1. Use the below given data set

Data Set

2. Perform the below given activities:

a. Predict the no of comments in next H hrs

Note:-

1. Use LASSO, Elastic Net and Ridge and other regression techniques that are covered in the

Module

**library**(tidyverse) **library**(caret) **library**(glmnet)

# Load the data and remove NAs data("PimaIndiansDiabetes2", package = "mlbench") PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2) # Inspect the data sample\_n(PimaIndiansDiabetes2, 3) # Split the data into training and test set set.seed(123) training.samples <- PimaIndiansDiabetes2$diabetes %>% createDataPartition(p = 0.8, list = FALSE) train.data <- PimaIndiansDiabetes2[training.samples, ] test.data <- PimaIndiansDiabetes2[-training.samples, ]

# Dumy code categorical predictor variables x <- model.matrix(diabetes~., train.data)[,-1] # Convert the outcome (class) to a numerical variable y <- ifelse(train.data$diabetes == "pos", 1, 0)

glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)

**library**(glmnet) # Find the best lambda using cross-validation set.seed(123) cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial") # Fit the final model on the training data model <- glmnet(x, y, alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min) # Display regression coefficients coef(model) # Make predictions on the test data x.test <- model.matrix(diabetes ~., test.data)[,-1] probabilities <- model %>% predict(newx = x.test) predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg") # Model accuracy observed.classes <- test.data$diabetes mean(predicted.classes == observed.classes)

**library**(glmnet) set.seed(123) cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial") plot(cv.lasso)

cv.lasso$lambda.min

cv.lasso$lambda.1se

coef(cv.lasso, cv.lasso$lambda.min)

coef(cv.lasso, cv.lasso$lambda.1se)

# Final model with lambda.min lasso.model <- glmnet(x, y, alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min) # Make prediction on test data x.test <- model.matrix(diabetes ~., test.data)[,-1] probabilities <- lasso.model %>% predict(newx = x.test) predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg") # Model accuracy observed.classes <- test.data$diabetes mean(predicted.classes == observed.classes)

# Final model with lambda.1se lasso.model <- glmnet(x, y, alpha = 1, family = "binomial", lambda = cv.lasso$lambda.1se) # Make prediction on test data x.test <- model.matrix(diabetes ~., test.data)[,-1] probabilities <- lasso.model %>% predict(newx = x.test) predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg") # Model accuracy rate observed.classes <- test.data$diabetes mean(predicted.classes == observed.classes)

# Fit the model full.model <- glm(diabetes ~., data = train.data, family = binomial) # Make predictions probabilities <- full.model %>% predict(test.data, type = "response") predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg") # Model accuracy observed.classes <- test.data$diabetes mean(predicted.classes == observed.classes)

2. Report the training accuracy and test accuracy

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | # load the libraries  library(caret)  library(klaR)  # load the iris dataset  data(iris)  # define an 80%/20% train/test split of the dataset  split=0.80  trainIndex <- createDataPartition(iris$Species, p=split, list=FALSE)  data\_train <- iris[ trainIndex,]  data\_test <- iris[-trainIndex,]  # train a naive bayes model  model <- NaiveBayes(Species~., data=data\_train)  # make predictions  x\_test <- data\_test[,1:4]  y\_test <- data\_test[,5]  predictions <- predict(model, x\_test)  # summarize results  confusionMatrix(predictions$class, y\_test)     |  |  |  |  | | --- | --- | --- | --- | |  | # load the library  library(caret)  # load the iris dataset  data(iris)  # define training control  train\_control <- trainControl(method="boot", number=100)  # train the model  model <- train(Species~., data=iris, trControl=train\_control, method="nb")  # summarize results  print(model)  library(caret)  # load the iris dataset  data(iris)  # define training control  train\_control <- trainControl(method="cv", number=10)  # fix the parameters of the algorithm  grid <- expand.grid(.fL=c(0), .usekernel=c(FALSE))  # train the model  model <- train(Species~., data=iris, trControl=train\_control, method="nb", tuneGrid=grid)  # summarize results  print(model)  library(caret)  # load the iris dataset  data(iris)  # define training control  train\_control <- trainControl(method="repeatedcv", number=10, repeats=3)  # train the model  model <- train(Species~., data=iris, trControl=train\_control, method="nb")  # summarize results  print(model)   |  |  | | --- | --- | | 4  5  6  7  8  9  10 | # load the library  library(caret)  # load the iris dataset  data(iris)  # define training control  train\_control <- trainControl(method="LOOCV")  # train the model  model <- train(Species~., data=iris, trControl=train\_control, method="nb")  # summarize results  print(model) | | |

3. compare with linear models and report the accuracy

Linear Regression

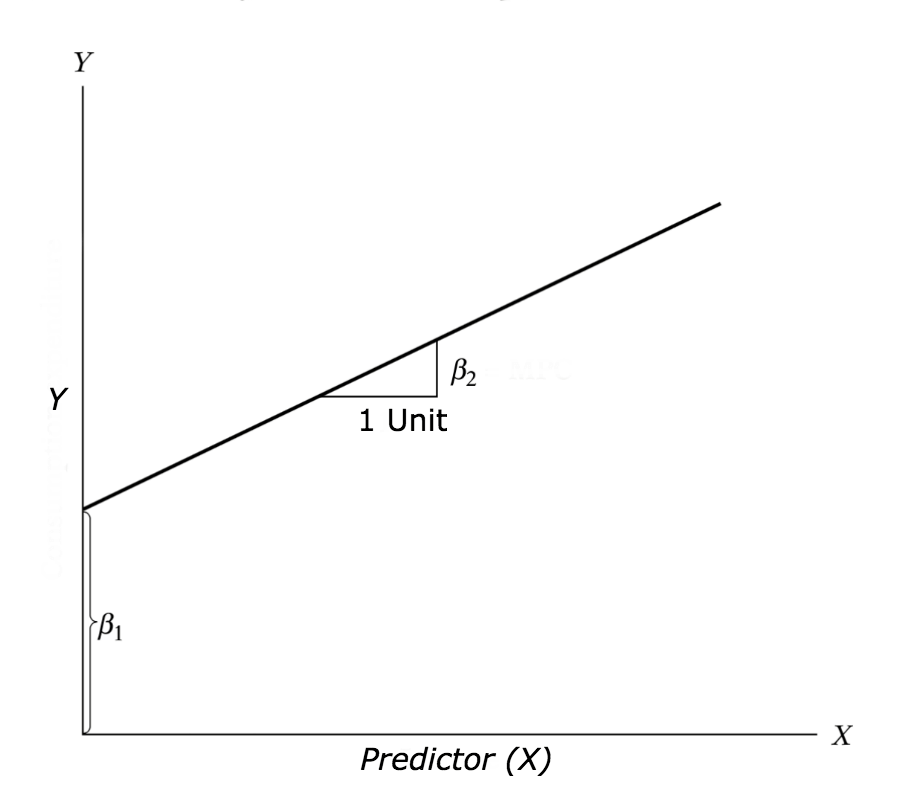
Linear regression is used to predict the value of an outcome variable *Y* based on one or more input predictor variables *X*. The aim is to establish a linear relationship (a mathematical formula) between the predictor variable(s) and the response variable, so that, we can use this formula to estimate the value of the response *Y*, when only the predictors (*Xs*) values are known.

Introduction

The aim of linear regression is to model a continuous variable *Y* as a mathematical function of one or more *X* variable(s), so that we can use this regression model to predict the *Y* when only the *X* is known. This mathematical equation can be generalized as follows:

*Y* = *β*1 + *β*2*X* + *ϵ*

where, *β*1 is the intercept and *β*2 is the slope. Collectively, they are called *regression coefficients*. *ϵ* is the error term, the part of *Y* the regression model is unable to explain.



Example Problem

For this analysis, we will use the *cars* dataset that comes with R by default. cars is a standard built-in dataset, that makes it convenient to demonstrate linear regression in a simple and easy to understand fashion. You can access this dataset simply by typing in cars in your R console. You will find that it consists of 50 observations(rows) and 2 variables (columns) – dist and speed. Lets print out the first six observations here..

**head**(cars) *# display the first 6 observations*

*#> speed dist*

*#> 1 4 2*

*#> 2 4 10*

*#> 3 7 4*

*#> 4 7 22*

*#> 5 8 16*

*#> 6 9 10*

Before we begin building the regression model, it is a good practice to analyze and understand the variables. The graphical analysis and correlation study below will help with this.

