# Market Segmentation and Preference Analysis

**CSDA1050 Group2 - Fanny, Deenu, Dave and Kaustubh**

04/26/2020

Before loading the data, we start by installing and evoking all the necessary libraries.

## 1 Introduction

Our primary dataset consists of 4 years of US Sales from the retail store `Superstore`. The data set was obtained from Kaggle at the following URL `https://www.kaggle.com/aksha17/superstore-sales`. This dataset is made of 21 variables over 9,994 rows, has 4 continuous variables (Sales, Quantity, Discount, and Profit), and the rest of the variables are categorical, and mainly consist of demographic and product information.

Let us begin with reading the data file downloaded from kaggle and store as dataframe object.

```
#Import the data
supstore_df=read.csv('/Users/deenuy/Documents/Google Drive/Personal/CSDA1050-Capstone-Project-Repo/Da

#supstore_df=read.csv('E:/Deenu/Personal Workspace/GoogleDrive/CSDA1050-Capstone-Project-Repo/Dataset

#Display table with 5 rows
show_table(supstore_df, 5, isknit = FALSE)
```

| | Row.... <int> | Order.ID <fctr> | Order.Date <fctr> | Ship.Date <fctr> | Ship.Mode <fctr> |
|---|---|---|---|---|---|
| 1 | 1 | CA-2016-152156 | 08-11-16 | 11-11-16 | Second Class |
| 2 | 2 | CA-2016-152156 | 08-11-16 | 11-11-16 | Second Class |
| 3 | 3 | CA-2016-138688 | 12-06-16 | 16-06-16 | Second Class |
| 4 | 4 | US-2015-108966 | 11-10-15 | 18-10-15 | Standard Class |
| 5 | 5 | US-2015-108966 | 11-10-15 | 18-10-15 | Standard Class |

5 rows | 1-6 of 22 columns

Print the dimension of dataframe object.

```
## [1] "Dataframe size is: 9994 x 21"
```

## 2 Data Preparation

Before we dive into exploratory analysis, it is important that the data is cleaned to extract meaningful insights. Let us address the following problems one by one as scope of data cleansing.

- Converting to factor variables, numeric variables & handling dates
- Removing least useful informations for downsizing the dataframe object
- Analyze and handle missing values
- Data trasformation, Checking Linearity, Skewness and generate plot histogram

### 2.0.1 Converting to factor variables, numeric variables & handling dates

Describe the data types for each column using function str().

```
#Summarize the data
str(supstore_df) ##to find data types
```

```
## 'data.frame':    9994 obs. of  21 variables:
##  $ Row.ID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Order.ID     : Factor w/ 5009 levels "CA-2014-100006",..: 2501 2501 2297 4373 4373 202 202 202
##  $ Order.Date   : Factor w/ 1237 levels "01-01-17","01-02-14",..: 318 318 461 436 436 341 341 341
##  $ Ship.Date    : Factor w/ 1334 levels "01-01-15","01-01-16",..: 478 478 675 776 776 586 586 586
##  $ Ship.Mode    : Factor w/ 4 levels "First Class",..: 3 3 3 4 4 4 4 4 4 4 ...
##  $ Customer.ID  : Factor w/ 793 levels "AA-10315","AA-10375",..: 144 144 240 706 706 89 89 89 89 8
##  $ Customer.Name: Factor w/ 793 levels "Aaron Bergman",..: 167 167 202 688 688 114 114 114 114 114
##  $ Segment      : Factor w/ 3 levels "Consumer","Corporate",..: 1 1 2 1 1 1 1 1 1 1 ...
##  $ Country      : Factor w/ 1 level "United States": 1 1 1 1 1 1 1 1 1 1 ...
##  $ City         : Factor w/ 531 levels "Aberdeen","Abilene",..: 195 195 267 154 154 267 267 267 26
##  $ State        : Factor w/ 49 levels "Alabama","Arizona",..: 16 16 4 9 9 4 4 4 4 4 ...
##  $ Postal.Code  : int  42420 42420 90036 33311 33311 90032 90032 90032 90032 90032 ...
##  $ Region       : Factor w/ 4 levels "Central","East",..: 3 3 4 3 3 4 4 4 4 4 ...
##  $ Product.ID   : Factor w/ 1862 levels "FUR-BO-10000112",..: 13 56 947 320 1317 186 563 1762 795
##  $ Category     : Factor w/ 3 levels "Furniture","Office Supplies",..: 1 1 2 1 2 1 2 3 2 2 ...
##  $ Sub.Category : Factor w/ 17 levels "Accessories",..: 5 6 11 17 15 10 3 14 4 2 ...
##  $ Product.Name : Factor w/ 1850 levels "\"While you Were Out\" Message Book, One Form per Page",
##  $ Sales        : num  262 731.9 14.6 957.6 22.4 ...
##  $ Quantity     : int  2 3 2 5 2 7 4 6 3 5 ...
##  $ Discount     : num  0 0 0 0.45 0.2 0 0 0.2 0.2 0 ...
##  $ Profit       : num  41.91 219.58 6.87 -383.03 2.52 ...
```

From above it is identified that data type needs to be fixed for attributes Order.date, ship.date and postal.code.

```
#Data format conversion process
supstore_df$Order.Date = parse_date_time(supstore_df$Order.Date, "dmy")
supstore_df$Ship.Date = parse_date_time(supstore_df$Ship.Date, "dmy")
supstore_df$Postal.Code = as.factor(supstore_df$Postal.Code)
```

Generate the summary to each column and collating the statistical results to understand our data with mean, median, min, max and null values.

```
#Summarize the data
summary(supstore_df)
```

```
##      Row.ID              Order.ID      Order.Date
##  Min.   :   1   CA-2017-100111:  14   Min.   :2014-01-03 00:00:00
##  1st Qu.:2499   CA-2017-157987:  12   1st Qu.:2015-05-23 00:00:00
##  Median :4998   CA-2016-165330:  11   Median :2016-06-26 00:00:00
##  Mean   :4998   US-2016-108504:  11   Mean   :2016-04-30 00:07:12
##  3rd Qu.:7496   CA-2015-131338:  10   3rd Qu.:2017-05-14 00:00:00
##  Max.   :9994   CA-2016-105732:  10   Max.   :2017-12-30 00:00:00
##                 (Other)       :9926
```

```
##    Ship.Date                         Ship.Mode      Customer.ID
##  Min.   :2014-01-07 00:00:00   First Class   :1538   WB-21850:  37
##  1st Qu.:2015-05-27 00:00:00   Same Day      : 543   JL-15835:  34
##  Median :2016-06-29 00:00:00   Second Class  :1945   MA-17560:  34
##  Mean   :2016-05-03 23:06:58   Standard Class:5968   PP-18955:  34
##  3rd Qu.:2017-05-18 00:00:00                         CK-12205:  32
##  Max.   :2018-01-05 00:00:00                         EH-13765:  32
##                                                      (Other) :9791
##             Customer.Name         Segment              Country
##  William Brown      :  37   Consumer   :5191   United States:9994
##  John Lee           :  34   Corporate  :3020
##  Matt Abelman       :  34   Home Office:1783
##  Paul Prost         :  34
##  Chloris Kastensmidt:  32

##  Edward Hooks       :  32
##  (Other)            :9791
##            City               State       Postal.Code       Region
##  New York City: 915   California  :2001   10035  : 263   Central:2323
##  Los Angeles  : 747   New York    :1128   10024  : 230   East   :2848
##  Philadelphia : 537   Texas       : 985   10009  : 229   South  :1620
##  San Francisco: 510   Pennsylvania: 587   94122  : 203   West   :3203
##  Seattle      : 428   Washington  : 506   10011  : 193
##  Houston      : 377   Illinois    : 492   94110  : 166
##  (Other)      :6480   (Other)     :4295   (Other):8710
##           Product.ID             Category         Sub.Category
##  OFF-PA-10001970:  19   Furniture      :2121   Binders    :1523
##  TEC-AC-10003832:  18   Office Supplies:6026   Paper      :1370
##  FUR-FU-10004270:  16   Technology     :1847   Furnishings: 957
##  FUR-CH-10001146:  15                          Phones     : 889
##  FUR-CH-10002647:  15                          Storage    : 846
##  TEC-AC-10002049:  15                          Art        : 796
##  (Other)        :9896                          (Other)    :3613
##                     Product.Name      Sales              Quantity
##  Staple envelope           :  48   Min.   :    0.444   Min.   : 1.00
##  Easy-staple paper         :  46   1st Qu.:   17.280   1st Qu.: 2.00
##  Staples                   :  46   Median :   54.490   Median : 3.00
##  Avery Non-Stick Binders   :  20   Mean   :  229.858   Mean   : 3.79
##  Staples in misc. colors   :  19   3rd Qu.:  209.940   3rd Qu.: 5.00
##  KI Adjustable-Height Table:  18   Max.   :22638.480   Max.   :14.00
##  (Other)                   :9797
##    Discount          Profit
##  Min.   :0.0000   Min.   :-6599.978
##  1st Qu.:0.0000   1st Qu.:    1.729
##  Median :0.2000   Median :    8.666
##  Mean   :0.1562   Mean   :   28.657
##  3rd Qu.:0.2000   3rd Qu.:   29.364
##  Max.   :0.8000   Max.   : 8399.976
##
```

### 2.0.2 Removing least useful informations for downsizing the dataframe object

In consideration of ethical machine learning framework, removing the customer name from our dataset to mask the identity.

```
#remove the columns
supstore_df = select(supstore_df, -"Customer.Name")

#list columns
names(supstore_df)
```

```
##  [1] "Row.ID"       "Order.ID"     "Order.Date"   "Ship.Date"    "Ship.Mode"
```

```
##  [6] "Customer.ID"  "Segment"      "Country"      "City"         "State"
## [11] "Postal.Code"  "Region"       "Product.ID"   "Category"     "Sub.Category"
## [16] "Product.Name" "Sales"        "Quantity"     "Discount"     "Profit"
```

### 2.0.3 Analyze and handle missing values

```
#Find is NA count and null values function
gen_report_na_func = function(dataset) {
  summary.na.df =
    dataset %>%
    map_dfr(~sum(is.na(.)))

  #Transpose the table
  summary.na.df = as.data.frame(t(as.matrix(summary.na.df)))
  setDT(summary.na.df, keep.rownames = "newname")
  names(summary.na.df)[1] = "column_names"
  names(summary.na.df)[2] = "cols"

  #Generate summary of NA in all the columns
  summary.na_tb =
    summary.na.df %>%
    group_by(column_names) %>%
    summarise(na_count = sum(cols), na_pcnt = (sum(cols)/nrow(dataset))*100) %>%
    arrange(desc(na_count))

  #Round of to two digit in percentage
  summary.na_tb$na_pcnt = round(summary.na_tb$na_pcnt,2)

  # View(summary.na_tb)
  show_table(summary.na_tb, 10, isknit = TRUE)
}

#Generate missing NA report
gen_report_na_func(supstore_df)
```

| column_names | na_count | na_pcnt |
|---|---|---|
| Category | 0 | 0 |
| City | 0 | 0 |
| Country | 0 | 0 |
| Customer.ID | 0 | 0 |
| Discount | 0 | 0 |
| Order.Date | 0 | 0 |
| Order.ID | 0 | 0 |
| Postal.Code | 0 | 0 |
| Product.ID | 0 | 0 |
| Product.Name | 0 | 0 |

```
#Adding columns for data and time individually.

supstore_df_new =
  supstore_df %>%
```

