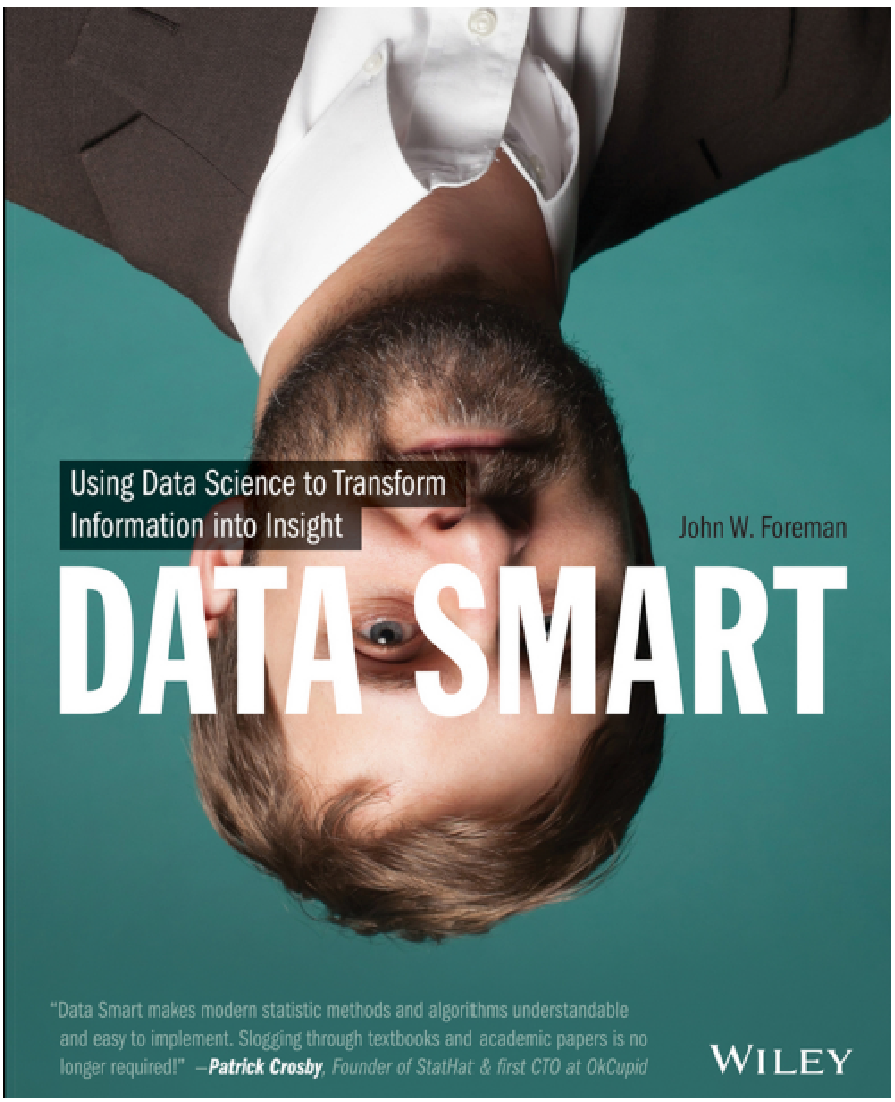
Data Smart
Using Data Science to Transform Information into Insight

RK

April 29, 2014

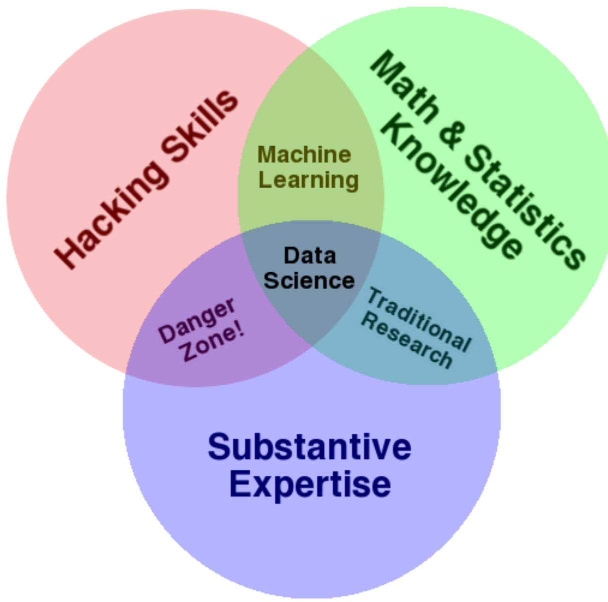
Abstract

The purpose of this document is to summarize the book "Data Smart", written by John W. Foreman and provide some additional R code to work with the book.



Summary

"Data Science" is a very loose word and can mean different things in different situations. However one thing is certain, the principles used in tacking problems are from diverse fields. Drew Conway has this Venn diagram on his blog :



In such a diverse field one does not know where to start and how to start. Someone has made a nice Metromap too. All said and done, this is a field that has considerable entry barriers. One needs to spend at least a few years to get the basics right to understand some basic algorithms.

Where does this book fit in? This book is apt for people who want to see what's going on behind various algorithms without the math. The book touches upon a dozen topics in data mining and explains the main principles of each of those topics via Excel. By restricting to Excel, the author enables a wider audience to get a glimpse of the various concepts. The ideal way to to read this book is by working out the various case studies that are mentioned in the book. I could not motivate myself to do the analysis in Excel, so replicated the analysis in R. In this document I have listed down some of the code to work through the book, that essentially replicates the results of the analysis done via Excel. But first a brief summary of the chapters in the book.

Chapter 1 is on Excel and can be speedread as I cannot imagine someone reading this book without ever working on Excel. **Chapter 2** discusses k-means clustering. It uses an offer-purchases dataset to segment the customers in to various clusters for better marketing. The k-means needs a distance metric and there are many to choose from based on the situation. The book shows that for the specific dataset used, correlation based distance or cosine similarity score is a better metric than euclidean distance.

Chapter 3 is on "Naive Bayes", a simple method that surprisingly performs better than many other algorithms. In fact the reason for its ubiquity stems from its simplicity; it does not overfit the data.Naive Bayes principle is applied on a set of tweets to classify them as business-related or junk. Obviously there is not much of math in this book as expected. So, the results from this chapter will motivate anyone to understand the reason why Naive Bayes works and understand why bias-variance tradeoff works very differently in a classification setting than a regression setting.

Chapter 4 is about optimization, the quintessential skillset that any data scientist needs to have. Using a case study, the author introduces Linear Programming, Integer programming, Mixed Integer programming and ways to convert a nonlinear optimization problem in to Linear Optimization problem. The good thing about this book and this chapter in particular is that there is a good sense of humor that the author brings along while explaining principles. That makes the book an immensely readable book.

Chapter 5 discusses graph analysis and uses the same dataset from one of the previous chapters to do an unsupervised learning. k-neighborhood and Modularity maximization procedures are used to group the customers in to communities. Even though Gephi is used for Visualization, `igraph` is powerful enough to give all the visualization features to an `R` user. **Chapter 6** is about regression. The book uses a sample dataset to explain the concepts of regression and logistic regression.All the creation of dummy variables, setting up the objective function etc. are done in Excel and the reader is made to understand the basic steps behind regression modeling.

Chapter 7 gives the reader an insight in to "wisdom of crowds" type models. The models discussed are Random Forest and Boosting. A reader who reaches until this point of the book is abundantly convinced that Excel is too painful use boosting techniques, where every model built on a bootstrapped sample has to be recorded as a macro and one has to run it manually to get estimates. In any case, the chapter does a wonderful job of explaining the nuts and bolts of Boosting.

Chapter 8 gives a crash course on exponential smoothing. It starts off with simple exponential smoothing and then moves on to Holt's trend-corrected exponential smoothing and finally ending with multiplicative Holt-Winters exponential smoothing. The basic limitation of these models is that there is no probabilistic framework around them. Hyndman has written a book on Exponential smoothing where he casts all the models in a State space framework that makes the models far more richer.

Chapter 9 talks about outlier detection and introduces three methods: indegree method, k-distance method , local outlier factor method. **Chapter 10** introduces some basic commands in R and then works out the k-means model, the regression model, the random forests model, forecasting model and outlier detection methods in R. **Chapter 11** is the concluding chapter in the book that talks about some soft skills that a data scientist should have in order to be effective in an organization.

What's in this document ?

The author does provide `R` code to work through some of the chapters covered in the book. In one sense, he handholds the reader in running functions from various libraries. However there are certain sections of the book for which `R` code is not made available or some different software like Gephi is used. In this document, I have included some additional comments for each of the chapters and `R` code to replicate almost all the analysis that is done through out the book.

Contents

1 Everything You Ever Needed to Know about Spreadsheets but Were Too Afraid to Ask

The first chapter is a basic crash course on excel that teaches common functions to an excel newbie. The functions explained are sort, match, index, offset, small, vlookup, filtering, sorting, pivot tables, array formulas, solver. It also suggests that the reader install OpenSolver(a solver on steroids) plug-in to work through some of the content in the book.

2 Cluster Analysis Part I: Using K-Means to Segment Your Customer Base

This chapter is about clustering a set of customers based on their transactions done in response to various deals. Each deal has certain characteristics and there are p offers over a certain time period. In the ML literature, p is the number of features. Every customer can be viewed in this p dimensional space.The j^{th} component is assigned 0 or 1 based on a customer response to the j^{th} deal. The example describes the algorithm in plain english. However the precise algo is also not very mathematical. It goes like this(for K clusters):

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing
   (a) For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster
   (b) Assign each observation to the cluster whose centroid is closest.

Why does the above work ? For whatever cluster configurations, one needs to formulate an objective function and minimize/maximize it. In this case, you are trying to minimize within-cluster variation. If we denote the within-cluster variation for C_k is a measure $W(C_k)$ then the objective function is

$$\min_{C_1, C_2, \ldots C_K} \sum_{k=1}^{K} W(C_k)$$

A common choice for $W(C_k)$ is

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

The within-cluster variation $W(C_k)$is the sum of pairwise squared Euclidean distances between the observations of all the $k$the cluster.

If you look at the algo and the method that is followed in the book, there is no mention of computing pairwise distances in each cluster. Instead the distance minimized are the distances to the centroid. The key identity that makes the pairwise distance computation redundant in each cluster is the following

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \overline{x}_{kj})^2$$

where $\overline{x}_{jk}$ denotes the mean of feature j in cluster k. Ok, now crunching the data.

```
input.file              <- "data/ch02/deals.csv"
deals                   <- read.csv(input.file,stringsAsFactors = FALSE,header = TRUE)
input.file              <- "data/ch02/offer.csv"
offers                  <- read.csv(input.file,stringsAsFactors = FALSE,header = TRUE)
colnames(offers)        <- c("name","offer")
offers$value            <- 1
offers.data             <- cast(offers, name~offer, sum)[,2:33]
rownames(offers.data) <- cast(offers, name~offer, sum)[,1]
set.seed(1)
km.out                  <- kmeans(offers.data,4,nstart=25)
deals.temp              <- cbind(deals, (t(km.out$centers)))
```

Analyzing based on first cluster centre

```
deals.temp[order(deals.temp$"1",decreasing = TRUE),1:6][1:10,]

##    Campaign      Variety Minimum Discount       Origin PastPeak
## 30 December       Malbec       6       54       France    FALSE
## 29 November Pinot Grigio       6       87       France    FALSE
## 7     March     Prosecco       6       40    Australia     TRUE
## 8     March    Espumante       6       45 South Africa    FALSE
## 18     July    Espumante       6       50       Oregon    FALSE
## 13      May       Merlot       6       43        Chile    FALSE
## 1   January       Malbec      72       56       France    FALSE
## 2   January   Pinot Noir      72       17       France    FALSE
## 3  February    Espumante     144       32       Oregon     TRUE
## 4  February    Champagne      72       48       France     TRUE
```

Analyzing based on second cluster centre

```
deals.temp[order(deals.temp$"2",decreasing = TRUE),1:6][1:10,]

##    Campaign            Variety Minimum Discount Origin PastPeak
## 22   August          Champagne      72       63 France    FALSE
## 31 December          Champagne      72       89 France    FALSE
## 6     March           Prosecco     144       86  Chile    FALSE
## 1   January             Malbec      72       56 France    FALSE
## 11      May          Champagne      72       85 France    FALSE
## 3  February          Espumante     144       32 Oregon     TRUE
## 4  February          Champagne      72       48 France     TRUE
## 14     June             Merlot      72       64  Chile    FALSE
## 15     June Cabernet Sauvignon     144       19  Italy    FALSE
## 30 December             Malbec       6       54 France    FALSE
```

Analyzing based on third cluster centre

```
deals.temp[order(deals.temp$"3",decreasing = TRUE),1:6][1:10,]
```

```
##     Campaign    Variety Minimum Discount       Origin PastPeak
## 24 September Pinot Noir       6       34        Italy    FALSE
## 26   October Pinot Noir     144       83    Australia    FALSE
## 17      July Pinot Noir      12       47      Germany    FALSE
## 2    January Pinot Noir      72       17       France    FALSE
## 1    January     Malbec      72       56       France    FALSE
## 10     April   Prosecco      72       52   California    FALSE
## 23 September Chardonnay     144       39 South Africa    FALSE
## 27   October  Champagne      72       88  New Zealand    FALSE
## 3   February  Espumante     144       32       Oregon     TRUE
## 4   February  Champagne      72       48       France     TRUE
```

