CRM with Classification

Marc Kullmann

7/21/2019

Contents

1 A Customer Relationship Management Analysis with Classification 1
1.1 Retrieving the Data . . . 1
1.2 Recoding . . . 2
1.3 Missing Value Treatment I . . . 3
1.4 Outlier Detection and Treatment . . . 4
1.4.1 Price Outliers . . . 4
1.4.2 Quantity Outliers . . . 8
1.5 Descriptive Analysis . . . 11
1.5.1 Statistics . . . 11
1.5.2 Graphics . . . 12
1.6 CRM - Recency Frequency Monetary Analysis . . . 15
1.6.1 Implementing the 80/20 Pareto Principle . . . 16
1.6.2 Customer Relation By Country . . . 17
1.7 Customer Segmentation with Hierarchical Clustering . . . 17

1 A Customer Relationship Management Analysis with Classification

1.1 Retrieving the Data

```
myurl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx"
download.file(url=myurl, destfile="online_retail.xlsx", mode="wb")
Online_Retail <- read_excel("online_retail.xlsx")

df <- as_tibble(Online_Retail)

head(df)
```

```
## # A tibble: 6 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate         UnitPrice
##   <chr>     <chr>     <chr>          <dbl> <dttm>                  <dbl>
## 1 536365    85123A    WHITE HANG~        6 2010-12-01 08:26:00      2.55
## 2 536365    71053     WHITE META~        6 2010-12-01 08:26:00      3.39
## 3 536365    84406B    CREAM CUPI~        8 2010-12-01 08:26:00      2.75
## 4 536365    84029G    KNITTED UN~        6 2010-12-01 08:26:00      3.39
## 5 536365    84029E    RED WOOLLY~        6 2010-12-01 08:26:00      3.39
## 6 536365    22752     SET 7 BABU~        2 2010-12-01 08:26:00      7.65
## # ... with 2 more variables: CustomerID <dbl>, Country <chr>
```

```
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    541909 obs. of  8 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
```

```
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPID HEARTS C
##  $ Quantity   : num  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct, format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num  17850 17850 17850 17850 17850 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

1.2 Recoding

```
df <- df %>%
  mutate(uniqueID = row_number(),
         InvoiceNo = as.factor(InvoiceNo),
         StockCode = as.factor(StockCode),
         Quantity = as.numeric(Quantity),
         InvoiceDate = ymd_hms(InvoiceDate),
         UnitPrice = as.numeric(UnitPrice),
         CustomerID = as.factor(CustomerID),
         Country = as.factor(Country),
         InvoiceSum = Quantity * UnitPrice)

df <- df %>%
  mutate(InvoiceYYYYMM = format(as.Date(InvoiceDate), "%Y-%m"),
         InvoiceMonth = month(InvoiceDate, label = TRUE),
         InvoiceYear = as.factor(year(InvoiceDate)),
         InvoiceHour = as.factor(hour(InvoiceDate)),
         InvoiceWeekday = wday(InvoiceDate, label = TRUE))

backup_df <- df

summary(df)
```

```
##    InvoiceNo        StockCode      Description           Quantity
##  573585 :  1114   85123A :  2313   Length:541909      Min.   :-80995.00
##  581219 :   749   22423  :  2203   Class :character   1st Qu.:     1.00
##  581492 :   731   85099B :  2159   Mode  :character   Median :     3.00
##  580729 :   721   47566  :  1727                      Mean   :     9.55
##  558475 :   705   20725  :  1639                      3rd Qu.:    10.00
##  579777 :   687   84879  :  1502                      Max.   : 80995.00
##  (Other):537202   (Other):530366
##    InvoiceDate                    UnitPrice           CustomerID
##  Min.   :2010-12-01 08:26:00   Min.   :-11062.06   17841  :  7983
##  1st Qu.:2011-03-28 11:34:00   1st Qu.:     1.25   14911  :  5903
##  Median :2011-07-19 17:17:00   Median :     2.08   14096  :  5128
##  Mean   :2011-07-04 13:34:57   Mean   :     4.61   12748  :  4642
##  3rd Qu.:2011-10-19 11:27:00   3rd Qu.:     4.13   14606  :  2782
##  Max.   :2011-12-09 12:50:00   Max.   : 38970.00   (Other):380391
##                                                    NA's   :135080
##            Country          uniqueID        InvoiceSum
##  United Kingdom:495478   Min.   :     1   Min.   :-168469.60
##  Germany       :  9495   1st Qu.:135478   1st Qu.:      3.40
##  France        :  8557   Median :270955   Median :      9.75
##  EIRE          :  8196   Mean   :270955   Mean   :     17.99
##  Spain         :  2533   3rd Qu.:406432   3rd Qu.:     17.40
##  Netherlands   :  2371   Max.   :541909   Max.   : 168469.60
```

```
##  (Other)       : 15279
##  InvoiceYYYYMM       InvoiceMonth    InvoiceYear    InvoiceHour
##  Length:541909      Nov    : 84711   2010: 42481   12     : 78709
##  Class :character   Dec    : 68006   2011:499428   15     : 77519
##  Mode  :character   Oct    : 60742                 13     : 72259
##                     Sep    : 50226                 14     : 67471
##                     Jul    : 39518                 11     : 57674
##                     May    : 37030                 16     : 54516
##                     (Other):201676                 (Other):133761
##  InvoiceWeekday
##  Sun: 64375
##  Mon: 95111
##  Tue:101808
##  Wed: 94565
##  Thu:103857
##  Fri: 82193
##  Sat:     0
```

1.3 Missing Value Treatment I

```
nas <-
  tibble(has.na = sapply(X = df, FUN = function(x){any(is.na(x))}),
         sum.na = sapply(X = df, FUN = function(x){sum(is.na(x))}),
         names = colnames(df))
nas
```

```
## # A tibble: 15 x 3
##    has.na sum.na names
##    <lgl>   <int> <chr>
##  1 FALSE       0 InvoiceNo
##  2 FALSE       0 StockCode
##  3 TRUE     1454 Description
##  4 FALSE       0 Quantity
##  5 FALSE       0 InvoiceDate
##  6 FALSE       0 UnitPrice
##  7 TRUE   135080 CustomerID
##  8 FALSE       0 Country
##  9 FALSE       0 uniqueID
## 10 FALSE       0 InvoiceSum
## 11 FALSE       0 InvoiceYYYYMM
## 12 FALSE       0 InvoiceMonth
## 13 FALSE       0 InvoiceYear
## 14 FALSE       0 InvoiceHour
## 15 FALSE       0 InvoiceWeekday
```

As we kann see CustomerID and Description has quite a lot of missing values which needs to investigated further therefore please refer to the appendix

```
adjustments <- c("wrongly", "found", "Found", "coded", "check", "damaged",
                 "incorrectly", "DOTCOM", "dotcom", "destroyed",
                 "thrown", "mystery", "amazon",
                 "Unsaleable", "Incorrect", "lost", "wet",
                 "sold", "AMAZON", "debt", "Bank Charges",
                 "BANK ACCOUNT", "SAMPLES")
```

```r
na <- df %>% filter(is.na(CustomerID) | is.na(Description)) %>% # filter categories
                                                               # with missing values
  filter((is.na(CustomerID) & is.na(Description)) |            # No connection
           (UnitPrice == 0 & Quantity < 0) |                   # Retour, no logical connection
           grepl(paste(adjustments, collapse="|"), Description) |
           grepl("\\?", Description) |                         # Keywords
           (is.na(CustomerID) & Description == "Manual") |     # Manual correction, without
                                                               # customer/description
           (is.na(CustomerID) & Description == "Discount"))    # Discount, without customer/descriptio

df <- df %>% filter(!(df$uniqueID %in% na$uniqueID))

nas <-
  tibble(has.na = sapply(X = df, FUN = function(x){any(is.na(x))}),
         sum.na = sapply(X = df, FUN = function(x){sum(is.na(x))}),
         names = colnames(df))
nas
```

```
## # A tibble: 15 x 3
##    has.na sum.na names
##    <lgl>   <int> <chr>
##  1 FALSE       0 InvoiceNo
##  2 FALSE       0 StockCode
##  3 FALSE       0 Description
##  4 FALSE       0 Quantity
##  5 FALSE       0 InvoiceDate
##  6 FALSE       0 UnitPrice
##  7 TRUE   132088 CustomerID
##  8 FALSE       0 Country
##  9 FALSE       0 uniqueID
## 10 FALSE       0 InvoiceSum
## 11 FALSE       0 InvoiceYYYYMM
## 12 FALSE       0 InvoiceMonth
## 13 FALSE       0 InvoiceYear
## 14 FALSE       0 InvoiceHour
## 15 FALSE       0 InvoiceWeekday
```

```r
cat(nas$sum.na[7]/length(df$CustomerID)*100, "% of the CustomerID's are NA's")
```

