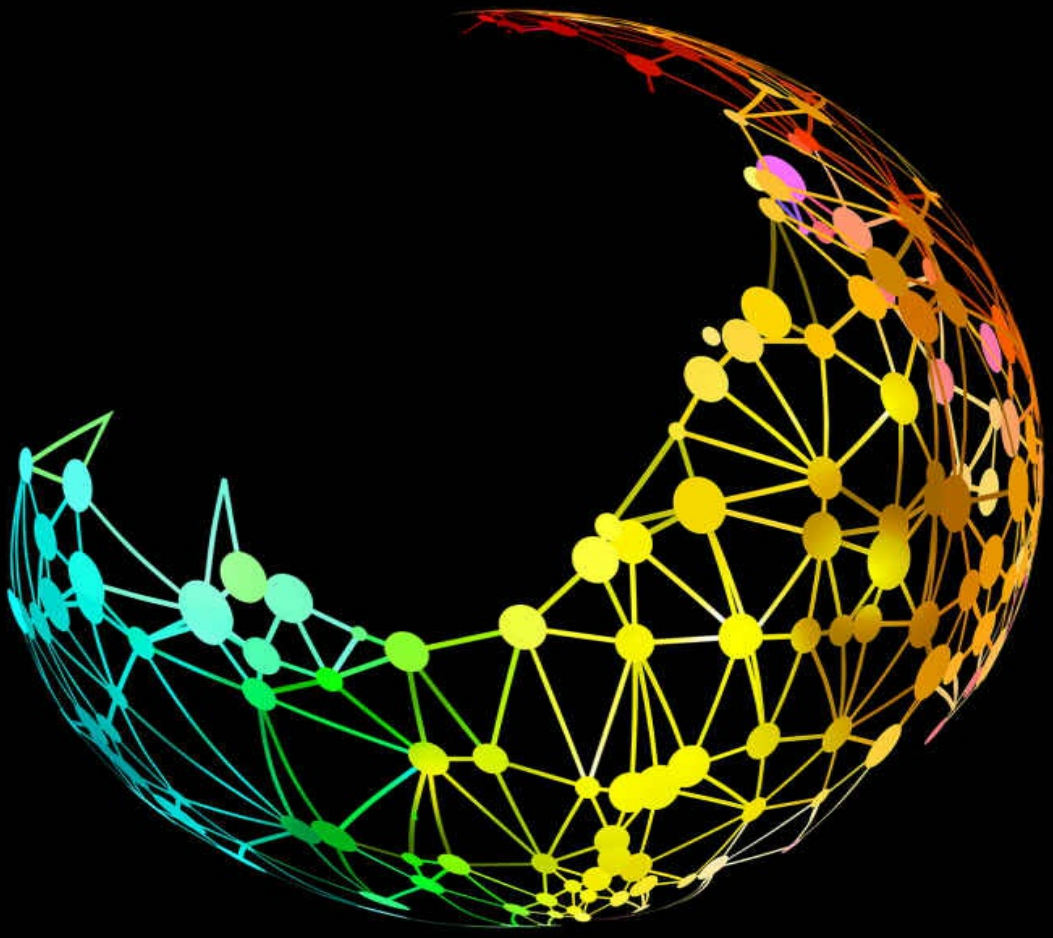
DATA SCIENCE

TIPS AND TRICKS TO LEARN DATA SCIENCE THEORIES EFFECTIVELY



WILLIAM VANCE

Data Science

Tips and Tricks to learn Data Science Theories Effectively

Contents

Introduction

There is a popular joke that a data scientist is someone who knows more computer science than a statistician and knows more statistics than a computer scientist. While it is true that to become a good data scientist, you don't need just the knowledge of computers and statistics, to become a professional data scientist, one must be exceedingly brilliant in both computer science and statistics. This is because the knowledge of both fields is what enables data scientists to be able to create insight from scattered data on the computer screen. In the same vein, the knowledge of fields outside these two helps the data scientist asks intelligent questions, both formal and informal, on data. He or she uses this to generate a clear insight.

This is why, in most cases, a data scientist is well trained in not just computer-related fields but also other fields like economics, business, law and many more. This helps the scientist to become vast in questions required in any domain he or she is working with. It also helps him or her summarize this question as required. A good example of a site that uses questions from additional domain aside computer and statistics is the dating site OkCupid. Before an individual can become a member of the group, he or she would be required to answer a lot of questions. The site uses the answers given to these questions to summarize the personal characteristics of the candidate and then merge him or her with the right person available. As a result, to be able to construct these questions intelligently, the scientist is required to be vast in the field of psychology and humanity.

Similarly, Facebook, one of the most popular social media, like the dating site mentioned above, also uses some personal questions. The reason for this is to make it easy for families and friends to find and connect with the person.

In relation to this, in 2012, during Obama campaign in the US, he employed a lot of data scientist to use their skill in finding out voters the parts of the country that need extra . This birthed the different fundraising appeals and programs that were put in place before the election. This singular effort played an important role in the president's reelection and shows that at each passing year, data scientists are becoming more needed and valuable.

The growing rate of both social media and the internet, as the basic channels, to get any information on any field imaginable, has generated more data than we can comprehend. At every passing, users upload videos on YouTube. Not only this, the rate at which people are using Facebook and Twitter grows at each passing day. Business is drawn to the digital world. Digital marketing is fast becoming entrenched — all this is more data than we can imagine.

These brief explanations show that the domain of data science is vast and required continuous training not just in the field of the computer but also in other fields where data scientists would be needed. Only a few scientists are actually armed with the required capacity. In a business organization, for instance, a data scientist is required to help generate applicable business intelligence.

However, data is not information, and until an analysis is added to it, it is just noise. This is where this book comes in. It explains the different theories and models, applications and calculations required in data science. The books explain the different aspects of data analysis and science.

Added to this is the indisputable fact that the world is fast drowning in data consumption. Every click on a website is being tracked and monitored by data science; every smartphone is built with a capacity to deliver your location at every time it is needed accurately. Quantified sellers are always on pedometers-on-steroid use to records their movement, heart rate, diet, habit, and even their sleeping method. The internet represents the happening place. It contains information ranging from the database, encyclopedia, domain-specific details on music, sport and any event of your choice. Not only this, on the internet, is information such as government statistics, academic write-ups, textbooks, products of different ranges, games and many more.

These data contain information that seems elusive to everyone. This information is what would be explained in this book. Data science has indeed risen more than we often thought. It is fast becoming the most sorted after the field in computer studies. The world is beginning to recognize the impact of the data scientist. This book focuses on explaining the rudimentary aspects of data science. It explains in detail the important theories and models used in data science. If the focus on thoroughly explaining how theories are applied in data science and how these theories can be used to explain some of the basic aspects of data science.

The book is divided into fifteen chapters, and each chapter focuses on explaining thoroughly and indelibly all there is to know about data science. The first few chapters explain in detail what data science is and the different aspects of data science. The other chapters concentrate on the various theories and models used in data science.

At the end of the book, the learner is expected to be knowledgeable in the use of theories and models in data science. He or she is also expected to be able to construct intelligent questions and forecast or predict the result of the questions and its importance to the domain at large.

Chapter One: What Is Data Science?

With the rising rate of technology today, data science is transforming businesses and careers. Both medical cares and law firms use data in their domain. Business organizations are not cut out of the use of data. According to Josh Wills of Cloudera, “A data scientist is a person who is better at statistics than any software engineer and better at software engineering than any statistician.” Therefore, a data scientist is someone who is excellent at software engineering and statistics. A data scientist knows all there is about business models and paradigms.

Data science covers generating new ideas and approaches and merging new theories with new algorithms. It entails the mastery of both statistics and computer studies. Data science focuses more on data and algorithms and the various theories and models that can be used to interpret the data and algorithms. Data science encompasses the creation of an excellent data analysis plan and the implementation of that data using theories and models.

To create a good data analysis plan, McKinsey argues that it must contain these three elements: analytic model, interlinked data inputs, and decision-support tools. Similarly, Halevy, Norvig, and Pereira argued in a seminal paper published in 2009 that even simple theories and models, with big data, have the potential to do better than complex models with fewer data. This is to show that the effectiveness of data analysis is not dependent on how large the data is. The size of the data has little to play in the effectiveness of the data. One of the popular data scientist Hilary Mason, explained that the making of “data products” entails three important aspects. The first is the data itself, the second is technical excellence which encompasses machine learning and the third part is people and process; this entails talent. A good example of a data product that comprises all three elements is Google Maps. Hilary went further to list out the skill required to be an excellent data scientist. These include math and stats, communication and coding.

However, before the mastery and implementation of these skills, every individual who aims to become a good data scientist must possess one valuable ability. This is the ability to ask relevant and excellent questions. The answer to these questions is what unlock values for consumers, companies, and even society at large. Therefore, becoming a good data scientist begins with the ability to ask relevant questions and to solve a problem. Becoming a good data science requires the mastery of not just computer or statistics, a data scientist is knowledgeable in other fields of study.

Two Broad Aspects of Data Science

Predictions and Forecasts

These two aspects play a very important role in the major purpose of data science. However, there is a difference between how these two works. Forecasts cover a range of outcomes, while prediction focuses on identifying just one outcome. Take, for instance, the statement "it will snow this week" is a prediction. However, when we say there is a 50% probability that it would snow this week. This means that there is also a 50% probability that it would not snow. Prediction portrays certainty while the forecast gives a range of possibilities.

The Four V's Of Data Science

The big data is made up of several V's however, for the purpose of this book, these four will be our concentration.

- Volume
- Velocity
- Variety
- Veracity

Volume

The volume of big data is fast exceeding the capacity of most databases. Data generation scale has risen to the extent that what used to be the record generated in years is now being generated in two days. This is confirmed by a very popular data analyst, Google's Eric Schmid. He pointed out that as of 2003, what was generated in the volume of big data is 5 exabytes of data - an exabyte is 1000x6 bytes or a billion billions of bytes. Today, what was generated in 2003 is generated in just two days.

The reason for this massive rise in data volume is simply the introduction of interaction data. This is the new kind of data that produces more results than the transaction data that was used before its development. Interaction data entails the recording of daily activities in apps, which may include activities in a browser, RFID data, geo-location data, and so on. All these activities gather more than millions of thousands of data in just a day. Truth is we are in the age of “internet of things” (or IoT); every little thing concerning us is made available on the internet or social media and this is generating constant significant growth in the quantity of data.

As such, a good data scientist is expected to be very skilled at managing data volume. This is not limited to the technical database alone but also the ability to build algorithms that help manage the size of the data intelligently. This is partly because, when dealing with big data, none of the correlations is left out, all of them are very important. Hence, to be able to manage this effectively, a good data scientist should be vast in techniques that would help extract causality from correlations.

Velocity

As the volume of data increases, the velocity also increases. Facebook entries increase every single second; tweets are growing on Twitter; at every second, information is seemed and generated by users. The rising increase of velocity also affects the volume of data. However, this might affect the window of data application.

Variety

The variety of data has grown higher than before. Before now, models that work with variables are very minimum. However, these have increased as computing power increases. The increase and change in velocity, volume, and variety of data required new econometrics and a couple of new tools for setting questions in data science.

As we progress in the book, we explained the different econometric techniques and modeling concepts that would be given.

However, it is important to note that data science is not limited to the analysis of large data; it also requires the creation of data.

Machine Learning

Machine learning is just an aspect of data science. This implies that data science entails more than just machine learning. Systems are usually trained on data to make decisions. However, this is a continuous process. As the system is continuously trained on decision making, the capacity of the system improves with more data. A good example of a system trained in decision making is the spam filter. As the data grows, the spam filter uses a Bayesian filter to change its decision making. This happens constantly and helps keep the filter ahead of spammers. It is this ability to stay ahead of spammers that them from gaming the filter.

However, machine-learning techniques favor data over judgment. Good data science is required to combine both aspects excellently. Machine learning is fast progressing; Hilary Mason highlighted four characteristics of machine learning. The characteristic is listed below and would be explained in detail in subsequent chapters.

1. Machine learning is usually based on a theoretical breakthrough and is therefore well-grounded in science.
1. Machine learning changes the existing economic paradigm.
2. The result is commoditization (e.g., Hadoop)
3. it makes available new data that leads to further data science.

Supervised and Unsupervised Learning

In explaining machine learning, it was mentioned that systems are trained in decision making, and this training is a continuous process. There are two majors ways a system can be trained. These include supervised and unsupervised learning.

In supervised learning, the output is produced based on the input data. To do this, the system is provided with historical data of input and output. The system then uses one of the several machine techniques to learn the relationship between the two. Judgment would also be used to decide which of the machine learning techniques would be appropriate. Examples of supervised learning include automated credit card approval and spam filter.

In unsupervised learning, inputs are arranged and reorganized in order to structure unlabeled data. This is done by reorganizing data and labeling them with tags. A good example is factor analysis

Privacy

As the volume, velocity, and variety of data increase, individual privacy is flooded. As humans, we are often torn between engaging in social interaction with others and maintaining our privacy, and technology has made it happen in such a way that the war against maintaining our privacy grows every blessed day. Daily we are in a struggle to keep our private information from leaking out to the public. The reason for this is simply as a result of the involvement of data science. Stakeholders, governments, business owners, and more useful data service to gain access to people's private information. We are very conversant with sites asking personal questions ranging from our age, marital status, place of origin, occupation and even address to our resident areas. These kinds of questions help leak out private information.

Additionally, the loss of privacy is also caused by what is known as "human profiling." This is a situation whereby the more we move our daily activities to the web, the more companies and organizations use data mining and analysis to construct profiles of who we are even more than we often realize. For instance, when we tweet "taking my dog for a walk," this information is incremented through data analysis as "owner of a pet" when we tweet " going home to cook for my kids," this is incremented as "a mother." This shows that a little information that appears like an ordinary tweet to us on Facebook or Twitter reveals more than we often know. In succinct, a machine knows you better than you think.

Aside from tweets on social media, phone calls, GPS location tracker, and emails are also part of what companies and organizations can use to create human profiling. Those who don't often tweet information about themselves are being recorded as people of low digital profile. However, to create a balance, it is advisable not to hide so as not to be known but to maintain an average profile as possible.

Human profiling means the separation of a targeted space for a separate audience or group of people. This allows attention to be paid to that group of persons through what is known as price discrimination. For instance, if my profile shows that I am an influential person and very rich, I am likely to start receiving internet sales pitches from popular companies who have gathered that I often buy the product they are into. Profiling allows companies or business organization to reach their audience faster and more accurately.

Profiling is also used to snare terrorists, however, care should be taken not to engage in excessive profiling. We are in an age where there is an interesting battle between man and machine for their privacy. Let's be careful what information we disclose on social media.

Theories, Models, Intuition, Causality, Prediction, Correlation

Data science entails the implementation of theories and models. Data science also makes use of intuition, causality, prediction, and correlation. Theories are a statement about how the world should be or should not be. These statements are often derived from axioms that are assumed on the nature of the world or from existing theories. Models, however, are the implementation of theories. This is often achieved through the use of algorithms and variables. Intuition is the result of a running model. This means intuition is a profound understanding of the world with the aid of data, theories, and models.

Once the intuition for the result of a model is established, what is left is to determine if the relationship observed between model and intuition is that of prediction, causality, or correlation. Causality is usually stated in a mathematical form or structure. Theories might be causal. To arrive at a causal effect, it must be deeply entrenched in data. This is why causality is very difficult to establish, even with the use of theoretical foundations.

At the end of the inference chain in data science, the movement between two variables is often determined by correlation. Correlation is of utmost importance to firms hoping to tease out information from big data. Although correlation deals with a linear relationship between variables, it could also lay the background for finding out nonlinear relationship, an aspect which is becoming more and more flexible with the use of data.

In data science, a relationship is a multifaceted correlation among people. Social media, like Twitter, Facebook, Instagram, and so on, use graph theory to datafy human relationships. The aim of this is to understand how people relate to each other to make some profit from it. Data science encompasses the understanding of how humans relate with one another and to understand the behavior of a human generally. This aspect is the focus of social science.

Conclusion

This chapter explained in detail who a data scientist is, what is data science and the features a good data science should have. The chapter also looked at the characteristics of good data, machine learning, and the two major types of machine learning. In the subsequent chapters of this book, we will consider theories, models, data application and techniques. We will also explore some of the recent technologies created for big data and data science.

Chapter Two: Getting Started with Data Science

This chapter explores some of the mathematical models, statistics, and algebra of data science. We would be looking at some equations prevalent in data analysis and how business organizations use this in carrying out their duties.

Data analysis calls for technical expertise and excellence. It calls for the ability to use various quantitative tools. These tools range from statistics to calculus and algebra, and of course, econometrics. There are various tools used in data analysis; in this chapter, some of the tools would be explained in detail. The outline that would be covered in this chapter include:

- Exponentials, Logarithms, and Compounding
- Normal distribution
- Poisson distribution
- Vector Algebra
- Matrix calculus
- Diversification

Exponentials, Logarithms, and Compounding

In this section, we would start our explanation from the most basic mathematical constant we are familiar with. This is “e = 2.718281828...”, which is also the function “exp(·).” This function is usually written as "ex." Here, x can either be a real or complex variable. This type of mathematical constant is very popular in finance and other related areas. In finance, we use the constant in the continuous compounding and discounting of money at a stipulated interest rate which is (r) and a time frame (t).

Let's assume y = ex, any change in the value of x would also result in a change in the percentage of y. The reason for this is simple. In (y) =x, In (.) is the inverse function of the exponential function and also a natural logistic function.

Remember that the first derivative of this function is the equation dy/dx =ex. e is a constant, and it is defined as the limit of a particular function

The limit of a successively shorter interval over discrete compounding is what is known as **exponential compounding**. Let's assume that time frame (t) is split into intervals per year. The equation for the compounding of a dollar from zero time frame to a time (t) years at a given interval (n) at per annum rate can be written as:

$$\left(1+\frac{r}{n}\right)^{nt}.$$

When the limit of n rises to infinity, it leads to continuous compounding. The equation is written like this:

$$\lim_{n\to\infty}\left(1+\frac{r}{n}\right)^{nt}=\lim_{n\to\infty}\left[\left(1+\frac{1}{n/r}\right)^{n/r}\right]^{tr}=e^{rt}$$

The above equation is just the forward value for one dollar. To calculate the present value, we do a reverse equation. Hence the price today of a dollar collected t years from today is P=e-rt. What we got now is a bond. The yield of this bond is:

$$r=-\frac{1}{t}\ln(P)$$

Duration is the negative of the percentage price sensitivity of a bond to changes in interest rate. The equation represents this:

$$-\frac{dP}{dr}\frac{1}{P}=-\left(-te^{-rt}\frac{1}{P}\right)=tP\frac{1}{P}=t.$$

The percentage price sensitivity of a bond in relation to its second derivative is its convexity. The equation for this is:

$$\frac{d^2P}{dr^2}\frac{1}{P} = t^2P\frac{1}{P} = t^2.$$

Normal Distribution

This aspect is the benchmark of many models in social science. This is because it is widely believed to produce virtually all the data needed in the big data. It is quite interesting that most phenomenon in the real world is "power law" distributed. This implies a very few observation of high value as against many observation of low value. In this type of distribution, the probability distribution does not have the features of normal distribution. Rather what we have here is left to right decline in a probability distribution.

A good example of data distributed in this format is income distribution (very few observations of high income and many observations of low income). Other examples include the population of cities, frequencies in language, and so on.

The normal distribution is very important in statistics. A good example of this type of distribution is human heights and stock returns. In a normal distribution, if x is normally distributed with the mean and variance, the probability density for x equals to

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]$$

The distribution function gives the cumulative probability

$$F(x) = \int_{-\infty}^{\cdot} f(u)du$$

and

$$F(x) = 1 - F(-x)$$

The notation N(·) or Φ(·) instead of F(·) is often used because the normal distribution is symmetric. The “standard normal” distribution is: x ~ N(0,1).

Poisson Distribution

This is also known as a rare-event distribution. The density function for this type of distribution is

$$f(n;\lambda) = \frac{e^{-\lambda}\lambda^n}{n!}$$

In this type of distribution, there is only one parameter, the mean **λ.** The function of density is above the discrete values of **n.** Both the variance and the mean in Poisson distribution is represented by **λ**. The Poisson is a discrete-support distribution, its values range from n = (0,1,2,3,4,5....)

$$E(x) = \int xf(x)dx$$

Moment of Continuous Random Variables

The formulas that would be reviewed in this sector are very necessary because any analysis of data entails the use of these formulas. In our review, we would use the random variable **x** and the probability density function **f(x)** to arrive at the first four moments.

Mean (first moment or average) =

$$E(x) = \int xf(x)dx$$

The power of the variable results in a higher **nth** order moment. These types of moments are non-central. The formula for this is

The next central moment is the variance. Moments of the demeaned variable is also known as central moments.

Variance = Var(x) = E[x−E(x)]2 = E(x2)−[E(x)]2

The square root of the variance is the standard deviation, i.e., σ = √Var(x). The next moment is skewness

The value of skewness is in relation to the degree of asymmetric in probability density. If there is more occurrence of values in the left-hand side than the right, the distribution is left-skewed. When the values fall more on the right-hand side, what we have is right-skewed.

The last normalized central distribution is the kurtosis.

The standard distribution value for Kurtosis is 3. Excess kurtosis occurs when the standard distribution value is minus 3. A distribution with excess kurtosis is called leptokurtic.

How to Combine Random Variables

Here are the simple formats to combine random variables

1. Means are scalable and addictive, E(ax +by) = aE(x)+bE(y)
2. When **a, b** are scaler values and **x, y** are addictive, the variance of random plus scaled variables is
   Var(ax +by) = a2Var(x)+b2Var(y)+2abCov(x,y)
3. The equation for the covariance and correlation between two random variables is

$$\begin{aligned} Cov(x,y) &= E(xy) - E(x)E(y) \\ Corr(x,y) &= \frac{Cov(x,y)}{\sqrt{Var(x)Var(y)}} \end{aligned}$$

Vector Algebra

In most of the models we will explore in this book, what we will be using are linear algebra and vector calculus. Linear algebra encompasses the use of both vector and matrices, while vector algebra and calculus are very effective in handling issues, including solutions in spaces of several variables.

A good example is a high dimension. In this book, the use of vector calculus would be examined in the context of a stock portfolio. The return of each stock is defined as:

$$\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ : \\ : \\ 1 \end{pmatrix}$$

What we have in the above equation is a random vector. This is because each return comes from its own distribution. Also, there is a correlation in the return of all these stocks.

We can also define a Unit vector as :

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ : \\ : \\ R_N \end{pmatrix}$$

The unit vector would be used further in subsequent chapters, especially for analysis. A set of portfolio weights represents a portfolio vector. This implies the fraction of the portfolio invested in each stock.

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ : \\ : \\ w_N \end{pmatrix}$$

The sum of all portfolios must be 1. The equation for this is:

$$\sum_{i=1}^{N} w_i = 1, \quad \mathbf{w}'\mathbf{1} = 1$$

A good observation of the line above would show that there are two ways to calculate the sum of a portfolio. The first way is by summation notation, while the second used a simple verbal algebraic statement. Vectors are represented by the two elements at the left-hand side of the equation while the elements at the right-hand side is a scalar.

Vector notation can also be used to compute statistics and the quantities of portfolios. The formula for the portfolio return is

$$R_p = \sum_{i=1}^{N} w_i R_i = \mathbf{w}'\mathbf{R}$$

In the above equation, the quantities at the left-hand side represent the scalar, while the right-hand side is the vector.

Diversification

Here we would examine the power of using vector algebra with an application. To explore how diversification works, we would be using vector and summation math. When the number of non-perfected correlations in a stock portfolio increases, diversification happens. This creates a reduction in portfolio variance. Now, to compute the variance, we would use the portfolio weight **w** and the covariance vector of stock return **R.** Σ represents this. In our calculation, the formula for a portfolio return variance would first be written as:

$$Var(w'R) = w'\Sigma w = \sum_{i=1}^{n} w_i^2\sigma_i^2 + \sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n} w_i w_j \sigma_{ij}$$

However, if the return is independent, the formula collapses to

$$Var(\mathbf{w}'\mathbf{R}) = \mathbf{w}'\mathbf{\Sigma}\mathbf{w} = \sum_{\mathbf{i=1}}^{\mathbf{n}} w_i^2\sigma_i^2$$

But if an equal amount is invested in each asset and return are independent, the formula we have is

$$\begin{aligned} Var(\mathbf{w}'\mathbf{R}) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\sigma_i^2}{n} + \frac{n-1}{n}\sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n}\frac{\sigma_{ij}}{n(n-1)} \\ &= \frac{1}{n}\bar{\sigma_i}^2 + \frac{n-1}{n}\bar{\sigma_{ij}} \\ &= \frac{1}{n}\bar{\sigma_i}^2 + \left(1-\frac{1}{n}\right)\bar{\sigma_{ij}} \end{aligned}$$

In the above equation, the first term is the average variance, while the second term is the average covariance. What we have at the end of our equation is an outstanding result of a diversified portfolio. In this type of portfolio, the variance of stock does not play any role in portfolio risk. The variance of the stock is the average of off-diagonal terms in the covariance matrix or vector.

Matrix Calculus

Matrix calculation is merely the function of countless variables. Just as a function can be amended in multivariable calculus, functions are also amendable in matrix calculus. However, the simplest among this is using vector and matrix. Here we can take the derivative of in just a single step. For instance, let's assume

$$\mathbf{B} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

and

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

The fraction for **f(w)** will be **wB**. What we have here is a function of two variables **w1,w2.** When we write out **f(w)** in long-form, what we will arrive at is **3w1 +4w2**. The derivative of f(w) in relation to w1 is ∂f/∂w1 = 3, while the derivative of f(w) in relation to w2 is ∂f/∂w2 = 4. When we compare this with vector **B,** the value for **df/dw** is B.

The insight in this form of calculation is that when vectors are treated as regular scalars and calculus are conducted accordingly, the result we will arrive at is a vector derivative.

Conclusion

In this chapter, we have successfully covered some models in data calculation. We have also explored some of the basic statistics in data science. We considered mathematical calculations like vector, matrix, calculus, variables, and so on. The next chapter will examine more about theories and models in data science.

Chapter Three: R - Statistic Packages

This chapter would examine some of the useful steps for using R - statistic packages. For a great user interface that comes with using the R package, it is advisable to download and install RStudio; this can be done by visiting www.rstudio.com. However, it is necessary first to install R from the R project page, www.r-project.org. Now let's get started with some R statistics basic programming skills. The outlines that would be covered in this chapter include:

- System command
- Matrix
- Descriptive statistics
- High-ordered moments
- Brownian motion in R
- GARCH/ARCH Model
- Heteroskedasticity
- Regression model

System Command

To access the system directly, you can issue system command using the following technique:

system("<command>")

For example

system(" ls_lt_ |_ grep_Das")

will list out all the directory entries that have my name in chronological order. However, this kind of command would not work on a Windows machine because I am using a UNIX command. This can only work with a Linux box or Mac.

Loading data

To get started with this, there is a need to get some data. Here are the steps to do this:

1. Go to Yahoo
2. Download and save some historical data into Excel spreadsheet
3. Restructure the order of the data chronologically
4. Save the work as a CSV file
5. Use the following method to read the file into R

```
> data = read.csv("goog.csv",header=TRUE)      #Read in the data
> n = dim(data)[1]
> n
[1] 1671
> data = data[n:1,]
```

If required, the last command in the above would reverse the structure of the data sequence. Stock can be download using the quantmod package.

Note: the drop-down menu on Window and Mac can be used to install a package. In Linux, use the package installer. The following command can also be used to achieve this.

install. packages("quantmod")

Now we can start using the package

```
> library(quantmod)
Loading required package: xts
Loading required package: zoo
> getSymbols(c("YHOO","AAPL","CSCO","IBM"))
[1] "YHOO" "AAPL" "CSCO" "IBM"
> yhoo = YHOO['2007-01-03::2015-01-07']
> aapl = AAPL['2007-01-03::2015-01-07']
> csco = CSCO['2007-01-03::2015-01-07']
> ibm = IBM['2007-01-03::2015-01-07']
```

We can also create a direct column of stock using the following formula:

```
> yhoo = as.matrix(YHOO[,6])
> aapl = as.matrix(AAPL[,6])
> csco = as.matrix(CSCO[,6])
> ibm = as.matrix(IBM[,6])
```

Next, concatenate the data columns into a single stock data set

```
> stkdata = cbind(yhoo,aapl,csco,ibm)
> dim(stkdata)
[1] 2018    4
```

Now we will log in return in continuous-time to compute daily returns. The mean returns include:

```
> n = length(stkdata[,1])
> n
[1] 2018
> rets = log(stkdata[2:n,]/stkdata[1:(n-1),])
> colMeans(rets)
YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted  IBM.Adjusted
 3.175185e-04  1.116251e-03  4.106314e-05  3.038824e-04
```

We will also compute the correlation matrix and the covariance matrix.

```
> cv = cov(rets)
> print(cv,2)
              YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted IBM.Adjusted
YHOO.Adjusted       0.00067       0.00020       0.00022       0.00015
AAPL.Adjusted       0.00020       0.00048       0.00021       0.00015
CSCO.Adjusted       0.00022       0.00021       0.00041       0.00017
IBM.Adjusted        0.00015       0.00015       0.00017       0.00021
> cr = cor(rets)
> print(cr,4)
              YHOO.Adjusted AAPL.Adjusted CSCO.Adjusted IBM.Adjusted
YHOO.Adjusted        1.0000        0.3577        0.4170        0.3900
AAPL.Adjusted        0.3577        1.0000        0.4872        0.4867
CSCO.Adjusted        0.4170        0.4872        1.0000        0.5842
IBM.Adjusted         0.3900        0.4867        0.5842        1.0000
```

In our program, we will notice that the print layout made it easy to select several significant digits.

To make the data files easy to work within all formats, you can use the reader package. It has many tangible functions.