Graphical Analysis

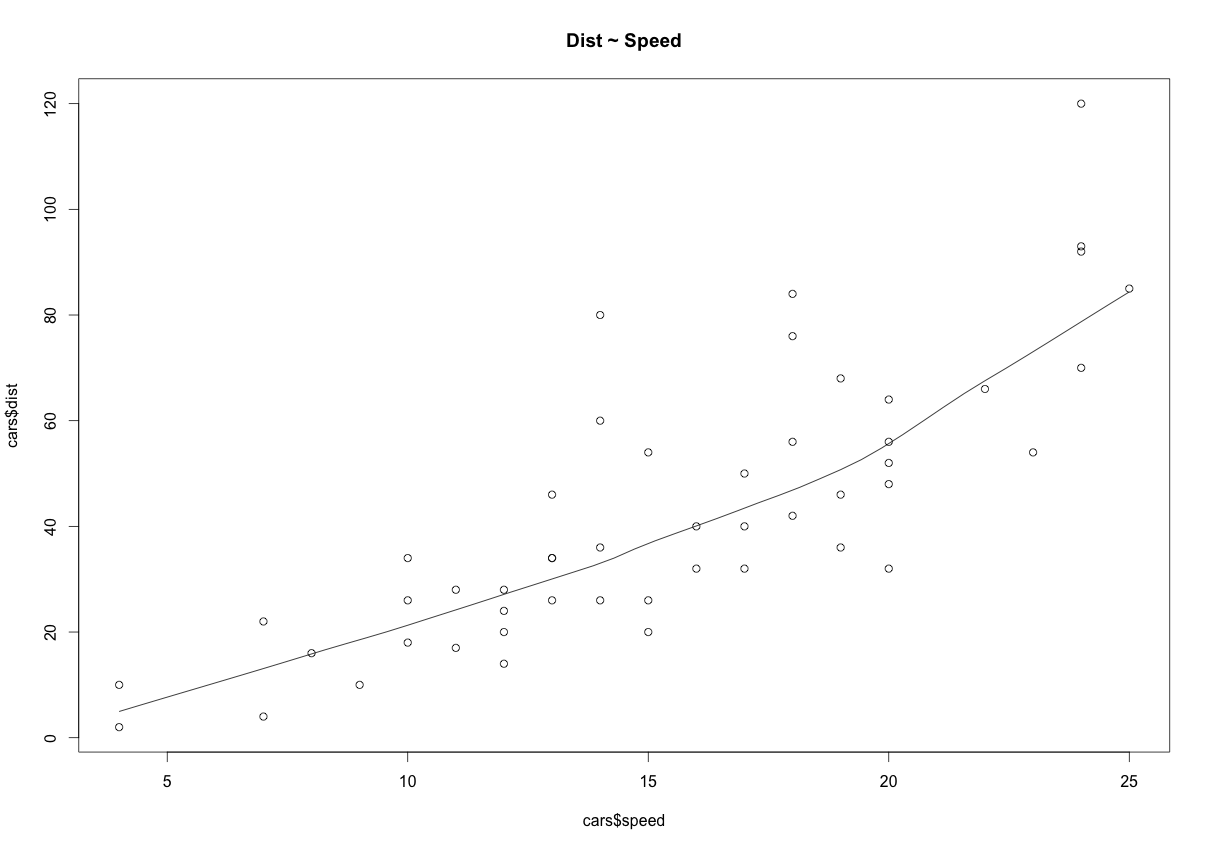
The aim of this exercise is to build a simple regression model that we can use to predict Distance (dist) by establishing a statistically significant linear relationship with Speed (speed). But before jumping in to the syntax, lets try to understand these variables graphically. Typically, for each of the independent variables (predictors), the following plots are drawn to visualize the following behavior:

1. **Scatter plot**: Visualize the linear relationship between the predictor and response
2. **Box plot**: To spot any outlier observations in the variable. Having outliers in your predictor can drastically affect the predictions as they can easily affect the direction/slope of the line of best fit.
3. **Density plot**: To see the distribution of the predictor variable. Ideally, a close to normal distribution (a bell shaped curve), without being skewed to the left or right is preferred. Let us see how to make each one of them.

Scatter Plot

Scatter plots can help visualize any linear relationships between the dependent (response) variable and independent (predictor) variables. Ideally, if you are having multiple predictor variables, a scatter plot is drawn for each one of them against the response, along with the line of best as seen below.

**scatter.smooth**(x=cars$speed, y=cars$dist, main="Dist ~ Speed") *# scatterplot*



The scatter plot along with the smoothing line above suggests a linearly increasing relationship between the ‘dist’ and ‘speed’ variables. This is a good thing, because, one of the underlying assumptions in linear regression is that the relationship between the response and predictor variables is linear and additive.

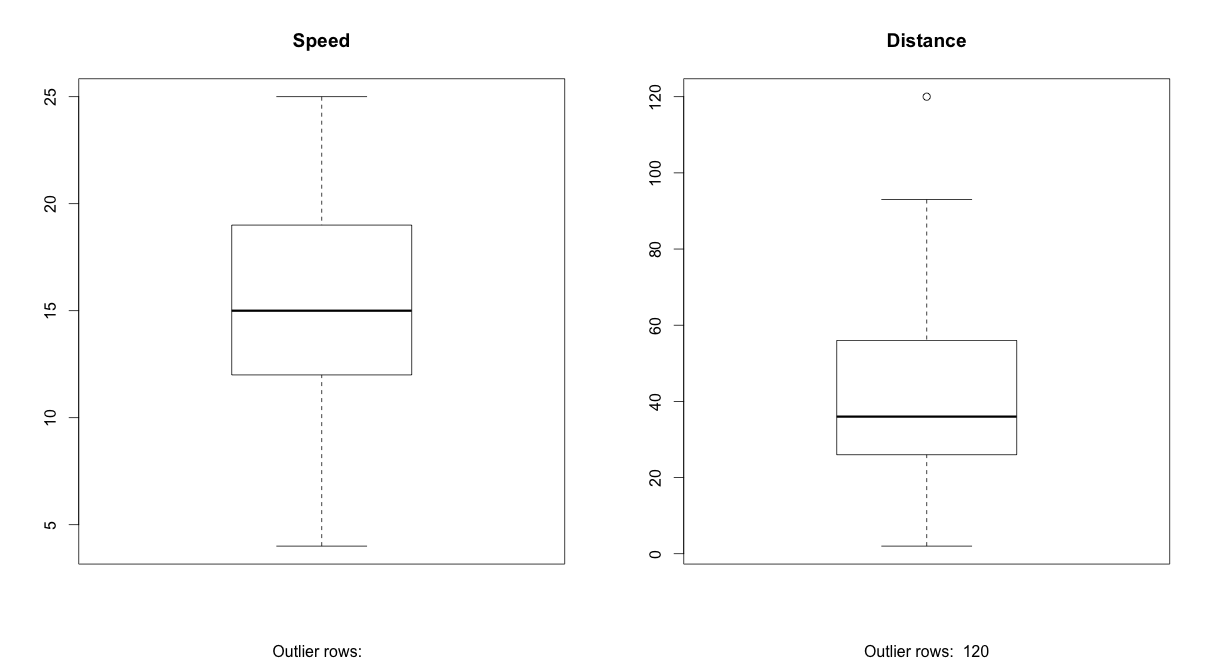
BoxPlot – Check for outliers

Generally, any datapoint that lies outside the 1.5 \* interquartile-range (1.5 \* *IQR*) is considered an outlier, where, IQR is calculated as the distance between the 25th percentile and 75th percentile values for that variable.

**par**(mfrow=**c**(1, 2)) *# divide graph area in 2 columns*

**boxplot**(cars$speed, main="Speed", sub=**paste**("Outlier rows: ", **boxplot.stats**(cars$speed)$out)) *# box plot for 'speed'*

**boxplot**(cars$dist, main="Distance", sub=**paste**("Outlier rows: ", **boxplot.stats**(cars$dist)$out)) *# box plot for 'distance'*



Density plot – Check if the response variable is close to normality

**library**(e1071)

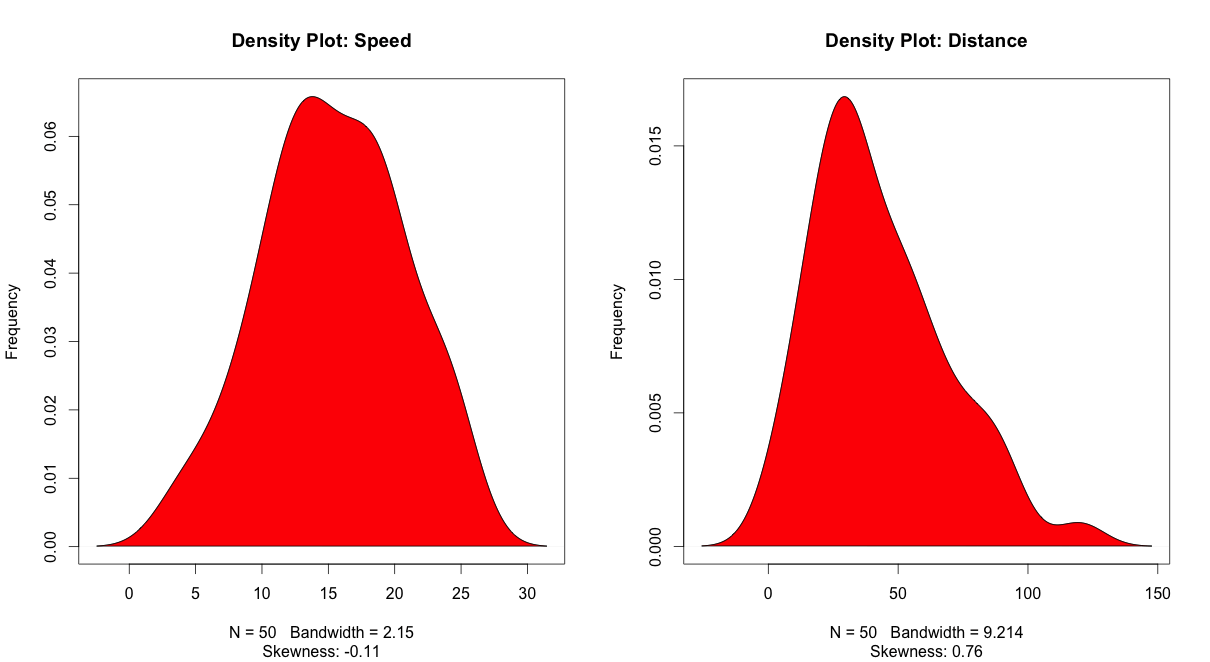
**par**(mfrow=**c**(1, 2)) *# divide graph area in 2 columns*

**plot**(**density**(cars$speed), main="Density Plot: Speed", ylab="Frequency", sub=**paste**("Skewness:", **round**(e1071::**skewness**(cars$speed), 2))) *# density plot for 'speed'*

**polygon**(**density**(cars$speed), col="red")

**plot**(**density**(cars$dist), main="Density Plot: Distance", ylab="Frequency", sub=**paste**("Skewness:", **round**(e1071::**skewness**(cars$dist), 2))) *# density plot for 'dist'*

**polygon**(**density**(cars$dist), col="red")



Correlation

Correlation is a statistical measure that suggests the level of linear dependence between two variables, that occur in pair – just like what we have here in speed and dist. Correlation can take values between -1 to +1. If we observe for every instance where speed increases, the distance also increases along with it, then there is a high positive correlation between them and therefore the correlation between them will be closer to 1. The opposite is true for an inverse relationship, in which case, the correlation between the variables will be close to -1.

