#Lets import the necessary libraries

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.3     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ tidyr::expand() masks Matrix::expand()
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ tidyr::pack()   masks Matrix::pack()
## ✖ dplyr::recode() masks arules::recode()
## ✖ tidyr::unpack() masks Matrix::unpack()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dply
r:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(RColorBrewer)
library(plotly)
```

```
##
## Attaching package: 'plotly'
##
## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise
##
## The following object is masked from 'package:ggplot2':
##
##     last_plot
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following object is masked from 'package:graphics':
##
##     layout
```

#Lets import the data

```
data <- read.csv("/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Data-Science-Tasks/al
l_data.csv")
```

#Data Cleaning

```
#Checking for missing values
any(is.na(data))
```

```
## [1] FALSE
```

Get the count of missing values in each column

```
missing_counts <- colSums(is.na(data))
missing_counts
```

```
##         Order.ID          Product Quantity.Ordered       Price.Each 
##                0                0                0                0 
##       Order.Date Purchase.Address 
##                0                0
```

deleting all null values

```
data <- na.omit(data)
```

Checking for duplicate values

```
any(duplicated(data))
```

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

Removing duplicate data using the unique function

```
data <- unique(data)
```

#Performing Exploratory data Analysis

Structure of the data

```
str(data)
```

```
## 'data.frame':    185688 obs. of  6 variables:
##  $ Order.ID        : chr  "194095" "194096" "194097" "194098" ...
##  $ Product         : chr  "Wired Headphones" "AA Batteries (4-pack)" "27in FHD Mon
itor" "Wired Headphones" ...
##  $ Quantity.Ordered: chr  "1" "1" "1" "1" ...
##  $ Price.Each      : chr  "11.99" "3.84" "149.99" "11.99" ...
##  $ Order.Date      : chr  "05/16/19 17:14" "05/19/19 14:43" "05/24/19 11:36" "05/0
2/19 20:40" ...
##  $ Purchase.Address: chr  "669 2nd St, New York City, NY 10001" "844 Walnut St, Da
llas, TX 75001" "164 Madison St, New York City, NY 10001" "622 Meadow St, Dallas, TX 
75001" ...
```

Descriptive Statistics of the data

```
summary(data)
```

```
##    Order.ID           Product          Quantity.Ordered    Price.Each       
##  Length:185688      Length:185688      Length:185688      Length:185688     
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##   Order.Date        Purchase.Address  
##  Length:185688      Length:185688     
##  Class :character   Class :character  
##  Mode  :character   Mode  :character  
```

Converting columns to numeric columns

```
data$Price.Each <- as.numeric(data$Price.Each)
```

```
## Warning: NAs introduced by coercion
```

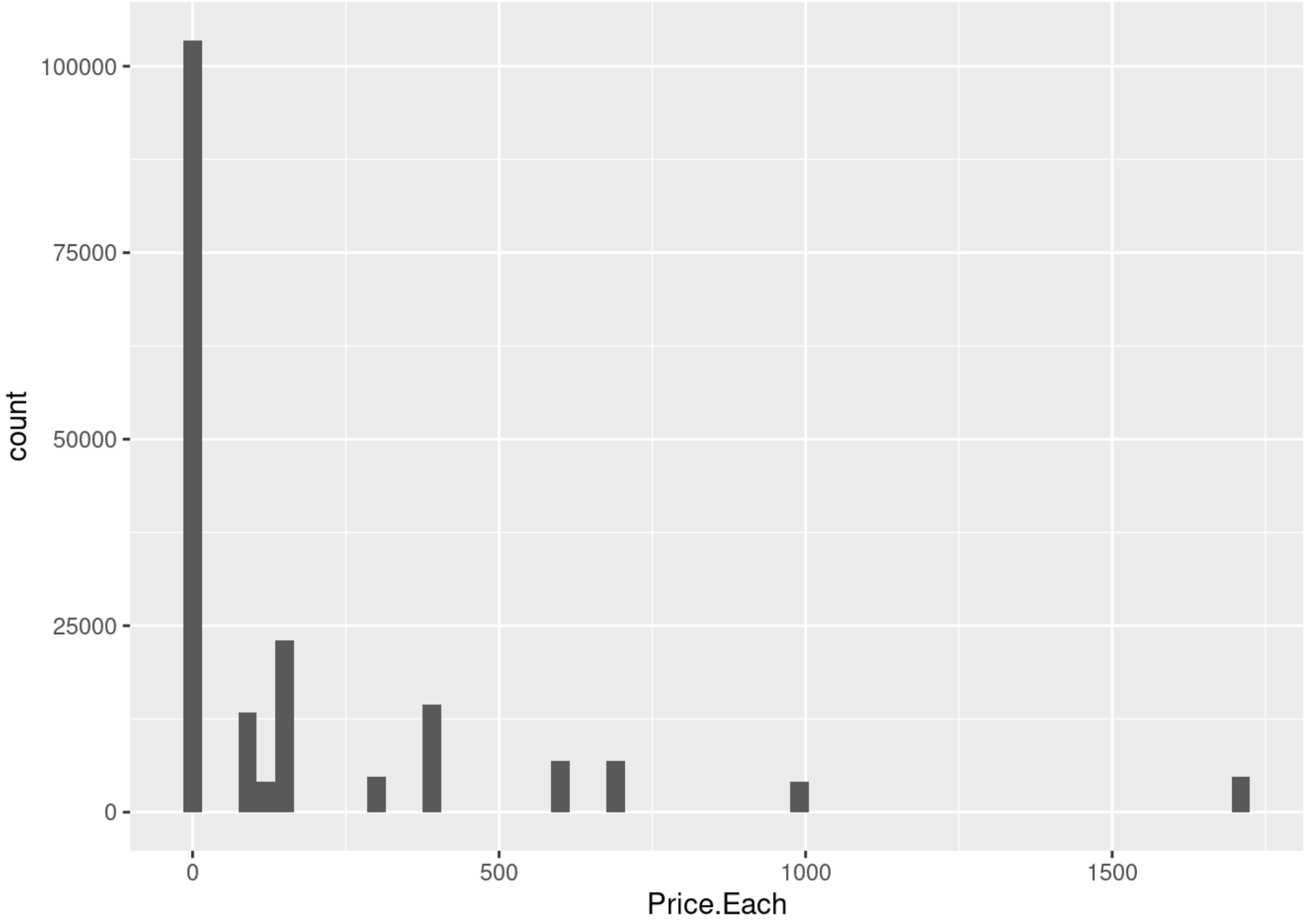
```
data$Quantity.Ordered <- as.integer(data$Quantity.Ordered)
```

