COMP 6115 - Assignment 1

Denecian Dennis - 620062729

Introduction

We are given a data set containing sales data. We asked to clean the data, generate and review some cluster analyses. The fist step we want to do is import the necessary libraries and the data set we will be working with.

```
suppressPackageStartupMessages(library("gdata"))
suppressPackageStartupMessages(library(dendextend))
```

```
## Warning: package 'dendextend' was built under R version 4.0.4
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.4
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.4
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(NbClust)
```

Importing the Data Set

Here we import the data set SalesData that was provided.

```
SalesData<-read.csv('SalesData.csv', header = TRUE)
```

Data Exploration I

Let us get an idea of the data by looking at the structure of the data

```
str(SalesData)
```

```
## 'data.frame':    10113 obs. of  9 variables:
##  $ Region        : chr  "Middle East and North Africa" "Central America and the Caribbean" "Sub-Saha
##  $ ItemType      : chr  "" "" "Fruits" "Personal Care" ...
##  $ Sales.Channel : chr  "Online" "Offline" "Offline" "Online" ...
##  $ Order.Priority: chr  "C" "L" "M" "M" ...
##  $ Order.Date    : chr  "10/8/2014" "2/22/2015" "12/9/2015" "9/17/2014" ...
##  $ Order.ID      : int  535113847 874708545 854349935 892836844 129280602 473105037 754046475 772153
##  $ Ship.Date     : chr  "10/23/2014" "2/27/2015" "1/18/2016" "10/12/2014" ...
##  $ Units.Sold    : int  934 4551 9986 9118 5858 1149 7964 6307 8217 2758 ...
##  $ Unit.Price    : num  152.58 437.2 9.33 81.73 668.27 ...
```

We notice a few things from the structure of the data.

1. Within the the data set there are 9 attributes and 10,113 rows of data.
2. The attribute ItemType has data points that are blank spaces
3. The attributes Order.Date and Ship Date are of type character and we would expect them both to be dates.

Let us examine a little further by looking at the the summary of each data attributes.

Region

```
unique(SalesData$Region)
```

```
##  [1] "Middle East and North Africa"      "Central America and the Caribbean"
##  [3] "Sub-Saharan Africa"                ""
##  [5] "??"                                "Europe"
##  [7] "Asia"                              "Australia and Oceania"
##  [9] "North America"                     "???"
## [11] "?"                                 "\"\""
```

Here we notice 7 distinct regions. We also notice data points that have some characters such as '??', ' ','??' etc that we will later replace with N/A since these appear to be typographical errors or invalid entries in the attribute.

ItemType

```
unique(SalesData$ItemType)
```

```
##  [1] ""                "Fruits"          "Personal Care"   "Household"
##  [5] "Clothes"         "Cosmetics"       "Snacks"          "Vegetables"
##  [9] "Office Supplies" "Beverages"       "Meat"            "Cereal"
## [13] "None"            "Baby Food"
```

For the ItemType attribute, we notice that there exist blank spaces in some of the data points and a categories that is labeled `None`. These will also be handle later.

Sales.Channel

```
unique(SalesData$Sales.Channel)
```

```
## [1] "Online"  "Offline" ""        "YES"
```

Again, here we notice blank spaces within the attribute and also a data point 'YES' that appears doesn't appear to be a valid response in this field. Let us look at it closer

```
nrow(SalesData[SalesData$Sales.Channel=='YES',])
```

```
## [1] 269
```

With 269 entries with 'YES' representing 2.66% of the data set, we can safely replace these entire with N/A as well.

Importing the Data Set II

Given that we know from above that there are some invalid data points, let us import the data set again with some constraints to insert N/A into data points where there exist known invalid entries.

```
SalesData<-read.csv('SalesData.csv', na.strings = c("","",
"??","???","?","\"\"","YES"), stringsAsFactors = TRUE)
```

Data Exploration II

Looking at the data structure again we have:

```
str(SalesData)
```

```
## 'data.frame':    10113 obs. of  9 variables:
##  $ Region        : Factor w/ 7 levels "Asia","Australia and Oceania",..: 5 3 7 7 NA NA NA NA NA NA .
##  $ ItemType      : Factor w/ 13 levels "Baby Food","Beverages",..: NA NA 6 11 7 4 5 6 12 5 ...
##  $ Sales.Channel : Factor w/ 2 levels "Offline","Online": 2 1 1 2 1 2 1 2 2 2 ...
##  $ Order.Priority: Factor w/ 4 levels "C","H","L","M": 1 3 4 4 2 1 4 1 2 2 ...
##  $ Order.Date    : Factor w/ 2688 levels "1/1/2010","1/1/2011",..: 442 983 866 2543 1039 966 1275 12
##  $ Order.ID      : int  535113847 874708545 854349935 892836844 129280602 473105037 754046475 772153
##  $ Ship.Date     : Factor w/ 2724 levels "1/1/2011","1/1/2012",..: 337 1016 73 255 1286 1022 1738 15
##  $ Units.Sold    : int  934 4551 9986 9118 5858 1149 7964 6307 8217 2758 ...
##  $ Unit.Price    : num  152.58 437.2 9.33 81.73 668.27 ...
```

Here we see that most attributes have been converted to factors excluding OrderID, Units.Sold and Unit.Price which are int, int and num respectively.

ItemType Missing Values

First we observe the summary of the data before we delete the row to ensure it was done properly

```
summary(SalesData$ItemType)
```

```
##       Baby Food       Beverages          Cereal         Clothes       Cosmetics
##             784             762             826             772             799
##          Fruits       Household            Meat            None Office Supplies
##             809             772             782             499             816
##   Personal Care          Snacks      Vegetables            NA's
##             856             833             801               2
```

Now we delete the two rows with the missing data.

```
SalesData <- SalesData[!is.na(SalesData$ItemType),]
summary(SalesData$ItemType)
```

```
##       Baby Food       Beverages          Cereal         Clothes       Cosmetics
##             784             762             826             772             799
##          Fruits       Household            Meat            None Office Supplies
##             809             772             782             499             816
##   Personal Care          Snacks      Vegetables
##             856             833             801
```

Item Type

Lets take a deeper look into the category labeled 'None

```
unique(SalesData$ItemType)
```

```
##  [1] Fruits          Personal Care   Household       Clothes
##  [5] Cosmetics       Snacks          Vegetables      Office Supplies
##  [9] Beverages       Meat            Cereal          None
```

```
## [13] Baby Food
## 13 Levels: Baby Food Beverages Cereal Clothes Cosmetics Fruits ... Vegetables
```

```
unique(SalesData$Unit.Price)
```

```
##  [1]   9.33  81.73 668.27 109.28 437.20 152.58 154.06 651.21  47.45 421.89
## [11] 205.70 255.28
```

We see here that there exist 13 `ItemType` including the category that is labeled `None` and 12 categories in the `Unit Price`. This implies that each item type has a set price and we can fill in the entries that are `None` using this information.

```
lst <- as.list(levels(SalesData$ItemType))

for (i in 1:length(lst)){
  print(paste('The unit price for',lst[i], 'is',
              mean(SalesData[SalesData$ItemType==lst[i],]$Unit.Price,
                   na.rm = TRUE)))
}
```

```
## [1] "The unit price for Baby Food is 255.28"
## [1] "The unit price for Beverages is 47.45"
## [1] "The unit price for Cereal is 205.7"
## [1] "The unit price for Clothes is 109.28"
## [1] "The unit price for Cosmetics is 437.2"
## [1] "The unit price for Fruits is 9.33"
## [1] "The unit price for Household is 668.27"
## [1] "The unit price for Meat is 421.89"
## [1] "The unit price for None is 277.732565130261"
## [1] "The unit price for Office Supplies is 651.21"
## [1] "The unit price for Personal Care is 81.73"
## [1] "The unit price for Snacks is 152.58"
## [1] "The unit price for Vegetables is 154.06"
```

