Projeto02.R

rodrigolima82

2019-06-04

```
### Prevendo Demanda de Estoque com Base em Vendas ###
## Formacao Cientista de Dados - DSA (https://www.datascienceacademy.com.br)##

### Preparacao e Carregamento dos datasets
# Os arquivos sao grandes por isso segue o caminho para baixar
# https://www.kaggle.com/c/grupo-bimbo-inventory-demand/data

# Verifica se existe o arquivo RDA no disco
# Se existir carrega o arquivo
# Carregando os Pacotes
library(data.table)
```

```
### Carregando o arquivo de Treino ###

trainData = "dados/train.data.Rda"

if (file.exists(trainData)) {
  dfTreino <- readRDS(file=trainData)
  rm('trainData')
}else{
  # Carregando o Dataset "TRAIN.CSV"
  # Os dados brutos contem 11 colunas (atributos).
  # A coluna "Demanda_uni_equil" eh o alvo.
  dtTreino  <- fread("dados/train.csv")
  dtCliente <- fread("dados/cliente_tabla.csv")
  dtProduto <- fread("dados/producto_tabla.csv")
  dtCidade  <- fread("dados/town_state.csv")

  # Criacao dos DataFrame
  dfTreino  <- as.data.frame(dtTreino)
  dfCliente <- as.data.frame(dtCliente)
  dfProduto <- as.data.frame(dtProduto)
  dfCidade  <- as.data.frame(dtCidade)

  # Remover os DataTable
  rm('dtTreino')
  rm('dtCliente')
  rm('dtProduto')
  rm('dtCidade')

  # Merge dos DataFrame
  dfTreino <- merge(dfTreino, dfCidade, by=c("Agencia_ID"))
  dfTreino <- merge(dfTreino, dfCliente, by=c("Cliente_ID"))
  dfTreino <- merge(dfTreino, dfProduto, by=c("Producto_ID"))

  # Salvando DF no disco para acelerar o trabalho
  saveRDS(dfTreino, file="dados/train.data.Rda")
```

```
  # Remover os DataFrame secundarios
  rm('dfCidade')
  rm('dfCliente')
  rm('dfProduto')

  # Os dados brutos apos o merge ficou com 15 colunas (atributos).
  # Visualizando dados do dataframe
  #View(dfTreino)
  #str(dfTreino)

}
```

```
# Carregando o arquivo de Teste
# Utilizado caso queira prever as demandas dos produtos
# no dataset de teste disponibilizado no kaggle

# testData = "dados/test.data.Rda"
#
# if (file.exists(testData)) {
#   dfTeste <- readRDS(file=testData)
# }else{
#   # Carregando o Dataset "TEST.CSV"
#   dtTeste   <- fread("dados/test.csv")
#
#   # Criacao dos DataFrame
#   dfTeste   <- as.data.frame(dtTeste)
#
#   # Remover os DataTable
#   rm('dtTeste')
#
#   # Salvando DF no disco para acelerar o trabalho
#   saveRDS(dfTeste, file="test.data.Rda")
#
# }
```

```
# Nome das variaveis

# dfTreino
# Semana, Agencia_ID (+Town, +State), Canal_ID, Ruta_SAK, Cliente_ID (+NombreCliente), Producto_ID (+No

### Data fields
# Semana - Week number (From Thursday to Wednesday)
# Agencia_ID - Sales Depot ID
# Canal_ID - Sales Channel ID
# Ruta_SAK - Route ID (Several routes = Sales Depot)
# Cliente_ID - Client ID
# NombreCliente - Client name
# Producto_ID - Product ID
# NombreProducto - Product Name
# Venta_uni_hoy - Sales unit this week (integer)
# Venta_hoy - Sales this week (unit: pesos)
# Dev_uni_proxima - Returns unit next week (integer)
# Dev_proxima - Returns next week (unit: pesos)
# Demanda_uni_equil - Adjusted Demand (integer) (This is the target you will predict)
```

```
### Analise exploratoria de dados ###
# Creditos ao Fabiensvs: https://www.kaggle.com/fabienvs

# Carregando os Pacotes
library(dplyr)
```

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

## The following objects are masked from 'package:data.table':
##
##     between, first, last

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

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

```
library(Hmisc)
```

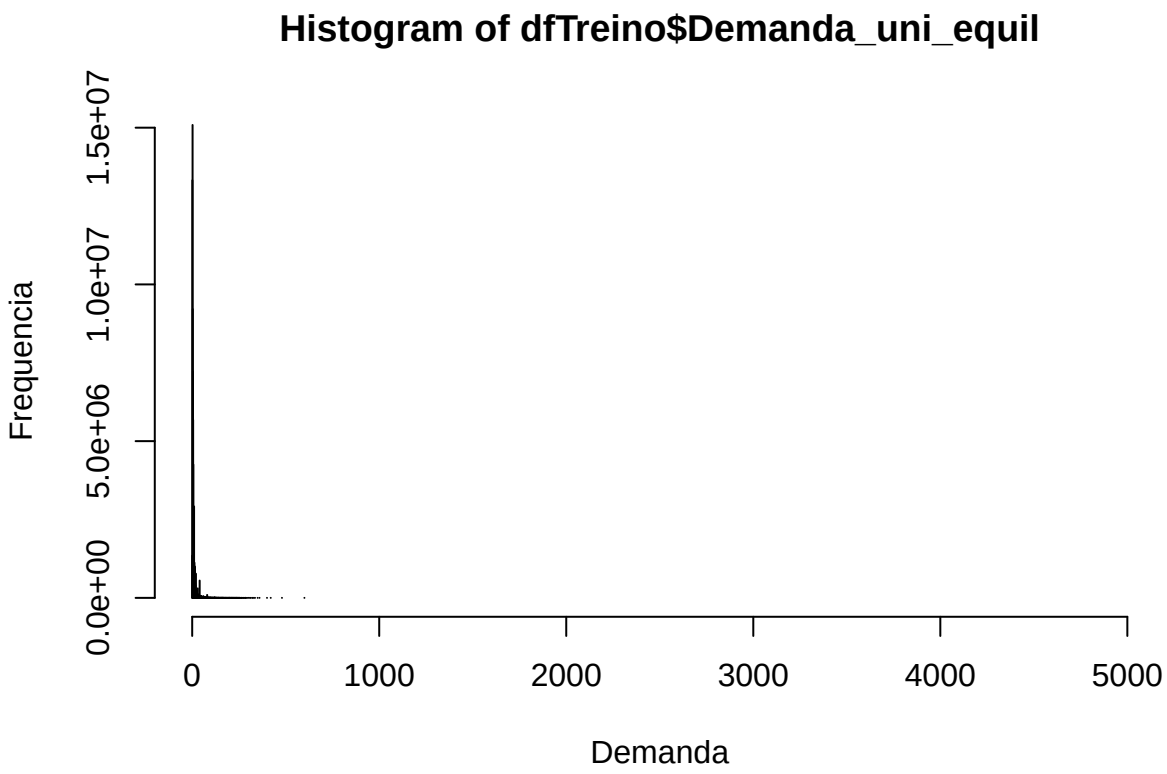
```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(stringr)
library(ggplot2)
library(scales)
library(treemap)

# Verificando os dados estatisticos da variavel 'Demanda_uni_equil'
# identificamos que 95% dos dados tem media 24 e que os maiores valores passam de 2000
# por isso serao removidos os valores de demanda acima de 30
describe(dfTreino$Demanda_uni_equil)
```

```
## dfTreino$Demanda_uni_equil
##        n  missing distinct     Info     Mean      Gmd      .05      .10
```

```
## 74773833        0     2091    0.983    7.255    8.988        1        1
##      .25      .50      .75      .90      .95
##        2        3        6       14       24
##
## lowest :    0    1    2    3    4, highest: 4975 4983 4997 4999 5000
```

```r
hist(dfTreino$Demanda_uni_equil, "FD", xlab="Demanda", ylab="Frequencia")
```




```r
# Avaliando a variavel PRODUTO e seus relacionamentos
# A variavel 'NombreProducto' contem alem do nome, suas caracteristicas (qtde pacotes, peso, marca)
head(dfTreino %>% distinct(NombreProducto))
```

```
##                          NombreProducto
## 1 Bimbollos Ext sAjonjoli 6p 480g BIM 41
## 2         Burritos Sincro 170g CU LON 53
## 3    Div Tira Mini Doradita 4p 45g TR 72
## 4      Pan Multigrano Linaza 540g BIM 73
## 5 Super Pan Bco Ajonjoli 680g SP WON 100
## 6     Wonder 100pct mediano 475g WON 106
```