A value closer to 0 suggests a weak relationship between the variables. A low correlation (-0.2 < x < 0.2) probably suggests that much of variation of the response variable (*Y*) is unexplained by the predictor (*X*), in which case, we should probably look for better explanatory variables.

**cor**(cars$speed, cars$dist) *# calculate correlation between speed and distance*

*#> [1] 0.8068949*

Build Linear Model

Now that we have seen the linear relationship pictorially in the scatter plot and by computing the correlation, lets see the syntax for building the linear model. The function used for building linear models is lm(). The lm() function takes in two main arguments, namely: 1. Formula 2. Data. The data is typically a data.frame and the formula is a object of class formula. But the most common convention is to write out the formula directly in place of the argument as written below.

linearMod <- **lm**(dist ~ speed, data=cars) *# build linear regression model on full data*

**print**(linearMod)

*#> Call:*

*#> lm(formula = dist ~ speed, data = cars)*

*#>*

*#> Coefficients:*

*#> (Intercept) speed*

*#> -17.579 3.932*

Now that we have built the linear model, we also have established the relationship between the predictor and response in the form of a mathematical formula for Distance (dist) as a function for speed. For the above output, you can notice the ‘Coefficients’ part having two components: *Intercept*: -17.579, *speed*: 3.932 These are also called the beta coefficients. In other words,  ***dist* = *Intercept* + (*β* ∗ *speed*)**=> dist = −17.579 + 3.932∗speed

Linear Regression Diagnostics

Now the linear model is built and we have a formula that we can use to predict the dist value if a corresponding speed is known. Is this enough to actually use this model? NO! Before using a regression model, you have to ensure that it is statistically significant. How do you ensure this? Lets begin by printing the summary statistics for linearMod.

**summary**(linearMod) *# model summary*

*#> Call:*

*#> lm(formula = dist ~ speed, data = cars)*

*#>*

*#> Residuals:*

*#> Min 1Q Median 3Q Max*

*#> -29.069 -9.525 -2.272 9.215 43.201*

*#>*

*#> Coefficients:*

*#> Estimate Std. Error t value Pr(>|t|)*

*#> (Intercept) -17.5791 6.7584 -2.601 0.0123 \**

*#> speed 3.9324 0.4155 9.464 1.49e-12 \*\*\**

*#> ---*

*#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1*

*#>*

*#> Residual standard error: 15.38 on 48 degrees of freedom*

*#> Multiple R-squared: 0.6511, Adjusted R-squared: 0.6438*

*#> F-statistic: 89.57 on 1 and 48 DF, p-value: 1.49e-12*

The p Value: Checking for statistical significance

The summary statistics above tells us a number of things. One of them is the model p-Value (bottom last line) and the p-Value of individual predictor variables (extreme right column under ‘Coefficients’). The p-Values are very important because, We can consider a linear model to be statistically significant only when both these p-Values are less that the pre-determined statistical significance level, which is ideally 0.05. This is visually interpreted by the significance stars at the end of the row. The more the stars beside the variable’s p-Value, the more significant the variable.

Null and alternate hypothesis

When there is a p-value, there is a hull and alternative hypothesis associated with it. In Linear Regression, the Null Hypothesis is that the coefficients associated with the variables is equal to zero. The alternate hypothesis is that the coefficients are not equal to zero (i.e. there exists a relationship between the independent variable in question and the dependent variable).

t-value

We can interpret the t-value something like this. A larger *t-value* indicates that it is less likely that the coefficient is not equal to zero purely by chance. So, higher the t-value, the better.

*Pr(>|t|)* or *p-value* is the probability that you get a t-value as high or higher than the observed value when the Null Hypothesis (the *β* coefficient is equal to zero or that there is no relationship) is true. So if the *Pr(>|t|)* is low, the coefficients are significant (significantly different from zero). If the *Pr(>|t|)* is high, the coefficients are not significant.

What this means to us? when p Value is less than significance level (< 0.05), we can safely reject the null hypothesis that the co-efficient *β* of the predictor is zero. In our case, linearMod, both these p-Values are well below the 0.05 threshold, so we can conclude our model is indeed statistically significant.

It is absolutely important for the model to be statistically significant before we can go ahead and use it to predict (or estimate) the dependent variable, otherwise, the confidence in predicted values from that model reduces and may be construed as an event of chance.

How to calculate the t Statistic and p-Values?

When the model co-efficients and standard error are known, the formula for calculating t Statistic and p-Value is as follows: 

t−Statistic=β−coefficientStd.Errort−Statistic=β−coefficientStd.Error

modelSummary <- **summary**(linearMod) *# capture model summary as an object*

modelCoeffs <- modelSummary$coefficients *# model coefficients*

beta.estimate <- modelCoeffs["speed", "Estimate"] *# get beta estimate for speed*

std.error <- modelCoeffs["speed", "Std. Error"] *# get std.error for speed*

t\_value <- beta.estimate/std.error *# calc t statistic*

p\_value <- 2\***pt**(-**abs**(t\_value), df=**nrow**(cars)-**ncol**(cars)) *# calc p Value*

f\_statistic <- linearMod$fstatistic[1] *# fstatistic*

f <- **summary**(linearMod)$fstatistic *# parameters for model p-value calc*

model\_p <- **pf**(f[1], f[2], f[3], lower=FALSE)

## t Value: 9.46399

## p Value: 1.489836e-12

## Model F Statistic: 89.56711 1 48

## Model p-Value: 1.489836e-12

R-Squared and Adj R-Squared

The actual information in a data is the total variation it contains, remember?. What R-Squared tells us is the proportion of variation in the dependent (response) variable that has been explained by this model.

R2=1−SSESSTR2=1−SSESST

where, *SSE* is the *sum of squared errors* given by SSE=∑ni(yi−yi^)2SSE=∑in(yi−yi^)2 and SST=∑ni(yi−yi¯)2SST=∑in(yi−yi¯)2is the *sum of squared total*. Here, yi^yi^ is the fitted value for observation *i* and y¯y¯ is the mean of *Y*.

We don’t necessarily discard a model based on a low R-Squared value. Its a better practice to look at the AIC and prediction accuracy on validation sample when deciding on the efficacy of a model.

**Now thats about R-Squared. What about adjusted R-Squared?** As you add more *X* variables to your model, the R-Squared value of the new bigger model will always be greater than that of the smaller subset. This is because, since all the variables in the original model is also present, their contribution to explain the dependent variable will be present in the super-set as well, therefore, whatever new variable we add can only add (if not significantly) to the variation that was already explained. It is here, the adjusted R-Squared value comes to help. Adj R-Squared penalizes total value for the number of terms (read predictors) in your model. Therefore when comparing nested models, it is a good practice to look at adj-R-squared value over R-squared.

R2adj=1−MSEMSTRadj2=1−MSEMST

where, *MSE* is the *mean squared error* given by MSE=SSE(n−q)MSE=SSE(n−q) and MST=SST(n−1)MST=SST(n−1) is the *mean squared total*, where *n* is the number of observations and *q* is the number of coefficients in the model.

Therefore, by moving around the numerators and denominators, the relationship between *R*2 and *Radj*2becomes:

R2adj=1−((1−R2)(n−1)n−q)Radj2=1−((1−R2)(n−1)n−q)

Standard Error and F-Statistic

Both standard errors and F-statistic are measures of goodness of fit.

Std.Error=MSE−−−−−√=SSEn−q−−−−−√Std.Error=MSE=SSEn−q

F−statistic=MSRMSEF−statistic=MSRMSE

where, *n* is the number of observations, *q* is the number of coefficients and *MSR* is the *mean square regression*, calculated as,

MSR=∑ni(yi−y¯^)q−1=SST−SSEq−1MSR=∑in(yi−y¯^)q−1=SST−SSEq−1

AIC and BIC

The Akaike’s information criterion - AIC (Akaike, 1974) and the Bayesian information criterion - BIC (Schwarz, 1978) are measures of the goodness of fit of an estimated statistical model and can also be used for model selection. Both criteria depend on the maximized value of the likelihood function L for the estimated model.

The AIC is defined as:

*AIC* = (−2) × *ln*(*L*) + (2×*k*)

where, k is the number of model parameters and the BIC is defined as:

*BIC* = (−2) × *ln*(*L*) + *k* × *ln*(*n*)

where, n is the sample size.

For model comparison, the model with the lowest AIC and BIC score is preferred.

**AIC**(linearMod) *# AIC => 419.1569*

**BIC**(linearMod) *# BIC => 424.8929*

How to know if the model is best fit for your data?