With the now known unit price for each Item type was can use this information to fill in the missing item type under the category of `None`.

```
SalesData[c(SalesData$Unit.Price==255.28 & SalesData$ItemType=='None'),]$ItemType <- 'Baby Food'
```

```
SalesData[c(SalesData$Unit.Price==47.45 & SalesData$ItemType=='None'),]$ItemType <- 'Beverages'
```

```
SalesData[c(SalesData$Unit.Price==205.7 & SalesData$ItemType=='None'),]$ItemType <- 'Cereal'
```

```
SalesData[c(SalesData$Unit.Price==109.28 & SalesData$ItemType=='None'),]$ItemType <- 'Clothes'
```

```
SalesData[c(SalesData$Unit.Price==437.20 & SalesData$ItemType=='None'),]$ItemType <- 'Cosmetics'
```

```
SalesData[c(SalesData$Unit.Price==9.33 & SalesData$ItemType=='None'),]$ItemType <- 'Fruits'
```

```
SalesData[c(SalesData$Unit.Price==668.27 & SalesData$ItemType=='None'),]$ItemType <- 'Household'
```

```
SalesData[c(SalesData$Unit.Price==421.89 & SalesData$ItemType=='None'),]$ItemType <- 'Meat'
```

```
SalesData[c(SalesData$Unit.Price==651.21 & SalesData$ItemType=='None'),]$ItemType <- 'Office Supplies'
```

```
SalesData[c(SalesData$Unit.Price==81.73 & SalesData$ItemType=='None'),]$ItemType <- 'Personal Care'
```

```
SalesData[c(SalesData$Unit.Price==152.58 & SalesData$ItemType=='None'),]$ItemType <- 'Snacks'
```

```
SalesData[c(SalesData$Unit.Price==154.06 & SalesData$ItemType=='None'),]$ItemType <- 'Vegetables'
```

```
SalesData$ItemType <- droplevels(SalesData$ItemType)
levels(SalesData$ItemType)
```

```
##  [1] "Baby Food"       "Beverages"       "Cereal"          "Clothes"
##  [5] "Cosmetics"       "Fruits"          "Household"       "Meat"
##  [9] "Office Supplies" "Personal Care"   "Snacks"          "Vegetables"
```

You can see from above that we have reclassified the data in `None` and removed the N/A's that was present in the data.

Order Priority

```
str(SalesData$Order.Priority)
```

```
##  Factor w/ 4 levels "C","H","L","M": 4 4 2 1 4 1 2 2 1 4 ...
```

```
summary(SalesData$Order.Priority)
```

```
##    C    H    L    M
## 2551 2521 2592 2447
```

Here we see that the exist four levels: C-Critical, H-High, L-Low and M-Medium. We also note that these levels are not ordered. Let us go ahead and do this.

```
SalesData$Order.Priority <- ordered(SalesData$Order.Priority, levels = c('C','H','M','L'))
is.ordered(SalesData$Order.Priority)
```

```
## [1] TRUE
```

```
levels(SalesData$Order.Priority)
```

```
## [1] "C" "H" "M" "L"
```

We see here that the levels have been ordered and is now in the order C>H>M>L.

Order Date

Let us have a look at the data to get the format to convert it to native R dates.

```
head(SalesData$Order.Date, 5)
```

```
## [1] 12/9/2015 9/17/2014 2/4/2010  2/20/2013 3/31/2013
## 2688 Levels: 1/1/2010 1/1/2011 1/1/2012 1/1/2013 1/1/2014 1/1/2015 ... 9/9/2016
```

Here we can infer that the format is month-day-year from dates such as 2/20/2013 etc.

```
SalesData$Order.Date <- as.Date(SalesData$Order.Date, format = "%m/%d/%Y")
str(SalesData$Order.Date)
```

```
##  Date[1:10111], format: "2015-12-09" "2014-09-17" "2010-02-04" "2013-02-20" "2013-03-31" ...
```

We can see that the Order data has successfully been transformed to a date format.

Order.ID

We want to check on the uniqueness of the order ID

```
if (length(unique(SalesData$Order.ID)) == length(SalesData$Order.ID)){
  print ('Order ID is unique')
```

```
}else{
  print ('Order ID id not uqique')
}
```

```
## [1] "Order ID is unique"
```

Since the order ID is unique, it will not provide any useful insights and we will thus remove it from the data set.

```
SalesData$Order.ID<-NULL
colnames(SalesData)
```

```
## [1] "Region"         "ItemType"       "Sales.Channel"  "Order.Priority"
## [5] "Order.Date"     "Ship.Date"      "Units.Sold"     "Unit.Price"
```

We can see that Order.ID has been removed from the data set.

Ship Date

```
head(SalesData$Ship.Date)
```

```
## [1] 1/18/2016  10/12/2014 3/5/2010   2/28/2013  5/3/2013   4/7/2012
## 2724 Levels: 1/1/2011 1/1/2012 1/1/2013 1/1/2014 1/1/2015 1/1/2016 ... 9/9/2017
```

Looking at the date, we can infer that the format is month-day-year from dates such as 2/28/2013 etc.

```
SalesData$Ship.Date <- as.Date(SalesData$Ship.Date, format = "%m/%d/%Y")
str(SalesData$Ship.Date)
```

```
##  Date[1:10111], format: "2016-01-18" "2014-10-12" "2010-03-05" "2013-02-28" "2013-05-03" ...
```

We can see that the ship data attribute has successfully been transformed to a date format.

Units Sold

```
str(SalesData$Units.Sold)
```

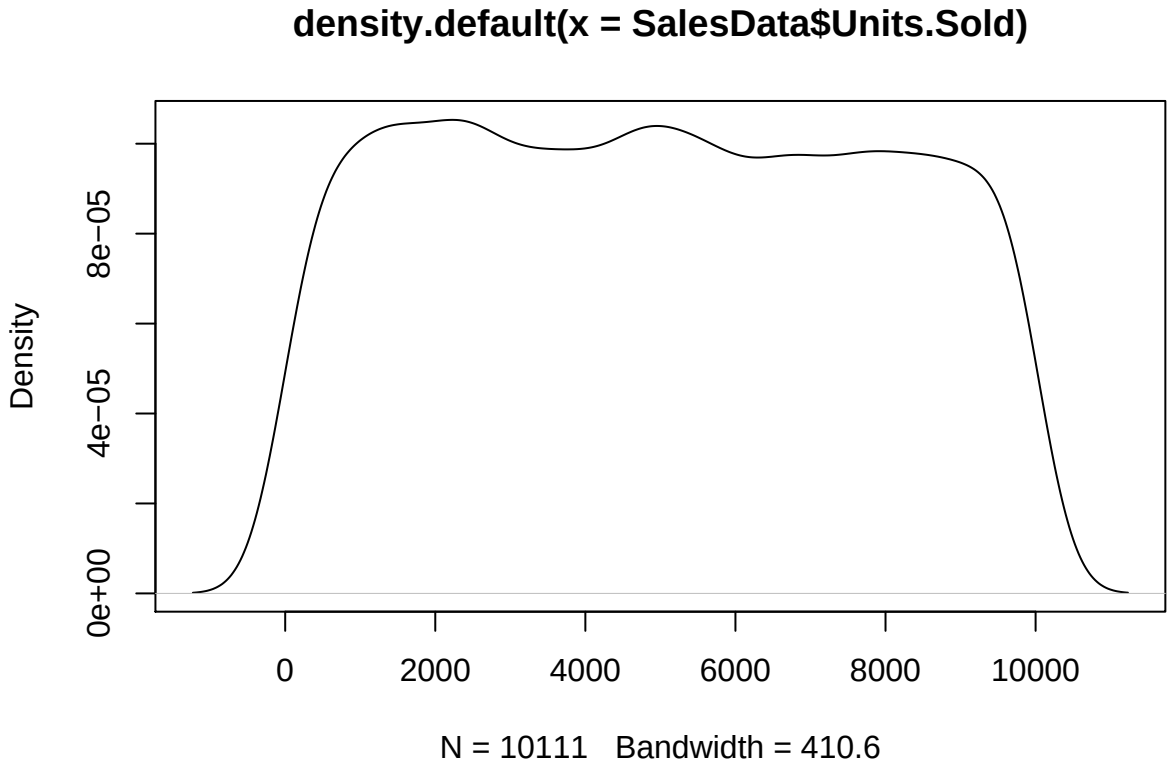
```
##  int [1:10111] 9986 9118 5858 1149 7964 6307 8217 2758 1031 1165 ...
```

```
summary(SalesData$Units.Sold)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1    2450    4920    4963    7462    9999
```

Here we see the data type as int, which is appropriate for the field type. We can also note that the distribution for Units sold appears to be fairly uniformed based on the summary statistics. Let us do a plot to graphically see the representation of the data set.

```
plot(density(SalesData$Units.Sold))
```



From the graphical representation, there also appears to be no outliers within this attribute.