Matrices

In this section, we would examine the basic command needed to manipulate and create a matrix in the R project. We will be creating a 4x3 matrix with some random numbers shown below:

```
> x = matrix(rnorm(12),4,3)
> x
           [,1]       [,2]       [,3]
[1,]  0.0625034  0.9256896  2.3989183
[2,] -0.5371860 -0.7497727 -0.0857688
[3,] -1.0416409  1.6175885  3.3755593
[4,]  0.3244804  0.1228325 -1.6494255
```

Transposing the matrix, notice that the dimensions are reversed:

When transposing the matrix, we would notice the reversion in the dimensions of the number

```
> print(t(x),3)
      [,1]    [,2]  [,3]   [,4]
[1,] 0.0625 -0.5372 -1.04  0.324
[2,] 0.9257 -0.7498  1.62  0.123
[3,] 2.3989 -0.0858  3.38 -1.649
```

For easy multiplication of a matrix, the matrix to be multiplied must conform with each other. This implies that the number of rows of the matrix at the right must be equal to the number of the columns of the matrix on the left. The resultant matrix that has the sum of the computational would contain the number of columns of the matrix at the left and the rows of the matrix at the right.

Descriptive Statistics

Here, we would be using the same data to compute different descriptive statistics. The first step to do this is to

- Read a CSV data file into R file

```
> data = read.csv("goog.csv",header=TRUE)      #Read in the data
> n = dim(data)[1]
> n
[1] 1671
> data = data[n:1,]
> dim(data)
[1] 1671    7
> s = data[,7]
```

- Now we have our stock data intact, we can compute daily returns and then convert the returns into an annualized return. The result of this action is shown below:

```
> rets = log(s[2:n]/s[1:(n-1)])
> rets_annual = rets*252
> print(c(mean(rets),mean(rets_annual)))
[1] 0.001044538 0.263223585
```

- When we compute the daily and annualized return, the result is as follows:

```
> r_sd = sd(rets)
> r_sd_annual = r_sd*sqrt(252)
> print(c(r_sd,r_sd_annual))
[1] 0.02266823 0.35984704
> #What if we take the stdev of
      annualized returns?
> print(sd(rets*252))
[1] 5.712395
> #Huh?
>
> print(sd(rets*252))/252
[1] 5.712395
[1] 0.02266823
> print(sd(rets*252))/sqrt(252)
[1] 5.712395
[1] 0.3598470
```

Higher-Ordered Moments

Two major moments arise in return distribution; they are skewness and kurtosis. To show how this works, we would be using a moment library in R.

$$\text{Skewness} = E[(X-\mu)3] \div \sigma 3$$

The meaning of skewness is that one tail is fatter than the other. A fatter right(left) tail means that the skewness is positive (negative).

$$\text{Kurtosis} = E[(X-\mu)4] \div \sigma 4$$

In Kurtosis, the two tails are fatter than the normal distribution. In a normal distribution, skewness is zero, and kurtosis is 3. Excess kurtosis occurs when the value of kurtosis is minus three.

Brownian Motion in R

Stock motion law often major in Brownian Motion, especially its geometry.

dS(t) = μdS(t) = μS(t) dt+σS(t) dB(t), S(0) = S0

This kind of equation is a stochastic differential equation (SDE). The equation is an SDE because it explains the random movement of the stock (t), the coefficient of the stock (μ) and (σ). The drift of the process of the stock is determined by μ while σ determines the volatility. Brownian motion determines the randomness B(t). Unlike the deterministic differential equation, which is only a function of time, this aspect is more general. In SDE, the solution is always a random function and not a determinant function. The time interval (h) solution is as follows:

$$S(t+h) = S(t)\exp\left[\left(\mu - \frac{1}{2}\sigma^2\right)h + \sigma B(h)\right]$$

The presence of B(h) ~ N(0,h) in the solution is what gives the function its random function quality. B(h) can also be written as the random variable (h) ~ N(0,h), where ~ N(0,1). The presence of the exponential return makes the price of stock lognormal.

$$R(t+h) = \ln\left(\frac{S(t+h)}{S(t)}\right) \sim N\left[\left(\mu - \frac{1}{2}\sigma^2\right)h, \sigma^2 h\right]$$

Maximum Likelihood Estimation

In MLE Estimation, our concern is to find the parameters {μ,σ} that causes the probability of seeing the empirical sequence of returns R(t). To carry out this estimation, we would be using a probability function. Here are the steps to carry out this estimation:

- A quick overview of the normal distribution x ~ N(μ,σ2)
- Next, we do a density function

$$\text{density function: } f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]$$

$$N(x) = 1 - N(-x)$$

$$F(x) = \int_{-\infty}^{x} f(u)du$$

- The formula for a standard normal distribution is x ~ N(0,1).For the accepted normal distribution: F(0)=12.
- In the following equation, the probability density is normal.

$$f[R(t)] = \frac{1}{\sqrt{2\pi\sigma^2 h}} \cdot \exp\left[-\frac{1}{2} \cdot \frac{(R(t)-\alpha)^2}{\sigma^2 h}\right]$$

- Where α=(μ−1/2σ2)h. For periods t=1,2,... T the entire series likelihood is

$$\prod_{t=1}^{T} f[R(t)]$$

- It is very simple (computationally) now to maximize.

$$\max_{\mu,\sigma} \mathcal{L} \equiv \sum_{t=1}^{T} \ln f[R(t)]$$

- This kind of action is known as log-likelihood; it is very easy to use in the R project. The first step to do this is to generate the log-likelihood function.

```
> LL = function(params,rets) {
+ alpha = params[1]; sigsq = params[2]
+ logf = -log(sqrt(2*pi*sigsq))
            - (rets-alpha)^2/(2*sigsq)
+ LL = -sum(logf)
+ }
```

- After this, we can now go ahead and do the MLE using the NLM (non-linear minimalization) package in R. This uses a Newton-type algorithm.

GARCH/ARCH Models

GARCH represents “Generalized Auto-Regressive Conditional Heteroskedasticity." Rob Engle was the first who invented ARCH, which later earned him a Nobel Prize. This was later extended to GARCH by Tim Bollerslev. The emphasis of ARCH models is that volatility tends to cluster, i.e., volatility for period t, is auto-correlated with volatility from the period (t −1), or other preceding periods. When a time series follows a random walk, it can be a model like this:

$$r_t = \mu + e_t, \quad e_t \sim N(0, \sigma_t^2)$$

Under ARCH, the variance is always auto-correlated. So we will have:

$$\sigma_t^2 = \beta_0 + \sum_{j=1}^{p} \beta_{1j} \sigma_{t-j}^2 + \sum_{k=1}^{q} \beta_{2k} e_{t-k}^2$$

In GARCH, the stock is conditionally normal and independent. However, because of the changes in variance, it is not identically distributed.

How Bivariate Random Variables Work

Two independent random (e1,e2) ~ N(0,1) can be converted into two correlated random variables (x1, x2) with correlation ρ using the following transformation method:

$$x_1 = e_1, \qquad x_2 = \rho \cdot e_1 + \sqrt{1-\rho^2} \cdot e_2$$

This implies that we can generate 10,000 pairs of variables using the R code explained below

```
> e = matrix(rnorm(20000),10000,2)
> cor(e)
             [,1]         [,2]
[1,] 1.000000000 0.007620184
[2,] 0.007620184 1.000000000
> cor(e[,1],e[,2])
[1] 0.007620184
> rho = 0.6
> x1 = e[,1]
> x2 = rho*e[,1]+sqrt(1-rho^2)*e[,2]
```

Multivariate Random Variables

This is generated by using the Cholesky decomposition. Cholesky stands for a covariance matrix, which is a product of two matrices. Covariance can be written in decomposed form $\Sigma = L\ L$. Here, L represents a lower triangular matrix. There can also be an alternative decomposition for upper triangular, here U = L. Each component that makes up the decomposition becomes a square root of the covariance matrix.

Cholesky decomposition is very good at generating a correlated random number from a distribution with mean vector μ and covariance matrix Σ. Assuming we have a scalar random variable e ~ (0,1) we can use this to change the variable into x ~ (μ,σ2), we generate e and then set $x = \mu+\sigma e$. However, if instead of a scalar, we have a vector e = [e1,e2,...,en] T ~ (0,I) . This can be transformed into a vector of correlated random variables x = [x1,x2,...,xn] ~ (μ,Σ), by computing: **x =μ+Le**

Portfolio Computation in R

Its variance usually calculates a portfolio's risk. When there is an increase in n (the number of securities in the portfolio), this initiates a reduction in the variance. This continues to the point that it becomes the same as the average covariance of the total assets. The following result shows what happens when the variance demo through the R function.

```
> sigport = function(n,sig_i2,sig_ij) {
+ cv = matrix(sig_ij,n,n)
+ diag(cv) = sig_i2
+ w = matrix(1/n,n,1)
+ result = sqrt(t(w) %*% cv %*% w)
+ }
>
> n = seq(5,100,5)
> n
 [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85
[18]  90  95 100
> risk_n = NULL
> for (nn in n) {
+ risk_n = c(risk_n,sigport(nn,0.04,0.01))
+ }
> risk_n
 [1] 0.1264911 0.1140175 0.1095445
     0.1072381 0.1058301 0.1048809
 [7] 0.1041976 0.1036822 0.1032796
     0.1029563 0.1026911 0.1024695
[13] 0.1022817 0.1021204 0.1019804
     0.1018577 0.1017494 0.1016530
[19] 0.1015667 0.1014889
>
```

Regression

A multivariate linear regression has:

$$Y = X \cdot \beta + e$$

This is the value for $Y \in Rt \times 1$, $X \in Rt \times n$, and $\beta \in Rn \times 1$. The overall regression solution is $\beta = (XX) - 1(XY) \in Rn \times 1$.

To arrive at the above result, we minimize the sum of squared error.

It is noteworthy that this expression is a scalar.

Heteroskedasticity

In simple linear regression, it is assumed that the standard error of the residual is the same for all observations. A lot of regression suffers from this type of situation. This type of error is what is known as "heteroskedastic error." "Hetero" means "different" while "skedastic" means "dependent on type."

A heteroskedastic error can be tested by using the standard Breusch-Pagan test available in R. This is found in the lmtest package and should be loaded before running the test.

```
> ncaa = read.table("ncaa.txt",header=TRUE)
> y = as.matrix(ncaa[3])
> x = as.matrix(ncaa[4:14])
> result = lm(y~x)
> library(lmtest)
Loading required package: zoo
> bptest(result)

        studentized Breusch-Pagan test

data:  result
BP = 15.5378, df = 11, p-value = 0.1592
```

In the above test, there is a little heteroskedastic error in the standard. This is seen in the appearance of the p-value. We would correct this using the hccm function. This stands for heteroskedasticity corrected covariance matrix would be as follows:\

```
> wuns = matrix(1,64,1)
> z = cbind(wuns,x)
> b = solve(t(z) %*% z) %*% (t(z) %*% y)
> result = lm(y~x)
> library(car)
> vb = hccm(result)
> stdb = sqrt(diag(vb))
> tstats = b/stdb
> tstats
            GMS
    -2.68006069
PTS -0.38212818
```

```
REB   2.38342637
AST  -0.40848721
TO   -0.28709450
A.T   0.65632053
STL   2.13627108
BLK   0.09548606
PF   -0.68036944
FG    3.52193532
FT    2.35677255
X3P   1.23897636
```

In the above program, we use the hccm to generate a new covariance matrix **vb** of the coefficients. Next, we generated the standard error as the square root of the diagonal of the covariance matrix. With the help of these revised standard errors, we divided the coefficients by the new standard error. This helps us to recompute the t-statistics.

Auto-Regressive Model

Whenever a data is autocorrelated, that is, generates a dependence on time, not giving an account of this is tantamount to unnecessary high statistical significance. This is because when an observation is correlated with time, they are often seen as independent, thereby limiting the true number of observations.

In an efficient market, the correlation of time from one period to the other should be close to zero.

```
> n = length(rets)
> n
[1] 1670
> cor(rets[1:(n-1)],rets[2:n])
[1] 0.007215026
```

The program above is for immediate consecutive periods, referred to as the first-ordered autocorrelation. This can be examined across many staggered periods by using the R functions in the package car.

```
> library(car)
> durbin.watson(rets,max.lag=10)
 [1] 1.974723 2.016951 1.984078 1.932000
     1.950987 2.101559 1.977719 1.838635
     2.052832 1.967741
> res = lm(rets[2:n]~rets[1:(n-1)])
> durbin.watson(res,max.lag=10)
 lag  Autocorrelation D-W Statistic p-value
   1    -0.0006436855      2.001125   0.938
   2    -0.0109757002      2.018298   0.724
   3    -0.0002853870      1.996723   0.982
   4     0.0252586312      1.945238   0.276
   5     0.0188824874      1.957564   0.402
   6    -0.0555810090      2.104550   0.020
   7     0.0020507562      1.989158   0.926
   8     0.0746953706      1.843219   0.004#
   9    -0.0375308940      2.067304   0.136
  10     0.0085641680      1.974756   0.772
 Alternative hypothesis: rho[lag] != 0
```

When the DW is close to 2, there is usually no traces of autocorrelation. When the DW statistic is less than two, it is positive autocorrelation; when it is greater than 2, it is negative autocorrelation.

Vector Auto-Regression (VAR)

This is very useful for estimating systems where the variables influence each other, and there are simultaneous regression equations. Therefore in VAR, each variable in a system depends on the lagged value of itself and on other variables. To choose the number of lag values, we use the econometrician to choose the expected decay in the time-dependence of the variable in VAR.

Conclusion

In this chapter, we explored in detail the various models and features of the R package. We also examined the types of regression models in R-statistic. In the next chapter, we will examine data handling using the R package.

Chapter Four: Data Handling and Other Useful Things

This chapter would focus on some of the alternative programs in R different from what we have examined in the previous chapter. We will also constantly draw reference from the topics under the R package treated in the former chapter. Here, we would explore some of the very strong packages of R. Especially those that use **sql-like** operations on both the handling of small data and the handling of big data. The topics we would be considered in this chapter include:

- Data extraction of stocks using quantmod.
- How to use the merge function
- How to **apply** a class of functions
- Getting interest data rate from FRED
- How to handle data using lubricate
- Using the data.table package

Data Extraction Of Stocks Using Quantmod

Here we will be using the stock package treated in the previous chapter to get a few initial data.

```
library(quantmod)
tickers = c("AAPL","YHOO","IBM","CSCO","C","GSPC")
getSymbols(tickers)

[1] "AAPL" "YHOO" "IBM"  "CSCO" "C"    "GSPC"
```

When the length of each stock series is printed, you will find out that they are not the same. Our next action is to covert closing adjusted prices of every stock into separate data.frames. Here are the steps to do this:

- Construct a list of data.frames. This is important because data.frames are stored in lists

```
df = list()
j = 0
for (t in tickers) {
  j = j + 1
  a = noquote(t)
  b = data.frame(get(a)[,6])
  b$dt = row.names(b)
  df[[j]] = b
}
```

- Each data.frames should have a column. This would be used later to join the separate stock data.frames that were previously created to a single composite data.frames.

```
stock_table = df[[1]]
for (j in 2:length(df)) {
  stock_table = merge(stock_table,df[[j]],by="dt")
}
dim(stock_table)

[1] 2222    7
```

- Next, we will use a join to integrate all the stocks, adjusted closing prices into one data.frame. The aim of this is to merge; this can be done through a union join or through an intersect join. Intersect join is the default.
  - We will observe that the stock table contains the number of rolls in the stock index. This has limited observations than individual stocks. Because what we are dealing with is an intersect join, part of the rows will be dropped.
  - Plot all stocks into a single data frame with the use of ggplots 2. This is more functional than the basic plot function. However, to use ggplots 2, we would first use the basic plot function.
  - Next, the data would be converted into returns. These could be either log returns or continuously compounded returns.
  - The data.frame returns can be used to present the descriptive statistics of returns
  - Next, the correlation matrix of returns should be computed.

- After this, the correlogram for the six return series should be displayed. This would help us see the relationship between all variables in the data set.

How to Use the Merge Function

Data frames are similar to tables or spreadsheets. However, they are very much like a database. When we want to merge two data frames, it is the same as joining two databases. R program has the merge function for this.

Now, let's assume we already have a list of ticker symbols that we want to produce a well-detailed data frame from these tickers. The first thing we would do is to go through the input name of the tickers. Let's assume the tickers are in a file named tickers.csv; the diameter of the file is the sign of a colon. This would be read like this:

tickers = read. table(" tickers . csv" ,header=FALSE, sep=" : ")

We arrive at two columns of data from the line of code read in the file. The upper part of the file contains the six rows listed below:

> head(tickers)

	V1	V2
1.	NasdaqGS	ACOR
2.	NasdaqGS	AKAM
3.	NYSE	ARE
4.	NasdaqGS	AMZN
5.	NasdaqGS	AAPL
6.	NasdaqGS	AREX

The next line identified below list out the numbers of input tickers while the third line renames the data frames columns. The tickers' column is renamed "symbols" because the data frame that would be merged with it shares a similar name. This column is the index that the two data frames are joined.

```
> n = dim(tickers)[1]
> n
[1] 98
> names(tickers) = c("Exchange","Symbol")
> head(tickers)
  Exchange Symbol
1 NasdaqGS   ACOR
2 NasdaqGS   AKAM
3     NYSE    ARE
4 NasdaqGS   AMZN
5 NasdaqGS   AAPL
6 NasdaqGS   AREX
```

The next action is to read in the list of every stock on NYSE, Nasdaq, and AMEX. This is shown as follows:

```
library(quantmod)
nasdaq_names = stockSymbols(exchange="NASDAQ")
nyse_names = stockSymbols(exchange="NYSE")
amex_names = stockSymbols(exchange="AMEX")
```

The upper part of the Nasdaq would contain the following:

```
6   AAPC     Atlantic Alliance Partnership Corp.     10.16   $105.54M     2015
          Sector                          Industry Exchange
1     Health Care                Major Pharmaceuticals   NASDAQ
2  Transportation     Air Freight/Delivery Services   NASDAQ
3         Finance                     Life Insurance   NASDAQ
4      Technology                     Semiconductors   NASDAQ
5   Capital Goods  Industrial Machinery/Components   NASDAQ
6         Finance                  Business Services   NASDAQ
```

Our next action is to join all three data frames into a single data frame. Then we check the number of rows in the merged file by checking the dimensions. These two actions are shown as follows:

c**o _ names = rbind** (nyse _ **names**, nasdaq _ **names**, amex _ **names)**

>**dim (co _ names)**

[1] 6801 8

Lastly, we would join the ticker symbol file and the exchange data into one using the merge function. This would extend the ticker file to contain information in the exchange file.

```
> result = merge(tickers,co_names,by="Symbol")
> head(result)
  Symbol Exchange.x                                  Name LastSale
1   AAPL    NasdaqGS                            Apple Inc.   119.30
2   ACOR    NasdaqGS              Acorda Therapeutics, Inc.    37.40
3   AKAM    NasdaqGS              Akamai Technologies, Inc.    56.92
4   AMZN    NasdaqGS                     Amazon.com, Inc.   668.45
5    ARE        NYSE Alexandria Real Estate Equities, Inc.    91.10
6   AREX    NasdaqGS               Approach Resources Inc.     2.24
  MarketCap IPOyear            Sector
1  $665.14B    1980        Technology
2    $1.61B    2006       Health Care
3   $10.13B    1999     Miscellaneous
4  $313.34B    1997 Consumer Services
5    $6.6B      NA Consumer Services
6   $90.65M    2007            Energy
                                                 Industry Exchange.y
1                                    Computer Manufacturing     NASDAQ
2 Biotechnology: Biological Products (No Diagnostic Substances)     NASDAQ
3                                        Business Services     NASDAQ
4                                Catalog/Specialty Distribution     NASDAQ
5                                 Real Estate Investment Trusts      NYSE
6                                       Oil & Gas Production     NASDAQ
```

Now, let's assume we wish to search for the names of the CEO of all the 98 companies listed in our program. Since we don't have any available document containing the information we seek,

we can easily download. However, a site such as Google Finance Page has the information we seek. Our next auction is to write R code and use this to scrape the data on the Google Finance page one after the other. Once we extract the CEO's name, we augment the tickers' data frame using the R code.

```
library(stringr)

#READ IN THE LIST OF TICKERS
tickers = read.table("tickers.csv",header=FALSE,sep=":")
n = dim(tickers)[1]
names(tickers) = c("Exchange","Symbol")
tickers$ceo = NA

#PULL CEO NAMES FROM GOOGLE FINANCE
for (j in 1:n) {
  url = paste("https://www.google.com/finance?q=",tickers[j,2],sep="")
  text = readLines(url)
  idx = grep("Chief Executive",text)
  if (length(idx)>0) {
    tickers[j,3] = str_split(text[idx-2],">")[[1]][2]
  }
  else {
    tickers[j,3] = NA
  }
  print(tickers[j,])
}

#WRITE CEO_NAMES TO CSV
write.table(tickers,file="ceo_names.csv",
            row.names=FALSE,sep=",")
```

We would notice that the R code that augments the tickers' data frame did this with the stringr package. This helps simplify the string. When we are through with the extraction of the names, we then search the line that has the name "Chief Executive." Here is the final data frame with the name of the CEOs

```
2 NasdaqGS   AKAM       F. Thomson Leighton
3      NYSE    ARE Joel S. Marcus J.D., CPA
4 NasdaqGS   AMZN          Jeffrey P. Bezos
5 NasdaqGS   AAPL           Timothy D. Cook
6 NasdaqGS   AREX             J. Ross Craft
```

How To Use The Apply Class Of Functions

Most times, functions are expected to be applied to many cases. The parameter for the cases may be provided in a matrix, vector, or lists. This is similar to using different sets of a parameter to repeat evaluations of a function by running a loop through a set of values. In the illustration below, we use the apply function to compute the means return of all stock. The function of the data it is merged with is the first argument, the second is either 1 (by rows) or 2 (by columns) while the function being evaluated is the third.

```
apply(rets[,1:(length(tickers)-1)],2,mean)

AAPL.Adjusted YHOO.Adjusted   IBM.Adjusted CSCO.Adjusted      C.Adjusted
 1.073902e-03  1.302309e-04  2.388207e-04  6.629946e-05 -9.833602e-04
```

We will notice that the function returns the means column of the data. Not only this, the function that applies to a list is the lappy, while sappy works with vectors and matrices. The Mapply uses multiple arguments. To verify our work, we can easily use the colMeans function.

```
colMeans(rets[,1:(length(tickers)-1)])

AAPL.Adjusted YHOO.Adjusted   IBM.Adjusted CSCO.Adjusted      C.Adjusted
 1.073902e-03  1.302309e-04  2.388207e-04  6.629946e-05 -9.833602e-04
```

How To Get Interest Rate Data From FRED

FRED stands for Federal Reserve Economic Data. This is an authenticated data interest rate source. It is managed by the St. Louis Reserve Bank and warehoused at this warehouse https://research.stlouisfed.org/fred2/. Now, let's assume we want to download the data directly using the R in FRED. For us to be able to achieve this, we would write out some codes. Although before the website was changed, there was a site for this since it is so, we would easily roll in our own code in R.

```
#FUNCTION TO READ IN CSV FILES FROM FRED
#Enter SeriesID as a text string
readFRED = function(SeriesID) {
  url = paste("https://research.stlouisfed.org/fred2/series/",SeriesID,
  "/downloaddata/",SeriesID,".csv",sep="")
  data = readLines(url)
  n = length(data)
  data = data[2:n]
  n = length(data)
  df = matrix(0,n,2)    #top line is header
  for (j in 1:n) {
    tmp = strsplit(data[j],",")
    df[j,1] = tmp[[1]][1]
    df[j,2] = tmp[[1]][2]
  }
  rate = as.numeric(df[,2])
  idx = which(rate>0)
  idx = setdiff(seq(1,n),idx)
  rate[idx] = -99
  date = df[,1]
  df = data.frame(date,rate)
  names(df)[2] = SeriesID
  result = df
}
```

We will use the function above to download our data and to produce a list of economic time series. The data would be used as an index to join the individual series as a single series. Also, we download maturity interest rates (yields) beginning from the maturity of one month (DGS1MO) to thirty years (DGS30).

```
k = 0
for (id in id_list) {
  out = readFRED(id)
  if (k>0) { rates = merge(rates,out,"date",all=TRUE) }
  else { rates = out }
  k = k + 1
}

> head(rates)
```

	date	DGS1MO	DGS3MO	DGS6MO	DGS1	DGS2	DGS3	DGS5	DGS7	DGS10	DGS20	DGS30
1	2001-07-31	3.67	3.54	3.47	3.53	3.79	4.06	4.57	4.86	5.07	5.61	5.51
2	2001-08-01	3.65	3.53	3.47	3.56	3.83	4.09	4.62	4.90	5.11	5.63	5.53
3	2001-08-02	3.65	3.53	3.46	3.57	3.89	4.17	4.69	4.97	5.17	5.68	5.57
4	2001-08-03	3.63	3.52	3.47	3.57	3.91	4.22	4.72	4.99	5.20	5.70	5.59
5	2001-08-06	3.62	3.52	3.47	3.56	3.88	4.17	4.71	4.99	5.19	5.70	5.59
6	2001-08-07	3.63	3.52	3.47	3.56	3.90	4.19	4.72	5.00	5.20	5.71	5.60

Now we have a data frame that contains all the series we are interested in. Next, we sort the data.frame by date, but before this we first convert the date into number strings as shown below

```
#CONVERT ALL DATES TO NUMERIC AND SORT BY DATE
dates = rates[,1]
library(stringr)
dates = as.numeric(str_replace_all(dates,"-",""))
res = sort(dates,index.return=TRUE)
for (j in 1:dim(rates)[2]) {
  rates[,j] = rates[res$ix,j]
}

> head(rates)
```

	date	DGS1MO	DGS3MO	DGS6MO	DGS1	DGS2	DGS3	DGS5	DGS7	DGS10	DGS20	DGS30
1	1962-01-02	NA	NA	NA	3.22	NA	3.70	3.88	NA	4.06	NA	

NA										
2	1962−01−03	NA	NA	NA	3.24	NA	3.70	3.87	NA	4.03	NA
NA										
3	1962−01−04	NA	NA	NA	3.24	NA	3.69	3.86	NA	3.99	NA
NA										
4	1962−01−05	NA	NA	NA	3.26	NA	3.71	3.89	NA	4.02	NA
NA										
5	1962−01−08	NA	NA	NA	3.31	NA	3.71	3.91	NA	4.03	NA
NA										
6	1962−01−09	NA	NA	NA	3.32	NA	3.74	3.93	NA	4.05	NA
NA										

NA represents missing values. Note that there are values represented by "-99." Although both NA and -99 can be wiped out, we leave them because they represent times when there was no yield for that maturity.

How To Handle Dates Using Lubricate

Assuming we want to sort out the data.frames of failed banks. We would need to do this month by month, day by day, and week by week. This definitely requires the use of dates package. A very unique and useful tool developed by Hadley Wickham is the **lubricate package**.

```
head(data)

                                Bank.Name           City ST  CERT
1                Hometown National Bank        Longview WA 35156
2                  The Bank of Georgia Peachtree City GA 35259
3                         Premier Bank         Denver CO 34112
4                       Edgebrook Bank        Chicago IL 57772
5                          Doral Bank      San Juan PR 32102
6 Capitol City Bank & Trust Company         Atlanta GA 33938
                    Acquiring.Institution Closing.Date Updated.Date count
1                          Twin City Bank      2-Oct-15     15-Oct-15     1
2                          Fidelity Bank       2-Oct-15     15-Oct-15     1
3              United Fidelity Bank, fsb      10-Jul-15     28-Jul-15     1
4               Republic Bank of Chicago       8-May-15     23-Jul-15     1
5           Banco Popular de Puerto Rico      27-Feb-15     13-May-15     1
6 First-Citizens Bank & Trust Company      13-Feb-15     21-Apr-15     1
       Cdate Cyear
1 2015-10-02  2015
2 2015-10-02  2015
3 2015-07-10  2015
4 2015-05-08  2015
5 2015-02-27  2015
6 2015-02-13  2015

library(lubridate)
data$Cdate = dmy(data$Closing.Date)
data$Cyear = year(data$Cdate)
fd = aggregate(count~Cyear,data,sum)
print(fd)

  Cyear count
1  2000     2
```

```
2    2001     4
3    2002    11
4    2003     3
5    2004     4
6    2007     3
7    2008    25
8    2009   140
9    2010   157
10   2011    92
11   2012    51
12   2013    24
13   2014    18
14   2015     8

plot(count~Cyear,data=fd,type="l",lwd=3,col="red"xlab="Year")
grid(lwd=3)
```

We would do the same sorting we did here with a month to see if we will record any form of seasonality.

Let's do the same thing by month to see if there is seasonality

```
data$Cmonth = month(data$Cdate)
fd = aggregate(count~Cmonth,data,sum)
print(fd)
```

```
   Cmonth count
1       1    49
2       2    44
3       3    38
4       4    57
5       5    40
6       6    36
7       7    74
8       8    40
9       9    37
10     10    58
11     11    35
12     12    34

plot(count~Cmonth,data=fd,type="l",lwd=3,col="green"); grid(lwd=3)
```

There is no seasonality with the monthly sorting, let's try with daily sorting

```
data$Cday = day(data$Cdate)
fd = aggregate(count~Cday,data,sum)
print(fd)
```

```
   Cday count
1     1     8
2     2    20
3     3     3
4     4    21
5     5    15
6     6    13
7     7    20
8     8    14
9     9    10
10   10    14
11   11    17
12   12    10
13   13    14
14   14    20
15   15    20
16   16    22
17   17    23
18   18    21
19   19    29
20   20    27
21   21    17
22   22    18
23   23    30
24   24    19
25   25    13
26   26    15
27   27    18
28   28    18
29   29    15
30   30    30
31   31     8
```

```
plot(count~Cday,data=fd,type="l",lwd=3,col="blue"); grid(lwd=3)
```

From our counts, we would observe that counts are indeed lower at the start and end of each month.