The most common metrics to look at while selecting the model are:

| **STATISTIC** | **CRITERION** |
| --- | --- |
| R-Squared | Higher the better *(> 0.70)* |
| Adj R-Squared | Higher the better |
| F-Statistic | Higher the better |
| Std. Error | Closer to zero the better |
| t-statistic | Should be greater 1.96 for p-value to be less than 0.05 |
| AIC | Lower the better |
| BIC | Lower the better |
| Mallows cp | Should be close to the number of predictors in model |
| MAPE (Mean absolute percentage error) | Lower the better |
| MSE (Mean squared error) | Lower the better |
| Min\_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted)) | Higher the better |

Predicting Linear Models

So far we have seen how to build a linear regression model using the whole dataset. If we build it that way, there is no way to tell how the model will perform with new data. So the preferred practice is to split your dataset into a 80:20 sample (training:test), then, build the model on the 80% sample and then use the model thus built to predict the dependent variable on test data.

Doing it this way, we will have the model predicted values for the 20% data (test) as well as the actuals (from the original dataset). By calculating accuracy measures (like min\_max accuracy) and error rates (MAPE or MSE), we can find out the prediction accuracy of the model. Now, lets see how to actually do this..

Step 1: Create the training (development) and test (validation) data samples from original data.

*# Create Training and Test data -*

**set.seed**(100) *# setting seed to reproduce results of random sampling*

trainingRowIndex <- **sample**(1:**nrow**(cars), 0.8\***nrow**(cars)) *# row indices for training data*

trainingData <- cars[trainingRowIndex, ] *# model training data*

testData <- cars[-trainingRowIndex, ] *# test data*

Step 2: Develop the model on the training data and use it to predict the distance on test data

*# Build the model on training data -*

lmMod <- **lm**(dist ~ speed, data=trainingData) *# build the model*

distPred <- **predict**(lmMod, testData) *# predict distance*

Step 3: Review diagnostic measures.

**summary** (lmMod) *# model summary*

*#>*

*#> Call:*

*#> lm(formula = dist ~ speed, data = trainingData)*

*#>*

*#> Residuals:*

*#> Min 1Q Median 3Q Max*

*#> -23.350 -10.771 -2.137 9.255 42.231*

*#>*

*#> Coefficients:*

*#> Estimate Std. Error t value Pr(>|t|)*

*#> (Intercept) -22.657 7.999 -2.833 0.00735 \*\**

*#> speed 4.316 0.487 8.863 8.73e-11 \*\*\**

*#> ---*

*#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1*

*#>*

*#> Residual standard error: 15.84 on 38 degrees of freedom*

*#> Multiple R-squared: 0.674, Adjusted R-squared: 0.6654*

*#> F-statistic: 78.56 on 1 and 38 DF, p-value: 8.734e-11*

**AIC** (lmMod) *# Calculate akaike information criterion*

*#> [1] 338.4489*

From the model summary, the model p value and predictor’s p value are less than the significance level, so we know we have a statistically significant model. Also, the R-Sq and Adj R-Sq are comparative to the original model built on full data.

Step 4: Calculate prediction accuracy and error rates

A simple correlation between the actuals and predicted values can be used as a form of accuracy measure. A higher correlation accuracy implies that the actuals and predicted values have similar directional movement, i.e. when the actuals values increase the predicteds also increase and vice-versa.

actuals\_preds <- **data.frame**(**cbind**(actuals=testData$dist, predicteds=distPred)) *# make actuals\_predicteds dataframe.*

correlation\_accuracy <- **cor**(actuals\_preds) *# 82.7%*

**head**(actuals\_preds)

*#> actuals predicteds*

*#> 1 2 -5.392776*

*#> 4 22 7.555787*

*#> 8 26 20.504349*

*#> 20 26 37.769100*

*#> 26 54 42.085287*

*#> 31 50 50.717663*

Now lets calculate the Min Max accuracy and MAPE: 

MinMaxAccuracy=mean(min(actuals,predicteds)max(actuals,predicteds))MinMaxAccuracy=mean(min(actuals,predicteds)max(actuals,predicteds))

MeanAbsolutePercentageError (MAPE)=mean(abs(predicteds−actuals)actuals)MeanAbsolutePercentageError (MAPE)=mean(abs(predicteds−actuals)actuals)

min\_max\_accuracy <- **mean**(**apply**(actuals\_preds, 1, min) / **apply**(actuals\_preds, 1, max))

*# => 58.42%, min\_max accuracy*

mape <- **mean**(**abs**((actuals\_preds$predicteds - actuals\_preds$actuals))/actuals\_preds$actuals)

*# => 48.38%, mean absolute percentage deviation*

k- Fold Cross validation

Suppose, the model predicts satisfactorily on the 20% split (test data), is that enough to believe that your model will perform equally well all the time? It is important to rigorously test the model’s performance as much as possible. One way is to ensure that the model equation you have will perform well, when it is ‘built’ on a different subset of training data and predicted on the remaining data.

How to do this is? Split your data into ‘k’ mutually exclusive random sample portions. Keeping each portion as test data, we build the model on the remaining (k-1 portion) data and calculate the mean squared error of the predictions. This is done for each of the ‘k’ random sample portions. Then finally, the average of these mean squared errors (for ‘k’ portions) is computed. We can use this metric to compare different linear models.

By doing this, we need to check two things:

1. If the model’s prediction accuracy isn’t varying too much for any one particular sample, and
2. If the lines of best fit don’t vary too much with respect the the slope and level.

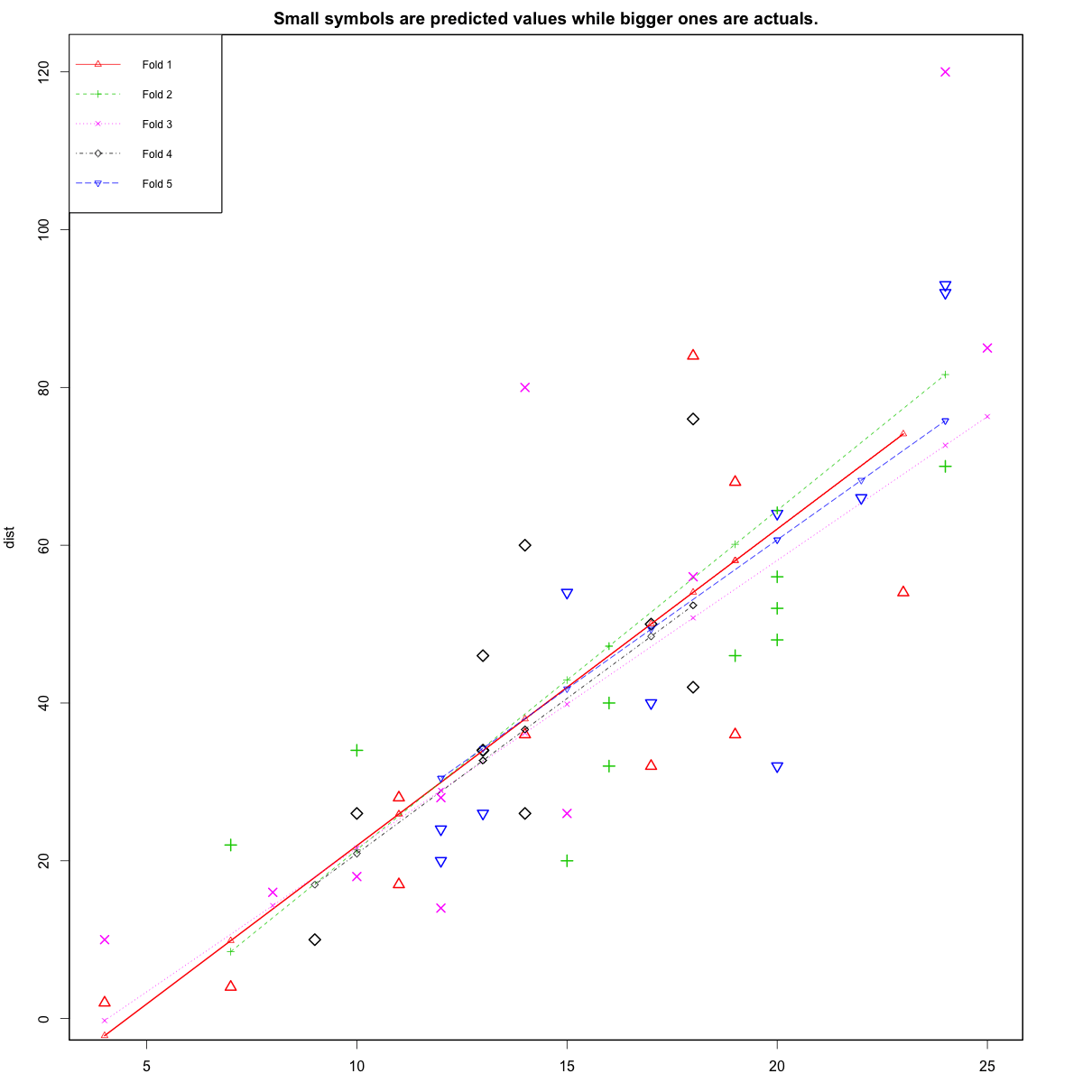
In other words, they should be parallel and as close to each other as possible. You can find a more detailed explanation for interpreting the cross validation charts when you learn about advanced linear model building.

**library**(DAAG)

cvResults <- **suppressWarnings**(**CVlm**(df=cars, form.lm=dist ~ speed, m=5, dots=FALSE, seed=29, legend.pos="topleft", printit=FALSE, main="Small symbols are predicted values while bigger ones are actuals.")); *# performs the CV*

**attr**(cvResults, 'ms') *# => 251.2783 mean squared error*

In the below plot, Are the dashed lines parallel? Are the small and big symbols are not over dispersed for one particular color?



Where to go from here?

