Market Basket Analysis

Andreu Oros, Sergi Pallice, Joël Ribera

2019-07-25

Contents

Executive Summary

Introduction

Danielle has asked the team to perform a market basket analysis to help Blackwell's board of directors better understand the clientele that Electronidex is currently serving and if Electronidex would be an optimal acquisition.

A dataset of transactions has been provided. The dataset contains 9835 transactions and 125 different products over a 30-day period, or about 327 transactions a day. This tells us the retailer is neither large, nor small.

Results, conclusions and recommendations

After the analysis, we conclude Electronidex's sales are to be categorized in two forms: retail (B2C) and corporate (B2B). Had we had this information previous to our analysis, it would have saved time in the exploration phase. Interesting patterns and item relationships found:

Retailers buy the following products togeher:

lhs	Var.2	rhs
{Samsung Monitor}	=>	{CYBERPOWER Gamer Desktop}
{CYBERPOWER Gamer Desktop}	=>	{Apple Earpods}
{Apple Earpods}	=>	{CYBERPOWER Gamer Desktop}

Retailers buy the following categories togeher:

	lhs	Var.2	rhs
4	{Accessories,Keyboard}	=>	{Desktop}
5	{Accessories,Keyboard}	=>	{Desktop}
6	{Computer Cords,Mouse and Keyboard Combo}	=>	{Laptops}

Corporates buy the following products togeher:

	lhs	Var.2	rhs
7	{HP Laptop}	=>	{iMac}
8	{ViewSonic Monitor}	=>	{iMac}
9	{iMac}	=>	{HP Laptop}

Corporates buy the following products togeher:

	lhs	Var.2	rhs
10	{Laptops}	=>	{Desktop}
11	{Laptops,Monitors}	=>	{Desktop}
12	{Desktop}	=>	{Laptops}

Would Blackwell benefit in selling Electronidex's items? As a conclusion, we do recommend that Blackwell acquires Electronidex for the following reasons:

- **Product portfolio diversification and customer base expansion:** Having previously analyzed the current products that Blackwell supplies to its customers, we believe that new product categories will help reach new customers. Additionally, all clients acquired with Eltronidex will become part of Blackwell's customer base. The combination of diversifaction and expansion will consequentially increase Blackwell's revenue.
- **Beneficial Product Association:** Since Blackwell currently works with close to 50% of brands shared by Electronidex; Blackwell can take advantage of the discovered associations. This will imply the average amount of Blackwell's items per transaction will be increased.
- **New potential strategies to adopt by Blackwell's sales team:** The aforementioned results opens new marketing possibilities. Extracting the associations with high confidence and low support (sold less often), it is possible to boost sales for those products (through advertisement and other sales actions). The increase in sales number of the said products will bring in increased revenue since we are confident the new transactions will increase sales volume. Blackwell could create some item packs of those items that are more often purchased at the same transaction, in order to also increase sales volume. Additionally, the rules that we found can be also helpful to provide recommendations to users when they purchase through the e-commerce platorm.

Limitations and observations:

Properties of the dataset:

1. The iMac is the product most bought, in 20% of all transactions. This high number stands out considering the large variety of products, especially being the iMac a pricey product. If this number is representative of all sales throughout the year, then Electronidex is potentially profitable.
2. The mean of items bought per transaction is almost 5. Logically, we would say most people in the real world would buy 1 or 2 items per transactions most frequently, in an eletronics store.

Technical Report

Loading Packages and importing datasets

```
pacman::p_load(readr, rstudioapi, ggplot2, party, dplyr, arules, arulesViz,
    RColorBrewer, readxl, tidyr, Hmisc)
setwd("..")
trans <- read.transactions("./Datasets/ElectronidexTransactions2017.csv", format = "basket",
    sep = ",", rm.duplicates = TRUE)
labels <- read_excel("./Datasets/ElectronidexItems2017.xlsx")

results <- read.csv("./Datasets/results.csv")
```

Preprocessing

```
labels[is.na(labels)] <- "Unknown"
summary(trans)
```

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 125 columns (items) and a density of 0.03506172

most frequent items:
                   iMac                HP Laptop CYBERPOWER Gamer Desktop
                   2519                     1909                     1809
          Apple Earpods        Apple MacBook Air                  (Other)
                   1715                     1530                    33622

element (itemset/transaction) length distribution:
sizes
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
   2 2163 1647 1294 1021  856  646  540  439  353  247  171  119   77   72
  15   16   17   18   19   20   21   22   23   25   26   27   29   30
  56   41   26   20   10   10   10    5    3    1    1    3    1    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   2.000   3.000   4.383   6.000  30.000

includes extended item information - examples:
                            labels
1 1TB Portable External Hard Drive
2 2TB Portable External Hard Drive
3                   3-Button Mouse
```

```
trans@itemInfo$labels <- labels$ProductType
trans@itemInfo$category <- labels$Category

trans_matrix <- as(trans, "matrix")
trans_df <- as.data.frame(trans_matrix)
# Turning into a sparematrix of 1s and 0s.
```

```
for (i in 1:ncol(trans_df)) {
    trans_df[, i] <- as.integer(trans_df[, i])
}
```

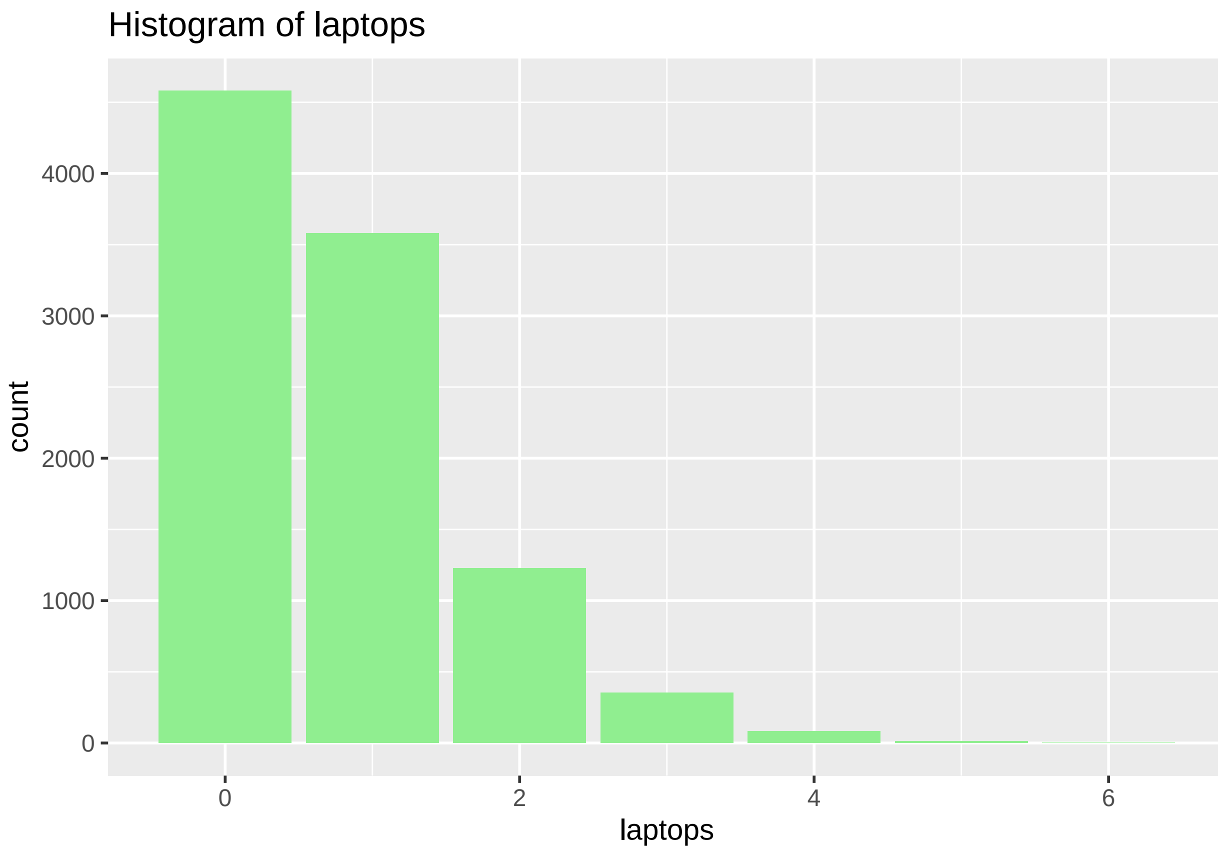
Feature Engineering

```
# Counting number of items
nitems <- c()
for (i in 1:nrow(trans_df)) {
    nitems <- c(nitems, sum(trans_df[i, ]))
}
trans_df$nitems <- nitems
trans_df$laptops <- trans_df[, which(colnames(trans_df) == "LG Touchscreen Laptop")] +
    trans_df[, which(colnames(trans_df) == "Acer Aspire")] +
trans_df[, which(colnames(trans_df) == "HP Laptop")] + trans_df[, which(colnames(trans_df) ==
    "ASUS Chromebook")] + trans_df[, which(colnames(trans_df) == "Apple Macbook Pro")] +
    trans_df[, which(colnames(trans_df) == "Apple MacBook Air")] + trans_df[,
    which(colnames(trans_df) == "Dell Laptop")] + trans_df[, which(colnames(trans_df) ==
    "Eluktronics Pro Gaming Laptop")] + trans_df[, which(colnames(trans_df) ==
    "Alienware AW17R4-7345SLV-PUS 17\" Laptop")] + trans_df[, which(colnames(trans_df) ==
    "HP Notebook Touchscreen Laptop PC")]
trans_df$desktop <- trans_df[, which(colnames(trans_df) == "Lenovo Desktop Computer")] +
    trans_df[, which(colnames(trans_df) == "iMac")] + trans_df[, which(colnames(trans_df) ==
    "HP Desktop")] + trans_df[, which(colnames(trans_df) == "ASUS Desktop")] +
    trans_df[, which(colnames(trans_df) == "Dell Desktop")] + trans_df[, which(colnames(trans_df) ==
    "Intel Desktop")] + trans_df[, which(colnames(trans_df) == "Acer Desktop")] +
    trans_df[, which(colnames(trans_df) == "CYBERPOWER Gamer Desktop")] + trans_df[,
    which(colnames(trans_df) == "Dell 2 Desktop")]
trans_df$tablet <- trans_df[, which(colnames(trans_df) == "iPad")] + trans_df[,
    which(colnames(trans_df) == "iPad Pro")] + trans_df[, which(colnames(trans_df) ==
    "Fire HD Tablet")] + trans_df[, which(colnames(trans_df) == "Samsung Galaxy Tab")] +
    trans_df[, which(colnames(trans_df) == "Kindle")]
trans_df$printer <- trans_df$`Epson Printer` + trans_df$`HP Wireless Printer` +
    trans_df$`Canon Office Printer` + trans_df$`Brother Printer` + trans_df$`DYMO Label Manker`
trans_df$nmain <- trans_df$printer + trans_df$laptops + trans_df$desktop + trans_df$tablet
trans_df$ncomp <- trans_df$nitems - trans_df$nmain
trans_df$value <- 10 * trans_df$nmain + trans_df$ncomp
```

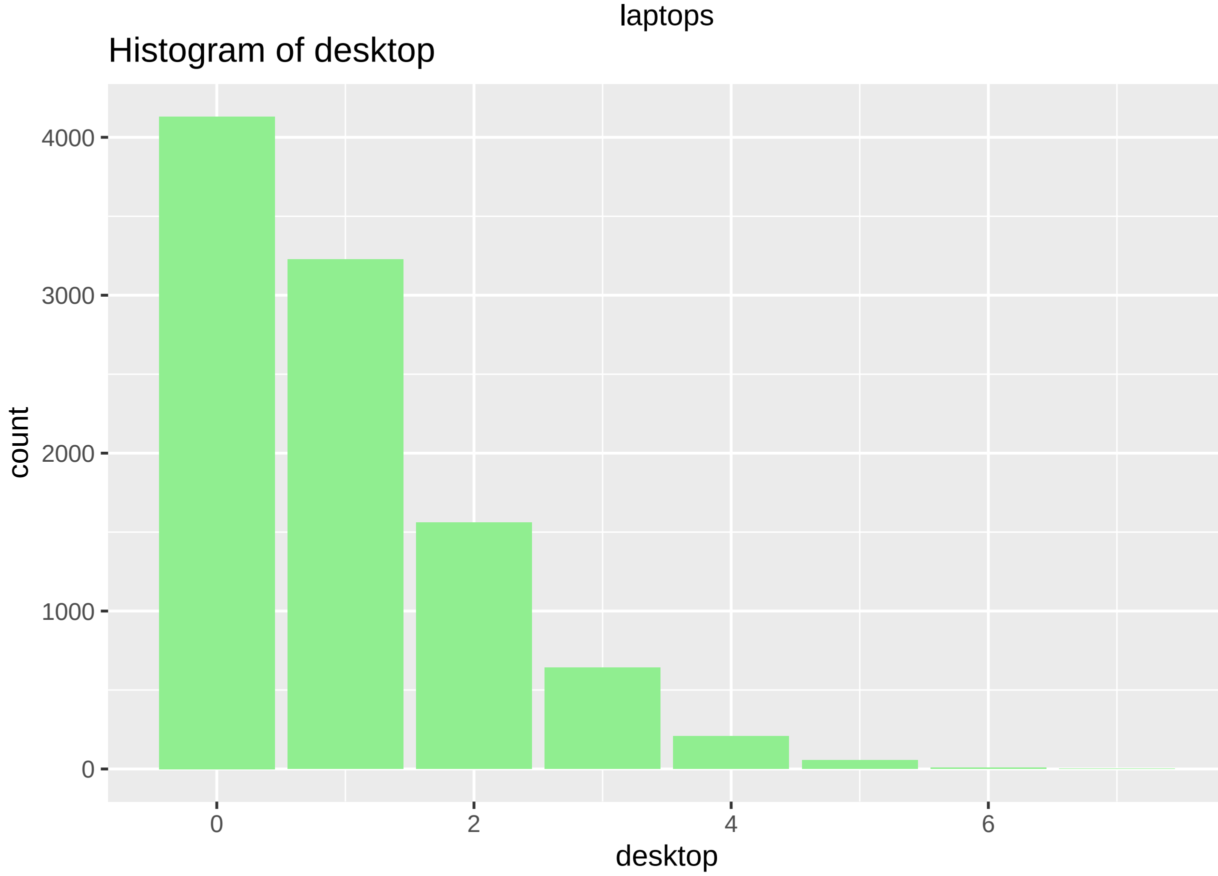
Visualizations

Here we make several plots to see the distribution of our variables and also see the frequency of the products bought.