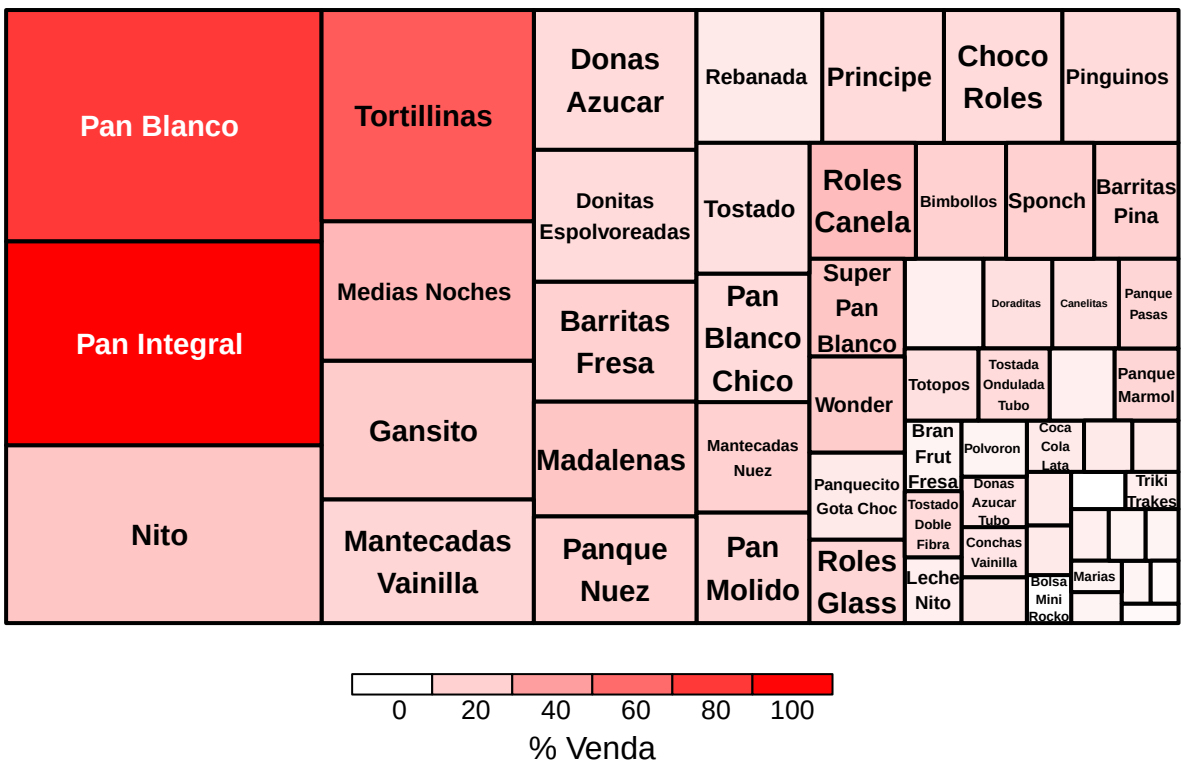
```r
# Primeiro, vamos extrair apenas o NOME DO PRODUTO da coluna NombreProducto
# Criando coluna 'Prod_nome'
# Removendo a matrix criada (temporaria)
m <- str_extract_all(dfTreino$NombreProducto, '^(\\D*)', simplify = TRUE)
dfTreino$Prod_nome <- as.vector(m)
head(dfTreino %>% distinct(Prod_nome))
```

```
##                  Prod_nome
## 1 Bimbollos Ext sAjonjoli
## 2         Burritos Sincro
## 3  Div Tira Mini Doradita
## 4   Pan Multigrano Linaza
```

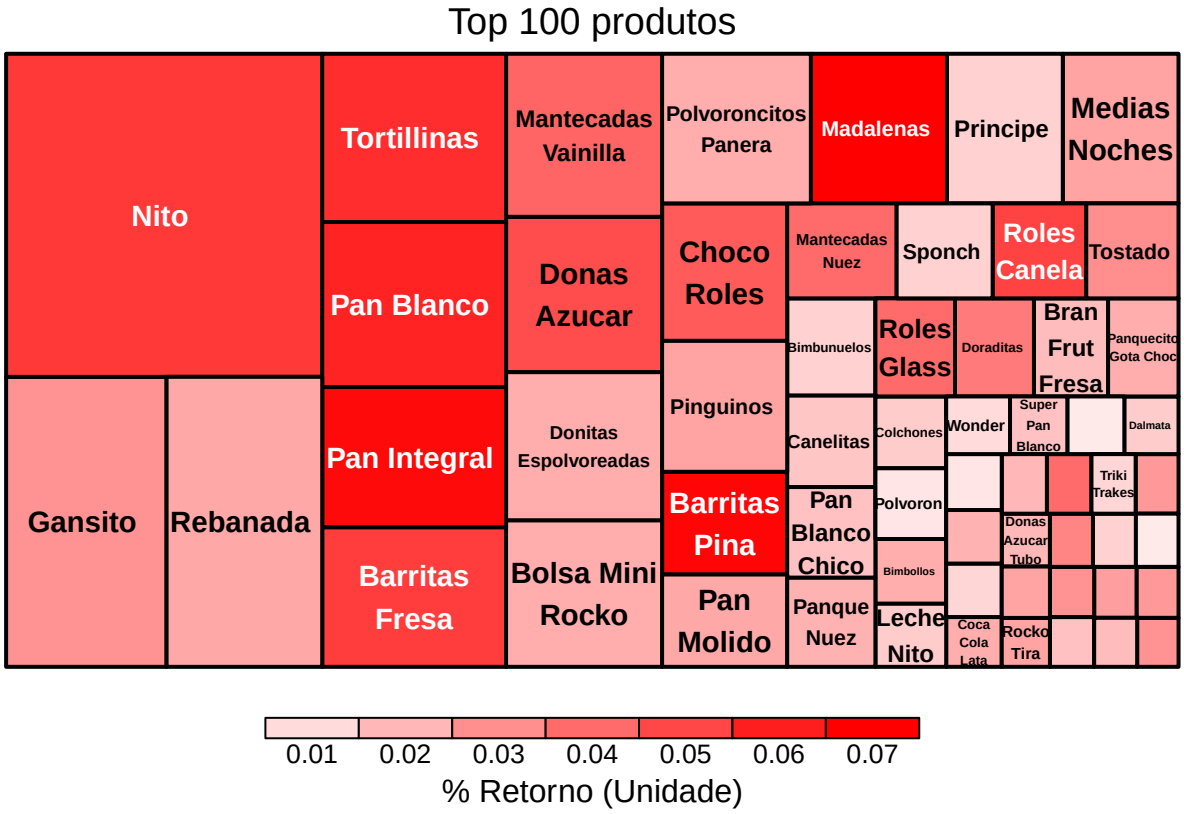
```
## 5  Super Pan Bco Ajonjoli
## 6                  Wonder
```

```
rm(m)
```

```
# Analisando dados de Produtos
produto <- dfTreino %>%
  group_by(Producto_ID, Prod_nome) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima),
            Liquida = sum(Demanda_uni_equil)) %>%
  mutate(Avg_Venda = Venda / Unid,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  filter(!is.nan(Avg_Venda)) %>%
  arrange(desc(Unid))

# Observa-se que os produtos com mais vendas sao Paes Branco e Integrais, alem do Nito e Tortillinas
treemap(produto[1:100, ],
        index=c("Prod_nome"), vSize="Venda", vColor="Avg_Venda",
        palette=c("#FFFFFF","#FFFFFF","#FF0000"),
        type="value", title.legend="% Venda", title="Top 100 produtos")
```

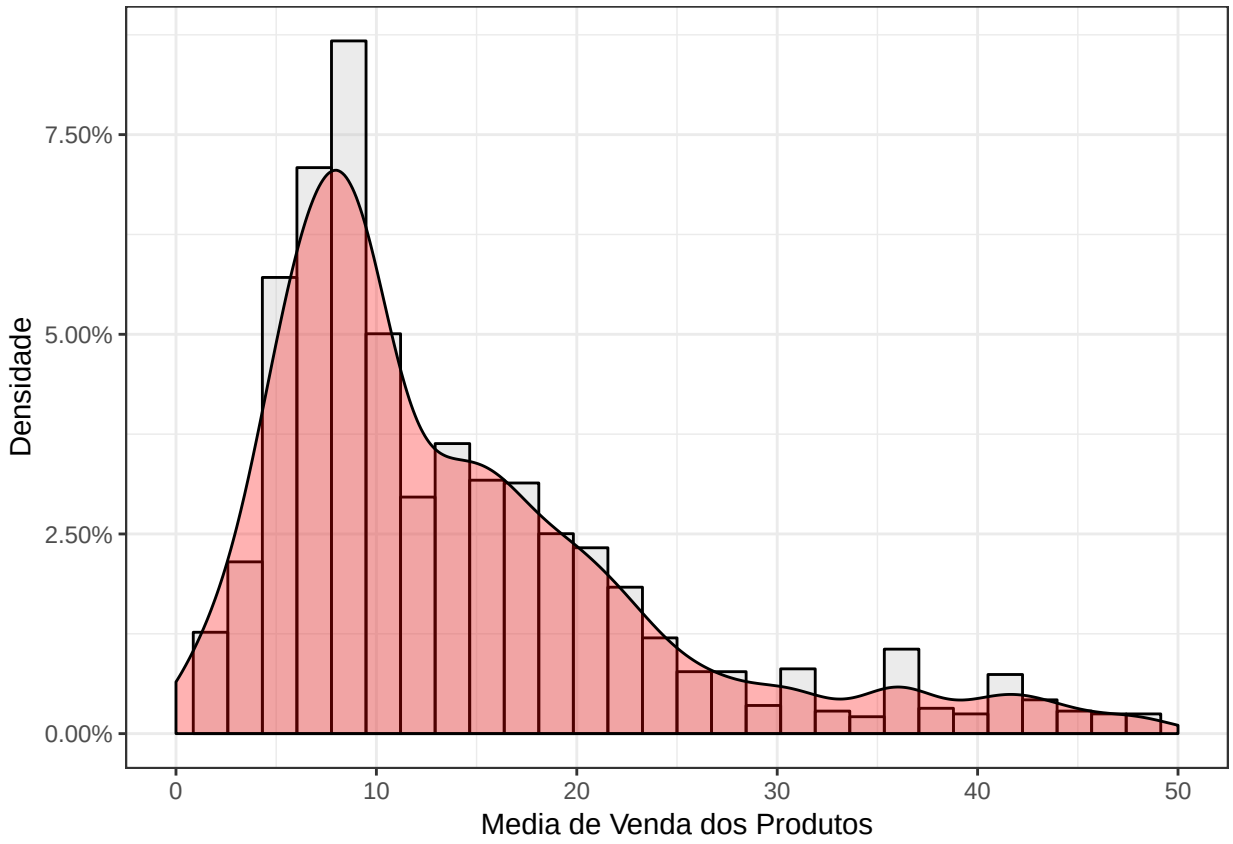



```
# Observa-se que os produtos com mais retorno de vendas sao Nito, Gansito e Rebanada
treemap(produto[1:100, ],
        index=c("Prod_nome"), vSize="Unid", vColor="Taxa_Ret",
        palette=c("#FFFFFF","#FFFFFF","#FF0000"),
        type="value", title.legend="% Retorno (Unidade)", title="Top 100 produtos")
```




```
# Visualizando os dados do histograma (media de venda dos produtos)
ggplot(produto, aes(x=Avg_Venda))+
  geom_histogram(aes(y=..density..), fill="gray", color="black", alpha="0.3")+
  geom_density(fill="red", alpha="0.3")+
  scale_x_continuous(name="Media de Venda dos Produtos", lim=c(0, 50))+
  scale_y_continuous(name="Densidade", labels=percent)+
  theme_bw()
```