We have covered the basic concepts about linear regression. Besides these, you need to understand that linear regression is based on certain underlying [assumptions](http://r-statistics.co/Assumptions-of-Linear-Regression.html) that must be taken care especially when working with multiple *Xs*. Once you are familiar with that, the [advanced regression models](http://r-statistics.co/adv-regression-models-html) will show you around the various special cases where a different form of regression would be more suitable.

1. create a graph displaying the accuracy of all models

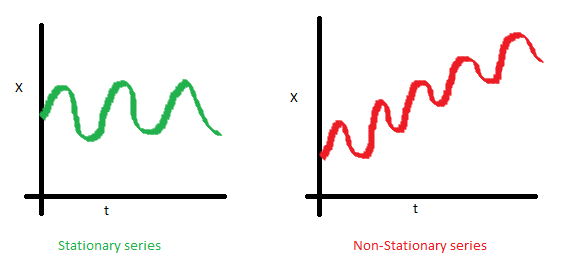
**1. Basics – Time Series Modeling**

Let’s begin from basics.  This includes stationary series, random walks , Rho Coefficient, Dickey Fuller Test of Stationarity. If these terms are already scaring you, don’t worry – they will become clear in a bit and I bet you will start enjoying the subject as I explain it.

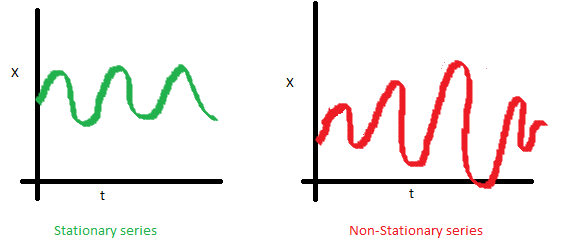
**Stationary Series**

There are three basic criterion for a series to be classified as stationary series :

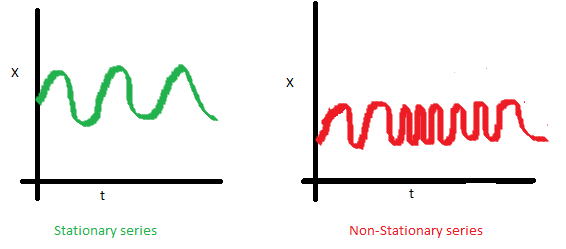
1. The mean of the series should not be a function of time rather should be a constant. The image below has the left hand graph satisfying the condition whereas the graph in red has a time dependent mean.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Mean_nonstationary.png)

2. The variance of the series should not a be a function of time. This property is known as homoscedasticity. Following graph depicts what is and what is not a stationary series. (Notice the varying spread of distribution in the right hand graph)

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Var_nonstationary.png)

3. The covariance of the i th term and the (i + m) th term should not be a function of time. In the following graph, you will notice the spread becomes closer as the time increases. Hence, the covariance is not constant with time for the ‘red series’.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Cov_nonstationary.png)

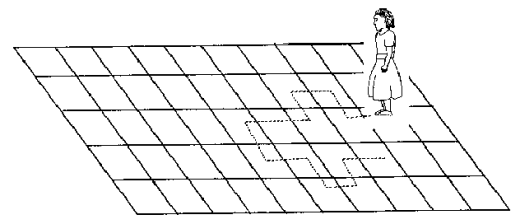
**Why do I care about ‘stationarity’ of a time series?**

The reason I took up this section first was that until unless your time series is stationary, you cannot build a time series model. In cases where the stationary criterion are violated, the first requisite becomes to stationarize the time series and then try stochastic models to predict this time series. There are multiple ways of bringing this stationarity. Some of them are Detrending, Differencing etc.

**Random Walk**

This is the most basic concept of the time series. You might know the concept well. But, I found many people in the industry who interprets random walk as a stationary process. In this section with the help of some mathematics, I will make this concept crystal clear for ever. Let’s take an example.

**Example:** Imagine a girl moving randomly on a giant chess board. In this case, next position of the girl is only dependent on the last position.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/RandomWalk.gif)

(Source: http://scifun.chem.wisc.edu/WOP/RandomWalk.html )

Now imagine, you are sitting in another room and are not able to see the girl. You want to predict the position of the girl with time. How accurate will you be? Of course you will become more and more inaccurate as the position of the girl changes. At t=0 you exactly know where the girl is. Next time, she can only move to 8 squares and hence your probability dips to 1/8 instead of 1 and it keeps on going down. Now let’s try to formulate this series :

X(t) = X(t-1) + Er(t)

where Er(t) is the error at time point t. This is the randomness the girl brings at every point in time.

Now, if we recursively fit in all the Xs, we will finally end up to the following equation :

X(t) = X(0) + Sum(Er(1),Er(2),Er(3).....Er(t))

Now, lets try validating our assumptions of stationary series on this random walk formulation:

**1. Is the Mean constant ?**

E[X(t)] = E[X(0)] + Sum(E[Er(1)],E[Er(2)],E[Er(3)].....E[Er(t)])

We know that Expectation of any Error will be zero as it is random.

Hence we get E[X(t)] = E[X(0)] = Constant.

**2. Is the Variance constant?**

Var[X(t)] = Var[X(0)] + Sum(Var[Er(1)],Var[Er(2)],Var[Er(3)].....Var[Er(t)])

Var[X(t)] = t \* Var(Error) = Time dependent.

Hence, we infer that the random walk is not a stationary process as it has a time variant variance. Also, if we check the covariance, we see that too is dependent on time.

**Let’s spice up things a bit,**

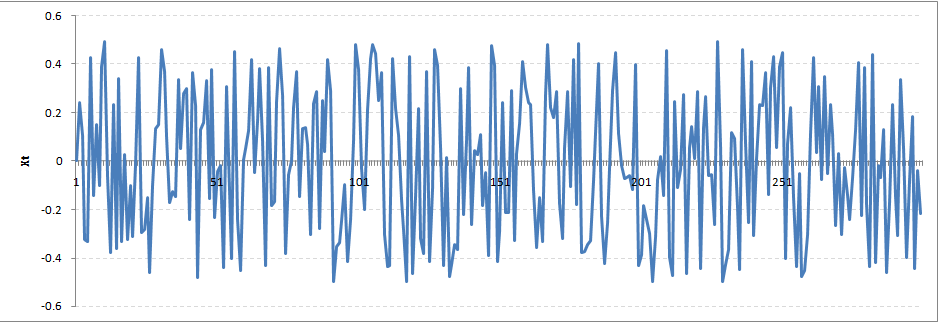
We already know that a random walk is a non-stationary process. Let us introduce a new coefficient in the equation to see if we can make the formulation stationary.

**Introduced coefficient : Rho**

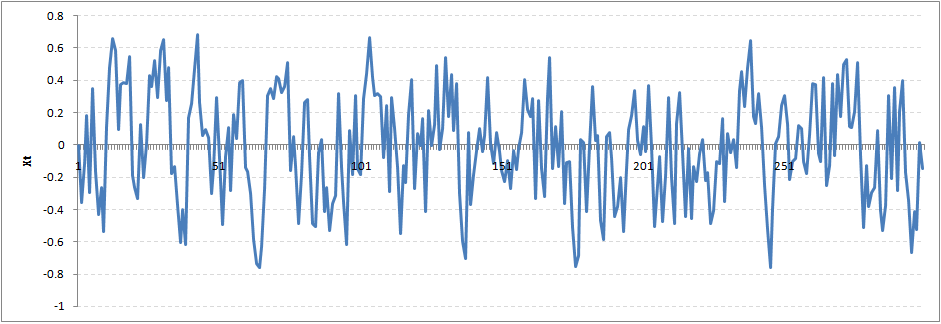
X(t) = Rho \* X(t-1) + Er(t)

Now, we will vary the value of Rho to see if we can make the series stationary. Here we will interpret the scatter visually and not do any test to check stationarity.

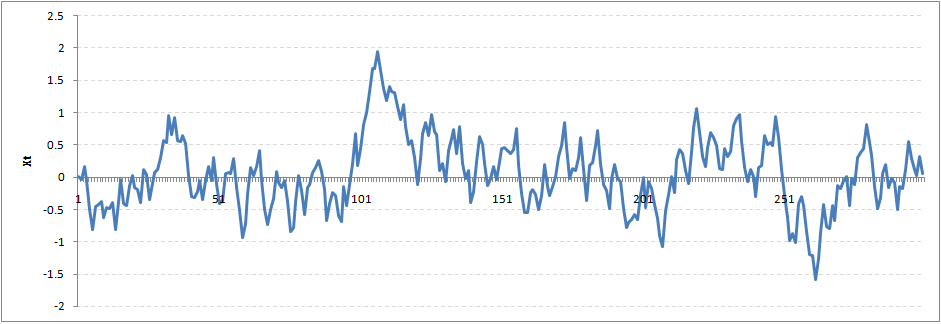
Let’s start with a perfectly stationary series with Rho = 0 . Here is the plot for the time series :

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/rho0.png)

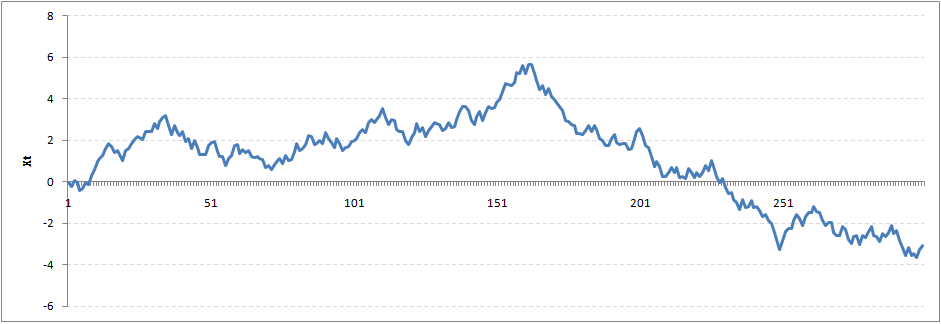
Increase the value of Rho to 0.5 gives us following graph :

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/rho5.png)

You might notice that our cycles have become broader but essentially there does not seem to be a serious violation of stationary assumptions. Let’s now take a more extreme case of Rho = 0.9

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/rho9.png)

We still see that the X returns back from extreme values to zero after some intervals. This series also is not violating non-stationarity significantly. Now, let’s take a look at the random walk with rho = 1.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/rho1.png)

This obviously is an violation to stationary conditions. What makes rho = 1 a special case which comes out badly in stationary test? We will find the mathematical reason to this.