Using The Data.Table Package

This is a very brilliant package written by Matt Dowle. The function of the package is to allow data.frame works as a database. Not only this, but it also allows the proper and effective handling of massive quantities of data. The effectiveness IP address of a company known as h2o:http://h2o.ai/ has now embedded this technology. To see how this works, we will be using some downloadable crime data statistics for California. Next, we will create a csv file and place our data inside so that it can be easily read into R.

data = read . csv ("CA_Crimes_Data_2004−2013.csv", header=TRUE)

Now it is easy to convert the data into a database

library (data . table)

D_T = **as . data. table(data)**

Now let's see how this works, we will notice that the syntax of this looks very much like the syntax of data.frame. As a result, we would only print a section of the name and not all.

print (dim(D_T)

```
[1] 7301   69

print(names(D_T))

 [1] "Year"              "County"            "NCICCode"
 [4] "Violent_sum"       "Homicide_sum"      "ForRape_sum"
 [7] "Robbery_sum"       "AggAssault_sum"    "Property_sum"
[10] "Burglary_sum"      "VehicleTheft_sum"  "LTtotal_sum"
....

head(D_T)
```

One of the unique characteristics of the database is that it can be index by making any column in the index key. Once this is done, it is easier to compute subtotals and even generate plots from them.

```
setkey(D_T, Year)

crime = 6
res = D_T[,sum(ForRape_sum),by=Year]
print(res)

    Year   V1
 1: 2004 9598
 2: 2005 9345
 3: 2006 9213
 4: 2007 9047
 5: 2008 8906
 6: 2009 8698
 7: 2010 8325
 8: 2011 7678
 9: 2012 7828
10: 2013 7459

class(res)

[1] "data.table" "data.frame"
```

We would notice that the type of output generated looks like that of the data.table. It also includes classes from DataFrames too. Our next action is to plot the result of the data.table the same way we plot that of the data.frame.

Using the p l y r Table

Hadley Wickham writes this package. It is very useful to apply functions to tables of data (data.frames). In our program, we would want also to write a custom function, it is in writing this function that this package comes in. In R function, we can either use the p l y r class of package or the data.table to handle data.frame as database.

```
require(plyr)
library(dplyr)
```

Next, we would use the filter function to subset the rows of the dataset we want to select for further analysis.

Also, Data.table provides a unique way to carry out statistics. Below are the steps to do this:

1. Group data by standpoint.
2. Use the groups to produce statistics
3. Choose the option that allows you to count the number of trips beginning from the first station and also allows you to calculate the average time of each trips.

Conclusion

This chapter explains in detail how data is handled in R packages. Explanations on how to merge

data to functions, apply functions to data and use the various options available for handling big and small data were provided. The next chapter examines Data statistics.

Chapter Five: Markowitz Mean-Variance Problem

This chapter examines the Markowitz mean-variance problem. This type of problem is not only popular in data science, but its solution is also widely used. In this chapter, we will cover the following outlines:

- Markowitz mean-variance problem
- How to solve the problem using the quadprog package
- Risk Budgeting

Markowitz Mean-Variance Problem

This is a very popular portfolio optimization. The solution to this type of problem is still widely used today. However, our major aim in this chapter is on the portfolio of **n** asset. This implies that the return of E (rp), and a variance denoted as Var(rp). Our portfolio weight is represented by w ∈ Rn. The meaning of this is that in allocating values to the assets, take, for instance, we want to allocate $1 to the asset. It means that each $1 is allocated into various assets. The total value of the sum of our weight is 1.

Quadratic (Markowitz) Problem

This optimization problem can be defined as this. We want our result to achieve the pre-specific level of expected mean return, and its variance(risk) avoided as much as we can.

$$\min_{\underline{w}} \quad \frac{1}{2}\underline{w}'\underline{\Sigma}\,\underline{w}$$

subject to

$$\begin{aligned} \underline{w}'\underline{\mu} &= E(r_p) \\ \underline{w}'\underline{1} &= 1 \end{aligned}$$

The ½ we have in front of the above variance is for mathematical neatness. The function of this would be explained as we progress in this chapter. The scaling of the objective function by a constant does not affect the minimized solution. There are two types of constraints working with our variance above. The expected mean return is forced into a specific mean return E(rp) by the first constraint. The second constraint, also known as the fully invested constraint ensures that the weight of the portfolio is up to 1. These two constraints are equality constraints.

The type of problem explained above is a Lagrangian problem; it requires that we use the Lagrangian multipliers {λ1,λ2} to embed the constraints into the objective function. What we will have after this action is a civilization problem.

We will take derivative with respect to w, λ1, and λ2, to minimize this function and then arrive at the first-order conditions started as follows:

$$\begin{aligned} \frac{\partial L}{\partial \underline{w}} &= \underline{\Sigma}\,\underline{w} - \lambda_1\underline{\mu} - \lambda_2\underline{1} = \underline{0} \qquad (*) \\ \frac{\partial L}{\partial \lambda_1} &= E(r_p) - \underline{w}'\underline{\mu} = 0 \\ \frac{\partial L}{\partial \lambda_2} &= 1 - \underline{w}'\underline{1} = 0 \end{aligned}$$

The first equation represented by (*) is an n equation system. This is because the derivative is taken with respect to all the elements of the vector w. This is why we arrive at a total of (n+2) as our first-order condition. From(*)

$$\begin{aligned} \underline{w} &= \Sigma^{-1}(\lambda_1 \underline{\mu} + \lambda_2 \underline{1}) \\ &= \lambda_1 \Sigma^{-1} \underline{\mu} + \lambda_2 \Sigma^{-1} \underline{1} \quad (**) \end{aligned}$$

Premultiply (**) by $\underline{\mu}'$:

$$\underline{\mu}' \underline{w} = \lambda_1 \underbrace{\underline{\mu}' \Sigma^{-1} \underline{\mu}}_{B} + \lambda_2 \underbrace{\underline{\mu}' \Sigma^{-1} \underline{1}}_{A} = E(r_p)$$

Also premultiply (**) by $\underline{1}'$:

$$\underline{1}' \underline{w} = \lambda_1 \underbrace{\underline{1}' \Sigma^{-1} \underline{\mu}}_{A} + \lambda_2 \underbrace{\underline{1}' \Sigma^{-1} \underline{1}}_{C} = 1$$

Solve for λ_1, λ_2

$$\lambda_1 = \frac{CE(r_p) - A}{D}$$

$$\lambda_2 = \frac{B - AE(r_p)}{D}$$

where $D = BC - A^2$

Let's take note of these observations:

Since Σ is positive, this means that Σ^{-1} would also be positive and: B>0, C>0.

Taking the solutions for λ1,λ2, we would find out the solution for **w** using this formula

$$\underline{w} = \underbrace{\frac{1}{D}\left[B\Sigma^{-1}\underline{1} - A\Sigma^{-1}\underline{\mu}\right]}_{\underline{g}} + \underbrace{\frac{1}{D}\left[C\Sigma^{-1}\underline{\mu} - A\Sigma^{-1}\underline{1}\right]}_{\underline{h}} \cdot E(r_p)$$

The above equation is the expression for the optimization equation weight when the variance is minimized for a given amount of expected return E(rp). Once the inputs to the problems μ and Σ is given, the vectors g and h become fixed.

E(rp) can be varied to get a set of frontier (optimal or efficient) portfolios w

$$\underline{w} = \underline{g} + \underline{h}\, E(r_p)$$

$$\begin{aligned} if \quad E(r_p) &= 0, \ \underline{w} = \underline{g} \\ if \quad E(r_p) &= 1, \ \underline{w} = \underline{g} + \underline{h} \end{aligned}$$

Note that

$$\underline{w} = \underline{g} + \underline{h}\, E(r_p) = [1 - E(r_p)]\, \underline{g} + E(r_p)[\underline{g} + \underline{h}]$$

```
> Er = 0.10
> wts = markowitz(mu,cv,Er)
> print(wts)
          [,1]
[1,] 0.3209169
[2,] 0.4223496
[3,] 0.2567335
```

Therefore, these two portfolios g, g, and h produce the entire frontier.

Solution in R

We can use R to create a function to return the optimal portfolio weight. To do this, we will use the following formula

We can call the function of an expected return and then enter the example of a mean return vector and the covariance matrix of returns.

```
#PARAMETERS
mu = matrix(c(0.02,0.10,0.20),3,1)
n = length(mu)
cv = matrix(c(0.0001,0,0,0,0.04,0.02,0,0.02,0.16),n,n)
Er = 0.18

#SOLVE PORTFOLIO PROBLEM
wts = markowitz(mu,cv,Er)
print(wts)
```

The output is the vector of the optimal portfolio weight.

However, we will get a different output when the expected return is changed to 0.10

```
> Er = 0.10
> wts = markowitz(mu,cv,Er)
> print(wts)
          [,1]
[1,] 0.3209169
[2,] 0.4223496
[3,] 0.2567335
```

To get the expected return of 0.18 in the first example, we would notice that we shorten some low-risk assets and lengthen some medium and high-risk assets. However, when the expected return was reduced to 0.10, all the weight are positive.

How To Solve The Problem Using The Quadprog Package

This is an optimizer that uses linear constraint to take a quadratic objective function. As a result, this is exactly what we need to solve the mean-variance portfolio problem we just treated. Another significant use of this package is that we can use additional inequality constraints. For instance, whenever we don't feel like granting short sales of any asset, we can easily bound the weight to lie between zero and one. The manual below shows the specification of the quadprog package.

```
Description

This routine implements the dual method of Goldfarb and
Idnani (1982, 1983) for solving quadratic programming
problems of the form min(-d^T b + 1/2 b^T D b) with the
constraints A^T b >= b_0.
(note: b here is the weights vector in our problem)

Usage
solve.QP(Dmat, dvec, Amat, bvec, meq=0, factorized=FALSE)

Arguments
Dmat        matrix appearing in the quadratic function to be minimized.
dvec        vector appearing in the quadratic function to be minimized.
Amat        matrix defining the constraints under which we want
            to minimize the quadratic function.
bvec        vector holding the values of b_0 (defaults to zero).
meq         the first meq constraints are treated as equality
            constraints, all further as inequality constraints
            (defaults to 0).
factorized          logical flag: if TRUE, then we are passing R^(-1)
            (where D = R^T R) instead of the matrix D in the
            argument Dmat.
```

In the setup of the problem, we are dealing with, with no short cuts and three securities. We will have the following bvec and Amat\

$$A = \begin{bmatrix} \mu_1 & 1 & 1 & 0 & 0 \\ \mu_2 & 1 & 0 & 1 & 0 \\ \mu_3 & 1 & 0 & 0 & 1 \end{bmatrix}; \quad b_0 = \begin{bmatrix} E(r_p) \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The constraints will be modulated by meq = 2. This states that the first two constraints will be equality constraints, while the last two will be greater than equal to a constraint.

$A'w \geq b_0$, i.e.,

$$\begin{aligned} w_1\mu_1 + w_2\mu_2 + w_3\mu_3 &= E(r_p) \\ w_1 1 + w_2 1 + w_3 1 &= 1 \\ w_1 &\geq 0 \\ w_2 &\geq 0 \\ w_3 &\geq 0 \end{aligned}$$

The package code would be run in this format:

```
library(quadprog)
nss = 1                    #Equals 1 if no short sales allowed
Bmat = matrix(0,n,n)       #No Short sales matrix
diag(Bmat) = 1
Amat = matrix(c(mu,1,1,1),n,2)
if (nss==1) { Amat = matrix(c(Amat,Bmat),n,2+n) }
dvec = matrix(0,n,1)
bvec = matrix(c(Er,1),2,1)
if (nss==1) { bvec = t(c(bvec,matrix(0,3,1))) }
sol = solve.QP(cv,dvec,Amat,bvec,meq=2)
print(sol$solution)
```

After running the code, our expected result would be 0.18, with a short-selling that allows:

[1] −0.3575931 0.8436676 0.5139255

This is exactly the same result we got in Markowitz's solution. When we restrict short-selling, we will arrive at the same 0.10 we got in Markowitz.

Risk Budgeting

A single problem can have a different view of the Markowitz optimization problem. To control this, we use one of the recent approaches to risk portfolio construction. We construct a portfolio where the risk construction of all assets is equal. This approach is known as **"Risk Parity."** Another portfolio where all the risk contributes the same quota of the total return of the portfolio would also be created. This type of approach is known as the **"Performance Parity Approach."**

Assuming its weights portray the portfolio, the risk becomes the function of its weight and is denoted by R(w). The standard deviation of the portfolio is as follows:

$$R(\mathbf{w}) = \sigma(\mathbf{w}) = \sqrt{\mathbf{w}^\top \mathbf{\Sigma} \mathbf{w}}$$

The risk function of this kind of risk is homogenous. This implies that if the size of the portfolio is doubled, then the risk measures also doubles. This is also known as the homogeneity property of risk measurement. Homogeneity is one of the coherence in risk measurement explained by Eber, Artzner, Health, and Delbaen (1999). Once a risk measurement meets the requirements of homogeneity, the next step is to apply Euler's theory to decompose the risk into the amount given by each asset.

Let's assume that we define the risk measurement to be the standard deviation of portfolio return; the risk decomposition would require the measurement of the risk along with all its weight. This is shown as follows:

$$R(\mathbf{w}) = \sum_{j=1}^{n} w_j \frac{\partial R(\mathbf{w})}{\partial w_j}$$

We can verify the sum of total risk using this procedure:

$$\begin{aligned}
\sum_{j=1}^{n} w_j \frac{\partial R(\mathbf{w})}{\partial w_j} &= [w_1 \quad w_2 \quad \dots \quad w_n] \cdot [\mathbf{\Sigma} \mathbf{w} / \sigma(\mathbf{w})] \\
&= \mathbf{w}^\top \cdot [\mathbf{\Sigma} \mathbf{w} / \sigma(\mathbf{w})] \\
&= \frac{\sigma(\mathbf{w})^2}{\sigma(\mathbf{w})} \\
&= \sigma(\mathbf{w}) \\
&= R(\mathbf{w})
\end{aligned}$$

Conclusion

In this chapter, we observe the Markowitz problem in Data science and the various packages that can solve this problem. The next chapter examines Bayes' theorems and the types of models under it.

Chapter Six: Bayes Theorem

This theorem deals with coincidence and reality. A very good explanation of the theory is explicated on Wikipedia http://en.wikipedia.org/wiki/Bayes theorem and a video by Professor Persi Diaconis's talk on Bayes on Yahoo video. In business, we often encounter questions bothering on reality and coincidence. A good example of the question is, is Warren Buffet's investment success a coincidence? How do we answer the question? Do we use our prior knowledge of the probability of Buffet being able to beat the market, or do we check the performance of the business over time? It is in answering this question that Buffet rule comes in. The rule follows from the decomposition of joint probability. Here is the formula

$Pr[A \cap B] = Pr(A|B)\ Pr(B) = Pr(B|A)\ Pr(A)$

The last two terms in the equation can be restated like this:

$$Pr(A|B) = \frac{Pr(B|A)\ Pr(A)}{Pr(B)}$$

or

$$Pr(B|A) = \frac{Pr(A|B)\ Pr(B)}{Pr(A)}$$

The example we would be using is the Aid Test

This is a very intriguing test. Applying the Bayes Theorem implies that if you are diagnosed with Aid, there is a chance that you don't have the disease; however, if you are diagnosed with not having the disease, there is a good chance that this is true.

We would use the equation { Pos, Neg} as a positive or negative diagnosis of having AIDS. While {Dis, NoDis} would represent having or not having the diseases. Taking the US report on AIDS as a case study, there are over 1.5 million AID cases in a population of over 300 million people in the USA. This implies that the probability of people with AID in the country is 0.5%. Since the probability percentage is half a percent, in doing a random test to discover someone with AID, we would use half a percent probability. The percentage of accuracy is 99%. Our equation would be as follows:

$Pr(Pos|Dis) = 0.99$

For those without the disease, the accuracy test is

$Pr(Neg|NoDis) = 0.95$

In finding out the probability of having the disease when the test says so, we would compute our confirmation accuracy of the AIDS test using Bayle's Rule

$$
\begin{aligned}
Pr(Dis|Pos) &= \frac{Pr(Pos|Dis)Pr(Dis)}{Pr(Pos)} \\
&= \frac{Pr(Pos|Dis)Pr(Dis)}{Pr(Pos|Dis)Pr(Dis) + Pr(Pos|NoDis)Pr(NoDis)} \\
&= \frac{0.99 \times 0.005}{(0.99)(0.005) + (0.05)(0.995)} \\
&= 0.0904936
\end{aligned}
$$

From our calculation above, the chance of having the diseases when the test is positive is 9%. Now we would calculate the chance of not having it when the test says positive

Pr(NoDis|Pos) = 1− Pr(Dis|Pos) = 1−0.09 = 0.91

The question now is, what is the chance of having the disease when the test says negative? This is often a worry to some. Using Bayle's theorem, our calculation would be as follows:

$$
\begin{aligned}
Pr(Dis|Neg) &= \frac{Pr(Neg|Dis)Pr(Dis)}{Pr(Neg)} \\
&= \frac{Pr(Neg|Dis)Pr(Dis)}{Pr(Neg|Dis)Pr(Dis) + Pr(Neg|NoDis)Pr(NoDis)} \\
&= \frac{0.01 \times 0.005}{(0.01)(0.005) + (0.95)(0.995)} \\
&= 0.000053
\end{aligned}
$$

From our test, when the test is negative, there is a very slim chance that you might have it, so there is nothing to worry about.

Correlated Default (Conditional Default)

Bayes' theorem is very effective for verifying conditional default information. Bond fault managers are not as concerned with the correlation of defaults in the bond of their portfolio as much as they are concerned with the conditional default of bond. This means that they are concerned with the conditional probability of bond. To calculate this, some of the modern financial institutions already develop tools to obtain the conditional default of firms.

Let's assume that we already know that firm 1 has a default probability P1 = 1%, and firm 2 has a default probability P2=3%. Assuming the default of both firms is 40% in a year, however, if either bond default, what is the probability of default of the other conditional on the first default?

Despite the limited information on the firm's probability of default, we can still use Bayes theorem to define the conditional probability of interest. Here are the steps to calculate this:

define di, i = 1,2. This is the default indicator for the two firms

define di = 1 if the firms default.

define di =0 if the firms did not.

We would note the following in our Bayes application

E(d1) = 1.p1 +0.(1− p1) = p1 = 0.01.

Likewise

E(d2) = 1.p2 +0.(1− p2) = p2 = 0.03.

With Bernoulli distribution, we would be able to determine the standard deviation of d1 and d2.

$$\sigma_1 = \sqrt{p_1(1-p_1)} = \sqrt{(0.01)(0.99)} = 0.099499$$
$$\sigma_2 = \sqrt{p_2(1-p_2)} = \sqrt{(0.03)(0.97)} = 0.17059$$

Now, we note that

$$\begin{aligned} Cov(d_1, d_2) &= E(d_1.d_2) - E(d_1)E(d_2) \\ \rho\sigma_1\sigma_2 &= E(d_1.d_2) - p_1p_2 \\ (0.4)(0.099499)(0.17059) &= E(d_1.d_2) - (0.01)(0.03) \\ E(d_1.d_2) &= 0.0070894 \\ E(d_1.d_2) &\equiv p_{12} \end{aligned}$$

In the above calculation, p12 is the default probability for the two firms. Our conditional probabilities would be:

p(d1|d2) = p12/p2 = 0.0070894/0.03 = 0.23631

p(d2|d1) = p12/p1 = 0.0070894/0.01 = 0.70894

From the result of this conditional probability, it can be summarized that once the firm begins to defect, the default contagion would start getting severe.

Continuous and More Formal Exposition

There is some very significant expression in Bayesian approaches. These expressions are **posterior**, **prior,** and **likelihood**. These expressions would be explained in detail in this section. Usually, in standard notation, we are concerned with the parameter of a θ, the mean of a distribution of some data x. However, in Bayesian theory, we won't only be concentrating on the value of θ, we would also be exploring the distribution value of θ beginning with some prior assumption about this distribution. Therefore, we would begin with p(θ); this is referred to as prior distribution. We then move to the data x and combine our prior distribution value to it to get the posterior distribution p(θ|x). However, to do this, we are required to compute the probability of seeing the data x given our prior p(θ). This probability is given by the likelihood function L(x|θ). Assuming that we already know the variance of our data x as o^2. When we apply our Bayesian theory, we would have:

$$p(\theta|x) = \frac{L(x|\theta)\ p(\theta)}{\int L(x|\theta)\ p(\theta)\ d\theta} \propto L(x|\theta)\ p(\theta)$$

If we assume that both the prior distribution for the mean and the likelihood are normal, then we would have:

If this be the case, our posterior value would be

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2}\frac{(\theta-\mu_0)^2}{\sigma_0^2}\right] \sim N[\theta|\mu_0,\sigma_0^2] \propto \exp\left[-\frac{1}{2}\frac{(\theta-\mu_0)^2}{\sigma_0^2}\right]$$

$$L(x|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(x-\theta)^2}{\sigma^2}\right] \sim N[x|\theta,\sigma^2] \propto \exp\left[-\frac{1}{2}\frac{(x-\theta)^2}{\sigma^2}\right]$$

When the prior distribution and posterior distribution are of the same form, they are a "**conjugate**" with respect to the specific likelihood function. However, if we observe n new value of x, the new posterior would be:

$$p(\theta|x) \sim N\left[\frac{\tau_0}{\tau_0+n\tau}\mu_0 + \frac{\tau}{\tau_0+n\tau}\sum_{j=1}^{n} x_j,\ \ \frac{1}{\tau_0+n\tau}\right]$$

Bayes Net

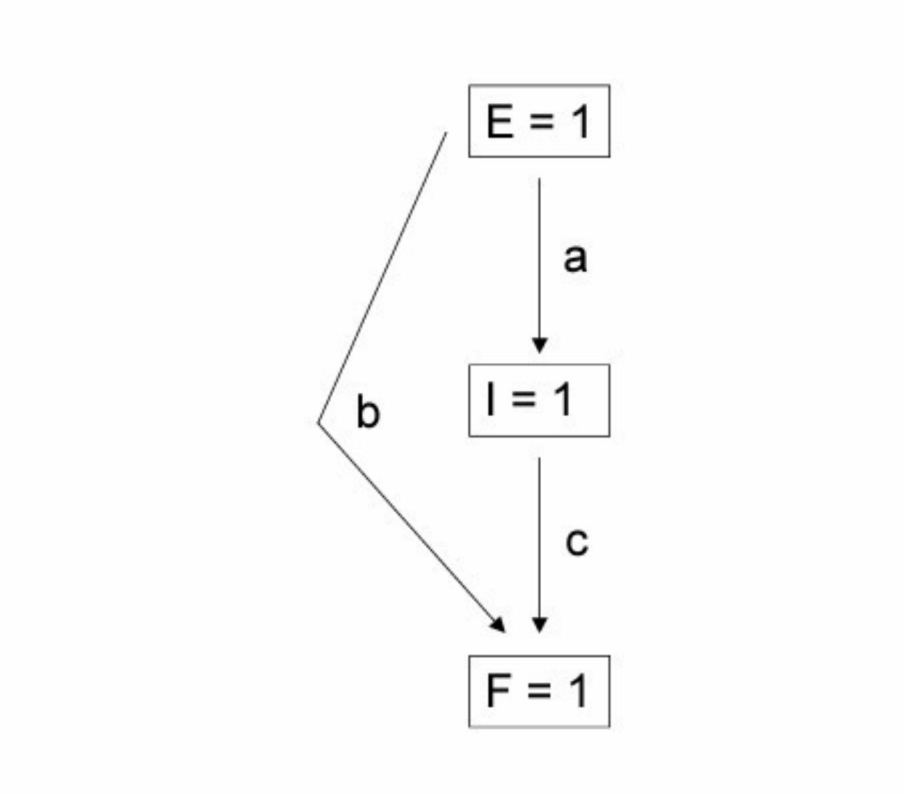
Bayes Net is a network diagram that can be used to visualize joint distributions over several outcomes/events and higher-dimension Bayes problem. The net is a directed acyclic graph (referred to as DAG). This means that circles are not permitted in the graph.

To understand how Bayes Networks, we would be using an example of economic distress. Distress can be noticed at these three levels: economy level (E = 1), industry level (I = 1), and the firm level (F = 1). Economy distress can result in industry distress, but this may or may not lead to firm distress. The diagram below shows the flow of causality. It is noteworthy that the probability in our first table is unconditional, but all others are conditional.

E	Prob
1	0.10
0	0.90

E	I	Conditional Prob	Channel
1	1	0.60	a
1	0	0.40	
0	1	0.20	–
0	0	0.80	

In our conditional probabilities, each pair adds up to 1. The channels in the table the arrows in the Bayes net diagram.



E	I	F	Conditional Prob	Channel
1	1	1	0.95	a+c
1	1	0	0.05	
1	0	1	0.70	b
1	0	0	0.30	
0	1	1	0.80	c
0	1	0	0.20	
0	0	1	0.10	–
0	0	0	0.90	

In the first diagram, we would notice that there are three channels in the Bayes net. Channel a stands for the inducement of industry distress from economic distress; channel b stands for the inducement of firm distress directly from industry distress. The last channel c stands for the inducement of firm distress directly from industry distress.

The question that arises from this net is, what is the probability that the industry is distressed if the firm is in distressed? The calculation for this problem is stipulated below:

$$
\begin{aligned}
Pr(I=1|F=1) &= \frac{Pr(F=1|I=1)\cdot Pr(I=1)}{Pr(F=1)} \\
\\
Pr(F=1|I=1)\cdot Pr(I=1) &= Pr(F=1|I=1)\cdot Pr(I=1|E=1)\cdot Pr(E=1) \\
&\quad +Pr(F=1|I=1)\cdot Pr(I=1|E=0)\cdot Pr(E=0) \\
&= 0.95\times 0.6\times 0.1 + 0.8\times 0.2\times 0.9 = 0.201 \\
\\
Pr(F=1|I=0)\cdot Pr(I=0) &= Pr(F=1|I=0)\cdot Pr(I=0|E=1)\cdot Pr(E=1) \\
&\quad +Pr(F=1|I=0)\cdot Pr(I=0|E=0)\cdot Pr(E=0) \\
&= 0.7\times 0.4\times 0.1 + 0.1\times 0.8\times 0.9 = 0.100 \\
\\
Pr(F=1) &= Pr(F=1|I=1)\cdot Pr(I=1) \\
&\quad +Pr(F=1|I=0)\cdot Pr(I=0) = 0.301 \\
\\
Pr(I=1|F=1) &= \frac{Pr(F=1|I=1)\cdot Pr(I=1)}{Pr(F=1)} = \frac{0.201}{0.301} = 0.6677741
\end{aligned}
$$

Bayes Rule in Marketing

In one of the widest market research campaign: pilot marketing, Bayes showed up in a very easy manner. Let's assume we have a project with a value x. Now, if the product fails (F), the payoff is -70; however, if it is successful (S), the payoff is +100. The probability of these two happenings is

Pr(S) = 0.7, Pr(F) = 0.3

We can easily check that our expected end is E(x) = 49. Assuming we were able to get protection for a failed product, the protection would be a put option of the real option; it's worth rate would be 0.3 ×70 = 21. Since the put option is what saves all the loss recorded by the failed product, value is the expected loss, condition on loss. This is usually seen as the value of "perfect information" by market researchers.

However, suppose there is an intermediate choice rather than proceeding with the product launch after the odds, we would have done a pilot test. Although this is not always accurate, it is reasonably sophisticated.

The test signal of the pilot test is (T+) or failure (T-). Our probabilities in pilot test would be as follows:

$$\begin{aligned} Pr(T+|S) &= 0.8 \\ Pr(T-|S) &= 0.2 \\ Pr(T+|F) &= 0.3 \\ Pr(T-|F) &= 0.7 \end{aligned}$$

The pilot test above gives only a valid reading of success 80% of the time. The probability that the pilot signal gives a positive result can be computed as follows:

Pr(T+) = Pr(T+|S)Pr(S)+Pr(T+|F)Pr(F)

= (0.8)(0.7) +(0.3)(0.3) = 0.65

Negative result can be computed as follows:

Pr(T−) = Pr(T−|S)Pr(S)+Pr(T−|F)Pr(F)

= (0.2)(0.7) +(0.7)(0.3) = 0.35

This would allow us to compute the following:

$$\begin{aligned} Pr(S|T+) &= \frac{Pr(T+|S)Pr(S)}{Pr(T+)} = \frac{(0.8)(0.7)}{0.65} = 0.86154 \\ Pr(S|T-) &= \frac{Pr(T-|S)Pr(S)}{Pr(T-)} = \frac{(0.2)(0.7)}{0.35} = 0.4 \\ Pr(F|T+) &= \frac{Pr(T+|F)Pr(F)}{Pr(T+)} = \frac{(0.3)(0.3)}{0.65} = 0.13846 \\ Pr(F|T-) &= \frac{Pr(T-|F)Pr(F)}{Pr(T-)} = \frac{(0.7)(0.3)}{0.35} = 0.6 \end{aligned}$$

Now that we have these conditional probabilities, let us re-evaluate our product launch. If the result of the pilot test is positive, what do we expect of the value of our product launch. This would be as follows:

$$E(x|T+) = 100Pr(S|T+)+(-70)Pr(F|T+)$$

$$= 100(0.86154)-70(0.13846)$$

$$= 76.462$$

But if the test is negative, the value of our launch is

$$E(x|T-) = 100Pr(S|T-)+(-70)Pr(F|T-)$$

$$= 100(0.4)-70(0.6)$$

$$= -2$$

Now that we know the value of both the negative pilot test and positive pilot test, our overall value of pilot test would be:

$$E(x) = E(x|T+)Pr(T+)+E(x|T-)Pr(T-)$$

$$= 76.462(0.65) +(0)(0.35)$$

$$= 49.70$$

Without the pilot test, the incremental value over the case is 0.70.

