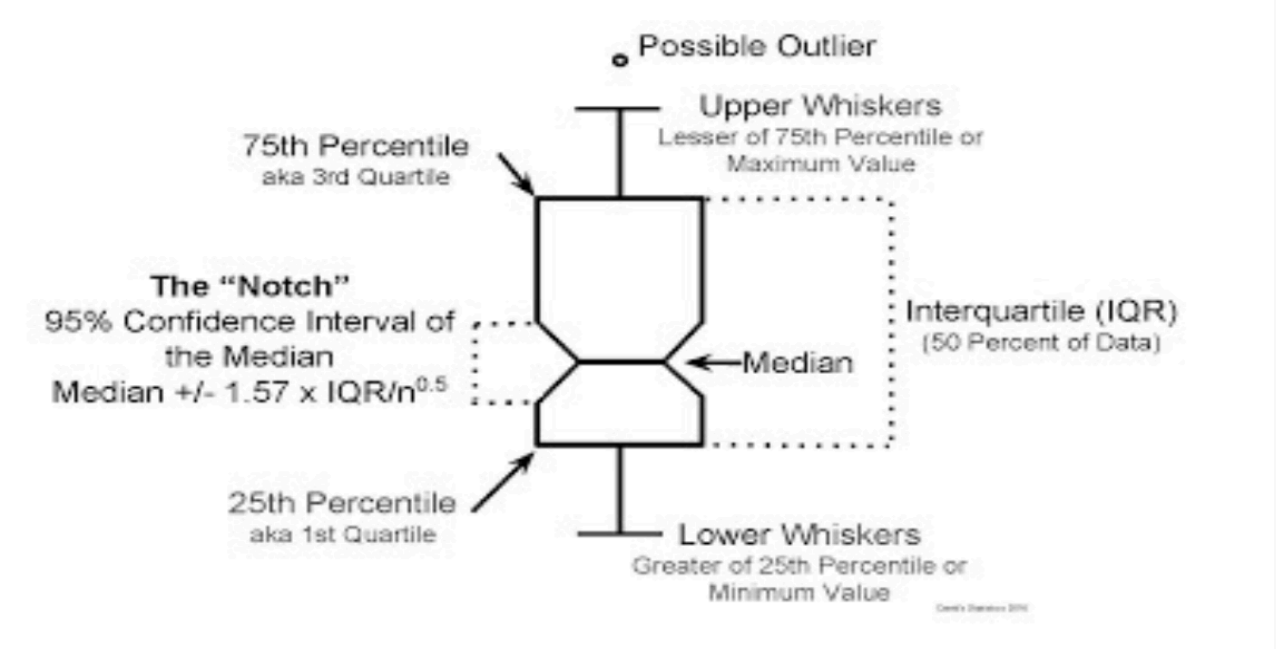
INTRODUCTION TO DATA ANALYSIS

Partha Padmanabhan

UCSC Silicon Valley Extension
YOUR UC IN SILICON VALLEY

Lecture 5 – R Statistics

Box Plot with a Notch



*Notch indicates the confidence interval around the median which is normally based on the median +/- 1.57 x IQR/sqrt of n. (Interquartile range)
https://sites.google.com/site/davidsstatistics/home/notched-box-plots

Basic Descriptive Statistics

How to get the descriptive statistics for the list of variables from your dataset?

```
install.packages(“pastecs”)
library(pastecs)
```

Let us read a file to use this package

```
ff <- read.table("forestfires.csv", sep=",", header = T)
head(ff)
```

How do we get the descriptive statistics?

```
attach(ff)
tvalues <- cbind(FFMC,DMC,DC,ISI,temp)
stat.desc(tvalues)
```

How to change the scientific notation and change the display format?

