Forecasting Sales Demand for Inventory

Naiara de Almeida Pantuza

23/01/2020

Description

Planning a celebration is a balancing act of preparing just enough food to go around without being stuck eating the same leftovers for the next week. The key is anticipating how many guests will come. Grupo Bimbo must weigh similar considerations as it strives to meet daily consumer demand for fresh bakery products on the shelves of over 1 million stores along its 45,000 routes across Mexico. In this project, I am suppose to develop a model to accurately forecast inventory demand based on historical sales data.

I viewed features histograms provided automatically by the Microsoft Azure "Unpack Zipped Datasets" function and noticed that the numbers of records for each week are similar. Therefore, I divided dataset into 7 parts, take samples of same size from each part and join it in a single training dataset for data exploratory analysis. This process was executed by a python script.

```
# Setting work directory
setwd("/home/naiara/Documentos/DataScience/FCD/BigDataRAzure/Project_Inventory_Demand")
```

Feature Engineering

Getting dataset and treating it. In the "cliente_tabla" table I will consider the first line of each Cliente_ID if it is duplicated.

I got a sample of the training data set, as well as the remaining datasets available in kaggle. Then, I combine the training, sample, customer, product and region datasets into a single dataset to facilitate exploratory analysis. I did the same for the training and submission dataset (dataset with the correct test predictions).

```
# Getting and visualizing all datasets
# Get the first 10.000.000 rows of train dataset
train <- read.csv(unz("data/sample_train.zip", "sample_train.csv"))

# Getting customer data
customer <- read.csv(unz("data/cliente_tabla.csv.zip", "cliente_tabla.csv"))

# Getting product data
product <- read.csv(unz("data/producto_tabla.csv.zip", "producto_tabla.csv"))

# Getting test data
test <- read.csv(unz("data/test.csv.zip", "test.csv"))

# Getting town/state data
townState <- read.csv(unz("data/town_state.csv.zip", "town_state.csv"))

# Getting sample_submission data
```

```r
submission <- read.csv(unz("data/sample_submission.csv.zip", "sample_submission.csv"))

# Visualizing each dataset
head(train)
```

```
##   X Unnamed..0 Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID
## 1 0    5279830      3       1550        1     1205    2334521        1150
## 2 1     470313      3       1120        1     1468      17199        1212
## 3 2    5917554      3       1620        1     2113    8107299       31310
## 4 3    8499126      3       2036        1     1260    1344810        1242
## 5 4    6993670      3       1914        1     2161    1817134       36339
## 6 5    2140920      3       1224        1     4471     440523       43069
##   Venta_uni_hoy Venta_hoy Dev_uni_proxima Dev_proxima Demanda_uni_equil
## 1             1     14.76               0          0                 1
## 2             4     33.52               0          0                 4
## 3             6     37.50               0          0                 6
## 4             5     38.20               0          0                 5
## 5             2     19.98               0          0                 2
## 6             3     22.23               0          0                 3
```

```r
head(customer)
```

```
##   Cliente_ID                          NombreCliente
## 1          0                             SIN NOMBRE
## 2          1                        OXXO XINANTECATL
## 3          2                             SIN NOMBRE
## 4          3                              EL MORENO
## 5          4 SDN SER  DE ALIM  CUERPO SA CIA  DE INT
## 6          4    SDN SER DE ALIM CUERPO SA CIA DE INT
```

```r
head(product)
```

```
##   Producto_ID                         NombreProducto
## 1           0                     NO IDENTIFICADO 0
## 2           9              Capuccino Moka 750g NES 9
## 3          41 Bimbollos Ext sAjonjoli 6p 480g BIM 41
## 4          53         Burritos Sincro 170g CU LON 53
## 5          72    Div Tira Mini Doradita 4p 45g TR 72
## 6          73      Pan Multigrano Linaza 540g BIM 73
```

```r
head(test)
```

```
##   id Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID
## 1  0     11       4037        1     2209    4639078       35305
## 2  1     11       2237        1     1226    4705135        1238
## 3  2     10       2045        1     2831    4549769       32940
## 4  3     11       1227        1     4448    4717855       43066
## 5  4     11       1219        1     1130     966351        1277
## 6  5     11       1146        4     6601    1741414         972
```

```r
head(townState)
```

```
##   Agencia_ID                  Town            State
## 1       1110    2008 AG. LAGO FILT     MÉXICO, D.F.
## 2       1111 2002 AG. AZCAPOTZALCO     MÉXICO, D.F.
## 3       1112   2004 AG. CUAUTITLAN ESTADO DE MÉXICO
## 4       1113    2008 AG. LAGO FILT     MÉXICO, D.F.
```

```
## 5       1114  2029 AG.IZTAPALAPA 2     MÉXICO, D.F.
## 6       1116  2011 AG. SAN ANTONIO     MÉXICO, D.F.
```

```
head(submission)
```

```
##   id Demanda_uni_equil
## 1  0                 7
## 2  1                 7
## 3  2                 7
## 4  3                 7
## 5  4                 7
## 6  5                 7
```

```
# Eliminating columns created
train$X <- NULL
train$Unnamed..0 <- NULL

# Getting first row for each duplication
#Loading required library
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Eliminating duplicate ids in the clients' table
# Getting first row for each Cliente_ID
customer <- customer %>%
  group_by(Cliente_ID) %>%
  slice(1)

# Merging datasets for analysis
train_data <- merge(train, customer, by = "Cliente_ID")
train_data <- merge(train_data, product, by = "Producto_ID")
train_data <- merge(train_data, townState, by = "Agencia_ID")

# Merging test dataset and respective demand
test_data <- merge(test, submission, by = "id")
test_data <- merge(test_data, townState, by = "Agencia_ID")

# Saving merged datasets on csv files
# write.csv(train_data, "data/train_data.zip")
# write.csv(test_data, "data/test_data.zip")
```

I also cleaned and transformed the data before analyzing it.

```
# Removing extra variable
# train_data$X <- NULL
test_data$X <- NULL
test_data$id <- NULL
```

```r
# Checking test and train data
str(train_data)
```

```
## 'data.frame':    9800000 obs. of  15 variables:
##  $ Agencia_ID       : int  1110 1110 1110 1110 1110 1110 1110 1110 1110 1110 ...
##  $ Producto_ID      : int  35144 45143 32819 41938 1230 1146 3333 1242 5380 30574 ...
##  $ Cliente_ID       : int  2331691 1265976 1682455 1750334 1690065 50986 1239291 1780659 2048160 736
##  $ Semana           : int  8 4 4 8 7 7 8 7 9 7 ...
##  $ Canal_ID         : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ Ruta_SAK         : int  3303 3306 3316 3317 3303 3316 3316 3318 3318 3318 ...
##  $ Venta_uni_hoy    : int  160 40 8 2 8 1 2 8 2 25 ...
##  $ Venta_hoy        : num  3280 30 71.1 19.8 133.4 ...
##  $ Dev_uni_proxima  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Dev_proxima      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Demanda_uni_equil: int  160 40 8 2 8 1 2 8 2 25 ...
##  $ NombreCliente    : Factor w/ 311155 levels "007","056 THE AIRPORT MARKET",..: 79925 265106 256366
##  $ NombreProducto   : Factor w/ 2592 levels "100pct Whole Wheat 680g MTA ORO 43111",..: 2579 1733 54
##  $ Town             : Factor w/ 260 levels "2001 AG. ATIZAPAN",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ State            : Factor w/ 33 levels "AGUASCALIENTES",..: 15 15 15 15 15 15 15 15 15 15 ...
```

```r
str(test_data)
```

```
## 'data.frame':    6999251 obs. of  9 variables:
##  $ Agencia_ID       : int  1110 1110 1110 1110 1110 1110 1110 1110 1110 1110 ...
##  $ Semana           : int  10 11 10 10 11 10 10 11 11 10 ...
##  $ Canal_ID         : int  7 7 7 7 7 7 7 7 11 11 ...
##  $ Ruta_SAK         : int  3315 3316 3303 3312 3304 3309 3302 3316 3502 3502 ...
##  $ Cliente_ID       : int  1022370 1614912 1690065 1267811 20946 2055320 2052099 1546506 4337478 432
##  $ Producto_ID      : int  2233 1146 4245 2233 641 641 1242 1309 47611 1700 ...
##  $ Demanda_uni_equil: int  7 7 7 7 7 7 7 7 7 7 ...
##  $ Town             : Factor w/ 260 levels "2001 AG. ATIZAPAN",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ State            : Factor w/ 33 levels "AGUASCALIENTES",..: 15 15 15 15 15 15 15 15 15 15 ...
```

```r
head(test_data)
```

```
##   Agencia_ID Semana Canal_ID Ruta_SAK Cliente_ID Producto_ID Demanda_uni_equil
## 1       1110     10        7     3315    1022370        2233                 7
## 2       1110     11        7     3316    1614912        1146                 7
## 3       1110     10        7     3303    1690065        4245                 7
## 4       1110     10        7     3312    1267811        2233                 7
## 5       1110     11        7     3304      20946         641                 7
## 6       1110     10        7     3309    2055320         641                 7
##                 Town        State
## 1 2008 AG. LAGO FILT MÉXICO, D.F.
## 2 2008 AG. LAGO FILT MÉXICO, D.F.
## 3 2008 AG. LAGO FILT MÉXICO, D.F.
## 4 2008 AG. LAGO FILT MÉXICO, D.F.
## 5 2008 AG. LAGO FILT MÉXICO, D.F.
## 6 2008 AG. LAGO FILT MÉXICO, D.F.
```

```r
head(train_data)
```

```
##   Agencia_ID Producto_ID Cliente_ID Semana Canal_ID Ruta_SAK Venta_uni_hoy
## 1       1110       35144    2331691      8        7     3303           160
## 2       1110       45143    1265976      4        7     3306            40
## 3       1110       32819    1682455      4        7     3316             8
```

```
## 4       1110       41938    1750334      8        7     3317             2
## 5       1110        1230    1690065      7        7     3303             8
## 6       1110        1146      50986      7        7     3316             1
##   Venta_hoy Dev_uni_proxima Dev_proxima Demanda_uni_equil
## 1   3280.00               0           0               160
## 2     30.00               0           0                40
## 3     71.12               0           0                 8
## 4     19.82               0           0                 2
## 5    133.36               0           0                 8
## 6     21.39               0           0                 1
##                               NombreCliente
## 1                          DESAYUNOS GUADALUPE
## 2                                      ROCIO
## 3                         RAFAEL CAFE SA DE CV
## 4                            CAMIONETA RODANTE
## 5                                    DOC MAZK
## 6 DESARROLLADORA GASTRONOMICA NABANI SA DE CV
##                               NombreProducto               Town        State
## 1 Wonder 100pct con Ajonjoli 567g WON 35144 2008 AG. LAGO FILT MÉXICO, D.F.
## 2  Polvoroncitos Panera 40p 16 25g TR 45143 2008 AG. LAGO FILT MÉXICO, D.F.
## 3            Chocodonitas 6p 102g BIM 32819 2008 AG. LAGO FILT MÉXICO, D.F.
## 4            Mantecadas Nuez 123g BIM 41938 2008 AG. LAGO FILT MÉXICO, D.F.
## 5                Panque Pasas 255g BIM 1230 2008 AG. LAGO FILT MÉXICO, D.F.
## 6                Pan Integral 675g BIM 1146 2008 AG. LAGO FILT MÉXICO, D.F.
```

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

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

```
any(is.na(test_data))
```

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

```
# Converting categorical variables to factor
train_data[, 1:6] <- data.frame(lapply(train_data[,1:6],
                                       function (x) {return(x <- as.factor(x))}))

test_data[, 1:6] <- data.frame(lapply(test_data[,1:6],
                                      function (x) {return(x <- as.factor(x))}))
```

Exploratory Analysis

Here I did some exploration, statistical and graphic analysis. To facilitate exploratory analysis, I built the script "InventoryDemandTools.R", which contains variables and auxiliary functions. I colleted data statistic information and plotted histograms to see features' distribution.

```
# Loading Tools.R script
source("src/InventoryDemandTools.R")
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
```

```
##
##     combine
```

```
# Getting summary of train_data numerical features
summary(train_data[, NumColNames])
```

```
##  Venta_uni_hoy        Venta_hoy          Dev_uni_proxima      Dev_proxima
##  Min.   :   0.000   Min.   :     0.00   Min.   :   0.0000   Min.   :    0.00
##  1st Qu.:   2.000   1st Qu.:    16.76   1st Qu.:   0.0000   1st Qu.:    0.00
##  Median :   3.000   Median :    30.00   Median :   0.0000   Median :    0.00
##  Mean   :   7.304   Mean   :    68.45   Mean   :   0.1247   Mean   :    1.23
##  3rd Qu.:   7.000   3rd Qu.:    56.10   3rd Qu.:   0.0000   3rd Qu.:    0.00
##  Max.   :4800.000   Max.   :275061.60   Max.   :2400.0000   Max.   :44376.00
##  Demanda_uni_equil
##  Min.   :   0.000
##  1st Qu.:   2.000
##  Median :   3.000
##  Mean   :   7.218
##  3rd Qu.:   6.000
##  Max.   :4800.000
```

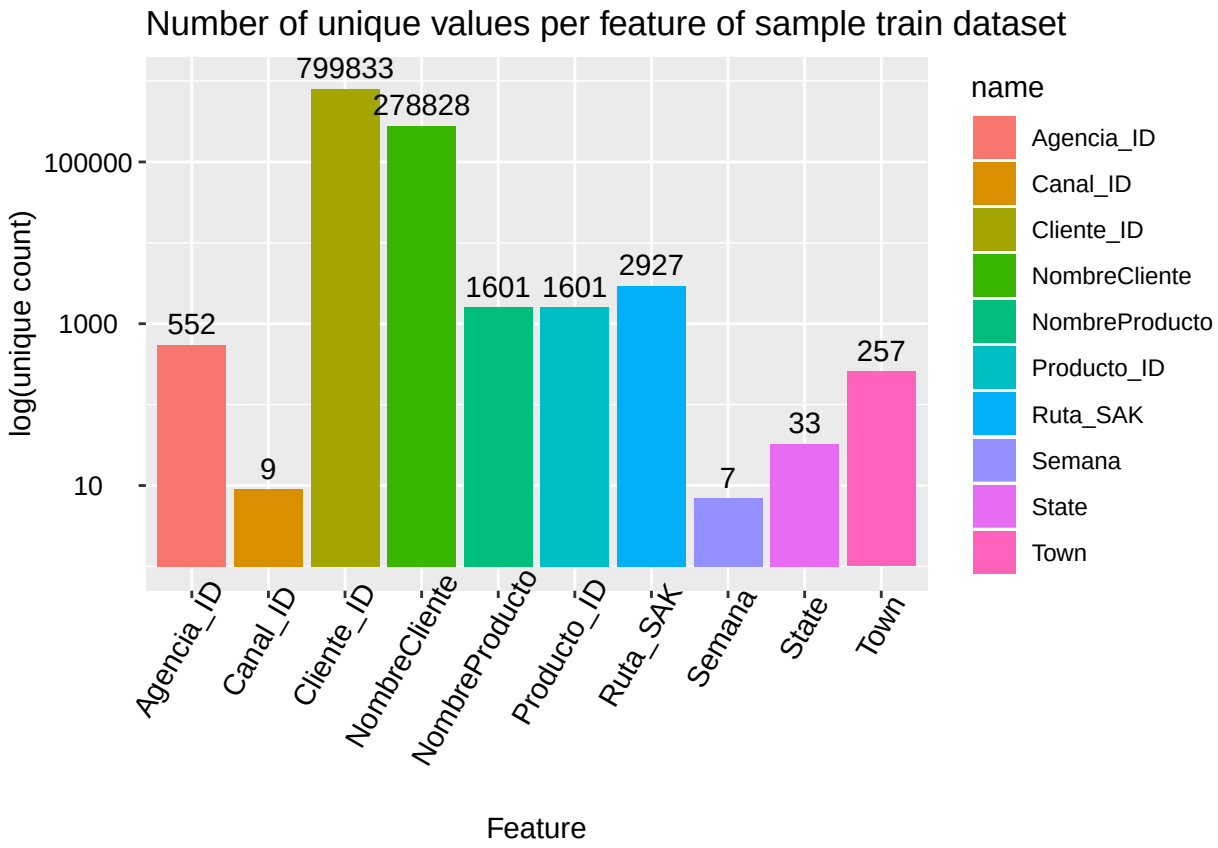
```
# Getting statistics of train_data categorial features
lapply(FacColNames, describe)
```

```
## [[1]]
##   statistic     values
## 1      name Agencia_ID
## 2     count    9800000
## 3    unique        552
## 4       top       1911
## 5      freq     106300
##
## [[2]]
##   statistic      values
## 1      name Producto_ID
## 2     count     9800000
## 3    unique        1601
## 4       top        1240
## 5      freq      283966
##
## [[3]]
##   statistic     values
## 1      name Cliente_ID
## 2     count    9800000
## 3    unique     799833
## 4       top     653378
## 5      freq      16357
##
## [[4]]
##   statistic  values
## 1      name  Semana
## 2     count 9800000
## 3    unique       7
## 4       top       3
## 5      freq 1474987
##
```