```
## 24.5099 % of the CustomerID's are NA's
```

As we can see, we reduced the amout of missing values without any relation to customers (no ID and stock adjustments) by almost 3000 observations. As we step further, we will take care of rest.

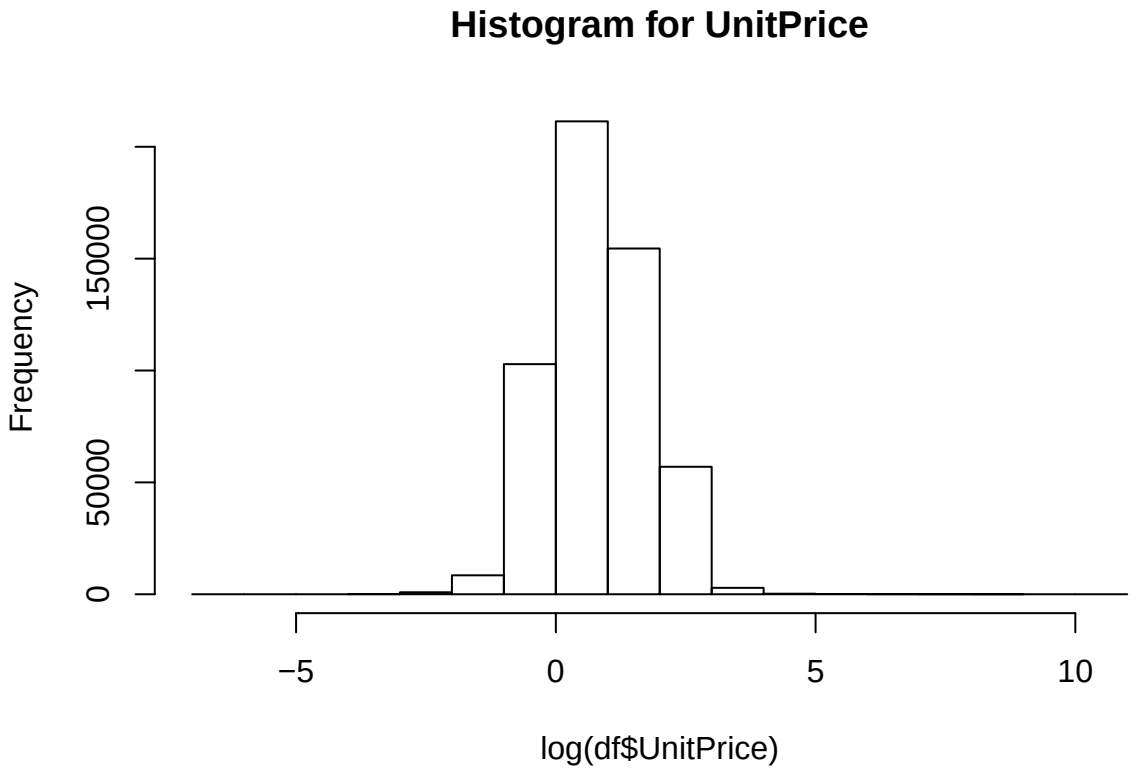
1.4 Outlier Detection and Treatment

1.4.1 Price Outliers

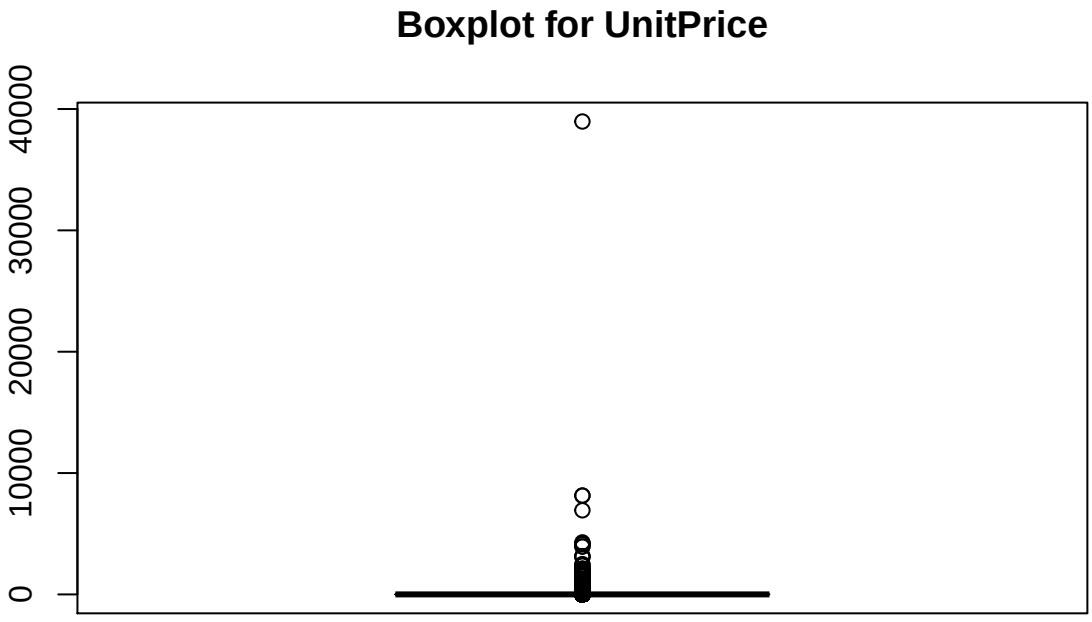
```r
range(df$UnitPrice)
```

```
## [1]     0 38970
```

```r
hist(x = log(df$UnitPrice), main = "Histogram for UnitPrice", breaks = 20)
```




```
box_price <- boxplot(x = df$UnitPrice,  main="Boxplot for UnitPrice")
```




```
box_price_out <- box_price$out

df_outlier_price <- filter(df, UnitPrice %in% box_price_out)
df_outlier_price %>% select(UnitPrice) %>% summary()
```

```
##    UnitPrice
##  Min.   :    8.47
##  1st Qu.:    9.95
```

```
##  Median :   11.02
##  Mean   :   18.85
##  3rd Qu.:   14.95
##  Max.   :38970.00
```

It seems there are still some reasonable prices if we check the UnitPrice summary, hence we take the last +/- 3 standard deviation (the two tailed 1% ends) approach.

```
df_outlier_price <- df %>% filter(UnitPrice >= 3*sd(UnitPrice) | UnitPrice <= -3*sd(UnitPrice))
df_outlier_price %>% select(UnitPrice) %>% summary()
```

```
##    UnitPrice
##  Min.   :  183
##  1st Qu.:  293
##  Median :  523
##  Mean   : 1190
##  3rd Qu.: 1126
##  Max.   :38970
```

```
table(df_outlier_price$Description)
```

```
##
##                CRUK Commission                         Discount
##                             12                                6
##                 DOTCOM POSTAGE  LOVE SEAT ANTIQUE WHITE METAL
##                             12                               12
##                         Manual PICNIC BASKET WICKER 60 PIECES
##                            102                                2
##                        POSTAGE   VINTAGE BLUE KITCHEN CABINET
##                             12                                3
##    VINTAGE RED KITCHEN CABINET
##                              8
```

The prices and the description here look much more unreasonable for our analysis. It seems there are 3 items left, so we exclude them.

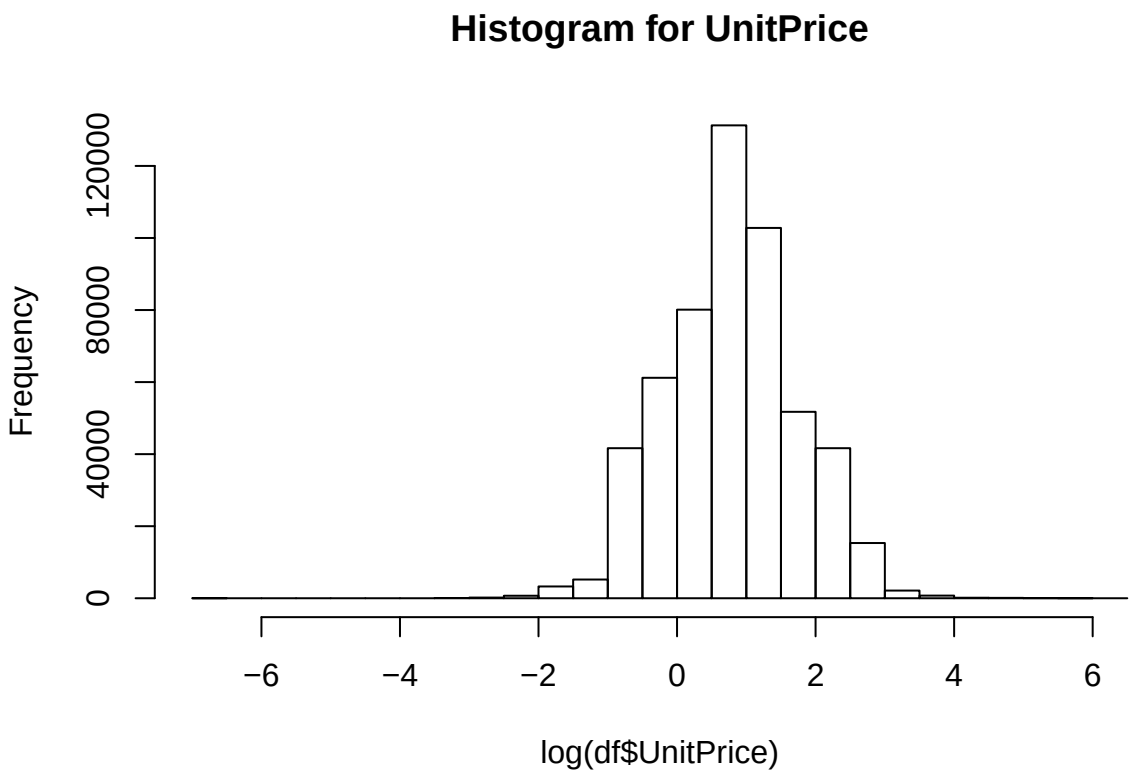
```
products <- c("PICNIC", "VINTAGE", "LOVE SEAT")
df_outlier_price <- filter(df_outlier_price, !(grepl(paste(products, collapse="|"), Description)))