Scipen:- A penalty to be applied when deciding to print numeric values in fixed or exponential notation. Positive values bias towards fixed and negative towards scientific notation: fixed notation will be preferred unless it is more than scipen digits wider.

```
options(scipen=100)
options(digits=2)
stat.desc(tvalues)
```

If you need only the descriptive statistics(mean, median,std.dev) , add the option:

```
stat.desc(tvalues, basic = F)
```

If you need only the basic statistics(number of observations etc.) add the option:

```
stat.desc(tvalues, desc = F)
```

Regression Analysis:

Multiple Regression

Logistic Regression

Stochastic Processes

Normal Distribution

Binomial Distribution

Random Number generation

Poisson Regression

Analysis of Covariance

Regression analysis

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.
The general mathematical equation for a linear regression is −

$$y = ax + b$$

Following is the description of the parameters used −

•**y** is the response variable.

•**x** is the predictor variable.

•**a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

•Carry out the experiment of gathering a sample of observed values of height and corresponding weight.

•Create a relationship model using the **lm()** functions in R.

•Find the coefficients from the model created and create the mathematical equation using these

•Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.

•To predict the weight of new persons, use the **predict()** function in R.

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in linear regression is −

```
lm(formula,data)
```

Following is the description of the parameters used −

- **formula** is a symbol presenting the relation between x and y.
- **data** is the vector on which the formula will be applied.

Create Relationship Model & get the Coefficients

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
print(relation)
```

Get the Summary of the Relationship

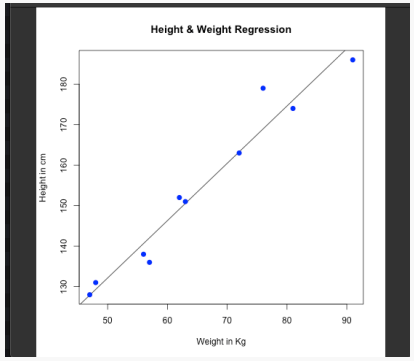
```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
print(summary(relation))
```

predict() Function
Syntax
The basic syntax for predict() in linear regression is −
predict(object, newdata) Following is the description of the parameters used −
•**object** is the formula which is already created using the lm() function.
•**newdata** is the vector containing the new value for predictor variable.
Predict the weight of new persons

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
# Find weight of a person with height 170.
a <- data.frame(x = 170)
 result <- predict(relation,a)
print(result)
```

Visualize the Regression Graphically

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)
# Give the chart file a name.
png(file = "linearregression.png")
# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression", abline(lm(x~y)),cex = 1.3,pch = 16,xlab
= "Weight in Kg",ylab = "Height in cm")
# Save the file.
dev.off()
```



R - Multiple Regression

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.
The general mathematical equation for multiple regression is −

y = a + b1x1 + b2x2 +...bnxn

Following is the description of the parameters used −
•**y** is the response variable.
•**a, b1, b2...bn** are the coefficients.
•**x1, x2, ...xn** are the predictor variables.
We create the regression model using the **lm()** function in R. The model determines the value of the coefficients using the input data. Next we can predict the value of the response variable for a given set of predictor variables using these coefficients
lm – Linear Model

Syntax

The basic syntax for **lm()** function in multiple regression is −

lm(y ~ x1+x2+x3...,data)

Following is the description of the parameters used −

•**formula** is a symbol presenting the relation between the response variable and predictor variables.

•**data** is the vector on which the formula will be applied.