Analyzing based on fourth cluster centre

```
deals.temp[order(deals.temp$"4",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign            Variety Minimum Discount       Origin PastPeak
## 31 December          Champagne      72       89       France    FALSE
## 4  February          Champagne      72       48       France     TRUE
## 9     April         Chardonnay     144       57        Chile    FALSE
## 11      May          Champagne      72       85       France    FALSE
## 6     March           Prosecco     144       86        Chile    FALSE
## 8     March          Espumante       6       45 South Africa    FALSE
## 14     June             Merlot      72       64        Chile    FALSE
## 16     June             Merlot      72       88   California    FALSE
## 20   August Cabernet Sauvignon      72       82        Italy    FALSE
## 27  October          Champagne      72       88  New Zealand    FALSE
```

As one can see, the above analysis does not given any conclusive results. Instead one can look at deal counts in each cluster

```
cluster               <-  data.frame(name = (rownames(offers.data)),
                                     cluster = km.out$cluster)
deals.by.cluster      <- merge(offers, cluster, all.x= T)
temp                  <- cast(deals.by.cluster, offer~cluster, sum)
temp                  <- cbind(deals,temp)
```

The first cluster is small timers

```
temp[order(temp$"1",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign      Variety Minimum Discount       Origin PastPeak
## 30 December       Malbec       6       54       France    FALSE
## 29 November Pinot Grigio       6       87       France    FALSE
```

```
## 7      March     Prosecco       6       40    Australia     TRUE
## 8      March    Espumante       6       45 South Africa    FALSE
## 18      July    Espumante       6       50       Oregon    FALSE
## 13       May       Merlot       6       43        Chile    FALSE
## 1    January       Malbec      72       56       France    FALSE
## 2    January   Pinot Noir      72       17       France    FALSE
## 3   February    Espumante     144       32       Oregon     TRUE
## 4   February    Champagne      72       48       France     TRUE
```

The second cluster is not clear

```
temp[order(temp$"2",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign            Variety Minimum Discount Origin PastPeak
## 22   August          Champagne      72       63 France    FALSE
## 31 December          Champagne      72       89 France    FALSE
## 6     March           Prosecco     144       86  Chile    FALSE
## 1   January             Malbec      72       56 France    FALSE
## 11      May          Champagne      72       85 France    FALSE
## 3  February          Espumante     144       32 Oregon     TRUE
## 4  February          Champagne      72       48 France     TRUE
## 14     June             Merlot      72       64  Chile    FALSE
## 15     June Cabernet Sauvignon     144       19  Italy    FALSE
## 30 December             Malbec       6       54 France    FALSE
```

The third cluster is Pinot Noir variety

```
temp[order(temp$"3",decreasing = TRUE),1:6][1:10,]
```

```
##     Campaign    Variety Minimum Discount       Origin PastPeak
## 24 September Pinot Noir       6       34        Italy    FALSE
## 26   October Pinot Noir     144       83    Australia    FALSE
## 17      July Pinot Noir      12       47      Germany    FALSE
## 2    January Pinot Noir      72       17       France    FALSE
## 1    January     Malbec      72       56       France    FALSE
## 10     April   Prosecco      72       52   California    FALSE
## 23 September Chardonnay     144       39 South Africa    FALSE
## 27   October  Champagne      72       88  New Zealand    FALSE
## 3   February  Espumante     144       32       Oregon     TRUE
## 4   February  Champagne      72       48       France     TRUE
```

The fourth cluster seems to like August Champaign

```
temp[order(temp$"4",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign            Variety Minimum Discount       Origin PastPeak
```

```
## 31 December           Champagne      72        89       France    FALSE
## 4  February           Champagne      72        48       France     TRUE
## 9      April        Chardonnay     144        57        Chile    FALSE
## 11       May          Champagne      72        85       France    FALSE
## 6      March           Prosecco     144        86        Chile    FALSE
## 8      March          Espumante       6        45 South Africa    FALSE
## 14      June             Merlot      72        64        Chile    FALSE
## 16      June             Merlot      72        88   California    FALSE
## 20    August Cabernet Sauvignon      72        82        Italy    FALSE
## 27   October          Champagne      72        88  New Zealand    FALSE
```

One can try K means for varying K and pick one of the k values. The chapter suggests another way to compare the K means across various k values, i.e by computing a score for your clusters called the *silhouette*. The following `R` code gives the metric. You can also use the `silhouette` function from the `cluster` package.

```
silhouette.rk     <- function(cluster,dist.euclidean){
  clusters          <- sort(unique(cluster$cluster))
  silh              <- numeric()

  for(i in cluster$id){
  temp              <- subset(cluster, id!=i)
  temp.cluster      <- subset(cluster, id==i)$cluster
  same.cluster      <- subset(temp, cluster == temp.cluster)
  diff.cluster      <- subset(temp, cluster != temp.cluster)
  i.star            <- pmin(i,same.cluster$id)
  j.star            <- pmax(i,same.cluster$id)
  within            <- mean(dist.euclidean[ n*(i.star-1) -
                       i.star*(i.star-1)/2 + j.star-i.star ])
  neighbor          <- min( sapply( clusters[-temp.cluster],function(j)
         {
    i.star  <- pmin(i,subset(diff.cluster, cluster== j)$id)
    j.star  <- pmax(i,subset(diff.cluster, cluster== j)$id)
    mean(dist.euclidean[ n*(i.star-1) - i.star*(i.star-1)/2 + j.star-i.star ])
        }
        ) )
    silh <- c(silh , (neighbor-within)/max(within, neighbor))
  }
  mean(silh)
}
```

For $K = 4$ clusters, one can calculate silhouette as follows:

```
set.seed(1)
dist.euclidean    <- dist(offers.data)
n                 <- attr(dist.euclidean, "Size")
```

```
km.out              <- kmeans(offers.data,4,nstart=25)
cluster             <-  data.frame(name = (rownames(offers.data)),
                        cluster = km.out$cluster, id = 1:nrow(cluster) )
print(silhouette.rk(cluster,dist.euclidean))

## [1] 0.1243

print((summary(silhouette(km.out$cluster,dist.euclidean)))$avg.width)

## [1] 0.1243
```

For $K = 5$ clusters, one can calculate silhouette as follows:

```
set.seed(1)
km.out                  <- kmeans(offers.data,5,nstart=25)
cluster             <-  data.frame(name = (rownames(offers.data)),
                                       cluster = km.out$cluster, id = 1:nrow(cluster) )
print(silhouette.rk(cluster,dist.euclidean))

## [1] 0.1231

print((summary(silhouette(km.out$cluster,dist.euclidean)))$avg.width)

## [1] 0.1231
```

The above metric shows that 5 clusters is no better than 4 clusters.
The chapter subsequently introduces a different way to do K means clustering, i.e. Spherical K means. This is a method where the dissimilarity measure is based on *correlation-based distance*. The package in `R` that does spherical K means is `skmeans`.

```
set.seed(1)
sk.out                  <- skmeans(as.matrix(offers.data), 5, method="genetic")
cluster                 <-  data.frame(name = (rownames(offers.data)),
                                       cluster = sk.out$cluster, id = 1:nrow(cluster) )
deals.by.cluster        <- merge(offers, cluster, all.x= T)
temp                    <- cast(deals.by.cluster, offer~cluster, sum)
temp                    <- cbind(deals,temp)
temp                    <- cbind(deals,temp)
```

The first cluster is Pinot Noir gang

```
temp[order(temp$"1",decreasing = TRUE),1:6][1:10,]

##     Campaign    Variety Minimum Discount       Origin PastPeak
## 24 September Pinot Noir       6       34        Italy    FALSE
## 26   October Pinot Noir     144       83    Australia    FALSE
## 2    January Pinot Noir      72       17       France    FALSE
```

```
## 17      July Pinot Noir      12        47      Germany    FALSE
## 1    January      Malbec      72        56       France    FALSE
## 10     April    Prosecco      72        52   California    FALSE
## 12       May    Prosecco      72        83    Australia    FALSE
## 16      June      Merlot      72        88   California    FALSE
## 23 September Chardonnay     144        39 South Africa    FALSE
## 27   October  Champagne      72        88  New Zealand    FALSE
```

The second cluster looks like small timers

```
temp[order(temp$"2",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign      Variety Minimum Discount       Origin PastPeak
## 8      March    Espumante       6       45 South Africa    FALSE
## 30 December       Malbec       6       54       France    FALSE
## 18      July    Espumante       6       50       Oregon    FALSE
## 29 November Pinot Grigio       6       87       France    FALSE
## 7      March     Prosecco       6       40    Australia     TRUE
## 13       May       Merlot       6       43        Chile    FALSE
## 6      March     Prosecco     144       86        Chile    FALSE
## 10     April     Prosecco      72       52   California    FALSE
## 11       May    Champagne      72       85       France    FALSE
## 21    August    Champagne      12       50   California    FALSE
```

The third cluster is is high volume deals segment

```
temp[order(temp$"3",decreasing = TRUE),1:6][1:10,]
```

```
##     Campaign            Variety Minimum Discount       Origin PastPeak
## 9      April         Chardonnay     144       57        Chile    FALSE
## 14      June             Merlot      72       64        Chile    FALSE
## 1    January             Malbec      72       56       France    FALSE
## 5   February Cabernet Sauvignon     144       44  New Zealand     TRUE
## 4   February          Champagne      72       48       France     TRUE
## 11       May          Champagne      72       85       France    FALSE
## 15      June Cabernet Sauvignon     144       19        Italy    FALSE
## 23 September         Chardonnay     144       39 South Africa    FALSE
## 26   October         Pinot Noir     144       83    Australia    FALSE
## 6      March           Prosecco     144       86        Chile    FALSE
```

The fourth cluster is France buyer segment

```
temp[order(temp$"4",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign            Variety Minimum Discount       Origin PastPeak
## 22   August          Champagne      72       63       France    FALSE
```

```
## 31 December           Champagne      72        89       France    FALSE
## 4  February           Champagne      72        48       France     TRUE
## 6      March          Prosecco     144        86        Chile    FALSE
## 11       May           Champagne      72        85       France    FALSE
## 3  February           Espumante     144        32       Oregon     TRUE
## 27  October           Champagne      72        88  New Zealand    FALSE
## 28 November Cabernet Sauvignon      12        56       France     TRUE
## 1   January              Malbec      72        56       France    FALSE
## 8      March           Espumante       6        45 South Africa    FALSE
```

The fifth cluster are those who buy only sparkling wine

```
temp[order(temp$"5",decreasing = TRUE),1:6][1:10,]

##    Campaign            Variety Minimum Discount     Origin PastPeak
## 7      March           Prosecco       6       40  Australia     TRUE
## 29 November       Pinot Grigio       6       87     France    FALSE
## 30 December             Malbec       6       54     France    FALSE
## 18      July          Espumante       6       50     Oregon    FALSE
## 10     April           Prosecco      72       52 California    FALSE
## 13       May             Merlot       6       43      Chile    FALSE
## 3  February          Espumante     144       32     Oregon     TRUE
## 12       May           Prosecco      72       83  Australia    FALSE
## 19      July          Champagne      12       66    Germany    FALSE
## 28 November Cabernet Sauvignon      12       56     France     TRUE
```

3 Naive Bayes and the Incredible Lightness of Being an Idiot

This chapter talks about using a particular form of Bayes theorem that makes it extremely easy to filter specific type of messages. The Author uses a set of tweets that has words relating to an App. Needless to say there is a lot of noise in twitter data. *Given a tweet, how does one identify whether the tweet is about a specific App or about something else ?.* This question is taken up in the chapter where Bayes is used to compute the posterior probabilities of a App given a tweet and posterior probabilities of a Noise given a tweet, compares both the probabilities and assigns it to the respective class. The logic is extremely simple. It is based on *bag of words* model. The following are the basic steps :

Model Building

- Collect all the words from the app tweets and compute the likelihood of the word given it comes from an app tweet
- Collect all the words from the non-app tweets and compute the likelihood of the word given it comes from an non-app tweet
- To deal with floating point underflow, calculate log likelihood of all probabilities
- Do additive smoothing to take care of rare words

Model Prediction

- For any test tweet, compute the posterior probability of app given test tweet and compute the posterior probability of non-app given test tweet
- Compute the Bayes factor and report the prediction class