table(df_outlier_price$Description)
```

```
##
## CRUK Commission        Discount  DOTCOM POSTAGE          Manual
##              12               6              12             102
##         POSTAGE
##              12
```

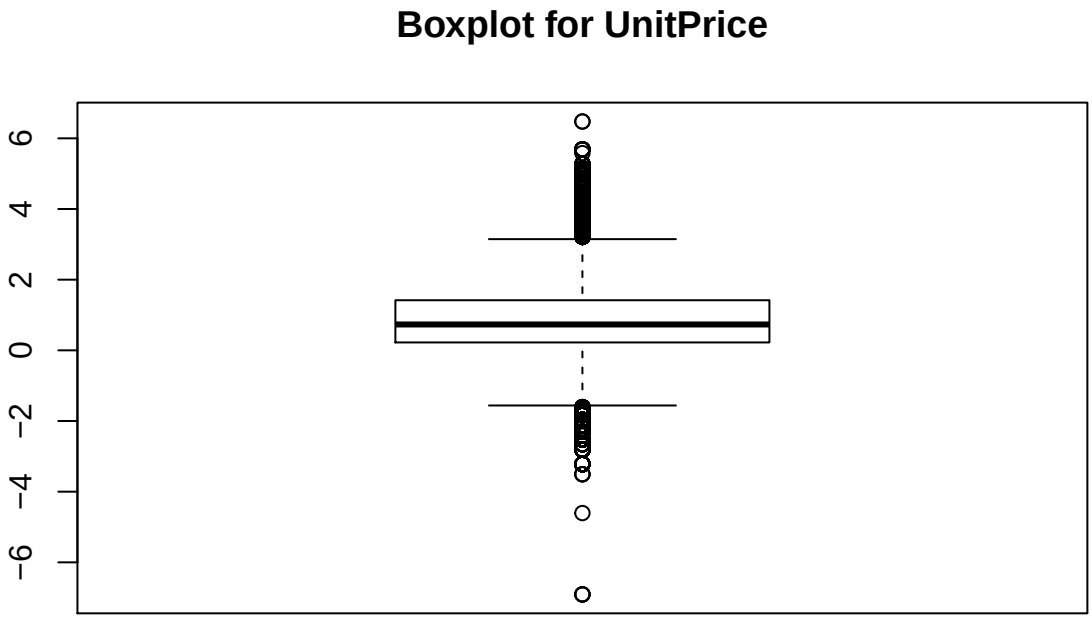
```
df <- df %>% filter(!(df$uniqueID %in% df_outlier_price$uniqueID))
```

Now we have only some fees and other expenses in out outlier data frame, which we can drop. Lets have a look into the distribution of the UnitPrice

```
hist(x = log(df$UnitPrice), main = "Histogram for UnitPrice", breaks = 20)
```




```
boxplot(x = log(df$UnitPrice),  main="Boxplot for UnitPrice")
```




```
cat("Negative Price:", any(df$UnitPrice < 0), "\n",
    "Price Range:", range(df$UnitPrice), "\n")
```

```
## Negative Price: FALSE
##  Price Range: 0 649.5
```

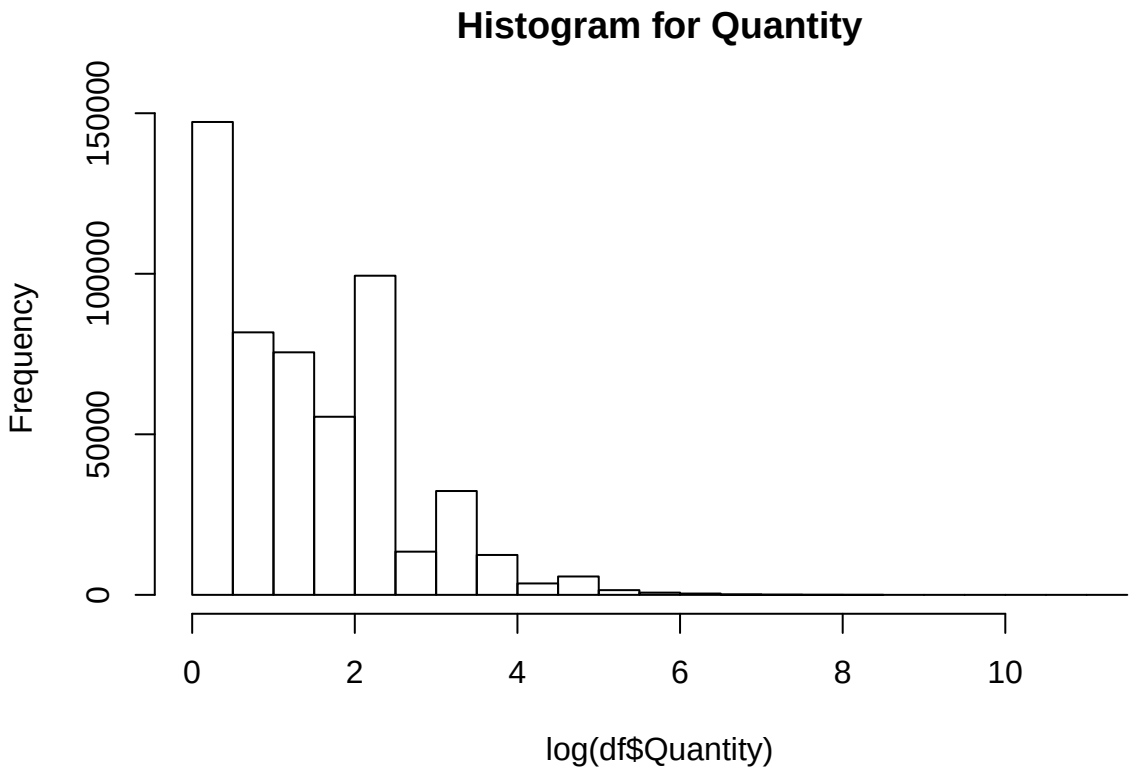
Both plots are reflecting a reasonable price distribution.

1.4.2 Quantity Outliers

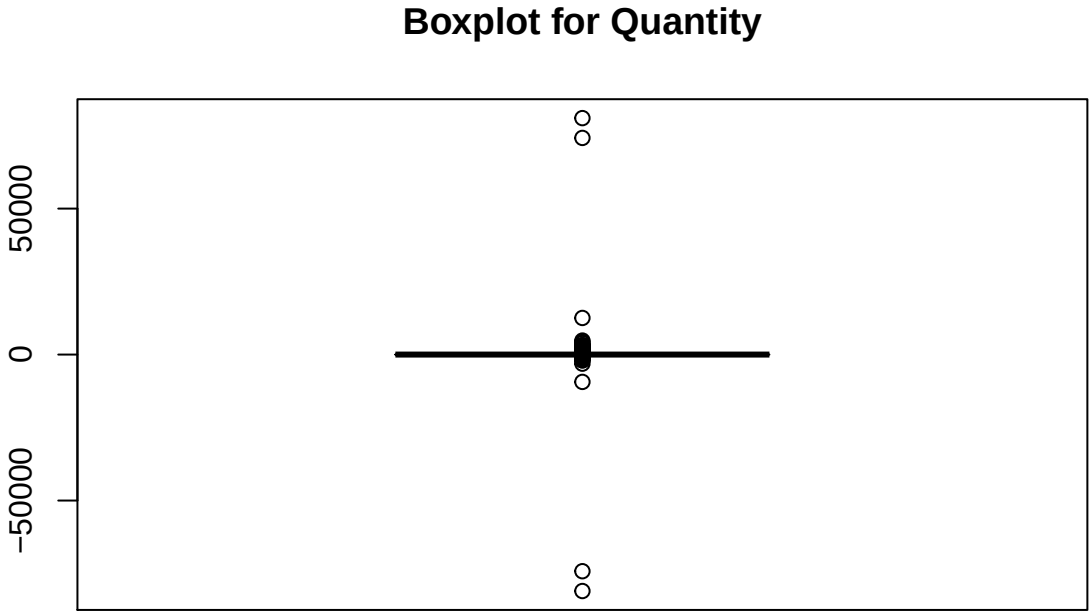
```
range(df$Quantity)
```