Bayes Models in Credit Rating Transitions

Most times, companies or business organizations are allocated to credit rating classes. Unlike default probability, credit rating is a more coarse bucket of credit. Also, updating the credit rating class in the section tends to be very slow. As a result, the DFG models use a Bayesian approach to develop a model of rating changes that uses contemporaneous data on default probabilities.

Accounting Fraud

Bayesian inference can also be used to detect accounting fraud and audits. When fraudulence is suspected, an auditor can use a Bayesian hypothesis of fraud to verify past data and assess the chance that the current fraud situation has been ongoing for a while.

Conclusion

In this chapter, we have examined the main focus and use of the Bayes Model. We examined Bayesian Net and how we use Bayesian to explain conditional default information. In the next chapter we examine News Analysis in Data science, algorithms, word count, and more.

Chapter Seven: More Than Words - Extracting Information From News

This chapter explains in detail the concept of news extracting. Wikipedia defines news analysis as the measurement of the various qualitative and quantitative attributes of textual news stories. Some of these attributes are sentiment, relevance, and novelty. Expressing news stories as numbers the manipulation of everyday information mathematically and statistically." The chapter examines the various analytical techniques in news extraction, the various news analytic software, method, and the sets of metrics that can be used for the assessments of analytic performance. The outlines that would be covered in this chapter include:

- What is News Analysis?
- Algorithms
- Scrapers and Crawlers
- Pre-possessing Test
- Term Frequency - Inverse Document Frequency (TF - IDF)
- Text Classification
- Word Count Multiplier
- Metrics
- Text Summarization

What is News Analysis

This is an umbrella term that covers a set of formulas, techniques, and statistics used to classify and summarized public sources of information. It also includes metrics that are used to assess analytics. The field of News analysis is very broad; it covers aspects such as machine learning, information retrieval, network theory, statistic learning theory, and collaborative filtering. However, all these can be broken into three broad categories of news analysis: text, content, and context.

Text in news analytics entails the visceral aspect of news, i.e., words, phrases, sentences, document headings and so on. The main purpose of analytics here is to convert text into information. This action is carried out by these three means:

- Signing the text
- Classifying the text
- Summarizing it into its main component.

During the summarization process, analytic discard text that is not relevant while separating information that is of higher signal content.

The next layer of news analytic is content. Content works on the domain of text by expanding its images, text forms (blogs, emails, pages, etc.), time, formats (XML, HTML), etc. Content enriches text such that it asserts quality and veracity that can be explored in analytics. For instance, a blog can be streamed to have a higher-quality than a stock message-board post; however, when financial information is streamed with Dow Jones, it can have more value than a blog.

The last layer of news analytic is the context. This is simply the relationship between information items. This can also refer to the network relationship of news. In exploring the relationship between context and news analytic, Das, Martinez-Jerez, and Tufano (2005), a clinical study of four companies examines the relationship of news analytic to message-board postings. Similarly, Das and Sisk (2005) explore the social networks of message-board postings to find out if the rules of a portfolio can be created with the network connections between stocks. A good example of an analytic that functions at all these three levels are Google's PageRank algorithms. Algorithms have a lot of features; the kernel of these features are context while others are text and content. Context is the kernel of algorithms because search is the most popularly used news analytics. However, this depends on the number of highly ranked pages pointing to it.

From our explanation so far, it can be deduced that news analytics is where algorithms and data meet. This is where tension is generated between the two. This is why there has been a heated debate on which of the two should be more than the other. This debate was brought up in a talk at the 17th ACM Conference on Information Knowledge and Management (CIKM '08), Peter Norvig, Google's director of research, made his preference by stating that it is better to have more data than algorithms. According to him, “data is more agile than code.” On the one hand, this might sound reasonable okay, on the other, too much data can make algorithms become useless thereby leading to overfitting.

When we talk about algorithms and data and which among the two should be more than the other, this debate made it seems as if there is no correlation or relationship between the two. However, this is not the case. To start with, news data shares the same three broad classifications that news analytic has, i.e., text, content, and context. The level of complexity of any of these three depends on which one is dominant. Generally, in news data, the simplest among the three is text analysis. The context that applies to network relationships can be quite difficult. For example, a community-detection algorithm can be very difficult to compare to word-count algorithms which are very simple, almost naive. The community-detection algorithm has more complicated memory requirements and logic.

The tension between the two aspects, News data, and news algorithms, is managed and controlled by domain specificity. This implies the quantity of customization needed to implement news analytic. It is quite interesting that low-complexity algorithms more domain specificity than high-complexity ones. For example, the previous illustration we use, community-detection would need little domain knowledge because it is applicable to a wide range of the graph. However, this is not the case with word-count algorithms. A word-count algorithm requires domain knowledge of grammar, lexicon and even syntax. Not only this, political messages would be read differently and separated from medical messages.

Algorithms

Crawlers and Scrapers

Crawlers are set.of algorithms that are used to generate a series of web pages that may be used to search for news content. The software derives it's name "crawler" from the way it works. It starts from some web pages and crawls to others. By this, the algorithms make it choose from the series of web pages it gathered. The commonest approach to choosing a page out of the numerous ones gathered is to move from the current page to a page that is linked to hyper-referenced. Significantly, a crawler uses heuristics to explore the tree from any given node and used this to determine useful paths among the numerous ones before choosing which ones to focus on.

Web scrapers download the details of any web page chosen; it may or may not format the web page for analysis. Virtually all programming language has its own modules used for web scrapping. The modules contain some inbuilt function that is directed connected to the web. Once the functions are opened, it makes it easy to download user-specific or crawler-specific URLs. The popularity of web analysis has made most statistic packages come with its own inbuilt web scraping functions. For instance, R functions come with its own web scraping function in its base distribution. Whenever we want to read a page into a vector line, we can easily download and use a single-line command.

Excel, which is the most widely used spreadsheet, has its own inbuilt web scraping function. This can be downloaded from the Data ----- GetExternal command tree. Once we download the web scraping function, it can be transferred into a worksheet and then operated as desired. We can also set up excel such that it refreshes the content constantly.

Gone are the days when to use web-scraping code; we will need to write it in Java, C, Python or Perl. Today, we can use tools like R to handle statistical analysis, algorithms, and data. With R, these three can be written within the same software. This is to say, data science progress daily.

Pre-Processing Text

Often times we think that no text can be dirtier than text from external sources; this is not the case. Text from web pages is dirtier than text from external sources. Before applying news analysis on algorithms, they must be first cleaned. The process of cleaning up algorithms before applying news analysis on them is what is known as pre-processing. The first process to cleaning algorithms is by using HTML cleanup; this process removes all HTML tags from the body of messages. Example of these tags include <p>,
&quot,etc. The next cleanup is with abbreviations. Here we expand abbreviations to their full forms. All abbreviated phrases and contractions are written out in full. For instance, it's is written out as it is, ain't is written out as are not, etc. The third cleanup is a negative expression. An expression containing negative words would mean the opposite of the negative expression. To handle this, we first detect the negative words such as not, no and never. Then we tag the remaining words in the sentence where the negative words are used. This would help reverse the meaning of the sentence.

Another significant aspect of pre-processing is the stem. This aspect deals with the root words. In stem, words are replaced and represented by their roots words. This would make it possible for

the tenses of the words not to be treated differently. There are various types of stemming algorithms available in a programming language. Popular among these stemming is Porter stemmer discovered in 1980. Stemming varies from language to language. Hence, it is language-dependent.

Term Frequency - Inverse Document Frequency (TF - IDF)

This is a scheme used to weigh the usefulness of rare words in a document. The TF-IDF uses a very easy calculation and does not have any strong theoretical basis. It is simply the importance of a word (w), in a document (d) in a corpus (C). Since this is a function of the three aspects, we will write it as TF-IDF(w, d, C), It is a product of term frequency (TF) and inverse document frequency (IDF).

$$f(w,d) = \frac{\#w \in d}{|d|}$$

Frequency is calculated like this

$$IDF(w,C) = \ln\left[\frac{|C|}{|d_{w\in d}|}\right]$$

Where d is the number of words in a document, the frequency equation would be rewritten as:

TF(w,d) = ln[f(w,d)]

The above equation is known as **Log Normalization**. There is another form of normalization known as **Double Normalization**. The formula for this is

$$TF(w,d) = \frac{1}{2} + \frac{1}{2}\frac{f(w,d)}{\max_{w\in d} f(w,d)}$$

The formula for inverse document frequency is:

$$IDF(w,C) = \ln\left[\frac{|C|}{|d_{w\in d}|}\right]$$

The formula for the score of weight for a given word w in document d and corpus c is

TF-IDF(w,d,C) = TF(w,d)× IDF(w,C)

We will illustrate this using the application below:

```
tdm_mat = as.matrix(tdm)  #Convert tdm into a matrix
print(dim(tdm_mat))
nw = dim(tdm_mat)[1]
nd = dim(tdm_mat)[2]
d = 13    #Choose document
w = "derivatives"    #Choose word

#COMPUTE TF
```

```
f = tdm_mat[w,d]/sum(tdm_mat[,d])
print(f)
TF = log(f)
print(TF)

#COMPUTE IDF
nw = length(which(tdm_mat[w,]>0))
print(nw)
IDF = nd/nw
print(IDF)

#COMPUTE TF–IDF
TF_IDF = TF*IDF
print(TF_IDF)   #With normalization
print(f*IDF)    #Without normalization
```

When we run this code, here is the result we will arrive at:

```
> print(TF_IDF)   #With normalization
[1] -30.74538
> print(f*IDF)    #Without normalization
[1] 2.257143
```

The code can be written into a function, after which we then examine the TF-IDF for all words. These can be used to weigh other words in further analysis.

Word Clouds

You can make a word cloud from this document. It would come out like this:

```
> library(wordcloud)
Loading required package: Rcpp
Loading required package: RColorBrewer
> tdm = as.matrix(tdm_text)
> wordcount = sort(rowSums(tdm),decreasing=TRUE)
> tdm_names = names(wordcount)
> wordcloud(tdm_names,wordcount)
```

Text Classification

Bayes Classifier

This is the most widely used classifier today. Bayes Classifier simply takes some part of the text and then assign it to one of the pre-determined set of category. The classifier is first trained on a pre-classified initial corpus before it is applied to the text. It is this trained data that produces the prior probabilities needed for the Bayesian analysis of the text. Next, we applied the classifier to an out of sample text to obtain the posterior probability of textual categories. The text is then applied to the category that has the highest posterior probability.

To see how this works, we would use an e1071 R package that contains the function of naive Bayes. Next, we would use iris data that contain detail of the flower. Then we will take a classifier to go through the flower data and identify which one among the numerous flower is it. To list out the set of data loaded on our R package, we would use the following

```
library(e1071)
data(iris)
res = naiveBayes(iris[,1:4], iris[,5])
> res

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = iris[, 1:4], y = iris[, 5])

A-priori probabilities:
iris[, 5]
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Conditional probabilities:
            Sepal.Length
iris[, 5]      [,1]      [,2]
  setosa      5.006 0.3524897
  versicolor  5.936 0.5161711
  virginica   6.588 0.6358796

            Sepal.Width
iris[, 5]      [,1]      [,2]
  setosa      3.428 0.3790644
  versicolor  2.770 0.3137983
  virginica   2.974 0.3224966

            Petal.Length
iris[, 5]      [,1]      [,2]
  setosa      1.462 0.1736640
  versicolor  4.260 0.4699110
  virginica   5.552 0.5518947

            Petal.Width
iris[, 5]      [,1]      [,2]
  setosa      0.246 0.1053856
  versicolor  1.326 0.1977527
  virginica   2.026 0.2746501
```

Next, we call a prediction test to predict a single data or to generate a confusion matrix in this format:

```
> predict(res,iris[3,1:4],type="raw")
     setosa    versicolor     virginica
[1,]      1 2.367113e-18 7.240956e-26
> out = table(predict(res, iris[,1:4]), iris[,5])
> print(out)

            setosa versicolor virginica
  setosa        50          0         0
  versicolor     0         47         3
  virginica      0          3        47
```

In the above table, the mean and standard deviation of the table is given. Basic Bayes calculation would take the following pattern:

$$Pr[F=1|a,b,c,d] = \frac{Pr[a|F=1]\cdot Pr[b|F=1]\cdot Pr[c|F=1]\cdot Pr[d|F=1]\cdot Pr(F=1)}{Pr[a,b,c,d|F=1]+Pr[a,b,c,d|F=2]+Pr[a,b,c,d|F=3]}$$

F stands for the type of flower, while a, b, c, and d stand for the four attributes of the flower. Note that we didn't compute the denominator because it is still the same for the calculation of Pr[F=1|a,b,c,d],Pr[F=2|a,b,c,d],or Pr[F=3|a, b, c, d]

Support Vector Machines (SVM)

This is a kind of classifier technique. It is very similar to cluster analysis but also applicable to very high+ dimensional spaces. SVM can be best described by taking every text message as a vector in high-dimension space. The number of data can be taken as similar to the number of words in a dictionary. As a very simple example, we would use the same flower data set we use in the naive Bayes.

```
#USING SVMs
> res = svm(iris[,1:4], iris[,5])
> out = table(predict(res, iris[,1:4]), iris[,5])
> print(out)

            setosa versicolor virginica
  setosa        50          0         0
  versicolor     0         48         2
  virginica      0          2        48
```

SMV is very fast and can be used in news analytics.

Word Count Multiplier

Word count is the simplest form of classifier. Every language inference works with words. This implies that words are the main factor of every language inference. Hence it varies from domain to domain. FC Bartlett states that Words can indicate the qualitative and relational features of a situation in their general aspect just as directly as, and perhaps even more satisfactorily than, they can describe its particular individuality, This is, in fact, what gives to language its intimate relation to thought processes."

To get started with a word count classifier, the user would first determine the lexicon of the words relating to the classification problem being treated. For example, if the text is to be classified into optimistic versus pessimistic economic news. The user would first want to separate the lexicon of the bad news from that of the good news. To do this, he or she would need the use of domain knowledge in designing the lexicon of the words. Hence, unlike the Bayesian classifier, a word count classifier is language-specific.

If, while counting out the numbers of words in each category, the number of words in each category exceeds the other, the text message is assigned to the aspect with the highest lexical counts.

Vector Distance Classifier (VDC)

In VDC, messages are seen as a word vector. As a result, every hand-tagged, pre-classified text message in the corpus of training becomes a comparison vector. This is called set the **Rule Set**. To assign a classification to a text message, it is first compared to the ruleset. Classification is assigned based on how the ruleset is to the vector space. The measure of proximity is provided by the angle between the message vector (M) and the vectors in the ruleset (S)

$$\cos(\theta) = \frac{M \cdot S}{||M|| \cdot ||S||}$$

A search engine specifically index page as a word vector. When a search query is presented A search engine essentially indexes pages by representing the text as a word vector. When a search query is presented, the vector distance $\cos(\theta) \in (0,1)$ is computed for the search query with all indexed pages to find the pages with which the angle is the least, i.e., where $\cos(\theta)$ is the greatest. Presenting the best-match ordered list is made by sorting all indexed pages by their angle with the search query.

In news analytics, when using the vector distance classier for news analytics, the classification algorithm takes the new text sample and then finds the best match by computing the angle of the message with all the text pages in the indexes training corpus. After this, pages with the same tags are classified as the best matches. To implement the classifier, all that is required is only linear algebra functions and sorting out routines readily available in virtually all the programming environments.

Discriminant-Based Classier

All the classifiers we have examined so far do not weigh words differently. It is either they do not weigh the words at all, as evidenced with SVM or Bayes classifier, or they weigh some parts of the words while ignoring the other, as is the case with word count classifiers. Discriminant-Based Classifier weighs words base on their discriminant value. Among the popularly used tool for this purpose is **Fisher's Discriminant**.

In our example, we will take the mean value of each term for each category as = **μi**. The mean stands for an average number of times word **w** appears in a text message of category i. The text message itself would be index **ad j.** To evaluate the number of times word **w** occur in a text message **j** of category **i,** our for this would be **mij.** The discriminant function can be written as:

$$F(w) = \frac{\frac{1}{|C|}\sum_{i \neq k}(\mu_i - \mu_k)^2}{\sum_i \frac{1}{n_i}\sum_j (m_{ij} - \mu_i)^2}$$

We would consider the case we observe previously in this study, the economic evaluation we grouped into an optimistic and pessimistic group. Let assume the word "dismal" appears once, in the entire text, the word would be grouped as pessimistic and would not appear in the optimistic class. The across-class variation of the word is positive, while the within-class variation is zero. In this kind of situation, the denominator of the equation would be zero. We would conclude by saying that the word "dismal" is an infinitely-powerful discriminant and should be evaluated with a large weight in any word-count algorithms.

Metrics

Analytics developed without metrics is incomplete. In every developing analytics, it is important to create measures that would examine whether or not the analytics are generating classifications that are economically useful, statistically useful, and stable. However, there are some criteria every analytic must meet for it to be statistically useful. These criteria would ensure the classification power and accuracy. When an analytic is both economically useful and statistically valid, it increases the quality of the analytics. Stability helps an analytic to perform effectively in-sample and out-of-sample.

Confusion Matrix

This is a classic tool used in assessing classification accuracy. For **n** categories, the confusion matrix would be of dimension **n × n**. The column stands for the correct category of the text, while the rows represent the category given by the analytic algorithm. For each cell **(i, j)**, the number of text messages in type **j** and classified as type **i** are contained in the cell matrix. The number of times the algorithm got the correct classification is stated in the cells on the diagonal of the confusion. When this is sorted out, every other cell is a classification error. The rows and columns of the classification can only be dependent on each other if an algorithm has no classification ability. The statistics examined for rejection under the null statistics are as follows:

$$\chi^2[dof = (n-1)^2] = \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{[A(i,j) - E(i,j)]^2}{E(i,j)}$$

A(i,j) represents the numbers observed in the confusion matrix, while E(i, j) stands for the expected numbers when there is no classification under the null. If T(j) stands for the total column and (Ti) stands for the total across row i of the confusion matrix, then

$$E(i,j) = \frac{T(i) \times T(j)}{\sum_{i=1}^{n} T(i)} \equiv \frac{T(i) \times T(j)}{\sum_{j=1}^{n} T(j)}$$

(n −1)2 can be used to calculate the degree of freedom of the x2 statistics. This statistic is very easy to calculate and can be used for any n model.

Precision and Recall

Two results can emerge from the creation of the confusion matrix. They are Precision or Recall.

Precision is also known as positive predictive value. This is simply the fraction of identified positives that are really positives. It is the measurement of the validity of precision. Take, for instance, we want to find out the number of people on LinkedIn who are looking for a job if our algorithms find **n** of these kinds of people while only **m** are looking for jobs. Our precision value would be **m/n.**

Recall, on the other hand, is also known as sensitivity. This is the number of positives that are truly identified. A recall is the measure of the completeness of prediction. Using our LinkedIn example, since the value of the actual people looking for a job is **m,** our recall formula would be **m/n.** For instance, let's assume our recall confusion matrix is

	Actual		
Predicted	Looking for Job	Not Looking	
Looking for Job	10	2	12
Not Looking	1	16	17
	11	18	29

For the above confusion matrix, our value for precision is 10/12, while recall is 10/11. This implies that precision is related to the probability of false positives (Type 1 error). This is one minus precision. However, recall is related to the probability of false-positive (Type 2 error). This is simply one minus recall.

Accuracy

The measure of algorithm accuracy over a classification scheme is simply the percentage of text that is accurately classified. This measurement can be done both out-of-sample and in-sample. Here is the formula to compute this off our confusion matrix

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} A(i,i)}{\sum_{j=1}^{n} T(j)}$$

False Positives

It is better to have a failure to classify than to have an improper classification. For instance, in a 2 ×2 scheme, i.e., a two-category n=2, every off-dimension matrix in the confusion matrix is a false positive. This implies that, when n >2, it means some classification errors are worse than the other.

Calculating the percentage of false positives is a very important metric to work with. This can be calculated by dividing the total classification undertaken by the weighted count or simple count of classification.

Sentiment Error

An aggregate measure of sentiment may be computed once many texts or articles are computed. This means that aggregation is very useful to cancel classification error.

Sentiment error is simply the percentage of the value we would get when there is no classification error and the percentage difference between the computed aggregate sentiments.

Correlation

Having examined some of the vital aspects of news analysis, the question that comes to mind is, how would the sentiment from news correlate with financial time series? Leinweber and Sisk provide the explanation to this question in their paper published in 2010.

In the paper, they explained crucial differences in cumulative excess returns between strong positive sentiment and strong negative sentiment days over prediction horizons of a week or a quarter. Therefore, it can be inferred that the event studied are focused on point-in-time correlation triggers. The visual correlation metric is the simplest correlation. Here we can see how the sentiments and the returns track each other.

Phase-Lag Metrics

A unique case of lead-lag analysis is the correlation across sentiments and return time series. This may be summed up as looking for correlations in the matrix. In simple term, a graphical lead-lag analysis finds graph pattern across two series and examine if there are any ways pattern in one time series can be predicted with the other. In other words, is there a way we can use the sentiment data generated in algorithms to in-stock series. This type of graphical examination is called the phase-lag analysis.

Economic Significant

We can evaluate news analysis using economic significance as a yardstick. In using economic significance as a yardstick, we would be asking the following question, do the algorithms help reduce risk my delivering profitable opportunities? Or does it not? This kind of evaluation would help us identify a set of stocks that would perform significantly better than the other.

Economic metrics contain a lot of research and performances for news analysis. In fact, Leinweber and Sisk, in the paper published in 2010, explained that there is exploitable alpha in news streams. Economic analysis can make use of risk management and credit analysis areas to validate news analysis.

Text Summarization

Text can be easily summarized using statistics. The simplest form of text summarizer works more on the sentence-based model used in sorting the sentences in a document in descending order. When this occurs, the most overlap words are arranged first, then others followed it. For instance, let's assume an article D has a sentence si,i = 1,2,...,m. In this m sentence, each si represents a set of words. To summarize the text, we would use the 3 similarity index to compute each pairwise overlap between sentences.

To get the sentence overlap, we would find the ratio of the size of the intersection of the two sentences, so and sj, divided by the size of the union of the two sets. Next, the similarity score of every sentence is computed as the row sum of the Jaccard similarity matrix.

$$J_{ij} = J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} = J_{ji}$$

After obtaining the row sum, we sort them out; the summary is the first n sentence based on the value.

$$\mathcal{S}_i = \sum_{i=1}^{m} J_{ij}$$

Conclusion

We have explained in detail what news analysis is and how it is carried out. We examined the vital features of news analysis and the different models that can be used to carry out this analysis. Also, we examined how errors can be avoided or contained to the barest minimum when carrying out news analysis. The vital aspect of word-count was explained in detail. In the next chapter, we look at one of the important models in data science.

Chapter Eight: Bass Model

This chapter explains in detail all there is to know about Base Model. The chapter would cover the following outlines:

- The Bass Model
- Calibration
- Sales Peak

The Bass Model

The Bass Model is one of the classic models in Marketing literature. This was discovered in 1969 and has become one of the best models for predicting the market share of products that are newly introduced and even matured products. The main focus of the model is the adoption rate of a product must follow these two basic conditions:

- the propensity of customers to adopt the product without the influence of social influences
- the additional propensity that the product would be adopted because other customers have.

This is why, at some point in a very good product, the influence of the early adopters get so strong that it affects or stir others to adopt the product. Usually, this is seen as a product of the network. However, Frank Bass had already completed all there is to know about the influence of early adopters on a very good product before the advent of the network effect. That is to say, product adoption resulting from the influence of early adopters is not necessarily a product of the network.

The bass model explains in detail how the information of the first few sales of a product can be used to predict or forecast the product's future sale. Although this model seems to be more of a marketing model, it can be used to determine the value of a start-up business by analyzing the cash flow of the business.

The Basic Idea

Here, we would follow the exposition of the Bass Model. Let's take, for instance, that cumulative probability of a product by a single individual from a time zone of zero to time t is F(t). The probability of product at time t is the density function f(t)=F(t). Given that there is no purchase so far, the rate of purchase would be

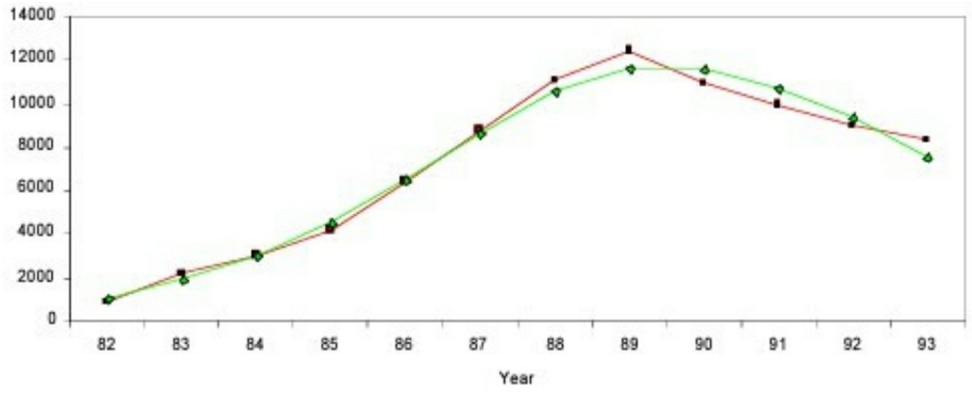
i.e.

$$\frac{f(t)}{1-F(t)}.$$

Modeling this is very similar to how we model the adoption rate of a product for a given time t. Using the Bass model, this adoption rate can be defined as :

$$\frac{f(t)}{1-F(t)} = p + q\,F(t).$$

P in the equation can be assumed to be the independent rate of a consumer adopting the product, while q is the rate of imitation. This is because it modulates the impact of the consumer adopting the product from the cumulative intensity of adoption F(t).



With our analysis of the p and q of the product, we can use our findings to forecast the adoption of the product.

Software

Free software can be used to solve an ordinary differential equation. Among the most popularly used open-source package is the Maxima. This is available for download in many places. Here is what the basic solution for differential equation in Maxima looks like:

Maxima 5.9.0 http://maxima.sourceforge.net

This was distributed under the GNU Public License. Bug reporting information is provided by the function bug_report()

```
(C1) depends(F,t);
(D1)                                [F(t)]
(C2) diff(F,t)=(1-F)*(p+q*F);
                             dF
(D2)                         -- = (1 - F) (F q + p)
                             dt
(C3) ode2(%,F,t);
                      LOG(F q + p) - LOG(F - 1)
(D3)                  ------------------------ = t + %C
                                q + p
```

Note that the function $^{1}/_{1-F}$ was processed from the left and not from the right as the software seems to be working. This is why Maxima would be used to solve the partial fraction results in simple integral. The result of this would be

```
(%i1) integrate((q/(p+q))/(p+q*F)+(1/(p+q))/(1-F),F);

                           log(q F + p)   log(1 - F)
(%o1)                      ------------ - ----------
                              q + p         q + p
```

The above result is the correct one. Another very simple tool that is effective in calculating small-scale symbolic calculation is WolframAlpha. This can be downloaded at www.wolframalpha.com.

Calibration

How do we find out the coefficient of p and q in our previous Bass model? Since we already have the current sales history of the product, this can be easily fit into the adoption curve. Below is the formula.to calculate this:

Sales in any period are: s(t) = m f(t).

Cumulative sales up to a given time t are: S(t) = m F(t)

Since we already have the formula, we will go ahead and substitute f(t) and F(t) in the Bass equation. This would give us:

$$\frac{s(t)/m}{1-S(t)/m} = p + q\, S(t)/m$$

This can be rewritten as

$$s(t) = [p+q\, S(t)/m][m-S(t)]$$

Therefore:

$$\begin{aligned} s(t) &= \beta_0 + \beta_1\, S(t) + \beta_2\, S(t)^2 \\ \beta_0 &= pm \\ \beta_1 &= q - p \\ \beta_2 &= -q/m \end{aligned}$$

We will be using this equation in another example to understand it perfectly. Now let us examine the ongoing sales for iPhone product as an example. First, we would read our quarterly sales already stored in a file; after this, we will carry out a Bass model analysis. Next, we will R code to compute it:

```
> #USING APPLE iPHONE SALES DATA
> data = read.table("iphone_sales.txt",header=TRUE)
> isales = data[,2]
> cum_isales = cumsum(isales)
> cum_isales2 = cum_isales^2
> res = lm(isales ~ cum_isales+cum_isales2)
> print(summary(res))

Call:
lm(formula = isales ~ cum_isales + cum_isales2)

Residuals:
    Min      1Q  Median      3Q     Max
-14.106  -2.877  -1.170   2.436  20.870

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.220e+00  2.194e+00   1.468   0.1533
cum_isales   1.216e-01  2.294e-02   5.301 1.22e-05 ***
cum_isales2 -6.893e-05  3.906e-05  -1.765   0.0885 .
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 7.326 on 28 degrees of freedom
Multiple R-squared:  0.854,     Adjusted R-squared:  0.8436
F-statistic: 81.89 on 2 and 28 DF,  p-value: 1.999e-12
```

Now we will fit in the model and then plot our actual sales overlaid on the forecast.

```
> #FIT THE MODEL
> m1 = (-b[2]+sqrt(b[2]^2-4*b[1]*b[3]))/(2*b[3])
> m2 = (-b[2]-sqrt(b[2]^2-4*b[1]*b[3]))/(2*b[3])
> print(c(m1,m2))
cum_isales cum_isales
 -26.09855 1790.23321
> m = max(m1,m2); print(m)
[1] 1790.233
> p = b[1]/m
> q = -m*b[3]
> print(c(p,q))
(Intercept) cum_isales2
 0.00179885  0.12339235
>
> #PLOT THE FITTED MODEL
> nqtrs = 100
> t=seq(0,nqtrs)
> fn_f = eval(ff)*m
> plot(t,fn_f,type="l")
> n = length(isales)
> lines(1:n,isales,col="red",lwd=2,lty=2)
>
```

Sales Peak

From our calculation so far, calculating the sales peak is very easy. All we need to differentiate f(t) with respect to t, and then set the result equal to zero. This is shown as follows:

$$t* = \text{argmaxt } f(t)$$

This is the same as the solution to $f^1(t)=0$.