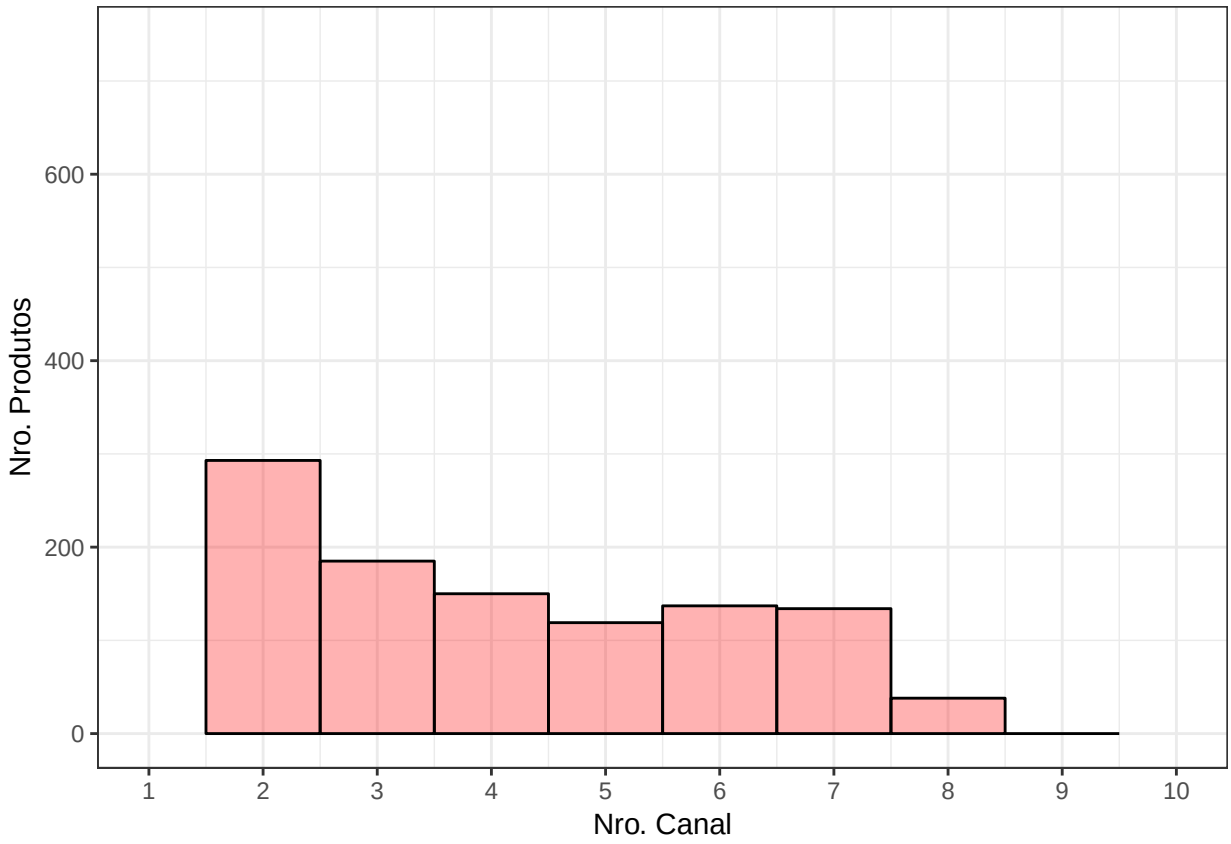
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 74 rows containing non-finite values (stat_bin).
## Warning: Removed 74 rows containing non-finite values (stat_density).
## Warning: Removed 2 rows containing missing values (geom_bar).
```




```
rm(produto)
```

```
# Analisando dados de Produtos por Canais
canal.produto <- dfTreino %>%
  group_by(Producto_ID) %>%
  summarise(n_canal = n_distinct(Canal_ID))

# Os produtos sao entregues por varios canais
ggplot(canal.produto)+
  geom_histogram(aes(x=n_canal), fill="red", color="black", alpha="0.3", binwidth=1)+
  scale_x_continuous(name="Nro. Canal", breaks=1:10, lim=c(1, 10))+
  scale_y_continuous(name="Nro. Produtos")+
  theme_bw()
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

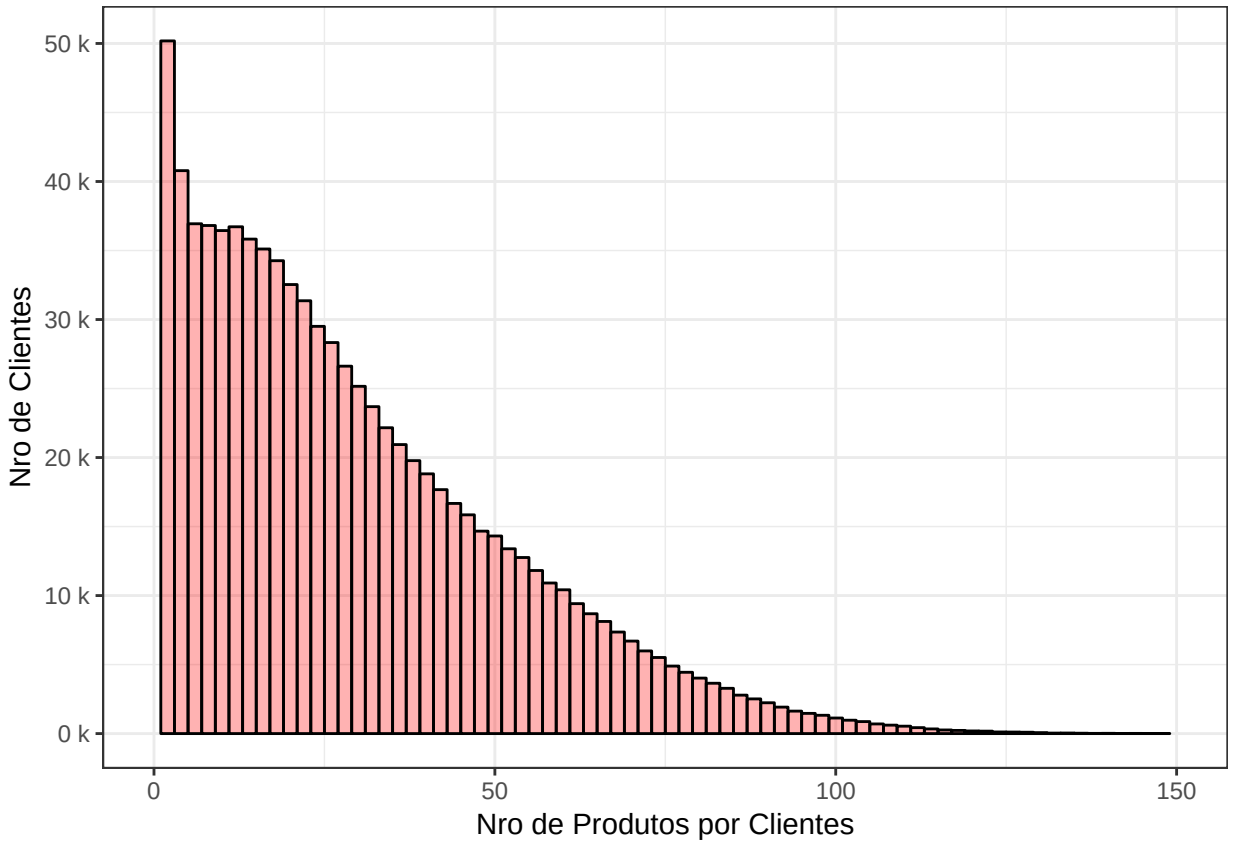



```
rm(canal.produto)
```

```
# Analisando dados de Produtos x Clientes
produto.cliente <- dfTreino %>%
  group_by(Cliente_ID) %>%
  summarise(n_produtos = n_distinct(Producto_ID))

ggplot(produto.cliente)+
  geom_histogram(aes(x=n_produtos), fill="red", color="black", alpha="0.3", binwidth=2)+
  scale_x_continuous(name="Nro de Produtos por Clientes", lim=c(0, 150))+
  scale_y_continuous(name="Nro de Clientes", labels=function(x)paste(x/1000, "k"))+
  theme_bw()
```

```
## Warning: Removed 47 rows containing non-finite values (stat_bin).
```