Unit Price

```
str(SalesData$Unit.Price)
```

```
##  num [1:10111] 9.33 81.73 668.27 109.28 437.2 ...
```

```
summary(SalesData$Unit.Price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.33   81.73  154.06  265.42  421.89  668.27
```

We see here the data type is num, which is appropriate for the attribute and that there are no unit prince below 0.

Cleaning Missing Values

So far we have applied some cleaning techniques to the data. Let's now take a look at the N/A's.

```
apply(SalesData,2,function(k) sum(is.na(k)))
```

```
##         Region       ItemType  Sales.Channel Order.Priority     Order.Date
##           1426              0            354              0              0
##      Ship.Date     Units.Sold     Unit.Price
##              0              0              0
```

See the table below to note the action that will be taken to resolve the N/A in the three attributes: Region, ItemType and Sale.Channel.

Attribute	Action
Region	All 1426 N/A's will be replaced with 'Unknown'
Sales.Channel	All 354 N/A's will be replaced with 'Unknown'

Region Missing Values

```
levels(SalesData$Region)
```

```
## [1] "Asia"                              "Australia and Oceania"
## [3] "Central America and the Caribbean" "Europe"
## [5] "Middle East and North Africa"      "North America"
## [7] "Sub-Saharan Africa"
```

```
levels(SalesData$Region)<-c(levels(SalesData$Region), "Unknown")
SalesData$Region[is.na(SalesData$Region)]<-"Unknown"
summary(SalesData$Region)
```

```
##                              Asia             Australia and Oceania
##                              1236                               693
## Central America and the Caribbean                           Europe
##                               889                              2264
##      Middle East and North Africa                     North America
##                              1171                               174
##                Sub-Saharan Africa                           Unknown
##                              2258                              1426
```

We see here that all 1426 missing values have now been assign the level unknown.

Sales.Channel Missing Values

Here we observe the levels that exists in Sale.Channel.

```
levels(SalesData$Sales.Channel)
```

```
## [1] "Offline" "Online"
```

We know there exist 354 missing values in Sales Channel. To change their values to 'Unknown' we do the following.

```
levels(SalesData$Sales.Channel)<-c(levels(SalesData$Sales.Channel), "Unknown")
SalesData$Sales.Channel[is.na(SalesData$Sales.Channel)]<-"Unknown"
summary(SalesData$Sales.Channel)
```

```
## Offline  Online Unknown
##    4884    4873     354
```

Here we see that the 354 missing values has been changed to unknown.

Before we move on let us check the consistency of the missing values in the data set.

```
apply(SalesData,2,function(k) sum(is.na(k)))
```

```
##         Region       ItemType  Sales.Channel Order.Priority     Order.Date
##              0              0              0              0              0
##      Ship.Date     Units.Sold     Unit.Price
```

```
##              0              0              0
```

Outliers

We know that there exist no outlier in all attributes except Unit.Price. Let us check this now.

```
RBound<-SalesData[SalesData$Unit.Price>(mean(SalesData$Unit.Price)+
(2*sd(SalesData$Unit.Price))),"Unit.Price"]
LBound<-SalesData[SalesData$Unit.Price<(mean(SalesData$Unit.Price) -
(2*sd(SalesData$Unit.Price))),"Unit.Price"]

RBound
```

```
## numeric(0)
```

```
LBound
```

```
## numeric(0)
```

Here we see that there are no data points that are two standard deviations to the left and right of the mean; hence there are no outliers.

Transformation

Adding New Attributes - Wait Time

We are asked to create a new attribute that will tell the number of days between ship date and the order date. Here we will call is `Wait.Time`.

First we want to have this field calculated, see below.

```
date.diff<- difftime(SalesData$Ship.Date, SalesData$Order.Date, units = c("days"))
head(date.diff,15)
```

```
## Time differences in days
##  [1] 40 25 29  8 33 12 17  7  5 24 38  9 15 19 50
```

Let us append it to the data set

```
SalesData$Wait.Time <- date.diff
head(SalesData[,c('Ship.Date', 'Order.Date', 'Wait.Time')])
```

```
##    Ship.Date Order.Date Wait.Time
## 3 2016-01-18 2015-12-09   40 days
## 4 2014-10-12 2014-09-17   25 days
## 5 2010-03-05 2010-02-04   29 days
## 6 2013-02-28 2013-02-20    8 days
## 7 2013-05-03 2013-03-31   33 days
## 8 2012-04-07 2012-03-26   12 days
```

Here we can see the the field had been correctly calculated and appended.

Adding New Attributes - Total Cost

We will also a the attribute Total Cost the will use the data in Unit Price and Units Sold. When that attribute is successfully created, the depended attributes can be remove.

```
SalesData$Total.Cost <- SalesData$Units.Sold * SalesData$Unit.Price
head(SalesData[,c('Unit.Price','Units.Sold','Total.Cost')])
```

```
##   Unit.Price Units.Sold Total.Cost
## 3       9.33       9986   93169.38
## 4      81.73       9118  745214.14
## 5     668.27       5858 3914725.66
## 6     109.28       1149  125562.72
## 7     437.20       7964 3481860.80
## 8       9.33       6307   58844.31
```

We can note that the new attribute was successfully created. We now remove the unwanted dependent attributes.

```
SalesData$Unit.Price <- NULL
SalesData$Units.Sold <- NULL
```

Convert Factors to Numerical Representation

Here we will convert the factors: `Region`, `Itemtype`, `Sales Channel`, and `Order Priority` to numeric representation.

```
SalesData$Region <- as.numeric(SalesData$Region)
SalesData$Sales.Channel <- as.numeric(SalesData$Sales.Channel)
SalesData$Order.Priority <- as.numeric(SalesData$Order.Priority)
SalesData$ItemType <- as.numeric(SalesData$ItemType)

head(SalesData[,c('Region','ItemType','Sales.Channel','Order.Priority')])
```

```
##   Region ItemType Sales.Channel Order.Priority
## 3      7        6             1              3
## 4      7       10             2              3
## 5      8        7             1              2
## 6      8        4             2              1
## 7      8        5             1              3
## 8      8        6             2              1
```

Here we can see that the specified columns was transformed to numeric representations.

Normalization and Binnings

See the table below to reference the techniques to be employed on the data set.

Attribute	Data Preparation Technique
Total Cost	Min-Max Normalization
Wait Time	Binning

Total Cost

```
summary(SalesData$Total.Cost)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     224  273146  773498 1322383 1799962 6679359
```

Given that the values for total cost is large, we will use the min-max normalization technique on the attribute.

```
TC <-  ((SalesData[,'Total.Cost']-min(SalesData$Total.Cost))/
          (max(SalesData$Total.Cost)-min(SalesData$Total.Cost)))*(12-1)+1
```

```
summary(TC)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.449   2.274   3.177   3.964  12.000
```

```
SalesData$Total.Cost <- TC
```

We see from above that the attribute was successfully normalized using the Decimal SCaling method.

Wait Time

The transformation key below tells how the attribute will be binned, the label that will be attached and description of the data in that bin

Wait Time Range	Details	Bin Label
0 - 10 days	On Time	1
11 - 20 days	Delayed	2
>20 days	Late	3

```
summary(SalesData$Wait.Time)
```

```
##   Length    Class     Mode
##    10111 difftime  numeric
```

```
SalesData$Wait.Time<-as.numeric(SalesData$Wait.Time)
WT <- cut(SalesData$Wait.Time, c(-1,10,20,100), right = TRUE, labels = c('One Time','Delayed','Late'))

SalesData$Wait.Cat <- WT
head(SalesData[,c('Wait.Time', 'Wait.Cat')],10)
```

```
##    Wait.Time Wait.Cat
## 3         40     Late
## 4         25     Late
## 5         29     Late
## 6          8 One Time
## 7         33     Late
## 8         12  Delayed
## 9         17  Delayed
## 10         7 One Time
## 11         5 One Time
## 12        24     Late
```

Here we see that the transformation appropriately describes the data. Let us now convert these labels to numeric representation and remove the `Wait.Time`, 'Order.Date' and 'Ship.Date' attributes.

```
SalesData$Wait.Cat <- as.numeric(SalesData$Wait.Cat)
SalesData$Wait.Time <- NULL
SalesData$Order.Date <- NULL
SalesData$Ship.Date <- NULL
```

Before we move on the the cluster analysis, let us write our clean data to a csv file for ease of use.