Let’s take expectation on each side of the equation  “X(t) = Rho \* X(t-1) + Er(t)”

E[X(t)] = Rho \*E[ X(t-1)]

This equation is very insightful. The next X (or at time point t) is being pulled down to Rho \* Last value of X.

For instance, if X(t – 1 ) = 1, E[X(t)] = 0.5 ( for Rho = 0.5) . Now, if X moves to any direction from zero, it is pulled back to zero in next step. The only component which can drive it even further is the error term. Error term is equally probable to go in either direction. What happens when the Rho becomes 1? No force can pull the X down in the next step.

**Dickey Fuller Test of Stationarity**

What you just learnt in the last section is formally known as Dickey Fuller test. Here is a small tweak which is made for our equation to convert it to a Dickey Fuller test:

X(t) = Rho \* X(t-1) + Er(t)

=>  X(t) - X(t-1) = (Rho - 1) X(t - 1) + Er(t)

We have to test if Rho – 1 is significantly different than zero or not. If the null hypothesis gets rejected, we’ll get a stationary time series.

Stationary testing and converting a series into a stationary series are the most critical processes in a time series modelling. You need to memorize each and every detail of this concept to move on to the next step of time series modelling.

Let’s now consider an example to show you what a time series looks like.

**2. Exploration of Time Series Data in R**

Here we’ll learn to handle time series data on R. Our scope will be restricted to data exploring in a time series type of data set and not go to building time series models.

I have used an inbuilt data set of R called AirPassengers. The dataset consists of monthly totals of international airline passengers, 1949 to 1960.

**Loading the Data Set**

Following is the code which will help you load the data set and spill out a few top level metrics.

> data(AirPassengers)

> class(AirPassengers)

[1] "ts"

#This tells you that the data series is in a time series format

> start(AirPassengers)

[1] 1949 1

#This is the start of the time series

> end(AirPassengers)

[1] 1960 12

#This is the end of the time series

> frequency(AirPassengers)

[1] 12

#The cycle of this time series is 12months in a year

> summary(AirPassengers)

Min. 1st Qu. Median Mean 3rd Qu. Max.

104.0 180.0 265.5 280.3 360.5 622.0

**Detailed Metrics**

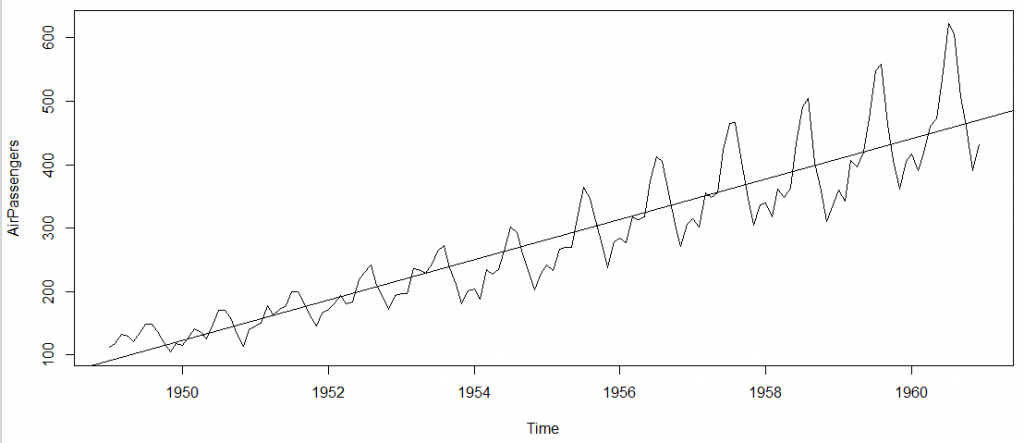
#The number of passengers are distributed across the spectrum

> plot(AirPassengers)

#This will plot the time series

>abline(reg=lm(AirPassengers~time(AirPassengers)))

# This will fit in a line

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_AP1.png)

Here are a few more operations you can do:

> cycle(AirPassengers)

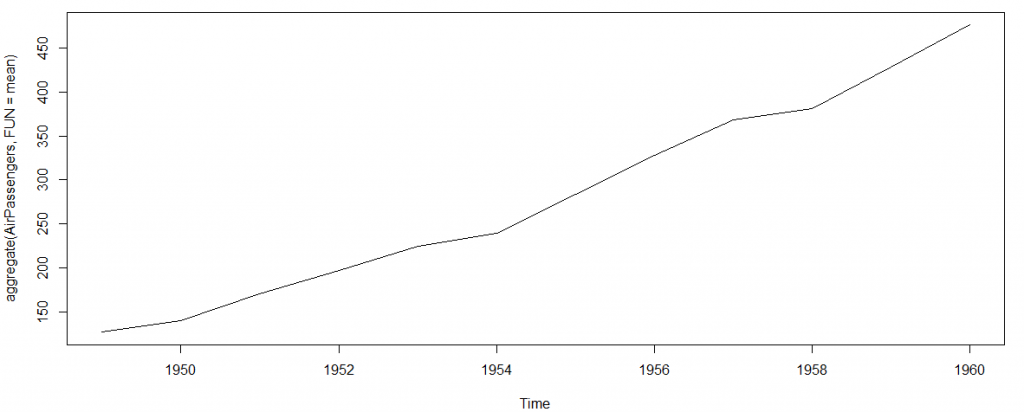
#This will print the cycle across years.

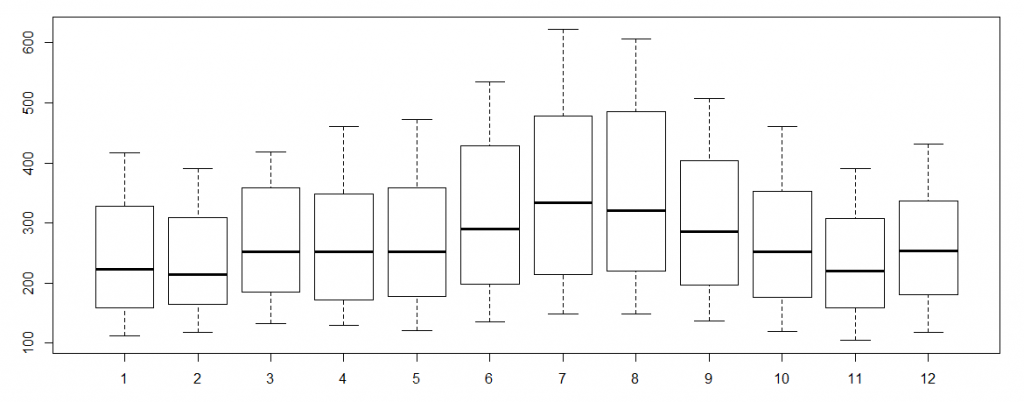
>plot(aggregate(AirPassengers,FUN=mean))

#This will aggregate the cycles and display a year on year trend

> boxplot(AirPassengers~cycle(AirPassengers))

#Box plot across months will give us a sense on seasonal effect

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_aggregate.png)

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_month_wise.png)

**Important Inferences**

1. The year on year trend clearly shows that the #passengers have been increasing without fail.
2. The variance and the mean value in July and August is much higher than rest of the months.
3. Even though the mean value of each month is quite different their variance is small. Hence, we have strong seasonal effect with a cycle of 12 months or less.

Exploring data becomes most important in a time series model – without this exploration, you will not know whether a series is stationary or not. As in this case we already know many details about the kind of model we are looking out for.

Let’s now take up a few time series models and their characteristics. We will also take this problem forward and make a few predictions.

**3. Introduction to ARMA Time Series Modeling**

ARMA models are commonly used in time series modeling. In ARMA model, AR stands for auto-regression and MA stands for moving average. If these words sound intimidating to you, worry not – I’ll simplify these concepts in next few minutes for you!

We will now develop a knack for these terms and understand the characteristics associated with these models. **But before we start, you should remember, AR or MA are not applicable on non-stationary series**.

In case you get a non stationary series, you first need to stationarize the series (by taking difference / transformation) and then choose from the available time series models.

First, I’ll explain each of these two models (AR & MA) individually. Next, we will look at the characteristics of these models.

**Auto-Regressive Time Series Model**

Let’s understanding AR models using the case below:

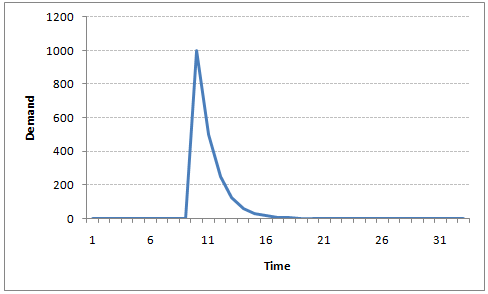
The current GDP of a country say x(t) is dependent on the last year’s GDP i.e. x(t – 1). The hypothesis being that the total cost of production of products & services in a country in a fiscal year (known as GDP) is dependent on the set up of manufacturing plants / services in the previous year and the newly set up industries / plants / services in the current year. But the primary component of the GDP is the former one.