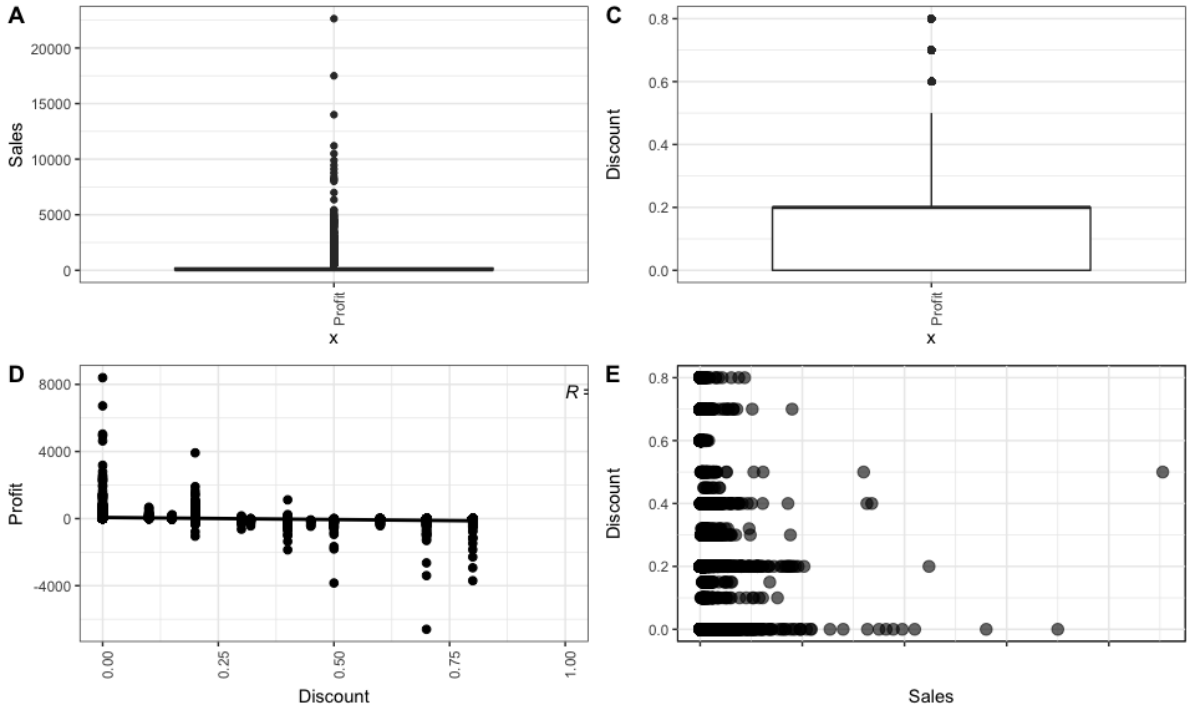
```
  mutate(Profit_pct = round( (Profit/Sales)*100 ,2),
         Ordered.Year = lubridate::year(Order.Date),
         Ordered.Quarter = lubridate::quarter(Order.Date),
         Ordered.Month = lubridate::month(Order.Date, label=TRUE, abbr=TRUE),
         Ordered.Day = lubridate::day(Order.Date),
         Ordered.Week.day = lubridate::wday(Order.Date, week_start = 1, label=TRUE, abbr=TRUE),
         Delivery_time = difftime(Ship.Date, Order.Date, units = c("days"))
         )

#summary(supstore_df_new)
```

### 2.0.4 Data trasformation, Checking Linearity, Skewness and generate plot histogram

Analyze the univariate outliers for the sales

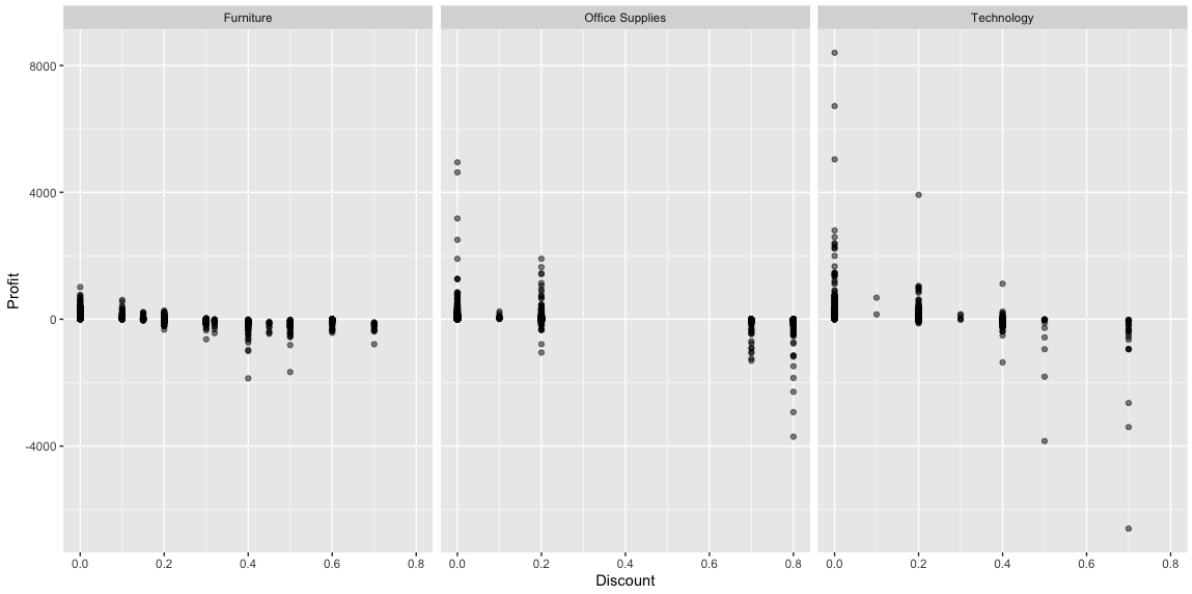
```
# use a box plot to see if we have outliers
a = ggplot(supstore_df, aes(x = "Profit", y = Sales)) +
    geom_boxplot() +
    theme_function()

# use a Normality Distribution by Q-Q plot to see if we have outliers
#qqnorm(supstore_df$Sales)
#qqline(supstore_df$Sales)

#The fitdistr( ) function in the MASS package provides maximum-likelihood fitting of univariate distr
#fitdistr(Victim.Age, "normal")

# use a box plot to see if we have outliers
c = ggplot(supstore_df, aes(x = "Profit", y = Discount)) +
    geom_boxplot() +
    theme_function()

# Scatter plots (sp)
d = ggscatter(supstore_df_new, x = "Discount", y = "Profit",
                add = "reg.line",               # Add regression line
                conf.int = TRUE,                # Add confidence interval
                ) +
  theme_function() +
  stat_cor(aes(color = Profit), label.x = 1)# Add correlation coefficient


e = ggscatter(supstore_df_new, x = "Sales", y = "Discount",
                palette = "jco",
                size = 3, alpha = 0.6)+
  theme_function() +
  border()

ggarrange(a, c, d, e + rremove("x.text"),
          labels = c("A", "C", "D", "E"),
          ncol = 2, nrow = 2, align = "hv")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



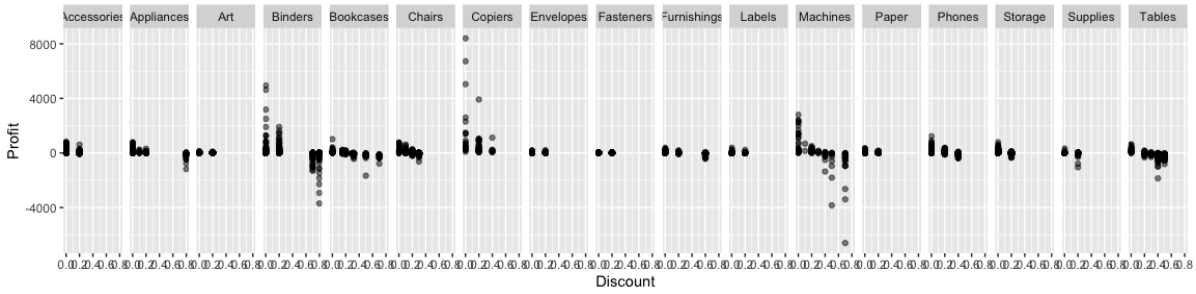

### 2.0.5 Analyzing the profit and discount by category

```
ggplot(supstore_df_new, aes(x = Discount, y = Profit)) +
  geom_point(alpha = 0.5) +
  scale_color_manual(values = c("TRUE" = "red", "FALSE" = "black"),
                     guide = FALSE) +
  facet_grid(~Category)
```



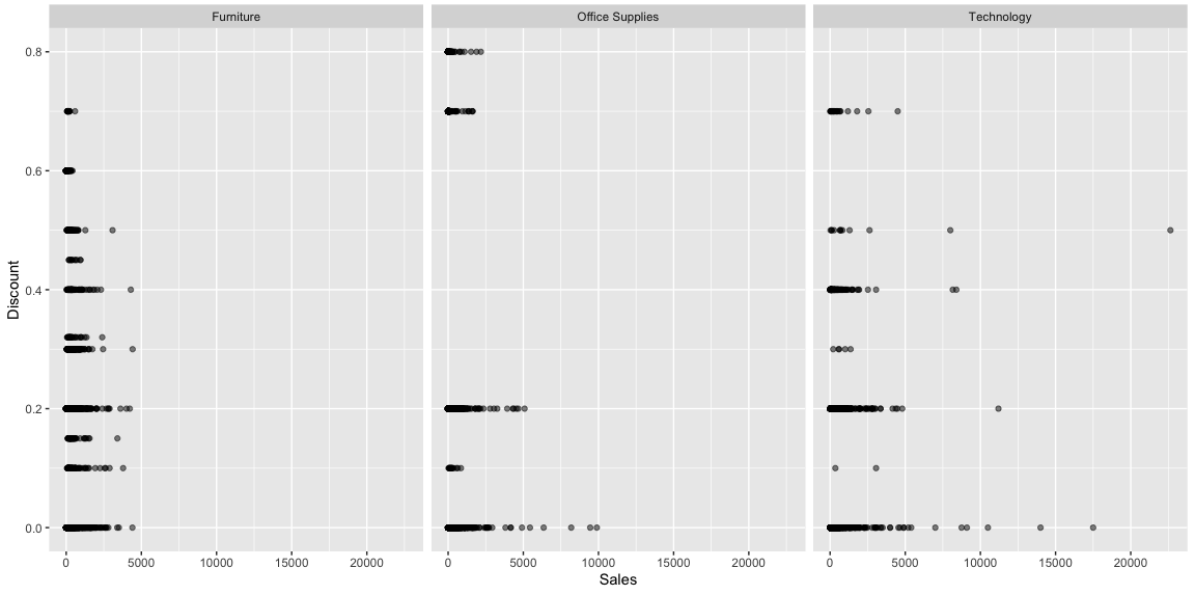

### 2.0.6 Analyzing the profit and discount by sub category

```
ggplot(supstore_df_new, aes(x = Discount, y = Profit)) +
  geom_point(alpha = 0.5) +
  scale_color_manual(values = c("TRUE" = "red", "FALSE" = "black"),
                     guide = FALSE) +
  facet_grid(~Sub.Category)
```



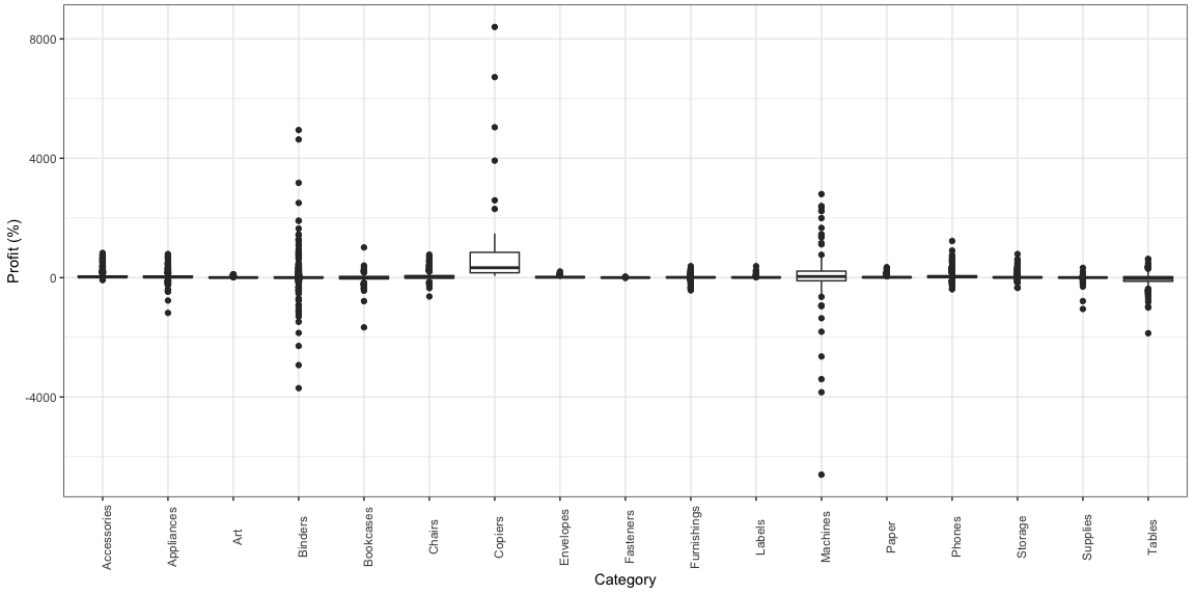

## 2.0.7 Analyzing the profit and discount by customer segment