The calculation is very simple, the formula is

$$t^* = \frac{-1}{p+q} \ln(p/q)$$

Therefore, for the values p = 0.01 and q = 0.2, we will have

$$t^* = \frac{-1}{0.01+0.2} \ln(0.01/0.2) = 14.2654 \text{ years.}$$

Now for our iPhone sales, the computation of the sales peak would give us:

```
> #PEAK SALES TIME POINT (IN QUARTERS)
> tstar = -1/(p+q)*log(p/q)
> print(tstar)
(Intercept)
   33.774111
   > length(isales)
[1] 31
```

In our calculation, we would observe that the peak happens in half a year. The number of quarter that passed before the sales peak is 31.

Conclusion

In this chapter, we carried out an extensive explanation of the Bass Model. Also, an explanation of how to use the Bass Model to determine the future of sales and calculate sales peak in business. In the next chapter, we examine how dimensions are extracted in Data science.

Chapter Nine: Extracting Dimensions: Discriminant and Factor Analysis

This chapter covers the analysis of large data sets. It explains in detail all there is to know about the analysis of large data. We would be using the two common approaches of large data analysis: Discriminant analysis and Factor analysis. The two data would help us understand the most important structural components of any big data. In discriminant analysis, for example, we would be developing models that would help us group population size into two broad components: males vs. female, immigrants versus indigene and so on. With factor analysis, we would be able to beat down large data on population into explanatory variables. Here are the outlines that would be covered in this chapter:

- Discriminant Analysis
- Notation and Assumption
- Discriminant Function
- Eigensystem
- Factor Analysis
- Difference between discriminant analysis and factor analysis
- Factor Rotation

Discriminant Analysis

Discriminant analysis is an attempt to explain categorical data by creating a dichotomous split of observations. For instance, let's assume that we want to split our large business data into two categories. One category is for the bad creditors, and the other is for the good creditors. In DA, the bad and good creditors are referred to as dependent variables or criterion variables. The variable we use in explaining the split in the criterion variable is referred to as explanatory or predictor variable. We can assume the criterion variable to be the left-hand side variables while the explanatory variable is the right-hand variables.

The significant property of DA is that left-hand variables are qualitative. This implies that aside from their numerical value, they are naturally of good qualities. A good example of how DA works is the admission process of universities and other tertiary institutions. Every university has a specific cut-off for each department a student might want to apply for. The cut-off mark is what separates the student that would be admitted from those that won't be admitted. Now, this cut-off mark is determined with the aid of DA.

In a very simple term, DA is the tool that quantitative explanatory variables are used to explain qualitative criterion variables. This does not mean that DA only works with two categories. The number of categories that we use DA is not restricted to just two; it starts from two or more.

Notation and Assumption

Let's assume that there are N groups or categories indexed by i = 2...N.

In each of the N groups, there are observations yj, indexed by j = 1...Mi. Note that the group does not necessarily need to have the same size.

We have a set of predictor or explanatory x = [x1,x2,...,xK]. There must be a valid reason for choosing this so that y can have a group where it belongs. Therefore, the value of the kth variable for group i, observation j, is denoted as xijk.

Observations must be mutually exclusive. This implies that each member of a group cannot belong to the other group.

Cov(xi) = Cov(xj). that is, the explanatory variable of all group have the same K×K covariance matrix.

Discriminant Function

The main focus of DA is to find a discriminant function that best defines and separates one group from the other. The most common approach is to use a linear DA. However, the function might be nonlinear. The function of the DA takes this formula:

$$D = a_1x_1 + a_2x_2 + \ldots + a_Kx_K = \sum_{k=1}^{K} a_kx_k$$

The discriminant weight is the **ak** coefficients.

In carrying out our analysis, we would need a score for the cut-off C. Take, for instance, N = 2. This implies that there are 2 groups, the observation would fall into group one if D >C while it will fall into group two if D ≤ C.

Therefore, the **objective** function is to select {{ak}, C} in such a way that classification error is reduced.

The formula C = D({xk};{ak} is the formula of a **hyperplane** that splits the observation space into different parts depending on the number of groups we are dealing with. If we are dealing with two groups, the space of observation would be split into two parts.

Implementation with R

In this section, we would be using data for the top 64 teams in the 2005-06 NCAA tournament and then implement a discriminant function model on the data. The data for the program is as follows:

	GMS	PTS	REB	AST	TO	A.T	STL	BLK	PF	FG	FT	X3P
1	6	84.2	41.5	17.8	12.8	1.39	6.7	3.8	16.7	0.514	0.664	0.417
2	6	74.5	34.0	19.0	10.2	1.87	8.0	1.7	16.5	0.457	0.753	0.361
3	5	77.4	35.4	13.6	11.0	1.24	5.4	4.2	16.6	0.479	0.702	0.376
4	5	80.8	37.8	13.0	12.6	1.03	8.4	2.4	19.8	0.445	0.783	0.329
5	4	79.8	35.0	15.8	14.5	1.09	6.0	6.5	13.3	0.542	0.759	0.397
6	4	72.8	32.3	12.8	13.5	0.94	7.3	3.5	19.5	0.510	0.663	0.400
7	4	68.8	31.0	13.0	11.3	1.16	3.8	0.8	14.0	0.467	0.753	0.429
8	4	81.0	28.5	19.0	14.8	1.29	6.8	3.5	18.8	0.509	0.762	0.467
9	3	62.7	36.0	8.3	15.3	0.54	8.0	4.7	19.7	0.407	0.716	0.328
10	3	65.3	26.7	13.0	14.0	0.93	11.3	5.7	17.7	0.409	0.827	0.377
11	3	75.3	29.0	16.0	13.0	1.23	8.0	0.3	17.7	0.483	0.827	0.476
12	3	65.7	41.3	8.7	14.3	0.60	9.3	4.3	19.7	0.360	0.692	0.279
13	3	59.7	34.7	13.3	16.7	0.80	4.7	2.0	17.3	0.472	0.579	0.357
14	3	88.0	33.3	17.0	11.3	1.50	6.7	1.3	19.7	0.508	0.696	0.358

15	3	76.3	27.7	16.3	11.7	1.40	7.0	3.0	18.7	0.457	0.750	0.405
16	3	69.7	32.7	16.3	12.3	1.32	8.3	1.3	14.3	0.509	0.646	0.308
17	2	72.5	33.5	15.0	14.5	1.03	8.5	2.0	22.5	0.390	0.667	0.283
18	2	69.5	37.0	13.0	13.5	0.96	5.0	5.0	14.5	0.464	0.744	0.250
19	2	66.0	33.0	12.0	17.5	0.69	8.5	6.0	25.5	0.387	0.818	0.341
20	2	67.0	32.0	11.0	12.0	0.92	8.5	1.5	21.5	0.440	0.781	0.406
21	2	64.5	43.0	15.5	15.0	1.03	10.0	5.0	20.0	0.391	0.528	0.286
22	2	71.0	30.5	13.0	10.5	1.24	8.0	1.0	25.0	0.410	0.818	0.323
23	2	80.0	38.5	20.0	20.5	0.98	7.0	4.0	18.0	0.520	0.700	0.522
24	2	87.5	41.5	19.5	16.5	1.18	8.5	2.5	20.0	0.465	0.667	0.333
25	2	71.0	40.5	9.5	10.5	0.90	8.5	3.0	19.0	0.393	0.794	0.156
26	2	60.5	35.5	9.5	12.5	0.76	7.0	0.0	15.5	0.341	0.760	0.326
27	2	79.0	33.0	14.0	10.0	1.40	3.0	1.0	18.0	0.459	0.700	0.409
28	2	74.0	39.0	11.0	9.5	1.16	5.0	5.5	19.0	0.437	0.660	0.433
29	2	63.0	29.5	15.0	9.5	1.58	7.0	1.5	22.5	0.429	0.767	0.283
30	2	68.0	36.5	14.0	9.0	1.56	4.5	6.0	19.0	0.398	0.634	0.364
31	2	71.5	42.0	13.5	11.5	1.17	3.5	3.0	15.5	0.463	0.600	0.241
32	2	60.0	40.5	10.5	11.0	0.95	7.0	4.0	15.5	0.371	0.651	0.261
33	2	73.5	32.5	13.0	13.5	0.96	5.5	1.0	15.0	0.470	0.684	0.433
34	1	70.0	30.0	9.0	5.0	1.80	6.0	3.0	19.0	0.381	0.720	0.222
35	1	66.0	27.0	16.0	13.0	1.23	5.0	2.0	15.0	0.433	0.533	0.300
36	1	68.0	34.0	19.0	14.0	1.36	9.0	4.0	20.0	0.446	0.250	0.375
37	1	68.0	42.0	10.0	21.0	0.48	6.0	5.0	26.0	0.359	0.727	0.194
38	1	53.0	41.0	8.0	17.0	0.47	9.0	1.0	18.0	0.333	0.600	0.217
39	1	77.0	33.0	15.0	18.0	0.83	5.0	0.0	16.0	0.508	0.250	0.450
40	1	61.0	27.0	12.0	17.0	0.71	8.0	3.0	16.0	0.420	0.846	0.400
41	1	55.0	42.0	11.0	17.0	0.65	6.0	3.0	19.0	0.404	0.455	0.250
42	1	47.0	35.0	6.0	17.0	0.35	9.0	4.0	20.0	0.298	0.750	0.160
43	1	57.0	37.0	8.0	24.0	0.33	9.0	3.0	12.0	0.418	0.889	0.250
44	1	62.0	33.0	8.0	20.0	0.40	8.0	5.0	21.0	0.391	0.654	0.500
45	1	65.0	34.0	17.0	17.0	1.00	11.0	2.0	19.0	0.352	0.500	0.333
46	1	71.0	30.0	10.0	10.0	1.00	7.0	3.0	20.0	0.424	0.722	0.348
47	1	54.0	35.0	12.0	22.0	0.55	5.0	1.0	19.0	0.404	0.667	0.300
48	1	57.0	40.0	2.0	5.0	0.40	5.0	6.0	16.0	0.353	0.667	0.500
49	1	81.0	30.0	13.0	15.0	0.87	9.0	1.0	29.0	0.426	0.846	0.350
50	1	62.0	37.0	14.0	18.0	0.78	7.0	0.0	21.0	0.453	0.556	0.333
51	1	67.0	37.0	12.0	16.0	0.75	8.0	2.0	16.0	0.353	0.867	0.214
52	1	53.0	32.0	15.0	12.0	1.25	6.0	3.0	16.0	0.364	0.600	0.368

```
53   1 73.0 34.0 17.0 19.0 0.89  3.0 3.0 20.0 0.520 0.750 0.391
54   1 71.0 29.0 16.0 10.0 1.60 10.0 6.0 21.0 0.344 0.857 0.393
55   1 46.0 30.0 10.0 11.0 0.91  3.0 1.0 23.0 0.365 0.500 0.333
56   1 64.0 35.0 14.0 17.0 0.82  5.0 1.0 20.0 0.441 0.545 0.333
57   1 64.0 43.0  5.0 11.0 0.45  6.0 1.0 20.0 0.339 0.760 0.294
58   1 63.0 34.0 14.0 13.0 1.08  5.0 3.0 15.0 0.435 0.815 0.091
59   1 63.0 36.0 11.0 20.0 0.55  8.0 2.0 18.0 0.397 0.643 0.381
60   1 52.0 35.0  8.0  8.0 1.00  4.0 2.0 15.0 0.415 0.500 0.235
61   1 50.0 19.0 10.0 17.0 0.59 12.0 2.0 22.0 0.444 0.700 0.300
62   1 56.0 42.0  3.0 20.0 0.15  2.0 2.0 17.0 0.333 0.818 0.200
63   1 54.0 22.0 13.0 10.0 1.30  6.0 1.0 20.0 0.415 0.889 0.222
64   1 64.0 36.0 16.0 13.0 1.23  4.0 0.0 19.0 0.367 0.833 0.385
```

Now we will run some of the command stored in the **lda.R** on the program

```
ncaa = read.table("ncaa.txt",header=TRUE)
x = as.matrix(ncaa[4:14])
y1 = 1:32
y1 = y1*0+1
y2 = y1*0
y = c(y1,y2)

library(MASS)
dm = lda(y~x)
```

Therefore the first 32 members of the team would form our category 1 (y=1), the last 32 would form the category 2 (y=0). The result of our discriminant analysis is:

```
> lda(y~x)
Call:
lda(y ~ x)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
      xPTS     xREB     xAST      xTO     xA.T     xSTL  xBLK     xPF
0 62.10938 33.85938 11.46875 15.01562 0.835625 6.609375 2.375 18.84375
1 72.09375 35.07500 14.02812 12.90000 1.120000 7.037500 3.125 18.46875
       xFG       xFT       xX3P
0 0.4001562 0.6685313 0.3142187
1 0.4464375 0.7144063 0.3525313

Coefficients of linear discriminants:
```

We can extract some useful result as follows:

```
> result = lda(y~x)
> result$prior
  0   1
0.5 0.5
> result$means
      xPTS     xREB     xAST      xTO     xA.T     xSTL  xBLK     xPF
0 62.10938 33.85938 11.46875 15.01562 0.835625 6.609375 2.375 18.84375
1 72.09375 35.07500 14.02812 12.90000 1.120000 7.037500 3.125 18.46875
       xFG       xFT       xX3P
0 0.4001562 0.6685313 0.3142187
1 0.4464375 0.7144063 0.3525313
> result$call
lda(formula = y ~ x)
> result$N
[1] 64
> result$svd
[1] 7.942264
```

The singular value decomposition value is contained in the last line. This is also the Fisher's discriminant level that provides the ratio of the between-and-within group standard deviation on the linear discriminant variables. The squares of these linear discriminant values are the canonical F-statistics.

Confusion Matrix

The confusion matrix has been explained previously in this study. It is a tabular presentation of both actual and predicted values. The following R command would be used to generate the confusion matrix for the basketball team in our previous example.

```
> result = lda(y~x)
> y_pred = predict(result)$class
> length(y_pred)
[1] 64
> table(y,y_pred)
   y_pred
y    0  1
  0 27  5
  1  5 27
```

In the above command, we would observe that both 5 and 64 have been wrongly classified. To assess this, we would be computing the x^2 statistics for the confusion matrix. To do this, we would first define the confusion matrix as

$$E = \begin{bmatrix} 16 & 16 \\ 16 & 16 \end{bmatrix}$$

The above matrix shows some classification ability. However, what happens when our model does not have any confusion ability? It means that our matrix would have no relationship between the rows and columns; hence the average number would be drawn based on the total of rows and columns. Since the total of rows and columns for our program is 32, our matrix with no confusion ability would look like this:

$$A = \begin{bmatrix} 27 & 5 \\ 5 & 27 \end{bmatrix}$$

The total number of squared normalized differences in the cell of an individual matrix is **Text Statistics**. The formula for this is:

$$\text{Test-Stat} = \sum_{i,j} \frac{[A_{ij} - E_{ij}]^2}{E_{ij}}$$

Splitting into Multiple Groups

If we want to split our NCAA team into groups, for instance, we want to split the group into four, we simply used the following commands:

```
> y1 = rep(3,16)
> y2 = rep(2,16)
> y3 = rep(1,16)
> y4 = rep(0,16)
> y = c(y1,y2,y3,y4)
> res = lda(y~x)
> res
Call:
lda(y ~ x)

Prior probabilities of groups:
   0    1    2    3
0.25 0.25 0.25 0.25

Group means:
      xPTS     xREB     xAST      xTO      xA.T    xSTL    xBLK     xPF       xFG
0 61.43750 33.18750 11.93750 14.37500 0.888750 6.12500 1.8750 19.5000 0.4006875
1 62.78125 34.53125 11.00000 15.65625 0.782500 7.09375 2.8750 18.1875 0.3996250
2 70.31250 36.59375 13.50000 12.71875 1.094375 6.84375 3.1875 19.4375 0.4223750
3 73.87500 33.55625 14.55625 13.08125 1.145625 7.23125 3.0625 17.5000 0.4705000
       xFT       xX3P
0 0.7174375 0.3014375
1 0.6196250 0.3270000
2 0.7055625 0.3260625
3 0.7232500 0.3790000

Coefficients of linear discriminants:
              LD1          LD2          LD3
xPTS  -0.03190376 -0.09589269 -0.03170138
xREB   0.16962627  0.08677669 -0.11932275
xAST   0.08820048  0.47175998  0.04601283
xTO   -0.20264768 -0.29407195 -0.02550334
xA.T   0.02619042 -3.28901817 -1.42081485
xSTL   0.23954511 -0.26327278 -0.02694612
xBLK   0.05424102 -0.14766348 -0.17703174
xPF    0.03678799  0.22610347 -0.09608475
xFG   21.25583140  0.48722022  9.50234314
xFT    5.42057568  6.39065311  2.72767409
xX3P   1.98050128 -2.74869782  0.90901853

Proportion of trace:
   LD1    LD2    LD3
0.6025 0.3101 0.0873
> predict(res)$class
 [1] 3 3 3 3 3 3 3 3 3 1 3 3 2 0 3 3 3 0 3 2 3 2 2 3 2 2 0 2 2 2 2 2 2 3 1 1 1 0 1
[39] 1 1 1 1 1 1 1 1 1 0 2 2 0 0 0 0 2 0 0 2 0 1 0 1 1 0 0
Levels: 0 1 2 3
> y
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1
[40] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> y_pred = predict(res)$class
> table(y,y_pred)
   y_pred
y    0  1  2  3
  0 10  3  3  0
  1  2 12  1  1
  2  2  0 11  3
  3  1  1  1 13
```

Eigen Systems

Here, we will be exploring some components of matrices that would help us in data classification. To get started, we would first download the Treasury Interest rate date from the FRED website: http://research.stlouisfed.org/fred2/. This can be assessed in a file named tryrates.txt. After this, we simply read the file

```
> rates = read.table("tryrates.txt",header=TRUE)
> names(rates)
[1] "DATE"    "FYGM3"   "FYGM6"   "FYGT1"   "FYGT2"   "FYGT3"   "FYGT5"   "FYGT7"
```

An M×M matrix A has attendant M eigenvectors V and eigenvalue λ if we can write

$\lambda V = AV$

Beginning with the A matrix, the decomposition for the eigenvalue would be both V and λ. For the Matrix M, we would be finding both the eigenvalue and eigenvectors since there is no explanation or equation for this. Hence we would require that λ=0. Then we would set Matrix A as the covariance matrix for the rates of different maturities.

```
> eigen(cov(rates))
$values
[1] 7.070996e+01 1.655049e+00 9.015819e-02 1.655911e-02 3.001199e-03
[6] 2.145993e-03 1.597282e-03 8.562439e-04

$vectors
           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
[1,] -0.3596990 -0.49201202  0.59353257 -0.38686589 -0.34419189 -0.07045281
[2,] -0.3581944 -0.40372601  0.06355170  0.20153645  0.79515713  0.07823632
[3,] -0.3875117 -0.28678312 -0.30984414  0.61694982 -0.45913099  0.20442661
[4,] -0.3753168 -0.01733899 -0.45669522 -0.19416861  0.03906518 -0.46590654
[5,] -0.3614653  0.13461055 -0.36505588 -0.41827644 -0.06076305 -0.14203743
[6,] -0.3405515  0.31741378 -0.01159915 -0.18845999 -0.03366277  0.72373049
[7,] -0.3260941  0.40838395  0.19017973 -0.05000002  0.16835391  0.09196861
[8,] -0.3135530  0.47616732  0.41174955  0.42239432 -0.06132982 -0.42147082
           [,7]        [,8]
[1,]  0.04282858  0.03645143
[2,] -0.15571962 -0.03744201
[3,]  0.10492279 -0.16540673
[4,]  0.30395044  0.54916644
[5,] -0.45521861 -0.55849003
[6,] -0.19935685  0.42773742
[7,]  0.70469469 -0.39347299
[8,] -0.35631546  0.13650940

> rcorr = cor(rates)
> rcorr
          FYGM3     FYGM6     FYGT1     FYGT2     FYGT3     FYGT5     FYGT7
FYGM3 1.0000000 0.9975369 0.9911255 0.9750889 0.9612253 0.9383289 0.9220409
FYGM6 0.9975369 1.0000000 0.9973496 0.9851248 0.9728437 0.9512659 0.9356033
FYGT1 0.9911255 0.9973496 1.0000000 0.9936959 0.9846924 0.9668591 0.9531304
FYGT2 0.9750889 0.9851248 0.9936959 1.0000000 0.9977673 0.9878921 0.9786511
FYGT3 0.9612253 0.9728437 0.9846924 0.9977673 1.0000000 0.9956215 0.9894029
FYGT5 0.9383289 0.9512659 0.9668591 0.9878921 0.9956215 1.0000000 0.9984354
FYGT7 0.9220409 0.9356033 0.9531304 0.9786511 0.9894029 0.9984354 1.0000000
FYGT10 0.9065636 0.9205419 0.9396863 0.9680926 0.9813066 0.9945691 0.9984927
         FYGT10
FYGM3 0.9065636
FYGM6 0.9205419
FYGT1 0.9396863
FYGT2 0.9680926
FYGT3 0.9813066
FYGT5 0.9945691
FYGT7 0.9984927
FYGT10 1.0000000
```

Next, we calculate the eigenvectors and the eigenvalues of the covariance matrix. Let's assume the covariance matrix is the total of the rate of connection between the rates of maturities in our data set. However, we don't know the number of dimensions present in this data. For each dimension of commonality, our focus is on the importance of the dimension (eigenvalue) and the influence of the dimension on each rate (value of eigenvector). The highest eigenvalue is the most important dimension. This is also known as "principal eigenvalue" with its corresponding principal eigenvector. The eigenvalue and the eigenvector are what make up the Eigen pair. This is why it is called the "eigenvalue decomposition matrix."

Factor Analysis

This is simply the use of eigenvalue decomposition in finding the basic structure of data. When we have a data set of both observation and explanatory variables, we use factor analysis to achieve decomposition of these two properties:

First, generate a reduced dimension set of explanatory variables. This is also known as extracted, derived, or discovered factors. The generated factors must be uncorrelated with each other.

Generate data reduction. This implies that you suggest a limited set of variables. For our set of variables, each subset is the manifestation of an abstract underlying dimension.

Notation

Original explanatory variables: xik,k = 1...K.

Observations: yi,i = 1...N.

Factors: Fj, j = 1...M.

M<K.

The Idea

We will notice from our observation of the rate data that there are eight different kinds of rates. Now to model the data of each of the eight data, we would need a separate driver leading to K = 8 underlying factor. However, this would go against the essence of factor analysis. Factor analysis aims at reducing the number of existing drivers. As a result, we would go with a smaller value of M < K factors.

The most important focus here is to project the variables $x \in RK$ onto the reduced factor set $F \in RM$. This would help us explain most of the variables by the factors. Therefore, what we are looking for is a relation:

$$x = BF$$

B in the equation stands for $B = \{bkj\} \in RK \times M$ is a matrix of factor "loadings" for the variables. With B matrix, we can represent x in smaller dimension M. The entries in matrix B can either be negative or positive. When the entries are negative, it means that the entries are negatively correlated with the factors. When it is positive, it is positively correlated. We aim to use the relation of y to a reduced F to replace the correlation of y to x.

Once the set of factors have been defined, the N observations y can be expressed in terms of factor through a factor score matrix $A = \{aij\} \in RN \times M$ in this form

$$y = AF$$

Also, the factor score can either be positive or negative. We have different ways in which transformation from factors to variables can be carried out. We would observe some of these ways as we progress.

Principal Components Analysis (PCA)

Principal Components Analysis take each component and view it as a weighted combination of the other variables. Although this is not how Factor Analysis implementation is carried out, it is one of the most popular ways in Factor Analysis.

The covariance matrix is the starting point of PCA. The main focus of the PCA is to extract the principal eigenvector by using the eigenvalue analysis of the matrix. The analysis can be done using the R command. Below is an example of this action:

```
> ncaa = read.table("ncaa.txt",header=TRUE)
> x = ncaa[4:14]
> result = princomp(x)
> screeplot(result)
> screeplot(result,type="lines")
```

The results are as follows:

```
> summary(result)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
Standard deviation     9.8747703 5.2870154 3.9577315 3.19879732 2.43526651
Proportion of Variance 0.5951046 0.1705927 0.0955943 0.06244717 0.03619364
Cumulative Proportion  0.5951046 0.7656973 0.8612916 0.92373878 0.95993242
                          Comp.6     Comp.7     Comp.8      Comp.9
Standard deviation     2.04505010 1.53272256 0.1314860827 1.062179e-01
Proportion of Variance 0.02552391 0.01433727 0.0001055113 6.885489e-05
Cumulative Proportion  0.98545633 0.99979360 0.9998991100 9.999680e-01
                          Comp.10      Comp.11
Standard deviation     6.591218e-02 3.007832e-02
Proportion of Variance 2.651372e-05 5.521365e-06
Cumulative Proportion  9.999945e-01 1.000000e-00
```

The Difference Between FA and PCA

The major difference between PCA and FA is that, for computational purposes, PCA assumes that all variables are common with all unique factors equal set to zero, while FA assumes that there is some unique variance. Also, PCA can be taken as a subset of FA. The FA model that is chosen, determines the level of unique variance. Hence we can summarize by saying the FA model is an open system while PCA is a closed system. FA factors focus on decomposing the correlation matrix into both unique portions and common ones.

Factor Rotation

When the factors are rotated, it sometimes makes the variables load better on the factors. Most times, the factors carry out this function automatically. This kind of action is called factor rotation.

Here are the steps to factor rotation

Recall that x variables were decomposed as follows:

$$x = BF + e$$

Since x is dimension K, $B \in R^{K \times M}$, $F \in R^{M}$, and e is a K-dimension vector. It means that $Cov(x) = BB + \psi$

Recall that the matrix of factor loading is B. If this is the case, the system remains unchanged if BG replaces b. Here $G \in R^{M \times M}$ and G are orthogonal. Next, we call G a "rotation" of B.

Conclusion

Discriminant Analysis and Factor Analysis are two important models that have made the extraction of big data an easy feat. This chapter has provided an extensive explanation of these two factors. In the next chapter, we examine an interesting topic in Data science: Auctions.

Chapter 10: Auction

This chapter examines the various types of auction formats we have and the different principles of revenue maximization and bidding theories associated with the auction. The outlines that would be covered in this chapter include:

- Auction
- Types of Auction
- How to determine the values of auction
- Types of Bidders
- Benchmark model
- Auction Math
- UPA and DPA
- Clicks

Auctions

Auction involves some of the oldest market forms today, but are still widely used for marketing mechanism and selling off assets. Hal Varian, Chief Economist at Google (NYT, Aug 1, 2002), gave a very interesting definition of an auction, he stated that “Auctions, one of the oldest ways to buy and sell, have been reborn and revitalized on the Internet.

One of the most popular online computer-managed auction is eBay. Because of its great advantage and its economic value, it has become relatively popular. An online computer-managed auction can be used for marketing almost any product.

Features in Auctions

There are various features in an auction, but the most important is the information asymmetric between seller and buyer. Although the basic assumption in the auction is that the seller knows more about the product than the buyer, it is not uncommon for buyers to have different information about the product. This is often because buyers always take note of negative information from other bidders. In this chapter, we would examine how information asymmetric plays a great role in bidding products.

Equally significant is the fact that the market mechanism for auction is relatively explicit. This means prices and revenue are direct consequences of the auction design. In contrast to this, other market mechanisms are usually more implicit than explicit. A very common example is the case of commodities. Here a market mechanism is based on demand and supply.

Examples of Auctions

We have various examples of an active auction, some of these include auction of arts and valuables, Google ad auctions, Treasury Securities, eBay, and even the New York Stock Exchange. All these are good examples of a continuous call auction. Depending on the product being auctioned off, an auction can be either single units or multiple. A good example of a single unit auction is arts, while Treasury Securities is an example of multiple auctions.

Types of Auctions

The main types of the auction include:

English (E), the highest bidder wins. This is an open kind of auction. It is called an open auction because the progression of bids is revealed to the participants. Generally, the price of products is in ascending order.

Dutch (D). This is also an open kind of auction. However, product prices in this type of auction are in descending order. The auctioneer starts from the highest prices to the lowest. The winner of the bid is usually the first bidder.

Ist price sealed auction (iP). Here, the bid is sealed and not revealed. The winner of the bidder is the highest bidder.

2nd price sealed bid (2P): this is very similar to the (iP). However, the only difference between the two is that unlike (iP) where the first highest bidder wins, here the second-highest bidder

wins.

Anglo-Dutch (AD): this type of auction starts off as an open auction but gets sealed when it is left with only two bidders.

How To Determine The Value Of An Auction

The two most important aspects of an auction are the value and the price. However, the value of a product to be auctioned can only be determined by the nature of the product. Here are two of the model to determine the value of a product being auctioned:

Independent private values model: this model states that the individual bidder determines the valuation of the product. This is very common with an art auction

Common-values model: Here, the bidders aim to discover the common price of the product being auction. This is because there is usually an after a market where common value is traded. A good example of this auction model is Treasury Securities.

Bidder Types

The types of the bidder and the assumption made by the bidders about the product determines the revenue that would be generated from the auction. There are two major types of bidder:

Symmetric: In this type of bidder, the bidders share the same probability distribution of products and stop-out (SP) prices. Stop-out price means the price of the lowest winning bid for the last unit sold. This assumption is very good when the competition is high.

Non-symmetric or Asymmetric. This bidder type has different values distribution. This usually occurs where the market is segmented. A good example is the bidding of firms in the M&A deal.

Benchmark Model (BM)

Benchmark Model is the simplest model that can be used to analyze auction. This model is based on four major assumptions. The assumptions are explained below:

Risk-neutrality of bidders: this implies that utility function is not needed in the analysis of auction

Private-values model: here, all bidders are welcome to their own reserved value for the products. This implies that there is a distribution of bidder's private value.

Symmetric bidders: all bidders the same distribution of product value. This was already explained in the types of bidders.

Winners' payment is based on bids alone.