Hence, we can formally write the equation of GDP as:

**x(t) = alpha \*  x(t – 1) + error (t)**

This equation is known as *AR(1) formulation*. The numeral one (1) denotes that the next instance is solely dependent on the previous instance.  The alpha is a coefficient which we seek so as to minimize the error function. Notice that x(t- 1) is indeed linked to x(t-2) in the same fashion. Hence, any shock to x(t) will gradually fade off in future.

For instance, let’s say x(t) is the number of juice bottles sold in a city on a particular day. During winters, very few vendors purchased juice bottles. Suddenly, on a particular day, the temperature rose and the demand of juice bottles soared to 1000. However, after a few days, the climate became cold again. But, knowing that the people got used to drinking juice during the hot days, there were 50% of the people still drinking juice during the cold days. In following days, the proportion went down to 25% (50% of 50%) and then gradually to a small number after significant number of days. The following graph explains the inertia property of AR series:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/AR1.png)

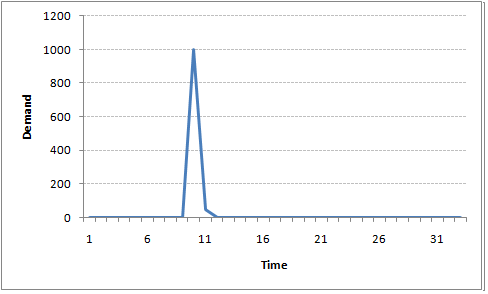
**Moving Average Time Series Model**

Let’s take another case to understand Moving average time series model.

A manufacturer produces a certain type of bag, which was readily available in the market. Being a competitive market, the sale of the bag stood at zero for many days. So, one day he did some experiment with the design and produced a different type of bag. This type of bag was not available anywhere in the market. Thus, he was able to sell the entire stock of 1000 bags (lets call this as x(t) ). The demand got so high that the bag ran out of stock. As a result, some 100 odd customers couldn’t purchase this bag. Lets call this gap as the error at that time point. With time, the bag had lost its woo factor. But still few customers were left who went empty handed the previous day. Following is a simple formulation to depict the scenario :

**x(t) = beta \*  error(t-1) + error (t)**

If we try plotting this graph, it will look something like this :

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/MA1.png)

Did you notice the difference between MA and AR model? In MA model, noise / shock quickly vanishes with time. The AR model has a much lasting effect of the shock.

**Difference between AR and MA models**

The primary difference between an AR and MA model is based on the correlation between time series objects at different time points. The correlation between x(t) and x(t-n) for n > order of MA is always zero. This directly flows from the fact that covariance between x(t) and x(t-n) is zero for MA models (something which we refer from the example taken in the previous section). However, the correlation of x(t) and x(t-n) gradually declines with n becoming larger in the AR model. This difference gets exploited irrespective of having the AR model or MA model. The correlation plot can give us the order of MA model.

**Exploiting ACF and PACF plots**

Once we have got the stationary time series, we must answer two primary questions:

*Q1. Is it an AR or MA process?*

*Q2. What order of AR or MA process do we need to use?*

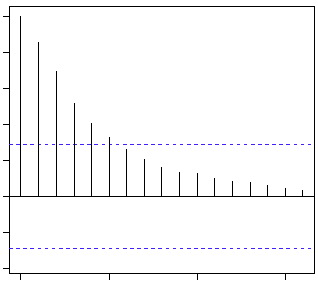
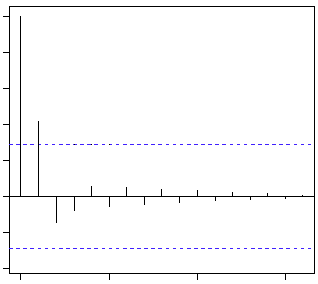
The trick to solve these questions is available in the previous section. Didn’t you notice?

The first question can be answered using Total Correlation Chart (also known as Auto – correlation Function / ACF). ACF is a plot of total correlation between different lag functions. For instance, in GDP problem, the GDP at time point t is x(t). We are interested in the correlation of x(t) with x(t-1) , x(t-2) and so on. Now let’s reflect on what we have learnt above.

In a moving average series of lag n, we will not get any correlation between x(t) and x(t – n -1) . Hence, the total correlation chart cuts off at nth lag. So it becomes simple to find the lag for a MA series. For an AR series this correlation will gradually go down without any cut off value. So what do we do if it is an AR series?

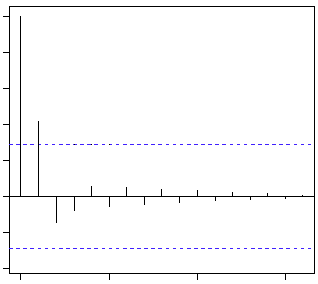
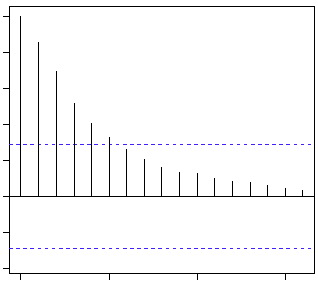
Here is the second trick. If we find out the partial correlation of each lag, it will cut off after the degree of AR series. For instance,if we have a AR(1) series,  if we exclude the effect of 1st lag (x (t-1) ), our 2nd lag (x (t-2) ) is independent of x(t). Hence, the partial correlation function (PACF) will drop sharply after the 1st lag. Following are the examples which will clarify any doubts you have on this concept :

**ACF**                                                                      **PACF**

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Gradual-decline.gif)[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/cut-off.gif)

The blue line above shows significantly different values than zero. Clearly, the graph above has a cut off on PACF curve after 2nd lag which means this is mostly an AR(2) process.

**ACF                                                                 PACF**

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/cut-off.gif)[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/Gradual-decline.gif)

Clearly, the graph above has a cut off on ACF curve after 2nd lag which means this is mostly a MA(2) process.

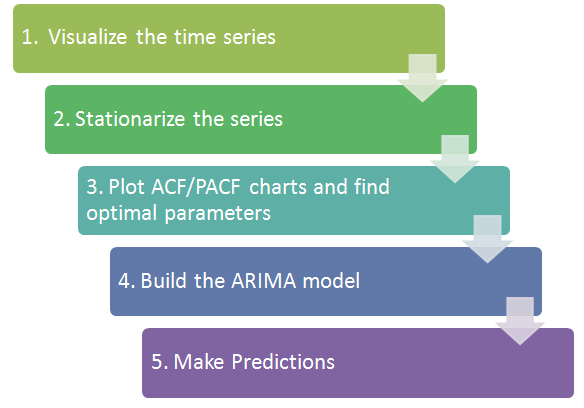
Till now, we have covered on how to identify the type of stationary series using ACF & PACF plots. Now, I’ll introduce you to a comprehensive framework to build a time series model.  In addition, we’ll also discuss about the practical applications of time series modelling.

**4. Framework and Application of ARIMA Time Series Modeling**

A quick revision, Till here we’ve learnt basics of time series modeling, time series in R and ARMA modeling. Now is the time to join these pieces and make an interesting story.

**Overview of the Framework**

This framework(shown below) specifies the step by step approach on ‘**How to do a Time Series Analysis**‘:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/flowchart.png)

As you would be aware, the first three steps have already been discussed above. Nevertheless, the same has been delineated briefly below:

**Step 1: Visualize the Time Series**

It is essential to analyze the trends prior to building any kind of time series model. The details we are interested in pertains to any kind of trend, seasonality or random behaviour in the series. We have covered this part in the second part of this series.

**Step 2: Stationarize the Series**

Once we know the patterns, trends, cycles and seasonality , we can check if the series is stationary or not. Dickey – Fuller is one of the popular test to check the same. We have covered this test in the [first part](https://www.analyticsvidhya.com/blog/2015/02/step-step-guide-learn-time-series/) of this article series. This doesn’t ends here! What if the series is found to be non-stationary?

There are three commonly used technique to make a time series stationary:

1.  **Detrending** : Here, we simply remove the trend component from the time series. For instance, the equation of my time series is:

**x(t) = (mean + trend \* t) + error**

We’ll simply remove the part in the parentheses and build model for the rest.

2. **Differencing** : This is the commonly used technique to remove non-stationarity. Here we try to model the differences of the terms and not the actual term. For instance,

**x(t) – x(t-1) = ARMA (p ,  q)**

This differencing is called as the Integration part in AR(I)MA. Now, we have three parameters

**p : AR**

**d : I**

**q : MA**

3. **Seasonality** : Seasonality can easily be incorporated in the ARIMA model directly. More on this has been discussed in the applications part below.

**Step 3: Find Optimal Parameters**

The parameters p,d,q can be found using  [ACF and PACF plots](https://www.analyticsvidhya.com/blog/2015/03/introduction-auto-regression-moving-average-time-series/). An addition to this approach is can be, if both ACF and PACF decreases gradually, it indicates that we need to make the time series stationary and introduce a value to “d”.

**Step 4: Build ARIMA Model**

With the parameters in hand, we can now try to build ARIMA model. The value found in the previous section might be an approximate estimate and we need to explore more (p,d,q) combinations. The one with the lowest BIC and AIC should be our choice. We can also try some models with a seasonal component. Just in case, we notice any seasonality in ACF/PACF plots.