```
input <- mtcars[,c("mpg","disp","hp","wt")]
print(head(input))
```

```
input <- mtcars[,c("mpg","disp","hp","wt")]
 # Create the relationship model.
model <- lm(mpg~disp+hp+wt, data = input)
# Show the model.
print(model)
# Get the Intercept and coefficients as vector elements.
cat("# # # # The Coefficient Values # # # ","\n")
a <- coef(model)[1]
print(a)
 Xdisp <- coef(model)[2]
Xhp <- coef(model)[3]
Xwt <- coef(model)[4]
print(Xdisp)
print(Xhp)
print(Xwt)
```

Create Equation for Regression Model

Based on the above intercept and coefficient values, we create the mathematical equation.

Y = a+Xdisp.x1+Xhp.x2+Xwt.x3

or Y = 37.15+(-0.000937)*x1+(-0.0311)*x2+(-3.8008)*x3

Apply Equation for predicting New Values

We can use the regression equation created above to predict the mileage when a new set of values for displacement, horse power and weight is provided.

For a car with disp = 221, hp = 102 and wt = 2.91 the predicted mileage is −

Y = 37.15+(-0.000937)*221+(-0.0311)*102+(-3.8008)*2.91

Y = 22.7104

R - Logistic Regression

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables. Logistic regression is useful when you are predicting a binary outcome from a set of continuous predictor variables.

The general mathematical equation for logistic regression is −

y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used −

- **y** is the response variable.
- **x** is the predictor variable.
- **a** and **b** are the coefficients which are numeric constants.

The function used to create the regression model is the **glm()** function.

Glm – Generalized Linear Model

Syntax

The basic syntax for **glm()** function in logistic regression is −

glm(formula,data,family)

Following is the description of the parameters used −

- **formula** is the symbol presenting the relationship between the variables.
- **data** is the data set giving the values of these variables.
- **family** is R object to specify the details of the model. It's value is binomial for logistic regression.

```
# Select some columns from mtcars.
input <- mtcars[,c("am","cyl","hp","wt")]
print(head(input))
```

Create Regression Model

We use the **glm()** function to create the regression model and get its summary for analysis.

```
input <- mtcars[,c("am","cyl","hp","wt")]
am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
print(summary(am.data))
```

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. If a Z-score is 0, it indicates that the data point's score is identical to the mean score. A Z-score of 1.0 would indicate a value that is one standard deviation from the mean. Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean.

The "Pr(>|z|)" is the so called "p-value" of the test for whether the coefficient point estimate is significantly different from 0.

In null hypothesis significance testing, the ***p*-value**] is the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct.[2][3] A very small *p*-value means that such an extreme observed outcome would be very unlikely under the null hypothesis. Reporting *p*-values of statistical tests is common practice in academic publications of many quantitative fields. Since the precise meaning of *p*-value is hard to grasp, misuse is widespread and has been a major topic in metascience

R - Logistic Regression

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1.
It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.
The general mathematical equation for logistic regression is −
y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used −
•**y** is the response variable.
•**x** is the predictor variable.
•**a** and **b** are the coefficients which are numeric constants.
The function used to create the regression model is the **glm()** function.
Syntax
The basic syntax for **glm()** function in logistic regression is −
glm(formula,data,family)

Following is the description of the parameters used −

•**formula** is the symbol presenting the relationship between the variables.

•**data** is the data set giving the values of these variables.

•**family** is R object to specify the details of the model. It's value is binomial for logistic regression.

Example

The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1). We can create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

```
# Select some columns form mtcars.
 input <- mtcars[,c("am","cyl","hp","wt")]
print(head(input))
```

Create Regression Model
We use the **glm()** function to create the regression model and get its summary for analysis.