```
## Warning: NAs introduced by coercion
```

Drawing histograms for the numeric variables in the dataset

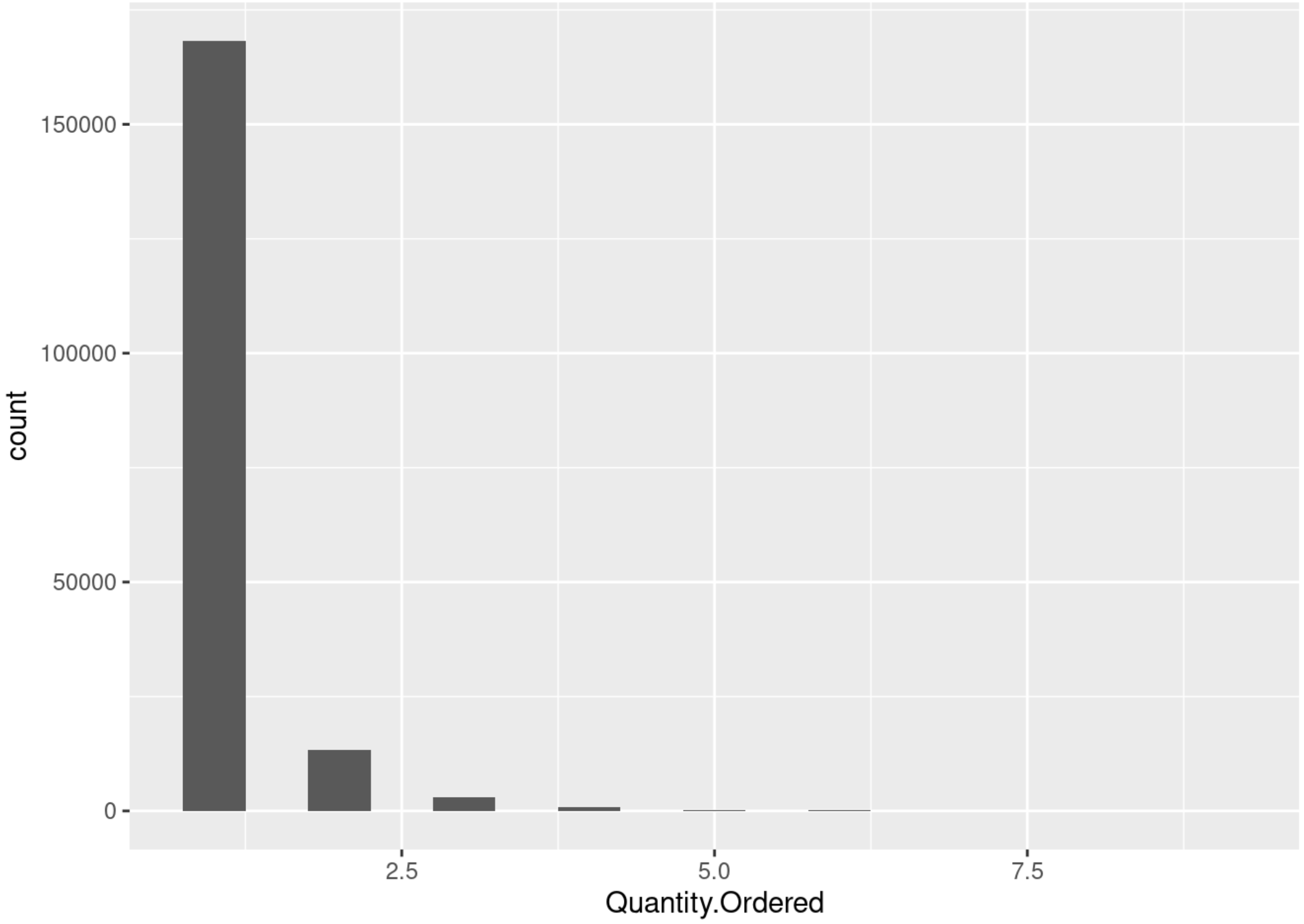
```
  ggplot(data,aes(x= Price.Each  )) +
  geom_histogram(binwidth = 30)
```