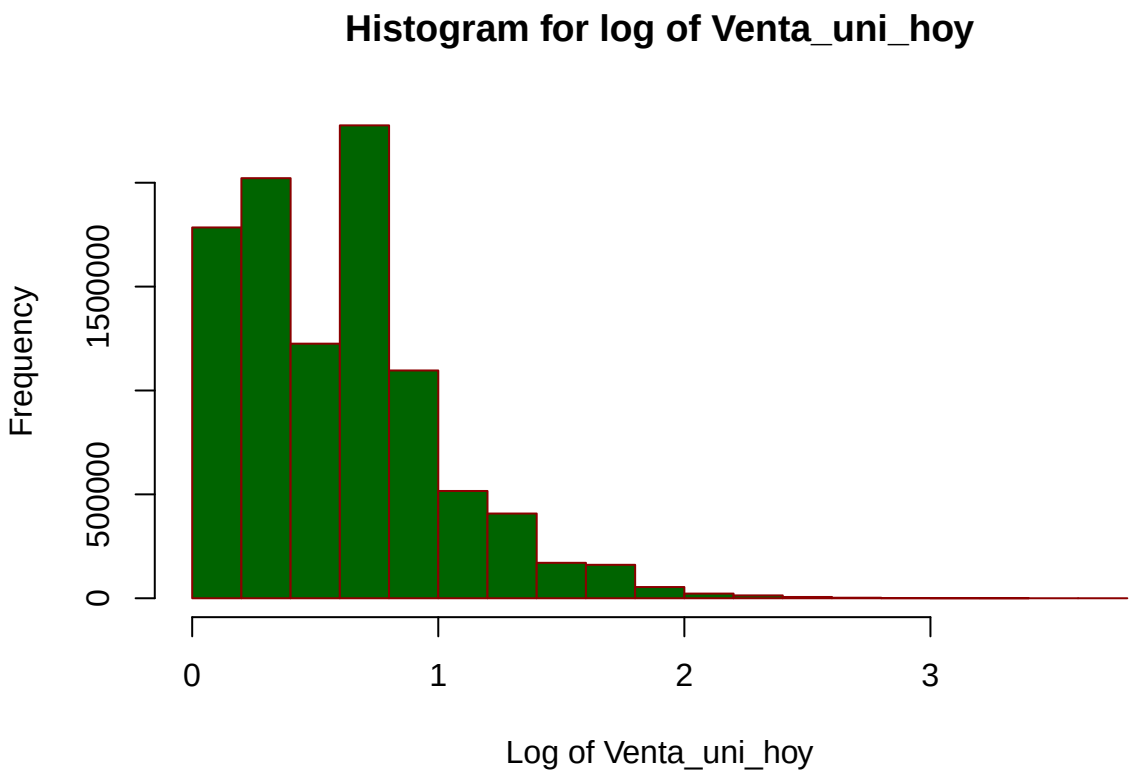
```
## [[5]]
##   statistic   values
## 1      name Canal_ID
## 2     count  9800000
## 3    unique        9
## 4       top        1
## 5      freq  8908869
##
## [[6]]
##   statistic   values
## 1      name Ruta_SAK
## 2     count  9800000
## 3    unique     2927
## 4       top     1201
## 5      freq    60999
##
## [[7]]
##   statistic          values
## 1      name   NombreCliente
## 2     count         9800000
## 3    unique          278828
## 4       top NO IDENTIFICADO
## 5      freq         1750928
##
## [[8]]
##   statistic                                values
## 1      name                         NombreProducto
## 2     count                                9800000
## 3    unique                                  1601
## 4       top Mantecadas Vainilla 4p 125g BIM 1240
## 5      freq                                 283966
##
## [[9]]
##   statistic               values
## 1      name                 Town
## 2     count              9800000
## 3    unique                  257
## 4       top 2017 AG. SANTA CLARA
## 5      freq               121405
##
## [[10]]
##   statistic           values
## 1      name            State
## 2     count          9800000
## 3    unique               33
## 4       top ESTADO DE MÉXICO
## 5      freq          1436825
```

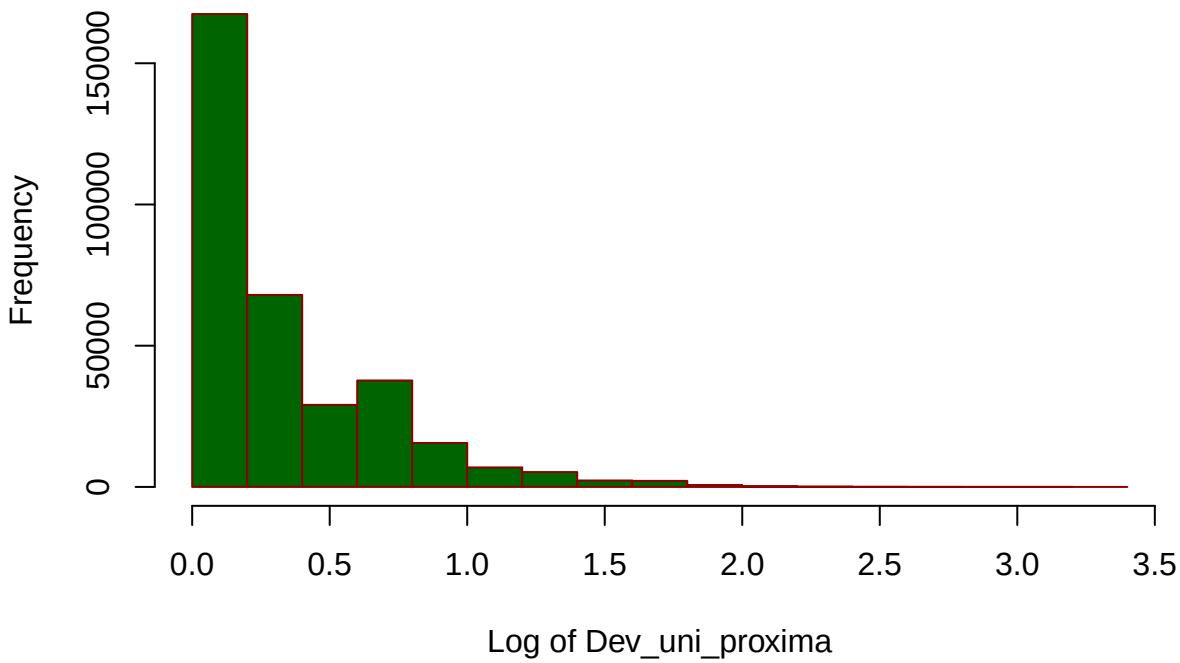
```r
# Visual analysis by plotting
# Loading libraries
library(dplyr)
library(ggplot2)

# Get the number of unique values for categoriacal dependent variables
```

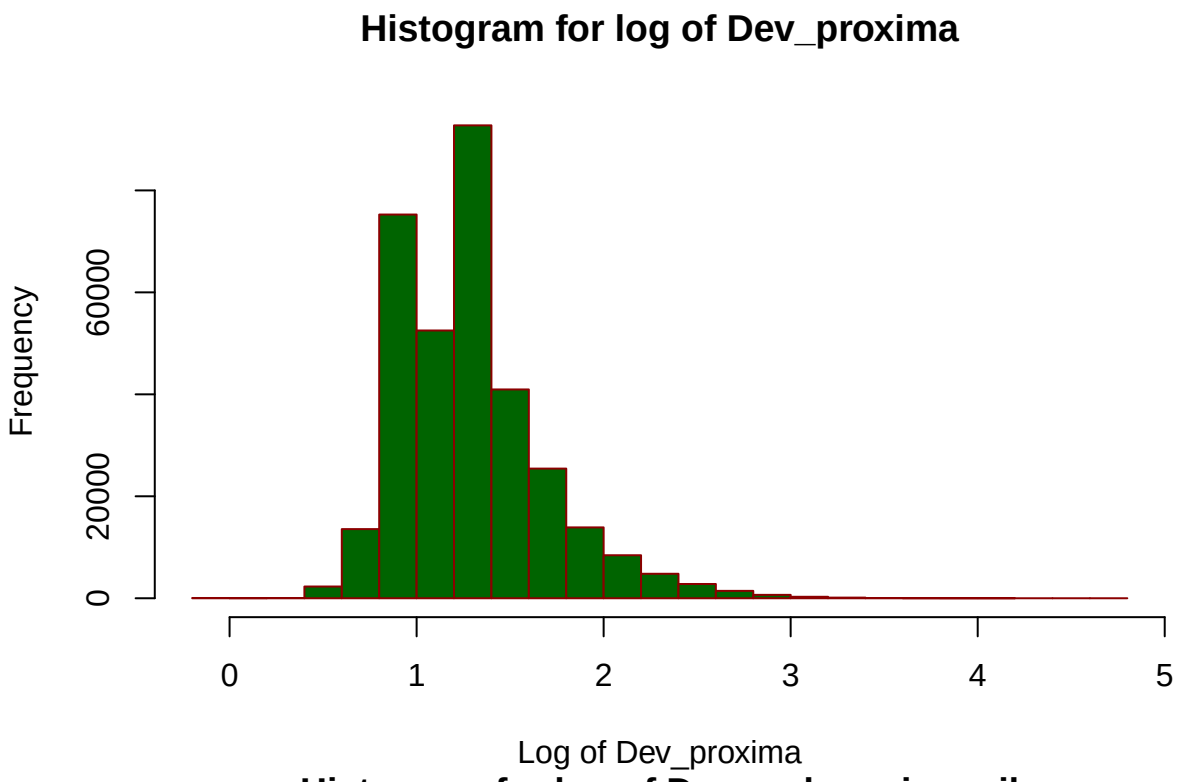
```
uniq <- sapply(train_data[, FacColNames], function(x) {return(length(unique(x)))})
uniq_data <- data.frame(value=uniq, name=names(uniq), row.names = NULL)
uniq <- NULL

# Disabling scientific notation in R
options(scipen = 999)

# Bar plot of the number of single values
uniq_plot <- ggplot(uniq_data, aes(x=name, y=value, fill=name)) +
  geom_bar(stat = "identity") +
  ggtitle("Number of unique values per feature of sample train dataset") +
  xlab("Feature") +
  ylab("log(unique count)") +
  scale_y_log10(limits = c(1,1e6)) +
  geom_text(aes(label = sprintf("%d", value), y=value),  vjust = -0.5) +
  theme(axis.text.x = element_text(angle = 60, hjust = 0.8, size=11,color="black")) +
  theme(axis.text.y = element_text(hjust = 0.5, size=10,color="black"))

uniq_plot
```

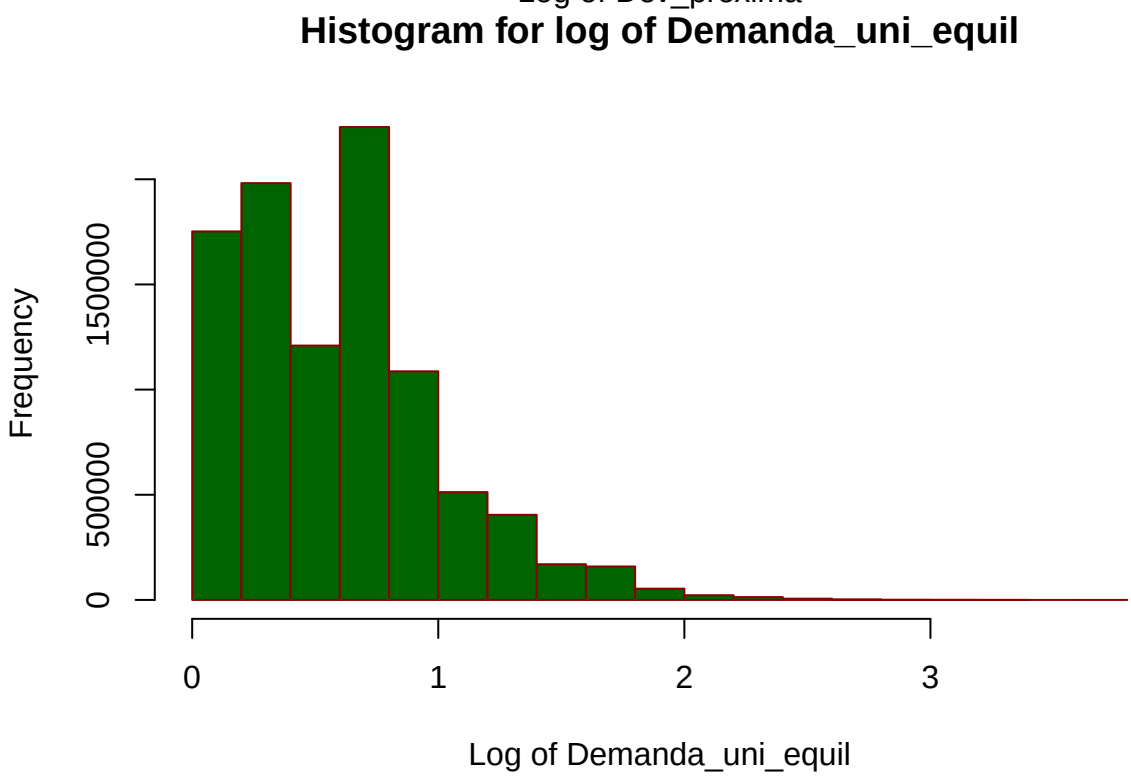



```
uniq_plot <- NULL

# Plotting htograms of train_data numeric features
for (name in NumColNames) { hist_graph(name)}
```