```
## [1] -80995  80995
```

```
hist(x = log(df$Quantity), main = "Histogram for Quantity", breaks = 20)
```




```
box_quantity <- boxplot(x = df$Quantity, main="Boxplot for Quantity")
```



```
box_quantity_out <- box_quantity$out

df_outlier_quantity <- filter(df, Quantity %in% box_quantity_out)
df_outlier_quantity %>% select(Quantity) %>% summary()
```

```
##     Quantity
##  Min.   :-80995.00
##  1st Qu.:    24.00
##  Median :    25.00
##  Mean   :    54.29
##  3rd Qu.:    48.00
##  Max.   : 80995.00
```

Similar to the UnitPrice case we still have some reasonable Quantity amounts in our current outlier DF

```
df_outlier_quantity <- df %>% filter(Quantity >= 3*sd(Quantity) |
                                       Quantity <= -3*sd(Quantity)) %>%
  arrange(desc(Quantity))
df_outlier_quantity %>% select(Quantity) %>% summary()
```

```
##     Quantity
##  Min.   :-80995
##  1st Qu.:   720
##  Median :   960
##  Mean   :   996
##  3rd Qu.:  1296
##  Max.   : 80995
```

```
head(table(df_outlier_quantity$Description), n = 7)
```

```
##
## 12 PENCILS SMALL TUBE RED RETROSPOT   5 HOOK HANGER RED MAGIC TOADSTOOL
##                                   1                                   1
##     60 CAKE CASES VINTAGE CHRISTMAS        60 TEATIME FAIRY CAKE CASES
##                                   1                                   1
##      72 SWEETHEART FAIRY CAKE CASES       ASSORTED COLOUR BIRD ORNAMENT
##                                   2                                   3
##      ASSORTED COLOUR T-LIGHT HOLDER
##                                   1
```

```
head(select(df_outlier_quantity,
            c("InvoiceNo", "StockCode", "Description",
              "Quantity", "InvoiceDate", "UnitPrice")),
     n = 7)
```

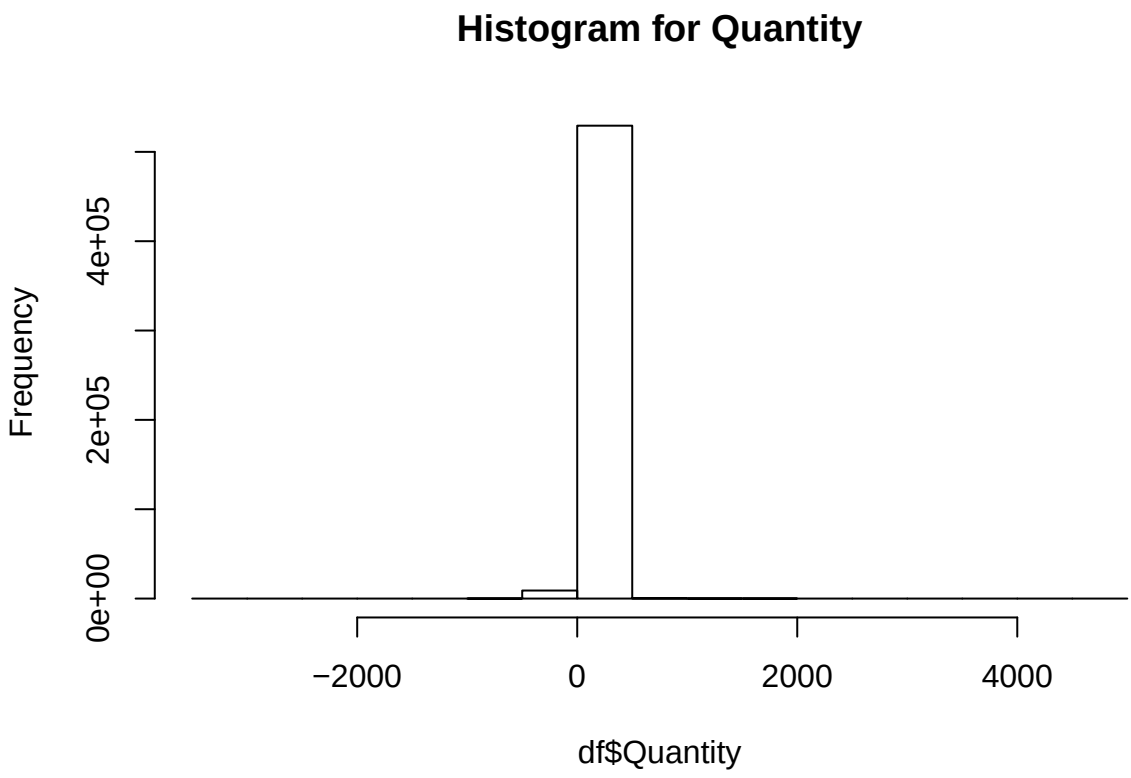
```
## # A tibble: 7 x 6
##   InvoiceNo StockCode Description    Quantity InvoiceDate         UnitPrice
##   <fct>     <fct>     <chr>             <dbl> <dttm>                  <dbl>
## 1 581483    23843     PAPER CRAFT ,~    80995 2011-12-09 09:15:00      2.08
## 2 541431    23166     MEDIUM CERAMI~    74215 2011-01-18 10:01:00      1.04
## 3 578841    84826     ASSTD DESIGN ~    12540 2011-11-25 15:57:00      0
## 4 573008    84077     WORLD WAR 2 G~     4800 2011-10-27 12:26:00      0.21
## 5 554868    22197     SMALL POPCORN~     4300 2011-05-27 10:52:00      0.72
## 6 544612    22053     EMPIRE DESIGN~     3906 2011-02-22 10:43:00      0.82
## 7 560599    18007     ESSENTIAL BAL~     3186 2011-07-19 17:04:00      0.06
```

```
tail(select(df_outlier_quantity,
            c("InvoiceNo", "StockCode", "Description",
              "Quantity", "InvoiceDate", "UnitPrice")),
     n = 7)
```

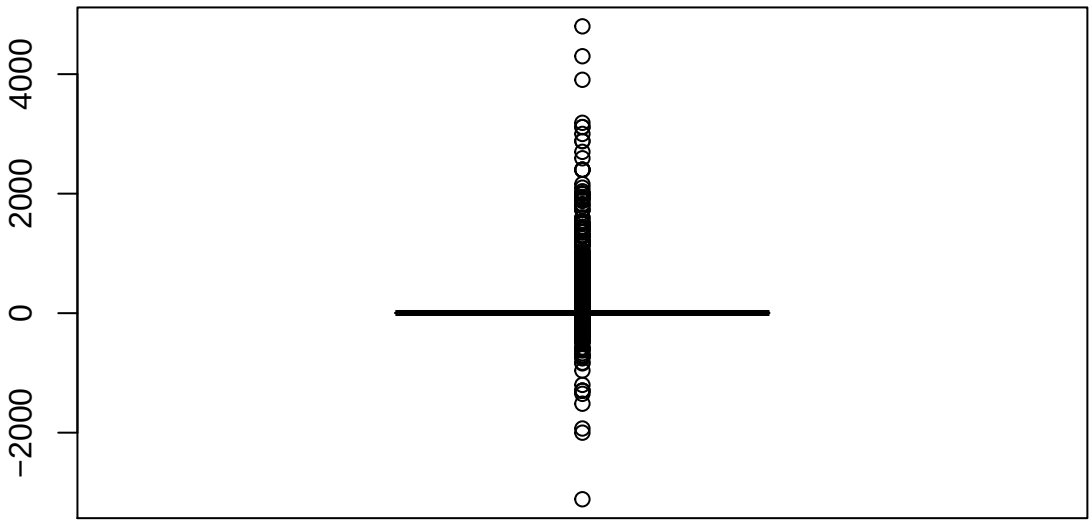
```
## # A tibble: 7 x 6
##   InvoiceNo StockCode Description    Quantity InvoiceDate         UnitPrice
##   <fct>     <fct>     <chr>             <dbl> <dttm>                  <dbl>
## 1 C556522   22920     HERB MARKER B~    -1515 2011-06-13 11:21:00      0.55
## 2 C550456   85123A    WHITE HANGING~    -1930 2011-04-18 13:08:00      2.55
## 3 C550456   21175     GIN + TONIC D~    -2000 2011-04-18 13:08:00      1.85
## 4 C550456   21108     FAIRY CAKE FL~    -3114 2011-04-18 13:08:00      2.1
## 5 C536757   84347     ROTATING SILV~    -9360 2010-12-02 14:23:00      0.03
## 6 C541433   23166     MEDIUM CERAMI~   -74215 2011-01-18 10:17:00      1.04
## 7 C581484   23843     PAPER CRAFT ,~   -80995 2011-12-09 09:27:00      2.08
```

As we can see the three biggest and three smallest in our outlier data are not reasonable to keep, as the two extremes (Quantity: 80995 and 74215) are directly corrected as well as the third positive one (Quantity: 12540), as it has a price of zero. The last observation with a negative Quantity of 9360 seems also not logical, judging from the StockCodes appearance. Which is why we are going to drop them.