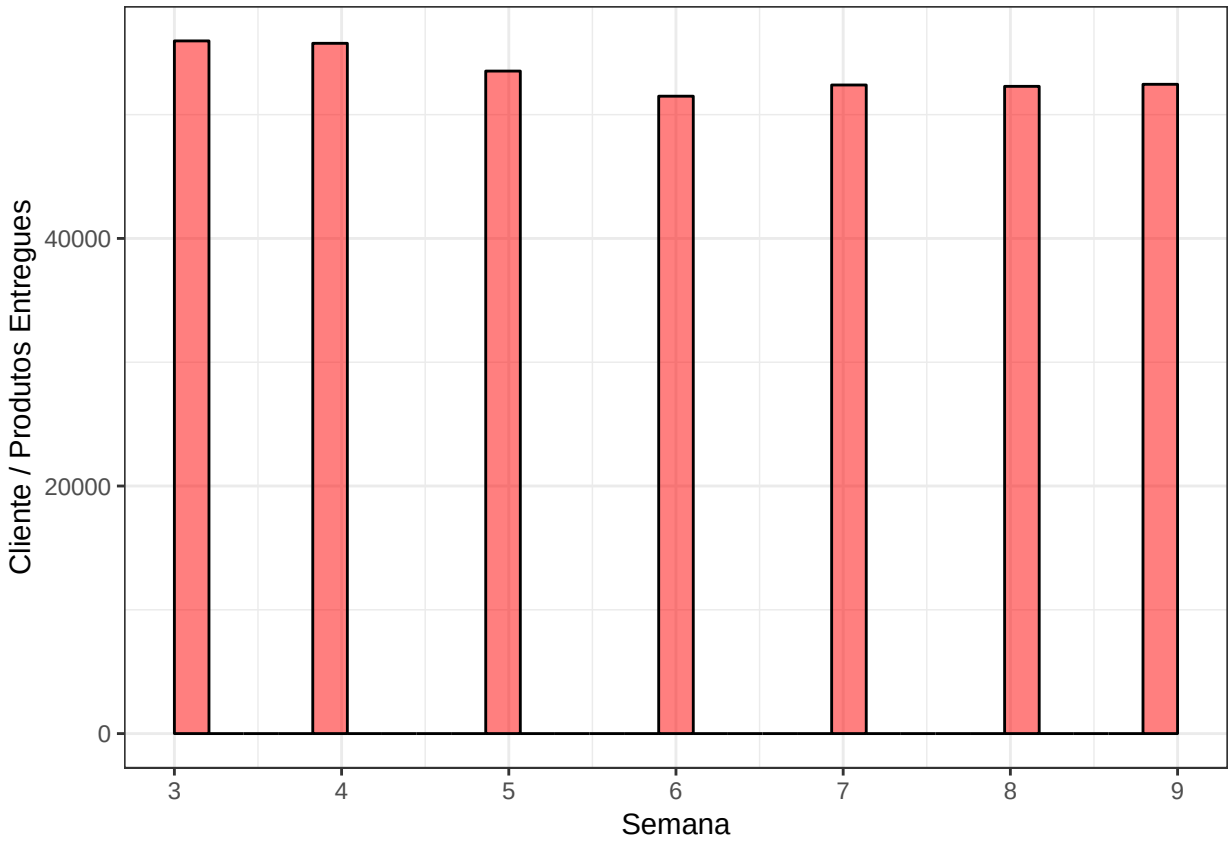
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



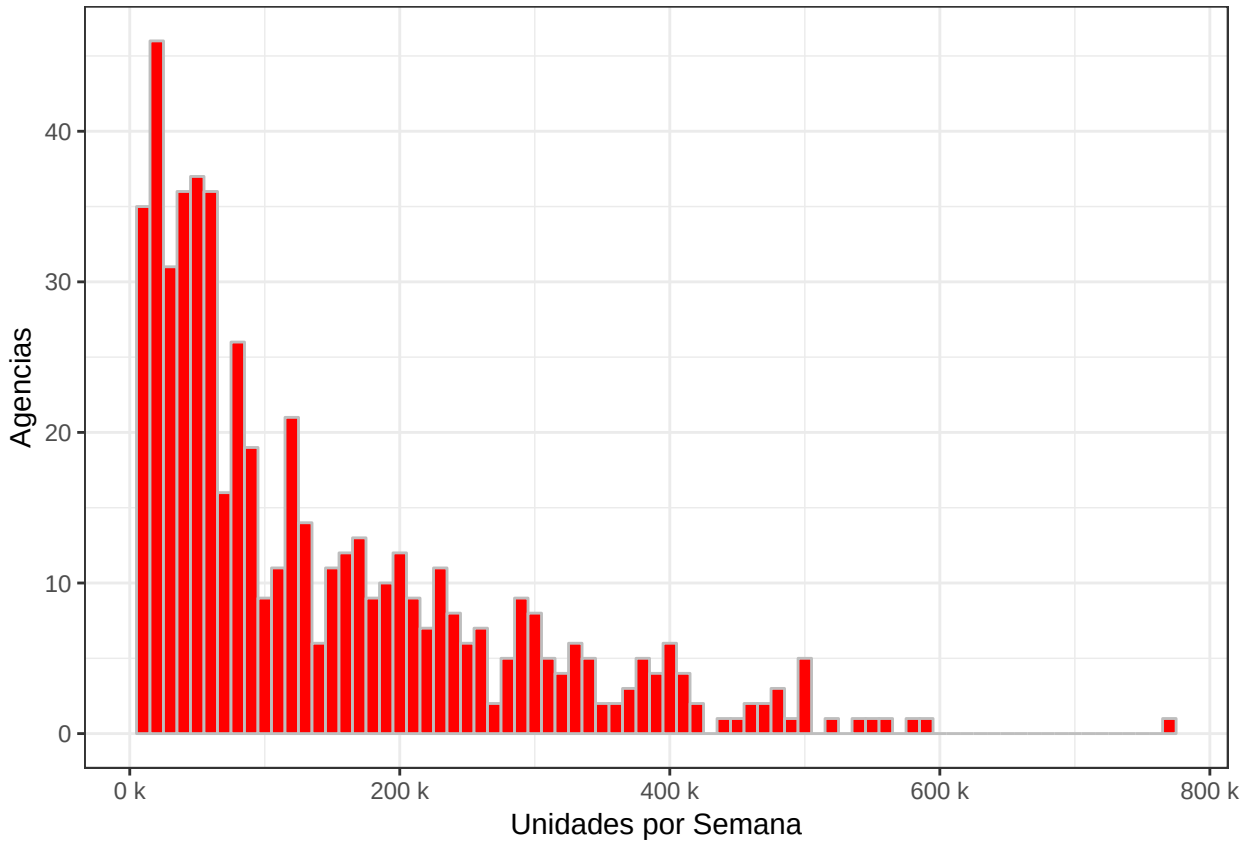

```
rm(produto.cliente)
```

```
# Analisando dados das Semanas
ggplot(dfTreino %>% sample_frac(0.005))+
  geom_histogram(aes(x=Semana), color="black", fill="red", alpha=0.5)+
  scale_x_continuous(breaks=1:10)+
  scale_y_continuous(name="Cliente / Produtos Entregues")+
  theme_bw()
```

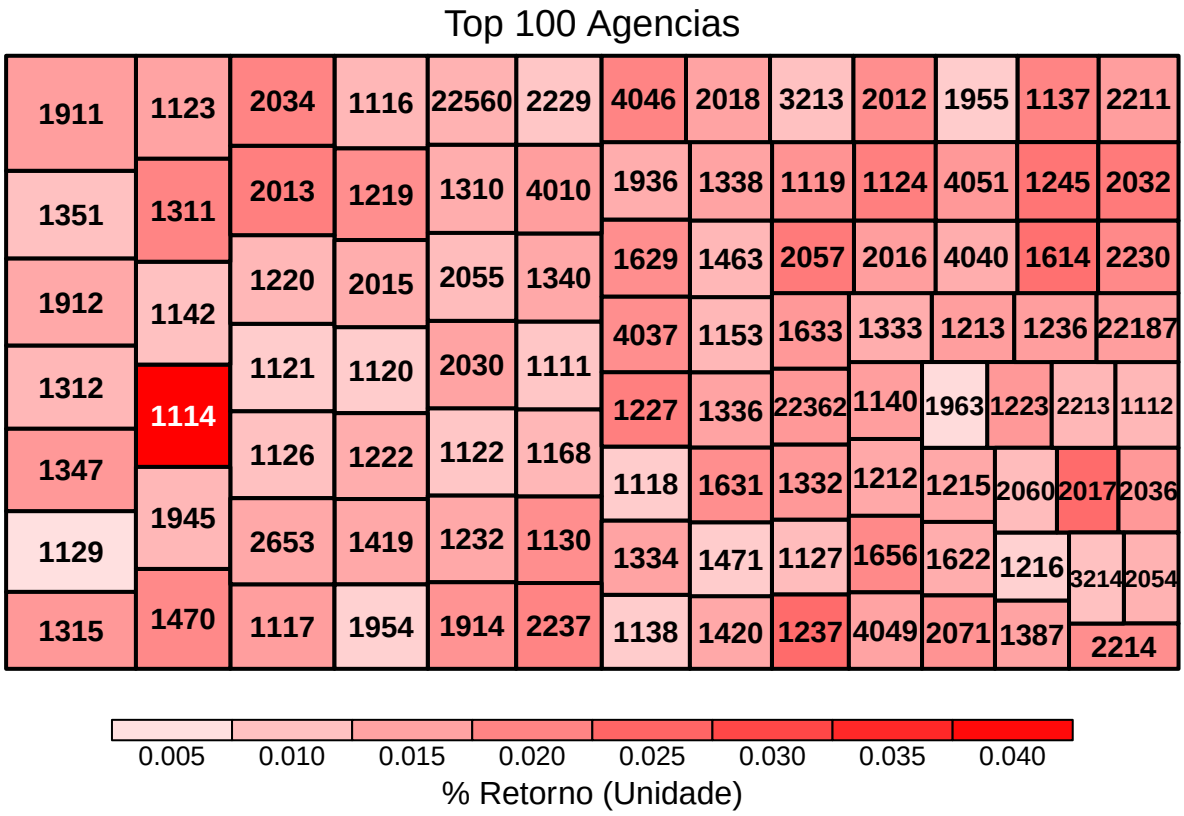
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Analisando dados das Agencias
agencia <- dfTreino %>%
  group_by(Agencia_ID) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima),
            Liquida = sum(Demanda_uni_equil)) %>%
  mutate(Venda_Liquida = Venda - Ret_Venda,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  arrange(desc(Unid))