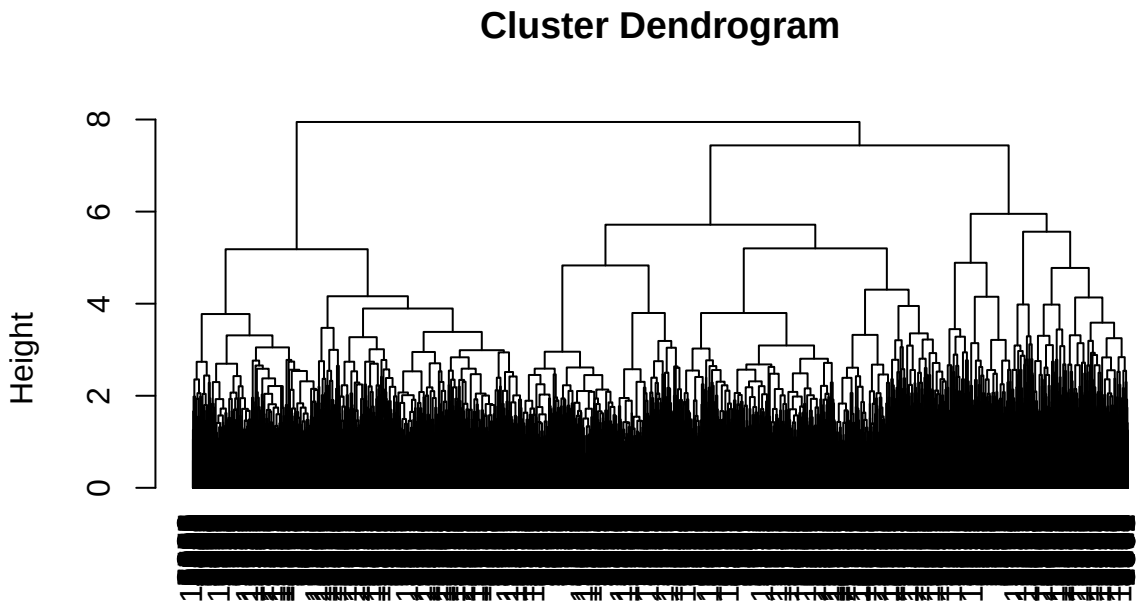
```
write.csv(SalesData, file='Clean_Sales_Data.csv', row.names = FALSE)
```

Cluster Analysis

We will employ cluster analysis to group the data that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. The K-Means and the Hierarchical Cluster analysis will be used with 4,6 and 8 clusters.

Agglomorative Cluster Analysis

```
dis.matrix <- dist(SalesData)
H.Cluster<-hclust(dis.matrix, method = 'average')
plot(H.Cluster, hang = -1)
```



Just by looking at the clusters above, we can see three cluster that seem to be grouped together. One to the left, another in the middle and the last one on the right.

```
HC.4 <- cutree(H.Cluster,4)
HC.6 <- cutree(H.Cluster,6)
HC.8 <- cutree(H.Cluster,8)
df <- data.frame(HC.4, HC.6, HC.8)
tail(df,10)
```

```
##       HC.4 HC.6 HC.8
## 10104    1    4    7
## 10105    1    4    7
## 10106    1    4    7
## 10107    1    4    5
## 10108    3    3    4
## 10109    3    3    4
```

```
## 10110    1    1    1
## 10111    4    5    6
## 10112    3    3    4
## 10113    4    5    6
```

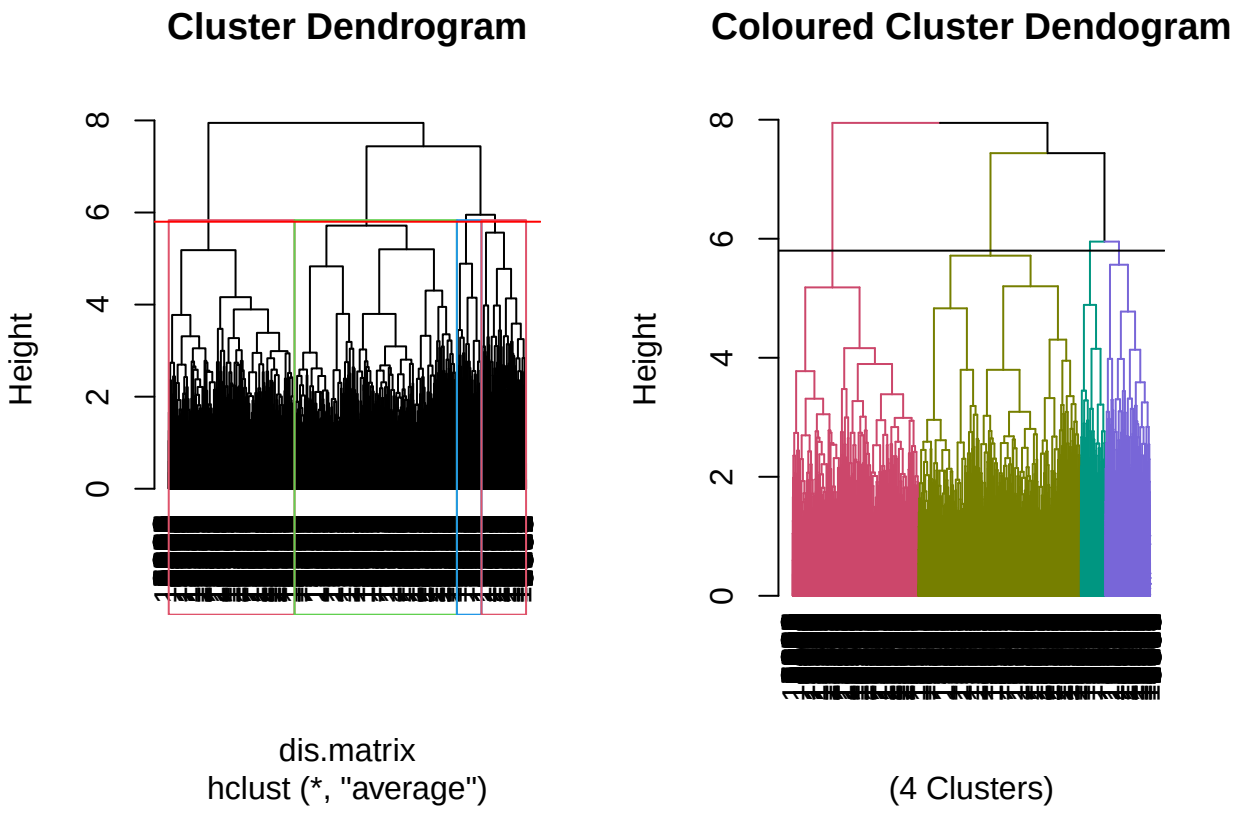
By observing the assigned cluster we can some consistency in the clustering, however, no strong inferences can be made by the data displayed above.

4 Cluster Analysis

```
par(mfrow=c(1,2))

plot(H.Cluster, hang = -1)
rect.hclust(H.Cluster, k = 4, border = 2:4)
abline(h = 5.8, col = 'red')

plot(color_branches(as.dendrogram(H.Cluster), h = 5.8),
     main = 'Coloured Cluster Dendogram',
     sub = '(4 Clusters)',
     ylab = 'Height')
abline(h = 5.8, col = 'black')
```




```
aggregate(SalesData, by=list(HC.4), FUN=mean)
```

```
##   Group.1   Region ItemType Sales.Channel Order.Priority Total.Cost Wait.Cat
## 1       1 4.818024 9.300392      1.555290       2.511319   2.108922 2.368960
## 2       2 7.352011 7.310345      1.494253       2.362069   7.448594 2.311782
```

```
## 3       3 4.820455 2.690643      1.558584       2.494521   2.286418 2.375386
## 4       4 3.209984 7.145800      1.553090       2.569731   7.224743 2.373217
```

```
table(HC.4)
```

```
## HC.4
##    1    2    3    4
## 4594  696 3559 1262
```

We know that the greater the variability is within the data, the stronger the attribute's influence on clustering. Additionally, From the normalization ranges we can deduce that `Region`, `Item Type` and `Total Cost` would have greater influence as there are more categories to consider.