```
extremes <- c(540422, 61620, 502123, 540423, 61625, 4288)
df_outlier_quantity <- df_outlier_quantity %>%
  filter(grepl(paste(extremes, collapse="|"), uniqueID))
df <- df %>% filter(!(df$uniqueID %in% df_outlier_quantity$uniqueID))

hist(x = df$Quantity, main = "Histogram for Quantity", breaks = 20)
```




```
boxplot(x = df$Quantity,  main="Boxplot for Quantity")
```



1.5 Descriptive Analysis

```
# Dummy Variables for NA and retoure
df <- df %>%
  mutate(ex1 = (str_extract(InvoiceNo, "[A-Z]")), # for retoure
         retoure = ifelse(test = is.na(ex1), F,
                          ifelse(ex1 == "C", yes = T, no = F)), # for Missing CustomerID
         CustomerID_Status = ifelse(is.na(CustomerID), yes = "NoID", no = "ID")) %>%
  select(-ex1)
```

1.5.1 Statistics

```
# Total statistics
ID_stat <- df %>% group_by(CustomerID_Status) %>%
  summarize(TotalInvoice = n_distinct(InvoiceNo),
            totInvoiceSum = sum(InvoiceSum),
            negInvoiceSum = sum(ifelse(InvoiceSum < 0, InvoiceSum, 0)),
            posInvoiceSum = sum(ifelse(InvoiceSum >= 0, InvoiceSum, 0)),
            maxInvoiceSum = max(InvoiceSum),
            minInvoiceSum = min(InvoiceSum),
            meanInvoiceSum = mean(InvoiceSum),
            medianInvoiceSum = median(InvoiceSum),
            totQuantity = sum(Quantity),
            negQuantity = sum(ifelse(Quantity < 0, Quantity, 0)),
            posQuantity = sum(ifelse(Quantity >= 0, Quantity, 0)),
            maxQuantity = max(Quantity),
            minQuantity = min(Quantity),
            meanQuantity = mean(Quantity),
            medianQuantity = median(Quantity)) %>%
  as.matrix(.) %>% t(.) %>% as_tibble(., rownames = "id") %>%
  rename(Statistics = id,
         NoNA = V1,
         OnlyNA = V2) %>%
```

```
  mutate(NoNA = round(as.numeric(NoNA), digits = 2),
         OnlyNA = round(as.numeric(OnlyNA), digits = 2))

ID_stat2 <- df %>%
  summarize(TotalInvoice = n_distinct(InvoiceNo),
            totInvoiceSum = sum(InvoiceSum),
            negInvoiceSum = sum(ifelse(InvoiceSum < 0, InvoiceSum, 0)),
            posInvoiceSum = sum(ifelse(InvoiceSum >= 0, InvoiceSum, 0)),
            maxInvoiceSum = max(InvoiceSum),
            minInvoiceSum = min(InvoiceSum),
            meanInvoiceSum = mean(InvoiceSum),
            medianInvoiceSum = median(InvoiceSum),
            totQuantity = sum(Quantity),
            negQuantity = sum(ifelse(Quantity < 0, Quantity, 0)),
            posQuantity = sum(ifelse(Quantity >= 0, Quantity, 0)),
            maxQuantity = max(Quantity),
            minQuantity = min(Quantity),
            meanQuantity = mean(Quantity),
            medianQuantity = median(Quantity)) %>%
  gather() %>%
  rename(Statistics = key,
         All_ID = value)
ID_stat <- ID_stat %>% inner_join(ID_stat2)
```

```
## Joining, by = "Statistics"
```

```
ID_stat
```

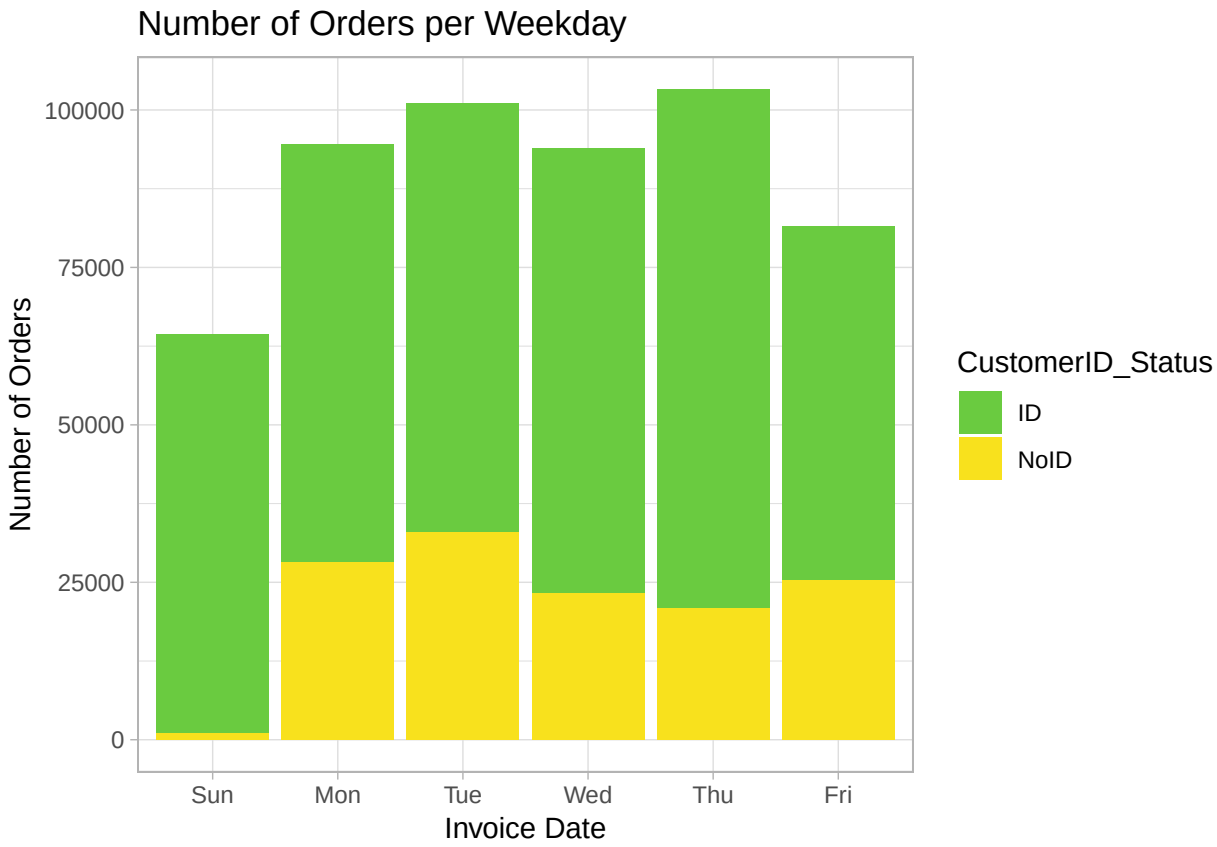
```
## # A tibble: 15 x 4
##    Statistics             NoNA      OnlyNA       All_ID
##    <chr>                 <dbl>       <dbl>        <dbl>
##  1 TotalInvoice          22073        1529        23602
##  2 totInvoiceSum       8356247     1506569      9862816.
##  3 negInvoiceSum       -238141.      -5113.      -243255.
##  4 posInvoiceSum       8594388     1511682     10106071.
##  5 maxInvoiceSum         38970        4782.        38970
##  6 minInvoiceSum         -6539.       -206.        -6539.
##  7 meanInvoiceSum           20.6        11.4          18.3
##  8 medianInvoiceSum         11.1         4.96          9.84
##  9 totQuantity         4903722      423815       5327537
## 10 negQuantity         -110158       -2576       -112734
## 11 posQuantity         5013880      426391       5440271
## 12 maxQuantity            4800        1820          4800
## 13 minQuantity           -3114        -144         -3114
## 14 meanQuantity             12.1         3.21          9.89
## 15 medianQuantity            5            1             3
```

1.5.2 Graphics

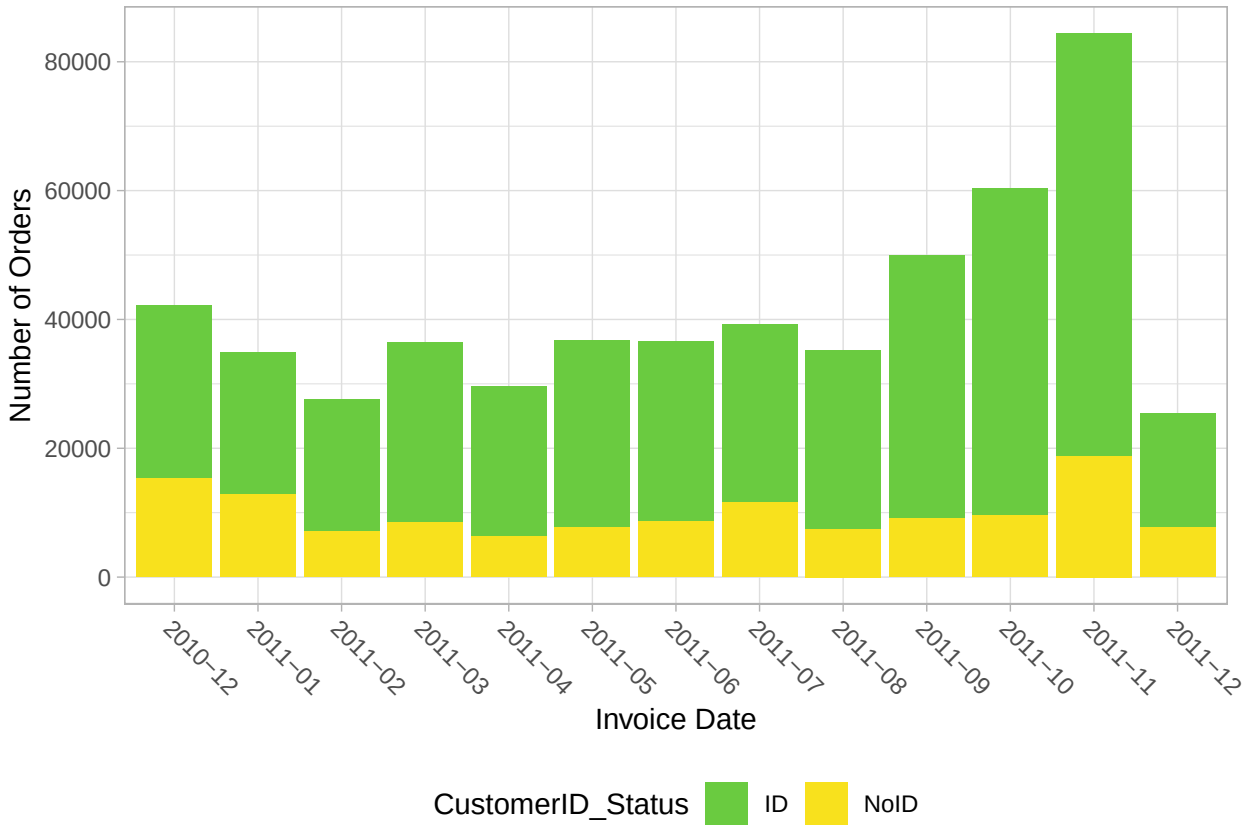
As we saw earlier, the missing values in Customer ID is fairly high, which is why we do not exclude them yet and include them into graphics to get an idea how they are distributed over time in realtion to the number of orders.