ggplot(agencia, aes(x=Unid/7))+
  geom_histogram(fill="red", color="gray", binwidth=10000)+
  scale_x_continuous(name="Unidades por Semana", labels=function(x)paste(x/1000, "k"))+
  scale_y_continuous(name="Agencias")+
  theme_bw()
```

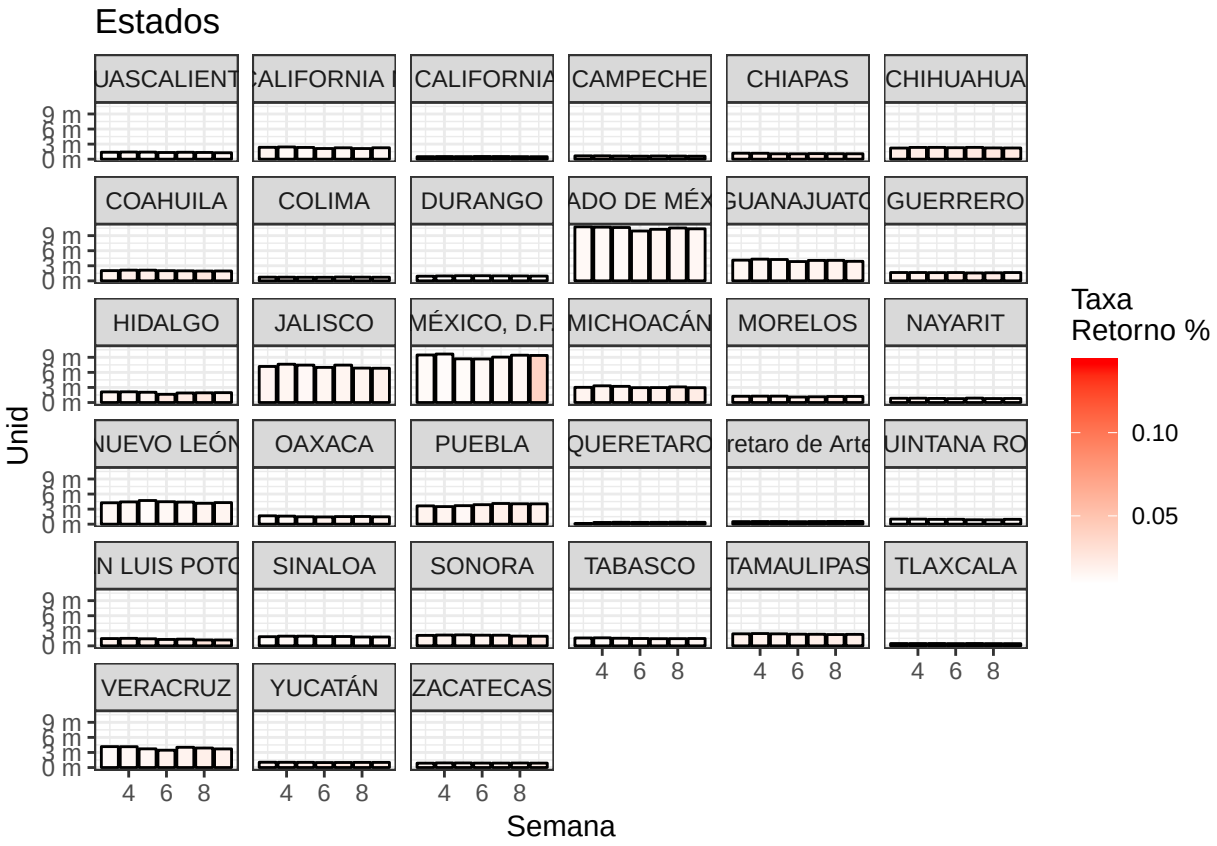



```
# Observa-se que a agencia 1114 tem uma alta taxa de retorno das unidades
treemap(agencia[1:100, ],
        index=c("Agencia_ID"), vSize="Unid", vColor="Taxa_Ret",
        palette=c("#FFFFFF","#FFFFFF","#FF0000"),
        type="value", title.legend="% Retorno (Unidade)", title="Top 100 Agencias")
```



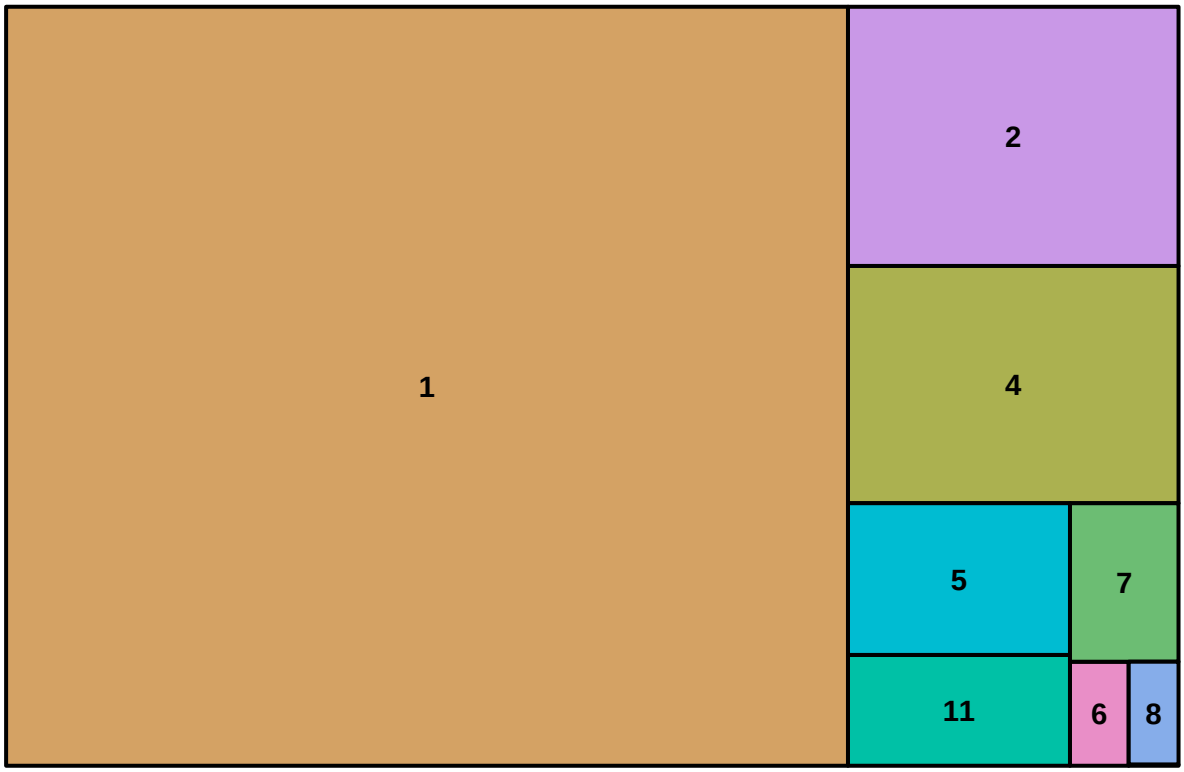

```
rm(agencia)
```

```
# Analisando dados de Estados
estado <- dfTreino %>%
  group_by(State, Semana) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima),
            Liquida = sum(Demanda_uni_equil)) %>%
  mutate(Avg_Venda = Venda / Unid,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  arrange(desc(Unid))