```
products <- c("laptops", "desktop","printer","tablet")
for (i in products){
  print(ggplot(trans_df, aes_string(x = i)) +
          geom_bar(fill = "lightgreen", bins = 100) +

      ggtitle(paste("Histogram of",i)))
}
```

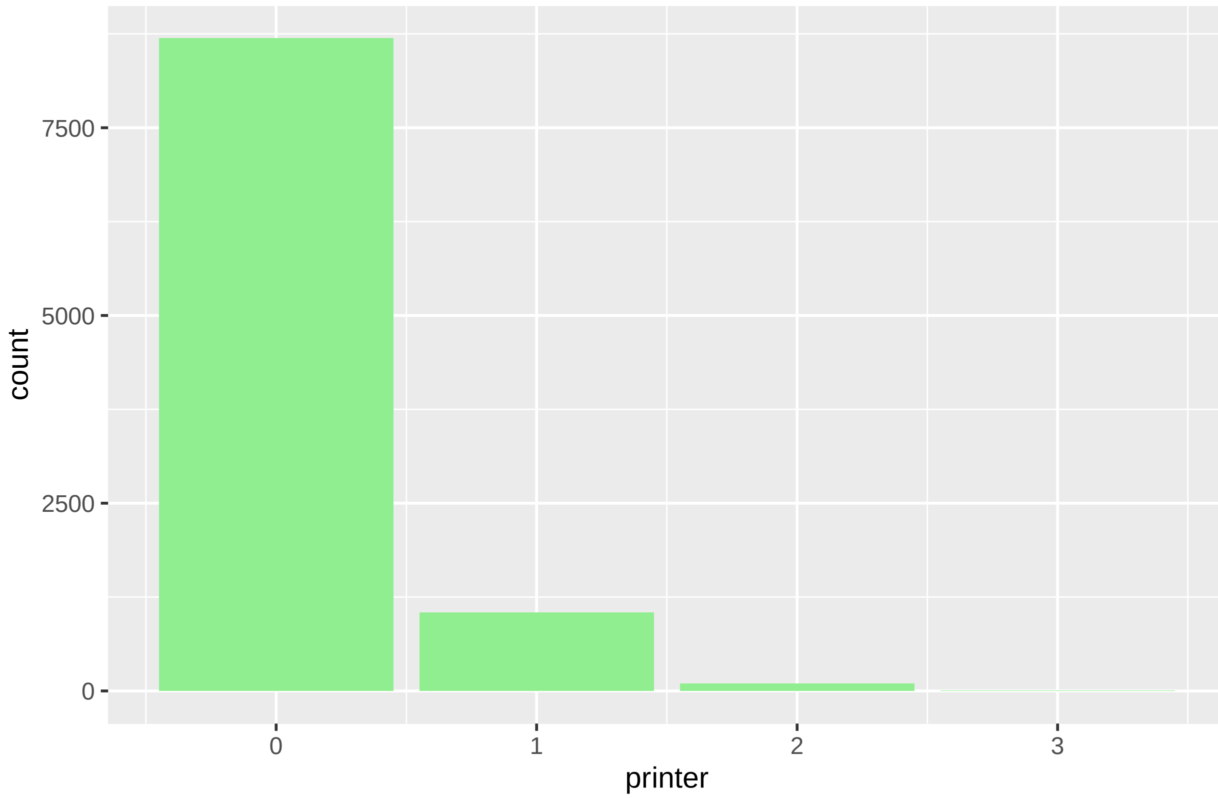


Histogram of laptops
count
4000
3000
2000
1000
0
0
2
4
6
laptops

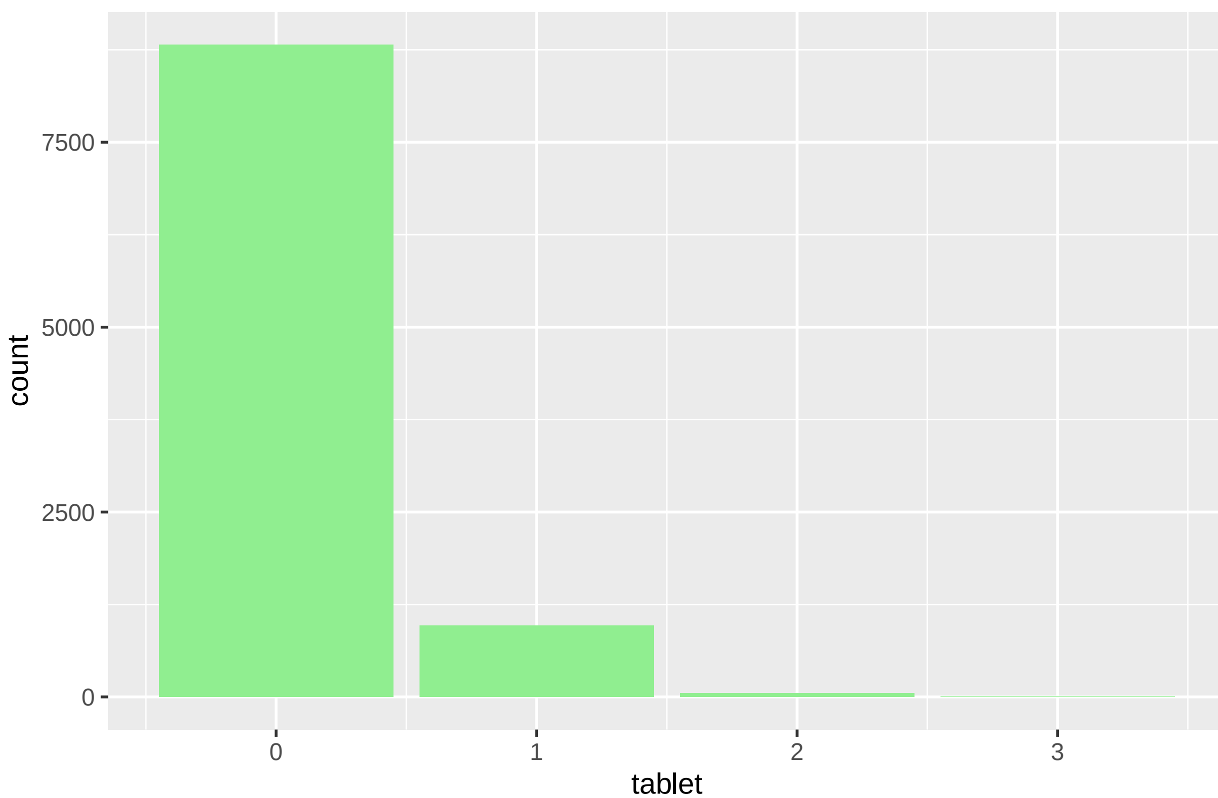


Histogram of desktop
count
4000
3000
2000
1000
0
0
2
4
6
desktop

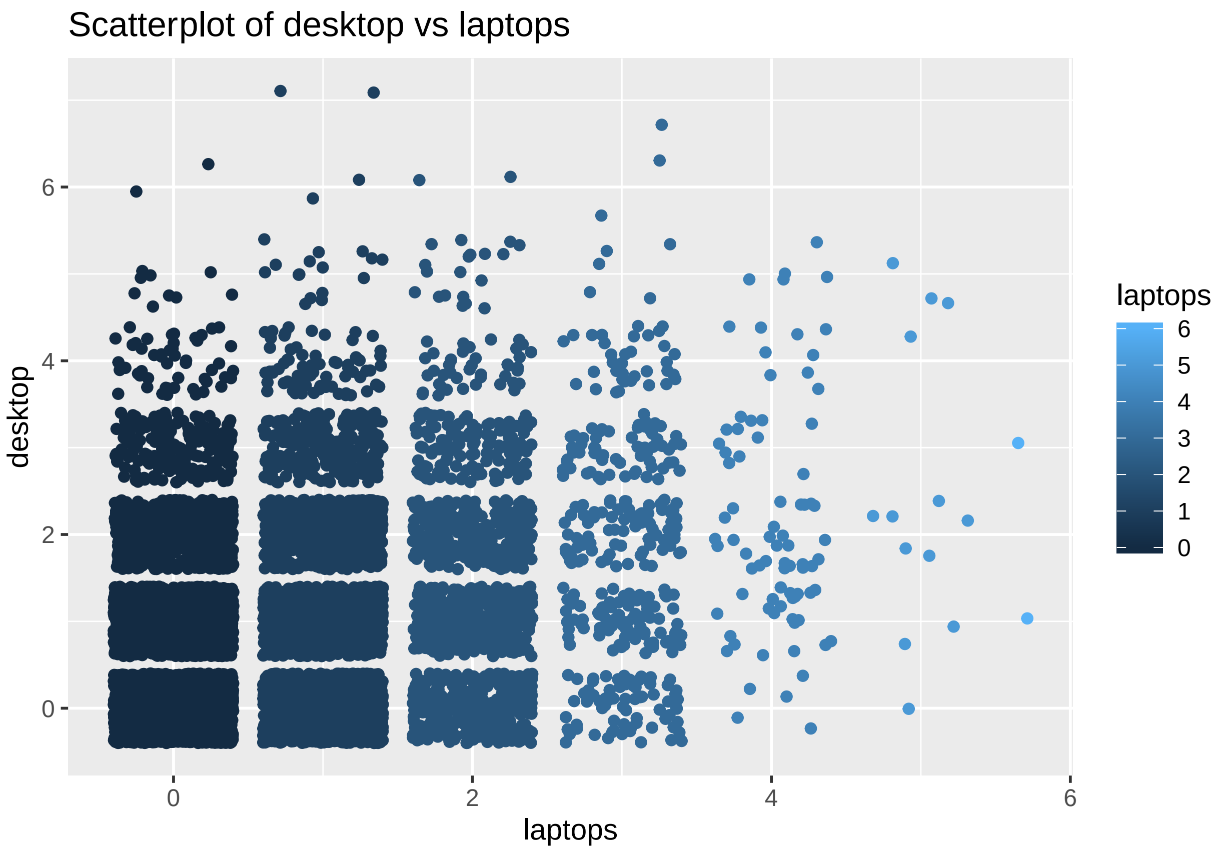
Histogram of printer



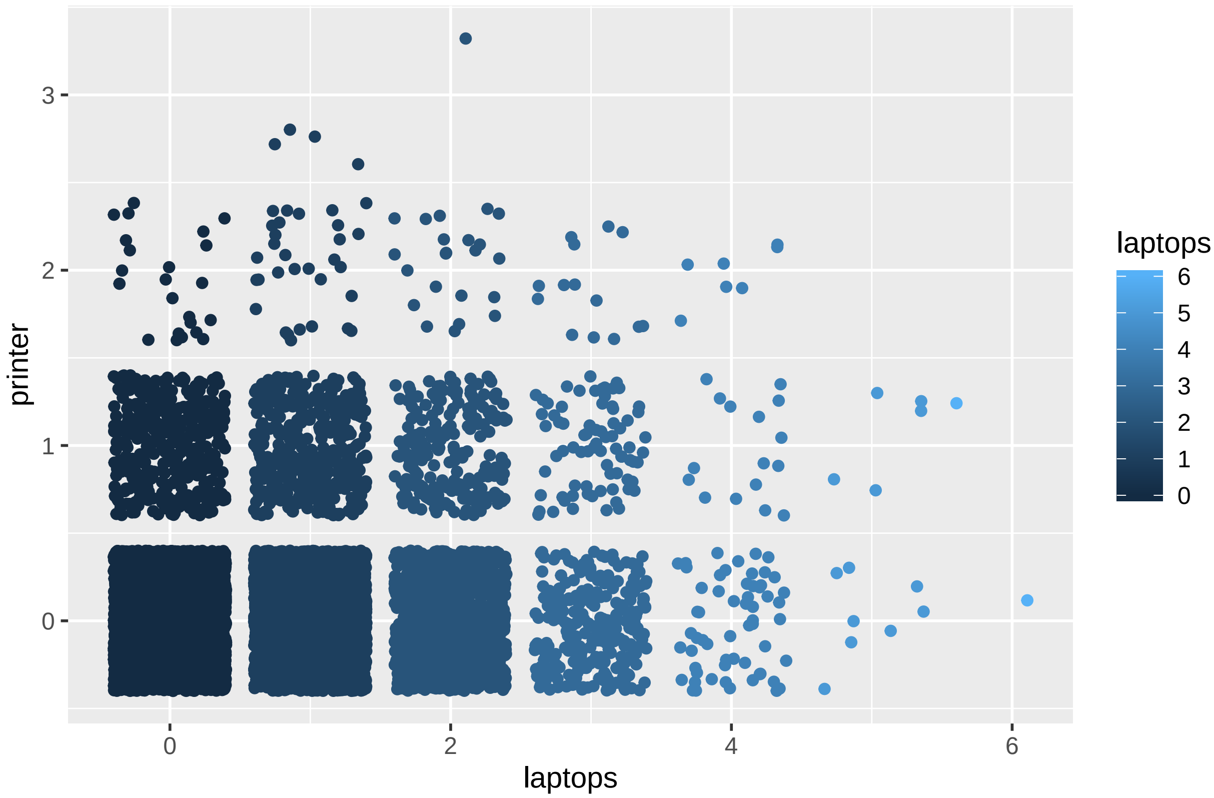
Histogram of tablet



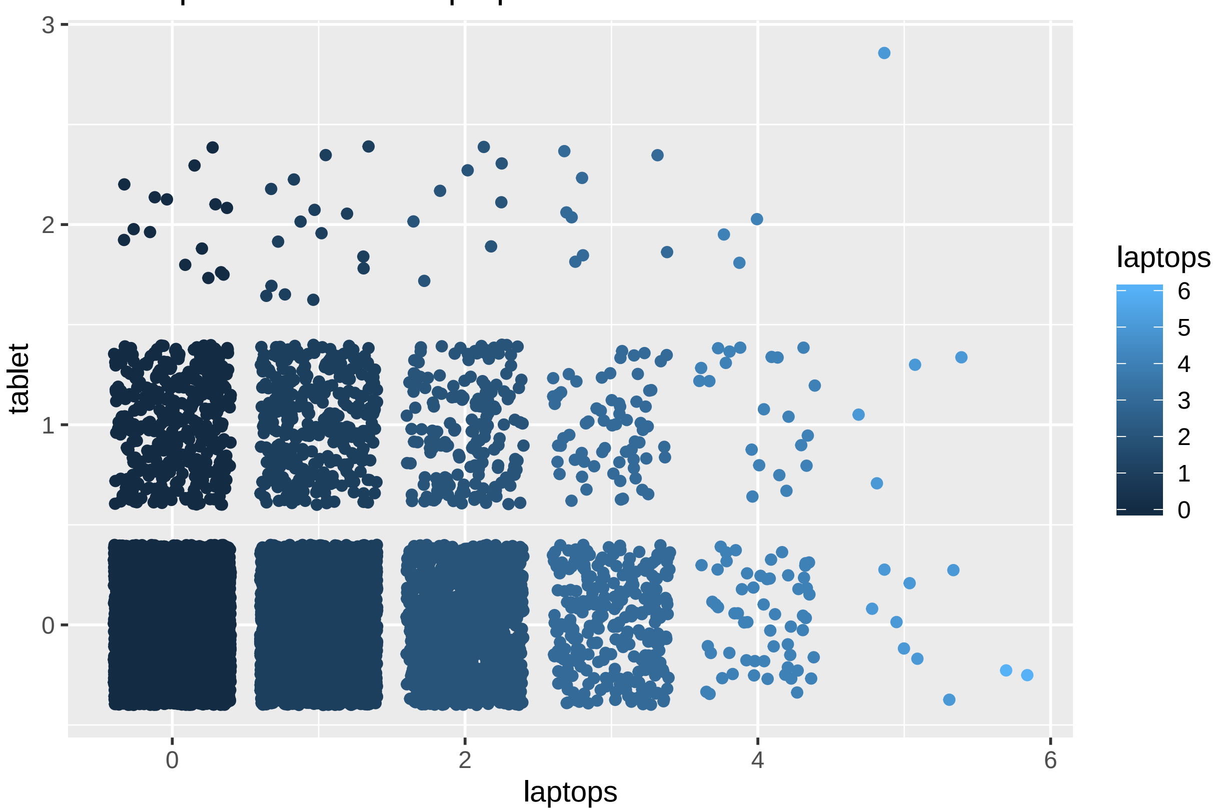
```r
for (i in products) {
    for (j in products[-1]) {
        if (i != j | j != i) {
            print(ggplot(trans_df, aes_string(x = i, y = j, color = i)) + geom_jitter() +
                ggtitle(paste("Scatterplot of", j, "vs", i)))