```
## Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
```



```
ggplot(data,aes(x= Quantity.Ordered  )) +
    geom_histogram(binwidth = 0.5)
```

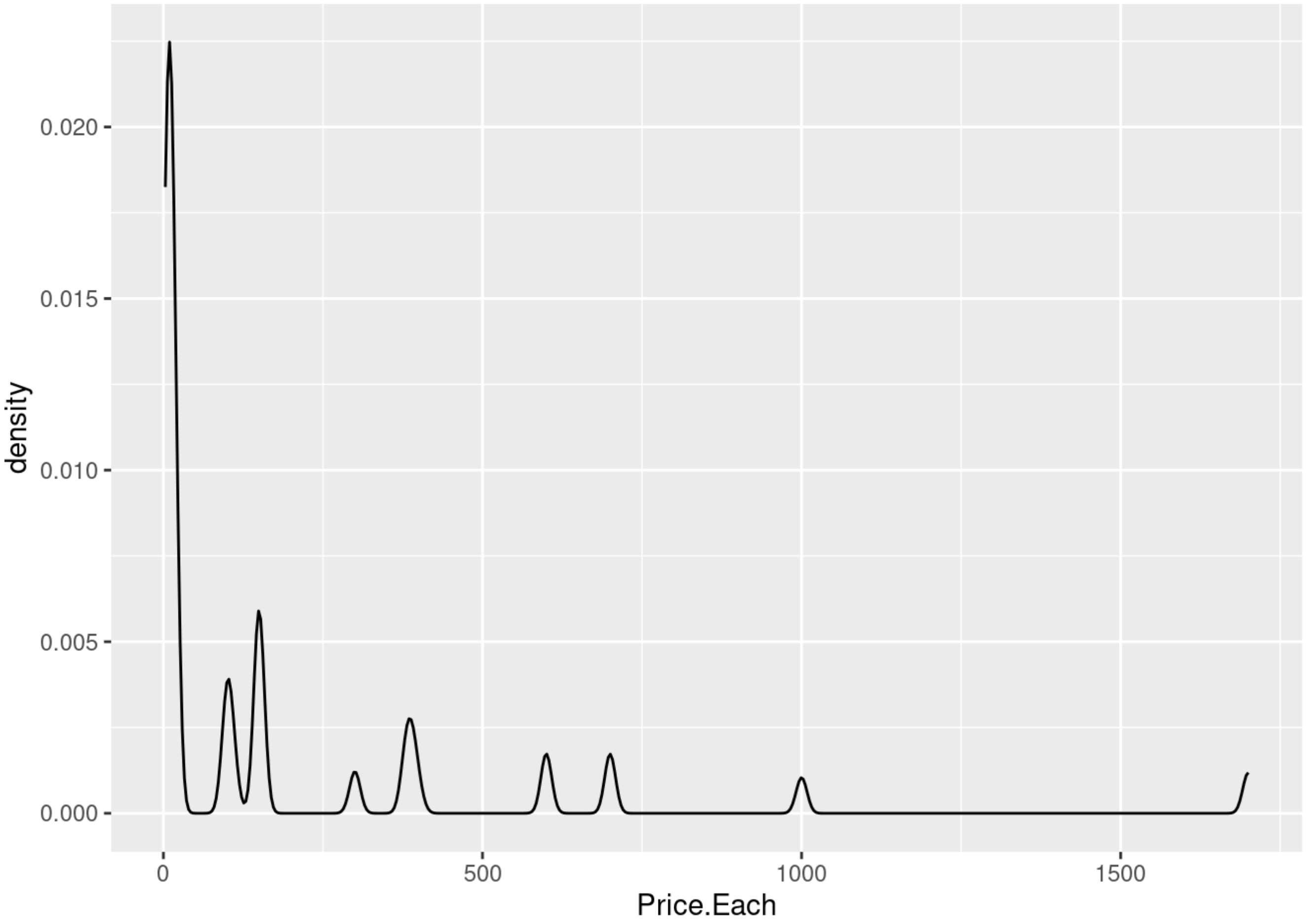
```
## Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
```



Drawing the density plots for the numeric variables, First for Price.Each