# Observa-se que apenas alguns estados tem um alto volume de venda semanal
ggplot(estado)+
  geom_bar(aes(x=Semana, y=Unid, fill=Taxa_Ret), stat="identity", color="black")+
  scale_y_continuous(labels=function(x)paste(x/1e6, "m"))+
  scale_fill_gradient(name="Taxa\nRetorno %", low="white", high="red")+
  facet_wrap(~State)+
  ggtitle("Estados")+
  theme_bw()
```

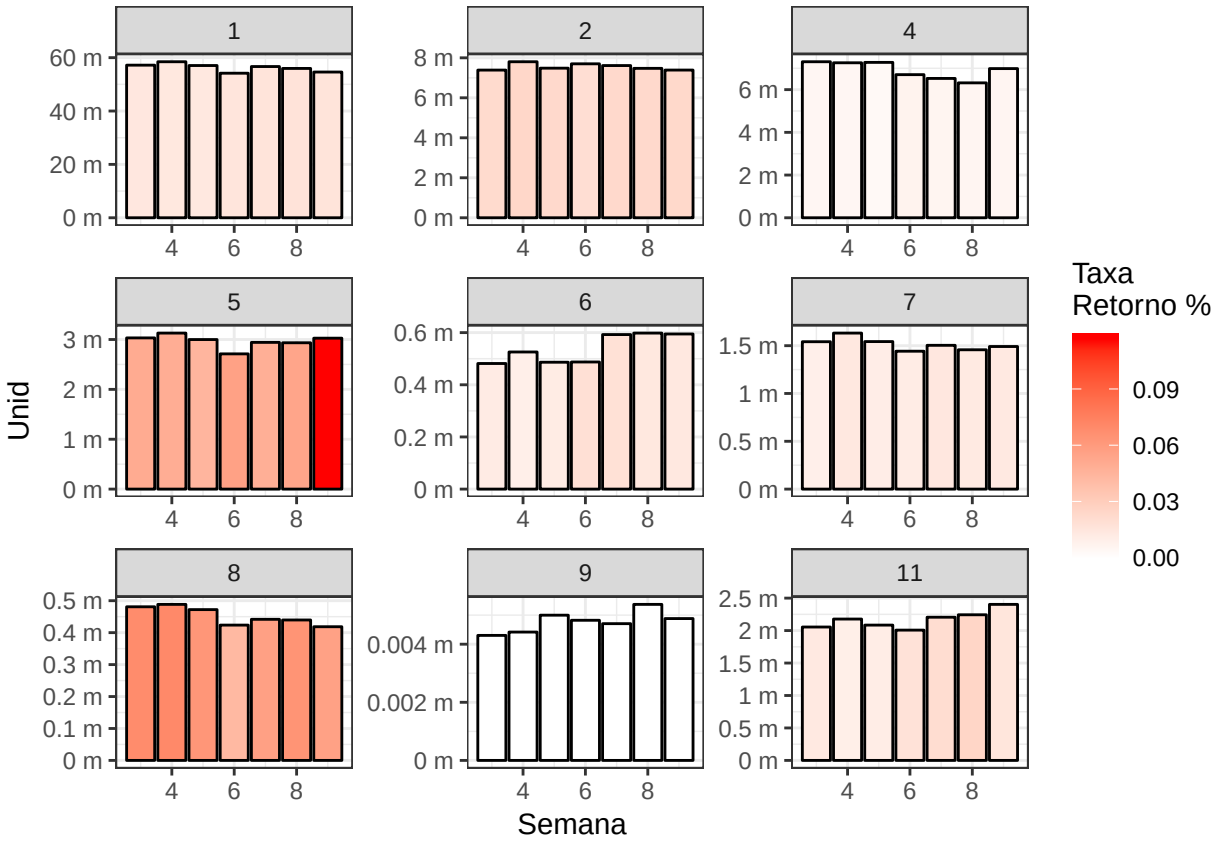



```
rm(estado)
```

```
# Analisando dados dos Canais de Venda
canal <- dfTreino %>%
  group_by(Canal_ID, Semana) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima),
            Liquida = sum(Demanda_uni_equil)) %>%
  mutate(Venda_Liquida = Venda - Ret_Venda,
         Avg_Venda = Venda / Unid,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  arrange(desc(Unid))

# Observa-se que o canal 1 tem muito mais movimentacao de Unidades
treemap(canal, index=c("Canal_ID"), vSize="Unid", type="index", title="Canais")
```

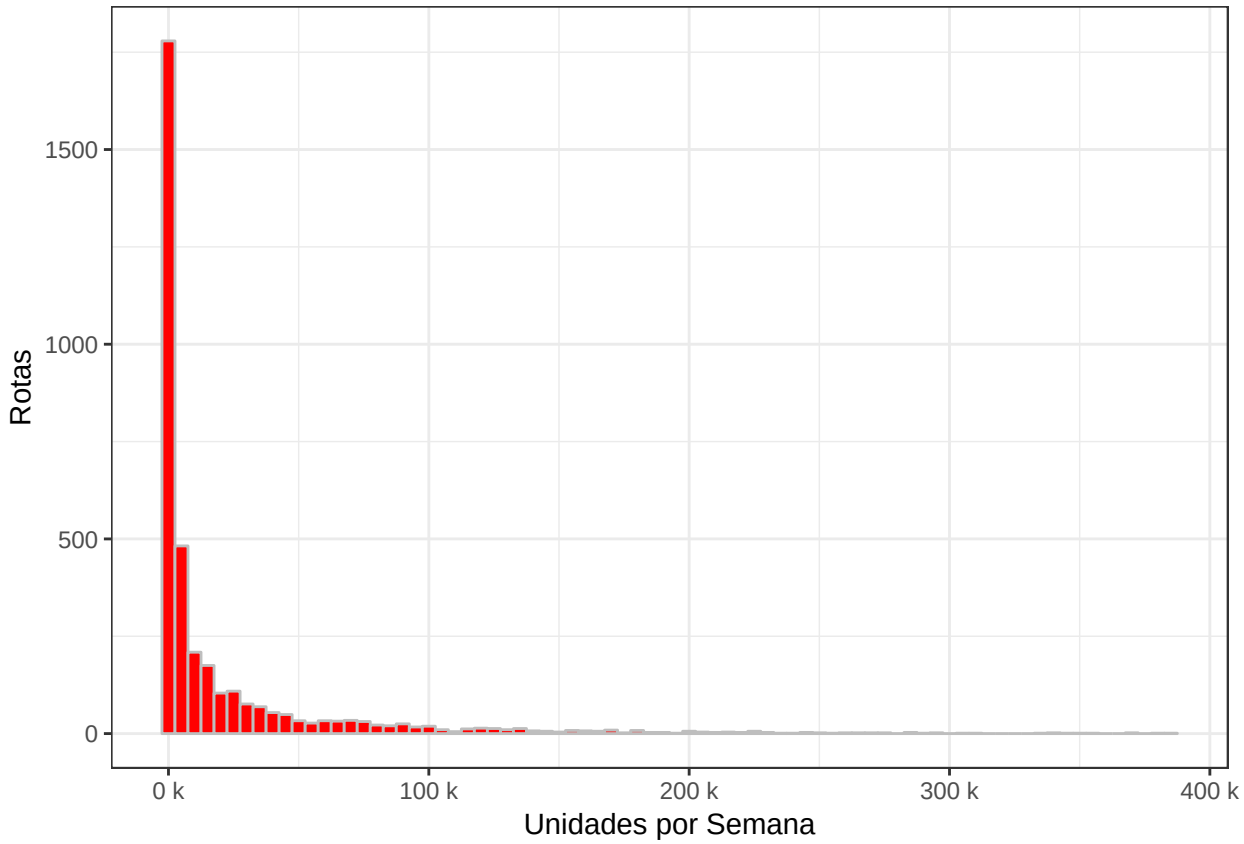



```
# Observa-se que o canal 5 e 8 tem maior taxa de retorno
ggplot(canal)+
  geom_bar(aes(x=Semana, y=Unid, fill=Taxa_Ret), stat="identity", color="black")+
  scale_y_continuous(labels=function(x)paste(x/1e6, "m"))+
  scale_fill_gradient(name="Taxa\nRetorno %", low="white", high="red")+
  facet_wrap(~Canal_ID, scale="free")+
  theme_bw()
```



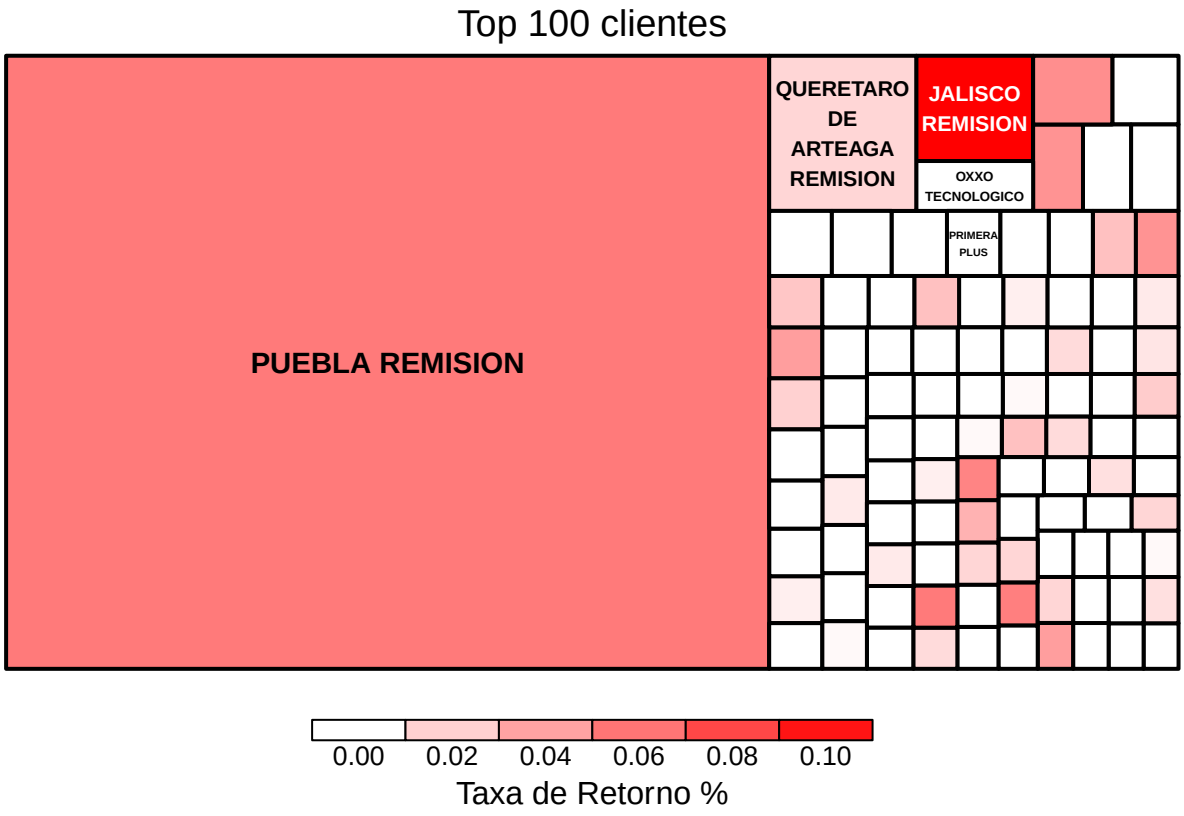