For the 4 cluster analysis we can note the following:

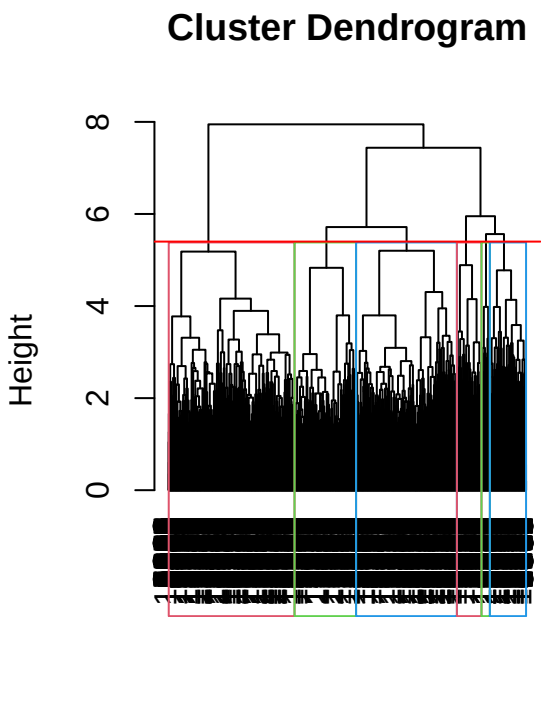
1. 'Item Type' had the most influence on the clusters
2. 'Wait Time' and 'Sales Channel' had almost no influence
3. Majority of the data was placed in cluster 1 and 3

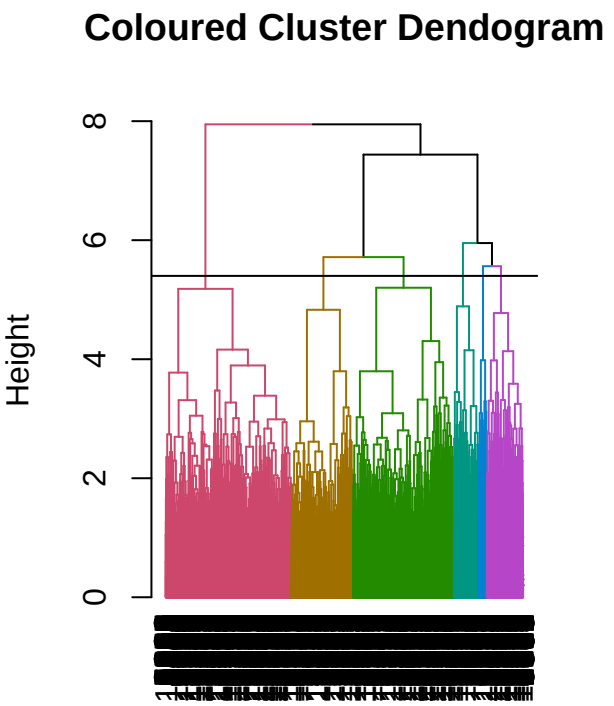
6 Cluster Analysis

```
par(mfrow=c(1,2))

plot(H.Cluster, hang = -1)
rect.hclust(H.Cluster, k = 6, border = 2:4)
abline(h = 5.4, col = 'red')

plot(color_branches(as.dendrogram(H.Cluster), h = 5.4),
     main = 'Coloured Cluster Dendogram',
     sub = '(6 Clusters)',
     ylab = 'Height')
abline(h = 5.4, col = 'black')
```






```
aggregate(SalesData, by=list(HC.6), FUN=mean)
```

```
##   Group.1   Region ItemType Sales.Channel Order.Priority Total.Cost Wait.Cat
## 1       1 7.350746 9.347302      1.549943       2.555109   2.097638 2.373708
## 2       2 7.352011 7.310345      1.494253       2.362069   7.448594 2.311782
## 3       3 4.820455 2.690643      1.558584       2.494521   2.286418 2.375386
## 4       4 3.271038 9.271739      1.558555       2.484572   2.115814 2.366059
## 5       5 2.974684 7.890295      1.502110       2.497890  10.584778 2.485232
## 6       6 3.264390 6.973659      1.564878       2.586341   6.447838 2.347317
```

```
table(HC.6)
```

```
## HC.6
##    1    2    3    4    5    6
## 1742  696 3559 2852  237 1025
```

For the 6 cluster analysis we can note the following:

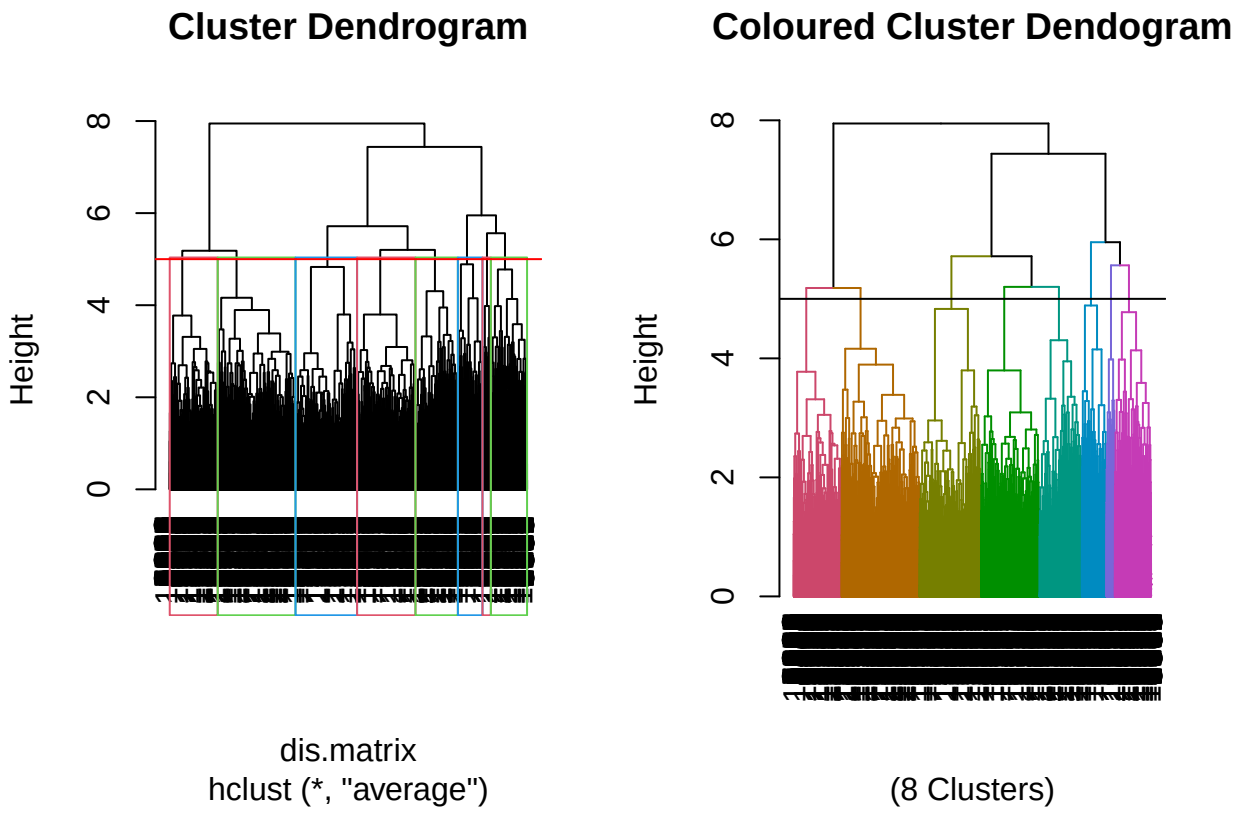
1. 'Total Cost' now has the most influence on the clusters
2. 'Wait Time','Order Priority' and 'Sales Channel' had almost no influence
3. Majority of the data is still in cluster 1 and 3
4. The number of data points in cluster 2 has not changed. That is, the similarity within this cluster is stronger that that of the others.