```
ggplot(data,aes(x= Price.Each  )) + geom_density()
```

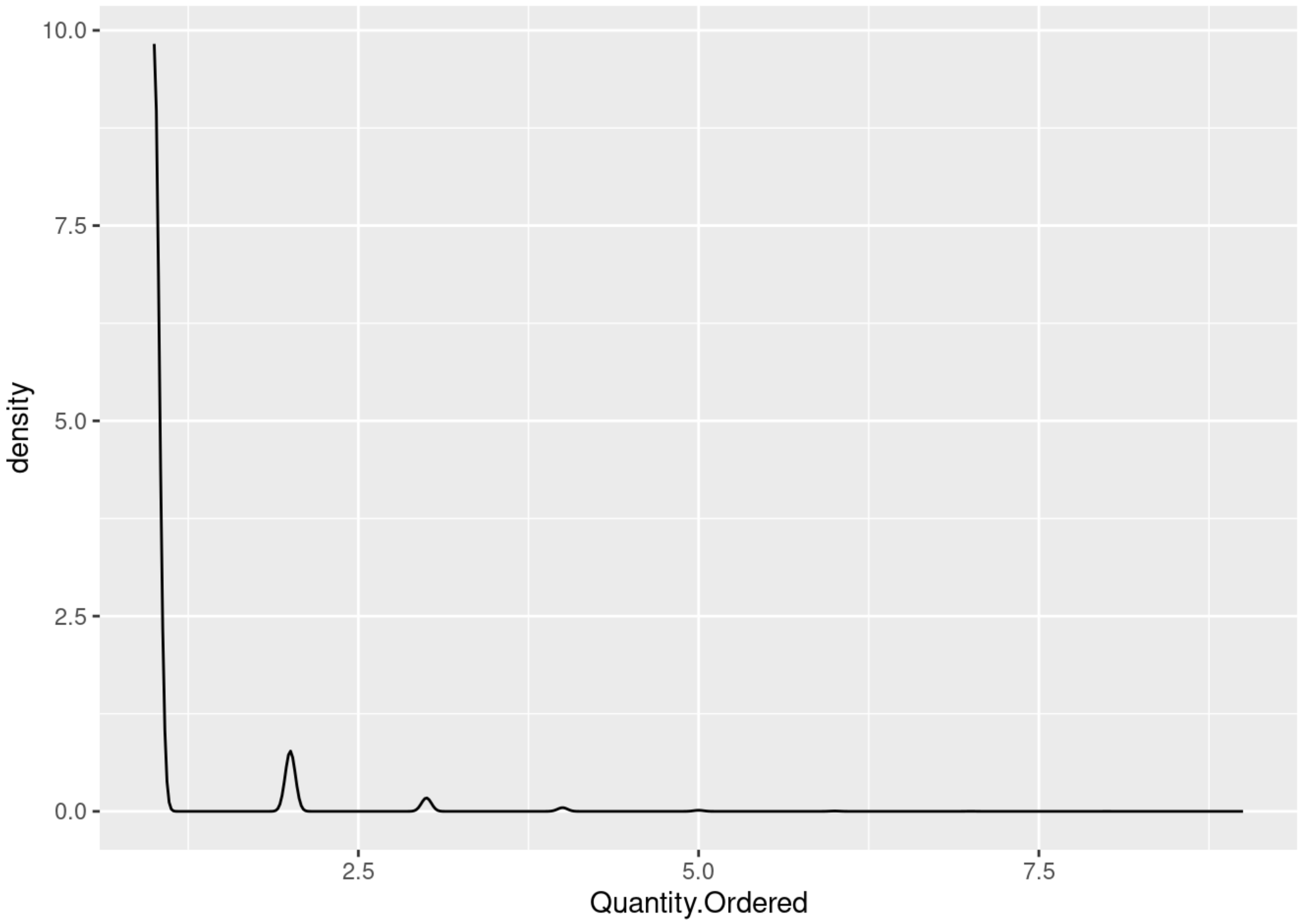
```
## Warning: Removed 2 rows containing non-finite values (`stat_density()`).
```



Density Plot for Quantity Ordered

```
ggplot(data,aes(x= Quantity.Ordered  )) + geom_density()
```

```
## Warning: Removed 2 rows containing non-finite values (`stat_density()`).
```



Data Preparation for Market Basket Analysis

```
transactionData <- ddply(data,c("Order.ID","Order.Date"),
                         function(df1)paste(df1$Product,
                                            collapse = ","))
```

Viewing the transaction data

```
head(transactionData)
```

```
##   Order.ID     Order.Date                     V1
## 1   194095 05/16/19 17:14       Wired Headphones
## 2   194096 05/19/19 14:43  AA Batteries (4-pack)
## 3   194097 05/24/19 11:36       27in FHD Monitor
## 4   194098 05/02/19 20:40       Wired Headphones
## 5   194099 05/11/19 22:55 AAA Batteries (4-pack)
## 6   194100 05/10/19 19:44                 iPhone
```

This format for transaction data is called the basket format. Next, you have to store this transaction data into a .csv (Comma Separated Values) file. For this, write.csv()

```
write.csv(transactionData,"/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Data-Science
-Tasks/transaction_data.csv", quote = FALSE, row.names = FALSE)
```

loading this transaction data into an object of the transaction class using read.transactions

```
transactions <- read.transactions("/home/owekitiibwa/Desktop/dataAnalysis/Pandas-Data
-Science-Tasks/transaction_data.csv", format = 'basket', sep=',')
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

Viewing the transaction object

```
transactions
```

```
## transactions in sparse format with
##  178440 transactions (rows) and
##  320857 items (columns)
```