```
rm(canal)
```

```
# Analisando dados das Rotas
rotas <- dfTreino %>% group_by(Ruta_SAK) %>%
  summarise(n_Agencias = n_distinct(Agencia_ID),
            n_Clientes = n_distinct(Cliente_ID),
            Unid = sum(Venta_uni_hoy),
            Ret_Unid = sum(Dev_uni_proxima)) %>%
  mutate(Return_Rate = Ret_Unid / (Ret_Unid)) %>%
  arrange(desc(Unid))

# Observa-se que mais de 80% das rotas movimentam menos de 30k de produtos por semana
ggplot(rotas, aes(x=Unid/7))+
  geom_histogram(fill="red", color="gray", binwidth=5000)+
  scale_x_continuous(name="Unidades por Semana", labels=function(x)paste(x/1000, "k"))+
  scale_y_continuous(name="Rotas")+
  theme_bw()
```




```
rm(rotas)
```

```
# Analisando dados de Clientes
cliente <- dfTreino %>%
  group_by(Cliente_ID, NombreCliente) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima),
            Liquida = sum(Demanda_uni_equil)) %>%
  mutate(Venda_Liquida = Venda - Ret_Venda,
         Avg_Venda = Venda / Unid,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  arrange(desc(Unid))

# Observa-se que o maior cliente é 'PUEBLA REMISION'
treemap(cliente[1:100, ],
        index=c("NombreCliente"), vSize="Unid", vColor="Taxa_Ret",
        palette=c("#FFFFFF","#FFFFFF","#FF0000"),
        type="value", title.legend="Taxa de Retorno %", title="Top 100 clientes")
```




```
rm(cliente)
```

```
### FEATURE SELECTION (Selecao de Variaveis)

# Buscando uma amostra do DataFrame de Treino para facilitar a analise
dfSample <- sample_n(dfTreino, 50000)

# Verificar se existem valores ausentes (missing) em cada coluna
# Dados NA nao encontrados
any(is.na(dfSample))
```

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

```
### Aplicando Engenharia de Atributos em Variaveis ###
# Verificando a taxa de retorno
# Observa-se que 96% das unidades nao tem retorno e menos de 2% retornaram 1 unidade
table(dfSample$Dev_uni_proxima)
```

```
##
##     0     1     2     3     4     5     6     7     8     9    10    11
## 48286   847   360   145    88    67    29    15    19    13    24    11
##    12    13    14    15    16    17    18    19    20    21    22    24
##    12     3     7     8     4     4     3     3     9     3     2     3
##    25    26    27    28    29    30    31    32    33    37    40    42
##     1     1     4     3     1     2     1     1     3     1     6     1
##    43    44    45    46    47    54    57    90    94   120
##     1     1     1     1     1     1     1     1     1     1
```

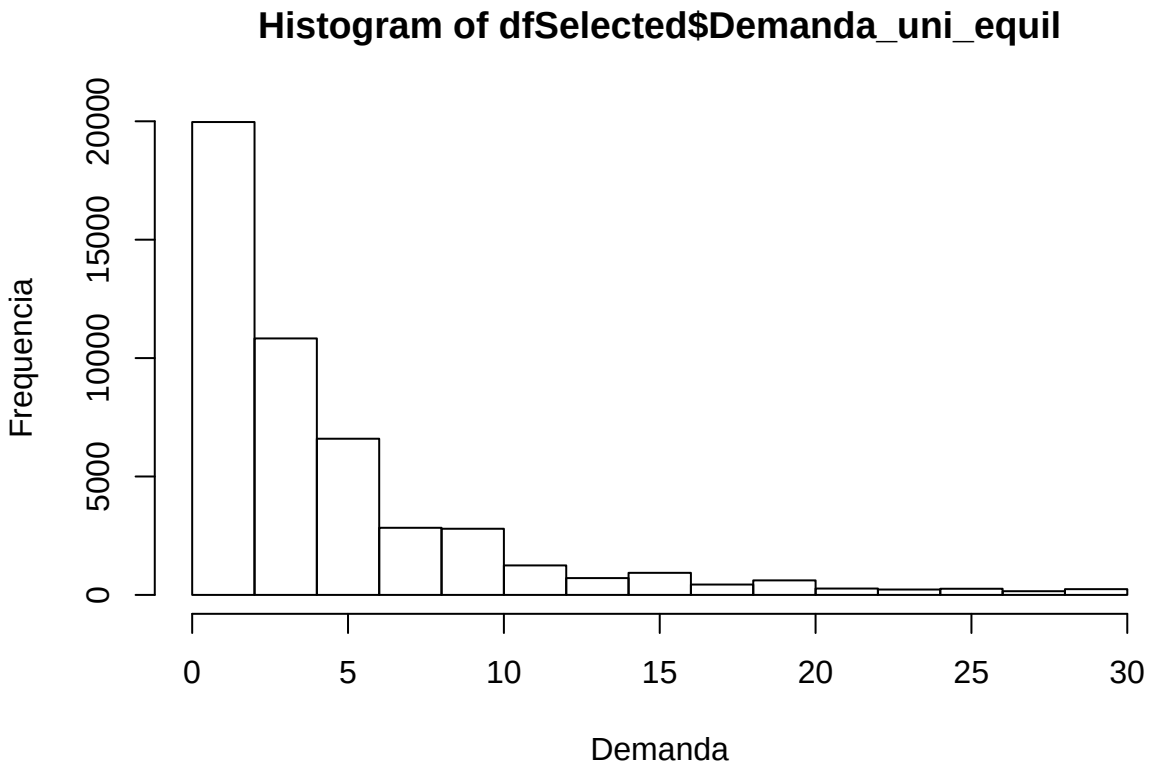
```
prop.table(table(dfSample$Dev_uni_proxima))
```

```
##
##       0       1       2       3       4       5       6       7       8
## 0.96572 0.01694 0.00720 0.00290 0.00176 0.00134 0.00058 0.00030 0.00038
##       9      10      11      12      13      14      15      16      17
## 0.00026 0.00048 0.00022 0.00024 0.00006 0.00014 0.00016 0.00008 0.00008
##      18      19      20      21      22      24      25      26      27
## 0.00006 0.00006 0.00018 0.00006 0.00004 0.00006 0.00002 0.00002 0.00008
##      28      29      30      31      32      33      37      40      42
## 0.00006 0.00002 0.00004 0.00002 0.00002 0.00006 0.00002 0.00012 0.00002
##      43      44      45      46      47      54      57      90      94
## 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002
##     120
## 0.00002
```

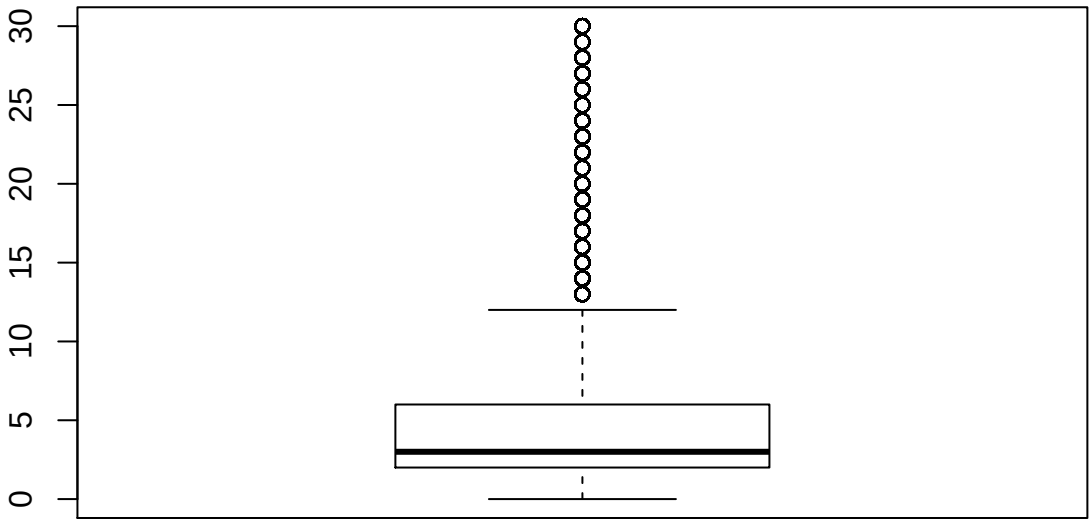
```
# Criando novas colunas no dataset de avaliacao
dfSelected <- dfSample %>%
  group_by(Semana, Agencia_ID, Canal_ID, Ruta_SAK, Cliente_ID, Producto_ID, Demanda_uni_equil) %>%
  summarise(Unid = sum(Venta_uni_hoy),
            Venda = sum(Venta_hoy),
            Ret_Unid = sum(Dev_uni_proxima),
            Ret_Venda = sum(Dev_proxima)) %>%
  mutate(Avg_Venda = Venda / Unid,
         Taxa_Ret = Ret_Unid / (Unid+Ret_Unid)) %>%
  filter(!is.nan(Avg_Venda))