The whole model can be coded in a few lines of R code

```
cleanup <- function(x){
  x     <-  gsub("[.]\\s"," ",x)
  x     <- gsub("[?!,;]"," ",x)
  x     <- strsplit(x,"(\\s)+")
  return(x[[1]])
}

getFrequencyCounts   <- function(input.file){
  tweets             <- read.csv(input.file,stringsAsFactors = FALSE,header = FALSE)
  tweets             <- apply(tweets, 1, tolower)
  words              <- sapply(tweets,cleanup)
  words              <- do.call("c",words)
  wordcount          <- table(words)
  wordfreq           <- as.data.frame(wordcount)
  colnames(wordfreq) <- c("words","freq")
  wordfreq$freq      <- wordfreq$freq + 1
  total              <- sum(wordfreq$freq)
  wordfreq$logprob   <- log(wordfreq$freq/total)
  result             <- list(logprob = wordfreq, total = total)
  return(result)
}
```

```
input.file            <- "data/ch03/app.csv"
app.res               <- getFrequencyCounts(input.file)
input.file            <- "data/ch03/other.csv"
other.res             <- getFrequencyCounts(input.file)
input.file            <- "data/ch03/test_set.csv"
tweets                <- read.csv(input.file,stringsAsFactors = FALSE,header = FALSE)
tweets                <- apply(tweets, 1, tolower)
words                 <- sapply(tweets,cleanup)

getScores             <- function(input){
  temp                <- data.frame ( words=input)
  temp1               <- merge(temp,app.res$logprob, all.x=T)
  temp1$logprob[is.na(temp1$logprob)] <- -log(app.res$total)
  app.score           <- sum(temp1$logprob)
  temp2               <- merge(temp,other.res$logprob, all.x=T)
  temp2$logprob[is.na(temp2$logprob)] <- -log(other.res$total)
  other.score        <- sum(temp2$logprob)
  return(app.score > other.score)
}
predicted.result              <- sapply(words,getScores)
attributes(predicted.result)  <- NULL
model.predict                 <- cbind(ifelse(predicted.result==TRUE,"APP","OTHERS"))
actual.result                 <- c(rep(TRUE,10),rep(FALSE,10))
model.actual                  <- c(rep("APP",10),rep("OTHERS",10))
model.comparison              <- data.frame(predict = model.predict, actual = model.actual)
model.comparison$value        <- 1
```

Here is the confusion matrix

```
print(cast(model.comparison,predict~actual,length))

##   predict APP OTHERS
## 1     APP  10      1
## 2  OTHERS   0      9
```

Naive Bayes seems to do be doing really well for this toydata set with just 1 false positive.

4 Optimization Modeling: Because That "Fresh Squeezed" Orange Juice Ain't Gonna Blend Itself

The author explains the principles of LP via a case study. The case involves deciding the procurement of juice in the Jan, Feb, Mar across 11 varieties. There a set of constraints under which the procurement can be made.

```
input.file     <- "data/ch04/specs.csv"
data           <- read.csv(input.file,stringsAsFactors = FALSE,header = TRUE)
colnames(data) <- c("variety","region","qty",
                    "ba","acid","astr","colr",
                    "price","shipping")
head(data)

##    variety     region  qty   ba acid astr colr price shipping
## 1   Hamlin     Brazil  672 10.5 0.01    3    3   500      100
## 2  Mosambi      India  400  6.5 0.01    7    1   310      150
## 3 Valencia    Florida 1200 12.0 0.01    3    3   750        0
## 4   Hamlin California  168 11.0 0.01    3    5   600       60
## 5  Gardner    Arizona   84 12.0 0.01    1    5   600       75
## 6  Sunstar      Texas  210 10.0 0.01    1    5   625       50
```

Objective & Constraints

```
obj     <- rep(data$price + data$shipping, times = 3)
lhs     <- matrix(data = 0, nrow= 100, ncol = 33)
rhs     <- matrix(data = 0, nrow= 100, ncol = 1)
dir     <- matrix(data = "", nrow= 100, ncol = 1)
reqd    <- c(600,600,700)
lt      <- "<="
gt      <- ">="
et      <- "="
j       <- 1

# Total procurement constraint
for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }

  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }

  if(k==3){
```

```
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  lhs[j,] <- temp
  rhs[j]  <- reqd[k]
  dir[j]  <- et
  j       <- j+1

}



# Valencia constraint
lhs[j,] <- c(rep(0,2),1,rep(0,8), rep(0,22))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,11),rep(0,2),1,rep(0,8), rep(0,11))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,22),rep(0,2),1,rep(0,8) )
rhs[j]  <- 280
dir[j]  <- gt
j       <- j+1



# Availability
j           <- 7
for(i in 1:11) {
  x         <-rep(0,11)
  x[i]      <- 1
  lhs[j,]   <- rep(x,times= 3)
  dir[j]    <- lt
  rhs[j]    <- data$qty[i]
  j         <- j+1
}

lowerlim  <- c(11.5,0.0075,0,4.5)
upperlim  <- c(12.5,0.01,4,5.5)

for(k in 1:3) {
  if(k==1){
```

```
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }
  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }
  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  l <- 1
  for(l in 1:4){
    temp1    <- temp*data[,l+3]/reqd[k]
    lhs[j,]  <- temp1
    rhs[j]   <- lowerlim[l]
    dir[j]   <- gt
    j        <- j+1
  }
  u <- 1
  for(u in 1:4){
    temp2     <- temp*data[,u+3]/reqd[k]
    lhs[j,]   <- temp2
    rhs[j]    <- upperlim[u]
    dir[j]    <- lt
    j         <- j+1
  }

}
lhs           <- lhs[1:(j-1),]
rhs           <- rhs[1:(j-1)]
dir           <- dir[1:(j-1)]
sol           <- lp(objective.in = obj, const.mat = lhs,
                   const.rhs = rhs,const.dir = dir)
df            <- (matrix(sol$sol, nrow=11, ncol = 3))
colnames(df) <- c("Jan","Feb","Mar")
rownames(df) <- as.character(data$variety)
sol$objval

## [1] 1226505
```

The solution from solving the constraint optimization is

```
df

##                     Jan Feb    Mar
## Hamlin             0.00   0   0.00
## Mosambi            0.00   0   0.00
```

```
## Valencia       240.00 240 280.00
## Hamlin         124.59   0  43.41
## Gardner          0.00  84   0.00
## Sunstar          3.00   0   0.00
## Jincheng        27.00   0   0.00
## Berna           49.76   0 118.24
## Verna           89.65 138  72.35
## Biondo Commune   0.00  24 186.00
## Belladonna      66.00 114   0.00
```

Objective becomes constraint

The previous LP problem is cast in recast where there is a constraint on the budget and there is leeway in relaxing quality standards.

```
lhs      <- matrix(data = 0, nrow= 100, ncol = 33)
rhs      <- matrix(data = 0, nrow= 100, ncol = 1)
dir      <- matrix(data = "", nrow= 100, ncol = 1)
reqd     <- c(600,600,700)
lt       <- "<="
gt       <- ">="
et       <- "="
j        <- 1

# Total procurement constraint
for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }

  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }

  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  lhs[j,] <- temp
  rhs[j]  <- reqd[k]
  dir[j]  <- et
  j       <- j+1

}
```

```
# Valencia constraint
lhs[j,] <- c(rep(0,2),1,rep(0,8), rep(0,22))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,11),rep(0,2),1,rep(0,8), rep(0,11))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,22),rep(0,2),1,rep(0,8) )
rhs[j]  <- 280
dir[j]  <- gt
j       <- j+1

# Availability
j          <- 7
for(i in 1:11) {
  x        <-rep(0,11)
  x[i]     <- 1
  lhs[j,]  <- rep(x,times= 3)
  dir[j]   <- lt
  rhs[j]   <- data$qty[i]
  j        <- j+1
}

extra.var  <- matrix(0,nrow = 100, ncol = 4)
lowerlim   <- c(11.5,0.0075,0,4.5)
upperlim   <- c(12.5,0.01,4,5.5)
delta      <- upperlim- lowerlim

for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }
  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }
  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
```

```
  }
  l <- 1
  for(l in 1:4){
    temp1   <- temp*data[,l+3]/reqd[k]
    lhs[j,] <- temp1
    extra.var[j,l] <- delta[l]
    rhs[j]  <- lowerlim[l]
    dir[j]  <- gt
    j       <- j+1
  }
  u <- 1
  for(u in 1:4){
    temp2    <- temp*data[,u+3]/reqd[k]
    lhs[j,]  <- temp2
    extra.var[j,u] <- - delta[u]
    rhs[j]   <- upperlim[u]
    dir[j]   <- lt
    j        <- j+1
  }

}
lhs[j,]      <- rep(data$price + data$shipping, times = 3)
dir[j,]      <- lt
BUDGET       <-  1170000

rhs[j]       <- BUDGET

lhs          <- lhs[1:j,]
rhs          <- rhs[1:j]
dir          <- dir[1:j]
extra.var    <- extra.var[1:j,]
lhs          <- cbind(lhs, extra.var)
obj          <- c(rep(0,33),rep(0.25,4))


sol          <- lp(objective.in = obj, const.mat = lhs,
                   const.rhs = rhs,const.dir = dir)
df           <- (matrix(sol$sol[1:33], nrow=11, ncol = 3))
relax        <- (matrix(sol$sol[34:37], nrow=4, ncol = 1))
colnames(df) <- c("Jan","Feb","Mar")
rownames(df) <- as.character(data$variety)
sol$objval

## [1] 0.2793
```

For the specified budget, the relaxation in quality constraints is

```
df

##                    Jan     Feb    Mar
## Hamlin            0.00   0.000   0.00
## Mosambi          68.53   2.842 216.63
## Valencia        240.00 240.000 280.00
## Hamlin            0.00   0.000   0.00
## Gardner           0.00   0.000   0.00
## Sunstar           0.00   0.000   0.00
## Jincheng          0.00   0.000   0.00
## Berna            13.89   0.000 148.11
## Verna           111.79 188.211   0.00
## Biondo Commune   94.89 115.105   0.00
## Belladonna       70.89  53.842  55.26

relax

##        [,1]
## [1,] 0.4095
## [2,] 0.0000
## [3,] 0.0000
## [4,] 0.5874

mean(relax)

## [1] 0.2492
```

One can also draw a curve between various budget levels and quality deterioration.

```
BUDGET       <- seq(1226505,1110000,-10000)
quality      <- numeric(length(BUDGET))
for(b in seq_along(BUDGET)){
  rhs[j]       <- BUDGET[b]
  sol          <- lp(objective.in = obj, const.mat = lhs,
                     const.rhs = rhs,const.dir = dir)
  relax        <- (mean(sol$sol[34:37]))
  quality[b]   <-  relax
}
```

```
par(mfrow=c(1,1))
plot(BUDGET/1000,quality, type ="l", xlab="K$",
     ylab = "Broadening of Quality  bands ",col="blue",cex.lab=0.8)
```