```
ggplot(supstore_df_new, aes(x = Sales, y = Discount)) +
  geom_point(alpha = 0.5) +
  scale_color_manual(values = c("TRUE" = "red", "FALSE" = "black"),
                     guide = FALSE) +
  facet_grid(~Category)
```




## 2.0.8 Analyze for any outliers in the sales by category

```
ggplot(supstore_df_new, aes(x = Sub.Category, y = Profit)) +
  geom_boxplot() +
  theme_function() +
  xlab("Category")+ ylab("Profit (%)")
```




# 3 Data Exploration and Analysis

### 3.0.1 Analysing the superstore customer orders by state or province in US

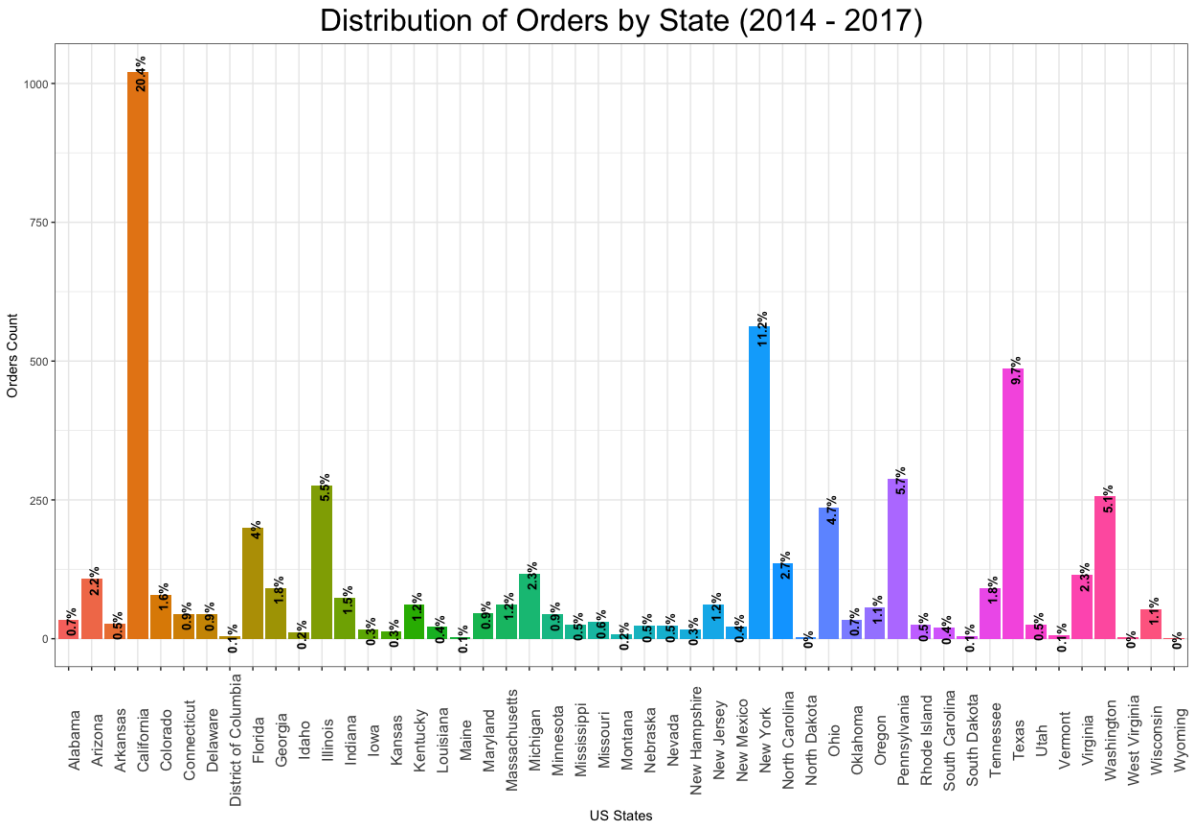
From top and bottow 5 states result and plot, we can see that the California, New York and Texas have high demand of customers with quantity of 1000+ orders and whereas states like Wyoming, West Virginia, North Dakota and Maine are in single digit orders. Does this data shows any significant opportunities for more promotional investments for marketing team?

```
#Group by function to count number of orders by state
supstore_df_bystate =
    supstore_df_new %>%
    group_by(State) %>%
    summarise(Orders = n_distinct(Order.ID),
              Average.profit = round(mean(Profit)),
              Average.sales = round(mean(Sales))) %>%
    mutate(Avg.sales.pct = round( (Average.sales/sum(Average.sales) * 100), 1),
           Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
    arrange(desc(Orders))

show_table(supstore_df_bystate, 10, isknit = TRUE)
```

| State | Orders | Average.profit | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|---|
| California | 1021 | 38 | 229 | 1.7 | 20.4 |
| New York | 562 | 66 | 276 | 2.0 | 11.2 |
| Texas | 487 | -26 | 173 | 1.3 | 9.7 |
| Pennsylvania | 288 | -27 | 198 | 1.5 | 5.7 |
| Illinois | 276 | -26 | 163 | 1.2 | 5.5 |
| Washington | 256 | 66 | 274 | 2.0 | 5.1 |
| Ohio | 236 | -36 | 167 | 1.2 | 4.7 |
| Florida | 200 | -9 | 234 | 1.7 | 4.0 |

| State | Orders | Average.profit | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|---|
| North Carolina | 136 | -30 | 223 | 1.6 | 2.7 |
| Michigan | 117 | 96 | 299 | 2.2 | 2.3 |

```
#Plot the orders by state
ggplot(supstore_df_bystate,
       aes(State, Orders, fill=State)) +
  geom_bar(stat="identity") +
  theme_bw() +
    theme(legend.position="none",
          plot.title = element_text(size=24, hjust= 0.5),
          axis.text.x = element_text(size=12, angle=90)
          ) +
  scale_color_gradient2() +
  labs(x="US States",y="Orders Count",title="Distribution of Orders by State (2014 - 2017)") +
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")),vjust=0.8,size=3.5,color='black',angle = 90, f
```




```
#Display least orders by state
knitr::kable(tail(supstore_df_bystate, 5))
```

| State | Orders | Average.profit | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|---|
| District of Columbia | 4 | 106 | 287 | 2.1 | 0.1 |
| Maine | 3 | 57 | 159 | 1.2 | 0.1 |
| North Dakota | 2 | 33 | 131 | 1.0 | 0.0 |
| West Virginia | 2 | 46 | 302 | 2.2 | 0.0 |

| State | Orders | Average.profit | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|---|
| Wyoming | 1 | 100 | 1603 | 11.8 | 0.0 |

```
#tail(supstore_df_bystate, 5)
```

### 3.0.2 Analyze the shipping mode preferences for online orders

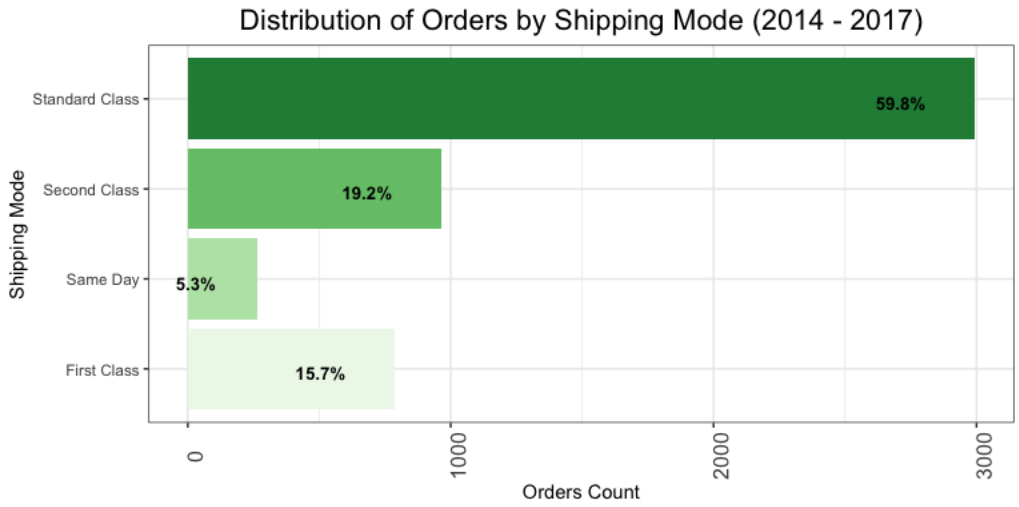
We can see that the most commonly used ship mode is Standard Shipping and the least used is Same Day shipping.

```
#Group by shipping mode
supstore_df_byShipMode =
  supstore_df %>%
  group_by(Ship.Mode) %>%
  summarise(Orders = n_distinct(Order.ID),
            Average.sales = round(mean(Sales))) %>%
  mutate(Avg.sales.pct = round( (Average.sales/sum(Average.sales) * 100), 1),
         Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange(desc(Orders.pct))

#View the results in table format
show_table(supstore_df_byShipMode, 5, isknit = TRUE)
```

| Ship.Mode | Orders | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|
| Standard Class | 2994 | 228 | 24.6 | 59.8 |
| Second Class | 964 | 236 | 25.4 | 19.2 |
| First Class | 787 | 228 | 24.6 | 15.7 |
| Same Day | 264 | 236 | 25.4 | 5.3 |

Plotting the shipping mode for visualization

```
# Create the barplot for orders placed by shipping mode preference
ggplot(supstore_df_byShipMode,
       aes(Ship.Mode, Orders, fill=Ship.Mode)) +
  geom_bar(stat="identity") +
  theme_bw() +
    theme(legend.position="none",
          plot.title = element_text(size=16, hjust= 0.5),
          axis.text.x = element_text(size=12, angle=90)
          ) +
  scale_fill_brewer(palette = "Greens") +
  coord_flip() +
  labs(x="Shipping Mode", y="Orders Count", title="Distribution of Orders by Shipping Mode (2014 - 2
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")),hjust = 1.8, vjust=0.8,size=3.5,color='black',
```




### 3.0.3 Analyzing top 5 customers for frequent orders

```
#Group by customer ID
supstore_df_byCustomerId =
  supstore_df %>%
  group_by(Customer.ID) %>%
  summarise(Orders = n_distinct(Order.ID)) %>%
  arrange(desc(Orders))

#View the results in table format
show_table(supstore_df_byCustomerId, 5, isknit = TRUE)
```

| Customer.ID | Orders |
|---|---|
| EP-13915 | 17 |
| CK-12205 | 13 |
| EA-14035 | 13 |
| JE-15745 | 13 |
| NS-18640 | 13 |