        }

    }
}
```

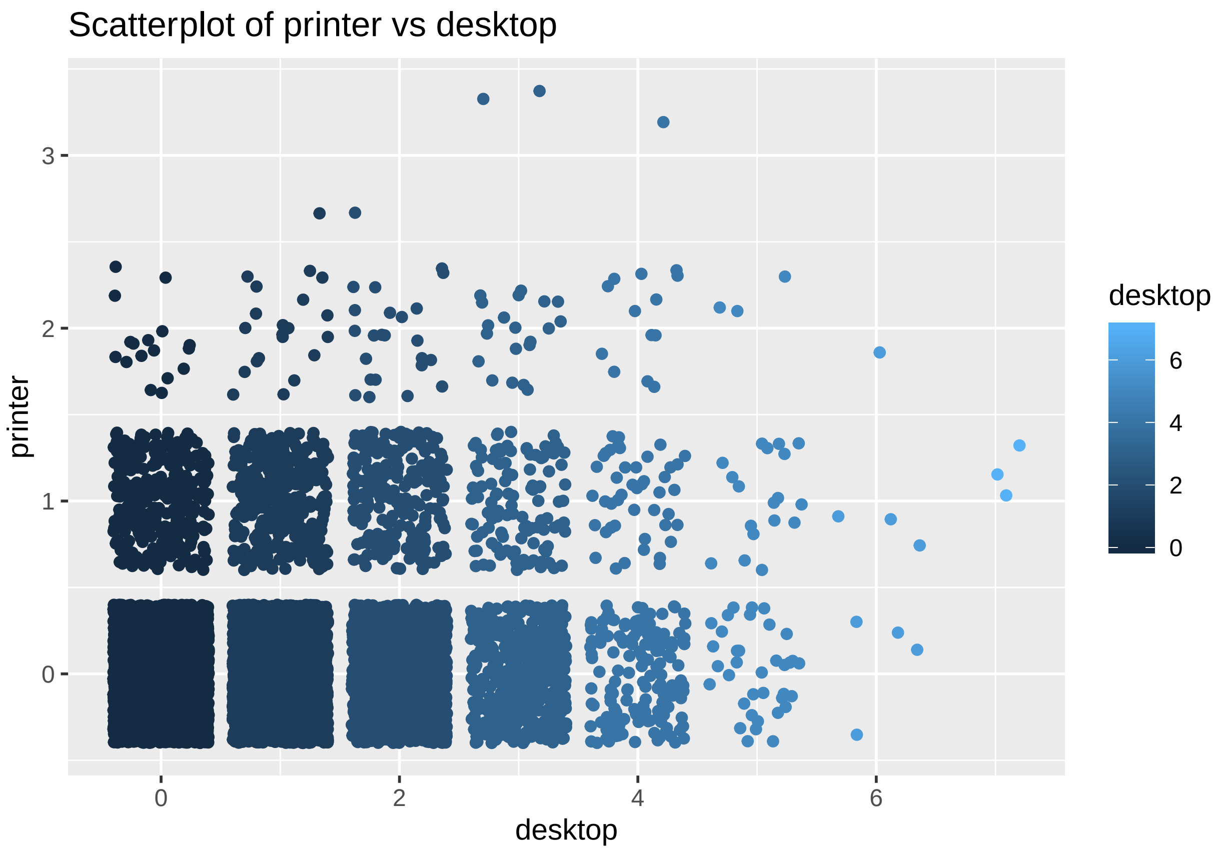


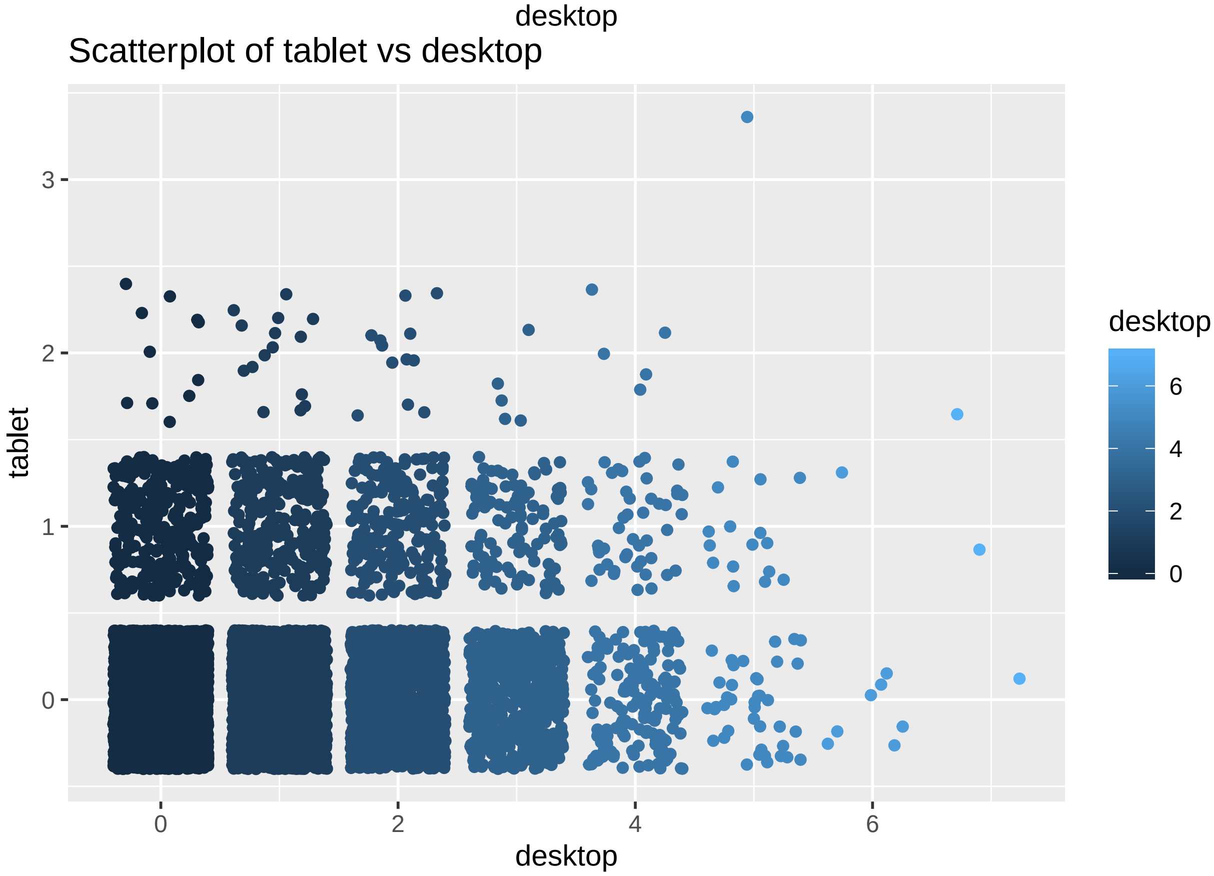
Scatterplot of printer vs laptops

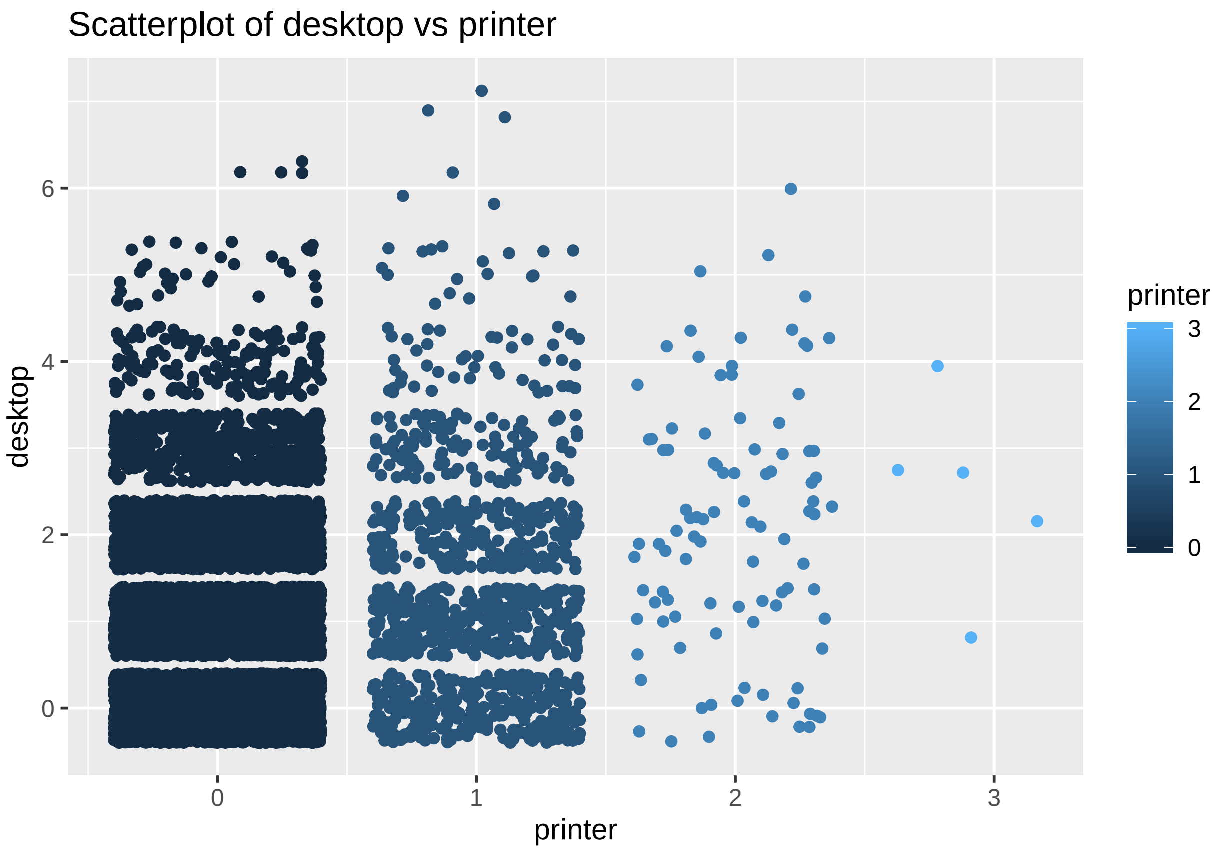


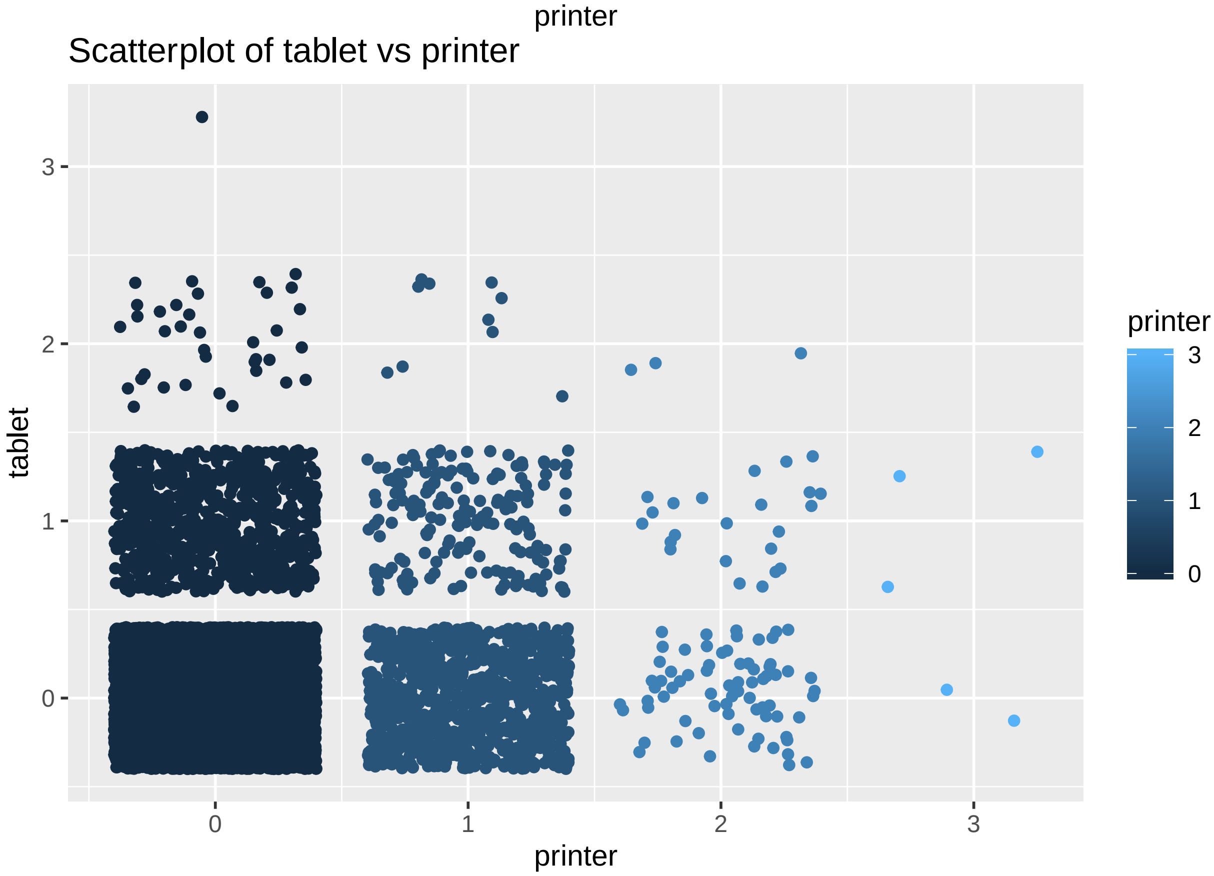
Scatterplot of tablet vs laptops

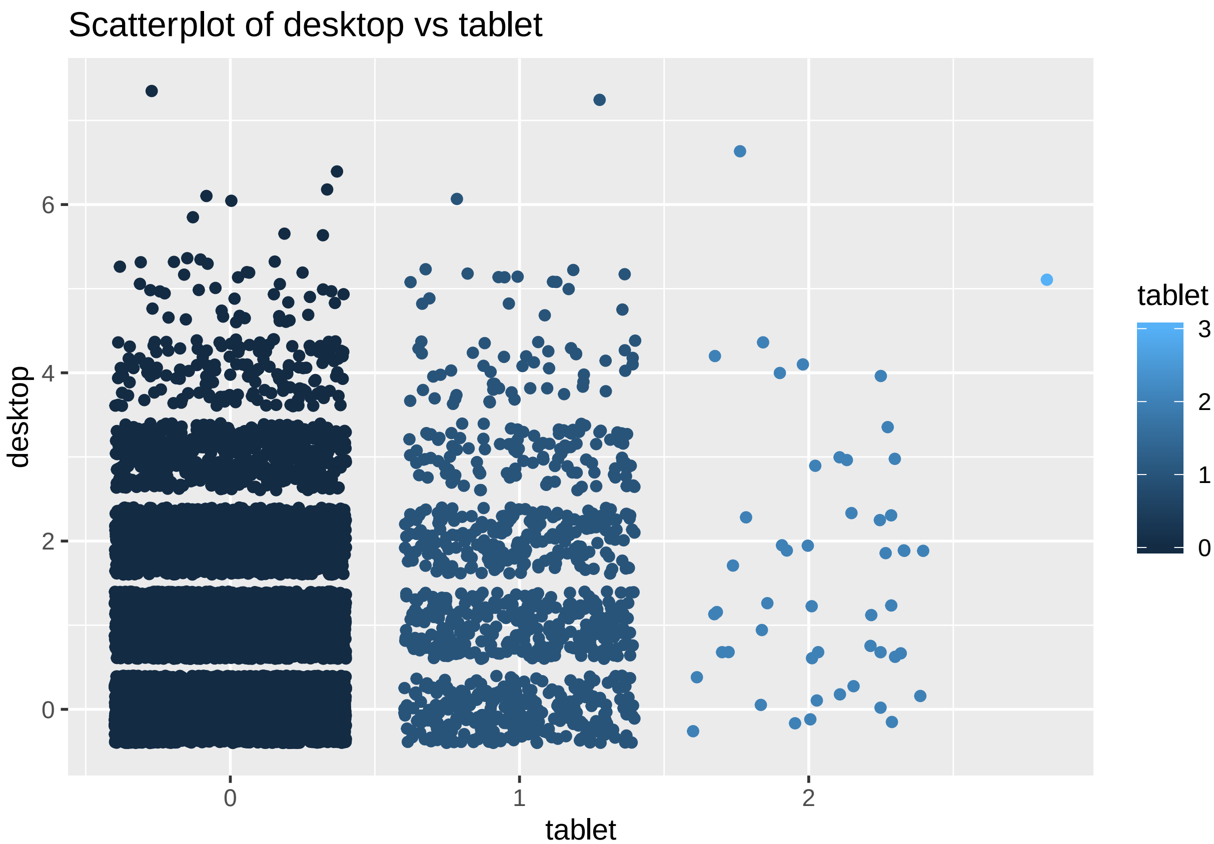


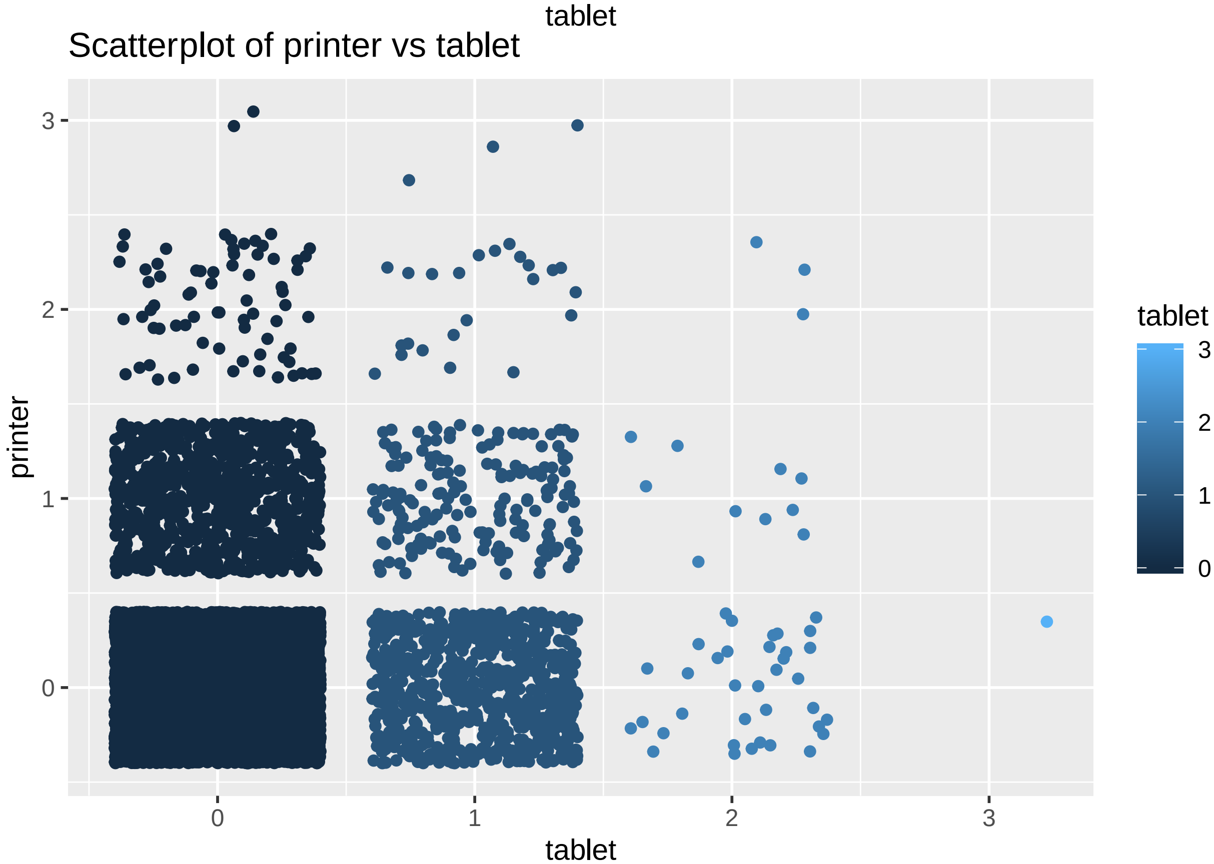

Scatterplot of printer vs desktop
printer
desktop
0
1
2
3
0
2
4
6
desktop
0
2
4
6