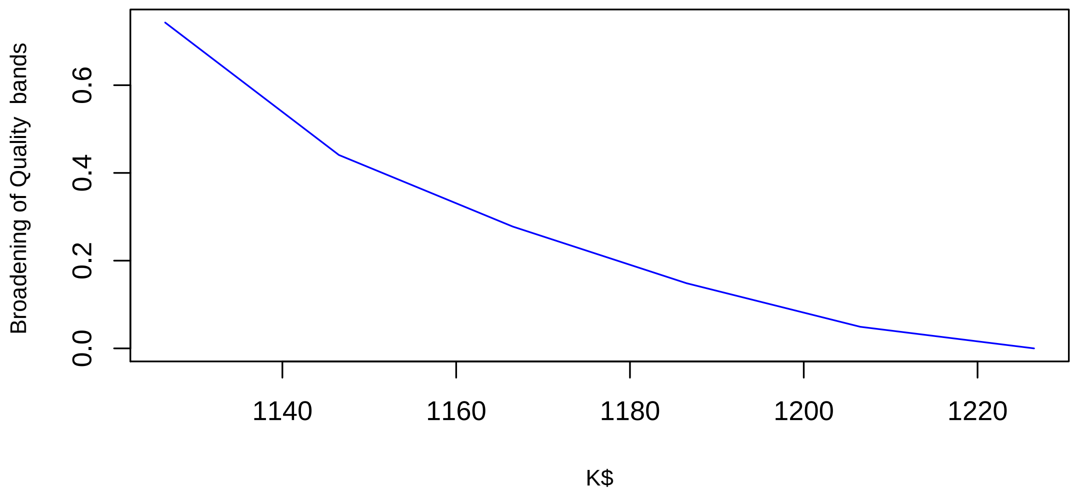


Figure 4.1: Quality Vs Cost

MiniMax

```
lhs     <- matrix(data = 0, nrow= 100, ncol = 33)
rhs     <- matrix(data = 0, nrow= 100, ncol = 1)
dir     <- matrix(data = "", nrow= 100, ncol = 1)
reqd    <- c(600,600,700)
lt      <- "<="
gt      <- ">="
et      <- "="
j       <- 1

# Total procurement constraint
for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }

  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }

  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
```

```
  lhs[j,] <- temp
  rhs[j]  <- reqd[k]
  dir[j]  <- et
  j       <- j+1

}



# Valencia constraint
lhs[j,] <- c(rep(0,2),1,rep(0,8), rep(0,22))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,11),rep(0,2),1,rep(0,8), rep(0,11))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,22),rep(0,2),1,rep(0,8) )
rhs[j]  <- 280
dir[j]  <- gt
j       <- j+1

# Availability
j          <- 7
for(i in 1:11) {
  x        <-rep(0,11)
  x[i]     <- 1
  lhs[j,]  <- rep(x,times= 3)
  dir[j]   <- lt
  rhs[j]   <- data$qty[i]
  j        <- j+1
}

extra.var  <- matrix(0,nrow = 100, ncol = 4)
lowerlim   <- c(11.5,0.0075,0,4.5)
upperlim   <- c(12.5,0.01,4,5.5)
delta      <- upperlim- lowerlim

for(k in 1:3) {

  if(k==1){
```

```
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }
  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }
  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  l <- 1
  for(l in 1:4){
    temp1   <- temp*data[,l+3]/reqd[k]
    lhs[j,] <- temp1
    extra.var[j,l] <- delta[l]
    rhs[j]  <- lowerlim[l]
    dir[j]  <- gt
    j       <- j+1
  }
  u <- 1
  for(u in 1:4){
    temp2    <- temp*data[,u+3]/reqd[k]
    lhs[j,]  <- temp2
    extra.var[j,u] <- - delta[u]
    rhs[j]   <- upperlim[u]
    dir[j]   <- lt
    j        <- j+1
  }

}

lhs[j,]           <- rep(data$price + data$shipping, times = 3)
dir[j]            <- lt
BUDGET            <-  1170000
rhs[j]            <- BUDGET

lhs               <- lhs[1:(j+4),]
extra.var         <- extra.var[1:(j+4),]
extra.var[(j+1):(j+4),c(1:4)] <- diag(4)
dir[(j+1):(j+4)] <- lt
max.var          <- c(rep(0,j), rep(-1,4) )

lhs              <- cbind(lhs, extra.var,max.var)
rhs              <- rhs[1:(j+4)]
dir              <- dir[1:(j+4)]
```

```
obj             <- c(rep(0,37),1)

sol             <- lp(objective.in = obj, const.mat = lhs,
                  const.rhs = rhs,const.dir = dir)
df              <- (matrix(sol$sol[1:33], nrow=11, ncol = 3))
relax           <- (matrix(sol$sol[34:37], nrow=4, ncol = 1))
colnames(df)    <- c("Jan","Feb","Mar")
rownames(df)    <- as.character(data$variety)
sol$objval

## [1] 0.5874
```

For the specified budget, the relaxation in quality constraints is

```
df

##                    Jan    Feb    Mar
## Hamlin            0.00   0.00   0.00
## Mosambi          73.11   0.00 214.89
## Valencia        240.00 240.00 280.00
## Hamlin            0.00   0.00   0.00
## Gardner           0.00   0.00   0.00
## Sunstar           0.00   0.00   0.00
## Jincheng          0.00   0.00   0.00
## Berna             0.00   0.00 162.00
## Verna           109.52 190.48   0.00
## Biondo Commune   92.62 117.38   0.00
## Belladonna       84.76  52.14  43.11

relax

##        [,1]
## [1,] 0.5874
## [2,] 0.0000
## [3,] 0.0000
## [4,] 0.5874
```

For some reason, I could not match the exact results for this section as given in the book. The book uses OpenSolver plugin. The author then goes in to integer and mixed integer programming for the same dataset.

Mixed Integer Programming

```
lhs     <- matrix(data = 0, nrow= 100, ncol = 33)
rhs     <- matrix(data = 0, nrow= 100, ncol = 1)
dir     <- matrix(data = "", nrow= 100, ncol = 1)
```

```
reqd    <- c(600,600,700)
lt      <- "<="
gt      <- ">="
et      <- "="
j       <- 1

# Total procurement constraint
for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }

  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }

  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  lhs[j,] <- temp
  rhs[j]  <- reqd[k]
  dir[j]  <- et
  j       <- j+1

}


# Valencia constraint
lhs[j,] <- c(rep(0,2),1,rep(0,8), rep(0,22))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,11),rep(0,2),1,rep(0,8), rep(0,11))
rhs[j]  <- 240
dir[j]  <- gt
j       <- j+1

lhs[j,] <- c(rep(0,22),rep(0,2),1,rep(0,8) )
rhs[j]  <- 280
dir[j]  <- gt
j       <- j+1
```

```
# Availability
j           <- 7
for(i in 1:11) {
  x         <-rep(0,11)
  x[i]      <- 1
  lhs[j,]   <- rep(x,times= 3)
  dir[j]    <- lt
  rhs[j]    <- data$qty[i]
  j         <- j+1
}

lowerlim  <- c(11.5,0.0075,0,4.5)
upperlim  <- c(12.5,0.01,4,5.5)

for(k in 1:3) {
  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }
  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }
  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  l <- 1
  for(l in 1:4){
    temp1    <- temp*data[,l+3]/reqd[k]
    lhs[j,] <- temp1
    rhs[j]   <- lowerlim[l]
    dir[j]   <- gt
    j        <- j+1
  }
  u <- 1
  for(u in 1:4){
    temp2     <- temp*data[,u+3]/reqd[k]
    lhs[j,]   <- temp2
    rhs[j]    <- upperlim[u]
    dir[j]    <- lt
    j         <- j+1
  }

}
```

```
lhs          <- lhs[1:(j-1),]
rhs          <- rhs[1:(j-1)]
dir          <- dir[1:(j-1)]
#---------- ADD THE INTEGER CONSTRAINTS

lhs1     <- matrix(data = 0, nrow= 100, ncol = 33)
rhs1     <- matrix(data = 0, nrow= 100, ncol = 1)
dir1     <- matrix(data = "", nrow= 100, ncol = 1)
j        <- 1

for(k in 1:3) {

  if(k==1){
    temp = c(rep(1,11), rep(0,11),rep(0,11))
  }

  if(k==2){
    temp = c(rep(0,11), rep(1,11),rep(0,11))
  }

  if(k==3){
    temp = c(rep(0,11), rep(0,11),rep(1,11))
  }
  lhs1[j,] <- temp
  rhs1[j]  <- 4
  dir1[j]  <- lt
  j       <- j+1
}

lhs1          <- lhs1[1:(j-1),]
rhs1          <- rhs1[1:(j-1)]
dir1          <- dir1[1:(j-1)]

lhs2         <- cbind(lhs,matrix(0,nrow = dim(lhs)[1],ncol=33))
lhs3         <- cbind(matrix(0,nrow = dim(lhs1)[1],ncol=33),lhs1)

lhs4          <- rbind(lhs2,lhs3)
rhs2          <- c(rhs,rhs1)
dir2          <- c(dir,dir1)

i             <- 1
```

```
for(i in 1:33) {
  x        <- rep(0,33)
  x[i]     <- 1
  y        <- rep(0,33)
  y[i]     <- 1
  lhs4     <- rbind(lhs4,c(-x,y*data$qty))
  dir2     <- c(dir2,gt)
  rhs2     <- c(rhs2,0)
}


obj           <- c(rep(data$price + data$shipping, times = 3),rep(0,33))
bvec          <- 34:66
sol           <- lp(objective.in = obj, const.mat = lhs4,
                     const.rhs = rhs2,const.dir = dir2,binary.vec = bvec)
df            <- (matrix(sol$sol, nrow=11, ncol = 3))
colnames(df) <- c("Jan","Feb","Mar")
rownames(df) <- as.character(data$variety)
sol$objval

## [1] 1230285
```

The solution from solving the constraint optimization that there should be only four suppliers every month.This objective values is better than that given in the book.

```
round(df)

##                 Jan Feb Mar
## Hamlin            0   0   0
## Mosambi           0   0   0
## Valencia        240 249 280
## Hamlin            0 168   0
## Gardner           0   0   0
## Sunstar           0   0 105
## Jincheng          0   0   0
## Berna             0   0 168
## Verna            70  83 147
## Biondo Commune  210   0   0
## Belladonna       80 100   0
```

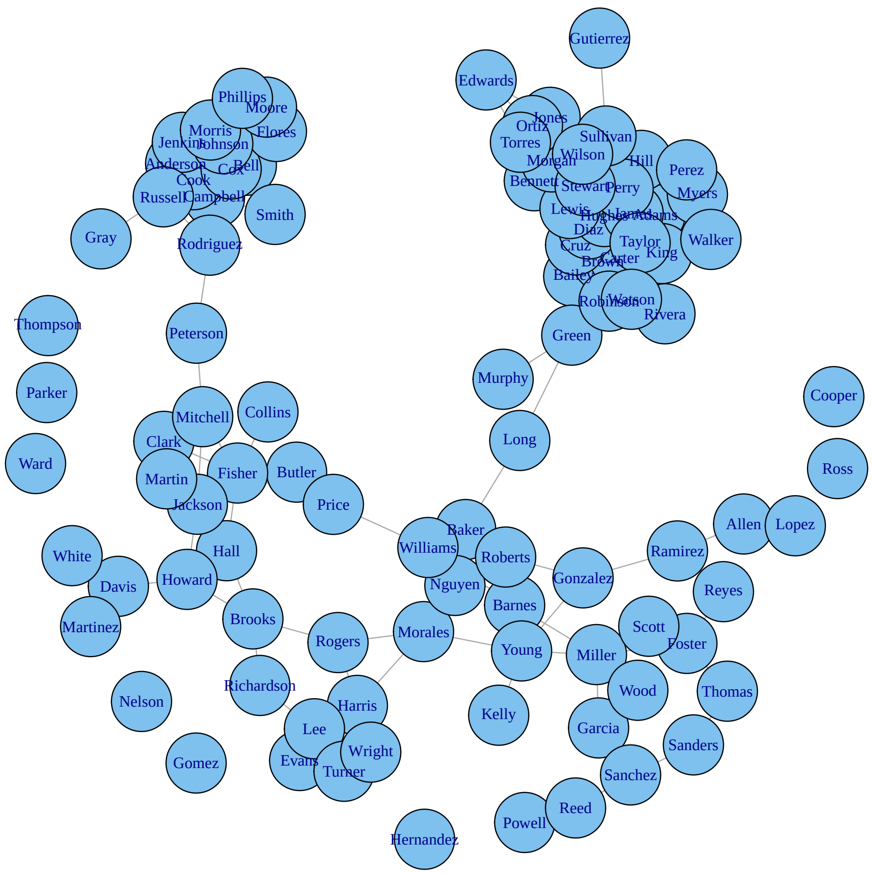
Towards the end of the section, the chapter talks about risk and using basic simulation , the author tries to get a handle on the risk distribution. I think the great thing about this chapter is the way the author communicates the various ideas of linear programming using excel effortlessly with a healthy dose of humor.

5 Cluster Analysis Part II : Network Graphs and Community Detection

This chapter introduces Graph analysis via Excel and Graph visualization via `Gephi`. However one can stay with in `R` and do the analysis and visualization all at one go. `igraph` package in `R` has extensive functionality for graph processing. The example used to illustrate the principles is the same example that is used in the chapter on clustering, i.e. wine data prospect clustering.

```
input.file          <- "data/ch02/deals.csv"
deals               <- read.csv(input.file,stringsAsFactors = FALSE,header = TRUE)
input.file          <- "data/ch02/offer.csv"
offers              <- read.csv(input.file,stringsAsFactors = FALSE,header = TRUE)
colnames(offers)    <- c("name","offer")
offers$value        <- 1
offers.data         <- cast(offers, name~offer, sum)[,2:33]
rownames(offers.data) <- cast(offers, name~offer, sum)[,1]
dist.cosine         <- dist(offers.data, method="cosine")
mat                 <- 1- as.matrix(dist.cosine)
diag(mat)           <- 0
graph.wine          <- graph.adjacency(mat,mode = c("undirected"))
pct.80              <- quantile(mat[which(mat>0)],prob=0.8)
mat                 <- apply(mat,1,function(z) ifelse(z>pct.80,1,0))
rownames(mat)       <- rownames(offers.data)
colnames(mat)       <- rownames(offers.data)
graph.wine          <- graph.adjacency(mat,mode = c("undirected"))
```

```
par(mfrow=c(1,1))
set.seed(12)
plot.igraph(graph.wine,
            vertex.label.cex=0.8,vertex.label= rownames(offers.data),
            layout=layout.fruchterman.reingold(graph.wine, niter=10000))
```



The above visualization is just the starting point. The author uses a method called modularity maximization for identifying communities. There is a function in the `igraph` package. The good thing about the book, as I keep repeating is, that you can see what exactly goes on behind the `optimal.community` function in excel. The fact that the algo uses mixed integer programming and that is uses a certain metric as objective function is all easy to see in an excel. Once you see that the workings in an excel, you will not hesitate to use a third party function that does the job.

```
set.seed(1)
out        <- optimal.community(graph.wine, weights = NULL)
sizes(out)

## Community sizes
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 23 20 15 12  8 14  1  1  1  1  1  1  1  1
```

```
mem                      <- (membership(out))
communities              <- data.frame(name = attr(mem,"names"), community = mem)
deals.by.community       <- merge(offers, communities, all.x= T)
temp                     <- cast(deals.by.community, offer~community, sum)
temp                     <- cbind(deals,temp)
```

Sum of orders by community

```
size.c <- colSums(temp[,8:21])
size.c

##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 67 76 37 49 16 53  4  4  2  4  5  2  4  1
```