```
ggplot(df) +
  geom_bar(aes(x = InvoiceWeekday, fill = CustomerID_Status)) +
```

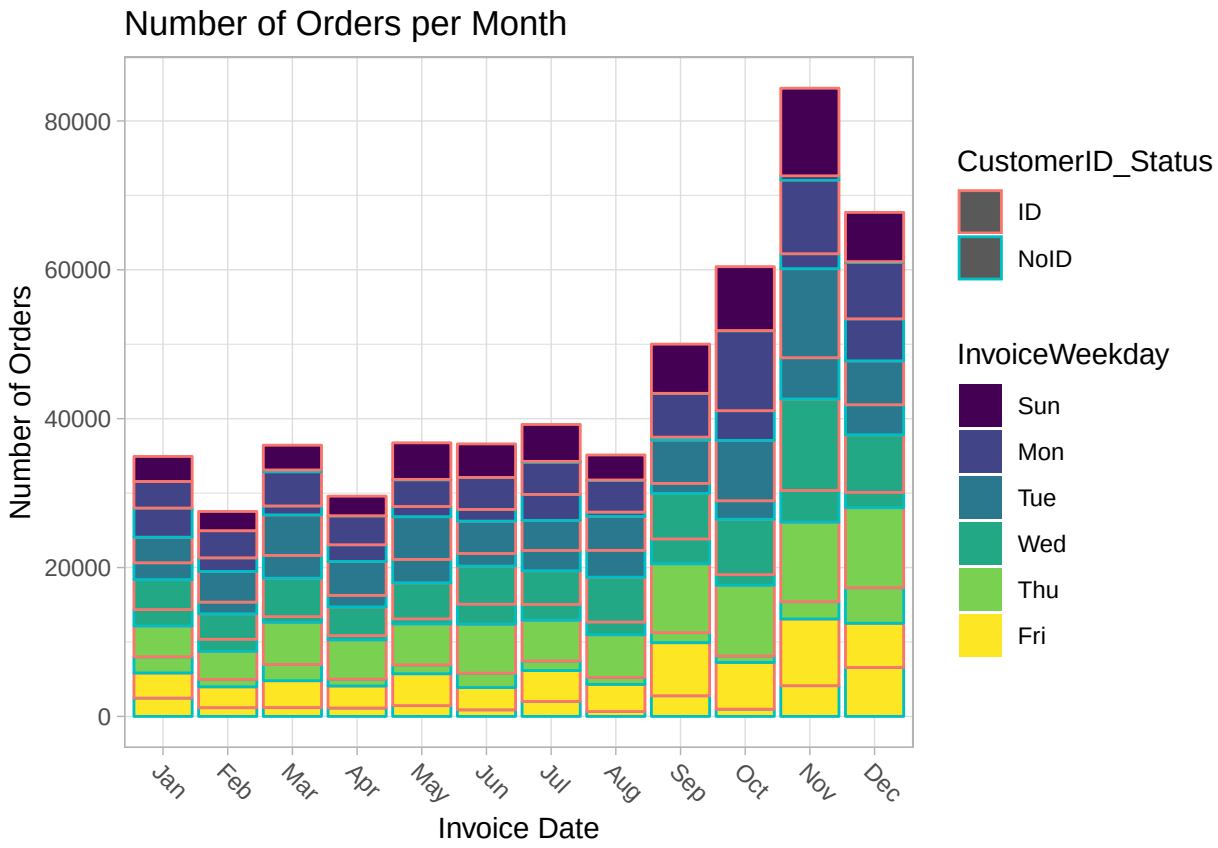
```
  labs(title = "Number of Orders per Weekday") +
  scale_fill_manual(values = c("#6ACB40", "#F8E11D")) +
  theme_light() +
  xlab("Invoice Date") + ylab("Number of Orders")
```




```
ggplot(df) +
  geom_bar(aes(x = InvoiceYYYYMM, fill = CustomerID_Status)) +
  theme_light() +
  scale_fill_manual(values = c("#6ACB40", "#F8E11D")) +
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = -45,
                                   vjust = 1,
                                   hjust = 0)) +
  xlab("Invoice Date") + ylab("Number of Orders")
```




```
ggplot(df) +
  geom_bar(aes(x = InvoiceMonth, fill=InvoiceWeekday, color = CustomerID_Status)) +
  theme_light() +
  theme(axis.text.x = element_text(angle = -45,
                                   vjust = 1,
                                   hjust = 0)) +
  labs(title = "Number of Orders per Month") +
  xlab("Invoice Date") + ylab("Number of Orders")
```



Judging from the three graphs above, we can conclude that the variation over time in amount of orders are very similar. Which is why we are going to exclude them in the following recency, frequency and monetary analysis.

1.6 CRM - Recency Frequency Monetary Analysis

```
# Construct Date Variables
latest <- max(df$InvoiceDate)
earliest <- min(df$InvoiceDate)