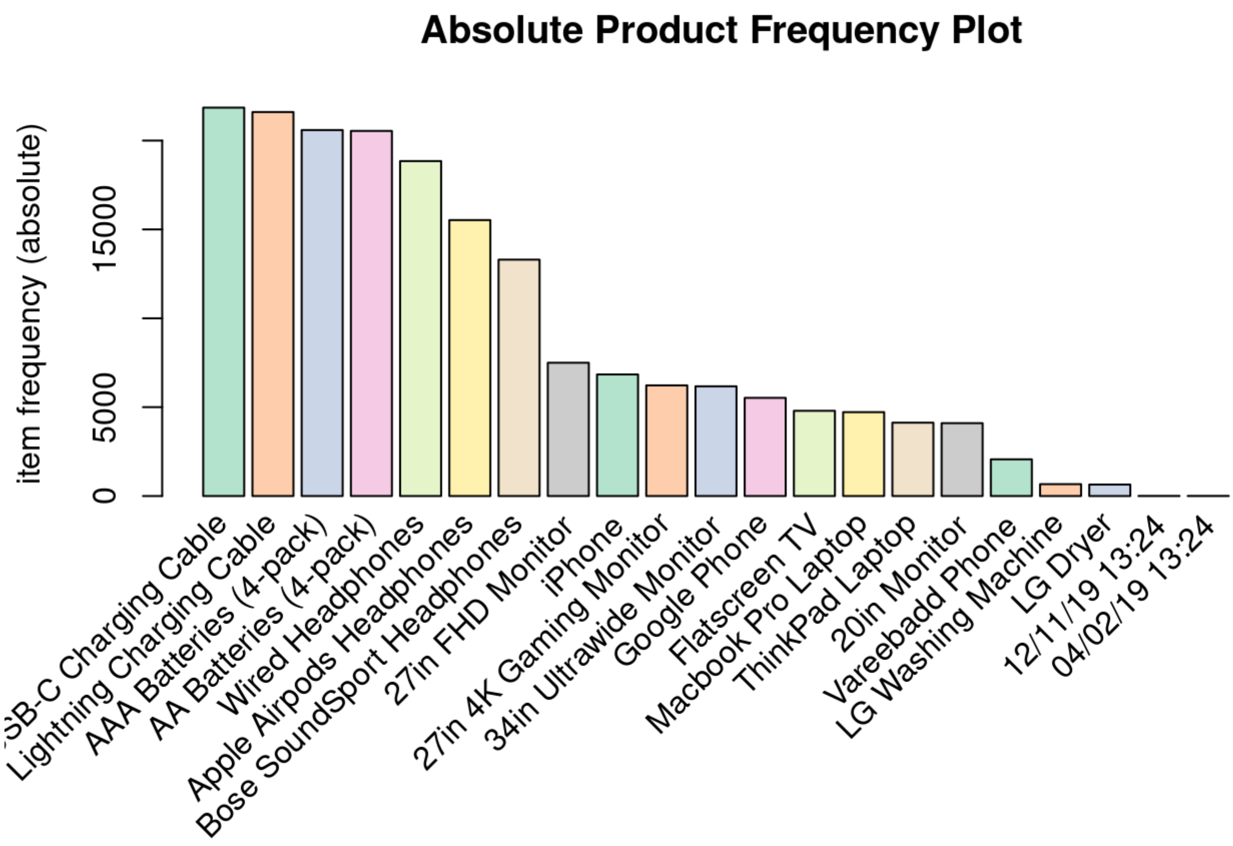
Summary of the transactions

```
summary(transactions)
```

```
## transactions as itemMatrix in sparse format with
##  178440 rows (elements/itemsets/transactions) and
##  320857 columns (items) and a density of 9.475698e-06
##
## most frequent items:
##     USB-C Charging Cable Lightning Charging Cable   AAA Batteries (4-pack)
##                    21855                    21604                    20593
##    AA Batteries (4-pack)         Wired Headphones                  (Other)
##                    20542                    18847                   439078
##
## element (itemset/transaction) length distribution:
## sizes
##      0      3      4      5      6      7
##      1 171607   6479    337     15      1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    3.00    3.00    3.04    3.00    7.00
##
## includes extended item information - examples:
##           labels
## 1 01/01/19 03:07
## 2 01/01/19 03:40
## 3 01/01/19 04:56
```