8 Cluster Analysis

```
par(mfrow=c(1,2))

plot(H.Cluster, hang = -1)
rect.hclust(H.Cluster, k = 8, border = 2:4)
abline(h = 5, col = 'red')

plot(color_branches(as.dendrogram(H.Cluster), h = 5),
     main = 'Coloured Cluster Dendogram',
     sub = '(8 Clusters)',
     ylab = 'Height')
abline(h = 5, col = 'black')
```




```
aggregate(SalesData, by=list(HC.8), FUN=mean)
```

```
##   Group.1   Region  ItemType Sales.Channel Order.Priority Total.Cost Wait.Cat
## 1       1 7.350746  9.347302      1.549943       2.555109   2.097638 2.373708
## 2       2 7.352011  7.310345      1.494253       2.362069   7.448594 2.311782
## 3       3 7.337261  2.712077      1.525773       2.492636   2.308118 2.390280
## 4       4 3.267606  2.677419      1.578828       2.495684   2.273030 2.366197
## 5       5 3.317500  7.003333      1.540000       2.440833   2.225493 2.380833
## 6       6 2.974684  7.890295      1.502110       2.497890  10.584778 2.485232
## 7       7 3.237288 10.919492      1.572034       2.516344   2.036144 2.355327
## 8       8 3.264390  6.973659      1.564878       2.586341   6.447838 2.347317
```

```
table(HC.8)
```

```
## HC.8
##    1    2    3    4    5    6    7    8
## 1742  696 1358 2201 1200  237 1652 1025
```

For the 8 cluster analysis we can note the following:

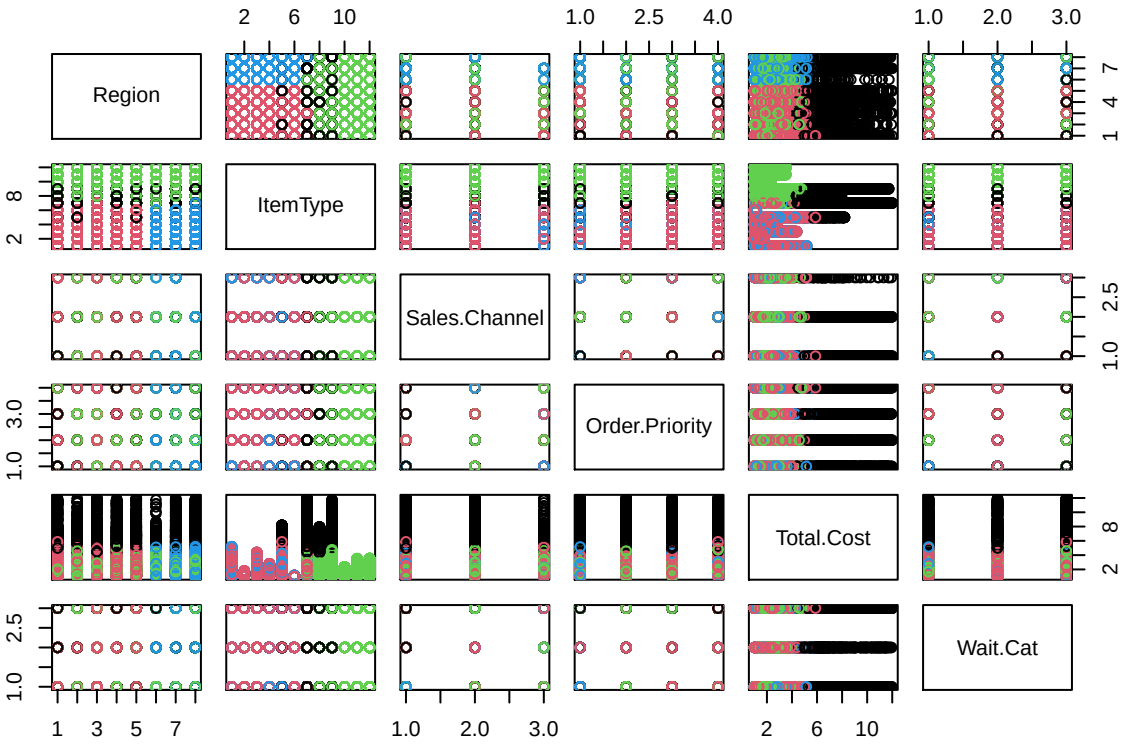
1. 'Total Cost' and 'Item Type' are both superior influences in forming the 8 clusters
2. 'Wait Time', 'Order Priority' and 'Sales Channel' had almost no influence
3. Majority of the data was placed in cluster 4
4. The number of data points in cluster 1 and cluster 2 has not changed. That is, the similarity within these cluster is very stronger and the dissimilarity with the other clusters are strong as well.
5. We can infer the a 3 or 4 cluster analysis would be a good fit for the data set.

K-Means Cluster Analysis

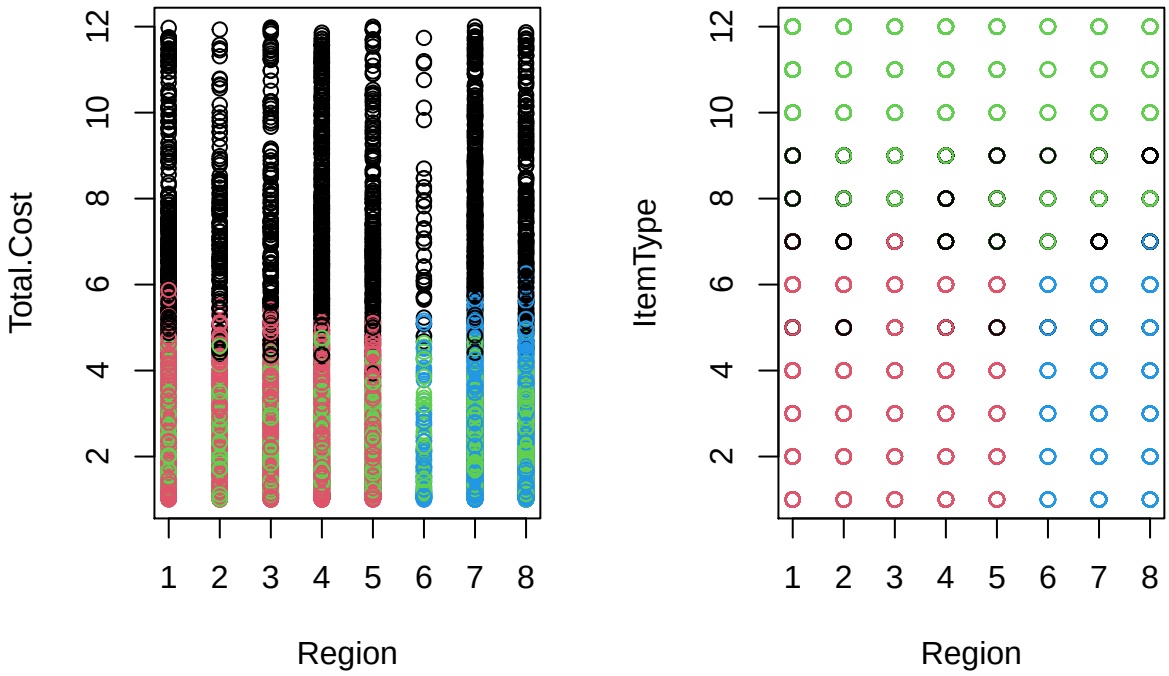
```
KM.4 <- kmeans(SalesData,4)
KM.6 <- kmeans(SalesData,6)
KM.8 <- kmeans(SalesData,8)
```