Scatterplot of tablet vs desktop
tablet
desktop
0
1
2
3
0
2
4
6
desktop
0
2
4
6

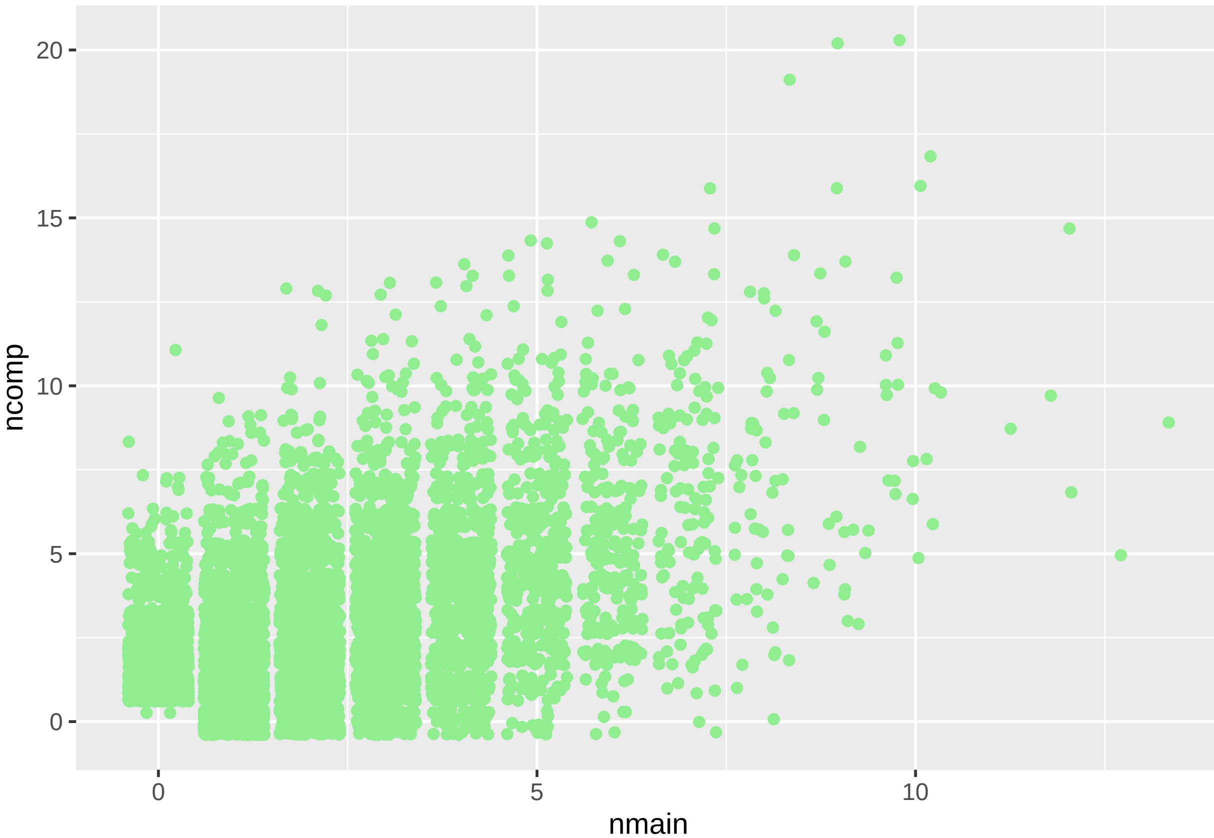

Scatterplot of desktop vs printer
desktop
printer
0
2
4
6
0
1
2
3
printer
3
2
1
0


Scatterplot of tablet vs printer
tablet
printer
0
1
2
3
0
1
2
3
printer
3
2
1
0


Scatterplot of desktop vs tablet
desktop
0
2
4
6
tablet
0
1
2
tablet
0
1
2
3


Scatterplot of printer vs tablet
printer
0
1
2
3
tablet
0
1
2
3
tablet
0
1
2
3