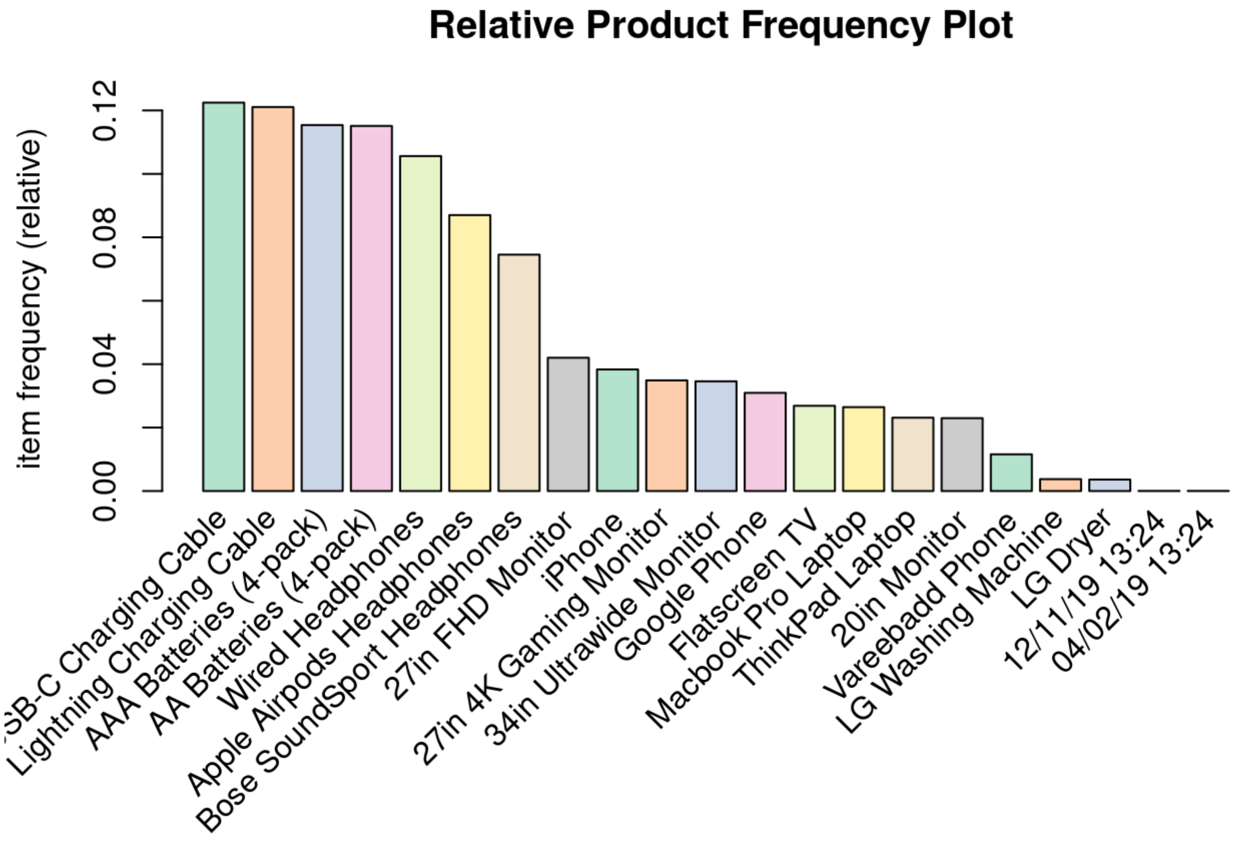
Drawing an Absolute Item Frequency Plot

```
itemFrequencyPlot(transactions,topN=21,type="absolute",col=brewer.pal(8,'Pastel2'), m
ain=" Absolute Product Frequency Plot")
```



Drawing an Relative Frequency Plot

```
itemFrequencyPlot(transactions,topN=21,type="relative",col=brewer.pal(8,'Pastel2'), m
ain=" Relative Product Frequency Plot")
```



#Generating Rules! Next step is to mine the rules using the APRIORI algorithm. The function apriori() is from package arules.

```
association_rules <- apriori(transactions, parameter = list(supp=0.0001, conf=0.4,max
len=10))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5   1e-04      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 17
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[320857 item(s), 178440 transaction(s)] done [0.48s].
## sorting and recoding items ... [19 item(s)] done [0.04s].
## creating transaction tree ... done [0.07s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [3 rule(s)] done [0.00s].
## creating S4 object  ... done [0.08s].
```

#Inspecting the rules Lets print number of rules

```
print(association_rules)
```

```
## set of 3 rules
```