```
## [1] "Number of unique orders observed is 793"
```

### 3.0.4 Analyze the catergorize that Superstore retailer have available online.

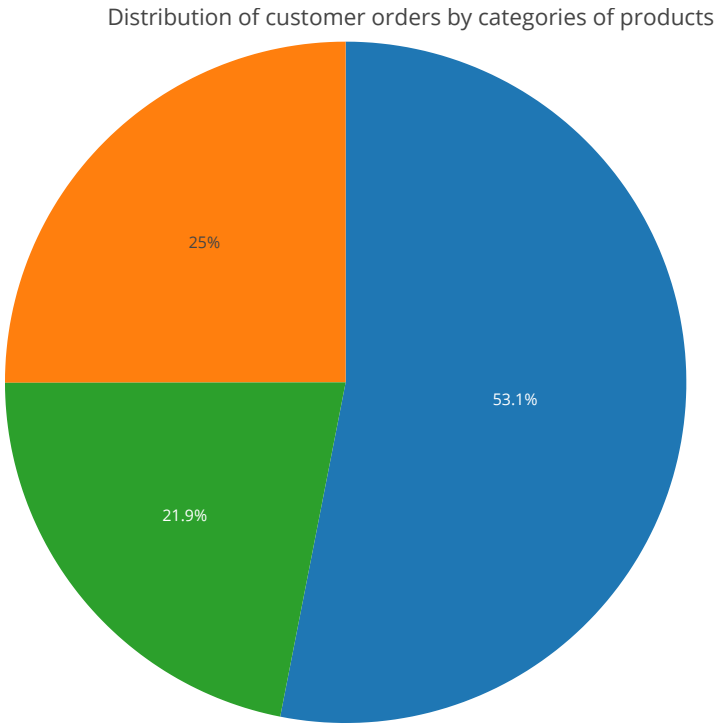
We can observe that office supplies have 60% of the consumer orders placed online between 2014-2017.

```
#Group by category of products
supstore_df_byCategory =
  supstore_df %>%
  group_by(Category) %>%
  summarise(Orders = n_distinct(Order.ID)) %>%
  mutate(Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange(desc(Orders.pct))

#View the results in table format
show_table(supstore_df_byCategory, 5, isknit = TRUE)
```

| Category | Orders | Orders.pct |
|---|---|---|
| Office Supplies | 3742 | 53.1 |
| Furniture | 1764 | 25.0 |
| Technology | 1544 | 21.9 |

Plot to visualize the orders by category

```
pie_cat = plot_ly(supstore_df_byCategory,  labels = ~Category,  values = ~Orders,  type = 'pie')
pie_cat = pie_cat %>% layout(title = 'Distribution of customer orders by categories of products',
                             xaxis = list(showgrid = FALSE,  zeroline = FALSE,  showticklabels = TR
                             yaxis = list(showgrid = FALSE,  zeroline = FALSE,  showticklabels = TR

pie_cat #Plotting pie chart of categories
```




### 3.0.5 Analyzing the subcategories of products

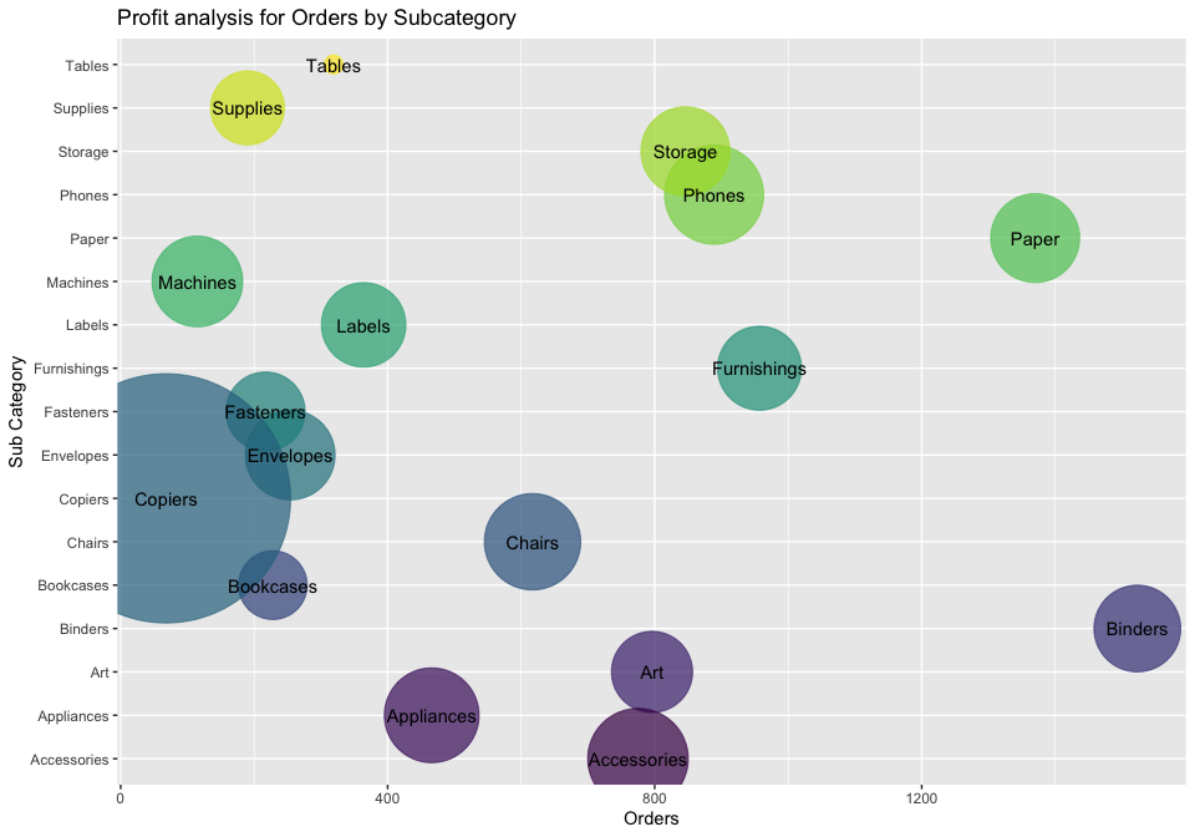
The top 10 results shows that there is more customer demands for home office products like binders, papers, phones, storage, chairs and accessories.

```
#Group by sub category of products
supstore_df_bySubCategory =
  supstore_df %>%
  group_by(Sub.Category, Category) %>%
  summarise(Orders = n(), Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))) %>%
  arrange(desc(Orders))

#View the results in table format
show_table(supstore_df_bySubCategory, 5, isknit = TRUE)
```

| Sub.Category | Category | Orders | Average.profit | Average.sales |
|---|---|---|---|---|
| Binders | Office Supplies | 1523 | 20 | 134 |
| Paper | Office Supplies | 1370 | 25 | 57 |
| Furnishings | Furniture | 957 | 14 | 96 |
| Phones | Technology | 889 | 50 | 371 |
| Storage | Office Supplies | 846 | 25 | 265 |

```
## [1] "Number of subcategory of products identified are 17"
```

```
# Initiate a ggplot
ggplot(supstore_df_bySubCategory, aes(x = Orders, y = Sub.Category)) +
  geom_point(aes(color = Sub.Category, size = Average.profit), alpha = 0.7) +
  geom_text(aes(label = Sub.Category) ) +
  theme(legend.position = "none") +
  scale_color_viridis_d(begin = 0) +
  scale_size(range = c(5, 70), guide = "none") +
  labs(title = "Profit analysis for Orders by Subcategory",
     x = "Orders",
     y = "Sub Category")
```




### 3.0.6 Analyze the online superstore retail sales by region in United States

Our observation is, southern region of United states has slightly lower online sales. Also, we can see that most of the orders are from the west, coinciding with the number of orders from California from our initial analysis.

```
#Group by region of products sales
supstore_df_byRegion =
  supstore_df %>%
  group_by(Region) %>%
  summarise(Orders = n_distinct(Order.ID),
            Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))) %>%
  mutate(Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange(desc(Orders))

#View the results in table format
show_table(supstore_df_byRegion, 5, isknit = TRUE)
```

| Region | Orders | Average.profit | Average.sales | Orders.pct |
|---|---|---|---|---|
| West | 1611 | 34 | 226 | 32.2 |
| East | 1401 | 32 | 238 | 28.0 |
| Central | 1175 | 17 | 216 | 23.5 |
| South | 822 | 29 | 242 | 16.4 |

### 3.0.7 Analyze the segements of customer orders by consumer type.

Below results shows the consumers are the most common customer segment with more than 50% of the total orders.

```
#Group by segments of consumer type
supstore_df_bySegments =
  supstore_df %>%
  group_by(Segment) %>%
  summarise(Orders = n_distinct(Order.ID),
            Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))) %>%
  mutate(Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange(desc(Orders))

#View the results in table format
show_table(supstore_df_bySegments, 5, isknit = TRUE)
```

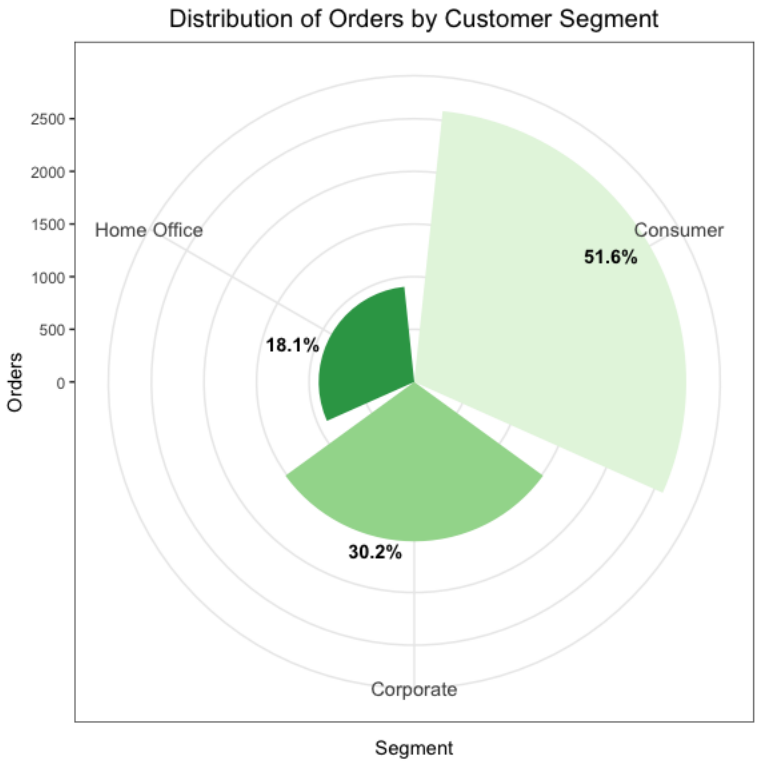
| Segment | Orders | Average.profit | Average.sales | Orders.pct |
|---|---|---|---|---|
| Consumer | 2586 | 26 | 224 | 51.6 |
| Corporate | 1514 | 30 | 234 | 30.2 |
| Home Office | 909 | 34 | 241 | 18.1 |

Plotting pie chart of composition of segment

Plot for visualization of orders by segment

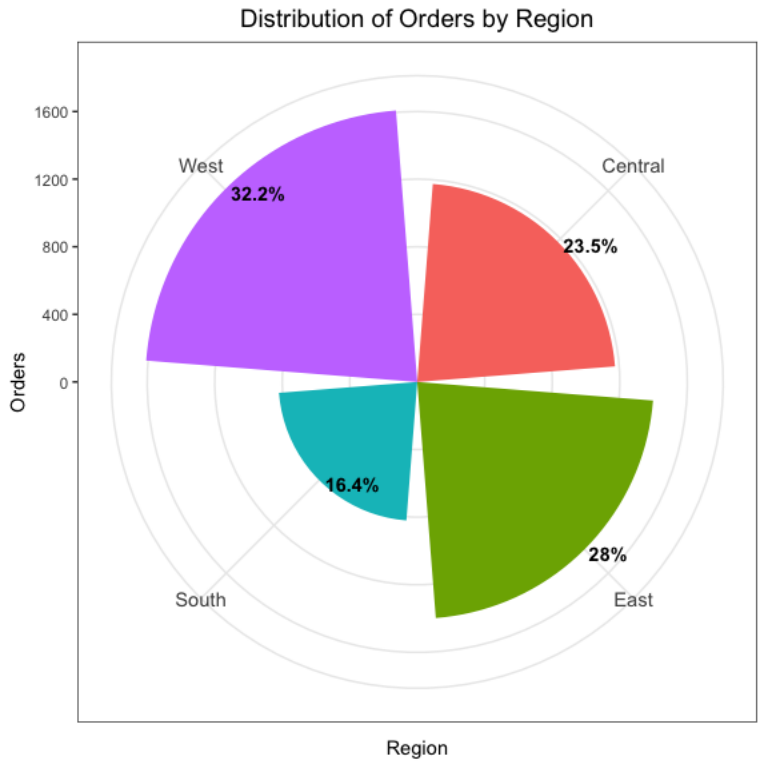
```
#make a coxcomb plot
ggplot(supstore_df_bySegments, aes(Segment, Orders, fill=Segment)) +
  geom_bar(stat="identity") +
      theme_bw() +
    theme(legend.position="none",
          plot.title = element_text(size=14, hjust= 0.5),
          axis.text.x = element_text(size=11, angle=0)
```