```r
ggplot(trans_df, aes(x = nmain, y = ncomp)) + geom_jitter(color = "lightgreen")
```

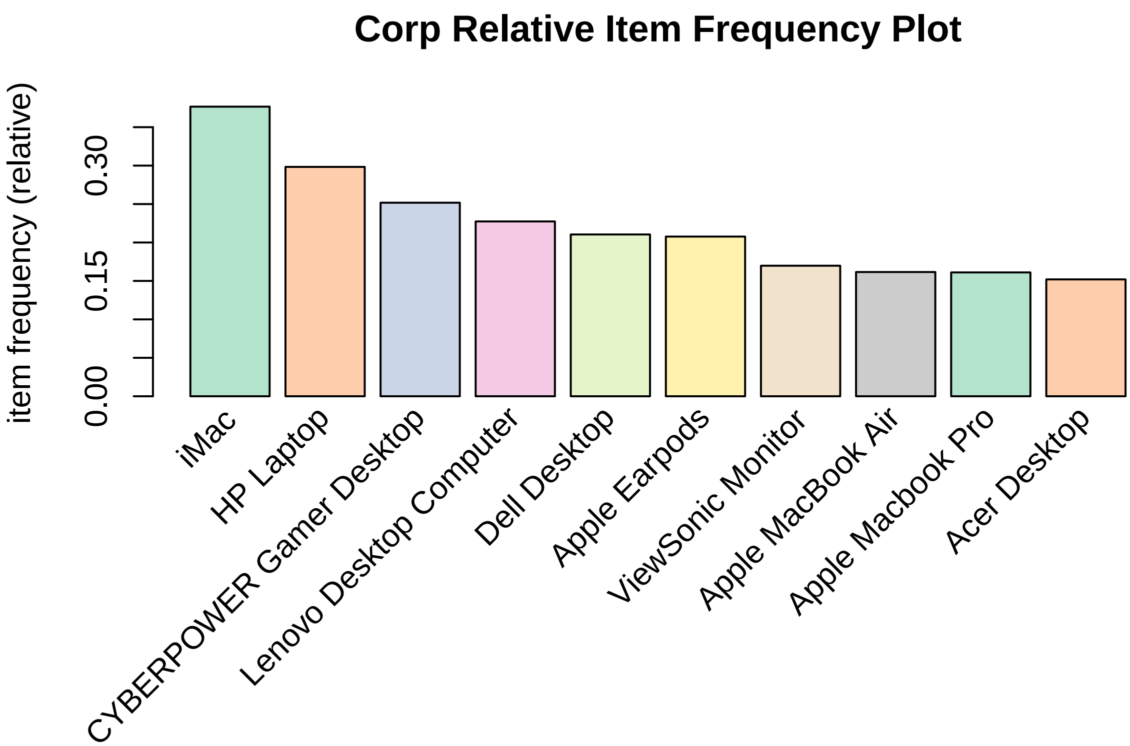


Splitting dataframe between corporates and retailers

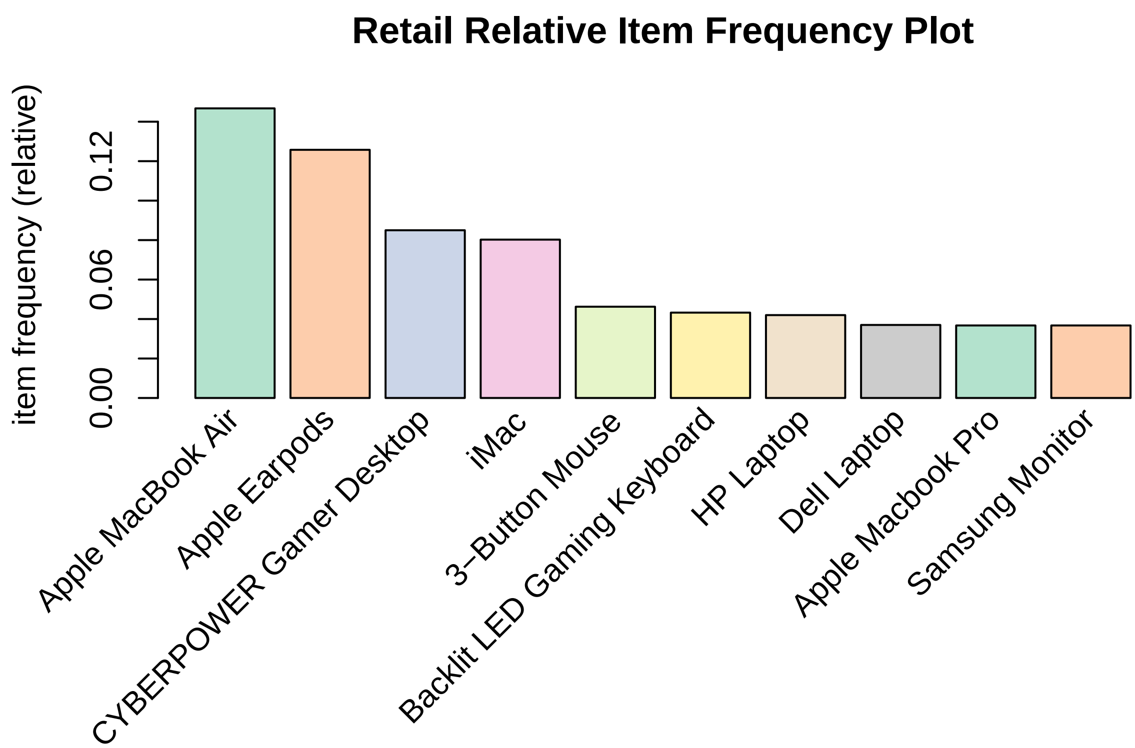
We'll split the dataframe to obtain the rules for each diferent types of costumers.