```
input <- mtcars[,c("am","cyl","hp","wt")]
am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
print(summary(am.data))
```

Conclusion
In the summary as the p-value in the last column is more than 0.05 for the variables "cyl" and "hp", we consider them to be insignificant in contributing to the value of the variable "am". Only weight (wt) impacts the "am" value in this regression model.

Stochastic Processes

Stochastic Process is generating random numbers by drawing from the existing distributions

- Uniform Distribution
- Normal Distribution
- Binomial Distribution

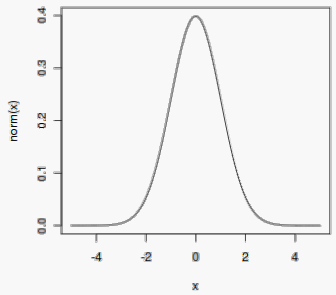
R - Normal Distribution

In a random collection of data from independent sources, it is generally observed that the distribution of data is normal. Which means, on plotting a graph with the value of the variable in the horizontal axis and the count of the values in the vertical axis we get a bell shape curve. The center of the curve represents the mean of the data set. In the graph, fifty percent of values lie to the left of the mean and the other fifty percent lie to the right of the graph. This is referred as normal distribution in statistics.

R has four in built functions to generate normal distribution. They are described below.

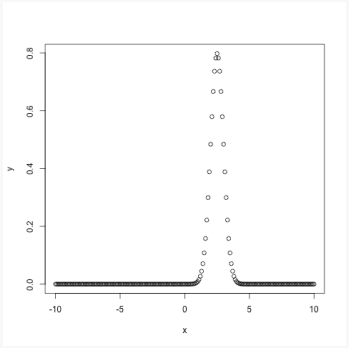
dnorm(x, mean, sd) pnorm(x, mean, sd) qnorm(p, mean, sd) rnorm(n, mean, sd) Following is the description of the parameters used in above functions −

- **x** is a vector of numbers.
- **p** is a vector of probabilities.
- **n** is number of observations(sample size).
- **mean** is the mean value of the sample data. It's default value is zero.
- **sd** is the standard deviation. It's default value is 1.



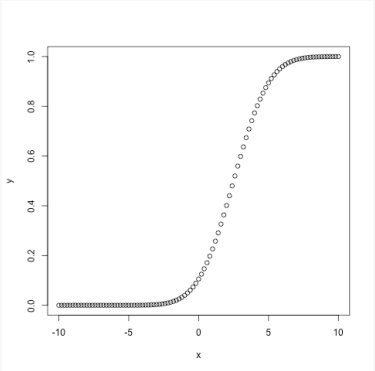
dnorm()
This function gives height of the probability distribution at each point for a given mean and standard deviation.

```
# Lets create a sequence of numbers between -10 and 10 increment by 0.1.
x <- seq(-10, 10, by = .1)
# Set the Mean as 2.5 and Standard Deviation as 0.5.
y <- dnorm(x, mean = 2.5, sd = 0.5)
# Give the chart file a name.
png(file = "dnorm.png")
plot(x,y)
# Save the file.
dev.off()
```



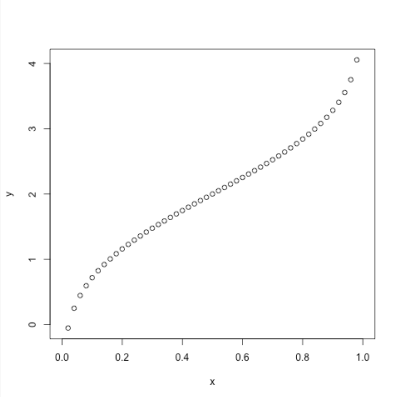
pnorm()
This function gives the probability of a normally distributed random number to be less that the value of a given number. It is also called "Cumulative Distribution Function".

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.2.
x <- seq(-10,10,by = .2)
# Choose the mean as 2.5 and standard deviation as 2.
y <- pnorm(x, mean = 2.5, sd = 2)
# Give the chart file a name.
png(file = "pnorm.png")
# Plot the graph.
plot(x,y)
# Save the file.
dev.off()
```



qnorm()
This function takes the probability value and gives a number whose cumulative value matches the probability value.

```
# Create a sequence of probability values incrementing by 0.02.
x <- seq(0, 1, by = 0.02)
# Choose the mean as 2 and standard deviation as 3.
y <- qnorm(x, mean = 2, sd = 1)
# Give the chart file a name.
png(file = "qnorm.png")
# Plot the graph.
plot(x,y)
# Save the file.
dev.off()
```



rnorm()
This function is used to generate random numbers whose distribution is normal. It takes the sample size as input and generates that many random numbers. We draw a histogram to show the distribution of the generated numbers.

```
# Create a sample of 50 numbers which are normally distributed.
y <- rnorm(50)
# Give the chart file a name.
png(file = "rnorm.png")
# Plot the histogram for this sample.
hist(y, main = "Normal Distribution")
# Save the file.
dev.off()
```