```
        ) +
  scale_fill_brewer(palette = "Greens") +
  coord_polar() +
  labs(title="Distribution of Orders by Customer Segment") +
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")),hjust=1.1,vjust=1.2,size=3.8,color='black',font
```



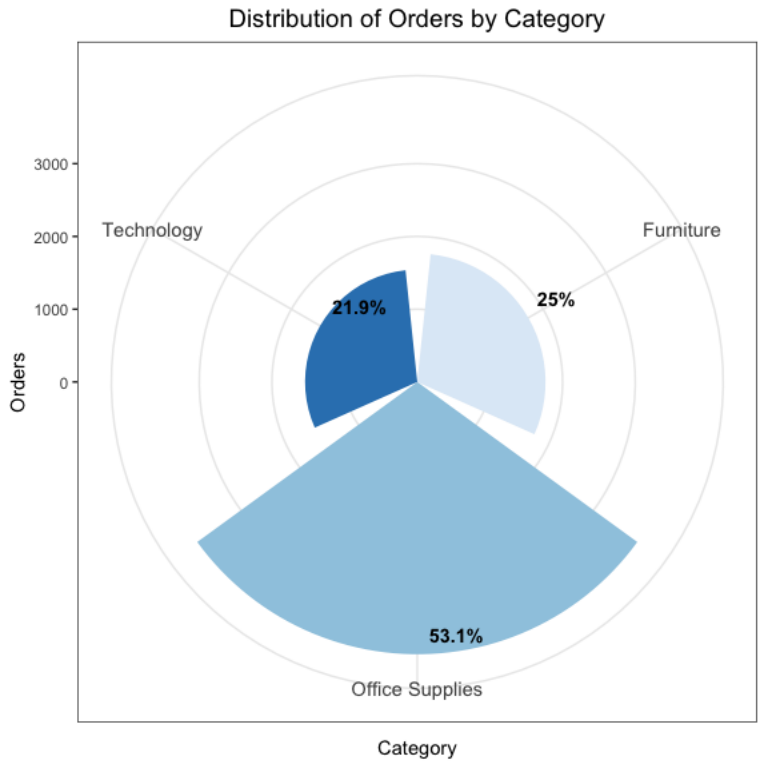

Plot for visualization of orders by region

```
#make a coxcomb plot
ggplot(supstore_df_byRegion, aes(Region, Orders, fill=Region)) +
  geom_bar(stat="identity") +
      theme_bw() +
    theme(legend.position="none",
          plot.title = element_text(size=14, hjust= 0.5),
          axis.text.x = element_text(size=11, angle=0)
          ) +
  scale_color_gradient2() +
  coord_polar() +
  labs(title="Distribution of Orders by Region") +
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")),hjust=-0.1,vjust=0.8,size=3.8,color='black',fo
```




Plot for visualization of orders by category

```
#make a coxcomb plot
ggplot(supstore_df_byCategory, aes(Category, Orders, fill=Category)) +
  geom_bar(stat="identity") +
      theme_bw() +
    theme(legend.position="none",
          plot.title = element_text(size=14, hjust= 0.5),
          axis.text.x = element_text(size=11, angle=0)
          ) +
  scale_fill_brewer(palette = "Blues") +
  coord_polar() +
  labs(title="Distribution of Orders by Category") +
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")),hjust=-0.2,vjust=-0.9,size=3.8,color='black',f
```




### 3.0.8 Analyzing the quantity

We wanted to see the quantity of products that customer frequently places online order.

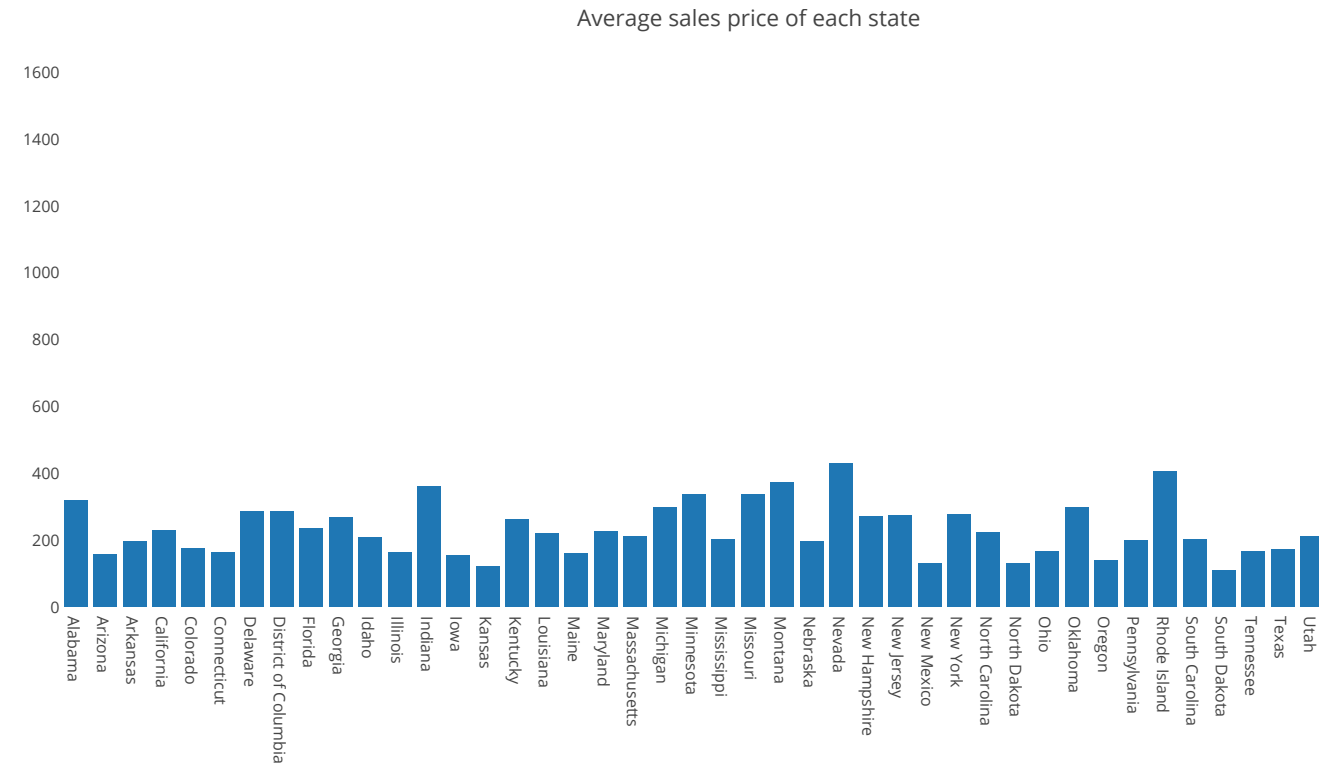
```
#Group by sub category of products
supstore_df_byQuantity =
  supstore_df %>%
  group_by(Quantity, Product.Name) %>%
  summarise(Frequency = n()) %>%
  mutate(Percentage = round( (Frequency/sum(Frequency) * 100), 1)) %>%
  arrange(desc(Frequency))

#View the results in table format
show_table(supstore_df_byQuantity, 10, isknit = TRUE)
```

| Quantity | Product.Name | Frequency | Percentage |
|---|---|---|---|
| 3 | Easy-staple paper | 16 | 0.7 |
| 2 | Staple envelope | 15 | 0.6 |
| 3 | Staple envelope | 12 | 0.5 |
| 3 | Staples | 11 | 0.5 |
| 2 | Easy-staple paper | 10 | 0.4 |
| 3 | Staple-based wall hangings | 9 | 0.4 |
| 2 | Staple remover | 8 | 0.3 |
| 2 | Staples | 8 | 0.3 |
| 5 | Staples | 8 | 0.7 |
| 4 | Easy-staple paper | 7 | 0.6 |

### 3.0.9 Average sales of orders per state

Creating new dataframe consisiting only of State and sales. We can see that Wyoming is an outlier which could be removed from the analysis for clustering.

```
df_state_sales = supstore_df %>%
  select(State,  Sales) #Creating new dataframe consisting only of State Column and Sales

df_state_avg_sales = df_state_sales %>%
  group_by(State) %>%
  summarise(mean_sales = mean(Sales)) #Creating another dataframe of State and sales

bar_avg_sales = plot_ly(y = df_state_avg_sales$mean_sales,  x = df_state_avg_sales$State,  type = "b
bar_avg_sales = bar_avg_sales%>% layout(title = 'Average sales price of each state',
                                        xaxis = list(showgrid = FALSE,  zeroline = FALSE,  showtick
                                        yaxis = list(showgrid = FALSE,  zeroline = FALSE,  showtick
bar_avg_sales #Plotting bar graph of average sales price for each state
```




### 3.0.10 Exploring the number of days required to ship orders

We can see that 49.5% of the orders shipped within 4-5 days.