Properties and Results of Benchmark Model

D = iP, that is, the Ist price and Dutch auction type are equivalent to bidders. This is because, in each auction type, the bidder has to choose how high or low he or she would bid without the knowledge of other bidders

In Benchmark Model, the most important thing is to bid according to how valuable the product is to you. This is obvious in D and iP because both mechanisms do not entail bidders seeing any other lower bids. Hence the bidder bids according to how valuable the product is to him or her and watch if the bid wins. In other mechanisms like the 2P, when you bid too high, you overplay, and when you bid too low, you lose. The best way to bid is according to how valuable the product is to you. For the E-auction mechanism, it is advisable to keep bidding until the price cross your level of valuation.

Equilibrium types:

Dominant: This is a situation whereby bidders bid with respect to their true valuation of the product, not minding what other bidders are bidding. Satisfied by E and 2P.

Nash: here, bids are chosen according to the best guess of other bidders' bid and hence satisfied by D and iP.

Auction Math

Now we will move away from the theoretical explanation of auction and apply some auction equilibrium formula. To start with, **F** would be the probability distribution of the bids while **vi** is the true value of the **ith** bidder on a **0** and **1** continuum. Let's say that we rank bidders in order of their true valuation **vi**. How then do we define **F(vi)**? Let's take for instance, that the bid is drawn from a beta distribution F on v $\in$ (0,1) so that the probability of a very low bid and a very high bid is lower than a bid around the mean of the distribution. Our for the expected difference between the first and second highest bidder **v1** and **v2** is:

$$D = [1 - F(v2)](v1 - v2)$$

This implies that the difference between the first and second bids would be multiplied by the probability that v2 is the second-highest bidder. Or we assume the probability to be that there is a higher bidder than v2. Now, from the first-order condition, i.e., the sellers' point of view, our formula is:

$$\frac{\partial D}{\partial v_1} = [1 - F(v_2)] - (v_1 - v_2)F'(v_1) = 0$$

Given that bidders are symmetric in BM, v1 $\equiv$ d v2. $\equiv$ d means equivalent in distribution. This means that:

$$v_1 - v_2 = \frac{1 - F(v_1)}{f(v_1)}$$

Since the expected revenue is equivalent to the expected second price, we would rearrange the equation to get our equation for the second price:

$$v_2 = v_1 - \frac{1 - F(v_1)}{f(v_1)}$$

Optimization By Bidders

The main aim of any bidder i is to find out the function/bidding rule B that is a function of the private value **vi** such that

bi = B(vi)

In the above equation, bi stands for the actual bidder; when there is any n bidder, we will have

$$\begin{aligned}\Pr[\text{bidder } i \text{ wins}] &= \Pr[b_i > B(v_j)], \quad \forall j \neq i, \\ &= [F(B^{-1}(b_i))]^{n-1}\end{aligned}$$

The goal of each bidder is to maximize his or her expected profit in relation to the true valuation of the product. This is:

$$\pi_i = (v_i - b_i)[F(B^{-1}(b_i))]^{n-1} = (v_i - b_i)[F(v_i)]^{n-1},$$

Now we are going to invoke the notion of bidder symmetry. The first step to this is to optimize by taking ∂πi/∂bi = 0. We can only arrive at this optimization formula by first taking the sum of all derivative of profit relative to the bidder's value like this:

$$\frac{d\pi_i}{dv_i} = \frac{\partial \pi_i}{\partial v_i} + \frac{\partial \pi_i}{\partial b_i}\frac{db_i}{dv_i} = \frac{\partial \pi_i}{\partial v_i}$$

Since ∂πi/∂bi = 0, the partial derivative of profit with respect to personal valuation is reduced.

The partial derivation is taken from this equation:

$$\frac{\partial \pi_i}{\partial v_i} = [F(B^{-1}(b_i))]^{n-1}$$

Next, we will take vi as the lowest bid, then integrate the two former equations to get:

$$\pi_i = \int_{v_l}^{v_i} [F(x)]^{n-1}\, dx$$

When we equate the formula for the value of expected profit, we would have:

$$b_i = v_i - \frac{\int_{v_l}^{v_i} [F(x)]^{n-1}\, dx}{[F(v_i)]^{n-1}} = B(v_i)$$

Assuming F is a uniform, we would have:

$$B(v) = \frac{(n-1)v}{n}$$

We will observe that our bid is shaded slightly from our personal valuation. This implies that we

bid less than the true value of the product; this would give room for profit. Bids are increased as the level. to personal value increase and bidders increase, that is :

$$\frac{\partial B}{\partial v_i} > 0, \qquad \frac{\partial B}{\partial n} > 0$$

Treasury Auctions

Our previous explanation is based on a single unit auction. In this section, we would be moving from a single unit to one of the most popular multiple units, Treasury auctions. Treasury auctions are the mechanisms employed by the government and other similar bodies to issue its bills, bonds, and notes. Usually, an auction is performed on Wednesday. This implies that bids are received up until the afternoon of the day it is to be auctioned. After which the quantities requested are supplied to the top bidders until there is no remaining supply of securities. Before the auction or trade, Treasury auction is referred to as pre-market or when-issued. It is in this market that bidders get an indication of prices that might result in a tighter clustering of the bid in the auction.

Treasury auction is made up of two broad types of dealers: the small independent dealers and the primary dealers. The primary dealers entail investment houses, big banks, and so on. Most times, the auction is played among the primary dealers. The primary dealers place competitive prices on the item to be auction while the small independent dealers play with non-competitive prices.

Usually, the value placed on the item being auctioned is based on information about the secondary market of the item. The secondary market occurs immediately after the primary market. This implies that the assumed profit the item is likely to attract at the secondary market is what influenced the bidder's price at the auction. The likely price of the item at the secondary market is usually gotten from the when-issued market.

The winner at the Treasury Securities often leaves with more regret than pleasure because he or she is aware that he has bid with more money that is overplayed. In Treasury securities, this phenomenon is referred to as the "winners curse." Before the auction takes place, the fed government and other participants in the Treasury Securities try to mitigate the winner's curse. This is because someone with less propensity of regret would bid at a higher price.

UPA or DPA

UPA stands for "uniform price auction," while DPA stands for "discriminating price auction." DPA is mostly used and more preferred in Treasury Securities while UPA is only introduced recently. For DPA, the highest bidder gets his bid quantity at the price he or she bids for. The next highest bidder gets the same and this continues until the last bidder and the last item. This implies that in Treasury Securities, each winning is filled at a price, hence bidding price varies. This is what is known and refer to as discriminating in price.

However, for UPA, the highest bidder gets his or her bid quantity at the stop-out price, i.e., the price of the last winning. The next highest bidder also gets the same until the Treasury Securities supply is exhausted. This means that UPA uses single-price auction.

Although DPA tends to yield more revenue, UPA has shown to be more successful. This is because the winners' curse is mitigated in UPA. All bidders bid the same, unlike DPA were to win, you would have to pay more than other bidders.

Mechanism Design

To achieve a good auction mechanism, consider the following:

- The starting price of the item to be auctioned off.
- Is collusion contained to the barest minimum?
- Is there a truthful value revelation? This is also referred to as "truthful bidding."
- Is the product efficient? that is, the maximization of utility of auctioneer and bidders
- Is it too expensive to play?
- Fairness to both sides, whether big or small, high or low.

Clicks (Advertising Auctions)

One of the popular program that allows easy creation of advertisements that would appear on important sites like the Google search result page and other related sites is the Google AdWords program. Google AdWords is different from the Google AdSense program. Google AdSense is the one that delivers Google AdWords to other sites. Depending on the type of ad displayed on the site, Google pays web publishers based on the number of clicks and the number of impressions gathered by the ad.

In this section, we would be explaining some of the basic features of a search engine advertisement model using the research paper written by Aggarwal, Goel, and Motwani (2006).

There are three stages of search page experience used by the search engine advertisement program. These stages include cost per click (CPC), cost by thousand views (CPM) and cost by acquisition(CPA). Among these three, CPC is the most widely used. Under CPC, we have two models:

a). Revenue ranking (the Google Model)

b). Direct ranking (the overture model).

The merchant pays the next click price. This price is different from that of the second auction. However, this statement is not so in revenue ranking, as would be seen in our example.

Asymmetric: there is no incentive to overbid, only to underbid

Iterative: this is a situation whereby a bidder places many bids and watch for the response of these bids. The reason for this is to uncover the ordering of bids by other bidders. But this is not as simple as it sounds. In fact, Google often provides the Google Bid Stimulator or GBS so that sellers can easily use AdWords to figure out optimal bids.

The utility of auctioneers and merchant will be maximized if revenue ranking is true. This is known as auction efficiency

Innovation: the laddered auction. Randomized weight is attached to bids. This implies that if the sum of weight is 1, then the type of ranking used is direct ranking. However, if it is CTR, i.e., Click Through Rate, then the type of ranking used is revenue-based, revenue ranking.

The following steps highlighted below can be used by merchants to figure out the maximum cost per click(CPA) of each.

Maximum Profitable CPA. This is simply the margin of profit on the product. For instance, if the cost price of the product is #200 and the selling price is #300, the profit margin is simply #100. This is also the maximum amount a seller would pay for the CPA.

Conversion Rate (CR). CR is simply the rate gathered based on the number of times a click results in a sale. To calculate this, we simply divide the number of sales by the number of clicks. For instance, if for every 100 clicks, 5 sales are recorded. The CR is 5%.

Value Per Click (VPR). This is simply the CR multiplied by CPA. Using our example, our VPR is simply 005 × 100 = #5

Determine the profit-maximizing CPC bid. The more the bid reduces, the more the number of clicks reduce and the more the CPC and revenue reduce. However, this might not affect the profit because it is possible for the profit after acquisition to rise. We can easily use the Google Bid Simulator to find the number of clicks expected at every turn. Also, it is important to note that the price you bid is not the same as the price for a click. This is because it is based on revenue ranking, i.e., a next-price auction. Hence, the Google model determines the actual price that would be paid. The equation for profits is :

(VPC − CPC) × #Clicks = (CPA× CR − CPC) × #Clicks

Therefore, for a #4 bid, the profit would be:

(5 −407.02/154) ×154 = $362.98

Next-Price Auction

The CPC of the next-price auction is based on the price of the click right after a bid is made. This implies that, if, for instance, the winning bid is for position j on the search screen, the price paid is that of the winning bid at position j +1.

Laddered Auction

The main idea of a laddered auction is to set the position of CPC as:

$$p_i = \sum_{j=i}^{K} \left(\frac{CTR_{i,j} - CTR_{i,j+1}}{CTR_{i,i}} \right) \frac{w_{j+1} b_{j+1}}{w_i}, \quad 1 \le i \le K$$

so that

$$\frac{\#Clicks_i}{\#Impressions_i} \times p_i = CTR_{ii} \times p_i = \sum_{j=i}^{K} \left(CTR_{i,j} - CTR_{i,j+1} \right) \frac{w_{j+1} b_{j+1}}{w_i}$$

The expected revenue to Google by ad impression is the **lhs**. This model aims to maximize revenue for Google and at the same time, make the auction system very easy and effective for merchants. If the result of the laddered auction is a truthful equilibrium, this is a good one for Google. Note that the weights **wi** are arbitrary. Hence it is not disclosed to the merchants.

Conclusion

From our explanation so far, it is obvious that auction is still in vogue and not yet sidelined. Additionally, it takes some mastery and skills to be able to perform effectively at any auction. Data science is indeed an all-encompassing domain. The next chapter will examine limited dependent variables.

Chapter 11: Limited Dependent Variables

This chapter examines the different approaches to creating and working with dependent variables. The chapter covers the following outlines:

- Limited Dependent Variables
- Logit
- Prohit
- Slopes

Limited Dependent Variables

Dependent variables are limited when the variables are discrete, binomial, or multinomial. However, most times, we use the continuous variables for the dependent (y) variable to run a regression. A good example is when we run regressions on an income of education. Hence we will need a different approach to run regression on these types of variables.

A unique type of limited dependent variables is discrete dependent variables. Some of the examples of models that use this dependent variable are Logit and Prohit model. They are often referred to as qualitative response (QR) models.

A discrete dependent variable often occurs as binary by taking values of {0,1}. When we regress this, we get a probability model. However, if we just regress from the left-hand side of one and zero on a suite of right-hand side variables, this could be fit as linear regression. If we should take another observation with the right-hand side value, for instance, x = {x1,x2,..., xk}we could use the fitted coefficient to compute the value of the y variables. Except by unusual coincidence, the value would not be 0 or 1.

In a limited dependent variable, we would also explain the reason for the results in the allocation of categories. There is also a relationship between limited dependent variables and classifier models. This is because classifier models focus on allocating observation to categories, in the same vein some examples of limited dependent variables focus on explaining whether a firm is syndicated or not, whether a person is employed or not, and whether or not a firm is solvent and so on.

It is important to note that most times, these fitted values might not even be between 0 and 1 in linear regression. It means that we could choose a nonlinear regression to ensure that the fitted value of y is restricted to 0 and 1. After this, we could get a model and fit in a probability. To achieve this, we use any of the two models mentioned in our explanation, i.e., Logit or Prohit.

Logit

This is also known as logistic regression. This type of model takes the form highlighted below:

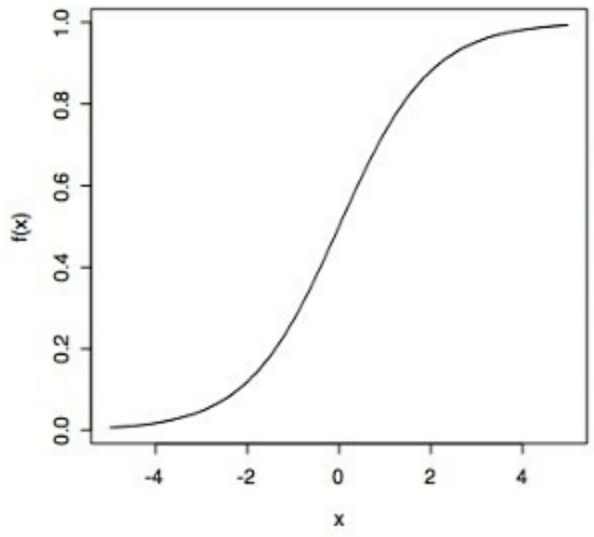
$$y = \frac{e^{f(x)}}{1 + e^{f(x)}}, \quad f(x) = \beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$$

Our focus here is to fit in the coefficient {β0, β1,..., βk}. Note that this would be done irrespective of coefficients (x) $\in$ (−∞,+∞), but y $\in$ (0,1). When f (x) → −∞, y → 0,and when f(x) → +∞, y → 1. This model can be rewritten as

$$y = \frac{e^{\beta' x}}{1 + e^{\beta' x}} \equiv \Lambda(\beta' x)$$

Here Λ (lambda) stands for logit.

The model generates an S-shaped curve which can be plotted in this form



The probability that y=1 is the fitted value of y.

Prohit

This is very similar to the Logit except that the normal distribution replaces the probability function. The equation for the nonlinear equation is as follows:

$$y = \Phi\,[f(x)],\ \ f(x) = \beta 0 + \beta 1x1 + \ldots \beta kx$$

The cumulative normal probably function is represented by the Φ sign. Like the Logit model, irrespective of the coefficients, $f(x) \in (-\infty, +\infty)$, but $y \in (0, 1)$. When $f(x) \rightarrow -\infty$, $y \rightarrow 0$, and when $f(x) \rightarrow +\infty$, $y \rightarrow 1$.

Analysis

The two models Prohit and Logit explain how we would set in our binary model, i.e.:

$$Pr[y = 1] = F(\beta^1 x)$$

Where x stands for the vector of explanatory variables, β is a vector of coefficient, and F is the Prohit/Logit function.

$$y\hat{} = F(\beta^1 x)$$

$y\hat{}$ stands for the fitted value of a given x and y. In any of these cases, the function of the Logit or Prohit remains, as we have stated above. Of course

$$Pr[y = 0] = 1-F(\beta x)$$

This model can be expressed in conditional expectation form as

$$E[y|x] = F(\beta^1 x)(y = 1)+[1-F(\beta^1 x)](y = 0) = F(\beta^1 x)$$

Slopes

It is very easy to observe how dependent variables change whenever there is a change in the right-hand side variables in linear regression. This is not so with nonlinear models. To get started with linear regression, let's recall that the value for **y** lies within the range of (0, 1). Our concern now is in how any change in the value of explanatory variables leads to a change in E(y|x). First, we would take the derivative as:

$$\frac{\partial E(y|x)}{\partial x} = F'(\beta' x)\beta \equiv f(\beta' x)\beta$$

Now we will compute this as the means of the regressor for each model. The following is the result of the Logit model:

```
(C1) F: exp(b*x)/(1+exp(b*x));
                                      b x
                                    %E
(D1)                              ---------
                                    b x
                                  %E     + 1
(C2) diff(F,x);
                          b x             2 b x
                       b %E            b %E
(D2)                   ------------ - ----------------
                         b x             b x       2
                       %E     + 1     (%E     + 1)
```

This can be written as:

$$\frac{\partial E(y|x)}{\partial x} = \beta\left(\frac{e^{\beta' x}}{1+e^{\beta' x}}\right)\left(1-\frac{e^{\beta' x}}{1+e^{\beta' x}}\right)$$

$$\frac{\partial E(y|x)}{\partial x} = \beta \cdot \Lambda(\beta' x) \cdot [1-\Lambda(\beta' x)]$$

This can be rewritten as:

$$\frac{\partial E(y|x)}{\partial x} = \phi(\beta' x)\beta$$

```
> h = glm(y~x, family=binomial(link="logit"))
> beta = h$coefficients
> beta
 (Intercept)          xPTS          xREB          xAST           xTO
-45.83315262  -0.06127422   0.49037435   0.16421685  -0.38404689
         xA.T          xSTL          xBLK           xPF          xFG
  1.56351478   0.78359670   0.07867125   0.02602243  46.21373793
          xFT          xX3P
 10.72992472   5.41984900
```

```
> dim(x)
[1] 64 11
> beta = as.matrix(beta)
> dim(beta)
[1] 12  1
> wuns = matrix(1,64,1)
> x = cbind(wuns,x)
> dim(x)
[1] 64 12
> xbar = as.matrix(colMeans(x))
> dim(xbar)
[1] 12  1
> xbar
            [,1]
     1.0000000
PTS 67.1015625
REB 34.4671875
AST 12.7484375
TO  13.9578125
A.T  0.9778125
STL  6.8234375
BLK  2.7500000
PF  18.6562500
FG   0.4232969
FT   0.6914687
X3P  0.3333750
> logitfunction = exp(t(beta) %*% xbar)/(1+exp(t(beta) %*% xbar))
> logitfunction
          [,1]
[1,] 0.5139925
> slopes = beta * logitfunction[1] * (1-logitfunction[1])
> slopes
                      [,1]
(Intercept) -11.449314459
xPTS         -0.015306558
xREB          0.122497576
xAST          0.041022062
xTO          -0.095936529
```

Now using the Prohit model, our result is :

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{j2\pi ft}\, df$$

Here the normal density function is φ(.)

xA.T	0.390572574
xSTL	0.195745753
xBLK	0.019652410
xPF	0.006500512
xFG	11.544386272
xFT	2.680380362
xX3P	1.353901094

Maximum-Likelihood Estimation (MLE)

For the above estimation, using the g l m function is done by R using MLE. To write out this formula, recall that we already agree that each LHS variable is y = {0,1}. The function is likely to take this shape:

$$L = \prod_{i=1}^{n} F(\beta' x)^{y_i}[1 - F(\beta' x)]^{1-y_i}$$

The log-likelihood would take this form:

$$\ln L = \sum_{i=1}^{n} [y_i \ln F(\beta' x) + (1 - y_i) \ln[1 - F(\beta' x)]]$$

While we take this derivative to maximize the log-likelihood

$$\frac{\partial \ln L}{\partial \beta} = \sum_{i=1}^{n} \left[y_i \frac{f(\beta' x)}{F(\beta' x)} - (1 - y_i) \frac{f(\beta' x)}{1 - F(\beta' x)} \right] \beta = 0$$

This gives us a system of the equation that can be used to solve β. Likelihood equation is a collective name for the system of first-order conditions. To get the t-stat for a coefficient, we simply divide its value by its standard deviation. The standard deviation is gotten from the answer to the question, how does the coefficient set β change when the log-likelihood changes? Our interest is in ∂β/∂lnL. The reciprocal of this has already been computed above. Next, we define:

g = ∂lnL /∂β

After this, we define the second derivative. This is also known as the Hessian matrix.

$$H = \frac{\partial^2 \ln L}{\partial \beta \partial \beta'}$$

Note that the following equation are valid:

E(g) = 0 (this is a vector)

Var(g) = E(gg)−E(g)2 = E(gg)

= −E(H) (this is a non-trivial proof)

Next, we call

I(β) = −E(H)

the information matrix. This is because heuristically, the variation in log-likelihood with changes in beta is given by Var(g) = −E(H) = I(β). The inverse of this gives the variance of β. Hence, we will have:

Var(β) → I(β)−1

To get the t-statistics, we divide the value of β by the square root of the diagonal of the matrix.

Multinomial Logit

To use this package, we would need the **nnet** package. This can take the following form:

$$\text{Prob}[y = j] = p_j = \frac{\exp(\beta_j' x)}{1 + \sum_{j=1}^{J} \exp(\beta_j' x)}$$

Then we set

$$\text{Prob}[y = 0] = p_0 = \frac{1}{1 + \sum_{j=1}^{J} \exp(\beta_j' x)}$$

Limited Dependent Variables in VC Syndication

It is indisputable that not all venture-backed firms would end up making a successful exit either through a buyout, an IPO, or through another exit route. Here we would be measuring the probability of a firm making a successful exit by examining a large sample of firms. Hence, a successful exit would be designated as S = 1 while an unsuccessful exit would be S = 0. We would fit a Prohit model to the data by using matrix X of explanatory variables. Next, we define S to be based on a latent threshold variable $S*$ such that:

$$S = \begin{cases} 1 & \text{if } S^* > 0 \\ 0 & \text{if } S^* \leq 0. \end{cases}$$

And the latent variable is modeled as

$$S^* = \gamma' X + u, \quad u \sim N(0, \sigma_u^2)$$

The probability of exit, which entails the E(S) for all financing rounds, is provided by the fitted model.

$$E(S) = E(S^* > 0) = E(u > -\gamma' X) = 1 - \Phi(-\gamma' X) = \Phi(\gamma' X),$$

Using the standard likelihood method, the vector of coefficient fitted in the Prohit model is γ. The cumulative normal distribution is represented by Φ(.).

Endogeneity

Assuming we want to look at the impact of syndication in a successful venture. Success in a syndicated venture is a product of two broad aspects of VC expertise. To start with, syndicate are very effective for picking good firm while VC is very effective for selecting a good project to invest in. VC is a selection hypothesis discovered by Lerner (1994). Since the process of syndicate entails the derivation of a second opinion by the VCs, this means that a syndicate provides evidence that the project is a very good one. Aside from this, the syndicate can be used to provide detailed monitoring as a result of its ability to bring a wide range of skills to the venture.

A dummy variable for syndication permits a firsthand estimation of whether or not syndication has any impact on performance, while a regression of variables allows for a return on different characteristics of the firm, although it can be said or assumed that syndicated firm is large of higher performance capacity, irrespective of whether they chose to syndicate or not. VC tends to prefer better firms that might likely syndicate. Not only can this VC identify these firms. In this kind of situation, the added value from syndicate is revealed by the coefficient of the dummy variables, especially when there is no value. As a result, we first correct specifications for endogeneity. Next, we check whether or not dummy variables are significant.

The correction specification that would be adopted for endogeneity is the one suggested by Greene (2011). The required model would be briefly summarized as follows. However, before then, here is the performance regression:

$$Y = \beta' X + \delta S + \epsilon, \quad \epsilon \sim N(0, \sigma_\epsilon^2)$$

For the above equation, Y stands for the performance variable while S is the dummy variable that takes the value of 1 if there is syndication, if there is no syndication, the value is zero. δ is the coefficient that reveals the differences in performance often caused by syndication. If there is no difference in performance, it means that there is no difference in performance across the two firms or that the X variables are enough to explain the differences in performance across the firms.

However, since this is the same variables that determine whether or not there is syndication, it means that we have an endogeneity issue. This would be resolved by adding some corrections to the above model. The sign € standing for the error term would be affected by our corrections. When the firm is syndicated, and our value for S is 1, then the adjustment in the € sign would be:

$$E(\epsilon|S=1) = E(\epsilon|S^* > 0) = E(\epsilon|u > -\gamma' X) = \rho\sigma_\epsilon \left[\frac{\phi(\gamma' X)}{\Phi(\gamma' X)}\right].$$

where ρ = Corr(,u), and the standard deviation of € is σ€. It means that

$$E(Y|S=1) = \beta' X + \delta + \rho\sigma_\epsilon \left[\frac{\phi(\gamma' X)}{\Phi(\gamma' X)}\right].$$

Note that $\phi(-\gamma' X) = \phi(\gamma' X)$, and $1 - \Phi(-\gamma' X) = \Phi(\gamma' X$

For firms that are not syndicated, our result would be:

$$E(Y|S=0) = \beta' X + \rho\sigma_\epsilon \left[\frac{-\phi(\gamma' X)}{1-\Phi(\gamma' X)}\right].$$

This can be estimated by linear cross-sectional regression as:

$$Y = \beta' X + \beta_m m'(\gamma' X)$$

where $m' = \frac{-\phi(\gamma' X)}{1-\Phi(\gamma' X)}$ and $\beta_m = \rho\sigma_\epsilon$.

The estimation model would take the form of both the none syndicated equation and the cross-regression model. If this is the case, β would be forced to remain constant in all the firms without initiating any additional constraint. Hence, the specification would maintain the same OLS form. However, if after the endogeneity correction, δ remains constant, this supports the hypothesis that syndication is an initiator of differences in performance. If the coefficients {δ, βm} remain significant, then for each syndicate performance round, the expected differences would be:

$$\delta + \beta_m \left[m(\gamma'_{ij} X_{ij}) - m'(\gamma'_{ij} X_{ij})\right], \quad \forall i, j.$$

The method explained above is one of the best ways to address the treatment effect. Another effective way to do this is first to use a Prohit model and then set m(γX) = Φ(γX). This approach is what is referred to as an instrumental variables approach.

Endogeneity - Some Theories to Wrap Up

This is a situation that arises as a result of the correlation of error in terms of regression and independent variables. Endogeneity can be highlighted as:

$$Y = \beta' X + u, \quad E(X \cdot u) \neq 0$$

This can happen in the following listed ways:

Measurement error: this occurs when X is measured for error. In such a situation, we have X = X+e. The regression formula is, therefore:

$$Y = \beta_0 + \beta_1(\bar{X} - e) + u = \beta_0 + \beta_1 \bar{X} + (u - \beta_1 e) = \beta_0 + \beta_1 \bar{X} + v$$

Hence we have

$$E(\bar{X} \cdot v) = E[(X + e)(u - \beta_1 e)] = -\beta_1 E(e^2) = -\beta_1 Var(e) \neq 0$$

Omitted variable: assuming the equation for the true model is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + u$$

But the problem now is that we don't X2, which is a correlate of X1. This implies that in the error term, we will no longer have: E(Xi · u) = 0, ∀ i.

Simultaneity: This is a situation where both Y and X are determined jointly. A good example is the joint use of high way and high school because they go together. The structural setting of this kind of situation can be highlighted as:

$$Y = \beta_0 + \beta_1 X + u, \qquad X = \alpha_0 + \alpha_1 Y + v$$

When we try to solve this equation, what we get is a reduced-form version of the model:

$$Cov(X, u) = Cov\left(\frac{v + \alpha_1 u}{1 - \alpha_1 \beta_1}, u\right) = \frac{\alpha_1}{1 - \alpha_1 \beta_1} \cdot Var(u)$$

Next, we compute the above and get:

$$Y = \frac{\beta_0 + \beta_1 \alpha_0}{1 - \alpha_1 \beta_1} + \frac{\beta v + u}{1 - \alpha_1 \beta_1}, \qquad X = \frac{\alpha_0 + \alpha_1 \beta_0}{1 - \alpha_1 \beta_1} + \frac{v + \alpha_1 u}{1 - \alpha_1 \beta_1}$$

Conclusion

- This chapter has extensively covered all there is to know about limited Dependent variables. In the next chapter we'll cover Fourier Analysis and Network Theory.

Chapter Twelve: Fourier Analysis And Network Theory

This chapter would cover the following outlines:

- Fourier Analysis
- Fourier series
- Solving the coefficients
- Complex Algebra
- Courier Transform

Fourier Analysis

This analysis entails numerous different connections between infinite series, vector theory, complex number, and geometry. Different applications such as fitting economic time series, wavelets, pricing time series, and generating risk-neutral pricing applications can be carried out using Fourier analysis.