4 Cluster Analysis

```
plot(SalesData,col=KM.4$cluster)
```



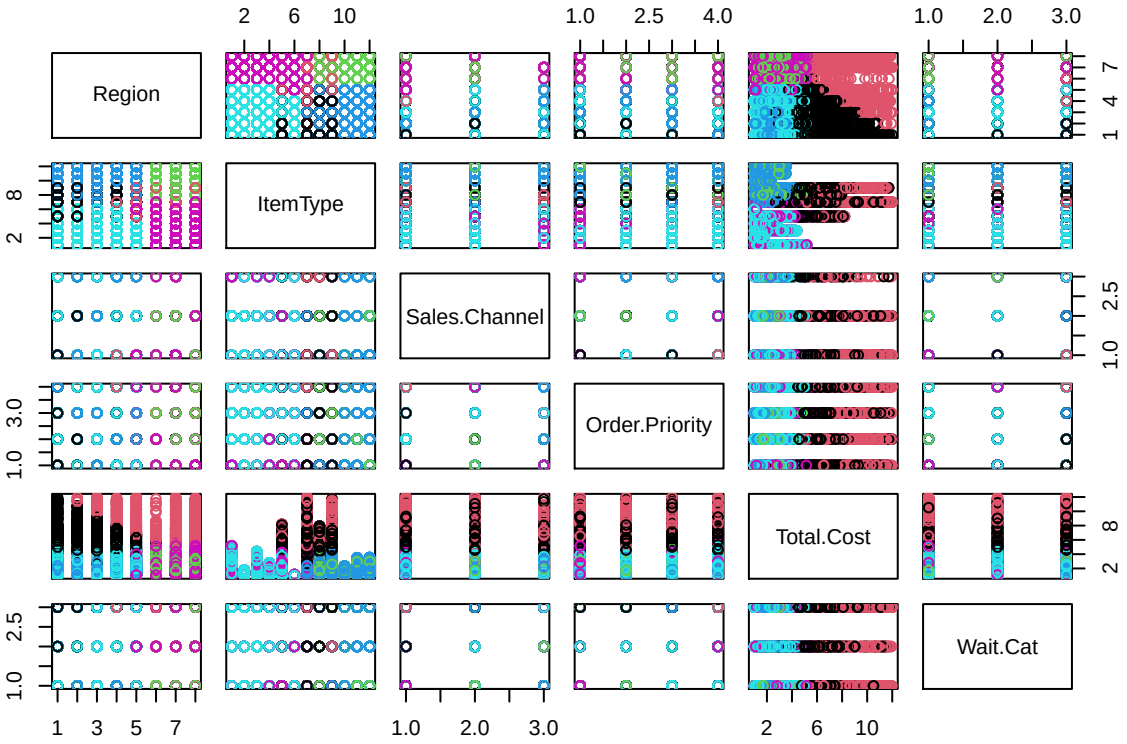
```
par(mfrow=c(1,2))
plot(Total.Cost ~ Region, SalesData, col=KM.4$cluster)
plot(ItemType ~ Region, SalesData, col=KM.4$cluster)
```



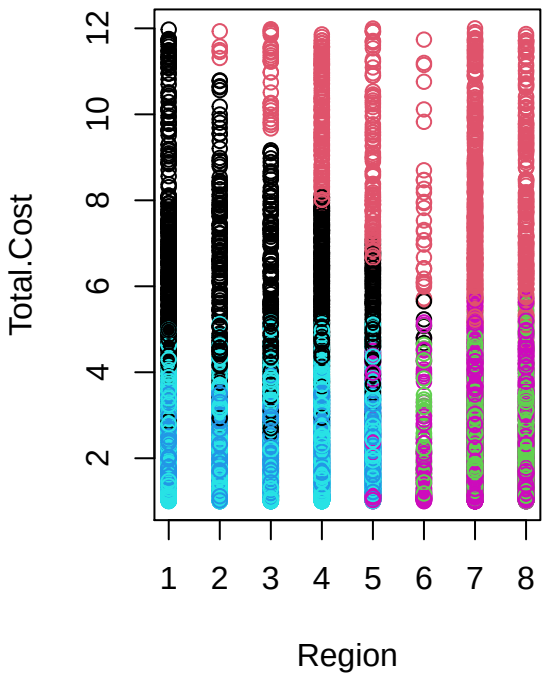
After viewing the visualizing the results we saw some interesting mapping the the data. We see some distinct clusters being formed when looking at `Total Cost` vs `Region` and `Item Type` vs `Region`. Taking a closer look, we can see three distinct clusters and a 4th clusters overlapping into all other 3 clusters.

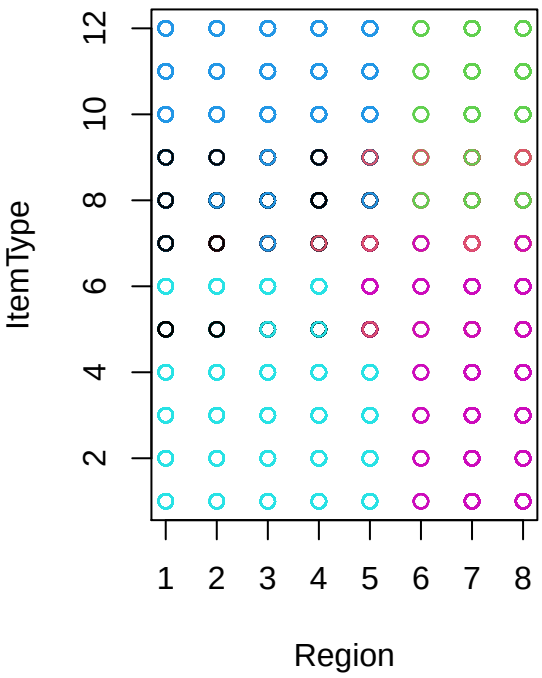
6 Cluster Analysis

```
plot(SalesData,col=KM.6$cluster)
```




```
par(mfrow=c(1,2))
plot(Total.Cost ~ Region, SalesData, col=KM.6$cluster)
plot(ItemType ~ Region, SalesData, col=KM.6$cluster)
```