```
#Group by region of products sales
supstore_df_byShippingDuration =
  supstore_df_new %>%
  group_by(Delivery_time) %>%
  summarise(Orders = n_distinct(Order.ID),
            Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))) %>%
  mutate(Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange(desc(Orders))

#View the results in table format
show_table(supstore_df_byShippingDuration, 10, isknit = TRUE)
```

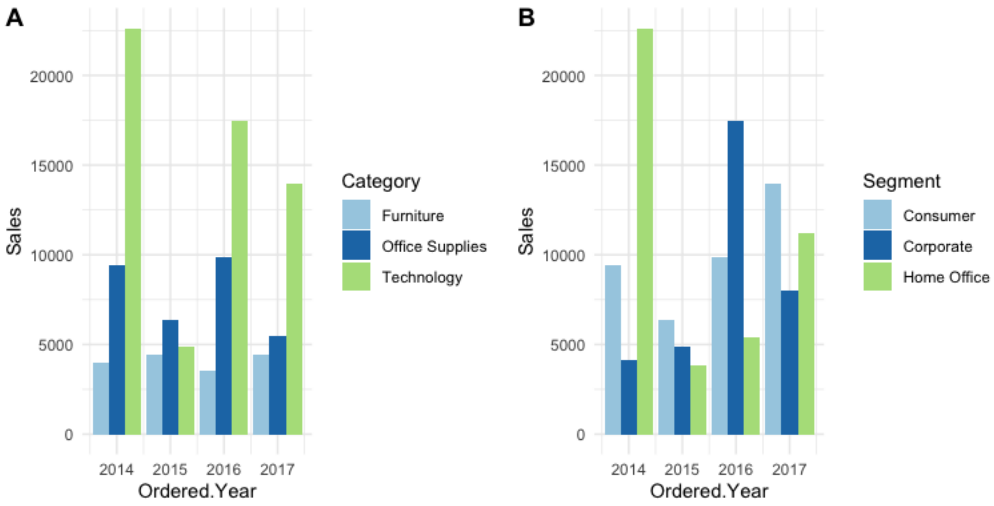
| Delivery_time | Orders | Average.profit | Average.sales | Orders.pct |
|---|---|---|---|---|
| 4 days | 1403 | 26 | 228 | 28.0 |
| 5 days | 1084 | 27 | 228 | 21.6 |
| 2 days | 675 | 40 | 276 | 13.5 |
| 6 days | 596 | 28 | 200 | 11.9 |
| 3 days | 509 | 27 | 204 | 10.2 |
| 7 days | 308 | 33 | 265 | 6.1 |
| 0 days | 252 | 30 | 241 | 5.0 |
| 1 days | 182 | 20 | 184 | 3.6 |

### 3.0.11 Analysing customer orders, saleas and profit by year on year

```
#orders grouped by year
supstore_df_byYoY =
  supstore_df_new %>%
  group_by(Sub.Category, Ordered.Year) %>%
  summarise(Orders = n(),
            Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))
            ) %>%
  mutate(Orders_pcnt = round( (Orders/sum(Orders) * 100), 1),
         Orders_YoY_rise =  round((Orders - lag(Orders))/lag(Orders) * 100),
         Sales_YoY_rise =  round((Average.sales - lag(Average.sales))/lag(Average.sales) * 100),
         Profit_YoY_rise =  round((Average.profit - lag(Average.profit))/lag(Average.profit) * 100)
         )

#View the results in table format
show_table(supstore_df_byYoY, 10, isknit = TRUE)
```

| Sub.Category | Ordered.Year | Orders | Average.profit | Average.sales | Orders_pcnt | Orders_YoY_rise |
|---|---|---|---|---|---|---|
| Accessories | 2014 | 148 | 43 | 169 | 19.1 | NA |
| Accessories | 2015 | 166 | 61 | 244 | 21.4 | 12 |
| Accessories | 2016 | 186 | 52 | 225 | 24.0 | 12 |
| Accessories | 2017 | 275 | 57 | 218 | 35.5 | 48 |
| Appliances | 2014 | 93 | 26 | 165 | 20.0 | NA |
| Appliances | 2015 | 94 | 27 | 247 | 20.2 | 1 |
| Appliances | 2016 | 114 | 47 | 229 | 24.5 | 21 |
| Appliances | 2017 | 165 | 48 | 260 | 35.4 | 45 |
| Art | 2014 | 164 | 9 | 37 | 20.6 | NA |
| Art | 2015 | 167 | 9 | 37 | 21.0 | 2 |

```
a = ggplot(data=supstore_df_new, aes(x=Ordered.Year, y=Sales, fill = Category)) +
  geom_bar(stat="identity", position=position_dodge())+
  scale_fill_brewer(palette="Paired")+
  theme_minimal()

b = ggplot(data=supstore_df_new, aes(x=Ordered.Year, y=Sales, fill = Segment)) +
  geom_bar(stat="identity", position=position_dodge())+
  scale_fill_brewer(palette="Paired")+
  theme_minimal()

ggarrange(a, b,
          labels = c("A", "B"),
          ncol = 2, nrow = 1, align = "hv")
```




### 3.0.12 Analysing sales by state and city

```
#Overall sales and profit grouped by state and city
supstore_df_sales_byCity =
  supstore_df_new %>%
  group_by(State, City) %>%
  summarise(Average.sales = round(mean(Sales)), Average.profit = round(mean(Profit))) %>%
  arrange(desc(Average.sales, Average.profit))

#View the results in table format
show_table(supstore_df_sales_byCity, 10, isknit = TRUE)
```

| State | City | Average.sales | Average.profit |
|---|---|---|---|
| New York | Jamestown | 2354 | 643 |
| Indiana | Lafayette | 1636 | 748 |
| Wyoming | Cheyenne | 1603 | 100 |
| Washington | Bellingham | 1263 | 204 |
| Missouri | Independence | 1209 | 488 |
| North Carolina | Burlington | 1153 | -536 |
| California | Burbank | 1082 | 255 |
| New York | Buffalo | 906 | 99 |

| State | City | Average.sales | Average.profit |
|---|---|---|---|
| Massachusetts | Beverly | 861 | 218 |
| Nevada | Sparks | 854 | 76 |

### 3.0.13 Analysing sales and profit by consumer segment and category

```
#Overall sales and profit grouped by consumer segment and category
supstore_df_sales_bCategory_byConsumer =
  supstore_df_new %>%
  group_by(Segment, Category) %>%
  summarise(Average.profit = round(mean(Profit)), Average.sales = round(mean(Sales))) %>%
  arrange(desc(Segment, Category))

#View the results in table format
show_table(supstore_df_sales_bCategory_byConsumer, 10, isknit = TRUE)
```

| Segment | Category | Average.profit | Average.sales |
|---|---|---|---|
| Home Office | Furniture | 11 | 337 |
| Home Office | Office Supplies | 24 | 115 |
| Home Office | Technology | 89 | 536 |
| Corporate | Furniture | 12 | 355 |
| Corporate | Office Supplies | 22 | 127 |
| Corporate | Technology | 80 | 445 |
| Consumer | Furniture | 6 | 351 |
| Consumer | Office Supplies | 18 | 116 |
| Consumer | Technology | 74 | 427 |

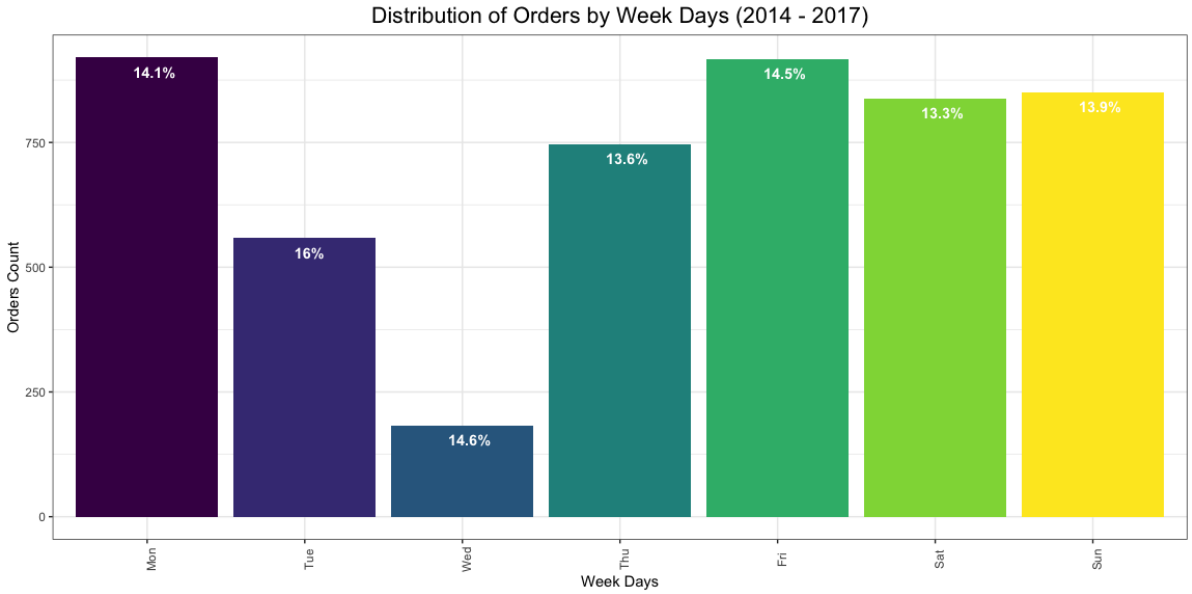
### 3.0.14 Analyze the product orders trend by week days

```
#Overall orders grouped by week days
supstore_df_orders_byweekdays =
  supstore_df_new %>%
  group_by(Ordered.Week.day) %>%
  summarise(Orders = n_distinct(Order.ID), Average.sales = round(mean(Sales))) %>%
  mutate(Avg.sales.pct = round( (Average.sales/sum(Average.sales) * 100), 1)) %>%
  arrange((Ordered.Week.day))

#View the results in table format
show_table(supstore_df_orders_byweekdays, 10, isknit = TRUE)
```

| Ordered.Week.day | Orders | Average.sales | Avg.sales.pct |
|---|---|---|---|
| Mon | 920 | 229 | 14.1 |
| Tue | 558 | 260 | 16.0 |
| Wed | 182 | 237 | 14.6 |
| Thu | 746 | 220 | 13.6 |

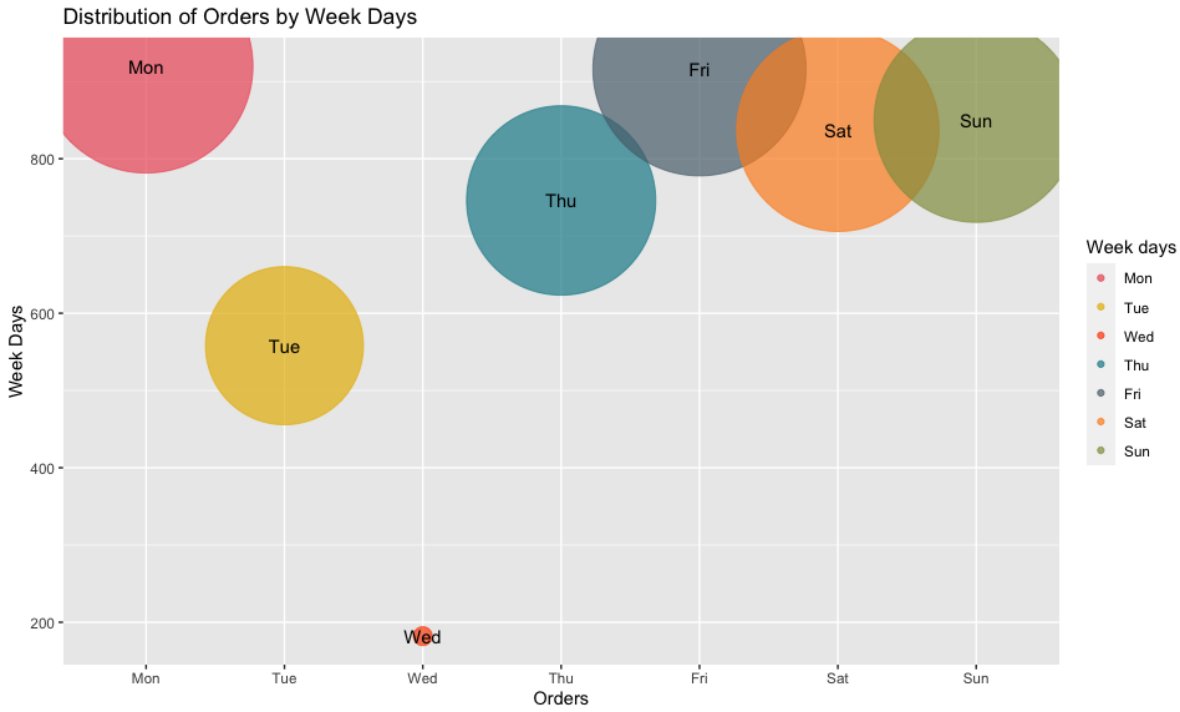
| Ordered.Week.day | Orders | Average.sales | Avg.sales.pct |
|---|---|---|---|
| Fri | 916 | 235 | 14.5 |
| Sat | 837 | 216 | 13.3 |
| Sun | 850 | 225 | 13.9 |