RFM <- df %>%
  group_by(CustomerID) %>%
  mutate(recency = as.numeric(latest - max(InvoiceDate)),
         frequency = n_distinct(InvoiceNo),
         monetary = sum(InvoiceSum)/n_distinct(CustomerID),
         quantity = sum(Quantity)) %>%
  distinct(Country, recency, frequency, monetary, quantity) %>%
  ungroup() %>%
  drop_na() %>%
  mutate(duplicate = duplicated(CustomerID)) %>%
  filter(duplicate == 0) %>%
  select(-duplicate)

RFM %>% summary()
```

```
##            Country        recency         frequency
##  United Kingdom:3943   Min.   :  0.00   Min.   :  1.000
```

```
##  Germany       :  95   1st Qu.: 17.36   1st Qu.:  1.000
##  France        :  87   Median : 49.92   Median :  3.000
##  Spain         :  28   Mean   : 91.96   Mean   :  5.058
##  Belgium       :  24   3rd Qu.:142.30   3rd Qu.:  5.000
##  Switzerland   :  20   Max.   :373.12   Max.   :243.000
##  (Other)       : 167
##     monetary           quantity        CustomerID
##  Min.   : -1192.2   Min.   :  -303   12347  :   1
##  1st Qu.:   295.4   1st Qu.:   153   12348  :   1
##  Median :   650.5   Median :   365   12349  :   1
##  Mean   :  1914.8   Mean   :  1124   12350  :   1
##  3rd Qu.:  1611.7   3rd Qu.:   963   12352  :   1
##  Max.   :279695.4   Max.   :196720   12353  :   1
##                                      (Other):4358
```

After understanding the difference of the missing values and its implications from our data set, we are going to drop the NAs

1.6.1 Implementing the 80/20 Pareto Principle

```
pareto8020money <- 0.8 * sum(RFM$monetary)
pareto8020freq <- 0.8 * sum(RFM$frequency)
pareto8020quant <- 0.8 * sum(RFM$quantity)

RFM <- RFM %>%
  mutate(monetaryrank = order(order(monetary, decreasing=FALSE)),
         quantityrank = order(order(quantity, decreasing=FALSE)),
         frequancyrank = order(order(frequency, decreasing=FALSE))) %>%
  arrange(monetaryrank) %>%
  mutate(pareto_money = ifelse(cumsum(monetary) <= pareto8020money,
                               "Low Value Customer", "High Value Customer")) %>%
  arrange(quantityrank) %>%
  mutate(pareto_quantity = ifelse(cumsum(quantity) <= pareto8020quant,
                                  "Low Quantity Customer", "High Quantity Customer")) %>%
  arrange(frequancyrank) %>%
  mutate(pareto_frequency = ifelse(cumsum(frequency) <= pareto8020freq,
                                   "Low Frequency Customer", "High Frequency Customer"))
RFM
```

```
## # A tibble: 4,364 x 12
##    Country recency frequency monetary quantity CustomerID monetaryrank
##    <fct>     <dbl>     <int>    <dbl>    <dbl> <fct>             <int>
##  1 United~    373.         1    490.       190 18074              1795
##  2 United~    373.         1     79.6        8 13747               147
##  3 Nether~    373.         1    193.        97 12791               667
##  4 United~    373.         1    243.       173 17908               895
##  5 United~    373.         1    233.       111 16583               861
##  6 United~    373.         1    277.       160 17968              1030
##  7 United~    373.         1    313.       197 14729              1192
##  8 United~    373.         1    161         38 14237               512
##  9 United~    373.         1    116.        51 15350               297
## 10 United~    373.         1    488.       160 15165              1791
## # ... with 4,354 more rows, and 5 more variables: quantityrank <int>,
## #   frequancyrank <int>, pareto_money <chr>, pareto_quantity <chr>,
## #   pareto_frequency <chr>
```

```
table(RFM$pareto_money, RFM$pareto_frequency, RFM$pareto_quantity)
```

```
## , ,  = High Quantity Customer
##
##
##                       High Frequency Customer Low Frequency Customer
##   High Value Customer                      12                      1
##   Low Value Customer                        2                      2
##
## , ,  = Low Quantity Customer
##
##
##                       High Frequency Customer Low Frequency Customer
##   High Value Customer                       2                      1
##   Low Value Customer                       76                   4268
```

1.6.2 Customer Relation By Country

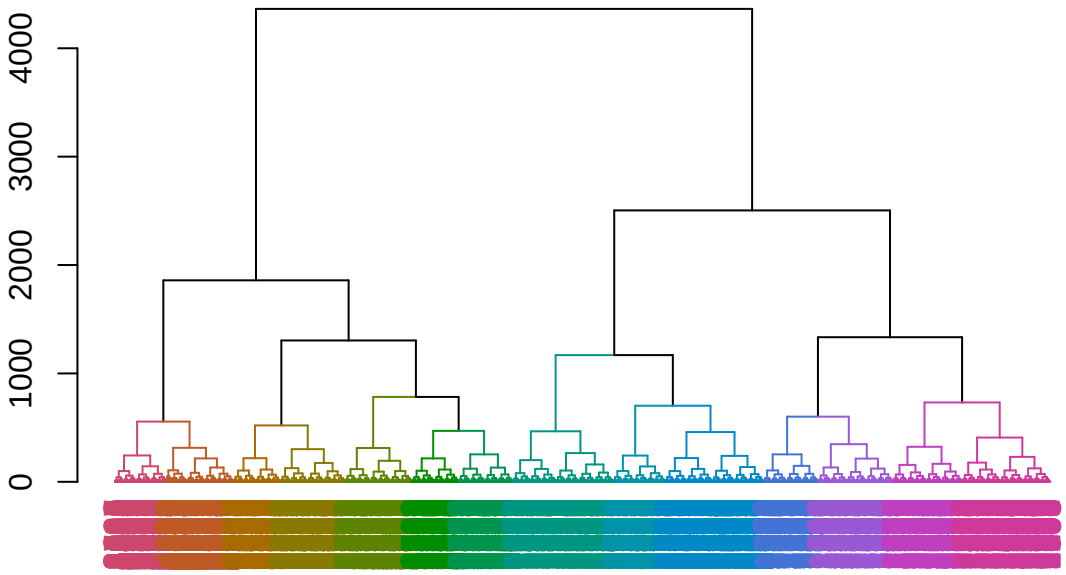
```
CustomerRelation <-
  aggregate(RFM[, c(2:5)],
            by = list(RFM$Country),
            FUN = function(x) {sum(x)})
CustomerRelation <- CustomerRelation %>%
  mutate(recency_split = round((recency/sum(recency)*100), digits = 3),
         frequency_split = round((frequency/sum(frequency)*100), digits = 3),
         monetary_split = round((monetary/sum(monetary)*100), digits = 3),
         quantity_split = round((quantity/sum(quantity)*100), digits = 3)) %>%
  arrange(desc(monetary_split, quantity_split, frequency_split))
head(CustomerRelation)
```

```
##          Group.1     recency frequency  monetary quantity recency_split
## 1 United Kingdom 361799.5146     19791 6822643.4  4005370        90.153
## 2    Netherlands    911.1069       100  284867.9   200129         0.227
## 3           EIRE    183.9465       313  251840.9   136331         0.046
## 4        Germany   7516.1188       594  220695.2   117445         1.873
## 5         France   7599.6500       453  196712.8   109848         1.894
## 6      Australia    895.9569        70  138380.1    84303         0.223
##   frequency_split monetary_split quantity_split
## 1          89.662         81.647         81.680
## 2           0.453          3.409          4.081
## 3           1.418          3.014          2.780
## 4           2.691          2.641          2.395
## 5           2.052          2.354          2.240
## 6           0.317          1.656          1.719
```

1.7 Customer Segmentation with Hierarchical Clustering

```
RFM_scaled <- RFM %>%
  mutate_at(c(2:5), funs(c(scale(.)))) # Scale Monetary, Recency, Frequency, Quantity

cluster <- RFM_scaled %>% # Select Monetary, Recency, Frequency, Quantity + Monetary Rank
  select(c(2:5,7))
```

```
# Initiate Cluster
h_complete <- hclust(dist(x = cluster, method = "euclidean"), method="complete")

# Plot Cluster
dendo <- as.dendrogram(h_complete)
dendo %>%
  set("branches_k_color", k = 14) %>%
  set("labels_col", k=14) %>%
  plot(horiz=FALSE, axes=TRUE)
```



Here we see how the hierarchical clustering split the data into one tree with many leafs. As we do not want to divide our the customers into too many segments, we choose 14, as this seems judging from the hight a good place.