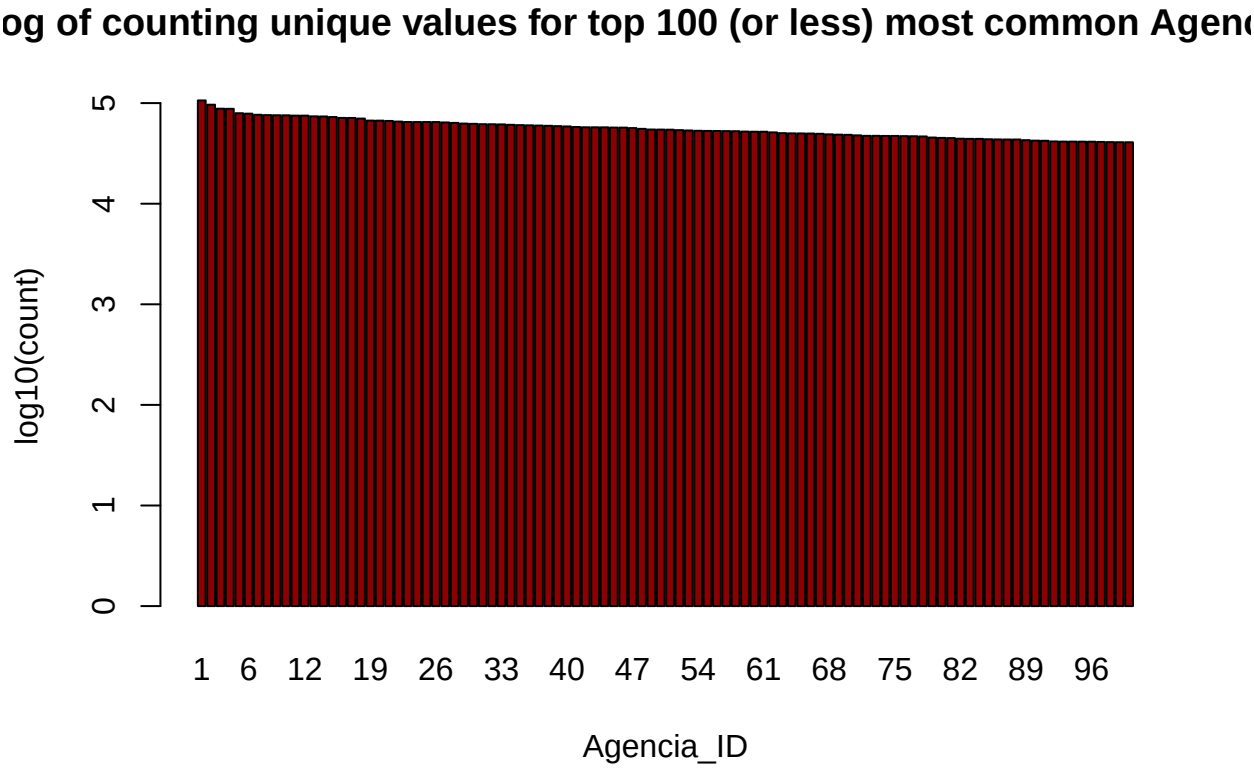

Histogram for log of Venta_uni_hoy
Frequency
0
500000
1500000
0
1
2
3
Log of Venta_uni_hoy

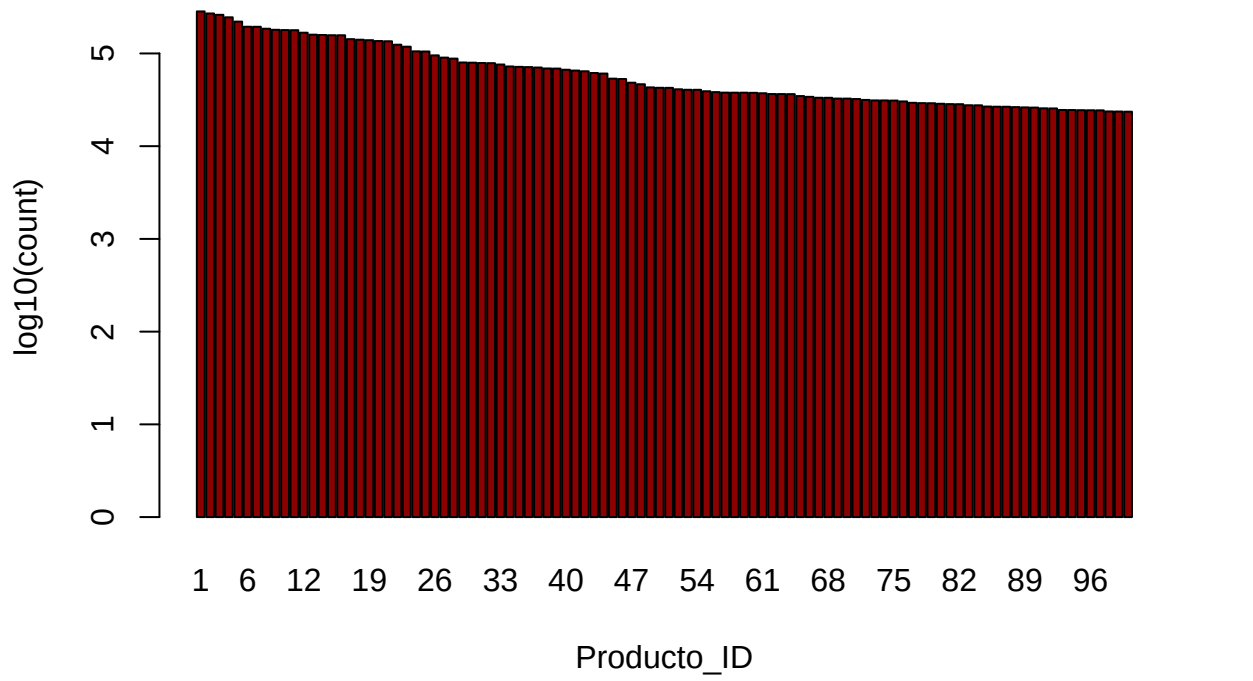

Histogram for log of Dev_uni_proxima
Frequency
0
50000
100000
150000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Log of Dev_uni_proxima

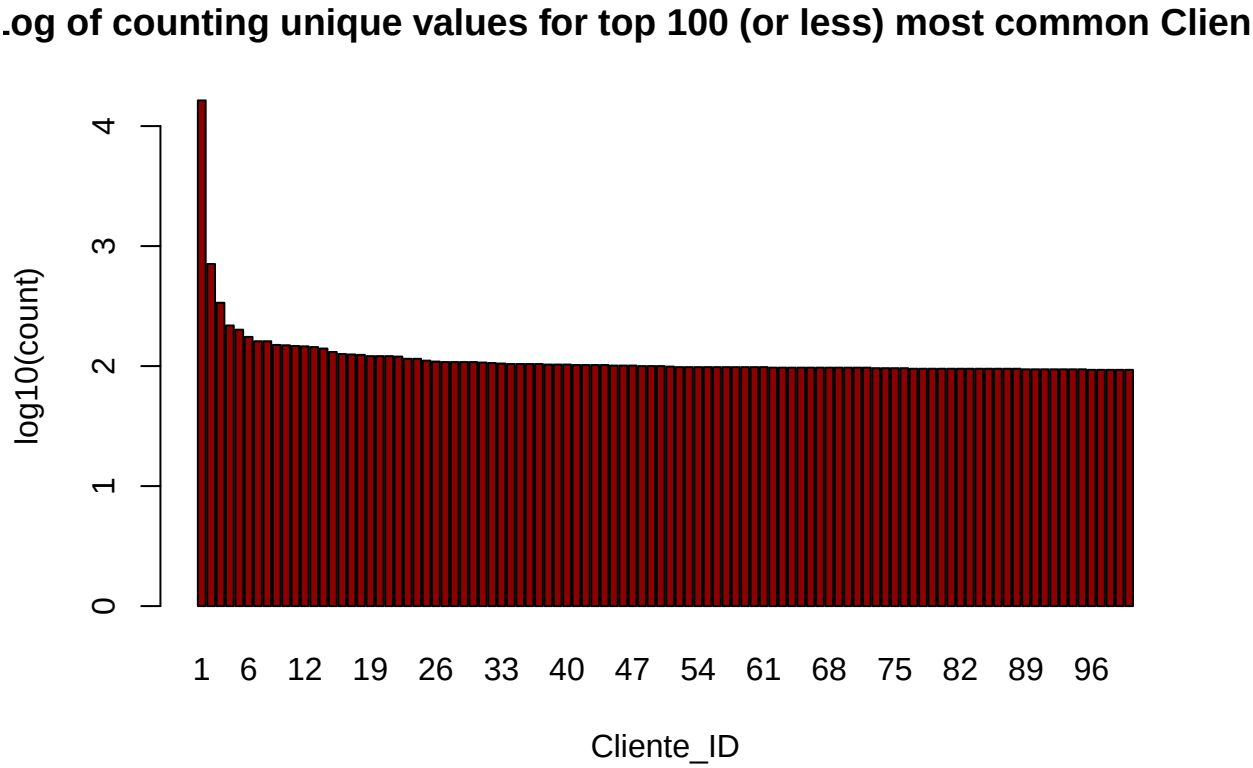


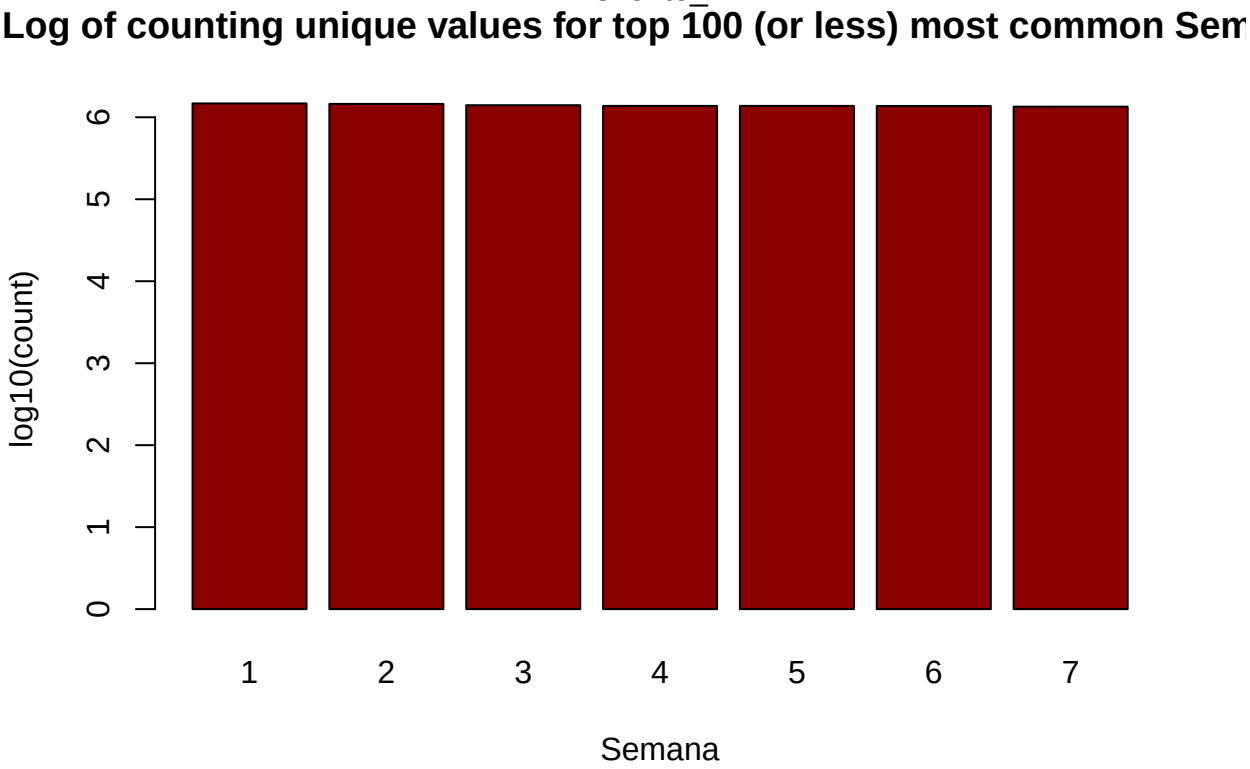


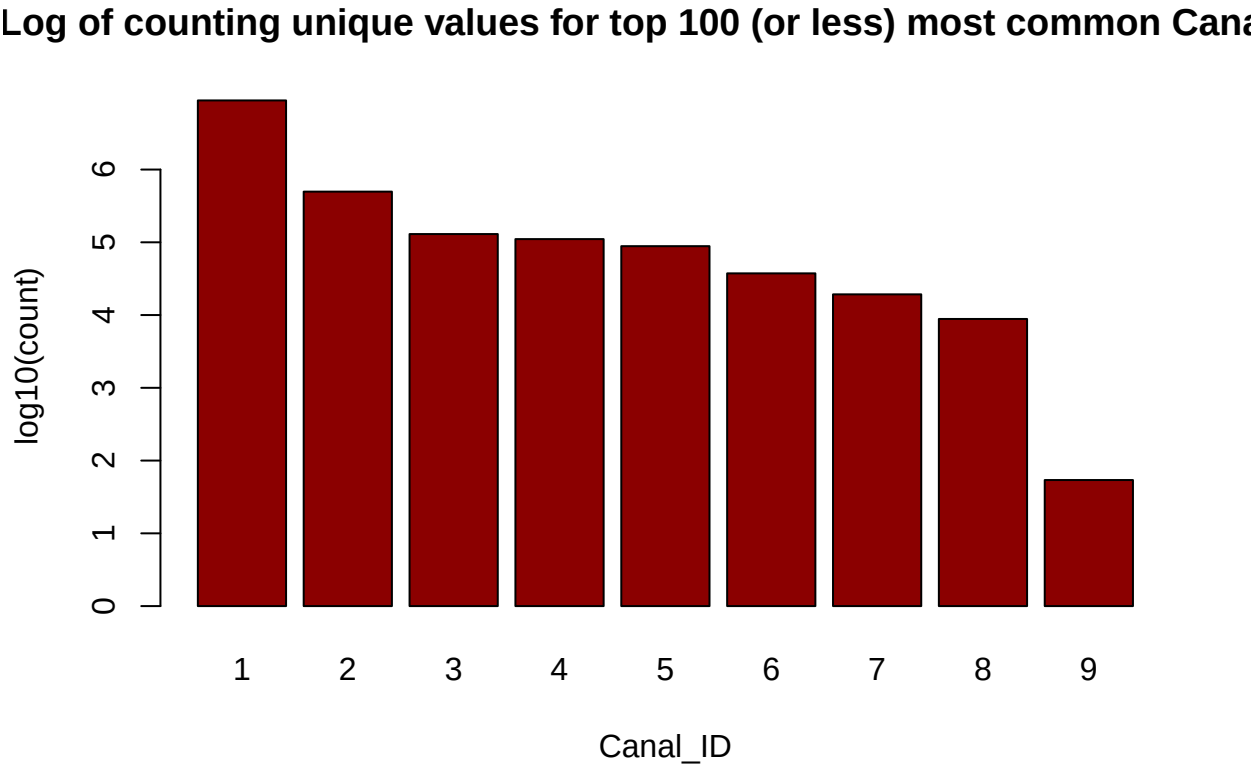

```
# Barplot of train_data categorical features
for (name in FacColNames) { freq_graph(name)}
```