```r
# Filtering. Corporate will be those transactions with more than 2 main
# products and 3 complements.
corporate <- filter(trans_df, nmain >= 2 | ncomp >= 3)
# Cleaning the new columns we created before
corporate <- corporate[, -which(colnames(corporate) %in% c("laptops", "desktop",
    "printer", "tablet", "nitems", "nmain", "value", "ncomp"))]
# Filtering. Retailers will be those transactions with less than 2 main
# products and 3 complements.
retailer <- filter(trans_df, nmain <= 1 & ncomp <= 2)
# Cleaning
retailer <- retailer[, -which(colnames(retailer) %in% c("laptops", "desktop",
    "printer", "tablet", "nitems", "nmain", "value", "ncomp"))]
# Transforming the dataframe into a transaction objecte
trans_corp <- as(corporate == 1, "transactions")
trans_retail <- as(retailer == 1, "transactions")
# Inserting labels and the level category
trans_corp@itemInfo$labels <- labels$ProductType
trans_corp@itemInfo$category <- labels$Category
trans_retail@itemInfo$labels <- labels$ProductType
trans_retail@itemInfo$category <- labels$Category
```

```r
itemFrequencyPlot(trans_corp, topN = 10, type = "relative", col = brewer.pal(8,
    "Pastel2"), main = "Corp Relative Item Frequency Plot")
```




```r
itemFrequencyPlot(trans_retail, topN = 10, type = "relative", col = brewer.pal(8,
    "Pastel2"), main = "Retail Relative Item Frequency Plot")
```



Creating rules via apriori algorithm

We will evaluate our rules by creating and average of the support, confidence and lift. First we'll rescale these three variables and then add them together.

```
# Function to create the average
averagetor <- function(corpro) {

    a <- scale(corpro[, which(colnames(corpro) %in% c("support", "confidence",
        "lift"))])

    average <- a[, which(colnames(a) == "support")] + a[, which(colnames(a) ==
        "confidence")] + a[, which(colnames(a) == "lift")]
    corpro <- cbind(corpro, average)
    corpro <- corpro[order(-average), ]

    return(corpro)
}
```

Rules for products in corporate transactions

```
rules_corpro <- apriori(trans_corp, parameter = list(supp = 0.001, conf = 0.01,
    minlen = 2))
rules_corpro <- rules_corpro[!is.redundant(rules_corpro)]

summary(rules_corpro)
as(rules_corpro, "data.frame")
# The most populars products
```

```r
corpro_sup <- inspect(head(sort(rules_corpro, by = "support"), 10))
# Items that have high chances of being bought together
corpro_con <- inspect(head(sort(rules_corpro, by = "confidence"), 10))
# Lift
corpro_lift <- inspect(head(sort(rules_corpro, by = "lift"), 10))

corpro <- rbind(corpro_sup, corpro_con, corpro_lift)
corpro <- averagetor(corpro)
```

```r
corpro[, which(colnames(corpro) %in% c("lhs", "Var.2", "rhs", "average"))]
```

	lhs	Var.2	rhs	average
[1]	{HP Laptop}	=>	{iMac}	2.4153369
[9]	{ViewSonic Monitor}	=>	{iMac}	1.7127774
[2]	{iMac}	=>	{HP Laptop}	1.5000665
[3]	{Lenovo Desktop Computer}	=>	{iMac}	1.0105471
[7]	{Dell Desktop}	=>	{iMac}	0.6431045
[4]	{iMac}	=>	{Lenovo Desktop Computer}	-0.7698555
[10]	{iMac}	=>	{ViewSonic Monitor}	-1.0079418
[8]	{iMac}	=>	{Dell Desktop}	-1.3475267
[5]	{CYBERPOWER Gamer Desktop}	=>	{iMac}	-1.4284940
[6]	{iMac}	=>	{CYBERPOWER Gamer Desktop}	-2.7280144

Rules for categories in corporate transactions

```r
trans_corcat <- aggregate(trans_corp, by = "category")
rules_corcat <- apriori(trans_corcat, parameter = list(supp = 0.001, conf = 0.01,
    minlen = 2, maxlen = 4))
rules_corcat <- rules_corcat[!is.redundant(rules_corcat)]

summary(rules_corcat)
as(rules_corcat, "data.frame")
# The most populars categories
corcat_sup <- inspect(head(sort(rules_corcat, by = "support"), 10))
# Items that have high chances of being bought together
corcat_con <- inspect(head(sort(rules_corcat, by = "confidence"), 10))
# Lift
corcat_lift <- inspect(head(sort(rules_corcat, by = "lift"), 10))

corcat <- rbind(corcat_sup, corcat_con, corcat_lift)

corcat <- averagetor(corcat)
```

```r
corcat[, which(colnames(corcat) %in% c("lhs", "Var.2", "rhs", "average"))]
```

	lhs	Var.2	rhs	average
[1]	{Laptops}	=>	{Desktop}	2.0262154
[7]	{Laptops,Monitors}	=>	{Desktop}	1.7223116
[2]	{Desktop}	=>	{Laptops}	1.0851044

	lhs	Var.2	rhs	average
[3]	{Monitors}	=>	{Desktop}	0.7676997
[8]	{Desktop,Monitors}	=>	{Laptops}	0.7320255
[5]	{Monitors}	=>	{Laptops}	-0.7576991
[9]	{Desktop,Laptops}	=>	{Monitors}	-0.7578959
[10]	{Computer Mice}	=>	{Desktop}	-1.2056863
[4]	{Desktop}	=>	{Monitors}	-1.5134233
[6]	{Laptops}	=>	{Monitors}	-2.0986520

Rules for products in retailers transactions

```
rules_retpro <- apriori(trans_retail, parameter = list(supp = 0.001, conf = 0.01,
    minlen = 2))
rules_retpro <- rules_retpro[!is.redundant(rules_retpro)]

summary(rules_retpro)
as(rules_retpro, "data.frame")
# The most populars products
retpro_sup <- inspect(head(sort(rules_retpro, by = "support"), 10))
# Items that have high chances of being bought together
retpro_con <- inspect(head(sort(rules_retpro, by = "confidence"), 10))
# Lift
retpro_lift <- inspect(head(sort(rules_retpro, by = "lift"), 10))
retpro <- rbind(retpro_sup, retpro_con, retpro_lift)
retpro <- averagetor(retpro)
```

```
retpro[, which(colnames(retpro) %in% c("lhs", "Var.2", "rhs", "average"))]
```

	lhs	Var.2	rhs	average
[7]	{Samsung Monitor}	=>	{CYBERPOWER Gamer Desktop}	2.4223682
[1]	{CYBERPOWER Gamer Desktop}	=>	{Apple Earpods}	2.0435876
[2]	{Apple Earpods}	=>	{CYBERPOWER Gamer Desktop}	1.2268133
[5]	{3-Button Mouse}	=>	{Apple Earpods}	0.7512219
[9]	{Apple Macbook Pro}	=>	{Apple Earpods}	0.6750203
[8]	{CYBERPOWER Gamer Desktop}	=>	{Samsung Monitor}	0.4484622
[6]	{Apple Earpods}	=>	{3-Button Mouse}	-1.1497497
[10]	{Apple Earpods}	=>	{Apple Macbook Pro}	-1.6413135
[4]	{Apple Earpods}	=>	{Apple MacBook Air}	-2.3047997
[3]	{Apple MacBook Air}	=>	{Apple Earpods}	-2.4716106

Rules for categories in retailers transactions

```
trans_retcat <- aggregate(trans_retail, by = "category")
rules_retcat <- apriori(trans_retcat, parameter = list(supp = 0.001, conf = 0.01,
    minlen = 2, maxlen = 4))
rules_retcat <- rules_retcat[!is.redundant(rules_retcat)]
summary(rules_retcat)
as(rules_retcat, "data.frame")
# The most populars products
```

```r
retcat_sup <- inspect(head(sort(rules_retcat, by = "support"), 10))
# Items that have high chances of being bought together
retcat_con <- inspect(head(sort(rules_retcat, by = "confidence"), 10))
# Lift
retcat_lift <- inspect(head(sort(rules_retcat, by = "lift"), 10))

retcat <- rbind(retcat_sup, retcat_con, retcat_lift)
retcat <- averagetor(retcat)
```

```r
retcat[, which(colnames(retcat) %in% c("lhs", "Var.2", "rhs", "average"))]
```

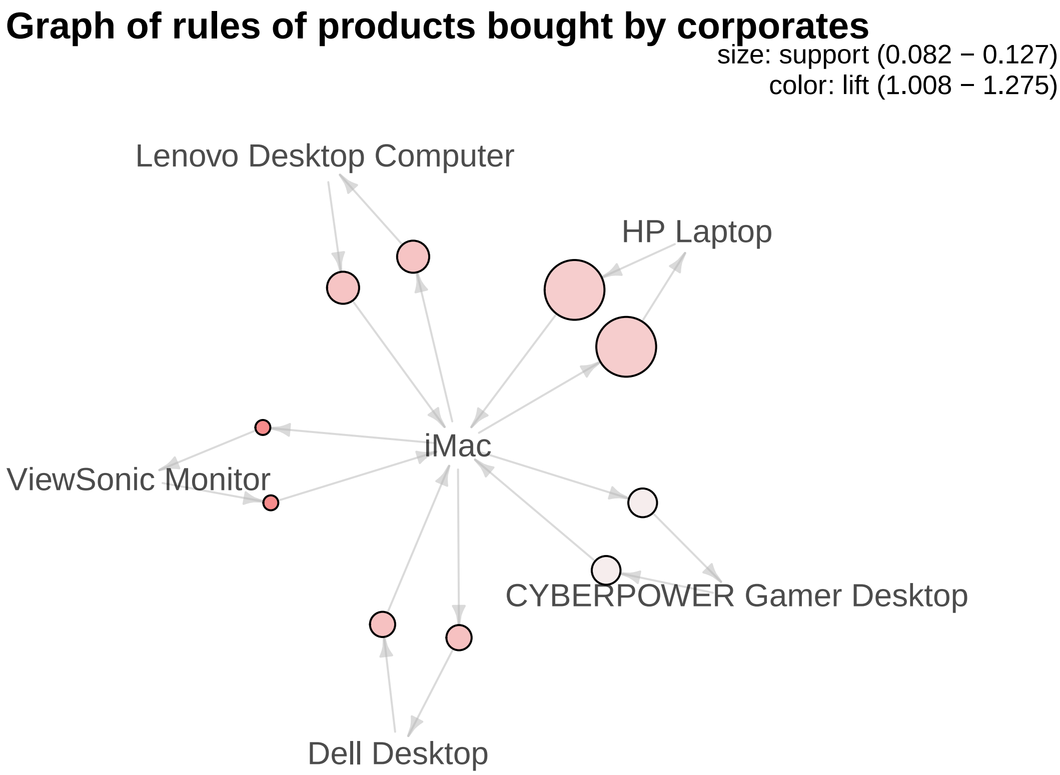
	lhs	Var.2	rhs	average
[1]	{Monitors}	=>	{Desktop}	4.7116644
[2]	{Desktop}	=>	{Monitors}	2.9661851
[3]	{Computer Mice}	=>	{Desktop}	1.1870590
[7]	{Keyboard}	=>	{Desktop}	0.8514579
[4]	{Desktop}	=>	{Computer Mice}	-0.7414707
[9]	{Active Headphones}	=>	{Laptops}	-0.9944768
[8]	{Desktop}	=>	{Keyboard}	-1.1448269
[5]	{Monitors}	=>	{Laptops}	-1.4346892
[10]	{Laptops}	=>	{Active Headphones}	-2.6141875
[6]	{Laptops}	=>	{Monitors}	-2.7867154

Rules visualizations

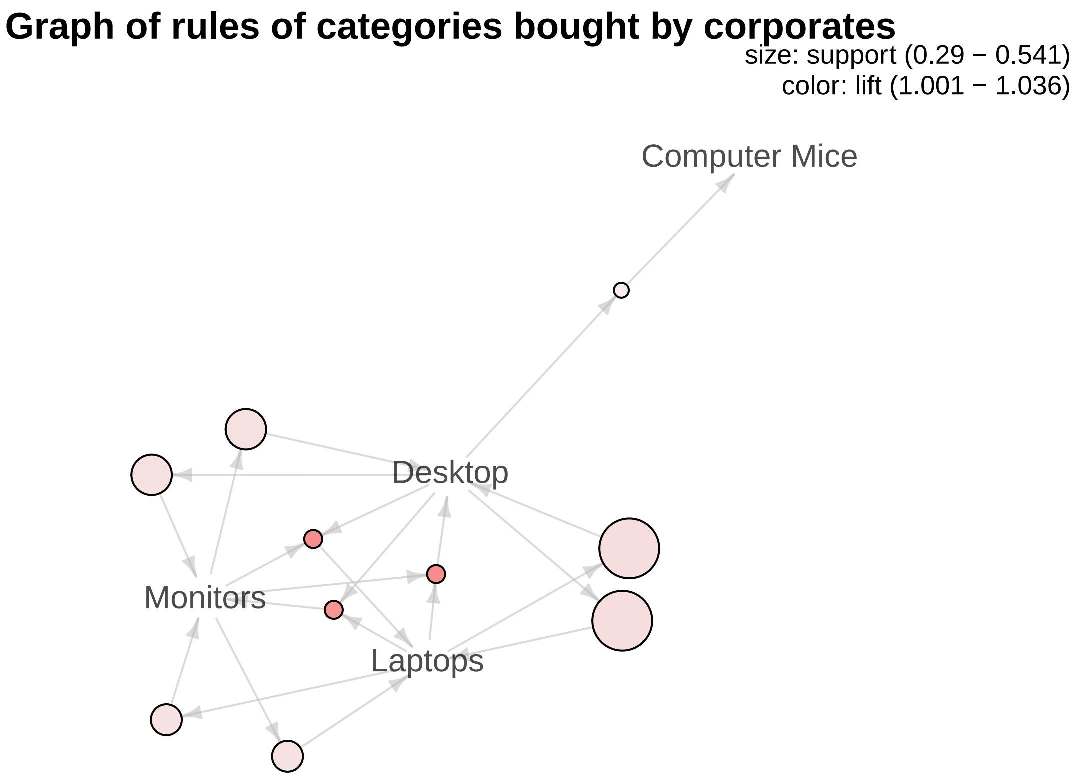
```r
methods <- c("graph", "scatterplot")
for (i in methods) {
    plot(rules_corpro, method = i, control = list(type = "items"), max = 10,
        main = paste(capitalize(i), "of rules of products bought by corporates"))
    plot(rules_corcat, method = i, control = list(type = "items"), max = 10,
        main = paste(capitalize(i), "of rules of categories bought by corporates"))
    plot(rules_retpro, method = i, control = list(type = "items"), max = 10,
        main = paste(capitalize(i), "of rules of products bought by retailers"))
    plot(rules_retcat, method = i, control = list(type = "items"), max = 10,
        main = paste(capitalize(i), "of rules of categories bought by retailers"))
}
```