```
# Cut Tree with 14 clusters
cluster14 <- cutree(h_complete, k = 14)
table(cluster14)
```

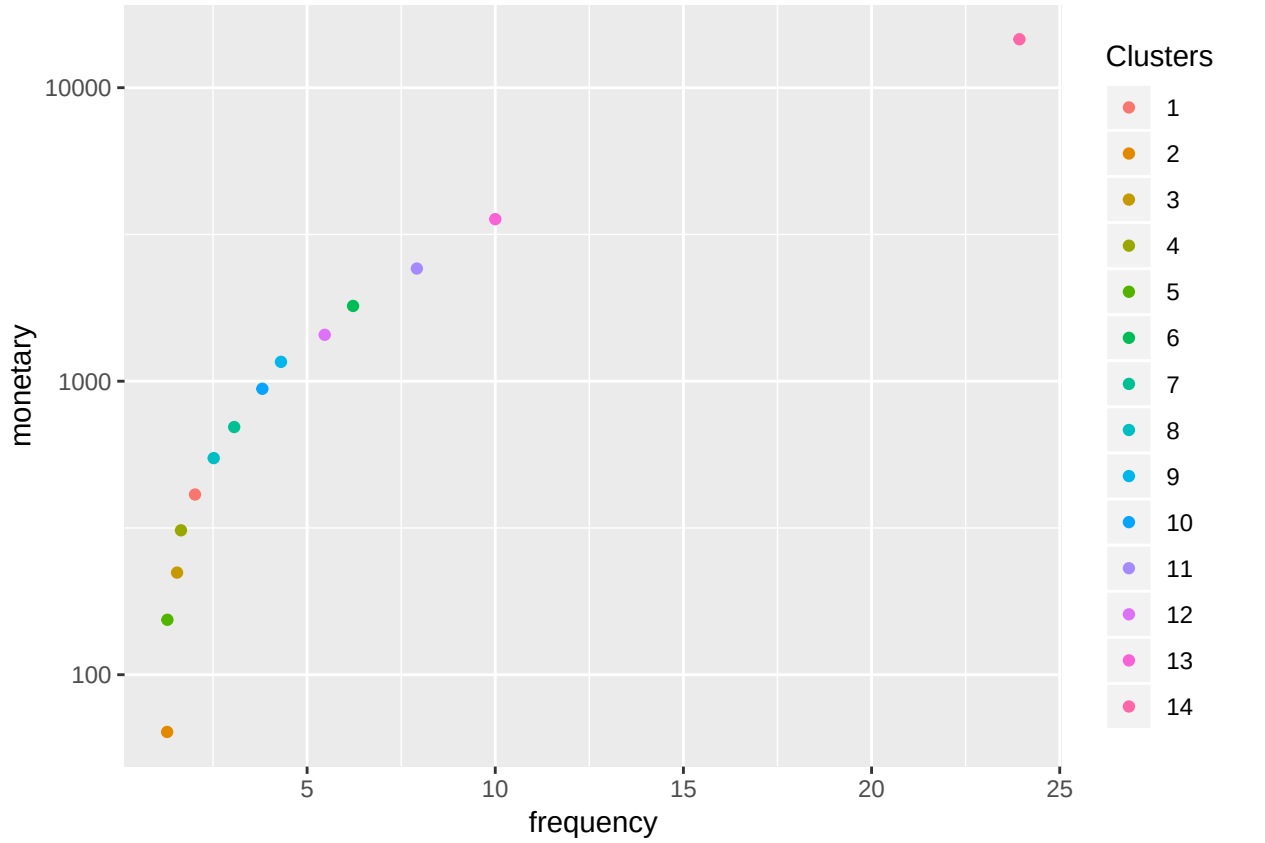
```
## cluster14
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14
## 467 348 409 324 254 217 460 243 219 302 313 254 244 310
```

```
cluster_df <- aggregate(RFM[, c(2:5)], by = list(cluster14), mean)
cluster_df
```

```
##    Group.1   recency frequency    monetary   quantity
## 1        1 114.91489  2.021413   411.03384  249.00214
## 2        2 171.69155  1.281609    63.82695   45.76437
## 3        3 139.99217  1.545232   222.77271  144.00244
## 4        4 137.52967  1.648148   310.38994  206.10185
## 5        5 159.32015  1.287402   153.84406  102.42913
## 6        6  52.26139  6.221198  1804.71332 1164.41935
## 7        7  77.85566  3.063043   697.99207  444.64565
## 8        8  99.39622  2.518519   546.91128  314.16049
## 9        9  63.26090  4.305936  1163.71059  714.47489
## 10      10  67.52067  3.811258   942.16702  565.92715
## 11      11  38.19044  7.916933  2420.20927 1466.51438
## 12      12  50.97549  5.468504  1438.99441  870.24409
## 13      13  34.25451 10.000000  3565.88660 2154.99180
```

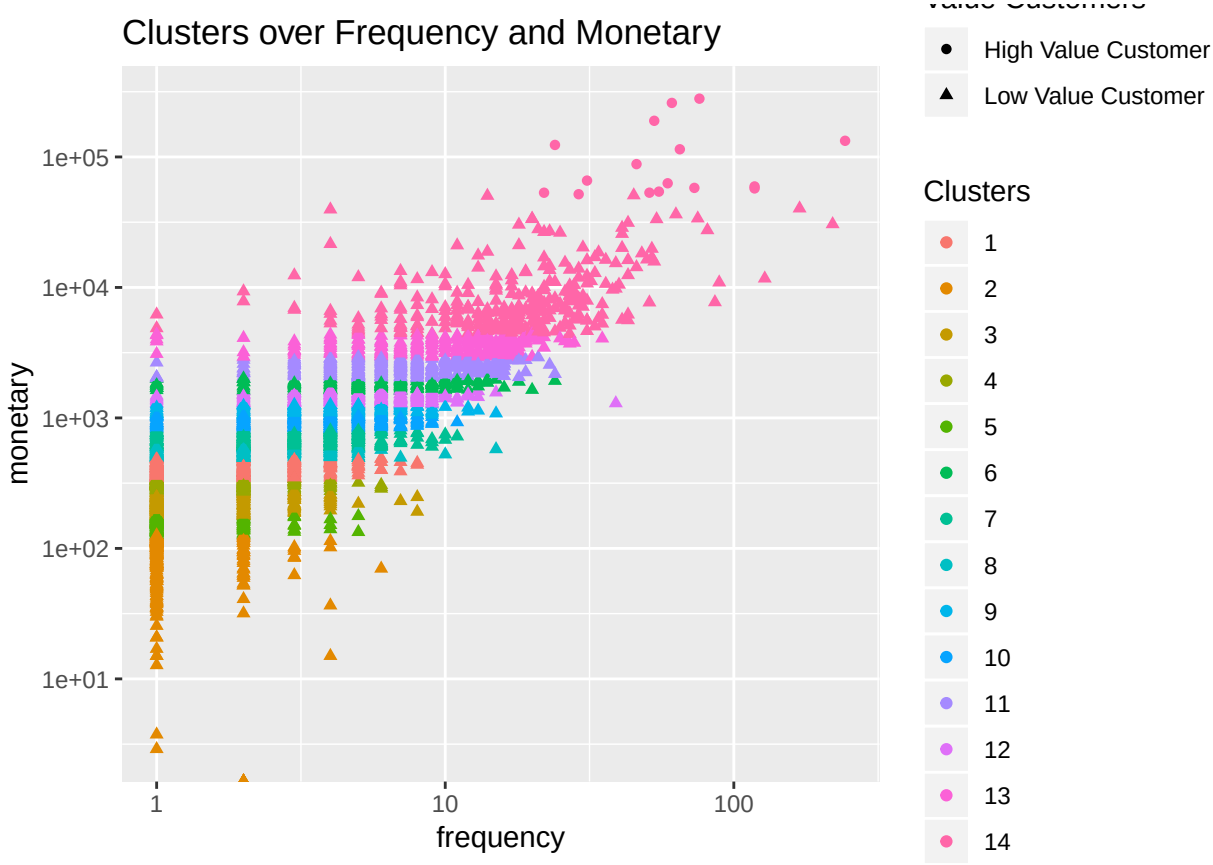
```
## 14      14  21.95309 23.929032 14623.33523 8235.50000
```

```
ggplot(cluster_df) +
  geom_point(aes(x = frequency, y = monetary, color = as.factor(Group.1))) +
  scale_y_log10() +
  labels(color = "Clusters")
```



Here we can see how the means of each cluster is distributed. As we used the ranks of out RFM+Q, the groups are not as structured as one would expect. Lets have a look at the overall data.

```
# Combine The clusters into the original RFM data frame
RFM <- mutate(RFM, cluster14 = cluster14)

ggplot(RFM) +
  geom_point(aes(x = frequency, y = monetary, color = as.factor(cluster14), shape = as.factor(pareto_mo
  scale_y_log10() +
  scale_x_log10() +
  labs(color='Clusters') +
  labs(shape='Value Customers') +
  labs(title = "Clusters over Frequency and Monetary")
```


Clusters over Frequency and Monetary
monetary
1e+05
1e+04
1e+03
1e+02
1e+01
1
10
100
frequency
High Value Customer
Low Value Customer
Clusters
1
2
3
4
5
6
7
8
9
10
11
12
13
14