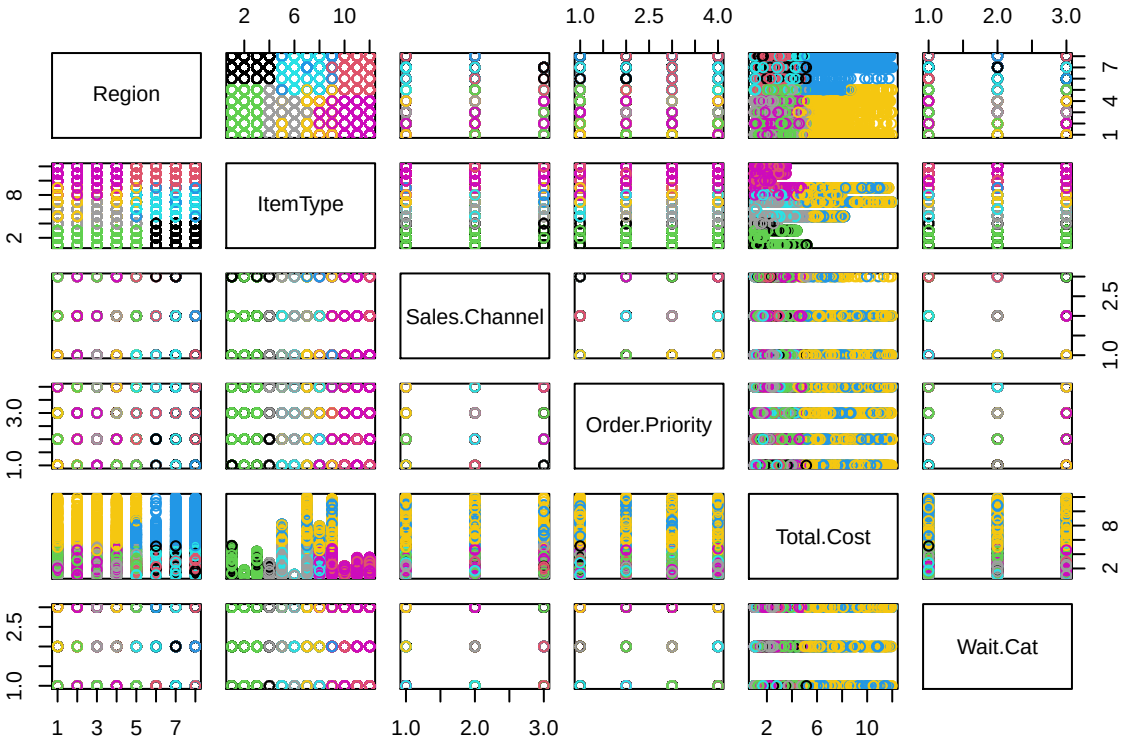




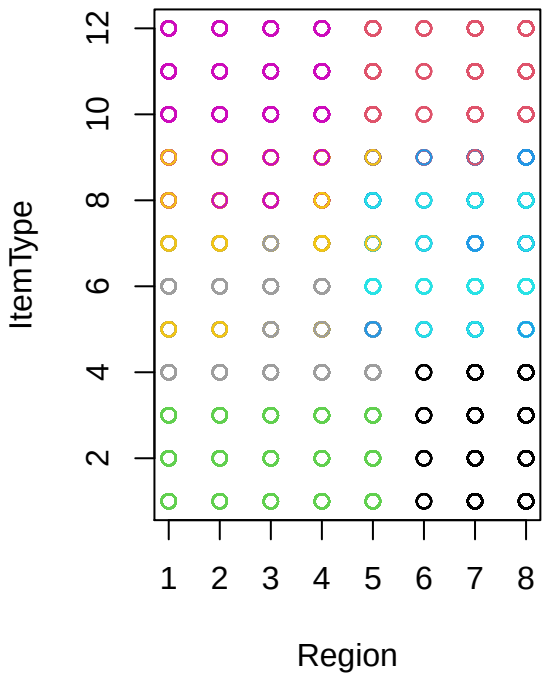
Looking at the visualization results from the 6 cluster analysis, we see 1 distinct distinct, coloured black, cluster being formed when looking at `Total Cost` vs `Region`. For the other clusters there, seems to be overlapping. Looking at the pattern, it would appear that the other clusters would be best separated into two clusters. When looking at the`Item Type` vs `Region` we can see 5 distinct clusters and the 6th cluster overlapping into all other 5 clusters.

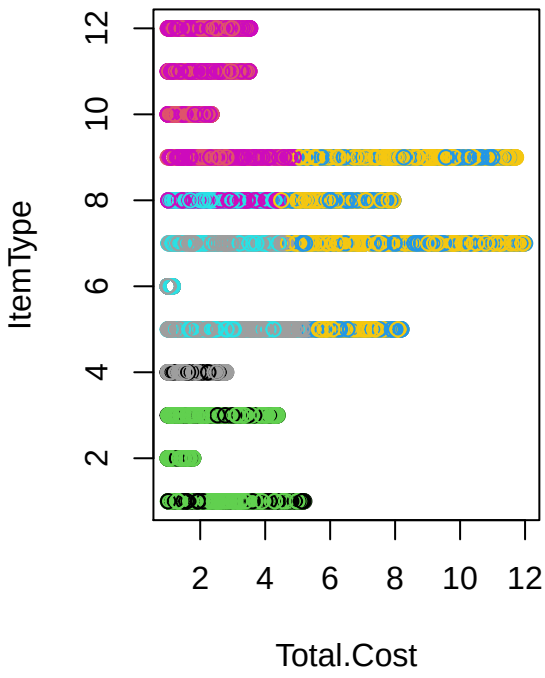
8 Cluster Analysis

```
plot(SalesData,col=KM.8$cluster)
```




```
par(mfrow=c(1,2))
plot(ItemType ~ Region, SalesData, col=KM.8$cluster)
plot(ItemType ~ Total.Cost, SalesData, col=KM.8$cluster)
```



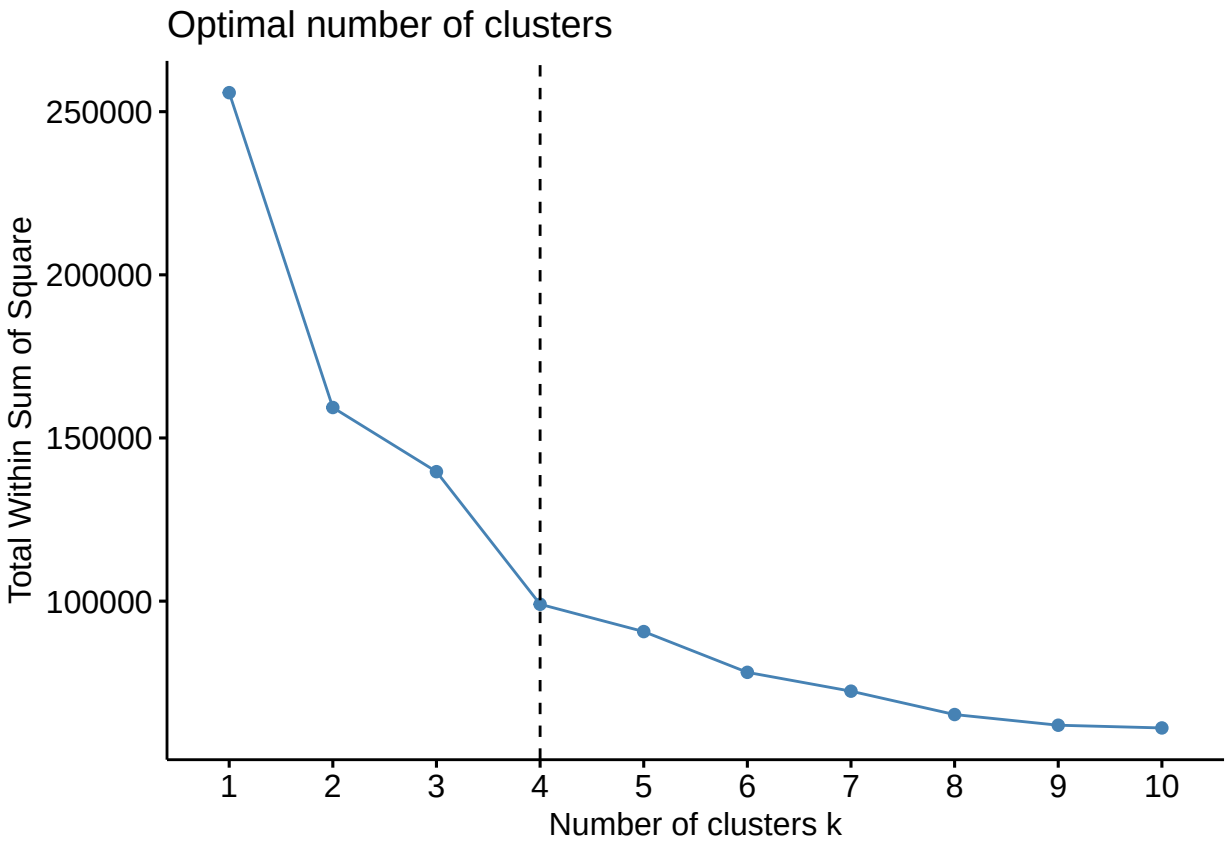


Now looking at the 8 cluster visualization and taking a deeper look into the attributes that seemingly presents patterns, we notice however that there are significant overlapping in some of the clusters. Look at `Item Type` vs `Region` we can see 3 distinct clusters (blue, black and green). For the other 4 clusters, there were significant overlapping where these 4 groups could be viewed as two separate groups.

With the `Total Cost` vs `Item Type` we can see significant overlapping, however some distinct groups can be inferred. Looking at the visualization you can see 4 section that stand out, which would imply that they would provide more insights with a 4 cluster analysis.

Closing Remarks

```
fviz_nbclust(SalesData, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)
```




```
  labs(subtitle = "Elbow method")
```

```
## $subtitle
## [1] "Elbow method"
##
## attr(,"class")
## [1] "labels"
```

K-means clustering define clusters such that the total intra-cluster variation (total within-cluster sum of square) is minimized. The elbow method looks at the percentage of variance explained as a function of the number of clusters. One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. Here the bend occurs at 4 clusters. This method tells us that the most ideal number of clusters is 4.

In closing, in the agglomorative clustering technique we had seen where the ideal number of clusters was inferred to be 3 or 4. We also saw this with the k-means cluster visualization. A confirmation from the elbow method would here assists in choosing 4 clusters over 3 for this data set.