```
Available control parameters (with default values):
main     =  Graph for 10 rules
nodeColors   =  c("#66CC6680", "#9999CC80")
nodeCol  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF"
edgeCol  =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF"
alpha    =  0.5
cex  =  1
itemLabels   =  TRUE
labelCol     =  #000000B3
measureLabels    =  FALSE
precision    =  3
layout   =  NULL
```

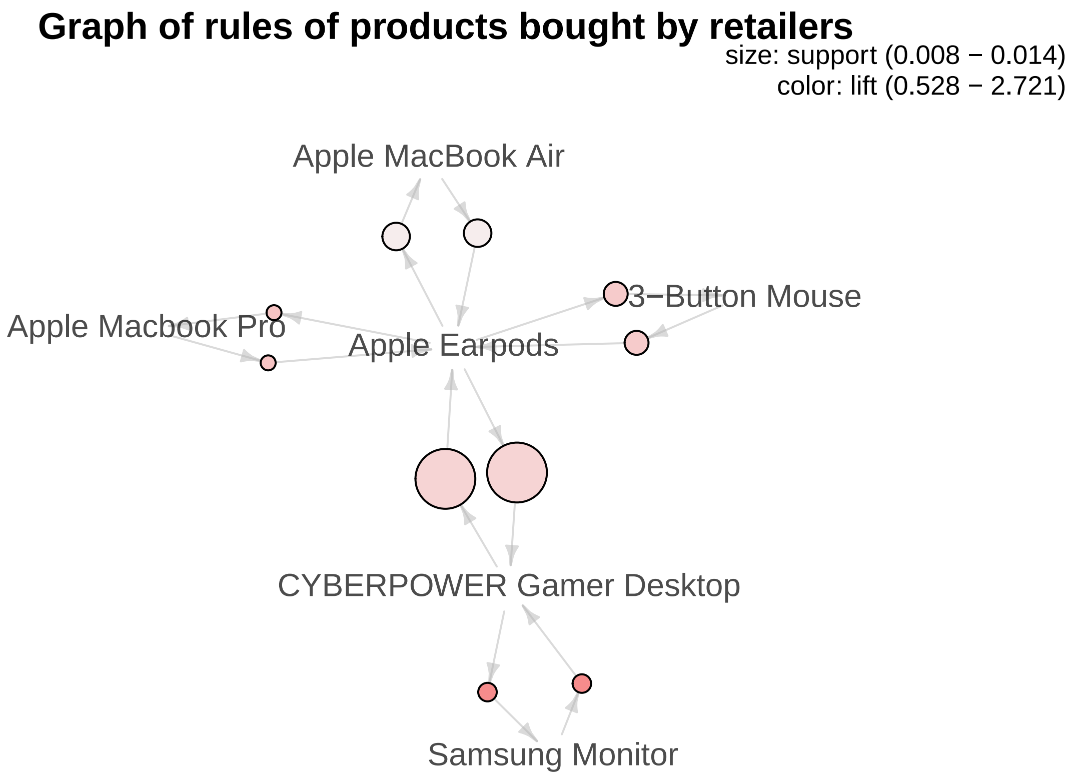
```
layoutParams     =  list()
arrowSize    =  0.5
engine   =  igraph
plot     =  TRUE
plot_options     =  list()
max  =  100
verbose  =  FALSE
```



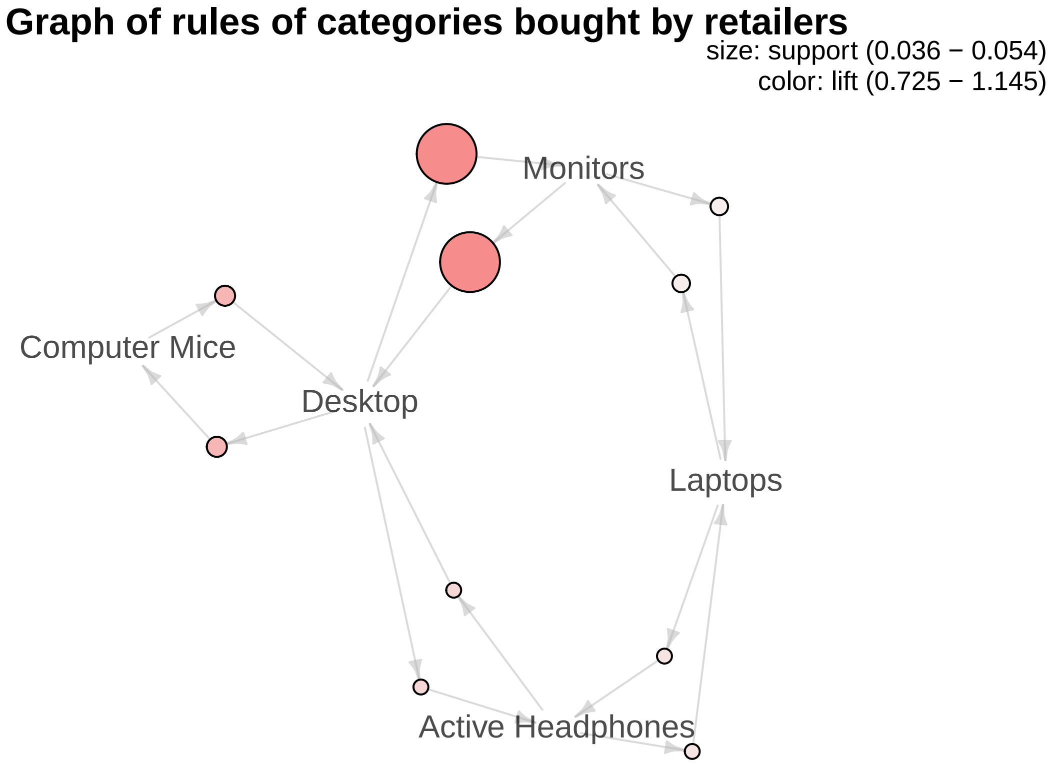

```
Available control parameters (with default values):
main     =  Graph for 10 rules
nodeColors    =  c("#66CC6680", "#9999CC80")
nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF"
edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF"
alpha    =  0.5
cex   =  1
itemLabels    =  TRUE
labelCol     =  #000000B3
measureLabels     =  FALSE
precision    =  3
layout   =  NULL
layoutParams     =  list()
arrowSize    =  0.5
engine   =  igraph
plot     =  TRUE
plot_options     =  list()
max  =  100
verbose  =  FALSE
```



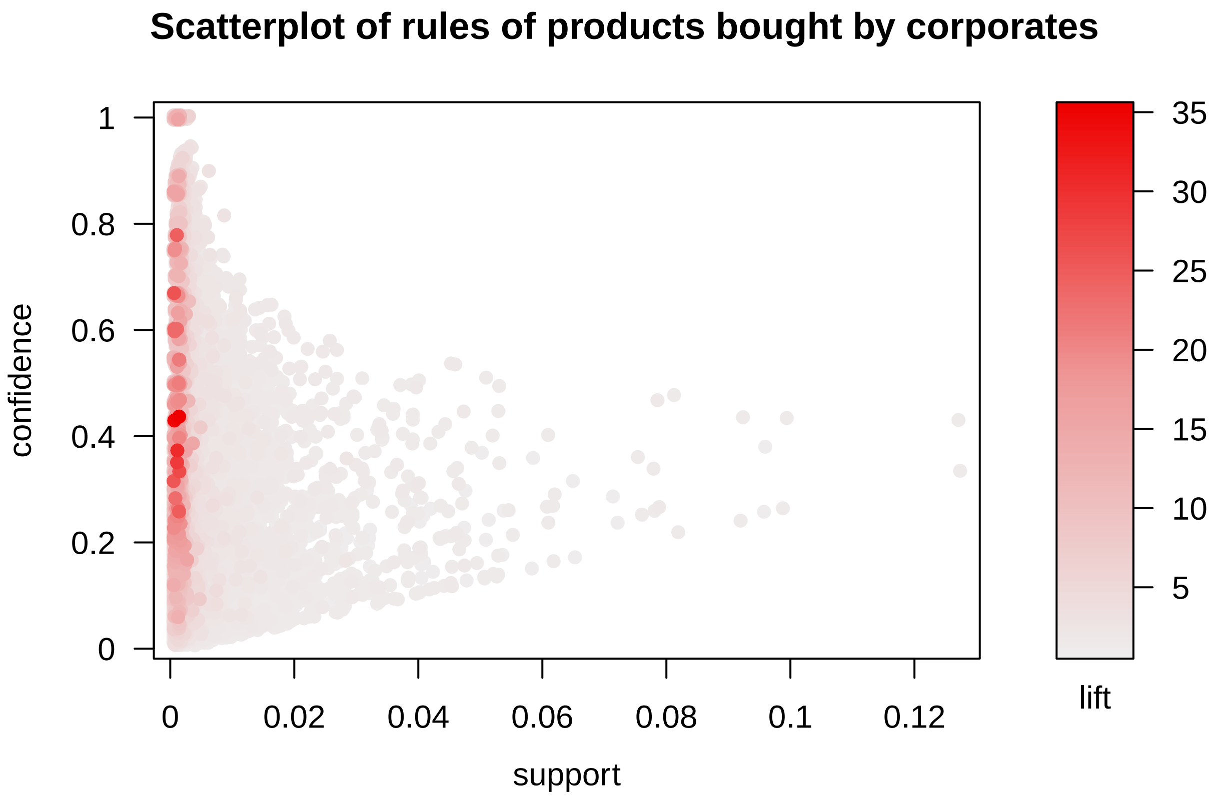