```
#Plot the orders by state
ggplot(supstore_df_orders_byweekdays,
       aes(Ordered.Week.day, Orders, fill=Ordered.Week.day)) +
  geom_bar(stat="identity") +
  theme_function() +
  scale_color_gradient2() +
  labs(x="Week Days",y="Orders Count",title="Distribution of Orders by Week Days (2014 - 2017)")+
  geom_text(aes(label=paste0(Avg.sales.pct,"%",sep=" ")),hjust=0.3,vjust=1.8,size=3.8,color='white',
```




Visualizing in bubbl chart

```
# Initiate a ggplot
ggplot(supstore_df_orders_byweekdays, aes(x = Ordered.Week.day, y = Orders)) +
  geom_point(aes(color = Ordered.Week.day, size = Orders), alpha = 0.7) +
  geom_text(aes(label = Ordered.Week.day) ) +
  scale_color_manual(name = "Week days", values = c("#ED5B67", "#E7B800", "#FC4E07","#168B98", "#596
  scale_size(range = c(5, 60), guide = "none") +
  labs(title = "Distribution of Orders by Week Days",
       x = "Orders",
       y = "Week Days")
```




### 3.0.15 Analyze the product orders trend by Months

```
#Overall orders grouped by week days
supstore_df_orders_byMonths =
  supstore_df_new %>%
  group_by(Ordered.Month) %>%
  summarise(Orders = n_distinct(Order.ID),
            Average.profit = round(mean(Profit)),
            Average.sales = round(mean(Sales))) %>%
  mutate(Avg.sales.pct = round( (Average.sales/sum(Average.sales) * 100), 1),
         Orders.pct = round( (Orders/sum(Orders) * 100), 1)) %>%
  arrange((Ordered.Month))

#View the results in table format
show_table(supstore_df_orders_byMonths, 10, isknit = TRUE)
```

| Ordered.Month | Orders | Average.profit | Average.sales | Avg.sales.pct | Orders.pct |
|---|---|---|---|---|---|
| Jan | 178 | 24 | 249 | 9.1 | 3.6 |
| Feb | 162 | 34 | 199 | 7.3 | 3.2 |
| Mar | 354 | 41 | 295 | 10.8 | 7.1 |
| Apr | 343 | 17 | 206 | 7.5 | 6.8 |
| May | 369 | 30 | 211 | 7.7 | 7.4 |
| Jun | 364 | 30 | 213 | 7.8 | 7.3 |
| Jul | 338 | 19 | 207 | 7.5 | 6.7 |
| Aug | 341 | 31 | 225 | 8.2 | 6.8 |
| Sep | 688 | 27 | 222 | 8.1 | 13.7 |
| Oct | 417 | 39 | 245 | 8.9 | 8.3 |

# 4 Data Modeling - Clustering Analysis

We wanted to investigate any further patterns or segments in the order transactions, profit, discount and sales. To do so, we will be employing the hierarchical and k-medoid clustering methods.

Cluster analysis is a multivariate data mining technique whose goal is to group objects based on a set of user selected characteristics. The clusters should exhibit high internal homogeneity and high external heterogeneity, meaning that when plotted geometrically, objects within clusters should be very close together and clusters will be far apart.

Due to the categorical nature of our dependent variables, time (hourly range) and area names, we used hierarchical clustering to analyse our objective. The aim was to set up a hierarchy of clusters, not only to categorize the data into a set number of clusters, but also to have a sequence of groupings at different levels.

## 4.1 Hierarchical Clustering:

The hierarchical clustering technique is one of the most popular clustering techniques for unsupervised machine learning algorithm.

The Hierarchical clustering is an alternative approach to k-means clustering for identifying groups in the dataset. It does not require us to pre-specify the number of clusters to be generated as is required by the kmeans approach. Furthermore, hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram.

Hierarchical clustering can be divided into two main types: agglomerative and divisive. Agglomerative clustering: It's also known as AGNES (Agglomerative Nesting). It works in a bottom-up manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are members of just one single big cluster (root).

Divisive hierarchical clustering: It's also known as DIANA (Divise Analysis) and it works in a top-down manner. The algorithm is an inverse order of AGNES. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.

Agglomerative clustering is good at identifying small clusters. Divisive hierarchical clustering is good at identifying large clusters. In our implementation we have followed both methods to determine right clustering group.

Clustering Distance Measures: The classification of observations into groups requires some methods for computing the distance or the (dis)similarity between each pair of observations. The result of this computation is known as a dissimilarity or distance matrix. There are many methods to calculate this distance information.

In our implementation we have used the Euclidean method for computing the distance (Euclidean Method: It is the distance two points in either the plane or 3-dimensional space measures the length of a segment connecting the two points).

Below ordered steps are followed to accomplish the objective. 1. Calculate the distance matrix for hierarchical clustering 2. Choose a linkage method (single, complete, average and ward) to determine dissimilarity between two clusters of observations by correlation distant coefficient. i. Method of single linkage or nearest neighbour. Proximity between two clusters is the proximity between their two closest objects. ii. Method of complete linkage or farthest neighbour. Proximity between two clusters is the proximity between their two most distant objects iii. Method of between-group average linkage. Proximity between two clusters is the arithmetic mean of all the proximities between the objects of one, on one side, and the objects of the other, on the other side. iv. Ward's method, or minimal increase of sum-of-squares. Proximity between two clusters is the magnitude by which the summed square in their joint cluster will be greater than the

combined summed square in these two clusters. 3. Plot the data as a dendrogram 4. Plot the data using fviz_cluster function from the factoextra package to visualize the result in a scatter plot

### 4.1.1 Data Transformation for Hierarchical Clustering

To perform a cluster analysis in R, generally, the data should be prepared as follows: 1. Rows are observations (individuals) and columns are variables 2. Any missing value in the data must be removed or estimated. 3. The data must be standardized (i.e., scaled) to make variables comparable. Recall that, standardization consists of transforming the variables such that they have mean zero and standard deviation one.

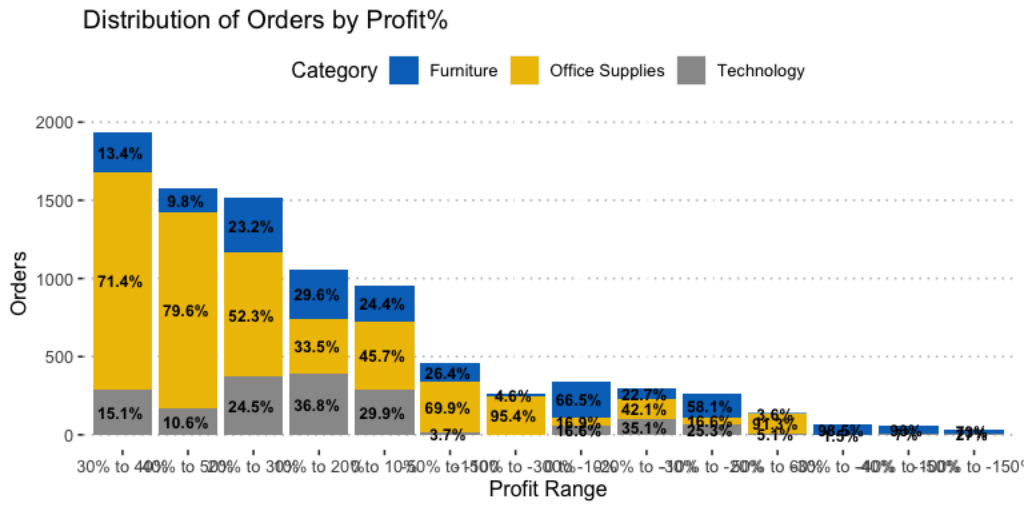
```
#Binning or adding categorical values based on ranges in continous data
setDT(supstore_df_new)[Profit_pct > 100, Profit_pct_range := "> 100%"]
supstore_df_new[Profit_pct >= 0 & Profit_pct <10, Profit_pct_range := "0 to 10%"]
supstore_df_new[Profit_pct >= 10 & Profit_pct <20, Profit_pct_range := "10% to 20%"]
supstore_df_new[Profit_pct >= 20 & Profit_pct <30, Profit_pct_range := "20% to 30%"]
supstore_df_new[Profit_pct >= 30 & Profit_pct <40, Profit_pct_range := "30% to 40%"]
supstore_df_new[Profit_pct >= 40 & Profit_pct <50, Profit_pct_range := "40% to 50%"]
supstore_df_new[Profit_pct >= 50 & Profit_pct <60, Profit_pct_range := "50% to 60%"]
supstore_df_new[Profit_pct <= 0 & Profit_pct > -10, Profit_pct_range := "0 to -10%"]
supstore_df_new[Profit_pct <= -10 & Profit_pct > -20, Profit_pct_range := "-10% to -20%"]
supstore_df_new[Profit_pct <= -20 & Profit_pct > -30, Profit_pct_range := "-20% to -30%"]
supstore_df_new[Profit_pct <= -30 & Profit_pct > -40, Profit_pct_range := "-30% to -40%"]
supstore_df_new[Profit_pct <= -40 & Profit_pct > -50, Profit_pct_range := "-40% to -50%"]
supstore_df_new[Profit_pct <= -50 & Profit_pct > -100, Profit_pct_range := "-50% to -100"]
supstore_df_new[Profit_pct <= -100 & Profit_pct > -150, Profit_pct_range := "-100% to -150%"]
supstore_df_new[Profit_pct <= -150 & Profit_pct > -300, Profit_pct_range := "-150% to -300%"]

#Numeric conversion
supstore_df_new$Profit_pct_range = as.factor(supstore_df_new$Profit_pct_range)
supstore_df_new$Profit_pct = as.numeric(supstore_df_new$Profit_pct)

#Hierarchical Clustering Dataset by Area Name
hcus_by_Profit_pct =
supstore_df_new %>%
  select(Profit_pct_range, Profit_pct, Sales, Quantity, Discount, Customer.ID, Sub.Category, Segment
  arrange(Profit_pct_range)

hcus_by_Profit_pct2 =
hcus_by_Profit_pct %>%
  #filter(Victim.Age <= 85) %>%
  select(Profit_pct_range, Sales, Quantity, Discount, Customer.ID, Sub.Category, Segment) %>%
  group_by(Profit_pct_range) %>%
  summarise(total_orders = n(), avg.sales = round(mean(Sales)), avg.Discount = round(mean(Discount))
  arrange(desc(Profit_pct_range))

#Converting Area.Name column as rownames
row.names(hcus_by_Profit_pct2) = hcus_by_Profit_pct2$Profit_pct_range
hcus_by_Profit_pct2[1] = NULL

# Applying the transformation with scaling function to centers and/or scales the columns of a numeri
hcus_by_Profit_pct3 = scale(hcus_by_Profit_pct2)
```

### 4.1.2 Analyze the product by profit margins

```
#Plot the orders by state
library(viridis)
```

```
## Loading required package: viridisLite
```

```
ggplot(supstore_df_byProfit,
       aes(reorder(Profit_pct_range, -Orders), Orders, fill=Category)) +
  geom_bar(position="stack", stat="identity") +
  scale_fill_manual(values = c("#0073C2FF", "#EFC000FF", "#999999"))+
  theme_pubclean() +
  labs(x="Profit Range", y="Orders", title="Distribution of Orders by Profit%") +
  geom_text(aes(label=paste0(Orders.pct,"%",sep=" ")), size=3.2, color='black', fontface='bold', pos
```




## 4.2 Hierarchical Clustering Analysis - Unsupervised learning - by Profit Range

Implementing Hierarchical Clustering using Agglomerative Clustering using four linkage methods:

We can perform agglomerative Hierarchical Clustering with hclust in R. First we compute the dissimilarity values with dist 'Euclidean Method' and then feed these values into hclust and specify the agglomeration method to be used (i.e. "complete", "average", "single", "ward.D"). We can then plot the dendrogram.

We can see the differences these approaches from above dendrograms.

We have used agnes function to get the agglomerative coefficient, which measures the amount of clustering structure found (values closer to 1 suggest strong clustering structure).

Following is a custom function to loop all four linkage methods and generate the coefficient to determine the strongest clustering method using ANGES (Agglomerative Clustering in R).