Community coded 2 with size 76: the high volume community

```
temp[order(temp$"2",decreasing = TRUE),1:6][1:10,]

##     Campaign            Variety Minimum Discount       Origin PastPeak
## 9      April         Chardonnay     144       57        Chile    FALSE
## 14      June             Merlot      72       64        Chile    FALSE
## 31  December          Champagne      72       89       France    FALSE
## 22    August          Champagne      72       63       France    FALSE
## 6      March           Prosecco     144       86        Chile    FALSE
## 15      June Cabernet Sauvignon     144       19        Italy    FALSE
## 1    January             Malbec      72       56       France    FALSE
## 5   February Cabernet Sauvignon     144       44  New Zealand     TRUE
## 23 September         Chardonnay     144       39 South Africa    FALSE
## 4   February          Champagne      72       48       France     TRUE
```

Community coded 1 with size 67: the small timer community

```
temp[order(temp$"1",decreasing = TRUE),1:6][1:10,]

##    Campaign      Variety Minimum Discount       Origin PastPeak
## 30 December       Malbec       6       54       France    FALSE
## 7     March     Prosecco       6       40    Australia     TRUE
## 29 November Pinot Grigio       6       87       France    FALSE
```

```
## 18      July     Espumante        6        50        Oregon     FALSE
## 8      March     Espumante        6        45  South Africa     FALSE
## 13       May        Merlot        6        43         Chile     FALSE
## 12       May      Prosecco       72        83     Australia     FALSE
## 1    January        Malbec       72        56        France     FALSE
## 2    January   Pinot Noir       72        17        France     FALSE
## 3   February     Espumante      144        32        Oregon      TRUE
```

Community coded 6 with size 53 : the France community

```
temp[order(temp$"6",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign            Variety Minimum Discount    Origin PastPeak
## 22   August          Champagne      72       63    France    FALSE
## 11      May          Champagne      72       85    France    FALSE
## 1   January             Malbec      72       56    France    FALSE
## 2   January         Pinot Noir      72       17    France    FALSE
## 28 November Cabernet Sauvignon      12       56    France     TRUE
## 30 December             Malbec       6       54    France    FALSE
## 12      May           Prosecco      72       83 Australia    FALSE
## 25  October Cabernet Sauvignon      72       59    Oregon     TRUE
## 31 December          Champagne      72       89    France    FALSE
## 4  February          Champagne      72       48    France     TRUE
```

Community coded 4 with size 49 : Champagne community

```
temp[order(temp$"4",decreasing = TRUE),1:6][1:10,]
```

```
##    Campaign   Variety Minimum Discount       Origin PastPeak
## 22   August Champagne      72       63       France    FALSE
## 6     March  Prosecco     144       86        Chile    FALSE
## 3  February Espumante     144       32       Oregon     TRUE
## 7     March  Prosecco       6       40    Australia     TRUE
## 19     July Champagne      12       66      Germany    FALSE
## 27  October Champagne      72       88  New Zealand    FALSE
## 4  February Champagne      72       48       France     TRUE
## 31 December Champagne      72       89       France    FALSE
## 8     March Espumante       6       45 South Africa    FALSE
## 10    April  Prosecco      72       52   California    FALSE
```

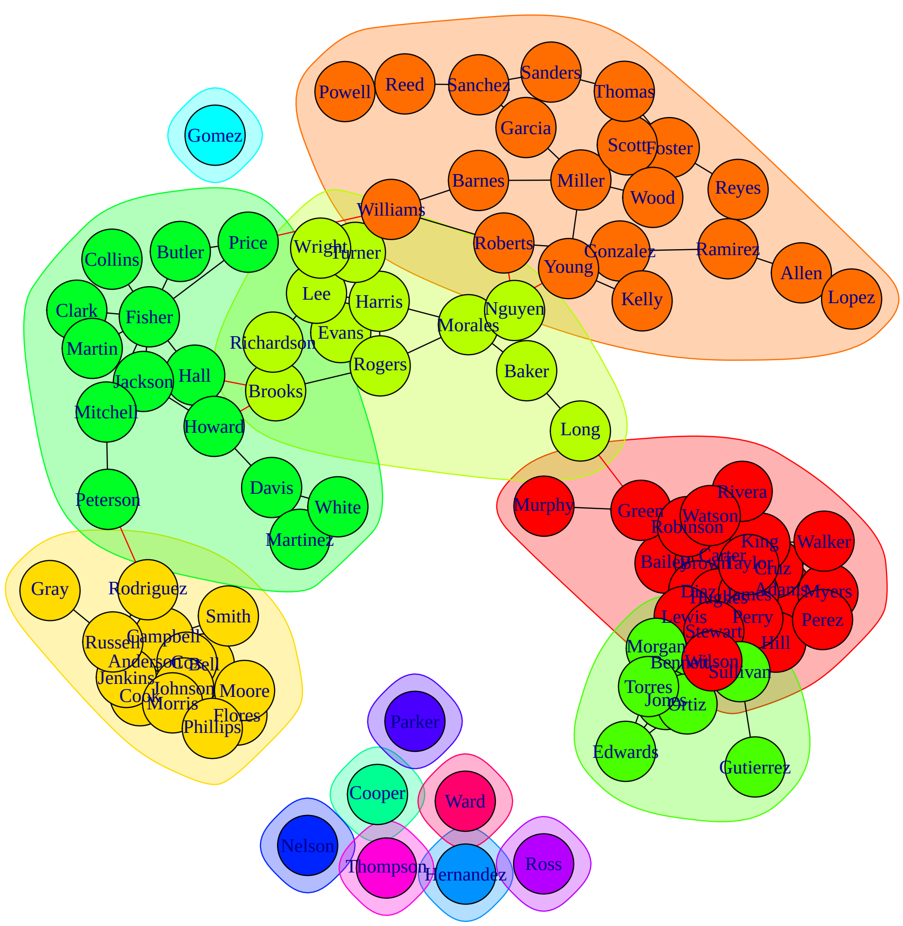
Community coded 3 with size 37 ,the Pinot Noir community

```
temp[order(temp$"3",decreasing = TRUE),1:6][1:10,]
```

```
##     Campaign            Variety Minimum Discount      Origin PastPeak
## 24 September         Pinot Noir       6       34       Italy    FALSE
```

```
## 26   October          Pinot Noir      144        83   Australia    FALSE
## 17      July          Pinot Noir       12        47     Germany    FALSE
## 2    January          Pinot Noir       72        17      France    FALSE
## 12       May            Prosecco       72        83   Australia    FALSE
## 16      June              Merlot       72        88  California    FALSE
## 1    January              Malbec       72        56      France    FALSE
## 3   February           Espumante      144        32      Oregon     TRUE
## 4   February           Champagne       72        48      France     TRUE
## 5   February Cabernet Sauvignon      144        44 New Zealand     TRUE
```

```r
plot(out,graph.wine)
```



This nice visualization of the communities is the highlight of using Network analysis. Bootstrapped classification methods do not give this ease of interpretation.

6 The Granddaddy of Supervised Artificial Intelligence - Regression

The chapter talks about the most widespread tool of any statistician - Regression. The dataset used to illustrate the concepts is a set of customer records and their purchases. Regression is a supervised learning algorithm and in this case, the criterion or the outcome variable is a dichotomous variable, i.e. whether the customer is pregnant or not. Since the modeling is done via Excel, the author manually shows the way to create dummy variables, use solver to compute the coefficients of the regression model. In the first go, the regression variable is fit with outcome variable taking a numerical value of 0 or 1. Obviously this is not a good idea when a linear regression model assumes the outcome to be a normal random variable and not a variable whose support is between 0 and 1. One needs to set up manually the RSS function to minimize it. Using solver one can get the coefficients, but one needs to do a manual set up in Excel to compute the following :

- Total sum of squares

$$\text{TSS} = \sum_i (y_i - \overline{y})^2$$

- Residual Sum of Squares

$$\text{RSS} = \sum_i (y_i - \hat{y}_i)^2$$

- R^2 test

$$R^2 = 1 - \frac{RSS}{TSS}$$

- F test (Try deriving the following expression from scratch)

$$\text{Fstat} = \frac{\frac{TSS-RSS}{(n-1)-(n-p)}}{\frac{RSS}{n-p}}$$

- Covariance matrix of the coefficients

$$\text{Cov}_\beta = \hat{\sigma}^2 (X'X)^{-1}$$

- t stat

$$t_j = \frac{\beta_j}{\hat{\sigma}\sqrt{(X'X)^{-1}_{jj}}}$$

All the hardwork that one needs to do in excel can be done in just a few lines of `R` code.

```
input.file             <- "data/ch06/Pregnancy.csv"
data                   <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)

test.file              <- "data/ch06/Pregnancy_Test.csv"
test                   <- read.csv(test.file,stringsAsFactors = TRUE,header = TRUE)

fit1                  <- lm(PREGNANT~., data)
summary(fit1)

##
## Call:
## lm(formula = PREGNANT ~ ., data = data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9814 -0.3400 -0.0301  0.3008  0.9758
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.4441     0.0260   17.10  < 2e-16 ***
## Implied.GenderM           -0.0715     0.0251   -2.85  0.00451 **
## Implied.GenderU            0.0268     0.0403    0.67  0.50615
## Home.Apt..PO.BoxH         -0.0146     0.0250   -0.59  0.55856
## Home.Apt..PO.BoxP          0.0133     0.0433    0.31  0.75884
## Pregnancy.Test             0.2164     0.0465    4.65  3.7e-06 ***
## Birth.Control             -0.2741     0.0348   -7.87  9.1e-15 ***
## Feminine.Hygiene          -0.2381     0.0343   -6.94  7.2e-12 ***
## Folic.Acid                 0.3456     0.0392    8.83  < 2e-16 ***
## Prenatal.Vitamins          0.2941     0.0360    8.16  1.0e-15 ***
## Prenatal.Yoga              0.3253     0.0893    3.64  0.00028 ***
## Body.Pillow                0.1936     0.0894    2.17  0.03057 *
## Ginger.Ale                 0.2299     0.0471    4.88  1.2e-06 ***
## Sea.Bands                  0.1458     0.0698    2.09  0.03706 *
## Stopped.buying.ciggies     0.1605     0.0417    3.85  0.00013 ***
## Cigarettes                -0.1591     0.0404   -3.94  8.7e-05 ***
## Smoking.Cessation          0.1647     0.0516    3.19  0.00146 **
## Stopped.buying.wine        0.1878     0.0359    5.23  2.1e-07 ***
## Wine                      -0.2075     0.0366   -5.66  2.0e-08 ***
## Maternity.Clothes          0.2399     0.0357    6.72  3.2e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.372 on 980 degrees of freedom
## Multiple R-squared:  0.458,Adjusted R-squared:  0.447
## F-statistic: 43.6 on 19 and 980 DF,  p-value: <2e-16
```

To draw the ROC curves for above model, one can use `ROCR` package

```
test.pred                  <- predict(fit1,test)
pred.lm                    <- prediction(test.pred,test$PREGNANT)
perf.lm                    <- performance(pred.lm,"tpr","fpr")
```

```
plot(perf.lm,xlim=c(0,1),ylim=c(0,1))
abline(h=seq(0,1,0.1), col="grey",lty="dashed")
```

The above ROC curve under a good model should hug the top left corner. So, there is a chance to improve this model. The fact that the predicted values of the model does not stay in $[0, 1]$ means that the model needs