**Step 5: Make Predictions**

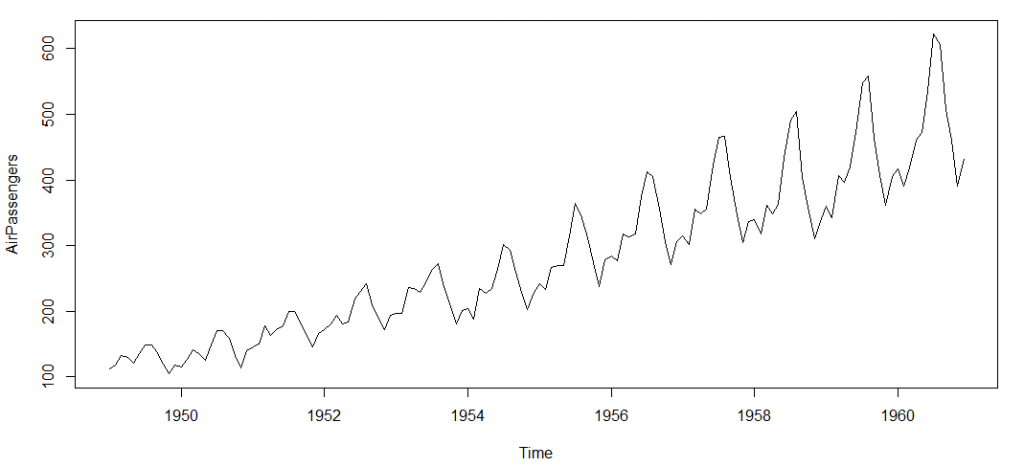
Once we have the final ARIMA model, we are now ready to make predictions on the future time points. We can also visualize the trends to cross validate if the model works fine.

**Applications of Time Series Model**

Now, we’ll use the same example that we have used above. Then, using time series, we’ll make future predictions. We recommend you to check out the example before proceeding further.

**Where did we start ?**

Following is the plot of the number of passengers with years. Try and make observations on this plot before moving further in the article.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_AP.png)

Here are my observations :

1. There is a trend component which grows the passenger year by year.

2. There looks to be a seasonal component which has a cycle less than 12 months.

3. The variance in the data keeps on increasing with time.

We know that we need to address two issues before we test stationary series. One, we need to remove unequal variances. We do this using log of the series. Two, we need to address the trend component. We do this by taking difference of the series. Now, let’s test the resultant series.

adf.test(diff(log(AirPassengers)), alternative="stationary", k=0)

Augmented Dickey-Fuller Test

data: diff(log(AirPassengers))

Dickey-Fuller = -9.6003, Lag order = 0,

p-value = 0.01

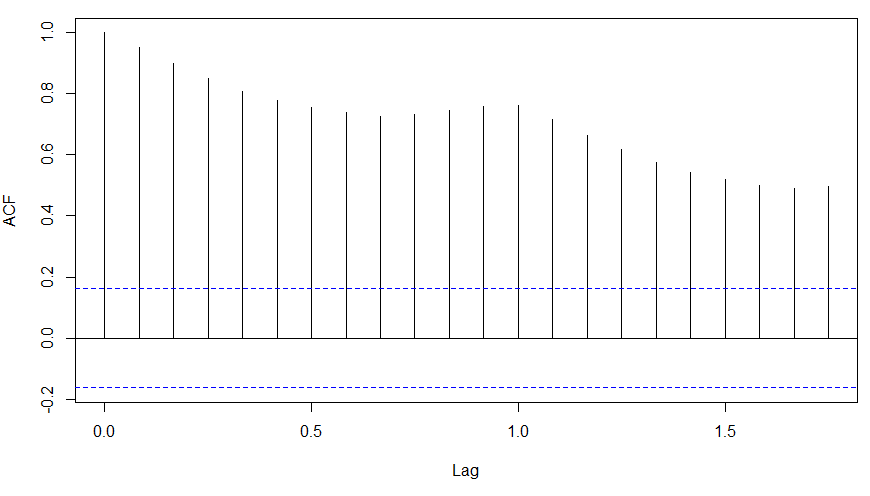
alternative hypothesis: stationary

We see that the series is stationary enough to do any kind of time series modelling.

Next step is to find the right parameters to be used in the ARIMA model. We already know that the ‘d’ component is 1 as we need 1 difference to make the series stationary. We do this using the Correlation plots. Following are the ACF plots for the series :

#ACF Plots

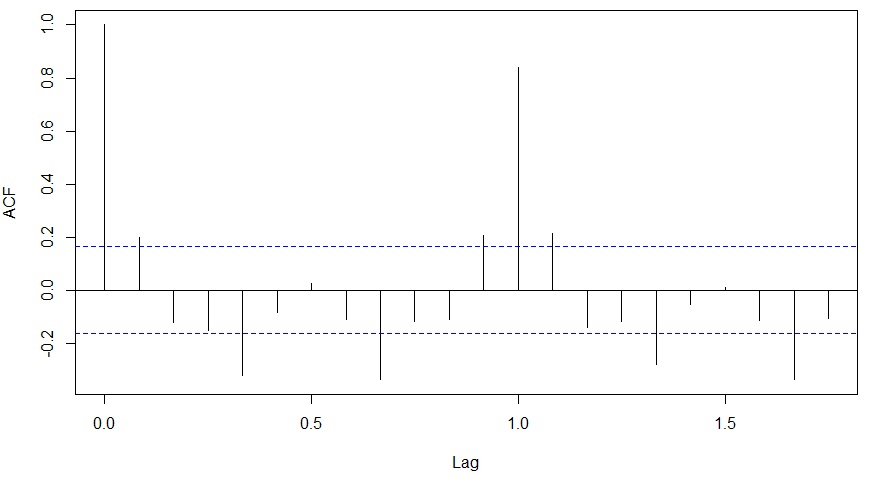
acf(log(AirPassengers))

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/ACF_original.png)

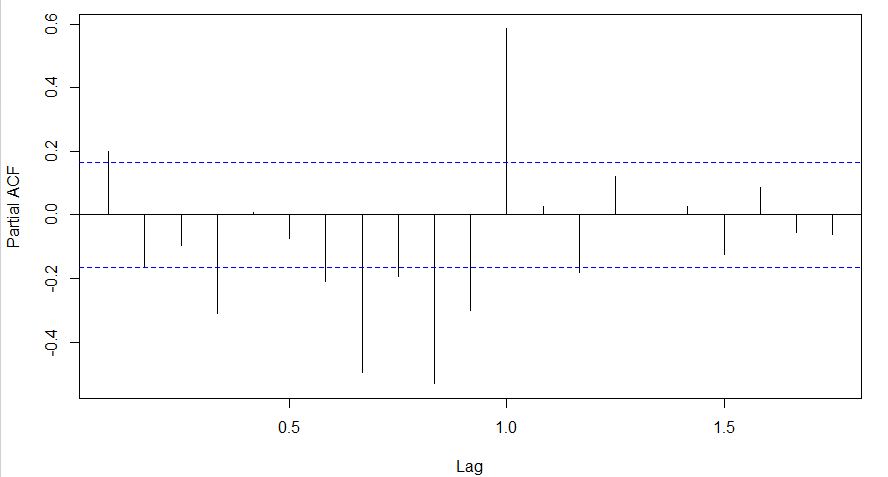
**What do you see in the chart shown above?**

Clearly, the decay of ACF chart is very slow, which means that the population is not stationary. We have already discussed above that we now intend to regress on the difference of logs rather than log directly. Let’s see how ACF and PACF curve come out after regressing on the difference.

acf(diff(log(AirPassengers)))

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/ACF-diff.png)

pacf(diff(log(AirPassengers)))

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/PACF-diff.png)

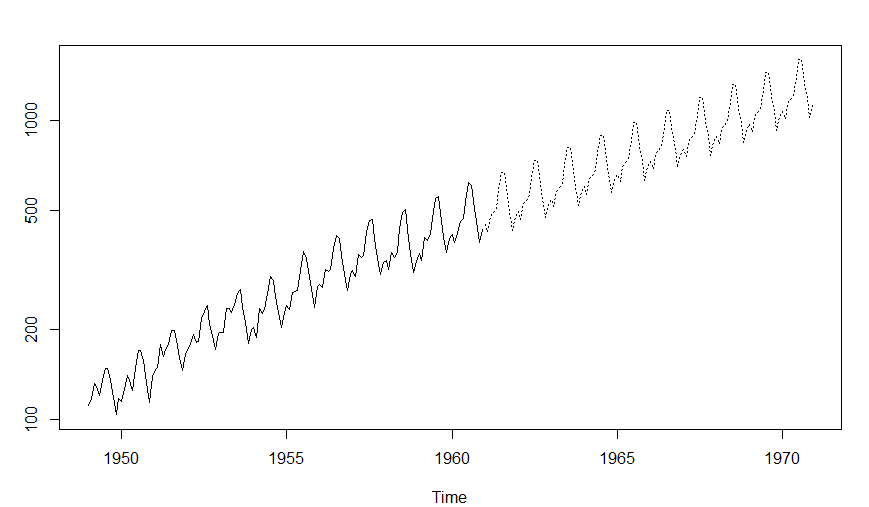
Clearly, ACF plot cuts off after the first lag. Hence, we understood that value of p should be 0 as the ACF is the curve getting a cut off. While value of q should be 1 or 2. After a few iterations, we found that (0,1,1) as (p,d,q) comes out to be the combination with least AIC and BIC.

Let’s fit an ARIMA model and predict the future 10 years. Also, we will try fitting in a seasonal component in the ARIMA formulation. Then, we will visualize the prediction along with the training data. You can use the following code to do the same :

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10\*12)

ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/02/predictions.png)

**End Notes**

With this, we come to this end of tutorial on Time Series Modeling. I hope this will help you to improve your knowledge to work on time based data. To reap maximum benefits out of this tutorial, I’d suggest you to practice these R codes side by side and check your progress.

Did you find the article useful? Share with us if you have done similar kind of analysis before. Do let us know your thoughts about this article in the box below.