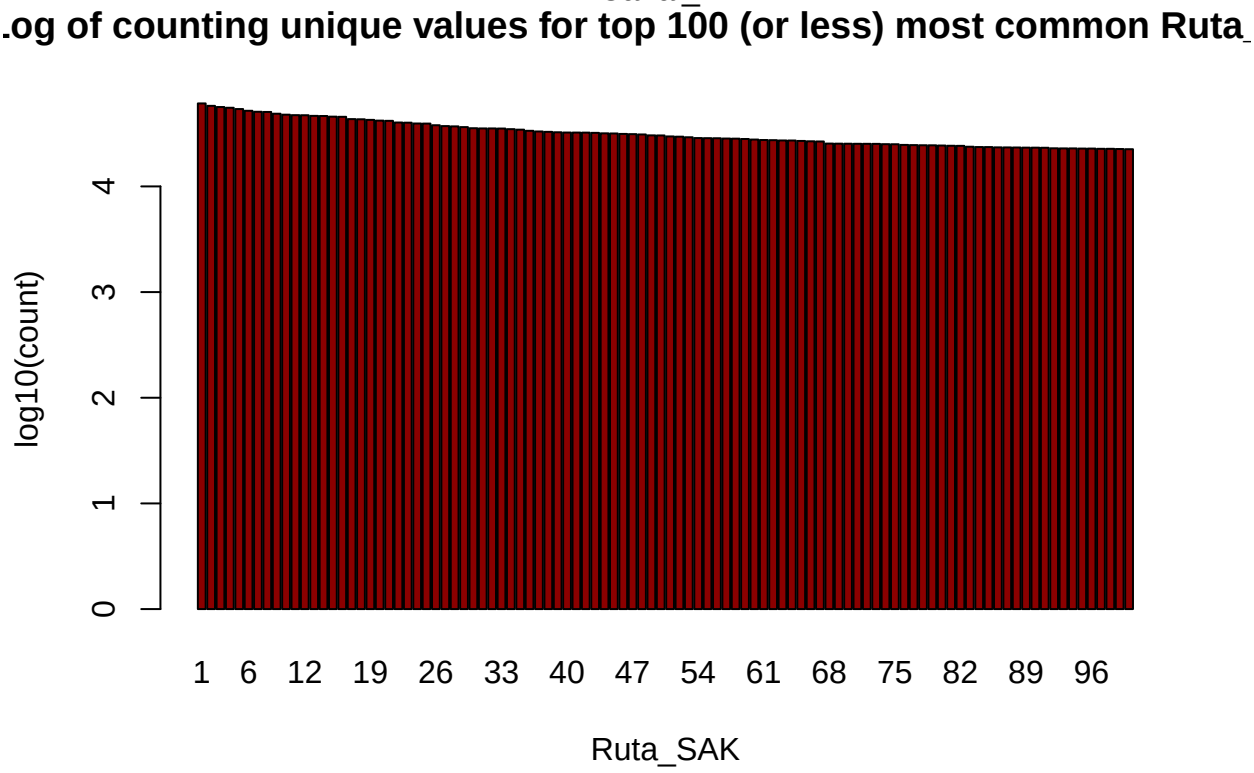

og of counting unique values for top 100 (or less) most common Agenc
log10(count)
0
1
2
3
4
5
1
6
12
19
26
33
40
47
54
61
68
75
82
89
96
Agencia_ID

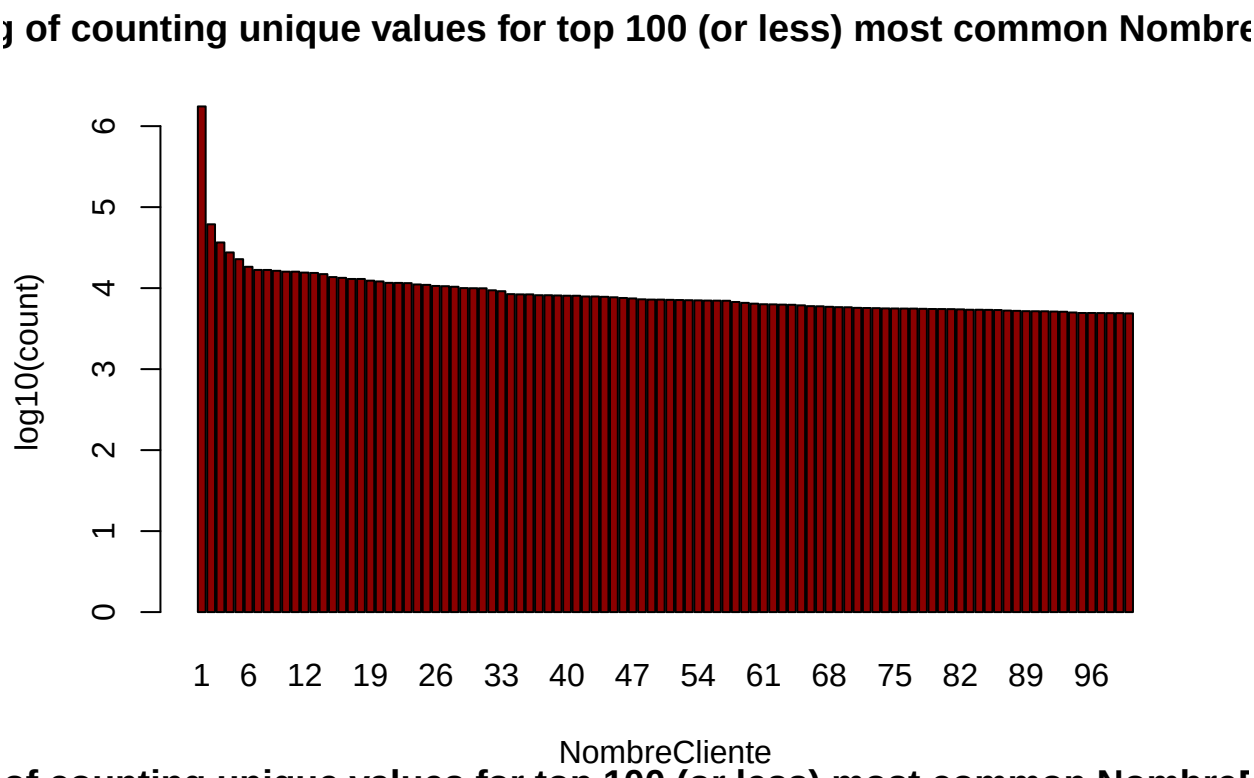

og of counting unique values for top 100 (or less) most common Produ
log10(count)
0
1
2
3
4
5
1
6
12
19
26
33
40
47
54
61
68
75
82
89
96
Producto_ID

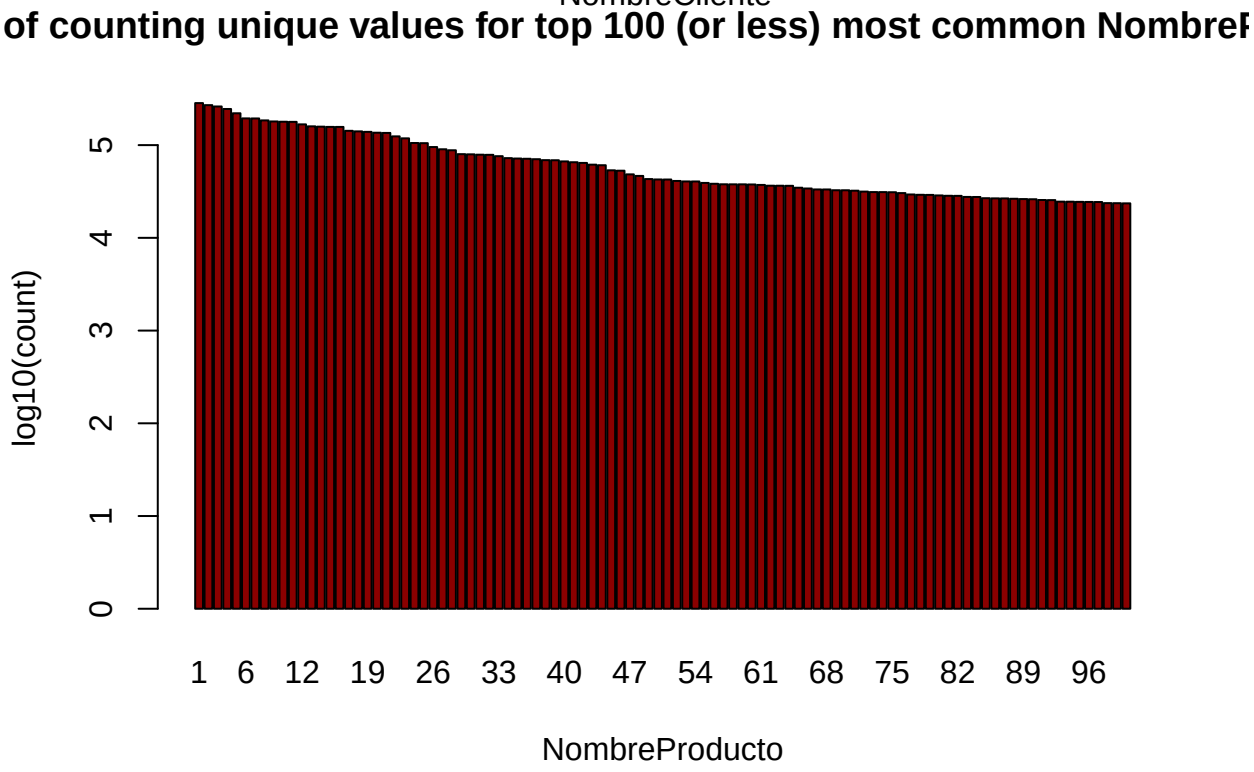

.og of counting unique values for top 100 (or less) most common Clien
log10(count)
0
1
2
3
4
1 6 12 19 26 33 40 47 54 61 68 75 82 89 96
Cliente_ID

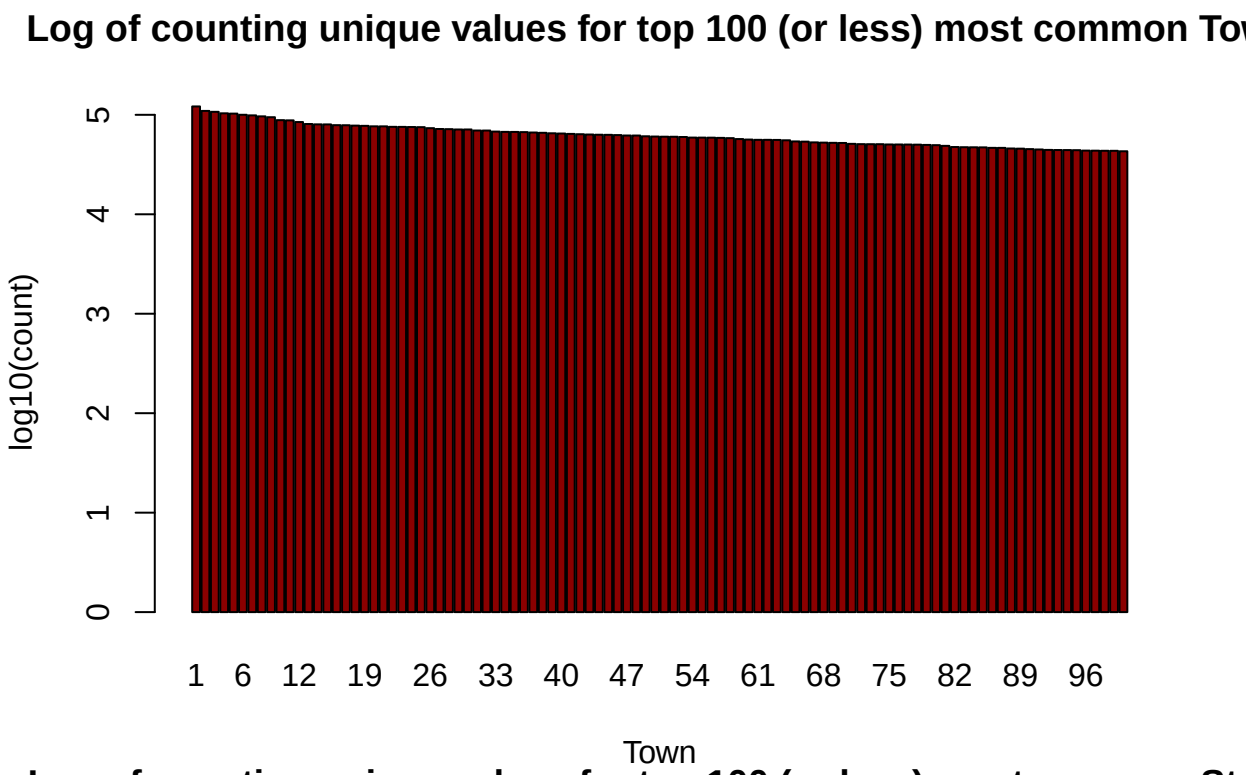

Log of counting unique values for top 100 (or less) most common Sem
log10(count)
0
1
2
3
4
5
6
1 2 3 4 5 6 7
Semana

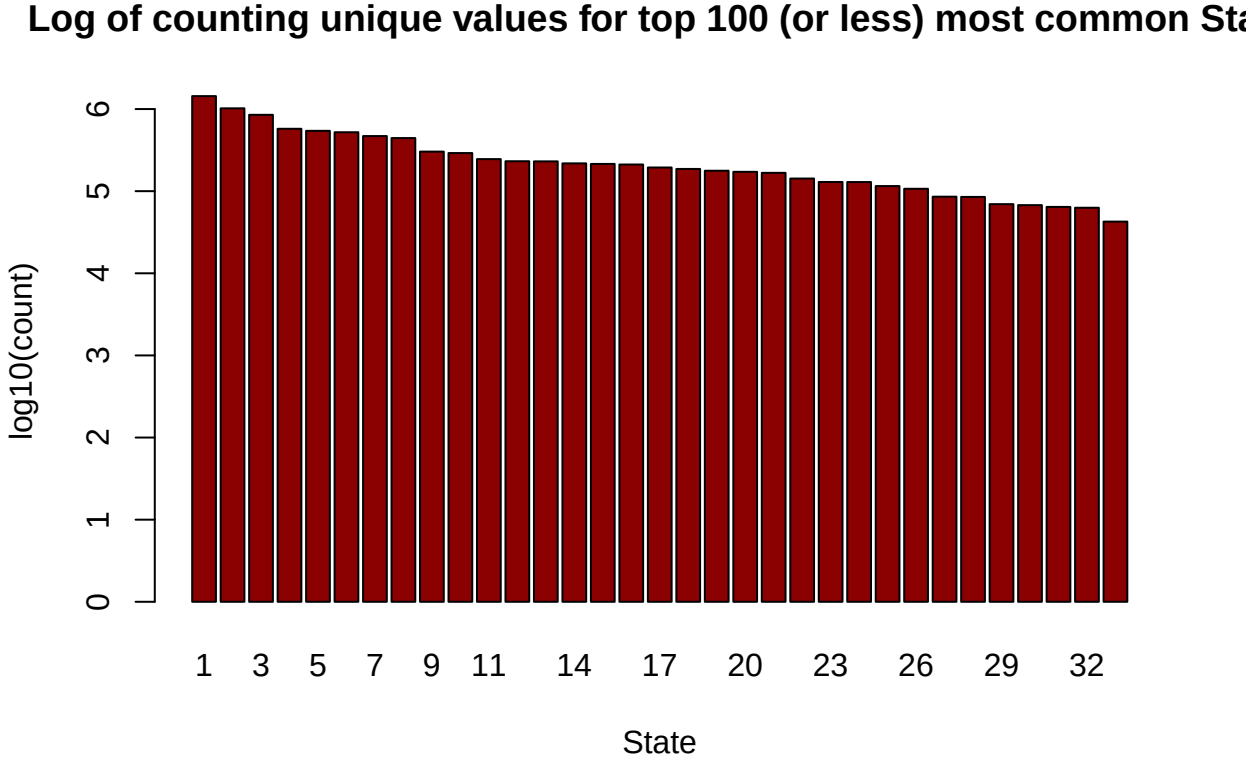

Log of counting unique values for top 100 (or less) most common Cana
log10(count)
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
Canal_ID


.og of counting unique values for top 100 (or less) most common Ruta_
log10(count)
0
1
2
3
4
1
6
12
19
26
33
40
47
54
61
68
75
82
89
96
Ruta_SAK


g of counting unique values for top 100 (or less) most common Nombre
log10(count)
0
1
2
3
4
5
6
1
6
12
19
26
33
40
47
54
61
68
75
82
89
96
NombreCliente


of counting unique values for top 100 (or less) most common NombreP
log10(count)
0
1
2
3
4
5
1
6
12
19
26
33
40
47
54
61
68
75
82
89
96
NombreProducto





Checking for relation between features

Here I check for correlation between features. To avoid processing problems I did correlation analysis with a less sample of the data.