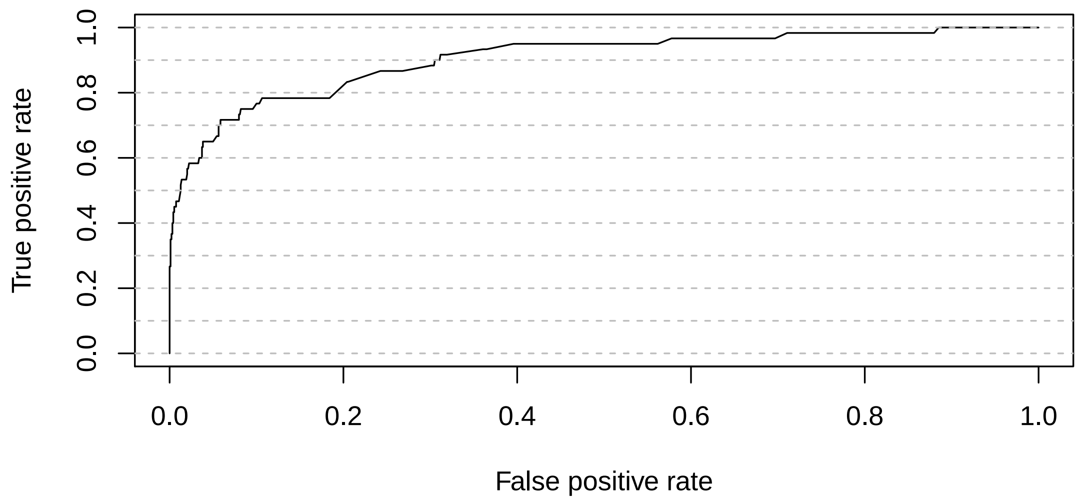


Figure 6.1: ROC curve

to be improved. The first tweak is to minimize a link function of the outcome variable than the outcome variable itself. The chapter shows the incorrect way of handling this situation and then shows the right way of estimating via likelihood procedure.Why does the MLE method work and the previous method fail is not explained, but any data analyst newbie will realize that sum of squares minimization in general is driven by a probability model and hence one needs to maximize log likelihood rather than any arbitrary function.

```
fit                  <- glm(PREGNANT~., data= data, family=binomial())
summary(fit)

##
## Call:
## glm(formula = PREGNANT ~ ., family = binomial(), data = data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.201  -0.557  -0.025   0.513   2.866
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.34360    0.18075   -1.90  0.05731 .
## Implied.GenderM          -0.45388    0.19757   -2.30  0.02160 *
## Implied.GenderU           0.14194    0.30759    0.46  0.64447
## Home.Apt..PO.BoxH        -0.17293    0.19459   -0.89  0.37418
## Home.Apt..PO.BoxP        -0.00281    0.33643   -0.01  0.99333
## Pregnancy.Test            2.37055    0.52178    4.54  5.5e-06 ***
## Birth.Control            -2.30027    0.36527   -6.30  3.0e-10 ***
```

```
## Feminine.Hygiene       -2.02856    0.34240   -5.92  3.1e-09 ***
## Folic.Acid              4.07767    0.76189    5.35  8.7e-08 ***
## Prenatal.Vitamins       2.47947    0.36906    6.72  1.8e-11 ***
## Prenatal.Yoga           2.92297    1.14699    2.55  0.01082 *
## Body.Pillow             1.26104    0.86062    1.47  0.14285
## Ginger.Ale              1.93850    0.42673    4.54  5.6e-06 ***
## Sea.Bands               1.10753    0.67343    1.64  0.10005
## Stopped.buying.ciggies  1.30222    0.34235    3.80  0.00014 ***
## Cigarettes             -1.44302    0.37012   -3.90  9.7e-05 ***
## Smoking.Cessation       1.79078    0.51261    3.49  0.00048 ***
## Stopped.buying.wine     1.38389    0.30588    4.52  6.1e-06 ***
## Wine                   -1.56554    0.34891   -4.49  7.2e-06 ***
## Maternity.Clothes       2.07820    0.32943    6.31  2.8e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.29  on 999  degrees of freedom
## Residual deviance:  744.11  on 980  degrees of freedom
## AIC: 784.1
##
## Number of Fisher Scoring iterations: 7
```

For testing the performance of the logistic regression, the following code can be used and ROC curves can be drawn to get a comparitive performance

```
test.pred                   <- predict(fit,newdata = test)
pred.logr                   <- prediction(ilogit(test.pred),test$PREGNANT)
perf.logr                   <- performance(pred.logr,"tpr","fpr")
```

```
plot(perf.lm,xlim=c(0,1),ylim=c(0,1),col="blue")
plot(perf.logr,xlim=c(0,1),ylim=c(0,1),add=T, col="red")
abline(h=seq(0,1,0.1), col="grey",lty="dashed")
legend("center",legend=c("Linear Regression","Logistic Regression"),fill=c("blue","red"),
       cex=0.7)
```

In this case, there does not seem to be much difference in using Linear Vs Logistic regression. To be technically correct, it is anyways better to use Logistic regression.

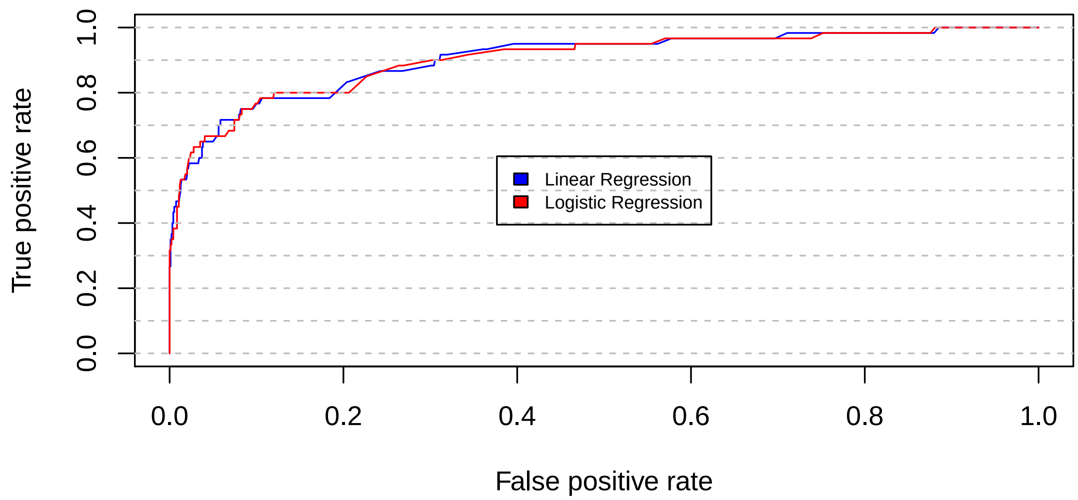


Figure 6.2: ROC curve comparison

7 Ensemble Models: A Whole Lot of Bad Pizza

The models introduced in this chapter belong to the category of "voting models". Such methods would have been unthinkable a few decades ago. With fast and cheap computing power, the methods mentioned in this chapter have become extremely popular. The chapters walks through two such methods

1. Random Forests
2. Boosting

Its really commendable that the author has taken pains to do everything in excel to show how the method really work. I came across these methods in the statistical learning books by Trevor Hastie and Rob Tibshirani. If you want the math behind it, the books by Trevor and Bob are priceless. However if you are the kind where you want to see something running and then want to explore the math, this chapter is perfect.

The dataset used in this chapter is the same as that used in the previous chapter. The author spends less time on data preparation and shows all the steps that are needed to run random bagging and boosting. In the last chapter of the book, he provides the code for random forests but not boosting.

Here is the `R` code that replicates Random Forest results from the chapter

```
input.file              <- "data/ch06/Pregnancy.csv"
data                    <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)
test.file               <- "data/ch06/Pregnancy_Test.csv"
test                    <- read.csv(test.file,stringsAsFactors = TRUE,header = TRUE)
data$PREGNANT           <- factor(data$PREGNANT)
test$PREGNANT           <- factor(test$PREGNANT)
data.rf                 <- randomForest(PREGNANT~.,data=data,importance=TRUE,
                                    replace=FALSE,maxnodes=2,mtry=17)
```

```
test.rf                 <- predict(data.rf,test,type="prob")
pred.rf                 <- prediction(test.rf[,2],test$PREGNANT)
perf.rf                 <- performance(pred.rf,"tpr","fpr")
```

```
par(mfrow=c(1,2))
varImpPlot(data.rf, type=2,cex=0.6,main="")
plot(perf.rf,xlim=c(0,1),ylim=c(0,1),lty=2)
```

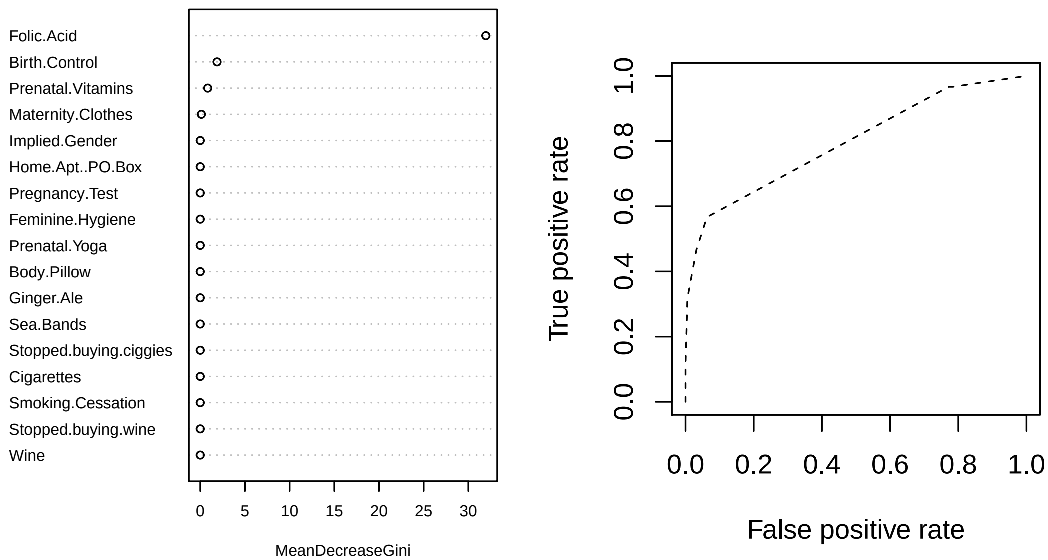


Figure 7.1: Importance Plot

Here is the R code that replicates Boosting results from the chapter

```
input.file   <- "data/ch06/Pregnancy.csv"
data         <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)
test.file    <- "data/ch06/Pregnancy_Test.csv"
test         <- read.csv(test.file,stringsAsFactors = TRUE,header = TRUE)
data.boost   <- gbm(PREGNANT~.,data=data, distribution="bernoulli")
test.boost   <- predict(data.boost,newdata = test,n.trees = 200,type="response")
pred.boost   <- prediction(test.boost,test$PREGNANT)
perf.boost   <- performance(pred.boost,"tpr","fpr")
```

Here is the list of influence variables

```
summary(data.boost,plotit=F)

##                                          var rel.inf
## Folic.Acid                        Folic.Acid  70.713
```

```
## Birth.Control                       Birth.Control  19.989
## Prenatal.Vitamins               Prenatal.Vitamins   4.938
## Maternity.Clothes               Maternity.Clothes   3.282
## Stopped.buying.wine           Stopped.buying.wine   1.078
## Implied.Gender                     Implied.Gender   0.000
## Home.Apt..PO.Box                 Home.Apt..PO.Box   0.000
## Pregnancy.Test                     Pregnancy.Test   0.000
## Feminine.Hygiene                 Feminine.Hygiene   0.000
## Prenatal.Yoga                       Prenatal.Yoga   0.000
## Body.Pillow                           Body.Pillow   0.000
## Ginger.Ale                             Ginger.Ale   0.000
## Sea.Bands                               Sea.Bands   0.000
## Stopped.buying.ciggies Stopped.buying.ciggies   0.000
## Cigarettes                             Cigarettes   0.000
## Smoking.Cessation               Smoking.Cessation   0.000
## Wine                                         Wine   0.000
```

Here is the comparison of ROC curves

```
plot(perf.rf,xlim=c(0,1),ylim=c(0,1),col="blue")
plot(perf.boost,xlim=c(0,1),ylim=c(0,1),add=T, col="red")
abline(h=seq(0,1,0.1), col="grey",lty="dashed")
legend("center",legend=c("Random Forest","Boosting"),fill=c("blue","red"), cex=0.7)
```

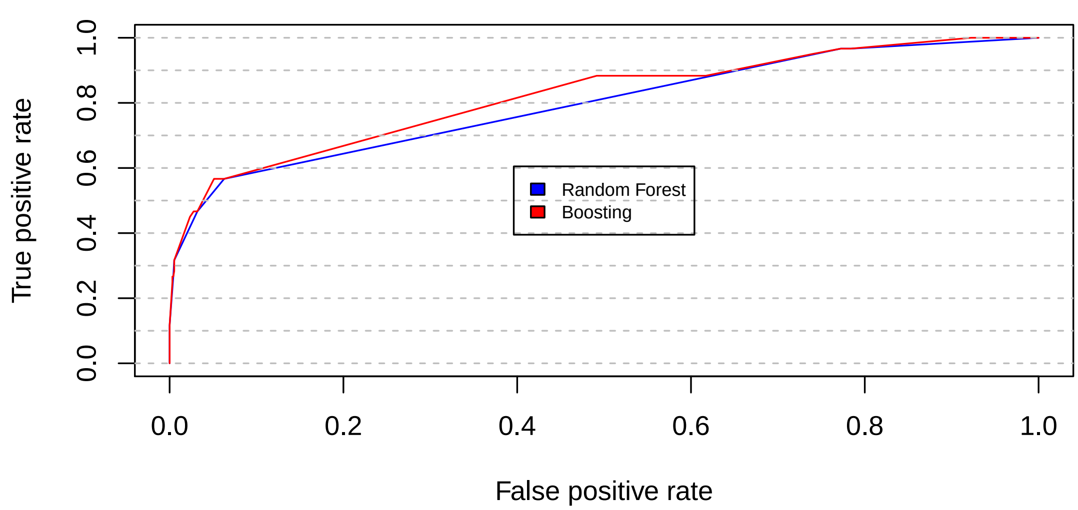


Figure 7.2: ROC curve

8 Forecasting: Breathe Easy; You Can't Win

This chapter starts off by introducing simple exponential smoothing that is basically a changing level model.Here is the R code to replicate the results from the book.

```
input.file       <- "data/ch10/SwordDemand.csv"
data             <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)
data.ts          <- ts(data, start=c(2010),frequency=12)
fit              <- HoltWinters(data.ts, beta=FALSE,
                                        gamma=FALSE, l.start = 163)
```