Fourier Series

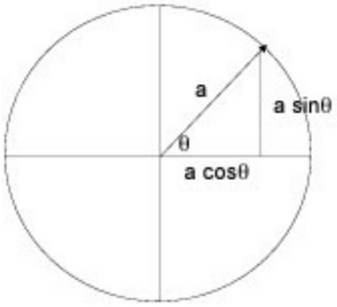
Introduction:

These are series used to determine periodical time series by combining sine and cosine waves. The time it takes one cycle of the wave is called "period" T, while the number of cycles per second is the "frequency of the wave" **f.** The formula for this is:

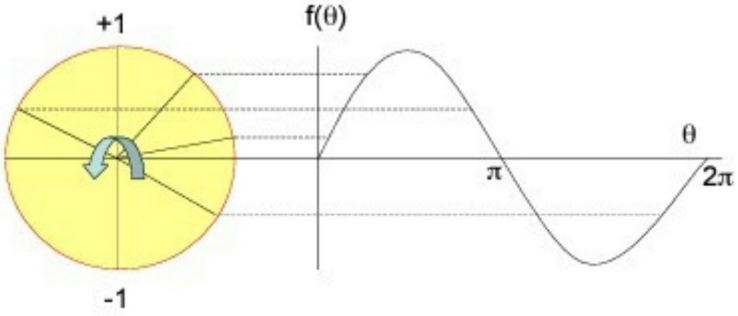
$$f = 1/T$$

Unit circle

We would be using some basic geometry to explain this



In our circle above, if a=1, the circle is the unit circle. There is a relationship or link between the unit circle and the sine wave. To understand this, we would plot another diagram:



In the second circle, the height of the unit vector on the circle traces out the circle as we rotate through the angles. For radius a, we would arrive at **a** sine wave with aptitude **a.** This can be written as:

$$f(\theta) = a\sin(\theta)$$

Angular Velocity

Velocity is simply the distance per time in a given direction. In angular velocity, distance is measured in degree, that is, the degree per unit of time. Usually, angular velocity is represented by the symbol **w.** The formula can be written as :

$$\begin{aligned} f(t) &= a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \\ &= a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n}{T} t + b_n \sin \frac{2\pi n}{T} t \right) \end{aligned}$$

$$\omega = \frac{\theta}{T}, \quad \theta = \omega T$$

The function in the first equation can be written in time as :

f (t) = asinωt

Fourier Series

As already explained in the short introduction, a Fourier series is simply the collection of both sine and cosine waves. When these two are summed up, it would closely approximate any given waveform. Fourier series can be expressed in terms of both sine and cosine wave as :

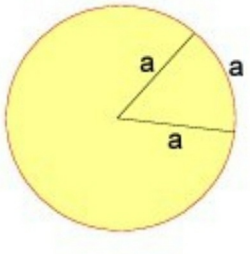
$$f(\theta) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\theta + b_n \sin n\theta)$$

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)$$

We would need the a0 because the waves may not be asymmetric around the x-axis.

Radian

The figure below defines the angle of a radian. Degrees are presented in units of radians.



The angle in the above diagram is a radian. This is approximately 57.2958 degrees. This is a bit lower than the 60 degrees expected of an equilateral triangle. Note that since the circumference is 2πa, 57.2958π = 57.2958×3.142 = 180 degrees. Hence for the unit circle, we would have:

$$\begin{aligned} 2\pi &= 360 \text{ (degrees)} \\ \omega &= \frac{360}{T} \\ \omega &= \frac{2\pi}{T} \end{aligned}$$

The Fourier series can be rewritten as:

Our next action is how to solve the coefficient.

How to solve the coefficient

The first thing to note is that sine and cosine are orthogonal. Hence, their inner product is zero. Therefore we will have:

$$\begin{aligned} \int_0^T \sin(nt).\cos(mt)\,dt &= 0, \forall n, m \\ \int_0^T \sin(nt).\sin(mt)\,dt &= 0, \forall n \neq m \\ \int_0^T \cos(nt).\cos(mt)\,dt &= 0, \forall n \neq m \end{aligned}$$

This implies that when we multiply two waves and then integrate the resultant wave from 0 to T

unless the two waves have the same frequency, our result would be zero (0). With this, the way we solve the coefficient of the Fourier series is highlighted below. We integrate both side of the equation above from 0 to T to arrive at this:

$$\int_0^T f(t) = \int_0^T a_0 \, dt + \int_0^T \left[\sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \, dt \right]$$

Except for the first term, all other terms are zero. This implies that for the terms we arrive at zero, the sine and cosine are integrated above the cycle. Hence we arrive at :

$$\int_0^T f(t) \, dt = a_0 T$$

or

$$a_0 = \frac{1}{T} \int_0^T f(t) \, dt$$

Let's try another integral:

$$\begin{aligned} \int_0^T f(t) \cos(\omega t) &= \int_0^T a_0 \cos(\omega t) \, dt \\ &\quad + \int_0^T \left[\sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \cos(\omega t) \, dt \right] \end{aligned}$$

Except for the terms in a1 cos(ωt) cos(ωt), all other terms are zero. This is because we are multiplying two waves with the same frequency. Therefore we would get:

$$\begin{aligned} \int_0^T f(t) \cos(\omega t) &= \int_0^T a_1 \cos(\omega t) \cos(\omega t) \, dt \\ &= a_1 \frac{T}{2} \end{aligned}$$

How do we arrive at the above?

Note that integrating cos(ωt) over one circle would be equals to zero for unit amplitude. When we multiply cos (ωt) by itself, we flip all the wave segments from below the zero lines to above the zero lines. With this, half the area from 0 to T is filled by the product wave. Hence, we get T/2. Therefore:

$$a_1 = \frac{2}{T} \int_0^T f(t) \cos(\omega t)$$

We can use this method to solve all **an** all we do is multiply by cos (nωt) and then integrate. We can also use this to solve **bn.** We simply multiply by a sine (nωt) and then integrate. This forms the basis of the result of Fourier series coefficients highlighted below:

$$
\begin{aligned}
a_0 &= \frac{1}{T}\int_{-T/2}^{T/2} f(t)\,dt = \frac{1}{T}\int_0^T f(t)\,dt \\
a_n &= \frac{1}{T/2}\int_{-T/2}^{T/2} f(t)\cos(n\omega t)\,dt = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt \\
b_n &= \frac{1}{T/2}\int_{-T/2}^{T/2} f(t)\sin(n\omega t)\,dt = \frac{2}{T}\int_0^T f(t)\sin(n\omega t)\,dt
\end{aligned}
$$

Complex Algebra

Recall that :

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}.$$

and

$$e^{i\theta} = \sum_{n=0}^{\infty} \frac{1}{n!}(i\theta)^n$$

$$\begin{aligned} \cos(\theta) &= 1 + 0.\theta - \frac{1}{2!}\theta^2 + 0.\theta^3 + \frac{1}{4!}\theta^2 + \dots \\ i\sin(\theta) &= 0 + i\theta + 0.\theta^2 - \frac{1}{3!}i\theta^3 + 0.\theta^4 + \dots \end{aligned}$$

This generated the popular Euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

and the corresponding

$$e^{-i\theta} = \cos\theta - i\sin\theta$$

Also recall that, cos(−θ) = cos(θ), and sin(−θ) = −sin(θ). Also note that if θ = π, then

$$e^{-i\pi} + 1 = 0$$

This can be written as

$$e^{-i\pi} = \cos(\pi) - i\sin(\pi) = -1 + 0$$

And as an equation that entails five major mathematical constants and three operators. These are i,π,e,0,1}, and {+,−,=}.

From Trigs to Complex

Using the last two equations above, we would arrive at this:

$$\begin{aligned}\cos\theta &= \frac{1}{2}(e^{i\theta}+e^{-i\theta})\\ \sin\theta &= \frac{1}{2}i(e^{i\theta}-e^{-i\theta})\end{aligned}$$

Going back to the Fourier series:

$$\begin{aligned}f(t) &= a_0+\sum_{n=1}^{\infty}(a_n\cos n\omega t+b_n\sin n\omega t)\\ &= a_0+\sum_{n=1}^{\infty}\left(a_n\frac{1}{2}(e^{in\omega t}+e^{-in\omega t})+b_n\frac{1}{2i}(e^{in\omega t}-e^{-in\omega t})\right)\\ &= a_0+\sum_{n=1}^{\infty}\left(A_ne^{in\omega t}+B_ne^{-in\omega t}\right)\end{aligned}$$

where

$$A_n=\frac{1}{T}\int_0^T f(t)e^{-in\omega t}\,dt$$

$$B_n=\frac{1}{T}\int_0^T f(t)e^{in\omega t}\,dt$$

How then do we solve this? Start with:

Then

$$f(t)=a_0+\sum_{n=1}^{\infty}\left(a_n\frac{1}{2}(e^{in\omega t}+e^{-in\omega t})+b_n\frac{i}{2i^2}(e^{in\omega t}-e^{-in\omega t})\right)$$

$$=a_0+\sum_{n=1}^{\infty}\left(a_n\frac{1}{2}(e^{in\omega t}+e^{-in\omega t})+b_n\frac{i}{-2}(e^{in\omega t}-e^{-in\omega t})\right)$$

$$f(t)=a_0+\sum_{n=1}^{\infty}\left(\frac{1}{2}(a_n-ib_n)e^{in\omega t}+\frac{1}{2}(a_n+ib_n)e^{-in\omega t}\right)$$

How you get rid of a0

The first thing we would do here is to expand the first summation n =0; we would arrive at :

$$A_0e^{i0\omega t}=A_0\equiv a_0$$

The expression can be written as:

$$f(t)=\sum_{n=0}^{\infty}A_ne^{in\omega t}+\sum_{n=1}^{\infty}B_ne^{-in\omega t}$$

Collapsing and Simplifying

Our focus here is to collapse the two terms. First, let's take note of the following:

$$\sum_{n=1}^{2} x^n = x^1 + x^2 = \sum_{n=-2}^{-1} x^{-n} = x^2 + x^1$$

When we apply this idea, we will get:

$$\begin{aligned} f(t) &= \sum_{n=0}^{\infty} A_n e^{in\omega t} + \sum_{n=1}^{\infty} B_n e^{-in\omega t} \\ &= \sum_{n=0}^{\infty} A_n e^{in\omega t} + \sum_{n=-\infty}^{-1} B_{(-n)} e^{in\omega t} \\ &\quad \text{where} \\ &\quad B_{(-n)} = \frac{1}{T}\int_0^T f(t) e^{-in\omega t}\, dt = A_n \\ &= \sum_{n=-\infty}^{\infty} C_n e^{in\omega t} \\ &\quad \text{where} \\ &\quad C_n = \frac{1}{T}\int_0^T f(t) e^{-in\omega t}\, dt \end{aligned}$$

All we have to do is rename An to Cn for clarity. The big turnout of this process is that we have been able to contain {a0, a bn} into one coefficient set Cn. We will write the following for completeness sake:

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) = \sum_{n=-\infty}^{\infty} C_n e^{in\omega t}$$

Fourier Transform

With this technique, we can go from the Fourier series, which uses a period T to aperiodic waves. This is simply to let the period go to infinity. This implies that the frequency gets very small. To do our analysis, we will substitute f(t) with g(t). This is because we now need to use f or Δf to denote frequency. Recall that:

$$\omega = \frac{2\pi}{T} = 2\pi f, \quad n\omega = 2\pi f_n$$

To recap

$$\begin{aligned} g(t) &= \sum_{n=-\infty}^{\infty} C_n e^{in\omega t} = \sum_{n=-\infty}^{\infty} C_n e^{i2\pi ft} \\ C_n &= \frac{1}{T}\int_0^T g(t)e^{-in\omega t}\,dt \end{aligned}$$

This can be alternatively written in frequency terms as:

$$C_n = \Delta f \int_{-T/2}^{T/2} g(t)e^{-i2\pi f_n t}\,dt$$

Next, we substitute this into the formula of g(t) and get:

$$g(t) = \sum_{n=-\infty}^{\infty} \left[\Delta f \int_{-T/2}^{T/2} g(t)e^{-i2\pi f_n t}\,dt\right] e^{in\omega t}$$

Taking the formula of limits as:

$$g(t) = \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} \left[\int_{-T/2}^{T/2} g(t)e^{-i2\pi f_n t}\,dt\right] e^{i2\pi f_n t}\Delta f$$

This gives a double integral

$$g(t) = \int_{-\infty}^{\infty} \underbrace{\left[\int_{-\infty}^{\infty} g(t)e^{-i2\pi ft}\,dt\right]}_{G(f)} e^{i2\pi ft}\,df$$

df stands for frequency domain while dt stands for the time domain. Thus the **Fourier transform** moves from the time domain to a frequency domain.

$$G(f) = \int_{-\infty}^{\infty} g(t)e^{-i2\pi ft}\,dt$$

Inverse Fourier Transform moves from the frequency domain to the time domain.

$$C_n = \frac{1}{T}\int_0^T g(t)e^{-i2\pi f_n t}\,dt = \frac{1}{T}\int_0^T g(t)e^{-in\omega t}\,dt$$

Fourier coefficient is as follows:

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{j2\pi ft}\, df$$

We would note that there is an incredible similarity between transform and coefficient. Take note of the following:

The coefficients give the amplitude of each component wave

The transform gives the area of a component wave of frequency f. This can be seen because the transform does not have the divide by T in it.

For every frequency F, the transform gives the rate of occurrence with that frequency, relative to other waves.

The Fourier breaks a complex aperiodic wave into simple periodic waves.

Application To Probability Functions

Characteristic functions

The expectation of the following function of F gives the characteristics function of x

$$\phi(s) = E[e^{isx}] = \int_{-\infty}^{\infty} e^{isx} f(x)\, dx$$

where f(x) is the probability density of x. By Taylor series for eisx we have:

$$\begin{aligned}\int_{-\infty}^{\infty} e^{isx} f(x)\, dx &= \int_{-\infty}^{\infty} [1 + isx + \frac{1}{2}(isx)^2 + \ldots] f(x) dx \\ &= \sum_{j=0}^{\infty} \frac{(is)^j}{j!} m_j \\ &= 1 + (is)m_1 + \frac{1}{2}(is)^2 m_2 + \frac{1}{6}(is)^3 m_3 + \ldots\end{aligned}$$

Where mj is the jth moment. It is therefore easy to see that:

time domain

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{i2\pi ft}\, df$$

$$m_j = \frac{1}{i^j} \left[\frac{d\phi(s)}{ds} \right]_{s=0}$$

where $i = \sqrt{-1}$.

Network Theory

Network science is gaining a group in the business world today. The term "network effect" is widely used in conceptual terms to describe the gain from piggybacking on connections in the business world. The steady rise in the effect of the network on business today is only the tip of the iceberg. As the cost of the network and its analysis drops rapidly, business organizations would make use of them more and more.

Network theories are also used by firms to find communities of consumers to partition and drive traffic in their marketing efforts. Networks are also used to understand how information flows in Network.

Graph Theory

This is the first stage in understanding how network theory works. This is why a business student is usually taught a digression in graph theory in other to understand how a network works.

The graph is simply a picture of a network. The picture consists of the relationship between entities. The entities are called nodes or vertices (set V), while the relationship is called the edges of a graph (set E). Therefore a graph G is described as :

$$G = (V,E)$$

There are two basic types of graphs. When the edges $e \in E$ of a graph are not tipped with arrows signifying some causality or direction, this type of graph is called an “undirected” graph. However, if the graph is tipped with direction, it is called a “directed” graph. When the connections (edges) between vertices $v \in V$ have weights on them, the graph is called a “weighted graph.” However, when the graph has no weight on them, it's “unweighted.” For any pair of vertices (u,v) in an unweighted graph, we have:

$$w(u,v) = \begin{cases} w(u,v) = 1, & \text{if } (u,v) \in E \\ w(u,v) = 0, & \text{if } (u,v) \ni E \end{cases}$$

However, the value of w(u,v) is unrestricted in a weighted graph. This can also be negative.

A directed graph can either be cyclic or acyclic. For cyclic graphs, there is a path from a source node leading back to the node itself. This is not the case with acyclic graphs. Direct acyclic graphs are represented with the term "dag."

Moreover, a graph can be represented by its adjacent matrix. This is simply the matrix $A = \{w(u,v)\}, \forall$ u, v. We can also take the transpose of the matrix. In the case of a directed graph, this would simply reverse the direction of all the edges.

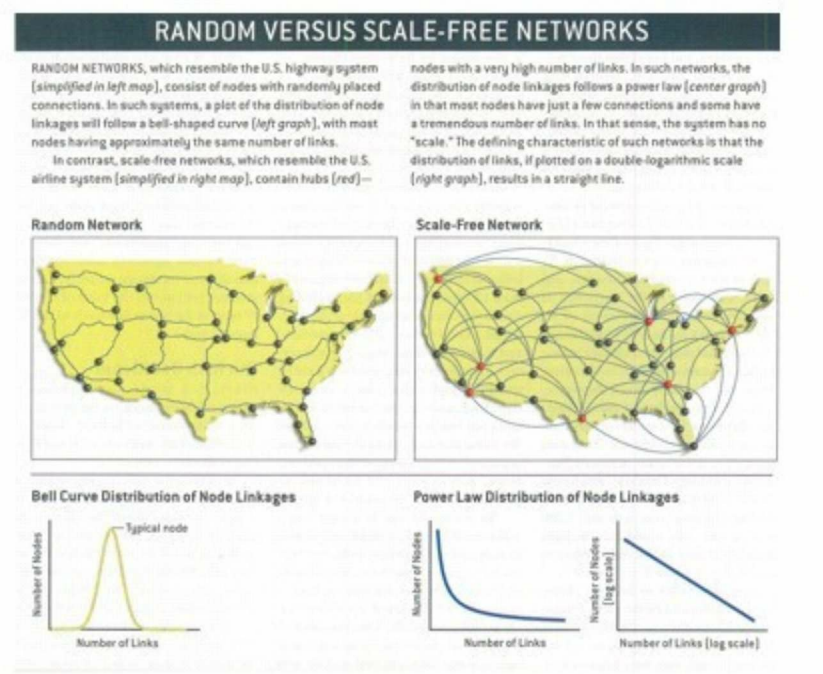
Features of Graphs

There are various features of a graph. These include the number of graph nodes and the distribution of links across nodes. The edges, which are the links and the structure of the nodes, determine the extent the nodes are connected, and the importance of individual nodes. This also determines the flow of networks.

A simple bifurcation of graphs suggest two types:

- Random graph
- Scale-free graph

These two graphs are portrayed in an article in the Scientific American written by Barabasi and Bonabeau (2003).



The random graph can be plotted by putting in place some sets of n nodes and then connecting pairs of nodes randomly with some probability p. The higher the connected probabilities, the more the edges contained in the graph. The distribution of the number of edges in each graph will either be more or less Gaussian because there is a mean number of edges (n.p) with some range around the mean. The left graph in the above figure is an example of this. In the graph, the distribution of links is shown in bell-shapes. When a number d gives the number of links of a node, the distribution of nodes in a random graph would be $f(d) \sim N(\mu,\sigma 2)$, where μ is the mean number of nodes with variance $\sigma 2$.

The structure of a scale-free graph is a hub and spoke. Most nodes in the graph have very few links. However, there are some nodes with a large number of links. In our graph, the distribution of links is shown on the right side. This is not bell-shaped at all, rather it is more exponential. Although there is a mean for this distribution, this is not representative of either the hub-nodes or

non-hub nodes. Since the mean is not representative of the population, the distribution of links in this type of graph is scale-free. The network of this type of graph is also known as a scale-free network.

The distribution of nodes in a scale-free graph is often approximated by a power-law distribution, i.e., $f(d) \sim d^{-\alpha}$, where usually, nature seems to have stipulated that $2 \leq \alpha \leq 3$, by some curious twist of fate. The log-log plot of this distribution is linear.

The majority of networks in the world today aim to be scale-free. The reason for this is explained in the article by Barabasi and Albert (1999). To explain this, they developed the Theory of Preferential Attachment, which stated that as network progress and new nodes are added to it, the new nodes usually attach itself to existing nodes that have most of the links. As a result, influential nodes gain more connection and evolves into a hub and spoke structure.

The structure of the graph also determines some of the properties of the graph. For instance, a scale-free graph performs excellently in the transmission of information, and in moving air traffic passengers. This is why airports are arranged in this format. A scale-free network is also very good for random attacks. If, for example, a terrorist attacks an airport, unless it hits a hub, the damage is usually minimal.

In the rest of this chapter, we would examine financial network risk and many more interesting graphs.

Chapter 13: Searching Graph

In the previous chapter, we provided some introduction to Graph Theory in data science; in this chapter, we would be exploring the two broad types of searching graphs. These include depth-first-search (DFS) and breadth-first search (BFS). The reason we are concerned with this is that DFS, i.e., depth-first-theory is very good at searching communities in social networks while BFS works well with finding the shortest connections in networks. This chapter would cover the following outlines:

- Depth-First-Search
- Breadth-First-Search
- Strongly Connected Components
- Dijkstra's Shortest Path Algorithm
- Degrees Distribution
- Diameter
- Fragility
- Centrality
- Communities
- Modularity

Depth-First-Search

This begins by taking a vertex and use this to produce a tree of connected vertices recurring downward until there is no way to do this again. Here are the algorithms for DFS:

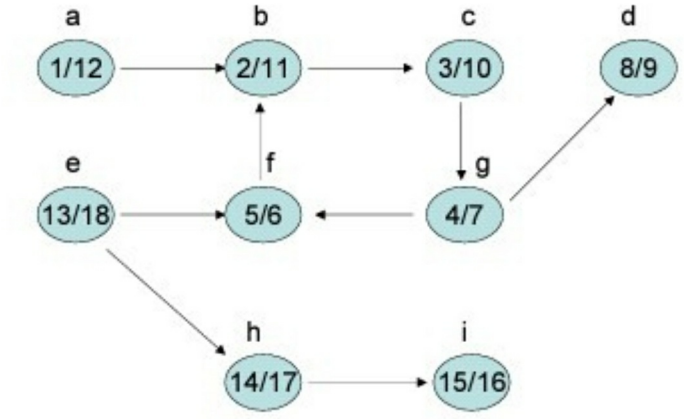
```
function DFS(u):
    for all v in SUCC(u):
        if notvisited(v):
            DFS(v)
    MARK(u)
```

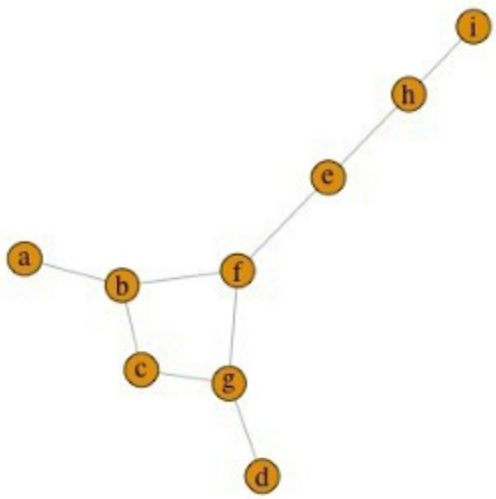
From this algorithm, we can generate two subtrees as follows:

$$a \to b \to c \to g \begin{cases} \nearrow f \\ \searrow d \end{cases}$$

$$e \to h \to i$$

The numbers of nodes show the sequence in which the nodes are accessed by the program. Usually, the output of a DFS is less detailed and presented in a very simple sequence in which the nodes are first visited. A good example of a DFS is the graph below:






```
> RNGkind("Mersenne-Twister")
> set.seed(123)
> g1 <- randomGraph(letters[1:10], 1:4, p=.3)
> g1
A graphNEL graph with undirected edges
Number of Nodes = 10
Number of Edges = 21
> edgeNames(g1)
 [1] "a~g" "a~i" "a~b" "a~d" "a~e" "a~f" "a~h" "b~f" "b~j"
[10] "b~d" "b~e" "b~h" "c~h" "d~e" "d~f" "d~h" "e~f" "e~h"
[19] "f~j" "f~h" "g~i"
> RNGkind()
[1] "Mersenne-Twister" "Inversion"
> DFS(g1,"a")
a b c d e f g h i j
0 1 6 2 3 4 8 5 9 7
```

We would notice that the DFS graph is a set of trees. The tree itself is a special kind of graph. It is inherently acyclic when the graph is acyclic. This implies that a cyclic graph would have the DFS trees at the back edges. This process can be interpreted as the partitioning of vertices into a subset of connected groups.

In applying this to business, it is necessary first to understand why they are different. Secondly, the ability to target these separate groups to different business questions and responses. Firms and business organizations that make use of these types of data use algorithms to find out "communities." Within communities, BFS is then applied to determine the connection of networks and the nearness of these connections.

Also, DFS can be used to find out the connectedness of the networks. With the use of DFS, we

can determine how close entities are to each other in a network. These analyses often suggest a "small world's phenomenon or what is colloquially referred to as " six degrees of separation."

Our next focus is to examine how DFS is implemented in the igraph package. We would be making use of this process all through this chapter. Our example below shows how a paired vertex list is used to create a graph.

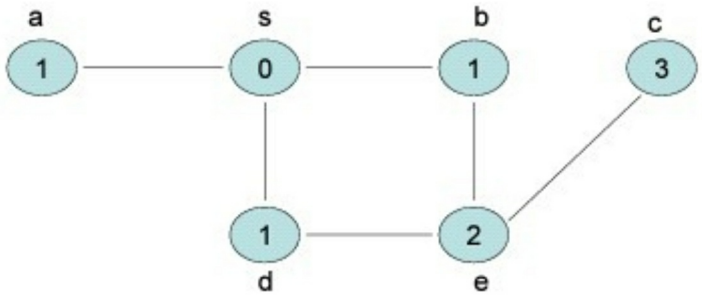
```
#CREATE A SIMPLE GRAPH
df = matrix(c("a","b","b","c","c","g",
              "f","b","g","d","g","f",
              "f","e","e","h","h","i"),ncol=2,byrow=TRUE)

g = graph.data.frame(df,directed=FALSE)
plot(g)

#DO DEPTH-FIRST SEARCH
dfs(g,"a")

$root
[1] 0

$neimode
[1] "out"

$order
+ 9/9 vertices, named:
[1] a b c g f e h i d

$order.out
NULL

$father
NULL
```

To verify our result, we would be plotting a graph

Breadth-First-Search

From a source vertex **s**, BFS explores the edges **E** to find out all the reachable vertices on the graph. This is carried out in a manner that proceeds to find a frontier of vertices **k** distant from **s**. The search moves on to locating vertices K + 1 away from the source after it has located all such vertices. The basic difference between DFS and BFS is that while BFS covers all the vertices while DFS goes all the way for without covering all the vertices at a single search.

To implement BFS, we first label each with its distance from the source vertex. An example of BFS is the graph below.

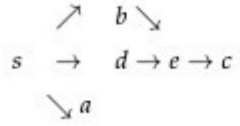


.

In the above graph, it is easy to determine the nearest graph. Now, when a positive reaction is gotten from someone in the population, this would help target the nearest neighbor cost-effectively. This is simply done by defining the edges of connections. The algorithms for BFS is as follows:

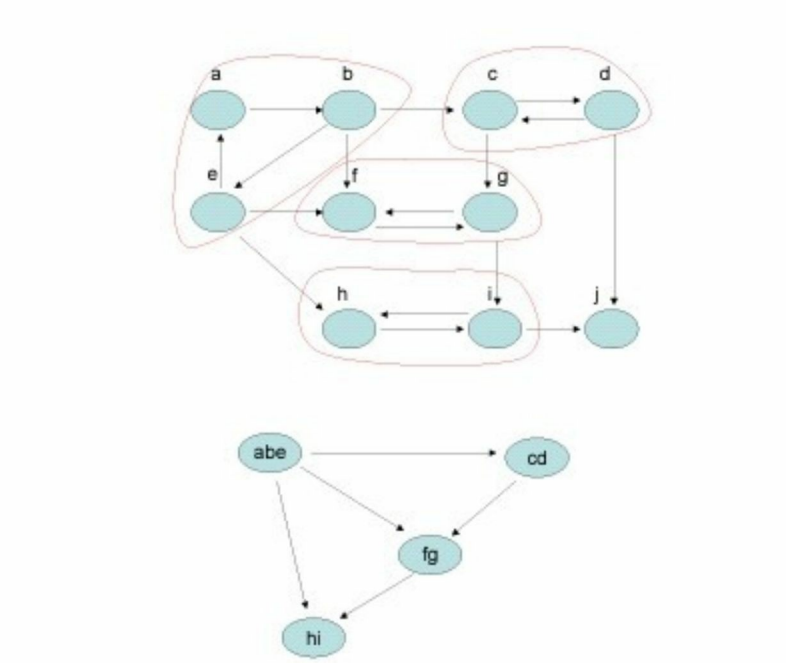
```
function BFS(s)
    MARK(s)
    Q = {s}
    T = {s}
    While Q ne { }:
        Choose u in Q
        Visit u
        for each v=SUCC(u):
            MARK(v)
            Q = Q + v
            T = T + (u,v)
```

BFS can also produce a tree, the level of the tree is determined by how close or far it is from the source vertex.



Strongly Connected Components

The best place to cluster the members of a network is in a directed graph. This is done by finding the strongly connected components on the graph. Hence, an SCC is a subgroup of the vertices $U \subset V$ in a graph with the properties for all pairs of its vertices $(u,v) \in U$, both vertices are reachable from each other. Below is an example of a graph broken down into its strongly connected components:



SCC is very useful for partitioning a graph into tight units. Not only this, but it also generates local feedbacks. This implies that when a member of SCC is targeted, all the members of the SCC components are targeted, and the stimulus is moved across the SCC.

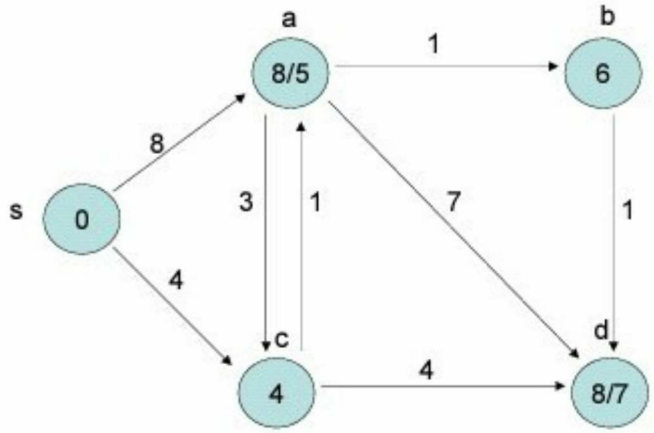
igraph has emerged as the most popular packages for analysis graphs. It has versions in Python, C, and R. This package can also be used to plot and generate the random graph in R.