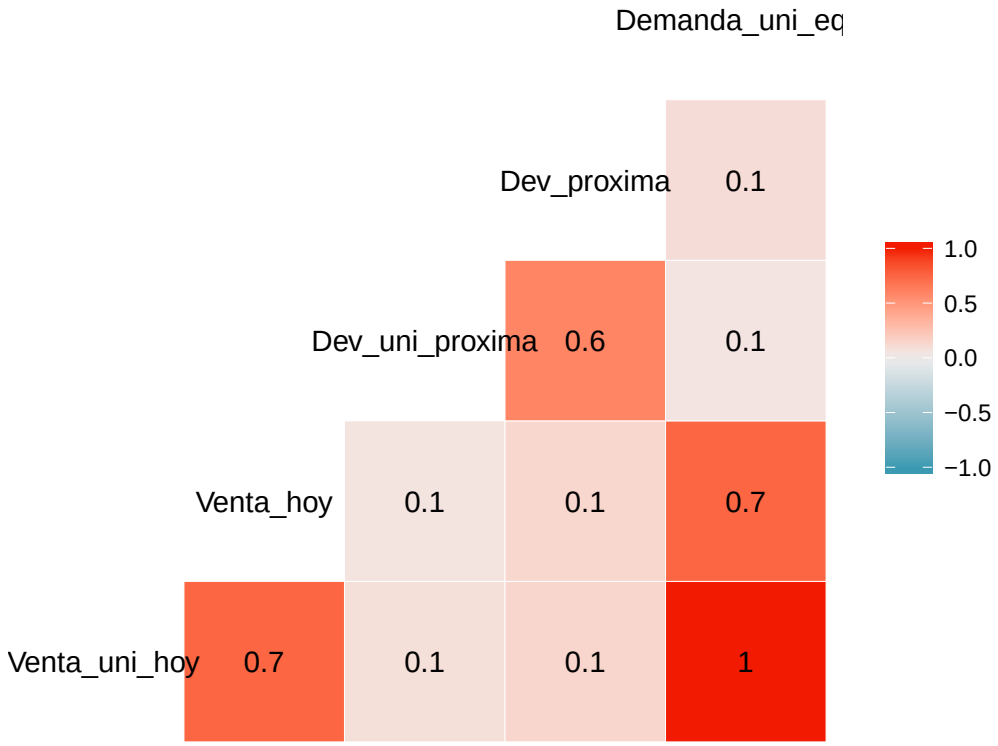
```
# Getting random sample
set.seed(10000000)
```

```
index <- sample(1:nrow(train_data), size=100000)
sample_train_data <- train_data[index,]

# getting correlation values
cor(sample_train_data[, NumColNames])
```

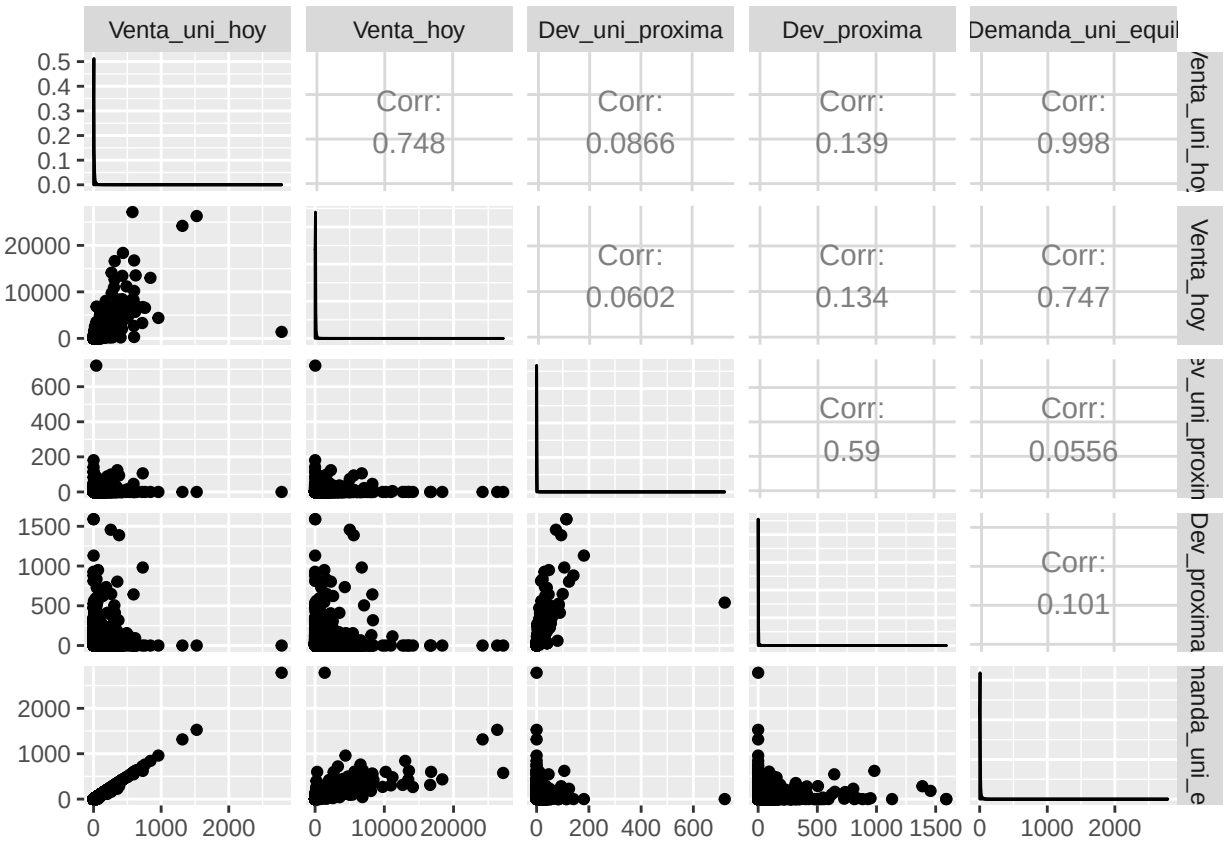
```
##                   Venta_uni_hoy Venta_hoy Dev_uni_proxima Dev_proxima
## Venta_uni_hoy         1.0000000 0.7478626      0.08658960   0.1391385
## Venta_hoy             0.7478626 1.0000000      0.06018880   0.1336498
## Dev_uni_proxima       0.0865896 0.0601888      1.00000000   0.5901745
## Dev_proxima           0.1391385 0.1336498      0.59017446   1.0000000
## Demanda_uni_equil     0.9983284 0.7468079      0.05560754   0.1012736
##                   Demanda_uni_equil
## Venta_uni_hoy            0.99832842
## Venta_hoy                0.74680788
## Dev_uni_proxima          0.05560754
## Dev_proxima              0.10127361
## Demanda_uni_equil        1.00000000
```

```
# Install package
# install.packages("GGally")

# Different correlation plots
GGally::ggcorr(sample_train_data[, NumColNames], palette = "RdBu", label = TRUE)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```