R - Binomial Distribution

The binomial distribution model deals with finding the probability of success of an event which has only two possible outcomes in a series of experiments. For example, tossing of a coin always gives a head or a tail. The probability of finding exactly 3 heads in tossing a coin repeatedly for 10 times is estimated during the binomial distribution.
R has four in-built functions to generate binomial distribution. They are described below.
dbinom(x, size, prob)
pbinom(x, size, prob)
qbinom(p, size, prob)
rbinom(n, size, prob)
Following is the description of the parameters used −
•**x** is a vector of numbers.
•**p** is a vector of probabilities.
•**n** is number of observations.
•**size** is the number of trials.
•**prob** is the probability of success of each trial.
dbinom()

dbinom()
This function gives the probability density distribution at each point.

```
# Create a sample of 50 numbers which are incremented by 1.
x <- seq(0,50,by = 1)
# Create the binomial distribution.
y <- dbinom(x,50,0.5)
# Give the chart file a name.
png(file = "dbinom.png")
# Plot the graph for this sample.
plot(x,y)
# Save the file.
dev.off()
```

pbinom()
This function gives the cumulative probability of an event. It is a single value representing the probability.

```
# Probability of getting 26 or less heads from a 51 tosses of a coin.
x <- pbinom(26,51,0.5)
print(x)
```

qbinom()
This function takes the probability value and gives a number whose cumulative value matches the probability value.

```
# How many heads will have a probability of 0.25 will come out when a coin #
is tossed 51 times.
x <- qbinom(0.25,51,1/2)
print(x)
```

rbinom()
This function generates required number of random values of given probability from a given sample.

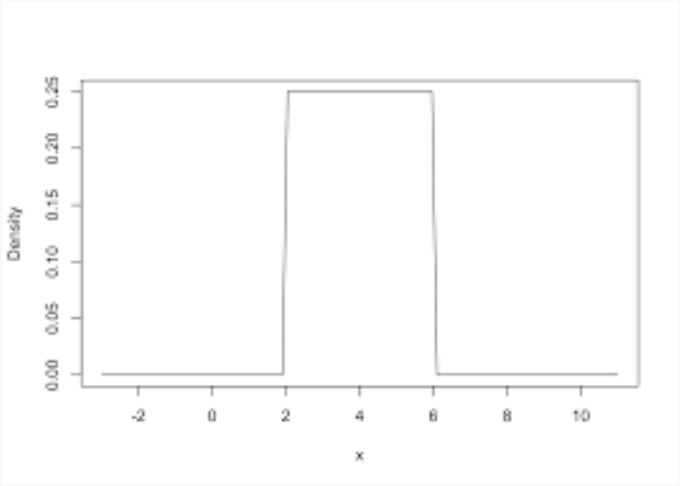
```
# Find 8 random values from a sample of 150 with probability of 0.4.
x <- rbinom(8,150,.4)
print(x)
```

Random Number generation

Draw from existing distributions:

a) Uniform Distribution

The **continuous uniform distribution** is the probability distribution of random number selection from the continuous interval between *a* and *b*



R command:

runif(n,min,max)

n= Number of times that we want to draw from the distribution

Min & Max (Range of values)

R - Poisson Regression

Poisson Regression involves regression models in which the response variable is in the form of counts and not fractional numbers. For example, the count of number of births or number of wins in a football match series. Also the values of the response variables follow a Poisson distribution.
The general mathematical equation for Poisson regression is −
log(y) = a + b1x1 + b2x2 + bnxn..... Following is the description of the parameters used −
•**y** is the response variable.
•**a** and **b** are the numeric coefficients.
•**x** is the predictor variable.
The function used to create the Poisson regression model is the **glm()** function.
Syntax
The basic syntax for **glm()** function in Poisson regression is −
glm(formula,data,family)

Following is the description of the parameters used in above functions −

•**formula** is the symbol presenting the relationship between the variables.

•**data** is the data set giving the values of these variables.

•**family** is R object to specify the details of the model. It's value is 'Poisson' for Logistic Regression.