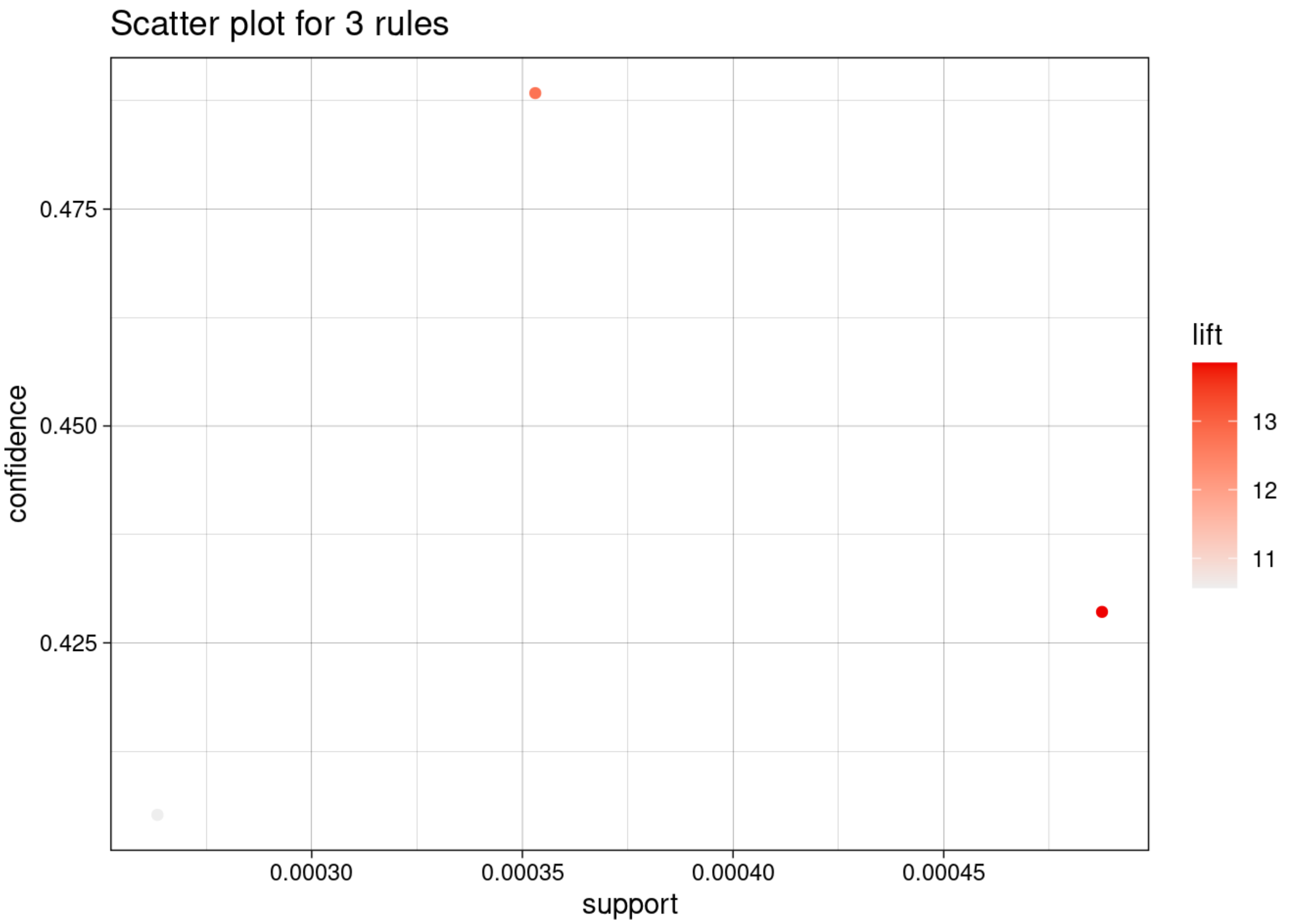
Displaying the rules

```
inspect(association_rules)
```

```
##     lhs                             rhs                 support confidence     cove
rage     lift count
## [1] {USB-C Charging Cable,
##      Wired Headphones}          => {Google Phone} 0.0004875588  0.4285714 0.001137
6373 13.84902    87
## [2] {Apple Airpods Headphones,
##      Lightning Charging Cable}  => {iPhone}       0.0002633939  0.4051724 0.000650
0785 10.57002    47
## [3] {Lightning Charging Cable,
##      Wired Headphones}          => {iPhone}       0.0003530599  0.4883721 0.000722
9321 12.74051    63
```