```
GGally::ggpairs(sample_train_data[, NumColNames])
```



As noted above, there are strong positive correlations between the following variables: demand and number of sales, demand and value of sales, number of sales and value of sales. There is a parcial positive correlation between the amount of returns and its value. Finally, the variables demand and returns' amount, number of sales and returns' amount, values of sales and returns' amount have a weak positive correlation. No negative correlation was observed.

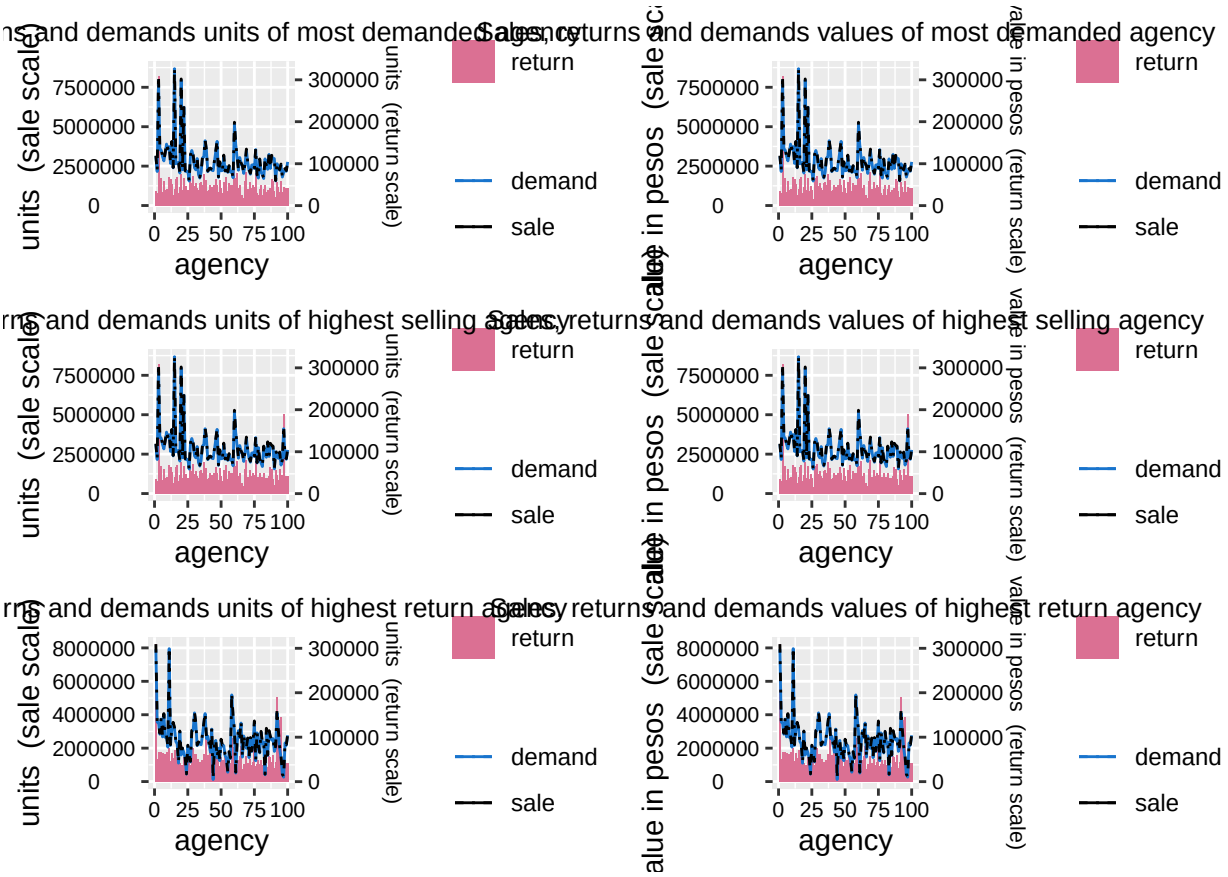
Checking for demand, sale and return patterns.

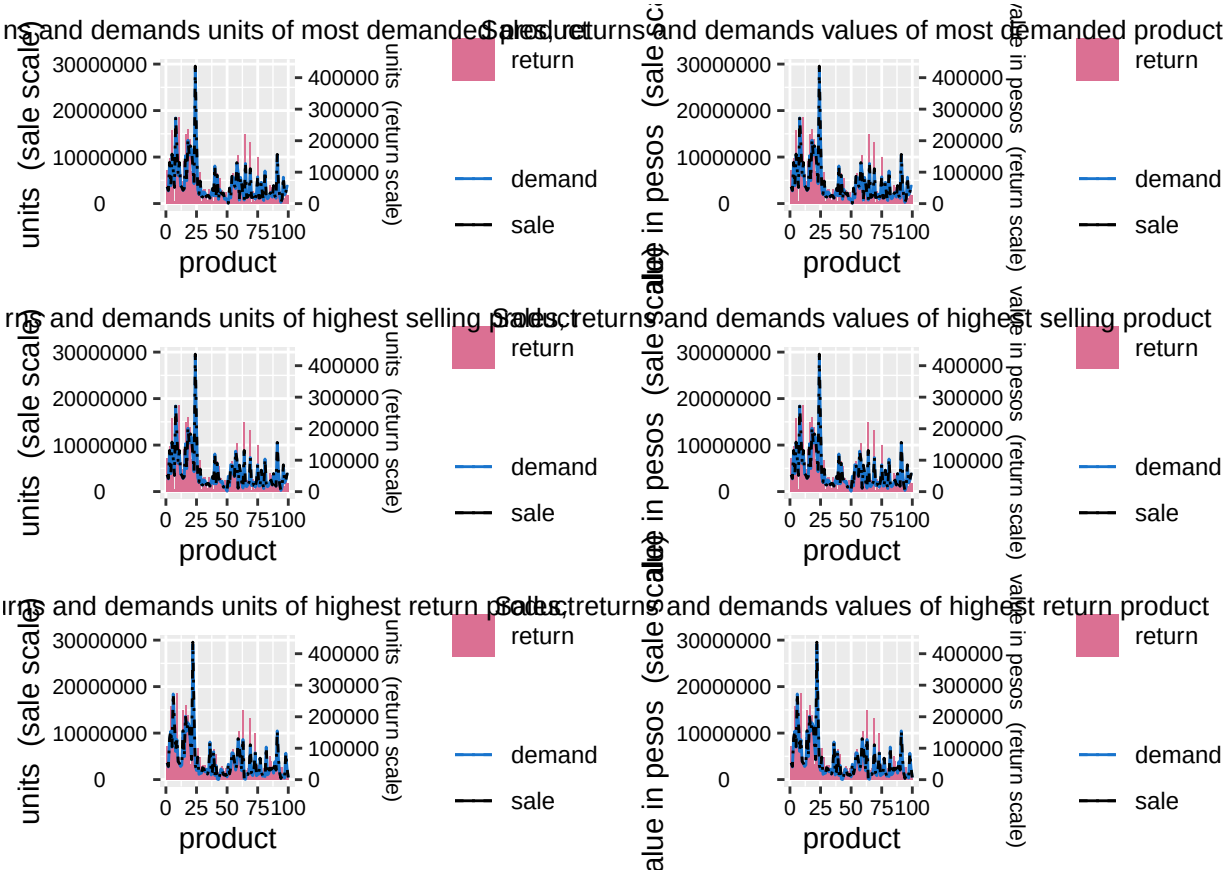
Then, I grouped variables to look for patterns. For better visualization of the graphs, I made the graphs with two y-axes to match the scales of returns and sales.

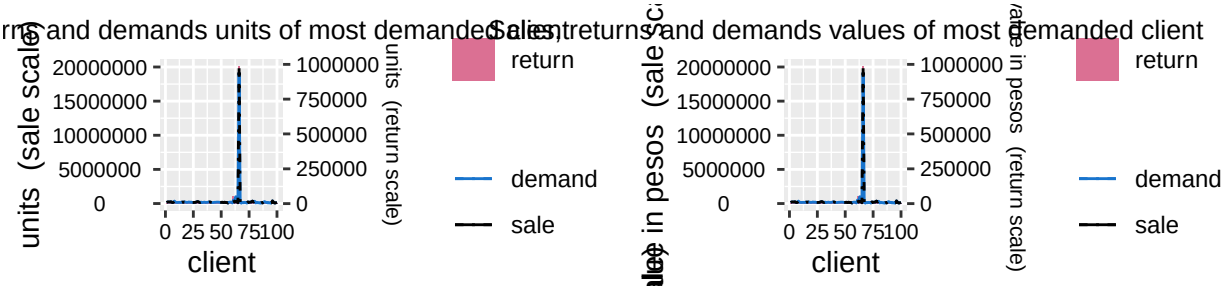
I grouped the training data by each of the following variables: "Agencia_ID", "Producto_ID", "Cliente_ID", "Semana", "Canal_ID", "Ruta_SAK", "Town", "State". Then, I collected and visualized demand, sales and returns (units and price in pesos). For variables in many categories, I selected the first 100 records ordered in three different ways: number of demand, sales and returns. For these variables, 6 graphs were constructed: one graph with values in units and one for values in pesos for each of the three types of sorting. For variables of a few categories, 2 graphs (unit and pesos) were constructed for the grouped variable.

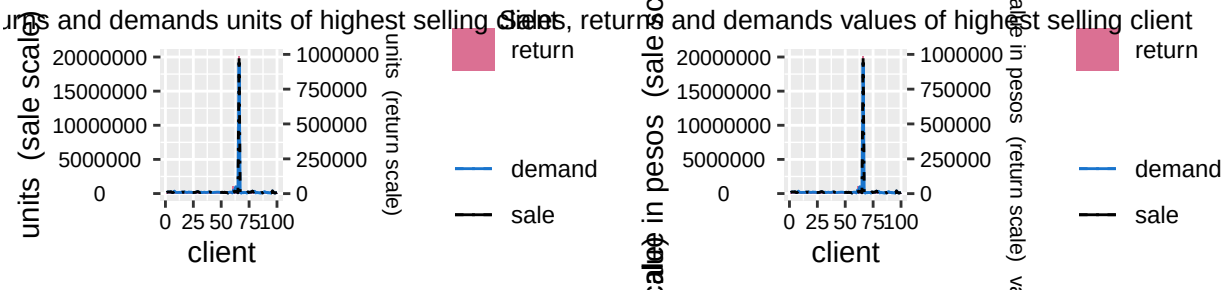
The x-axis does not represent the variable ids, just a sequence that varies from 1 to the total of plotted records.

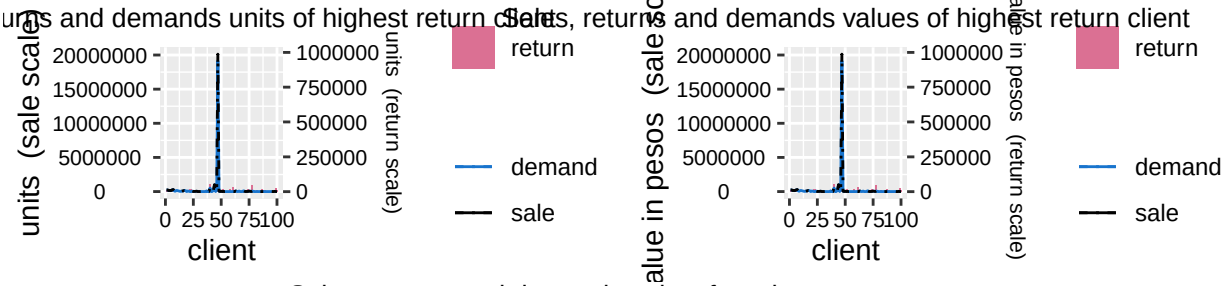
```
source("src/InventoryDemandTools.R")
# grouping data by features to check for pattens of sales, returns and demand.
# sales scale on the left and return scale on the right.
for (name in names(translatedNames)) { demand_grouped_by_graph(name)  }
```

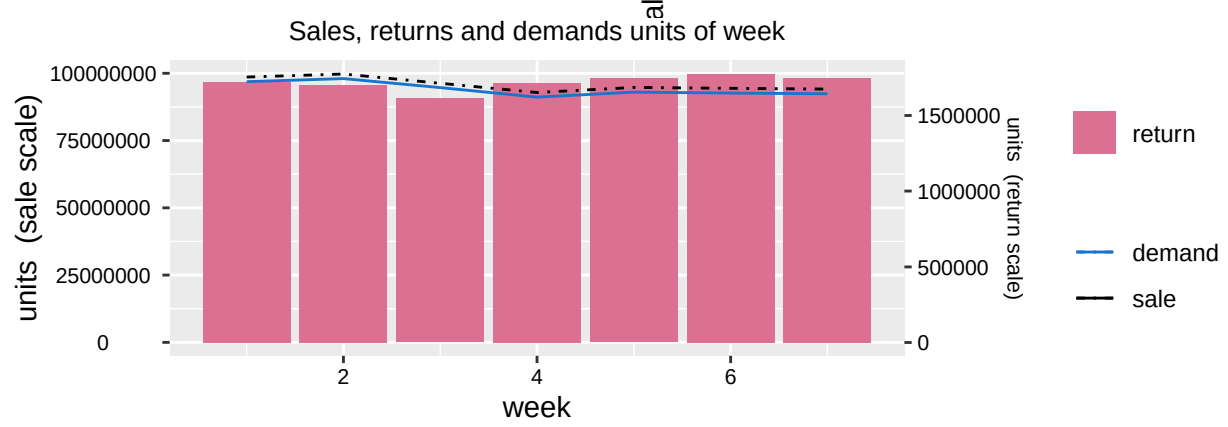

ns and demands units of most demanded agency
Sales, returns and demands values of most demanded agency
rn and demands units of highest selling agency
Sales, returns and demands values of highest selling agency
rn and demands units of highest return agency
Sales, returns and demands values of highest return agency
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
agency
return
demand
sale
0
25
50
75
100
2500000
5000000
7500000
2000000
4000000
6000000
8000000
100000
200000
300000

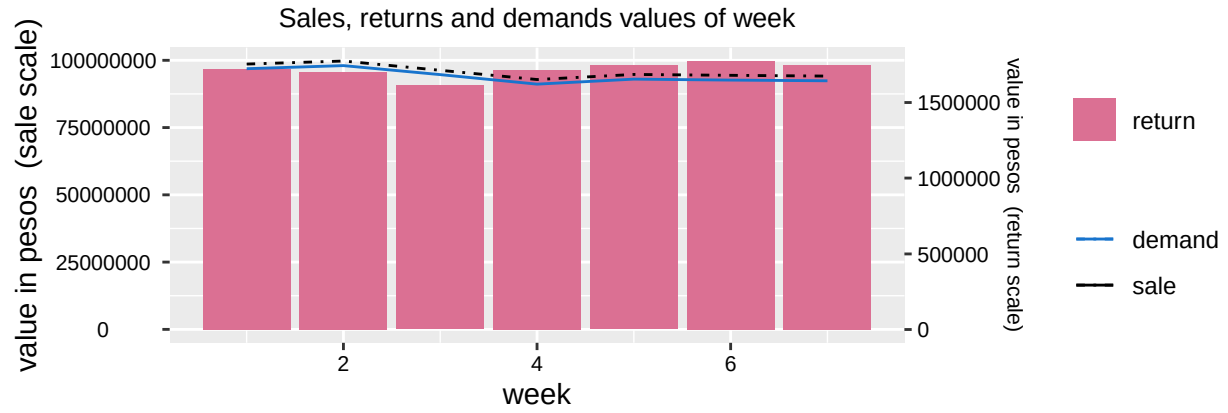

Sales, returns and demands units of most demanded product
Sales, returns and demands values of most demanded product
Sales, returns and demands units of highest selling product
Sales, returns and demands values of highest selling product
Sales, returns and demands units of highest return product
Sales, returns and demands values of highest return product
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
product
return
demand
sale
0
10000000
20000000
30000000
100000
200000
300000
400000
0 25 50 75100


rns and demands units of most demanded client
Sales, returns and demands values of most demanded client
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
client
return
demand
sale

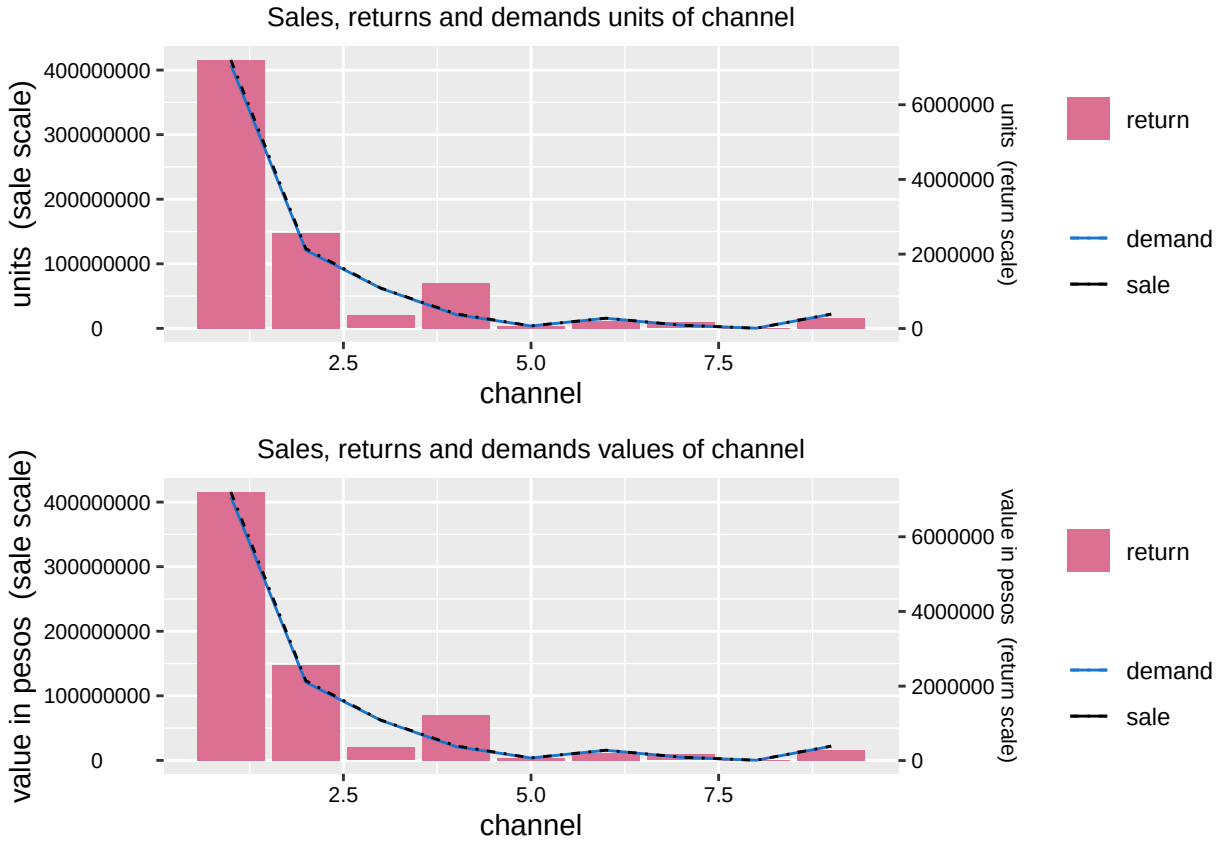

urns and demands units of highest selling client
Sales, returns and demands values of highest selling client
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
client
return
demand
sale


urns and demands units of highest return client
Sales, returns and demands values of highest return client
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
client
return
demand
sale

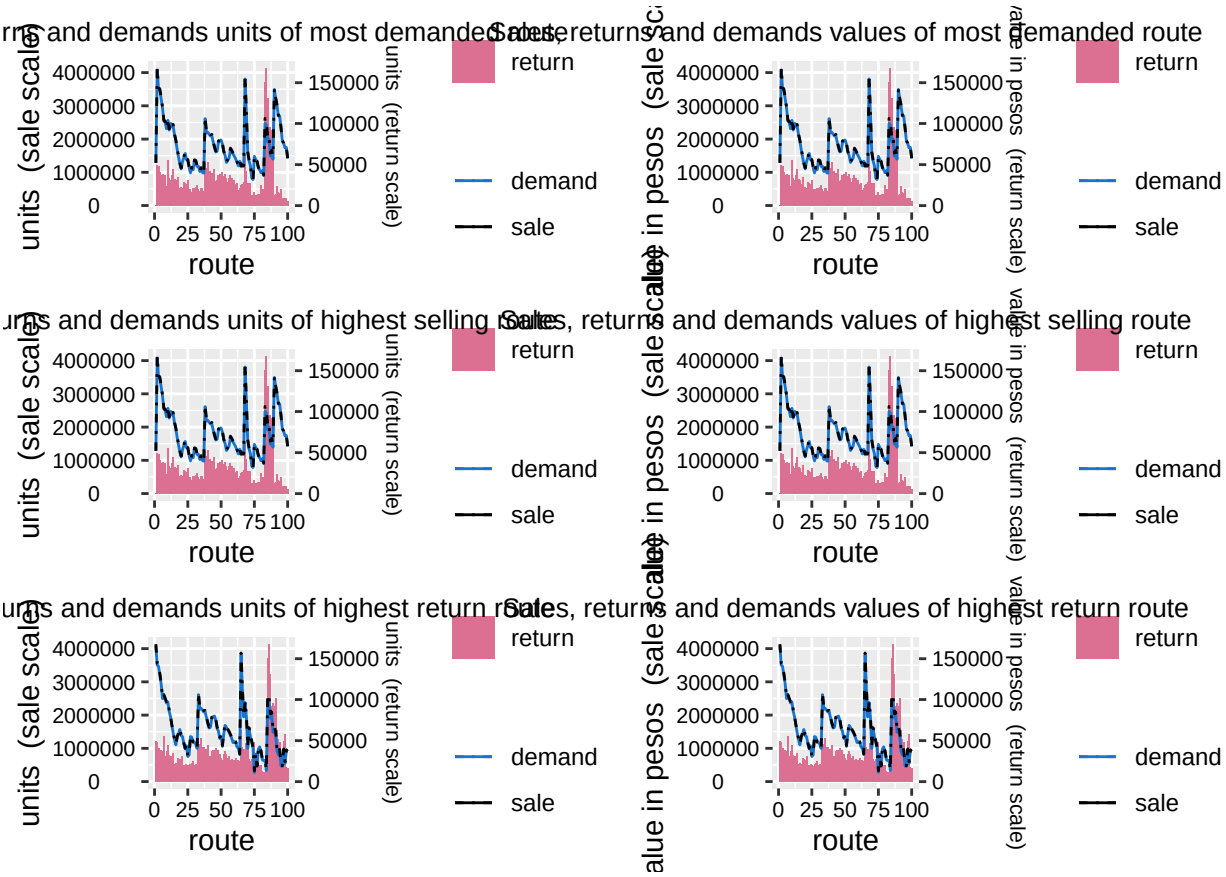

Sales, returns and demands units of week
units (sale scale)
units (return scale)
week
return
demand
sale

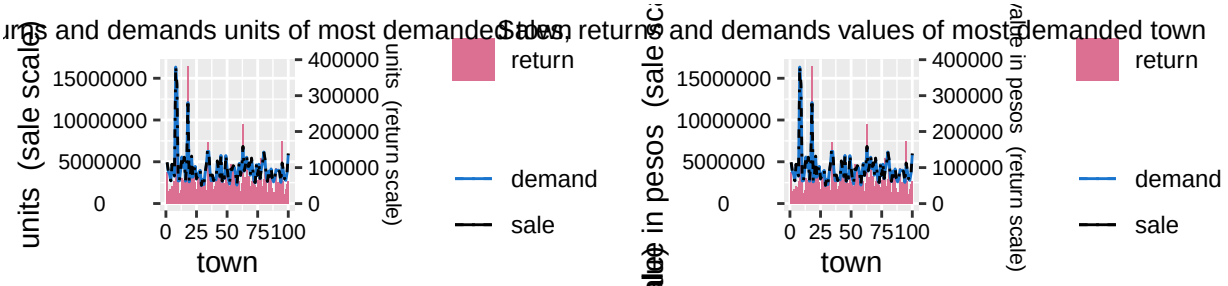

Sales, returns and demands values of week
value in pesos (sale scale)
value in pesos (return scale)
week
return
demand
sale

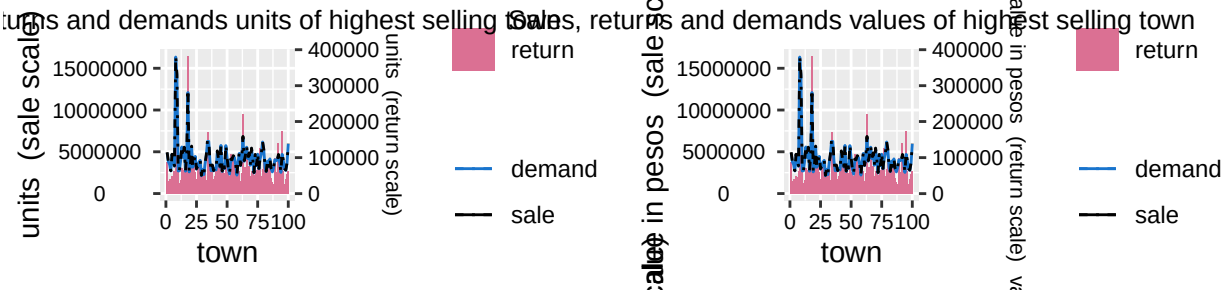
```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