Example

We have the in-built data set "warpbreaks" which describes the effect of wool type (A or B) and tension (low, medium or high) on the number of warp breaks per loom. Let's consider "breaks" as the response variable which is a count of number of breaks. The wool "type" and "tension" are taken as predictor variables.

input Data

```
input <- warpbreaks
print(head(input))
```

Create Regression Model

```
output <-glm(formula = breaks ~ wool+tension, data = warpbreaks, family = poisson)
print(summary(output))
```

Logistics Regression Usage:
Logistic regression is used when your outcome variable (dependent variable) is mutually exclusive(dichotomous).Basically it would refer to the research type of questions: Yes or No. In other words, it can also be pass or fail, hit or miss, success or failure, alive or dead etc. Logistic regression estimates the probability of getting one of your two outcomes (i.e., the probability of voting vs. not voting) given a predictor/independent variable(s).

Poisson regression Usage:
Poisson regression AKA log-linear model, is used when your outcome variable is a count (i.e., numeric). However it doesn't have to be a continuous variable. Examples :how many heart attacks ones had, or, as in survival analysis, how many days from outbreak until infection. The Poisson distribution is unique in its nature that its mean and variance are equal. Using our count variables from above, this could be a sample that contains individuals with and without heart disease: those without heart disease cause a disproportionate amount of zeros in the data and those with heart disease trail off in a tail to the right with increasing amounts of heart attacks. This is why logistic and Poisson regressions go together in research: there is a dichotomous outcome inherent in a Poisson distribution. However, the "hits" in the logistic question can't be understood without further conducting the Poisson regression.

R - Analysis of Covariance

We use Regression analysis to create models which describe the effect of variation in predictor variables on the response variable. Sometimes, if we have a categorical variable with values like Yes/No or Male/Female etc. The simple regression analysis gives multiple results for each value of the categorical variable. In such scenario, we can study the effect of the categorical variable by using it along with the predictor variable and comparing the regression lines for each level of the categorical variable. Such an analysis is termed as **Analysis of Covariance** also called as **ANCOVA**.

Example

Consider the R built in data set mtcars. In it we observe that the field "am" represents the type of transmission (auto or manual). It is a categorical variable with values 0 and 1. The miles per gallon value(mpg) of a car can also depend on it besides the value of horse power("hp").

We study the effect of the value of "am" on the regression between "mpg" and "hp". It is done by using the **aov()** function followed by the **anova()** function to compare the multiple regressions.

The **ANOVA** test (or **Analysis of Variance**) is used to compare the mean of multiple groups. The name ANOVA refers to variances and the main goal of ANOVA is to investigate differences in means.

There are different types of ANOVA for comparing independent groups:
•**One-way ANOVA**:. The simplest case of ANOVA test where the data are organized into several groups according to only one single grouping variable (also called factor variable). Other synonyms are: *1 way ANOVA*, *one-factor ANOVA* and *between-subject ANOVA*.

•**two-way ANOVA** can be used to evaluate simultaneously the effect of two different grouping variables on a continuous outcome variable. Other synonyms are: *two factorial design*, *factorial anova* or *two-way between-subjects ANOVA*.
•**three-way ANOVA** can be used to evaluate simultaneously the effect of three different grouping variables on a continuous outcome variable. Other synonyms are: *factorial ANOVA* or *three-way between-subjects ANOVA*.

•Types of Anova:
•http://md.psych.bio.uni-goettingen.de/mv/unit/lm_cat/lm_cat_unbal_ss_explained.html

Type I (sequential or incremental SS)

Type I sums of squares are determined by considering each source (factor) sequentially, in the order they are listed in the model. The Type I SS may not be particularly useful for analyses of unbalanced, multi-way structures but may be useful for balanced data and nested models. Type I SS are also useful for parsimonious polynomial models (i.e. regressions), allowing the simpler components (e.g. linear) to explain as much variation as possible before resorting to models of higher complexity (e.g. quadratic, cubic, etc.). Also, comparing Type I and other types of sums of squares provides some information regarding the magnitude of imbalance in the data. Types II and III SS are also know as partial sums of squares, in which each effect is adjusted for other effects.