```
# function to compute coefficient for different agglomerative clustering methods
methods = c( "average", "single", "complete", "ward")
names(methods) = c( "average", "single", "complete", "ward")

ac = function(x) {
  agnes(hcus_by_Profit_pct3, method = x)$ac
}

map_dbl(methods, ac)
```

```
##   average    single  complete      ward
## 0.8888035 0.8635033 0.9199393 0.9354247
```

Here we see that Ward's method identifies the strongest clustering structure as distant coefficient is 0.93.

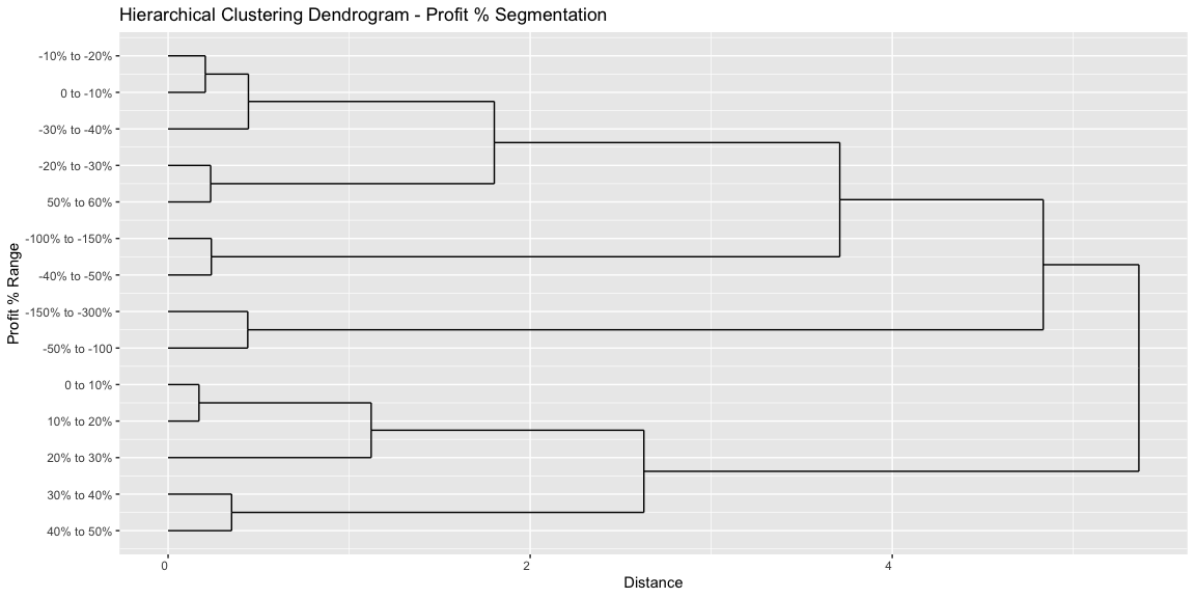
We will continue Agglomerative Hierarchical Clustering using ward method to plot the Dendrogram and scatter plot.

```
#printf("Here we see that Ward's method identifies the strongest clustering structure")

# We will continue Agglomerative Hierarchical Clustering using ward method
# Ward's method
d2 = dist(hcus_by_Profit_pct3, method = "euclidean")

hc5 = hclust(d2, method = "ward.D2")

# Cut tree into 4 groups
sub_grp = cutree(hc5, k = 5)

# Rotate the plot and remove default theme
#install.packages("ggdendro")
library("ggdendro")
```

```
##
## Attaching package: 'ggdendro'
```

```
## The following object is masked from 'package:dendextend':
##
##     theme_dendro
```

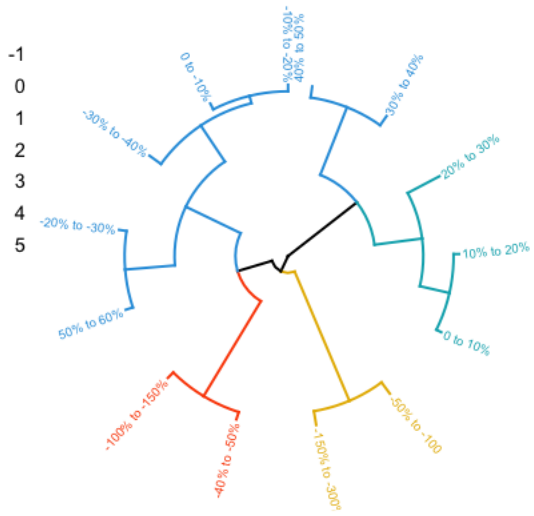
```
ggdendrogram(hc5, rotate = TRUE, theme_dendro = FALSE) +
  labs(title = "Hierarchical Clustering Dendrogram - Profit % Segmentation",
       x = "Profit % Range",
       y = "Distance")
```




Here is the enhanced Visualization Of Dendrogram. Draws easily beautiful dendrograms using either ggplot2. Provides also an option for drawing a circular dendrogram tree.

```
library(factoextra)
```

```
#the arguments main, sub, xlab, ylab to change plot titles as follow:
fviz_dend(hc5, k = 5,                 # Cut in four groups
          cex = 0.5,                 # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE,  # color labels by groups
          horiz = TRUE,              # Flip the chart
          ggtheme = theme_classic(),     # Change theme
          ylab = "Height",
          xlab = "Profit % Range",
          main = "Hierarchical Clustering Dendrogram - Profit % Segmentation",
          type = c("circular") #type = c("rectangle", "circular", "phylogenic"),

          )
```




## 4.3 Data Modeling - K-Medoids Clustering

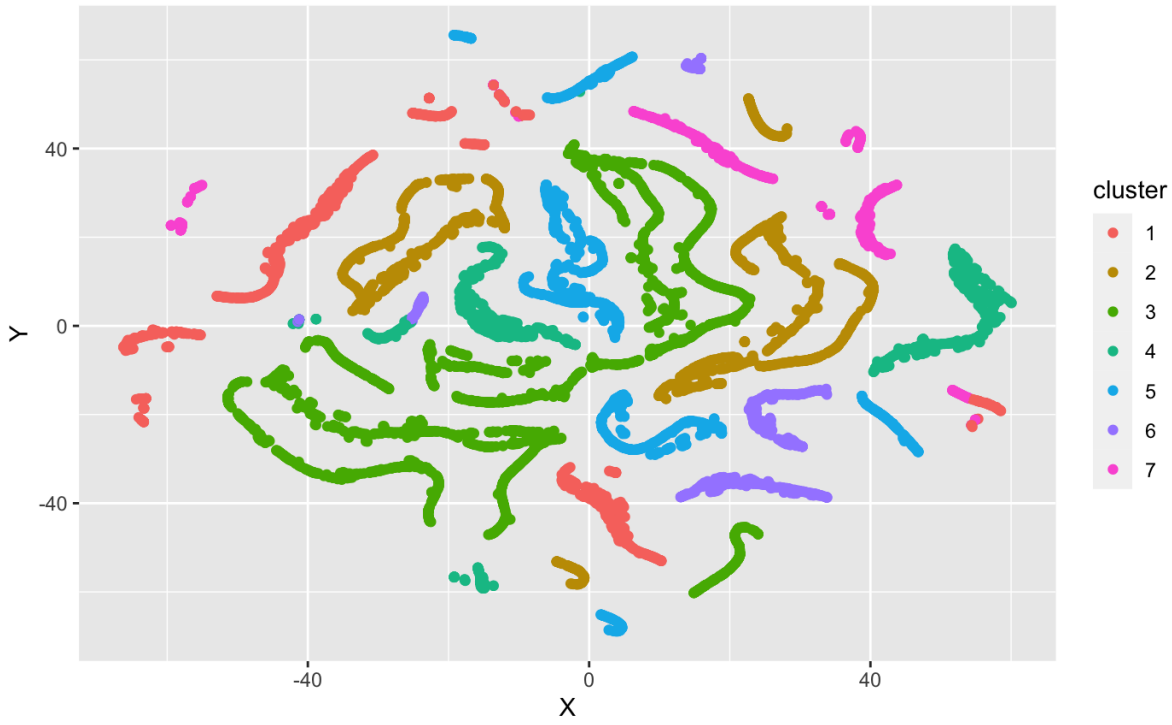
An alternative approach to hierarchical clustering is k-medoids clustering for identifying groups in the dataset. It does not require us to pre-specify the number of clusters to be generated as the calculation is done by the Sillhouette width method. We will first measure the gower distance and then then the sillhouette width, based on which we will decide how many clusters are optimal. Then we will attempt the clustering.

As suggested by the Silhouette width, we have 7 clusters as the most optimal number of clusters.

Now we will try to plot the 7 clusters

However, plotting the clusters, shows that the clusters formed are not ideal. Therefore, instead of creating 7 clusters, we would be creating 3 clusters instead.

```
knitr::include_graphics("/Users/deenuy/Documents/Google Drive/Personal/CSDA1050-Capstone-Project-Rep
```




After the 3 clusters have been created and summarized, we would attempt to visualize the clusters.

The 3 clusters have been plotted, however, the results were not optimal in determining the cluster groups due to no clear delineation of clusters/grouping.

In the earlier steps we have analyzed clustering models (Hierarchical and K-medoids) to dertermine any segments from given data and feed to appriori algorithm. But we have observed that adding segmentation is to appriori is condensing the outcome of recommendation list due to size of dataset which is less than 10000 transactions approx. Hence we have decided to exclude the segmentation from our algorithm implementation and applied appriori technique on whole dataset.

## 4.4 Data Modeling: Associative Rule Mining (Apriori)

The next step is to convert the data frame into basket format, based on the Order_ID. New Dataframe df_itemList created below. The `ddply()` function checks the names of the products and pivots them into one single line separating them by commas.

Here we select the `apriori algorithm` as the modeling technique for our `recommendation analysis`. At this point, after much research, we are assuming that the algorithm will work well with our updated data frame. We have pivoted the table to show the full order details on one line with the order no. and all associated products purchased separated by commas.

The data doesn't have any missing value which should facilitate the analysis. To increase the efficiency of our analysis we also specify the support, confidence, and lift of our analysis.

The `support has been set at 0.001` to measure the number of transactions in which an itemset appears at least `0.1%` of the time.

- Ex. Supp(x) = number of transactions in which x appears/Total number of transactions

The `confidence is set at 0.5` to measure the number of transactions in which the two or more items appear at least `50%` of the time.

- Ex. Conf(X à Y) = Supp(X ∪ Y)/Supp(X)

The `lift` determined by greater than 1 and higher value to show how likely an item is purchased when a second item is purchased.

- Ex. Lift(X à Y) = Supp(X ∪ Y)/Supp(X) * Supp(Y)

For this project, we have implemented apriori in two steps, one for frequent itemset generation with support (0.025) and confidence (50%), and second step to generate the candidate rules from each frequent itemset. The candidate rule provides the lhs (selected dataset item) and rhs (recommended corresponding dataset item list) components to prompt customers with a recommended list of items on the selected one.

Finally, we run the algorithm.

### 4.4.1 Assess the model

Once the algorithm implemented, we used the `inspect()` function to manipulate the arguments to verify if we can get to a more accurate recommendation.

Now, we will concatenate the product names as per the order number.

Manual testing with input “Acco Hanging Data Binders” in LHS and get RHS as recommended frequent item

### 4.4.2 Visualization for Apriori Algorithm

Now, ploting the vizualiztions. These functions can be accessed form the arulesViz library.

Inputs for Shiny Application Dashboard

### 4.4.3 Shiny Dashboard Implementation

Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R.

We have developed the Dashboard serving two functions: * A descriptive dashboard to simulate the performance of various segments * A recommender system to show the consumer preferences and purchasing behaviours

The dashboard is hosted on a secured Shiny cloud platform to simplify integration, improve agility, and ease the burden of implementation for Superstore IT resources.

Calculating execution run time