The smoothing parameter α and RMSE are

```
fit$alpha

## [1] 0.7297

sqrt(fit$SSE/35)

## [1] 20.39
```

```
par(mfrow=c(1,1))
plot.ts(data.ts , xaxt="n", lwd =2 , col = "grey",xlab="")
time    <- seq(as.Date("2010/1/1"),length.out = 36, by="1 month")
time    <- format(time, "%Y-%m")
axis(1, labels=time, at=time(data.ts))
points(fitted(fit), type="l",lty="dashed", col = "blue")
legend("topleft", legend=c("original time series","smoothed"),
       col  = c("grey","blue"), lty = c(1,2), lwd = c(2,1), cex = 0.8)
```

```
par(mfrow=c(1,1))
plot.ts(data.ts , xaxt="n", lwd =2 , col = "grey",
        xlim=c(2010,2014),xlab="")
timeidx <-seq( 2010, 2014,length.out = 48)
time    <- seq(as.Date("2010/1/1"),length.out = 48, by="1 month")
time    <- format(time, "%Y-%m")
axis(1, labels=time, at=timeidx)
points(fitted(fit), type="l",lty="dashed", col = "blue")
points(predict(fit, n.ahead=12), type="l", lty="dashed", col = "red", lwd  = 1)
legend("topleft", legend=c("original time series","smoothed","forecast"),
       col  = c("grey","blue","red"), lty = c(1,2,2), lwd = c(2,1), cex = 0.8)
```

Subsequently, a trend corrected exponential smoothing is done

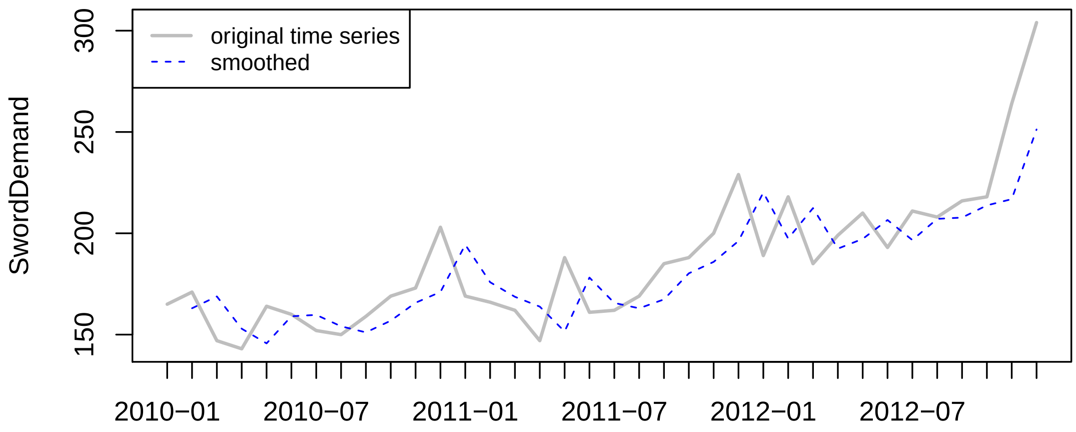


Figure 8.1: Sword Demand

```
fit   <- HoltWinters(data.ts, gamma=FALSE,l.start = 155.88,b.start = 0.8369 )
```

The smoothing parameter α , β and RMSE are

```
fit$alpha

## alpha
## 0.666

fit$beta

##    beta
## 0.05766

sqrt(fit$SSE/34)

## [1] 19.92
```

```
par(mfrow=c(1,1))
plot.ts(data.ts , xaxt="n", lwd =2 , col = "grey",
        xlim=c(2010,2014),ylim=c(100,400),xlab="")
timeidx <-seq( 2010, 2014,length.out = 48)
time    <- seq(as.Date("2010/1/1"),length.out = 48, by="1 month")
time    <- format(time, "%Y-%m")
axis(1, labels=time, at=timeidx)
points(fitted(fit), type="l",lty="dashed", col = "blue")
points(predict(fit, n.ahead=12), type="l", lty="dashed", col = "red", lwd  = 1)
```

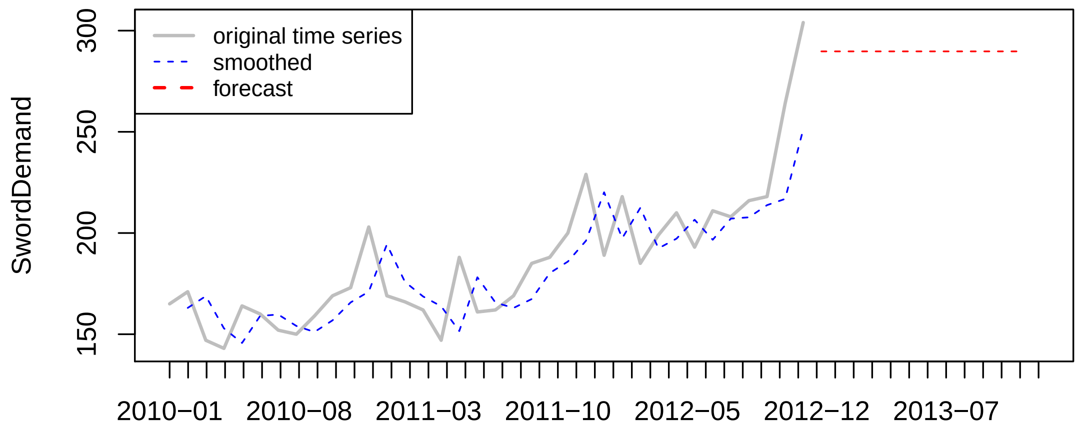


Figure 8.2: Sword Demand Prediction

```
legend("topleft", legend=c("original time series","smoothed","forecast"),
       col  = c("grey","blue","red"), lty = c(1,2,2), lwd = c(2,1), cex = 0.8)
```

One check residuals of the model using the `acf` plot

```
acf(data.ts[4:36]- c(fitted(fit)[1:33,1]) , main = "", xlim=c(1,15))
```

Clearly there is a seasonality in the data that needs to be modeled. The chapter uses Multiplicative Holt-Winters Exponential smoothing method

```
#Starting values as given in the book
s.start <- c(0.988233399,1.039459514,0.932933292,0.912597756,
             1.043010605,0.906442452,0.920837589,0.926620944,
             0.988490753,1.016201453,1.048052656,1.204004908)

fit   <- HoltWinters(data.ts, seasonal="multiplicative",
                     l.start = 144.42,b.start = 2.2095 ,s.start= s.start)
```

The smoothing parameter α , β and RMSE are

```
fit$alpha

##  alpha
## 0.2316

fit$beta
```

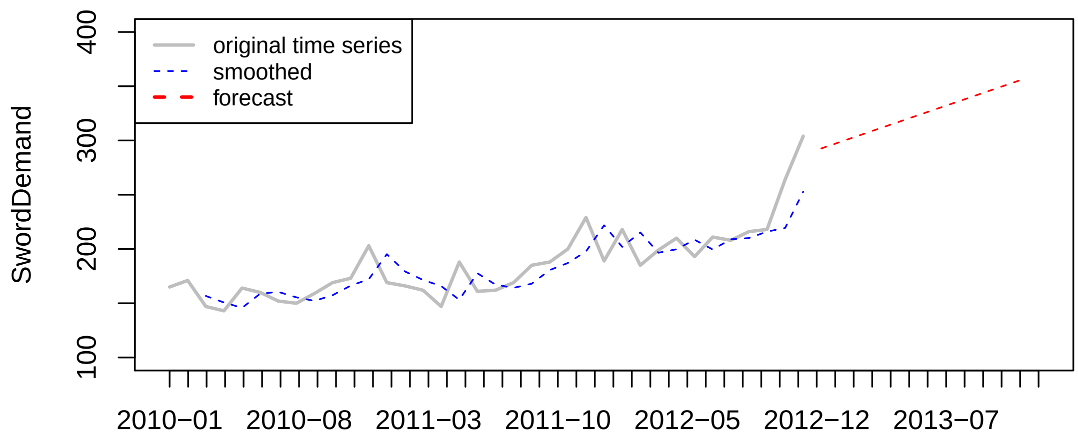


Figure 8.3: Sword Demand Prediction - Trend corrected smoothing

```
##   beta
## 0.1148

fit$gamma

## gamma
##     0

sqrt(fit$SSE/33)

## [1] 9.282
```

Check residuals of the model using the `acf` plot

```
acf(data.ts[14:36]- c(fitted(fit)[1:23,1]) , main = "", xlim=c(1,15))
```

```
par(mfrow=c(1,1))
plot.ts(data.ts , xaxt="n", lwd =2 , col = "grey",
        xlim=c(2010,2014),ylim=c(100,400),xlab="")
timeidx <-seq( 2010, 2014,length.out = 48)
time    <- seq(as.Date("2010/1/1"),length.out = 48, by="1 month")
time    <- format(time, "%Y-%m")
axis(1, labels=time, at=timeidx)
points(fitted(fit), type="l",lty="dashed", col = "blue")
points(predict(fit, n.ahead=12), type="l", lty="dashed", col = "red", lwd  = 1)
legend("topleft", legend=c("original time series","smoothed","forecast"),
       col  = c("grey","blue","red"), lty = c(1,2,2), lwd = c(2,1,1), cex = 0.8)
```

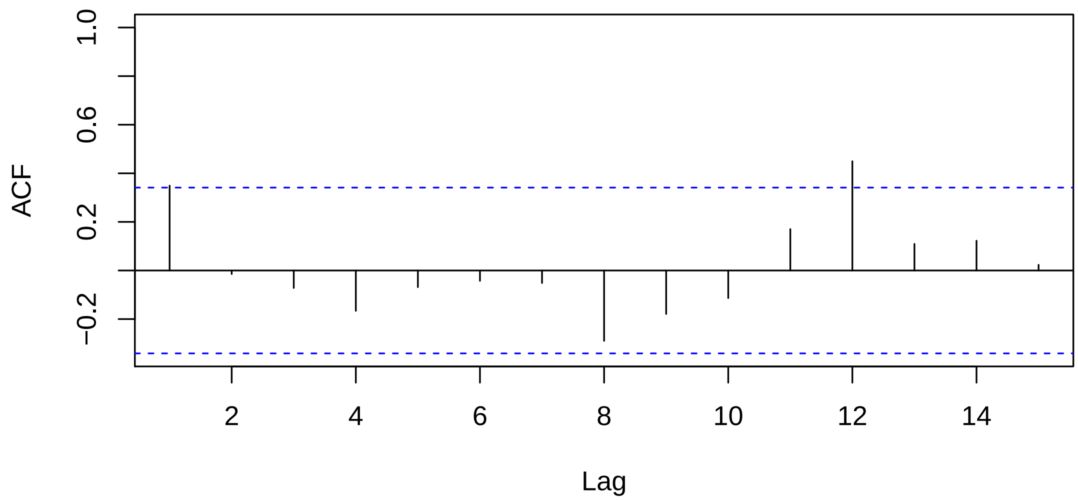


Figure 8.4: Checking residuals

The last part of the chapter puts confidence intervals around the prediction.

```
pred <- predict(fit, n.ahead=12, prediction.interval=T, level =0.95)
```

```
par(mfrow=c(1,1))
plot.ts(data.ts , xaxt="n", lwd =2 , col = "grey",
        xlim=c(2010,2014),ylim=c(100,400),xlab="")
timeidx <-seq( 2010, 2014,length.out = 48)
time    <- seq(as.Date("2010/1/1"),length.out = 48, by="1 month")
time    <- format(time, "%Y-%m")
axis(1, labels=time, at=timeidx)
points(fitted(fit), type="l",lty="dashed", col = "blue")
points(pred[,1], type="l", col = "red", lwd  = 2)
points(pred[,2], type="l", lty="dashed", col = "red", lwd  = 1)
points(pred[,3], type="l", lty="dashed", col = "red", lwd  = 1)
legend("topleft",
      legend=c("original time series","smoothed","forecast",
               "forecast-upper","forecast-lower"),
       col  = c("grey","blue","red","red","red"),
       lty = c(1,2,1,2,2), lwd = c(2,1,2,1,1), cex = 0.8)
```

The above analysis is also done in chapter 10 of this book using `forecast` package

```
sword.forecast <- forecast(data.ts)
results <- cbind(sword.forecast$mean,sword.forecast$lower[,2],sword.forecast$upper[,2])
colnames(results) <- c("mu","lower","upper")
```