Type III

Type III is also a partial SS approach, but it's a little easier to explain than Type II; so we'll start here. In this model, every effect is adjusted for all other effects. The Type III SS will produce the same SS as a Type I SS for a data set in which the missing data are replaced by the leastsquares estimates of the values. The Type III SS correspond to Yates' weighted squares of means analysis. One use of this SS is in situations that require a comparison of main effects even in the presence of interactions (something the Type II SS does not do and something, incidentally, that many statisticians say should not be done anyway!). In particular, the main effects A and B are adjusted for the interaction A*B, as long as all these terms are in the model. If the model contains only main effects, then you will find that the Type II and Type III analyses are the same.

Type II

Type II partial SS are a little more difficult to understand. Generally, the Type II SS for an effect U, which may be a main effect or interaction, is adjusted for an effect V if and only if V does not contain U. Specifically, for a two-factor structure with interaction, the main effects A and B are not adjusted for the A*B interaction because the interaction contains both A and B. Factor A is adjusted for B because the symbol B does not contain A. Similarly, B is adjusted for A. Finally, the A*B interaction is adjusted for each of the two main effects because neither main effect contains both A and B. Put another way, the Type II SS are adjusted for all factors that do not contain the complete set of letters in the effect. In some ways, you could think of it as a sequential, partial SS; in that it allows lower-order terms explain as much variation as possible, adjusting for one another, before letting higher-order terms take a crack at it.

Type IV

The Type IV functions were designed primarily for situations where there are empty cells, also known as "radical" data loss. The principles underlying the Type IV sums of squares are quite involved and can be discussed only in a framework using the general construction of estimable functions. It should be noted that the Type IV functions are not necessarily unique when there are empty cells but are identical to those provided by Type III when there are no empty cells.

Input Data

Create a data frame containing the fields "mpg", "hp" and "am" from the data set mtcars. Here we take "mpg" as the response variable, "hp" as the predictor variable and "am" as the categorical variable.

```
input <- mtcars[,c("am","mpg","hp")]
print(head(input))
```

ANCOVA Analysis

We create a regression model taking "hp" as the predictor variable and "mpg" as the response variable taking into account the interaction between "am" and "hp".

Model with interaction between categorical variable and predictor variable

```
# Get the dataset.
input <- mtcars
 # Create the regression model.
result <- aov(mpg~hp*am,data = input)
print(summary(result))
```

This result shows that both horse power and transmission type has significant effect on miles per gallon as the p value in both cases is less than 0.05. But the interaction between these two variables is not significant as the p-value is more than 0.05.

Model without interaction between categorical variable and predictor variable

```
# Get the dataset.
input <- mtcars
# Create the regression model.
result <- aov(mpg~hp+am,data = input)
print(summary(result))
```

This result shows that both horse power and transmission type has significant effect on miles per gallon as the p value in both cases is less than 0.05.

Comparing Two Models

Now we can compare the two models to conclude if the interaction of the variables is truly in-significant. For this we use the **anova()** function.

```
# Get the dataset.
input <- mtcars
# Create the regression models.
result1 <- aov(mpg~hp*am,data = input)
result2 <- aov(mpg~hp+am,data = input)
 # Compare the two models.
print(anova(result1,result2))
```

Ref.Df -> Residual Degrees of Freedom
RSS.Df -> Residual Sum of Squares

As the p-value is greater than 0.05 we conclude that the interaction between horse power and transmission type is not significant. So the mileage per gallon will depend in a similar manner on the horse power of the car in both auto and manual transmission mode.