```
Available control parameters (with default values):
main     =  Graph for 10 rules
nodeColors   =  c("#66CC6680", "#9999CC80")
nodeCol  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF"
edgeCol  =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF"
alpha    =  0.5
cex  =  1
itemLabels   =  TRUE
labelCol     =  #000000B3
measureLabels    =  FALSE
precision    =  3
layout   =  NULL
layoutParams     =  list()
arrowSize    =  0.5
engine   =  igraph
plot     =  TRUE
plot_options     =  list()
max  =  100
verbose  =  FALSE
```



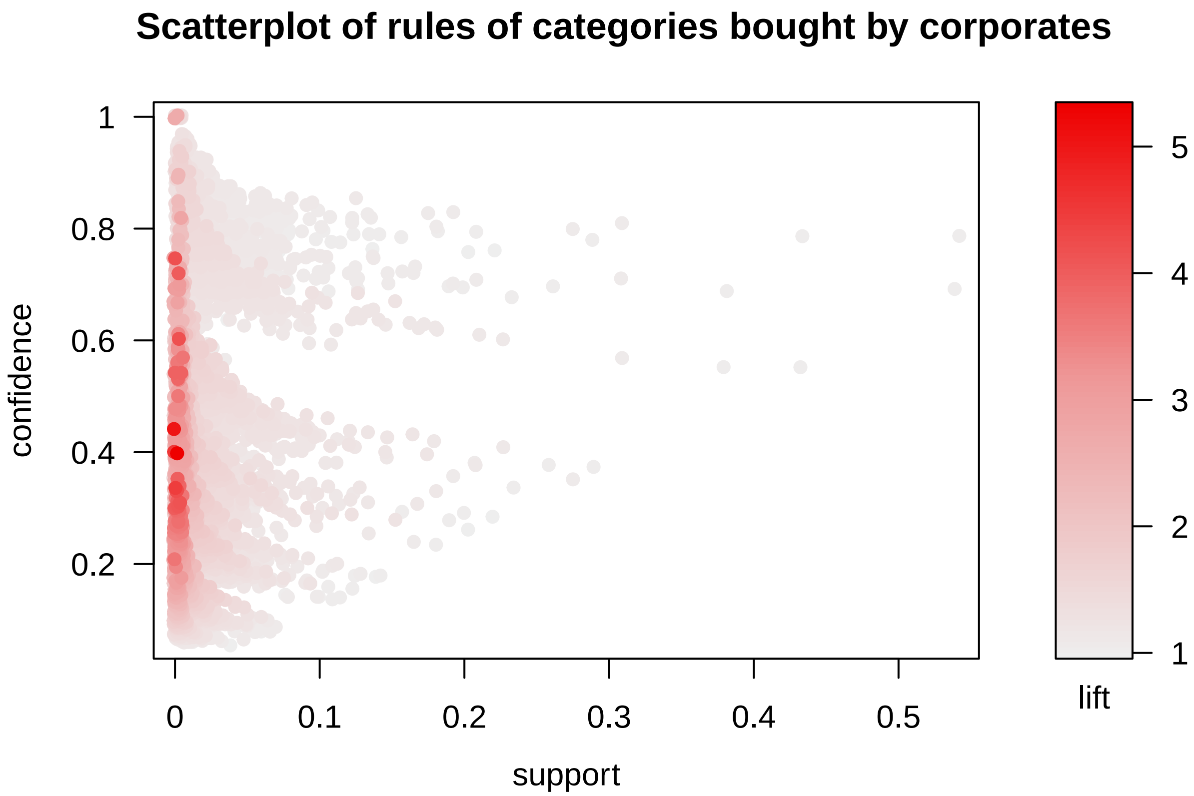

```
Available control parameters (with default values):
main     =  Graph for 10 rules
nodeColors   =  c("#66CC6680", "#9999CC80")
nodeCol  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF"
edgeCol  =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF"
alpha    =  0.5
cex  =  1
itemLabels   =  TRUE
labelCol     =  #000000B3
measureLabels    =  FALSE
precision    =  3
layout   =  NULL
layoutParams     =  list()
arrowSize    =  0.5
engine   =  igraph
plot     =  TRUE
plot_options     =  list()
max  =  100
verbose  =  FALSE
```



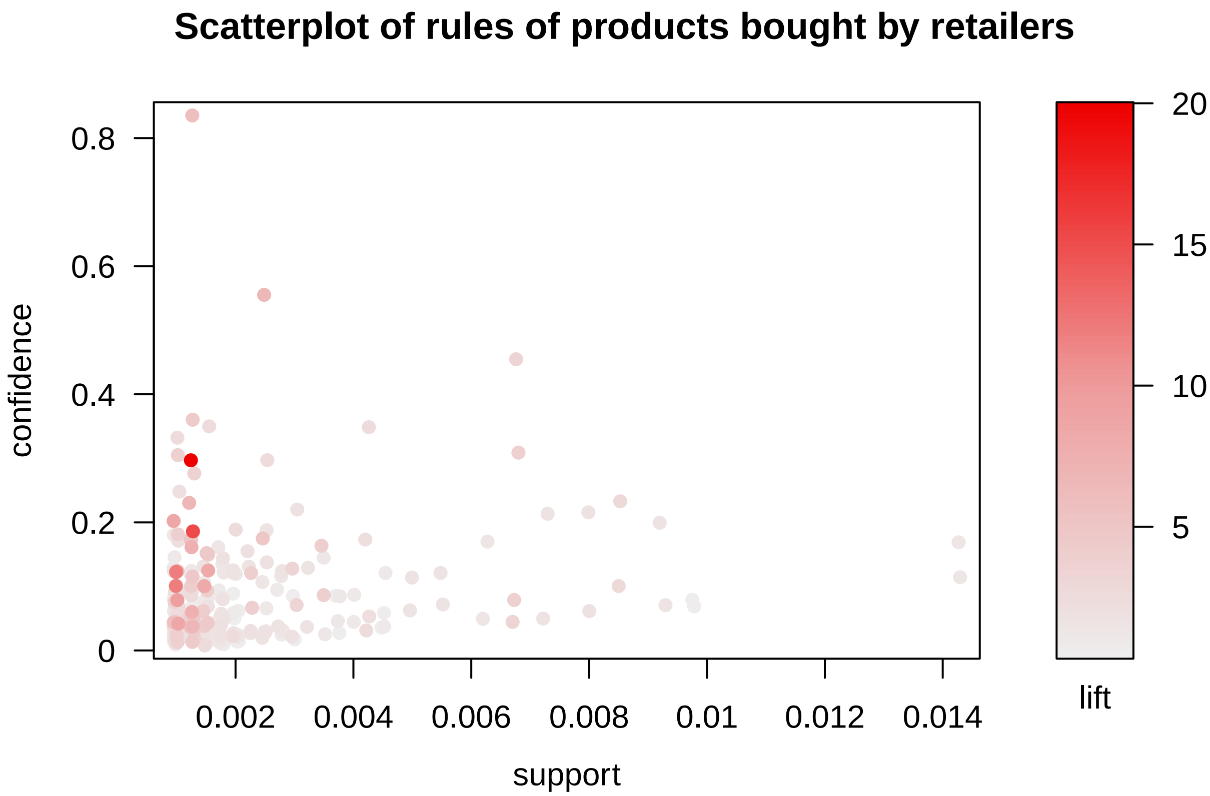

```
Available control parameters (with default values):
main     =  Scatter plot for 109029 rules
engine   =  default
pch  =  19
cex  =  0.5
xlim     =  NULL
ylim     =  NULL
zlim     =  NULL
alpha    =  NULL
col  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#
newpage  =  TRUE
jitter   =  NA
verbose  =  FALSE
```



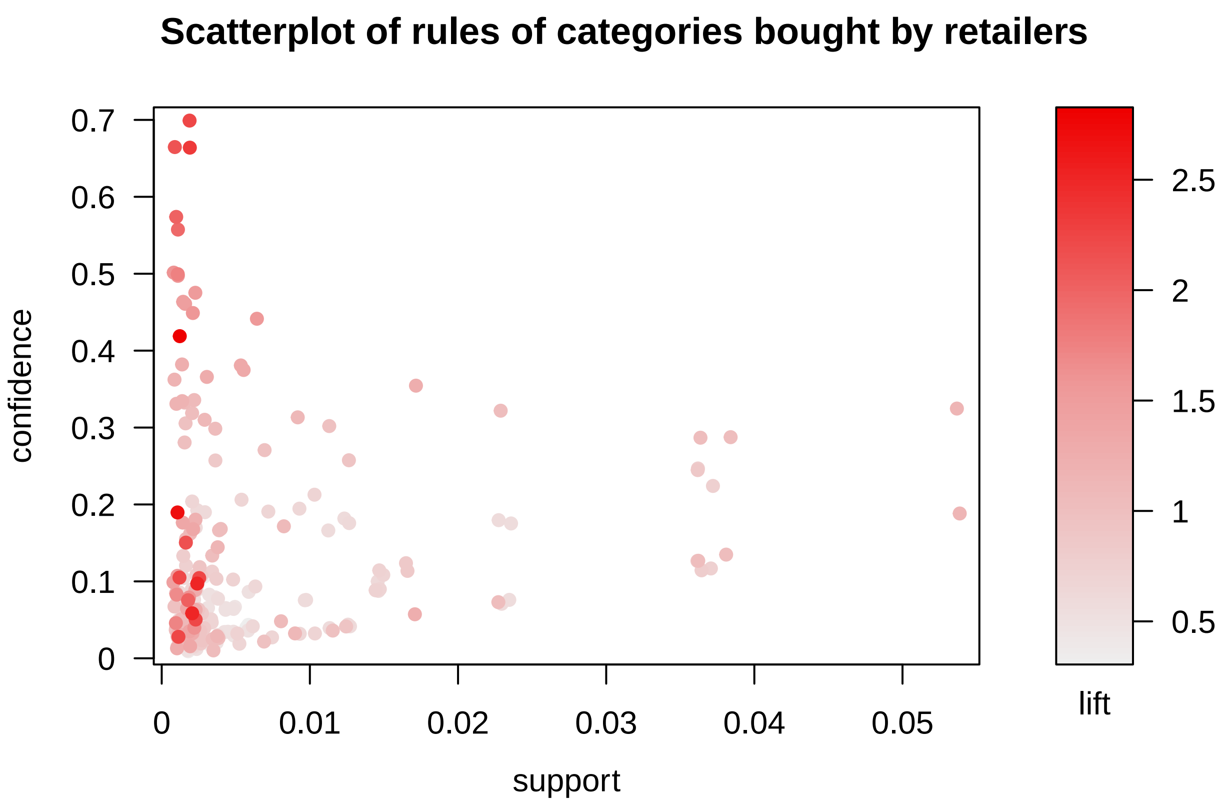

```
Available control parameters (with default values):
main     =  Scatter plot for 9246 rules
engine   =  default
pch  =  19
cex  =  0.5
xlim     =  NULL
ylim     =  NULL
zlim     =  NULL
alpha    =  NULL
col  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#E
newpage  =  TRUE
jitter   =  NA
verbose  =  FALSE
```




```
Available control parameters (with default values):
main     =  Scatter plot for 306 rules
engine   =  default
pch  =  19
cex  =  0.5
xlim     =  NULL
ylim     =  NULL
zlim     =  NULL
alpha    =  NULL
col  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#E
newpage  =  TRUE
jitter   =  NA
verbose  =  FALSE
```




```
Available control parameters (with default values):
main     =  Scatter plot for 235 rules
engine   =  default
pch  =  19
cex  =  0.5
xlim     =  NULL
ylim     =  NULL
zlim     =  NULL
alpha    =  NULL
col  =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#
newpage  =  TRUE
jitter   =  NA
verbose  =  FALSE
```




```
results <- rbind(retpro[1:3, which(colnames(retpro) %in% c("lhs", "Var.2", "rhs",
    "average"))], retcat[1:3, which(colnames(retcat) %in% c("lhs", "Var.2",
    "rhs", "average"))], corpro[1:3, which(colnames(corpro) %in% c("lhs", "Var.2",
    "rhs", "average"))], corcat[1:3, which(colnames(corcat) %in% c("lhs", "Var.2",
    "rhs", "average"))])

write.csv(results, "results.csv")
```