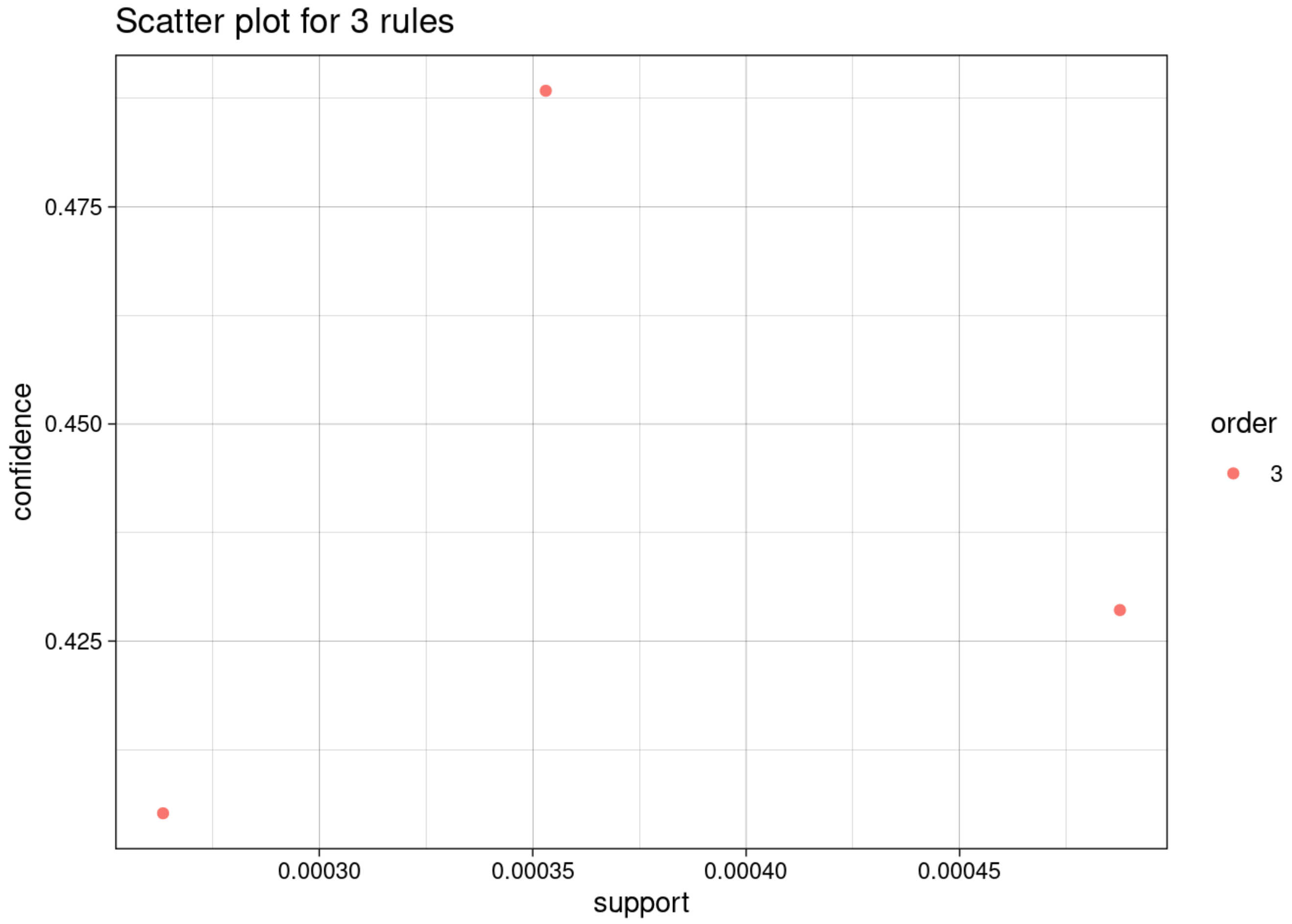
#Lets Visualize the Rules

```
plot(association_rules)
```



Lets create a two key plot

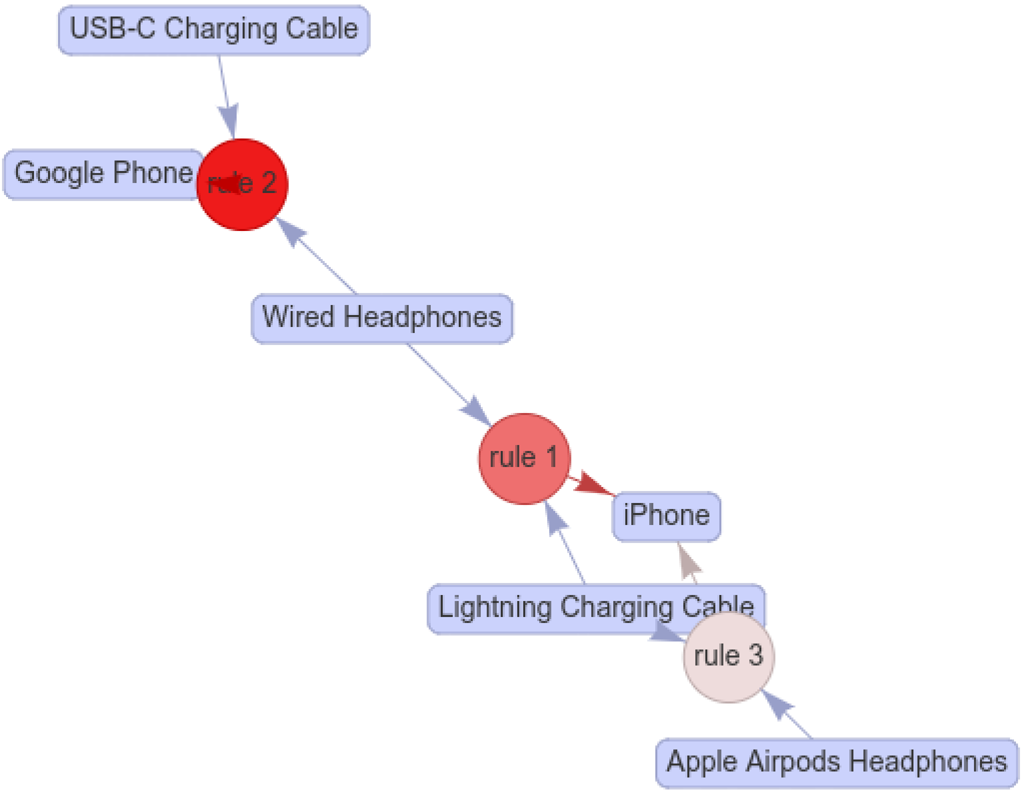
```
plot(association_rules,method="two-key plot")
```



Graph-Based Visualizations for association rules

```
top10subRules <- head(association_rules, by = "confidence")
plot(top10subRules, method = "graph",  engine = "htmlwidget")
```

Select by id


USB-C Charging Cable
Google Phone
rule 2
Wired Headphones
rule 1
iPhone
Lightning Charging Cable
rule 3
Apple Airpods Headphones