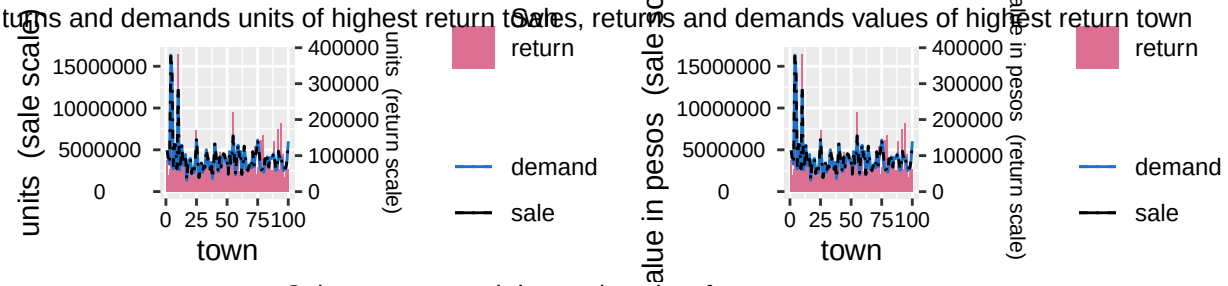


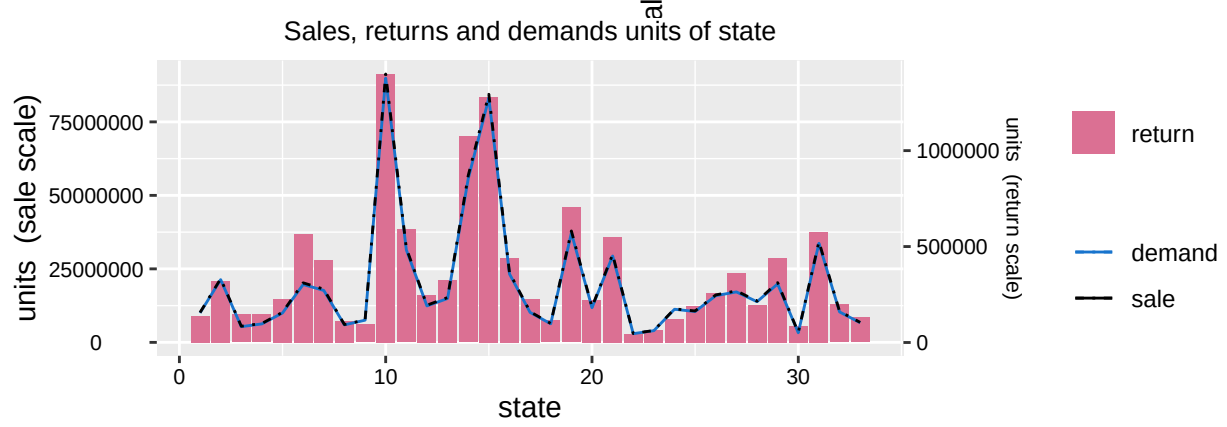

```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

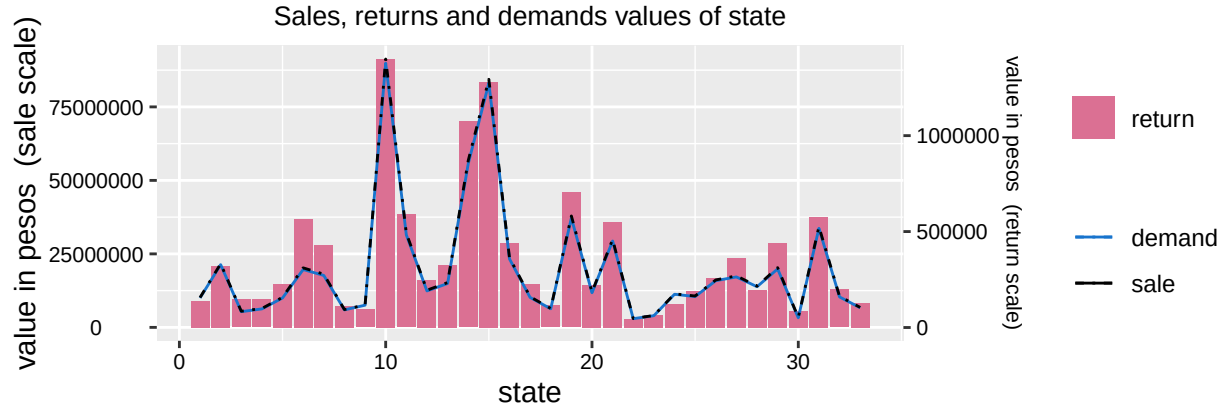

rns and demands units of most demanded route
Sales, returns and demands values of most demanded route
urns and demands units of highest selling route
Sales, returns and demands values of highest selling route
urns and demands units of highest return route
Sales, returns and demands values of highest return route
units (sale scale)
units (return scale)
value in pesos (sale scale)
value in pesos (return scale)
route
return
demand
sale
0
1000000
2000000
3000000
4000000
50000
100000
150000
0
25
50
75
100


urns and demands units of most demanded town
Sales, returns and demands values of most demanded town
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
15000000
10000000
5000000
0
400000
300000
200000
100000
0
0 25 50 75 100
town
return
demand
sale


turns and demands units of highest selling town
Sales, returns and demands values of highest selling town
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
15000000
10000000
5000000
0
400000
300000
200000
100000
0
0 25 50 75 100
town
return
demand
sale


turns and demands units of highest return town
Sales, returns and demands values of highest return town
units (sale scale)
value in pesos (sale scale)
units (return scale)
value in pesos (return scale)
15000000
10000000
5000000
0
400000
300000
200000
100000
0
0 25 50 75 100
town
return
demand
sale

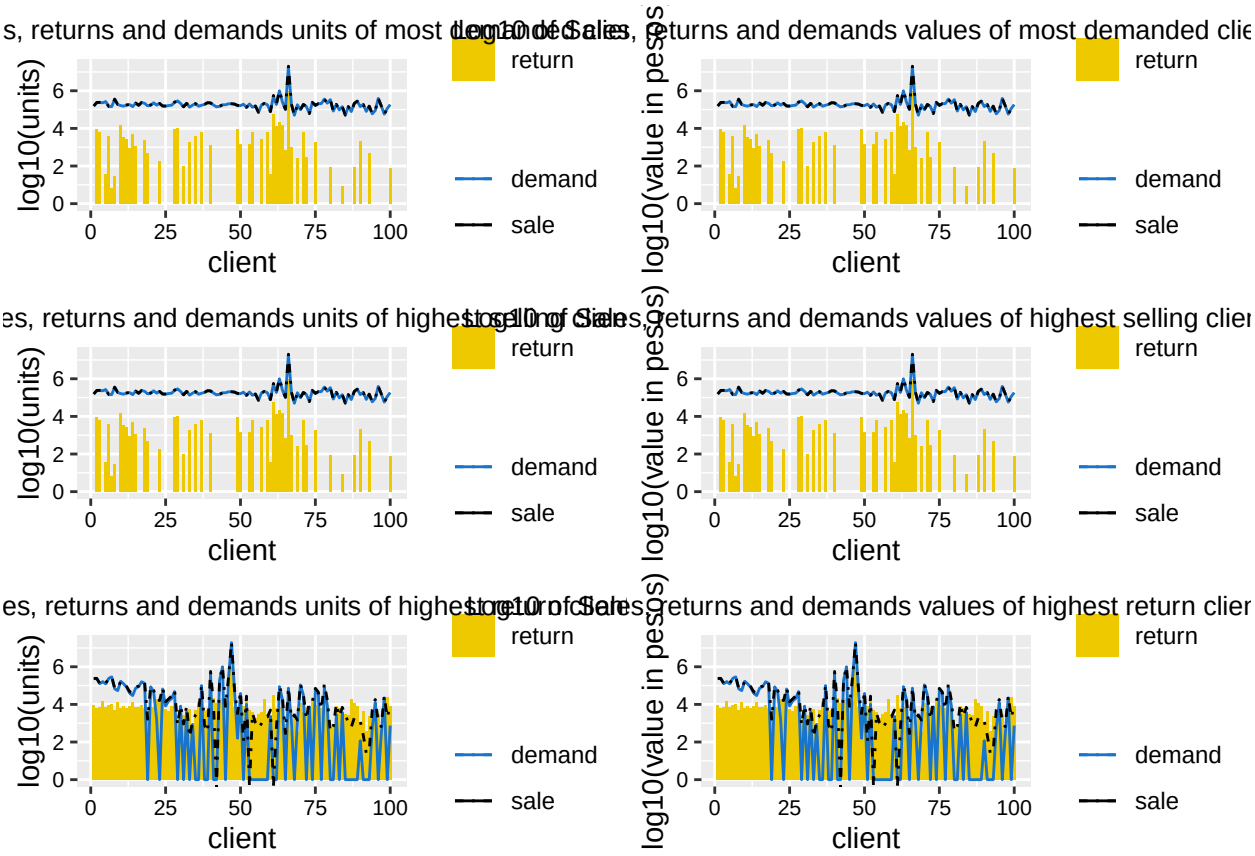

Sales, returns and demands units of state
units (sale scale)
units (return scale)
75000000
50000000
25000000
0
1000000
500000
0
0
10
20
30
state
return
demand
sale


Sales, returns and demands values of state
value in pesos (sale scale)
value in pesos (return scale)
75000000
50000000
25000000
0
1000000
500000
0
0
10
20
30
state
return
demand
sale

```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

```
# grouping data by client to check for pattens of sales, returns and demand.
# Values changed to log10 scale to better visualization.
log_demand_grouped_by_graph("Cliente_ID")
```

```
## Warning: Removed 54 rows containing missing values (geom_bar).

## Warning: Removed 54 rows containing missing values (geom_bar).

## Warning: Removed 54 rows containing missing values (geom_bar).

## Warning: Removed 54 rows containing missing values (geom_bar).
```



All graphs present practically equal curves for demands and sales, which confirms the high correlation value between both variables. In addition, the quantity and value graphs are very similar, which also points to a high correlation between the quantity and the value in pesos of the analyzed variables.

A brief description of what was observed in the graphs grouped by the variables:

- Agency: The values of return for best sellers do not differ much from the rate of highest returns. The demand and sale rates however differ, being less in the higher return ids.
- Channel: Channels with high demand have high returns, which points to a certain correlation between both. Approximately half of the channels have a higher return than demand/sale.
- Client: The values of return for best sellers do not differ much from the rate of highest returns. The

demand and sale rates however differ, being less in the higher return ids. The ids with the highest demand / return have these variables greater than the return. The highest return ids have many values with a higher return than demand and sale.

- Product: The values of return, sale and demand for best sellers do not differ much from the rate of highest returns. The ids with the highest demand / return have these variables greater than the return. The highest return ids have many values with a higher return than demand and sale. High-return ids tend to have high demand and sales. There are many ids with higher returns on demand and sales.
- Route: The values of return, sale and demand for best sellers do not differ much from the rate of highest returns. Most analyzed ids have higher demand and sales than return.
- State: Most ids have higher return than demand and sales. States with high demand seem to have high returns.
- Town: The values of return, sale and demand for best sellers do not differ much from the rate of highest returns. Cities with high demand seem to have high returns. Many ids have greater return than demand and sale.
- Week: All week values are similar. Most analyzed ids have higher return than demand and sales.

Checking for of negative financial balance patterns.

I decided to analyze the financial loss. I called it a financial loss when the difference between the sale price and the return price was less than zero. So I added a column with the price differences to the training dataset and filtered the value records less than zero. Then I grouped the data by variables to plot the number of records and their value. I made two barplots: one with the number of records per group and the other with the value per group. For variables in many categories, I selected the first 100 records ordered by money loss (module).

```
# Adding a column with the demand value (sales minus returns)
train_data$demand_value <- train_data$Venta_hoy - train_data$Dev_proxima

# financial loss dataframe
finantial_loss <- train_data[train_data$demand_value < 0 ,
                             c(names(translatedNames), "demand_value")]

# Checking the number of records with deficit values (negative demand value)
nrow(finantial_loss)
```

```
## [1] 81059
```

A brief description of what was observed in the graphs grouped by the variables:

- Agency: The number of agencies with a deficit in value is not highly correlated to their value. There is greater variation in the quantity compared to the values.
- Channel: Channels with a high return value have many records of monetary loss. Channel 1 has a greater financial loss, with amounts and registers' amount much higher than the others.
- Client: The number of clients with a deficit in value is not highly correlated to their value. There is a wide amplitude in the quantity and values. In general, the values and quantities per customer are low.
- Product: The number of products with a deficit in value is not highly correlated to their value. There is a wide variation in the quantity and values.
- Route: The values do not vary so much, the number of records per route is high.
- State: There is a wide amplitude in the quantity and values. The value and the quantity seem to be related.

- Town: The values do not vary so much, the number of records per town is high.
- Week: Values and quantities have very low variation. The value and the quantity seem to be related.