Analysis of Co-Variance:
IRIS Example:

```
df = iris

#Hypothesis : If the species of iris has any impact on other features of the flower
#Use case : If the species of iris has any impact on the petal length
#Run Levene's Test.
#What is Leven's test:
 #Levene's test is an inferential statistic used to assess the equality of variances for a variable calculated
 #for two or more groups.
#Some common statistical procedures assume that variances of the populations
#from which different samples are drawn are equal.
#Levene's test is not available in R. To run that, install "car" package.
#"car" -   Companion to Applied Regression

install.packages("car")
library(car)
leveneTest(Petal.Length~Species,df)
```

If the returned results are significant, proceed with the Anova.

```
#The below command will provide details if the Species impact the Petal Length.
fit = aov(Petal.Length ~ Species, df)

summary(fit)

# The above table provides more information. However, the focus is here on the the row Species, as
#this contains the information for the independent variable we specified, and the columns F-
#value and Pr(>F). In this scenario, the null hypothesis is that the species of the Iris don't have any
#impact on the petal length. We are looking for High F values & low p values. Here the F value is 1800
#and the p value is 0.0000000000000002 ( 2e-16). This proves that the Iris species has an impact on
the #Petal.Length.
#Now lets report the finding of ANOVA. The describeBy function of pysch package is used for this
detailed reporting.
install.packages("psych")
library(psych)
describeBy(df$Petal.Length, df$Species)
```

```
#Lets plot the output
library(ggplot2)
ggplot(df,aes(y=Petal.Length, x=Species, fill=Species))+
 stat_summary(fun ="mean", geom="bar",position="dodge")+
stat_summary(fun.data = mean_se, geom = "errorbar", position="dodge",width=.8)

#Now , lets perform the same analysis using the other features.
  fit2=aov(Petal.Length~Species+Sepal.Length,df)

Anova(fit2, type="III")
```

In row *Species,* column *Pr(>F),* which is the p-value, species still has a significant impact on the length of the petal, even when controlling for the length of the sepal.

Finding:

The covariate, sepal length, was significantly related to the flowers' petal length, F=194.95, p<.001. There was also a significant effect of the species of the plant on the petal length after controlling for the effect of the sepal length, F=624.99, p<.001.

R - Time Series Analysis

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called **time-series object**. It is also a R data object like a vector or data frame.

The time series object is created by using the **ts()** function.

Syntax

The basic syntax for **ts()** function in time series analysis is −

timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used −

- **data** is a vector or matrix containing the values used in the time series.
- **start** specifies the start time for the first observation in time series.
- **end** specifies the end time for the last observation in time series.
- **frequency** specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional.

Example

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R vector.
rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
 # Convert it to a time series object.
rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
# Print the timeseries data.
print(rainfall.timeseries)
# Give the chart file a name.
png(file = "rainfall.png")
 # Plot a graph of the time series.
 plot(rainfall.timeseries)
# Save the file.
dev.off()
```

Different Time Intervals

The value of the **frequency** parameter in the ts() function decides the time intervals at which the data points are measured. A value of 12 indicates that the time series is for 12 months. Other values and its meaning is as below −

- **frequency = 12** pegs the data points for every month of a year.
- **frequency = 4** pegs the data points for every quarter of a year.
- **frequency = 6** pegs the data points for every 10 minutes of an hour.
- **frequency = 24*6** pegs the data points for every 10 minutes of a day.

Multiple Time Series

We can plot multiple time series in one chart by combining both the series into a matrix.

```
# Get the data points in form of a R vector.
 rainfall1 <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
rainfall2 <- c(655,1306.9,1323.4,1172.2,562.2,824,822.4,1265.5,799.6,1105.6,1106.7,1337.8)
# Convert them to a matrix.
combined.rainfall <- matrix(c(rainfall1,rainfall2),nrow = 12)
# Convert it to a time series object.
rainfall.timeseries <- ts(combined.rainfall,start = c(2012,1),frequency = 12)
# Print the timeseries data.
print(rainfall.timeseries)
# Give the chart file a name.
png(file = "rainfall_combined.png")
 # Plot a graph of the time series.
plot(rainfall.timeseries, main = "Multiple Time Series")
# Save the file.
dev.off()
```