed forward neural network, the FFNN performed better than ARIMA, the errors were less and the confidence intervals were also sharp. FFNN always takes up the output of one hidden layer as an input of another hidden layer, so assuming that the model has 2 hidden layers the prediction of one price of the stock would be the input to other layer and predicting on a prediction is a mishap, this is why always remember that in forecasting you should always use single layered feed forward neural network. The ARIMA model to be honest according to me felt bogus and way out of the chart, first thing that ticked me off was that the fitted model came out to be ARIMA(5,2,0) a 0 Moving Average process ? Well, this means if I difference the data two times then I would get AR(5) process that means the data or stock prices were independent of previous errors/announcements/innovations/noise? And only depended upon the previous 5 stock prices only! Maybe I did some errors in the code or maybe as I have written Caution in Akaike Information Criteria that some times we can forget small change in AIC and aim for model with less complexity, maybe Auto Arima () routine has baffled me and there must have existed some ARIMA(4,2,1) with a little worse AIC but not too worse so that I could have picked it as a model. Also as

I forecasted Arima it showed an increasing trend in prices, this must have been because from 2005 to 2018 the stock prices of Yes Bank had a increasing trend and suddenly due to money laundering case it went down and we all can agree this bank is in the grave now so as 75% of the training set and 25% of the test set comes from whole data and most of the data has increasing trend with an sudden dip ARIMA thought it would go up obviously but it would not this bank is long gone, government has stepped in and taken in-charge with the help of RBI but I don't think it would rise as high as around 400 that was before 2018. One last thing, if you see the plot of volume purchased with time you could see a great and sudden spike in previous days this is because at one time when prices was 400 one could take 1 stock in 400rs and now as it is down to 10 you could take 40 stocks for 400 bucks, now it has become a penny stock so if I were to be a trader/investment banker, I would prefer this penny stock, would buy high volumes and would have made small profits from 40 stocks rather than a small profit of a stock of 400 bucks.