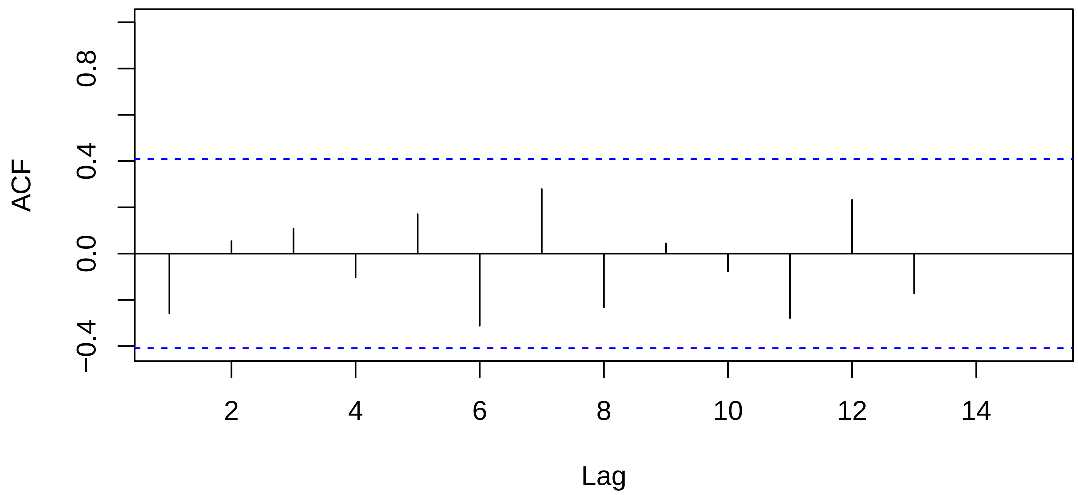


Figure 8.5: Checking residuals

```
sword.forecast$method

## [1] "ETS(M,M,M)"
```

```
plot(sword.forecast)
```

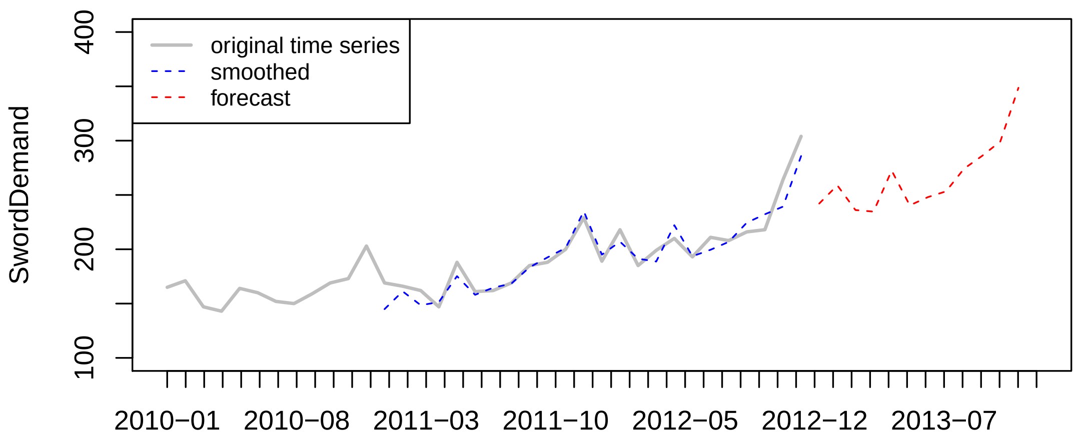


Figure 8.6: Sword Demand Prediction - Multiplicative HoltWinters smoothing

9 Outlier Detection: Just Because They're Odd Doesn't Mean They're Unimportant

The chapter starts by discussing Tukey fences as a visualizing tool to spot outliers. They come in two varieties. Innerfences are $1.5\,IQR$ and Outer fences are $3\,IQR$ on either side of the median

```
input.file  <- "data/ch10/PregnancyDuration.csv"
data        <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)
summary(data)

##  GestationDays
##  Min.   :240
##  1st Qu.:260
##  Median :267
##  Mean   :267
##  3rd Qu.:272
##  Max.   :349
```

```
IQR(data[,1])

## [1] 12

quantile(data[,1],prob=0.75)- quantile(data[,1],prob=0.25)

## 75%
##  12
```

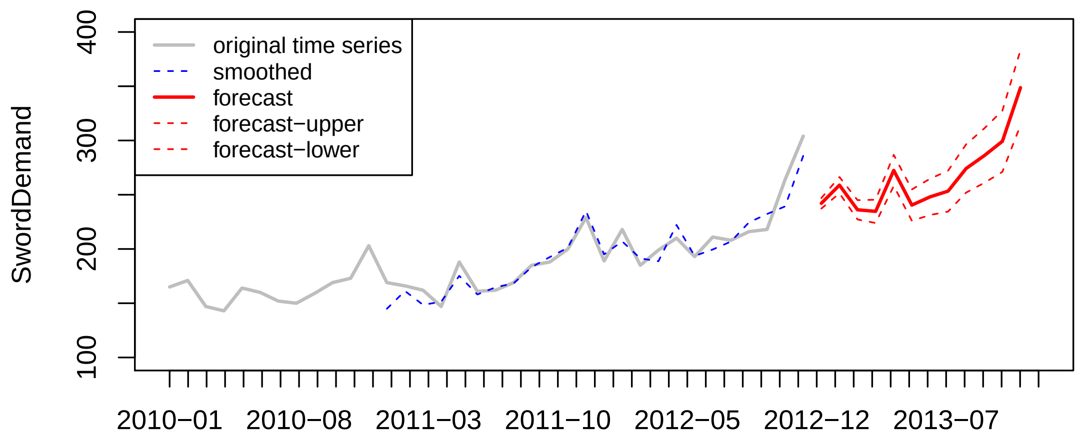


Figure 8.7: Sword Demand Prediction - Confidence bands

```
par(mfrow=c(1,2))
boxplot(data[,1], main="Inner fences",cex.main = 0.8)
boxplot(data[,1], range =3, main  = "Outer fences",cex.main =0.8)
```

One can directly see the stats with out plotting

```
stats  <- c(boxplot(data[,1], plot=F)$stats)
labels <- c("lower inner fence" ,"25th percentile" ,"median","75th percentile",
            "upper inner fence")
res    <- data.frame(labels = labels,value =stats)
```

```
stats  <- c(boxplot(data[,1],range = 3, plot=F)$stats)
labels <- c("lower outer fence" ,"25th percentile" ,"median","75th percentile",
            "upper outer fence")
res    <- cbind(labels,stats)
colnames(res) <- c("labels","value")
```

The chapter subsequently discussed three methods to quantify the outliers.

Method 1 : Indegree

```
input.file      <- "data/ch10/CallCenter.csv"
data            <- read.csv(input.file,stringsAsFactors = TRUE,header = TRUE)
data.scaled     <- scale(data[,2:11])
distance        <- as.matrix(dist(data.scaled))
```

mu	lower	upper
242.64	223.11	260.94
256.24	236.05	276.48
235.20	215.63	255.07
233.61	212.15	255.82
271.24	244.07	300.13
253.09	223.96	282.91
259.96	226.11	296.84
257.66	220.13	299.94
282.44	235.95	336.20
296.10	240.39	362.72
325.04	258.75	406.33
374.54	289.27	477.64
313.38	236.83	414.19
330.95	242.13	451.16
303.78	216.45	425.56
301.73	207.30	439.97
350.33	232.55	522.92
326.88	208.83	508.46
335.76	209.70	538.28
332.79	199.98	553.81
364.79	210.30	630.72
382.44	214.52	686.55
419.82	225.40	783.55
483.74	249.18	940.29

Table 1: Mean and confidence intervals for the prediction

labels	value
lower inner fence	242.00
25th percentile	260.00
median	267.00
75th percentile	272.00
upper inner fence	290.00

Table 2: Tukey Inner fences

```
ranks           <- t(apply(distance, 1, rank)-1)
top5s           <- apply(ranks, 2, function(z) sum(z <=5))-1
top10s          <- apply(ranks, 2, function(z) sum(z <=10))-1
top20s          <- apply(ranks, 2, function(z) sum(z <=20))-1
temp            <- cbind(data, top5s,top10s,top20s)
temp            <- temp[order(temp[,12],temp[,13],temp[,14]),]
```

Method 2 : k-distance

```
ranks           <- t(apply(distance, 1, rank)-1)
ranks[ranks!=5]  <- 0
ranks[ranks==5]  <- 1
temp             <- rowSums(ranks*distance)
temp            <- cbind(data, temp)
temp             <- temp[order(temp[,12],decreasing=T),]
```

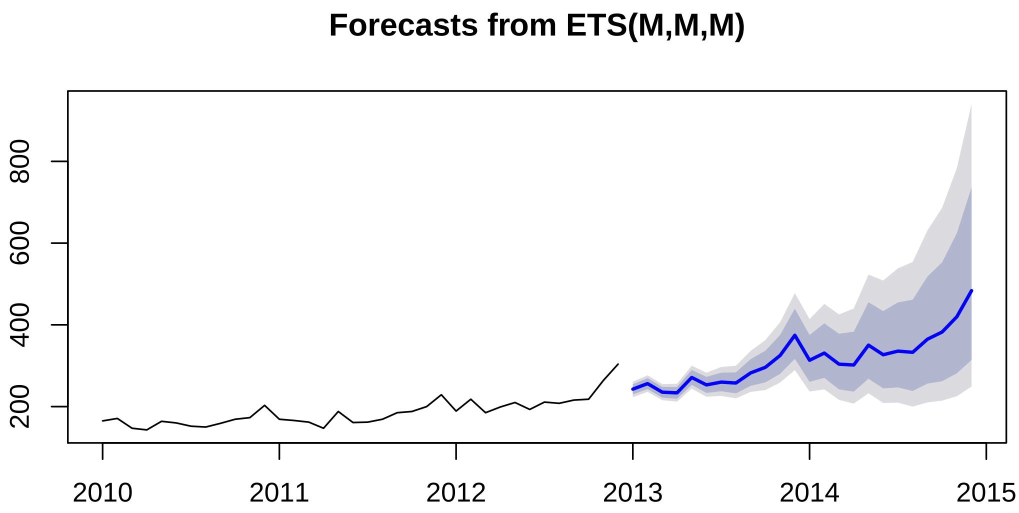


Figure 8.8: Sword Demand Prediction - Confidence bands

labels	value
lower outer fence	240
25th percentile	260
median	267
75th percentile	272
upper outer fence	303

Table 3: Tukey Outer fences

Method 3 : Local Outlier Factors

```
callcenter.lof <- lofactor(data.scaled, 5)
local.outliers <- data[which(callcenter.lof > 1.5),]
```

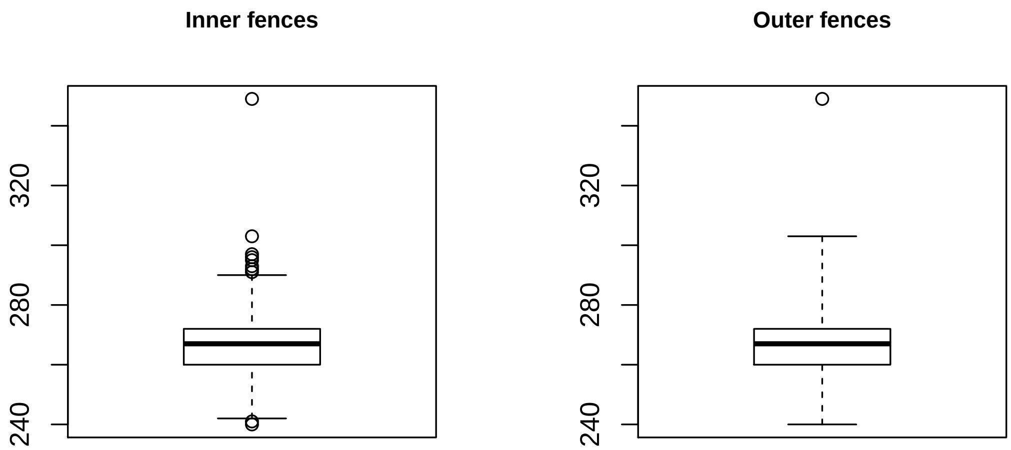


Figure 9.1: Tukey Fences

ID	AvgTix	Rating	Tardies	Graveyards	Weekends	SickDays
137155	165.30	4.49	1	3	2	1
143406	145.00	2.33	3	1	0	6

Table 4: Outliers based on Indegree

10 Moving from Spreadsheets into R

This chapter introduces some basic commands in `R` and then works out the Kmeans model, the regression model, the random forests model, forecasting model and outlier detection methods in R. In each of the case, the author provides abundant commentary for each of the packages and commands being used. The packages used to replicate the excel results from the previous chapter include

- `skmeans`
- `randomForest`
- `ROCR`
- `forecast`
- `DMwR`

11 Conclusion

The book concludes with some soft skills that a data scientist should have for him / her to be effective in an organization.

ID	PercSickOnFri	EmployeeDevHrs	ShiftSwapsReq	ShiftSwapsOffered
137155	0.00	30	1	7
143406	0.83	30	4	0

Table 5: Outliers based on Indegree

ID	AvgTix	Rating	Tardies	Graveyards	Weekends	SickDays
143406	145.00	2.33	3	1	0	6
137155	165.30	4.49	1	3	2	1

Table 6: Outliers based on k distance

ID	PercSickOnFri	EmployeeDevHrs	ShiftSwapsReq	ShiftSwapsOffered
143406	0.83	30	4	0
137155	0.00	30	1	7

Table 7: Outliers based on k distance

ID	AvgTix	Rating	Tardies	Graveyards	Weekends	SickDays
137155	165.30	4.49	1	3	2	1
143406	145.00	2.33	3	1	0	6

Table 8: Local Outlier Factors

ID	PercSickOnFri	EmployeeDevHrs	ShiftSwapsReq	ShiftSwapsOffered
137155	0.00	30	1	7
143406	0.83	30	4	0

Table 9: Local Outlier Factors