Dijkstra's Shortest Path Algorithm

This is one of the most widely used algorithms in theoretical computer science. When a source vertex is given on a weighted, directed graph, the algorithms find the shortest path to all other nodes from source **s. w(u, v)** denotes the weight between two vertices. Dijkstra's algorithm works for graphs where w(u,v)≥0. For negative weights, the Bellman-Ford algorithm is used. Below is the algorithms:

```
function DIJKSTRA(G,w,s)
S = { }
%S = Set of vertices whose shortest paths from
%source s have been found
Q = V(G)
while Q notequal { } :
    u = getMin(Q)
    S = S + u
```

Below is an example of a graph that Dijkstra algorithms have been applied to.




```
[8,]    2    4    1
> g = add.edges(graph.empty(5), t(el[,1:2]), weight=el[,3])
> shortest.paths(g)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    5    6    4    7
[2,]    5    0    1    1    2
[3,]    6    1    0    2    1
[4,]    4    1    2    0    3
[5,]    7    2    1    3    0
> get.shortest.paths(g,0)
[[1]]
[1] 0

[[2]]
[1] 0 3 1

[[3]]
[1] 0 3 1 2

[[4]]
[1] 0 3

[[5]]
[1] 0 3 1 2 4
```

Degree Distributions

The number of links a node has to other nodes in a network is the degree of a node. The degree distribution is the probability of distribution of the nodes. In a directed network, there are two types of degrees. One is for in-degree, and the other is for out-degree. However, in an undirected network, the degree distribution is simply the number of edges contained in a node. It is important to note that the weight of the edges does not play any role in computing the degree distribution of the nodes. Although there are times when there would be a need to avail of this information.

Diameter

This is the longest shortest distance between any two nodes across all the nodes. It can be computed as follows:

```
> print(diameter(g))
[1] 7
```

We can cross-check this using the following command:

```
> res = shortest.paths(g)
> res[which(res==Inf)]=-99
> max(res)
[1] 7
> length(which(res==7))
[1] 18
```

Note that the paths that are of length 7 are 18 in number. However, this is a duplicate. Thus we run these paths in the two directions. This would give us 9 pairs of nodes that have the longest shortest distance between them.

Fragility

This is simply a quality of a network based on its degree distribution. The question that arises from this is, in comparing two networks of the same degree, how do we assess on which network contagion is more likely? This can be the first finding if the network is a scale-free network. This is because a scale-free network tends to spread the variable of interest, irrespective of whether it is a flu, information, or financial malaise. Also, in a scale-free network, the greater the preponderance of central hubs, the greater the probability of networks. This is because a few nodes already have a concentration of the degree. Hence, the higher the concentration, the higher the scale-free, and the higher the fragility.

To measure concentration, the economist has been using a unique package for a long time. This package is the Herndahl-Hirschman index. The index is quite technical to compute because it is the average degree squared for n nodes, i.e.,

$$H = E(d^2) = \frac{1}{n}\sum_{j=1}^{n} d_j^2$$

The more degrees get concentrated on a few nodes, the more the metric **H** increases, keeping the total degrees of the network constant. For instance, let's assume we have a graph of three nodes each with a degree {1, 1, 4} versus another graph of three nodes {2,2,2}. The value for metric **H** would increase in the former than the latter. The former would have **H = 18,** while the latter would have **H = 12.** To calculate the fragility, we simply normalize H by the average degree. Here is the formula for this:

$$\text{Fragility} = \frac{E(d^2)}{E(d)}$$

In the three nodes example we are using, the fragility is simply 3 and 2 respectively. Other normalization can be chosen too, for instance, the denominator E $(d)^2$. To compute this is not simply, as it requires a single line of code.

Centrality

This is the property of vertices in a network. Taking the adjacent matrix A = {w(u, v)} we went ahead and generated a measure of the "influence" of all the vertices in a network. Taking the influence of the vertex **i** as **xi.** The influence of each vertex is contained in column **x,** what then is the influence? To answer this question, let's take some moment to observe the web page. The more the links on the web page, the more the influence from its main page to other pages. This shows that influence is interdependent.

$$x = A\,x$$

We can simply add a scalar to this to get:

$$\xi\, x = Ax$$

When we add a scalar, we get an eigensystem. When we decompose this, we get the principal eigenvector. The value of this gives us the influence of each member. With this method, we can find the most influential network. There are numerous applications to this data to real data. This eigenvector centrality is exactly what Google trademarked as PageRank, even though they did not invent eigenvector centrality.

Other concepts of centrality are "betweenness." This is simply the proportion of the shortest path that goes through a node in relation to the other paths that go through the node. The formula for this is :

$$B(v) = \sum_{a \neq v \neq b} \frac{n_{a,b}(v)}{n_{a,b}}$$

Here, **na,b** is the number of shortest paths from node a to node b, and **na,b(v)** are the number of those paths that traverse through vertex **v**. Below is an example:

```
> el <- matrix(nc=3, byrow=TRUE,
+              c(0,1,0, 0,2,2, 0,3,1, 1,2,0, 1,4,5, 1,5,2, 2,1,1, 2,3,1,
+                2,6,1, 3,2,0, 3,6,2, 4,5,2, 4,7,8, 5,2,2, 5,6,1, 5,8,1,
+                5,9,3, 7,5,1, 7,8,1, 8,9,4) )
> g = add.edges(graph.empty(10), t(el[,1:2]), weight=el[,3])
> res = betweenness(g)
> res
 [1]  0.0 18.0 17.0  0.5  5.0 19.5  0.0  0.5  0.5  0.0
```

Communities

These are simply the spatial agglomeration of the vertex that tends to connect with each other than with others. To identify these agglomerations is a cluster detection problem, a computationally difficult (NP-hard) one. This is so because we allow each cluster to have different sizes. This, in turn, permits porous boundaries such that members both within and outside their preferred clusters. The solution to this is where communities come in.

This is simply constructed by optimizing **modularity.** Modularity is a metric of the differences between the number of within-community construction and the expected number of constructions. Because of the large computational complexity involved in sifting through all possible partitions, identifying communities is not an easy feat.

The whole idea of community formation started with Simon (1962). In his view, he explained that complex systems with several entities usually have coherent subsystems, or communities, that serve specific functional purposes. To understand the functional forces underlying these entities, it is important to identify communities embedded in larger entities. To understand this, we would be examining more definitions of the community detection method.

The community detection method is the method in which nodes are partitioned into clusters with the tendency to interact with each other. Hence, all nodes cannot belong to the same community; neither can we fix the number of the community at a time. Also, we would allow each community to have different sizes. Having beaten our partitioning into a more flexible task, our challenge now is to find the best partition because the number of possible partitions is very large.

However, since the community detection method aims at identifying clusters that are internally tightknit. This is the same as finding a partition of clusters to maximize the observed number of connections between cluster members minus what is expected conditional on the connections within the cluster, aggregated across all clusters. Therefore we will go for partitioning with high modularity Q.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i \times d_j}{2m} \right] \cdot \delta(i, j)$$

In the above equation, **Aij** is the (i, j)-th entry in the adjacency matrix. This implies that i is the number of connections in which i and j are jointly connected. The total number of transactions or the degree of i that node i participated in is di = ∑j Aij. While m = 1 2 ∑ij Aij is the sum of all edge weights in matrix A. When nodes i and j are from the same community, the function δ(i, j) is an indicator equal to 1.0. However, when they are not, the function is zero. Q is bounded in [-1, +1]. When Q > 0, it implies that intra-community connections are more than the expected number.

Modularity

To understand this, we would use a very simple example before exploring the possible different interpretations of modularity. The calculation that would be adopted in our example is based on the measure given by Newman (2006). Also, since we have been using the igraph package in R, our codes to compute modularity would be presented with this package.

To start with, let's assume we have a network of five nodes {A,B,C,D,E}, and the weights of the edges are: A : B = 6, A : C = 5, B : C = 2, C : D = 2, and D : E = 10. Let's assume that a community detection algorithm assigned {A, B, C} to one community and {D, E} to another. This implies that we have only two communities. The adjacent matrix graph for this would be:

$$\{\delta_{ij}\} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Now, let's first detect the communities:

```
> library(igraph)
> A = matrix(c(0,6,5,0,0,6,0,2,0,0,5,2,0,2,0,0,0,2,0,10,0,0,0,10,0),5,5)
> g = graph.adjacency(A,mode="undirected",diag=FALSE)
> wtc = walktrap.community(g)
> res=community.to.membership(g,wtc$merges,steps=3)
> print(res)
$membership
[1] 1 1 1 0 0

$csize
[1] 2 3
```

This can be carried out with another algorithm called the "fast-greedy" approach

```
> g = graph.adjacency(A,mode="undirected",weighted=TRUE,diag=FALSE)
> fgc = fastgreedy.community(g,merges=TRUE,modularity=TRUE,
            weights=E(g)$weight)
> res = community.to.membership(g,fgc$merges,steps=3)
> res
$membership
[1] 0 0 0 1 1

$csize
[1] 3 2
```

The Kronecker delta matrix that examines this community detection would be:

$$\{A_{ij}\} = \begin{bmatrix} 0 & 6 & 5 & 0 & 0 \\ 6 & 0 & 2 & 0 & 0 \\ 5 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 10 \\ 0 & 0 & 0 & 10 & 0 \end{bmatrix}$$

The modularity score would be:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i \times d_j}{2m} \right] \cdot \delta_{ij}$$

Here, the sum of the edge weight in the graph is m = 1/2 ∑ij Aij = 1 2m·δij (13.4) 2 ∑i. Aij is the (i, j)-th entry in the adjacency matrix. This implies the weight of the edge between nodes i and j, while the degree of node **i** is di = ∑j Aij. The Kronecker's delta is the function δij. When the nodes **i** and **j** are from the same community, δij takes the value 1. However, when they are not from the same community, δij takes value zero. Matrix Aij − di× dj 2m is the center of this formula. It entails the modularity, which produces a score that increases when the number of connections within a community is more than the expected proportion of connections if they are assigned at random depending on the degree of each node. The score takes a value ranging from −1 to +1 as it is normalized by dividing by 2m. When Q > 0, it simply means that the number of connections within communities is more than that between communities. Below is the program code that takes in the adjacency matrix and delta matrix:

```
#MODULARITY
Amodularity = function(A, delta) {
    n = length(A[1,])
    d = matrix(0,n,1)
    for (j in 1:n) { d[j] = sum(A[j,]) }
        m = 0.5*sum(d)
        Q = 0
        for (i in 1:n) {
            for (j in 1:n) {
                Q = Q + (A[i,j] - d[i]*d[j]/(2*m))*delta[i,j]
            }
        }
        Q = Q/(2*m)
    }
```

Now, we would be computing modularity in the R package by using a canned function. To do this, we would first enter the two matrices and then call the function. We would also ensure that we get the same function as the formula above:

```
> A = matrix(c(0,6,5,0,0,6,0,2,0,0,5,2,0,2,0,0,0,2,0,10,0,0,0,10,0),5,5)
> delta = matrix(c(1,1,1,0,0,1,1,1,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,1,1),5,5)
> print(Amodularity(A, delta))
```

Next, we would be repeating the same analysis we did with the first program. The aim of this is to show how we can detect communities using walk trap algorithms. Then we will use this community to determine how modularity is computed. The first step to achieve this is to convert the adjacent matrix into a graph so that the community detection algorithms can use it.

```
> g = graph.adjacency(A,mode="undirected",weighted=TRUE, diag=FALSE)
```

We then pass this graph to the walktrap algorithm:

```
> wtc=walktrap.community(g, modularity=TRUE, weights=E(g)$weight)
> res=community.to.membership(g, wtc$merges, steps=3)
> print(res)
$membership
[1] 0 0 0 1 1

$csize
[1] 3 2
```

We would notice that in the above program, the algorithms have separated the first three nodes into one community and the last two nodes into another community. The size variable above shows the size of each community. The next thing to do now is computing the modularity.

```
> print(modularity(g, res$membership, weights=E(g)$weight))
[1] 0.4128
```

This is a confirmation of the value we arrived at when we used the implementation of the formula.

Conclusion

This chapter has extensively covered the rudimentary aspects of a graph in data science. The next chapter examines the Neural Network.

Chapter 14: Neural Networks

In this chapter, we would be treating one of the commonest nonlinear regressions. So far, what we have been concentrating on are linear regressions. This chapter provides a thorough analysis of nonlinear regression through the exploration of Neural Networks. The outlines that would be covered in this chapter include:

- Overview of Neural Networks
- Nonlinear Regression
- Perceptions
- Squashing functions
- Research applications

Overview of Neural Networks

These are some of the forms of nonlinear regression. Recall that in linear regression, we have:

$$Y = X^1\beta + e$$

Here our value for $X \in Rt \times n$, while the solution for regression is simply equal to $\beta = (X^1X)^{-1}(X^1Y)$.

To get this, we simply minimize the sum of squared error:

$$\begin{aligned}
\min_{\beta} e'e &= (Y - X'\beta)'(Y - X'\beta) \\
&= (Y - X'\beta)'Y - (Y - X'\beta)'(X'\beta) \\
&= Y'Y - (X'\beta)'Y - Y'(X'\beta) + \beta^2(X'X) \\
&= Y'Y - 2(X'\beta)'Y + \beta^2(X'X)
\end{aligned}$$

When we different w.r.t. β would give us the following f.o.c:

$$\begin{aligned}
2\beta(X'X) - 2(X'Y) &= \mathbf{0} \\
&\Longrightarrow \\
\beta &= (X'X)^{-1}(X'Y)
\end{aligned}$$

We would be examining this using the markowitzdata.txt data set.

```
> data = read.table("markowitzdata.txt",header=TRUE)
> dim(data)
[1] 1507   10
> names(data)
 [1] "X.DATE" "SUNW"   "MSFT"   "IBM"    "CSCO"   "AMZN"   "mktrf"
 [8] "smb"    "hml"    "rf"
> amzn = as.matrix(data[,6])
> f3 = as.matrix(data[,7:9])
```

```
> res = lm(amzn ~ f3)
> summary(res)

Call:
lm(formula = amzn ~ f3)

Residuals:
      Min        1Q    Median        3Q       Max
-0.225716 -0.014029 -0.001142  0.013335  0.329627

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0015168  0.0009284   1.634  0.10249
f3mktrf      1.4190809  0.1014850  13.983  < 2e-16 ***
f3smb        0.5228436  0.1738084   3.008  0.00267 **
f3hml       -1.1502401  0.2081942  -5.525 3.88e-08 ***
---
Signif. codes:  0 ï£¡***ï£¡ 0.001 ï£¡**ï£¡ 0.01 ï£¡*ï£¡ 0.05 ï£¡.ï£¡ 0.1 ï£¡ ï£¡ 1

Residual standard error: 0.03581 on 1503 degrees of freedom
Multiple R-squared: 0.2233,     Adjusted R-squared: 0.2218
F-statistic: 144.1 on 3 and 1503 DF,  p-value: < 2.2e-16

> wuns = matrix(1,length(amzn),1)
> x = cbind(wuns,f3)
> b = solve(t(x) %*% x) %*% (t(x) %*% amzn)
> b
               [,1]
       0.001516848
mktrf  1.419080894
smb    0.522843591
hml   -1.150240145
```

We would observe that, at the end of the program listing, our formula for the coefficients of the minimized least-squares problem $\beta=(XX)-1(XY)$ matches what we got from the regression command **lm**.

Nonlinear Regression

This type of regression takes this form:

$$Y = f(X;\beta) + e$$

In the above formula, f(.) is a nonlinear function. Computing the coefficient in the nonlinear regression is exactly the same as computing in linear regression.

$$\begin{aligned} \min_{\beta} e'e &= (Y - f(X;\beta))'(Y - f(X;\beta)) \\ &= Y'Y - 2f(X;\beta)'Y + f(X;\beta)'f(X;\beta) \end{aligned}$$

When we differentiate w.r.t. β would give us the following f.o.c:

$$\begin{aligned} -2\left(\frac{df(X;\beta)}{d\beta}\right)'Y + 2\left(\frac{df(X;\beta)}{d\beta}\right)'f(X;\beta) &= \mathbf{0} \\ \left(\frac{df(X;\beta)}{d\beta}\right)'Y &= \left(\frac{df(X;\beta)}{d\beta}\right)'f(X;\beta) \end{aligned}$$

Perceptrons

Neural networks are unique types of nonlinear regressions. This is because, in this type of nonlinear regression, the decision system for which NN is built imitates the same way the human brain works. This implies that the decision system of the Neural Networks works in a perceptive manner. Hence, the basic building block of NN is perceptron.

In Neural Networks, a perceptron is similar to the neuron in the human brain. Like the human brain, it first takes the input like the sensory in a real brain, and they produce an output signal. The entire network of perceptron in Neural Networks is called a neural net.

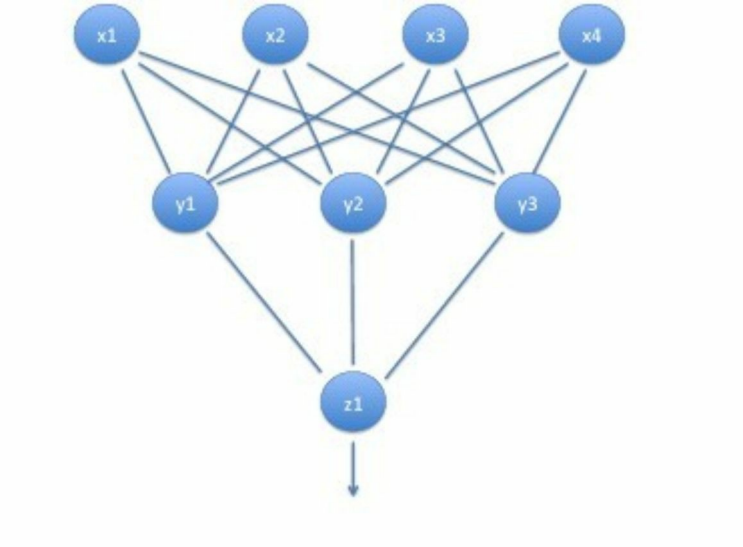
For instance, let's assume we want to carry out an application for a credit card. This would include that we provide some of our personal information like age, business, location, sex, and so on. This information is passed to a series of perceptrons in a parallel form. The layer of the first perceptron is the first "layer" of assessment. Each of the perceptrons then produces an output signal that might be sent to another series of the perceptron to work on; these second series also emit another output signal. The second layer of output is referred to as the "hidden" perceptron layer. After running a series of hidden layers, all the signals are then sent to a single perceptron; this last perceptron produces the signal for deciding whether or not you are qualified for a credit card.

From our explanation, it can be inferred that perceptron may emit continuous signal or binary (0,1) signals. In our credit card example, the perceptron of a signal is a binary one. A binary perceptron is implemented by means of "squashing" functions. A very simple squashing function is the one that issues a 1 if the function value is positive and a 0 if it is negative. The formula for this is highlighted below:

$$S(x) = \begin{cases} 1 & \text{if } g(x) > T \\ 0 & \text{if } g(x) \leq T \end{cases}$$

In the above equation, g(x) represent any function taking the negative or positive value.

When a neural network contains many layers, it is known as a “multi-layered” perceptrons. While all the perceptrons together are referred to as a big, single perceptrons. Below is an example of this kind of neural network:



Neural net models are very similar to Deep Learning. In deep learning, the number of hidden layers is significantly higher than what was usually arrived at in the past when computational power is generally limited. Today, deep learning nets has risen to above 20-30 layers. This resulted in the unique ability of neural nets imitating the same process the human brain works.

Most times, Binary NNs are seen as a category of classier systems. This is because, as a classifier system, they are often used to divide members of a population into different classes. Aside from Binary NNs, NNs with continuous output are fast becoming popular.

Squashing Functions

This is more general than the binary. In simple terms, squashing function is a process whereby the output signal is squashed into a narrow range, usually (0,1). A very common choice of squashing function is the sigmoid function, popularly known as a logistic function. The formula for this is

$$f(x) = \frac{1}{1 + e^{-w\,x}}$$

Where **w** is the adjustable weight. Another very common choice is the Probit function:

$$f\,(x) = \Phi(w\ x)$$

where $\Phi(\cdot)$ is the cumulative normal distribution function.

How do NN works?

The simplest way to see how NN works are to observe the simplest NN. This is simply an NN with a single perceptron producing a binary output. The perceptron has n inputs, with values xi,i = 1...n and current weights wi, i = 1...n. It generates an output y. The “net input” is defined as:

$$\sum_{i=1}^{n} w_i x_i$$

The output signal is y = 0 when the net input is greater than a threshold T. However; this is not the case is if it is less than T if it is less than T, the output is y = 0. The actual output is referred to as the “desired” output and is represented by d = {0,1}. Hence, the “training” data provided to the NN comprises both the inputs xi and the desired output d. The output of our single perceptron model will be the sigmoid function of the net input, which is

$$y = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_i x_i\right)}$$

The error in the NN for a given input set is

$$E = \frac{1}{2}\sum_{j=1}^{m}(y_j - d_j)^2$$

Here, the size of the training set is m in yj. To get the optimal NN, we simply find the optimal NN; we find the weights wi that minimizes this error function E. Once the optimal weights are gotten, we have a calibrated “feedforward” neural net.

The multilayered perceptron for a given squashing function f and input x = [x1,x2,...,xn], would give us an output at the hidden layer of

$$y(x) = f\left(w_0 + \sum_{j=1}^{n} w_j x_j\right)$$

The final output of NN is:

$$z(x) = f\left(w_0 + \sum_{i=1}^{N} w_i \cdot f\left(w_{0i} + \sum_{j=1}^{n} w_{ji} x_j\right)\right)$$

In the above equation, the nested structure of the neural net is obvious.

Logit/Probit Model

A good example of a Logit model is the special model we have above. However, the model becomes a probit regression model once the squashing function is taken to the cumulative normal distribution. However, whether Logit or Probit, the model is fitted by minimizing squared errors, not by maximum likelihood, which is how standard logit/probit models are parameterized.

Connection To Hyperplanes

It is important to note that in binary squashing functions, we passed the net input through a sigmoid function and then compared to the threshold level T. This sigmoid function is a monotone one. Hence, this means that there must be a level T in which the net input ∑ni=1 wi xi must be for the result to be on the cusp. The following is the equation for a hyperplane

$$\sum_{i=1}^{n} w_i x_i = T'$$

This also means that observations in n-dimensional space of the inputs xi must lie on one side or the other of this hyperplane. When it lies above the hyperplane, then y = 1, else y = 0. From our explanation so far, single perceptrons in neural nets have a simple geometrical intuition.

Feedback/Backpropagation

The major difference between ordinary neural net and nonlinear regressions is feedback. Feedback plays a vital role in the neural net performance. Neural nets learn from their feedbacks. The techniques used in implementing feedback is what is called backpropagation.

Assuming what we have is a calibrated NN. We would have to obtain another observation of data and then run it through the NN. To get the error for this observation, we would compare the value of the output y and the desired observation d. If the error detected is a very large one, the best way to correct this is to update the weight in the NN. This would enable it to self-correct. The process of updating the weight in NN to self-correct is what is known as "backpropagation."

The benefit of backpropagation is to avoid the long process of doing a full-refitting task. With just simple rules, correction can be made in a gradual process.

Let's look at backpropagation using a single perceptron on this very simple example. Considering the jth perceptron, the sigmoid value would be:

$$y_j = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_i x_{ij}\right)}$$

where yj is the output of the jth perceptron, and xij is the ith input to the jth perceptron. The error that would be gotten from this observation is (yj − dj). Recall that E = 1/2 ∑mj=1(yj − dj)2. The change in error can be computed with respect to the jth output.

$$\frac{\partial E}{\partial y_j} = y_j - d_j$$

Note also that

$$\frac{dy_j}{dx_{ij}} = y_j(1 - y_j)w_i$$

and

$$\frac{dy_j}{dw_i} = y_j(1 - y_j)x_{ij}$$

Next, we examine how the error changes with input values:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_{ij}} = (y_j - d_j)y_j(1 - y_j)w_i$$

We can now define the value of interest. This is simply the change in error value with respect to the weights

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dw_i} = (y_j - d_j)y_j(1 - y_j)x_{ij}, \forall i$$

In this case, we have a single equation for each observation j and each weight wi. It is important to note that wi apply to all perceptrons. A unique case is where every perceptron has its own perceptron, that is wij. Here, instead of updating just a single observation, updating would be done on many observations. If this is the case, the error derivative would be:

$$\partial E/\partial wi = \sum j(yj - dj)yj(1 - yj)xij$$

Therefore, wi would be reduced to E if $\partial E/\partial wi > 0$. How do we achieve this? It is in this aspect

that some art and judgment is implemented. To start with, when the weight wi needs to be shrunk, there is a tuning parameter $0 < \gamma < 1$ that works for this; similarly, when the derivative is $\partial E/\partial wi < 0$, then wi would be increased by dividing it by γ.

Chapter 15: One Or Zero: Optimal Digital Portfolio

Digital assets are a binary investment. This means that their payoff is either small or large. In this chapter, we would be exploring some key features of optimal digital portfolio assets; these include assets such as credit assets, venture investments and lotteries. The outlines that would be covered in this chapter include:

- Optimal Digital Portfolio
- Modeling Digital Portfolios

Optimal Digital Portfolio

These kinds of portfolio shares correlated assets with joint Bernoulli distributions. In our explanation, we would be using an easy and fast recursive technique to obtain the return on the distribution of the portfolio. We would also be using the example to generate guidelines on how digital asset investors should construct their portfolio. Recently, it has been discovered that portfolios are better when they are constructed homogeneous in the size of the assets.

It is important to note that the distribution of return on digital portfolios is usually fat-tailed and extremely skewed. The venture fund is a very good example of this kind of portfolio. Bernoulli distribution is a simple representation of digital portfolio payoff. Bernoulli distribution has very little or no payoff for a failed asset; however, its payoff for a successful portfolio is large. The probability of achieving success in digital investment is relatively small. Hence, optimizing the portfolio in digital investment is not amenable to the standard technique used for mean-variance optimization.

Therefore, in our explanation, we would be using a technique based on the standard recursive used for modeling the return distribution in Bernoulli distribution.

Modeling Digital Portfolio

Let's take, for instance, that an investor has an option of n investments in digital assets. The investment are indexed i = 1,2,..., n. For each investment, there is a probability of success qi that would yield si dollar. Given the probability (1 − qi), the investment and the start-up would fail. All the money used for the investment would become a total waste. The payout of cashflow for such investment is:

$$\text{Payoff} = C_i = \begin{cases} S_i & \text{with prob } q_i \\ 0 & \text{with prob } (1 - q_i) \end{cases}$$

Below is the function to determine whether an investment is successful or not :

$$y_i = \rho_i X + \sqrt{1 - \rho_i^2}\, Z_i, \quad i = 1 \ldots n$$

Here the ρi ∈ [0,1] is a coefficient that correlates a normalized common factor X ~ N(0,1) with a threshold yi. Correlation is driven among the digital assets in the portfolio by the common factor. Hence, we assume that Zi ~ N(0,1) and Corr(X,Zi) = 0, ∀ i. Therefore, ρi × ρj stands for the correlation between assets i and j.

Note that the mean and variance of yi are: E(yi) = 0,Var(yi) = 1, ∀ i. Conditional on X, the values of yi are all independent, as Corr(Zi,Zj) = 0. Next, we formalize the probability model for the success or failure of the investment by first defining a variable xi with distribution function F(·), such that F(xi) = qi, gives us the probability of success of the digital investment. Conditional on a fixed value of X, the probability of success of the ith investment would be:

$$p_i^X \equiv Pr[y_i < x_i | X]$$

Taking the normal of value to be F, we would have :

$$\begin{aligned} p_i^X &= Pr\left[\rho_i X + \sqrt{1 - \rho_i^2}\, Z_i < x_i | X\right] \\ &= Pr\left[Z_i < \frac{x_i - \rho_i X}{\sqrt{1 - \rho_i^2}} | X\right] \\ &= \Phi\left[\frac{F^{-1}(q_i) - \rho_i X}{\sqrt{1 - \rho_i}}\right] \end{aligned}$$

From the above equation, the cumulative normal density function is Φ(.). Taking the value for the level of the common factor as X, asset correlation ρ, and the unconditional success probabilities qi, our conditional success probability for each asset would be pXi. The more X varies, the more our pXi varies. We would be choosing the function F(xi) as the cumulative normal probability function for the numerical examples we would be treating.

An investment is deemed successful I it has a high payoff Si. The flow of cash from the portfolio is a random variable

$C = \sum_{i=1}^{n} C_i$.

The sum of all digital assets cashflow would give us the maximum cash flow that would be generated by the portfolio. This is because every single outcome is a success.

$$C_{max} = \sum_{i=1}^{n} S_i$$

To simplify the issue, we would simply assume Si is an integer, and we round off the amount nearest to the significant digit. Hence if the amount nearest to the significant digit is a million, each Si would be a million integer.

Let's recall that in our previous formula, conditional on a value of X, the probability of success of digital asset i is given as pXi. This recursion technique would be made it easy for us to generate the portfolio cashflow probability for each level of X. Based on this, we simply use the marginal distribution for X represented by g(X) to compose these conditional (on X) distributions into the unconditional distribution of all the portfolios. As a result, the total probability of cash flow from the portfolio, conditional on X would be f(C|X).

$$f(C) = \int_X f(C|X) \cdot g(X)\, dX$$

Conclusion

Data science is wider than computer science or statistic; however, to excel in the field, the knowledge of these two is very necessary. From our explanations so far, a lot has been explored under these two fields. For instance, Fourier analysis, Data extraction, Limited dependent variables are all under the field of statistics, while algorithms, linear and nonlinear regression, Auctions, Network theories, neural networks and more are prevalent in the field of computer science.

In explaining some of the theories in the book, we use a recursion technique borrowed from many different portfolios and examples. We also explicate the major difference between nonlinear and linear regression, using this as a background into exploring what optimal digital portfolio entails. Some popularly used theories in data science, such as Bass, Bayes, GARCH/ARCH, and many more, were explained in detail. Not only this, we explore different models like the Markowitz model, Eigensystem, Factor analysis and many more.

Important areas such as web sourcing with the use of API, Text classifier, Word-count classifier, and many more were observed in detail. The approaches employed in each chapter of the book are very simple. Broad statistics on models and theories were beaten done into explanatory details so that readers are able to grasp all these areas.

With consistent study and practice, data scientists are guaranteed to excel in their field.