Finally, I decided to calculate the percentage of records with zero sales and the percentage of products with no sales.

```
# Calculating the percentage of null sales.
round((length(train_data[train_data$Venta_uni_hoy == 0, "Venta_uni_hoy"])/nrow(train_data)) * 100, 2)
```

```
## [1] 0.4
```

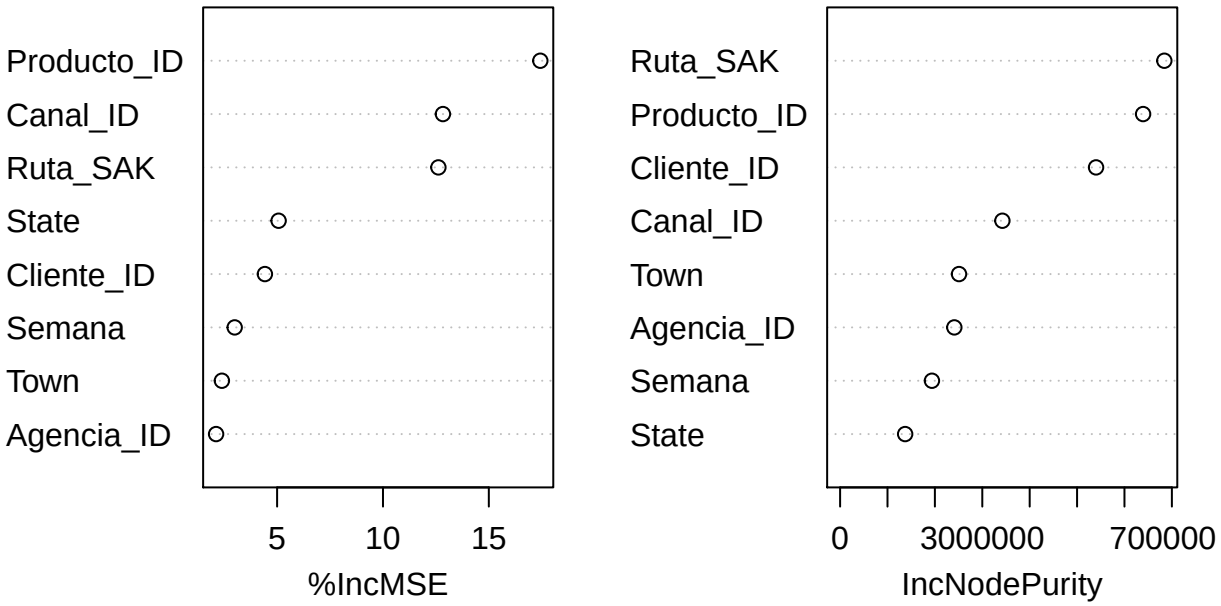
```
# Calculating the percentage of products that had no sales.
num_zero_sales <- train_data %>%
  group_by(Producto_ID) %>%
  summarise(sale_count = sum(Venta_uni_hoy)) %>%
  filter(sale_count == 0) %>%
  data.frame() %>%
  nrow()

round((num_zero_sales/length(unique(train_data$Producto_ID))) * 100, 2)
```

```
## [1] 3
```

Feature selection

To choose categorical variables, I created a random forest model using the sample train dataset.

```
# Feature selecting categorical variables
# Converting categorical variables to integer:
train_data[, names(translatedNames)] <- as.data.frame(lapply(train_data[, names(translatedNames)], as.i
sample_train_data[, names(translatedNames)] <- as.data.frame(lapply(sample_train_data[, names(translate
                                                              as.integer))

# Creating formula before creating model
Formula <- "Demanda_uni_equil ~ "
for (name in names(translatedNames)) { Formula <- paste(Formula, name, "+") }
Formula <- substr(Formula, start = 1, stop = nchar(Formula)-2)

# randomForest model using sample train dataset for avoid crashing.
select_model <- randomForest::randomForest (eval(parse(text = Formula)), data = sample_train_data,
                              ntree = 100, nodesize = 10,
                              importance = TRUE)

# Plotting features' importance
randomForest::varImpPlot(select_model)
```



Analyzing the importance plot I chose to use the following variables: Producto_ID, Ruta_SAK, Channel_ID and Agencia_ID.

Create Model

I created five models to choose the one with the best performance: logistic regression, rigde regression, random forest, k-nearest neighbor and xgboost. Before, I normalized the target variable. For the construction of the model, I used only the dependent categorical variables with a number of levels less than or equal to 100. To be able to perform the tests on my machine, I selected smaller samples of the training and test datasets in the proportion 70% and 30% respectively.

```
# Creating numeric feature for state
# Train dataset
train_data$StateNum <- train_data$State
levels(train_data$StateNum) <- seq(1:33)
train_data$StateNum <- as.integer(train_data$StateNum)

#Test dataset
test_data$StateNum <- test_data$State
levels(test_data$StateNum) <- seq(1:33)
test_data$StateNum <- as.integer(test_data$StateNum)

# Converting categorical features to numeric
for (name in ModelColNames) { train_data[, name] <- as.integer(train_data[, name])
                              test_data[, name] <- as.integer(test_data[, name]) }

# Saving transformed dataset
# write.csv(train_data[, c(ModelColNames, "Demanda_uni_equil")], "data/model_train_data.csv")
```

```r
# write.csv(test_data[, c(ModelColNames, "Demanda_uni_equil")], "data/model_test_data.csv")

model_train <- train_data[, c(ModelColNames, "Demanda_uni_equil")]
model_test <- test_data[, c(ModelColNames, "Demanda_uni_equil")]

# Using variables with cardinality less than or equal to 100.
Formula <- "Demanda_uni_equil ~ Canal_ID + StateNum + Semana"

# Normalize integer feature
for (name in names(translatedNames)[1:6]) { train_data[, name] <- normalize(train_data[, name]) }

# Getting data
# Train
# model_train <- read.csv(unz("data/model_train_data.zip", "model_train_data.csv"))
# model_train <- model_train[,2:ncol(model_train)]

# Test
# model_test <- read.csv(unz("data/model_test_data.zip", "model_test_data.csv"))
# model_test <- model_test[2:ncol(model_test)]

# Normalising train dataset
norm_model_train <- model_train
norm_model_train$Demanda_uni_equil <- normalize(norm_model_train$Demanda_uni_equil)


# Normalising test dataset
norm_model_test <- model_test
norm_model_test$Demanda_uni_equil <- equalNormalize(x=norm_model_test$Demanda_uni_equil,
                                                    y=model_train$Demanda_uni_equil)

# Getting dataset train sample for
set.seed(10000000)
index <- sample(1:nrow(norm_model_train), size=100000)
sample_norm_model_train <- norm_model_train[index,]

# Getting test dataset sample in proportion: training (70%) test (30%)
index <- sample(1:nrow(norm_model_test), size=round(100000*3/7))
sample_norm_test <- norm_model_test[index,]

# Logistic regression
logistic <- glm(formula = eval(parse(text = Formula)), data = sample_norm_model_train, family = "binomi
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
# randomForest model
forest <- randomForest::randomForest (eval(parse(text = Formula)), data = sample_norm_model_train,
                                      ntree = 100, nodesize = 10)

# Ridge Regression model
RidgeMod <- ridge::linearRidge(eval(parse(text = Formula)), data = sample_norm_model_train)

# k-nearest neighbor
knnTestPred <- DMwR::kNN(eval(parse(text = Formula)),
                         sample_norm_model_train,
```

```
                        sample_norm_test[,1:7],
                        norm=FALSE, k=13)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
knnTrainPred <- DMwR::kNN(eval(parse(text = Formula)),
                        sample_norm_model_train,
                        sample_norm_model_train[,1:7],
                        norm=FALSE, k=13)


# xgBoost model
xgboostmodel <- xgboost::xgboost(data = as.matrix(sapply(sample_norm_model_train[1:7], as.numeric)),
                                 label = sample_norm_model_train$Demanda_uni_equil,
                                 max.depth = 2, eta = 1, nthread = 2, nrounds = 2,
                                 objective = "binary:logistic", verbose = 1)
```

```
## [1]  train-error:0.001493
## [2]  train-error:0.001493
```

Score Models

After creating the models, I evaluated their scores. I calculated the square root of the mean error (RMSD) and the Mean absolute error (MAE) for each model for both train and test dataset:

```
# Predict
# logistic prediction
predicted <- data.frame(true_values = sample_norm_test$Demanda_uni_equil,
                        logistic = predict(logistic, sample_norm_test[,1:7], type="response"))

# random forest prediction
predicted$randomForest <- predict(forest, sample_norm_test[,1:7])

# ridge regression prediction
predicted$ridge <- predict(RidgeMod, sample_norm_test[,1:7])

# k-nearest neighbor prediction
predicted$knn <- as.numeric(knnTestPred)

# xgboost prediction
predicted$xgboost <- predict(xgboostmodel, as.matrix(sample_norm_test[,1:7]))


# appending predictions of the training datasets for calculating RMSD and MAE.
# logistic
predictedTrain <- data.frame(true_values = sample_norm_model_train$Demanda_uni_equil,
                             logistic = predict(logistic, sample_norm_model_train[,1:7], type="response
```

```r
# random forest train prediction
predictedTrain$randomForest <- predict(forest, sample_norm_model_train[,1:7])

# ridge regression train prediction
predictedTrain$ridge <- predict(RidgeMod, sample_norm_model_train[,1:7])

# k-nearest neighbor train prediction
predictedTrain$knn <- as.numeric(knnTrainPred)

# xgboost train prediction
predictedTrain$xgboost <- predict(xgboostmodel, as.matrix(sample_norm_model_train[,1:7]))

# Calculating RMSD and MAE
# Creating dataframe to store values
error <- data.frame(Models =c("Logistic Regression", "Ridge regression", "Random Forest",
                              "k-nearest neighbors", "xgBoost"),
                    Train_MAE = c(0,0,0,0,0),
                    Train_RMSE = c(0,0,0,0,0),
                    Test_MAE = c(0,0,0,0,0),
                    Test_RMSE = c(0,0,0,0,0))

# Getting MAE and RMSE for train prediction and test prediction
i <- 1
for (model in names(predictedTrain[2:6])) {
  error$Train_MAE[i] <-MAE(predictedTrain$true_values, predictedTrain[, model])
  error$Train_RMSE[i] <-RMSE(predictedTrain$true_values, predictedTrain[, model])
  error$Test_MAE[i] <-MAE(predicted$true_values, predicted[, model])
  error$Test_RMSE[i] <-RMSE(predicted$true_values, predicted[, model])
  i = i + 1
}

# Models it's RMSE and MAE
error
```

```
##                Models   Train_MAE  Train_RMSE     Test_MAE    Test_RMSE
## 1 Logistic Regression 0.001369024 0.004352634 0.0002195865 0.0006393147
## 2    Ridge regression 0.001253461 0.003975345 0.0004829048 0.0012560060
## 3       Random Forest 0.001357224 0.004333407 0.0002595084 0.0006165901
## 4 k-nearest neighbors 3.869657225 7.822355857 3.6510371750 5.5260722195
## 5             xgBoost 0.041065368 0.041198443 0.0409999550 0.0409999550
```

```r
# Smaller train_MAE model
error[error$Train_MAE == min(error$Train_MAE),]
```

```
##             Models   Train_MAE  Train_RMSE     Test_MAE   Test_RMSE
## 2 Ridge regression 0.001253461 0.003975345 0.0004829048 0.001256006
```

```r
# Smaller train_RMSE model
error[error$Train_RMSE == min(error$Train_RMSE),]
```

```
##             Models   Train_MAE  Train_RMSE     Test_MAE   Test_RMSE
## 2 Ridge regression 0.001253461 0.003975345 0.0004829048 0.001256006
```

```r
# Smaller test_MAE model
error[error$Test_MAE == min(error$Test_MAE),]
```

```
##                Models   Train_MAE  Train_RMSE     Test_MAE    Test_RMSE
## 1 Logistic Regression 0.001369024 0.004352634 0.0002195865 0.0006393147
```

```
# Smaller test_RMSE model
error[error$Test_RMSE == min(error$Test_RMSE),]
```

```
##          Models   Train_MAE  Train_RMSE     Test_MAE    Test_RMSE
## 3 Random Forest 0.001357224 0.004333407 0.0002595084 0.0006165901
```

I chose Logistic regression model for presenting low RMSE and MAE values for test dataset.

Model evaluation and optimization.

Once the model was chosen, I tried to optimize it and evaluate its performance. I ran a summary of the model and noticed that the variable "Canal_ID" has high significance. Then I tried to normalize all the dependent variables and calculate the MAE and the RMSD and the RMSD and there was no change in the values. So, I increased the weight of the channel variable (as it has high significance) and recalculated the values. Again, the MAE and the RMSD have not been changed.

```
# Summary model
summary(logistic)
```

```
##
## Call:
## glm(formula = eval(parse(text = Formula)), family = "binomial",
##     data = sample_norm_model_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.11199  -0.03046  -0.02228  -0.00288   2.13040
##
## Coefficients:
##              Estimate Std. Error z value             Pr(>|z|)
## (Intercept) -6.771982   0.253416 -26.723 < 0.0000000000000002 ***
## Canal_ID     0.189809   0.040758   4.657           0.00000321 ***
## StateNum    -0.001452   0.010280  -0.141                0.888
## Semana       0.002081   0.040662   0.051                0.959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 269.79  on 99999  degrees of freedom
## Residual deviance: 254.24  on 99996  degrees of freedom
## AIC: 316.97
##
## Number of Fisher Scoring iterations: 9
```

```
# Modifing dataframe to try to improve logistic regression model
new_train_df <- sample_norm_model_train[,c("Canal_ID", "StateNum", "Semana", "Demanda_uni_equil")]
new_test_df <- sample_norm_test[,c("Canal_ID", "StateNum", "Semana", "Demanda_uni_equil")]

# Normalizing all variables
new_train_df[, 1:3] <- data.frame(lapply(new_train_df[, 1:3], normalize))
new_test_df[, 1:3] <- data.frame(mapply(equalNormalize, new_test_df[, 1:3],
```

```
                                        sample_norm_model_train[, c("Canal_ID", "StateNum", "Semana")]))
```

```
# Using variables with cardinality less than or equal to 100.
Formula <- "Demanda_uni_equil ~ Canal_ID + StateNum + Semana"

# New model
newLogistic <- glm(formula = eval(parse(text = Formula)), data = new_train_df, family = "binomial")
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
# Evaluating new model
MAE(new_test_df$Demanda_uni_equil, predict(newLogistic, new_test_df[, 1:3], type="response"))
```

```
## [1] 0.0002195865
```

```
RMSE(new_test_df$Demanda_uni_equil, predict(newLogistic, new_test_df[, 1:3], type="response"))
```

```
## [1] 0.0006393147
```

```
# RMSE and MAE values didn't change, so I used original values for dependent variables
new_train_df <- sample_norm_model_train[,c("Canal_ID", "StateNum", "Semana", "Demanda_uni_equil")]
new_test_df <- sample_norm_test[,c("Canal_ID", "StateNum", "Semana", "Demanda_uni_equil")]

# As the variable "Canal_ID" has a high level of significance, I will increase its weight
# Multiplying channel by 3
new_train_df$Canal_ID <- 10*new_train_df$Canal_ID
new_test_df$Canal_ID <- 10*new_test_df$Canal_ID

# New model
newLogistic <- glm(formula = eval(parse(text = Formula)), data = new_train_df, family = "binomial")
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
# Evaluating new model
MAE(new_test_df$Demanda_uni_equil, predict(newLogistic, new_test_df[, 1:3], type="response"))
```

```
## [1] 0.0002195865
```

```
RMSE(new_test_df$Demanda_uni_equil, predict(newLogistic, new_test_df[, 1:3], type="response"))
```

```
## [1] 0.0006393147
```

RMSE and MAE values didn't change, so I evaluated original logistic model. I changed the predicted and actual values to their original non-formalized form and calculated the MAE and RMSD values.

```
# Getting not-normalized data
x <- round(original_values(x=predicted$logistic, y=model_train$Demanda_uni_equil))
y <- round(original_values(x=predicted$true_values, y=model_train$Demanda_uni_equil))
z <- round(original_values(x=predicted$randomForest, y=model_train$Demanda_uni_equil))

# not normalized data MAE and RMSE
MAE(x, y)
```

```
## [1] 0.9650699
```

```
RMSE(x, y)
```

```
## [1] 3.107091
```

The final version of the model presents MAE = 0.97 and RMSD = 3.12 for the evaluated samples.