# Removendo a variavel 'Unid' pois eh praticamente a mesma informacao da variavel target
# Ela permanecendo no dataset, prejudica a criacao do modelo
dfSelected$Unid <- NULL

# Verificando os dados estatisticos da variavel 'Demanda_uni_equil'
# identificamos que 95% dos dados tem media 24 e que os maiores valores passam de 2000
# por isso serao removidos os valores de demanda acima de 30
dfSelected <- dfSelected[-which(dfSelected$Demanda_uni_equil > 30),]
describe(dfSelected$Demanda_uni_equil)
```

```
## dfSelected$Demanda_uni_equil
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    48117        0       31     0.98    4.958    4.815        1        1
##      .25      .50      .75      .90      .95
##        2        3        6       11       16
##
## lowest :  0  1  2  3  4, highest: 26 27 28 29 30
```

```
hist(dfSelected$Demanda_uni_equil, xlab="Demanda", ylab="Frequencia")
```




```
boxplot(dfSelected$Demanda_uni_equil)
```




```
# Verificando a analise exploratoria, selecionamos somente o Canal 1
# pois tem o maior volume de dados e movimentacoes
#dfSelected <- dfSelected[which(dfSelected$Canal_ID %in% 1),]

# Transformando variaveis numericas em variaveis categoricas
# Funcao para converter variaveis para fator
to.factors <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- as.factor(df[[variable]])
  }
  return(df)
```

```
}

# Definindo as variaveis que serao transformadas em fator
# Convertendo variaveis para fator
categorical.vars <- c('Semana', 'Canal_ID')
dfSelected <- to.factors(dfSelected, variables = categorical.vars)
str(dfSelected)
```

```
## Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame':  48117 obs. of  12 variables:
##  $ Semana           : Factor w/ 7 levels "3","4","5","6",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Agencia_ID       : int  1110 1110 1110 1111 1111 1111 1111 1111 1111 1111 ...
##  $ Canal_ID         : Factor w/ 8 levels "1","2","4","5",..: 6 6 8 1 1 1 1 1 1 1 ...
##  $ Ruta_SAK         : int  3305 3316 3501 1003 1003 1003 1013 1013 1019 1020 ...
##  $ Cliente_ID       : int  2340436 2025225 4552320 49753 49920 1872816 1174627 1446502 1120431 67808
##  $ Producto_ID      : int  46772 325 1700 2233 693 1109 35727 2233 1160 1150 ...
##  $ Demanda_uni_equil: int  6 4 0 13 5 7 4 5 9 2 ...
##  $ Venda            : num  53.5 32.6 28 259.2 48 ...
##  $ Ret_Unid         : int  0 0 2 0 0 0 0 0 0 0 ...
##  $ Ret_Venda        : num  0 0 28 0 0 0 0 0 0 0 ...
##  $ Avg_Venda        : num  8.92 8.15 14 19.94 9.6 ...
##  $ Taxa_Ret         : num  0 0 0.5 0 0 0 0 0 0 0 ...
##  - attr(*, "groups")=Classes 'tbl_df', 'tbl' and 'data.frame':   48117 obs. of  7 variables:
##   ..$ Semana     : int  3 3 3 3 3 3 3 3 3 3 ...
##   ..$ Agencia_ID : int  1110 1110 1110 1111 1111 1111 1111 1111 1111 1111 ...
##   ..$ Canal_ID   : int  7 7 11 1 1 1 1 1 1 1 ...
##   ..$ Ruta_SAK   : int  3305 3316 3501 1003 1003 1003 1013 1013 1019 1020 ...
##   ..$ Cliente_ID : int  2340436 2025225 4552320 49753 49920 1872816 1174627 1446502 1120431 67808 ..
##   ..$ Producto_ID: int  46772 325 1700 2233 693 1109 35727 2233 1160 1150 ...
##   ..$ .rows      :List of 48117
##   .. ..$ : int 1
##   .. ..$ : int 2
##   .. ..$ : int 3
##   .. ..$ : int 4
##   .. ..$ : int 5
##   .. ..$ : int 6
##   .. ..$ : int 7
##   .. ..$ : int 8
##   .. ..$ : int 9
##   .. ..$ : int 10
##   .. ..$ : int 11
##   .. ..$ : int 12
##   .. ..$ : int 13
##   .. ..$ : int 14
##   .. ..$ : int 15
##   .. ..$ : int 16
##   .. ..$ : int 17
##   .. ..$ : int 18
##   .. ..$ : int 19
##   .. ..$ : int 20
##   .. ..$ : int 21
##   .. ..$ : int 22
##   .. ..$ : int 23
##   .. ..$ : int 24
##   .. ..$ : int 25
```

```
##   .. ..$ : int 26
##   .. ..$ : int 27
##   .. ..$ : int 28
##   .. ..$ : int 29
##   .. ..$ : int 30
##   .. ..$ : int 31
##   .. ..$ : int 32
##   .. ..$ : int 33
##   .. ..$ : int 34
##   .. ..$ : int 35
##   .. ..$ : int 36
##   .. ..$ : int 37
##   .. ..$ : int 38
##   .. ..$ : int 39
##   .. ..$ : int 40
##   .. ..$ : int 41
##   .. ..$ : int 42
##   .. ..$ : int 43
##   .. ..$ : int 44
##   .. ..$ : int 45
##   .. ..$ : int 46
##   .. ..$ : int 47
##   .. ..$ : int 48
##   .. ..$ : int 49
##   .. ..$ : int 50
##   .. ..$ : int 51
##   .. ..$ : int 52
##   .. ..$ : int 53
##   .. ..$ : int 54
##   .. ..$ : int 55
##   .. ..$ : int 56
##   .. ..$ : int 57
##   .. ..$ : int 58
##   .. ..$ : int 59
##   .. ..$ : int 60
##   .. ..$ : int 61
##   .. ..$ : int 62
##   .. ..$ : int 63
##   .. ..$ : int 64
##   .. ..$ : int 65
##   .. ..$ : int 66
##   .. ..$ : int 67
##   .. ..$ : int 68
##   .. ..$ : int 69
##   .. ..$ : int 70
##   .. ..$ : int 71
##   .. ..$ : int 72
##   .. ..$ : int 73
##   .. ..$ : int 74
##   .. ..$ : int 75
##   .. ..$ : int 76
##   .. ..$ : int 77
##   .. ..$ : int 78
##   .. ..$ : int 79
```

```
##   .. ..$ : int 80
##   .. ..$ : int 81
##   .. ..$ : int 82
##   .. ..$ : int 83
##   .. ..$ : int 84
##   .. ..$ : int 85
##   .. ..$ : int 86
##   .. ..$ : int 87
##   .. ..$ : int 88
##   .. ..$ : int 89
##   .. ..$ : int 90
##   .. ..$ : int 91
##   .. ..$ : int 92
##   .. ..$ : int 93
##   .. ..$ : int 94
##   .. ..$ : int 95
##   .. ..$ : int 96
##   .. ..$ : int 97
##   .. ..$ : int 98
##   .. ..$ : int 99
##   .. .. [list output truncated]
##   ..- attr(*, ".drop")= logi TRUE
```

```r
# Normalizando as variaveis numericas
# Funcao para normalizar variáveis numericas
scale.features <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- scale(df[[variable]], center=T, scale=T)
  }
  return(df)
}

# Definindo as variaveis que serao normalizadas
numeric.vars <- c('Demanda_uni_equil','Venda','Ret_Unid','Ret_Venda','Avg_Venda','Taxa_Ret')
dfSelected <- scale.features(dfSelected, numeric.vars)
str(dfSelected)
```

```
## Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame':  48117 obs. of  12 variables:
##  $ Semana           : Factor w/ 7 levels "3","4","5","6",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Agencia_ID       : int  1110 1110 1110 1111 1111 1111 1111 1111 1111 1111 ...
##  $ Canal_ID         : Factor w/ 8 levels "1","2","4","5",..: 6 6 8 1 1 1 1 1 1 1 ...
##  $ Ruta_SAK         : int  3305 3316 3501 1003 1003 1003 1013 1013 1019 1020 ...
##  $ Cliente_ID       : int  2340436 2025225 4552320 49753 49920 1872816 1174627 1446502 1120431 67808
##  $ Producto_ID      : int  46772 325 1700 2233 693 1109 35727 2233 1160 1150 ...
##  $ Demanda_uni_equil: num [1:48117, 1] 0.20294 -0.18671 -0.966 1.5667 0.00811 ...
##   ..- attr(*, "scaled:center")= num 4.96
##   ..- attr(*, "scaled:scale")= num 5.13
##  $ Venda            : num [1:48117, 1] 0.1315 -0.2308 -0.3105 3.6941 0.0359 ...
##   ..- attr(*, "scaled:center")= num 45.9
##   ..- attr(*, "scaled:scale")= num 57.7
##  $ Ret_Unid         : num [1:48117, 1] -0.0742 -0.0742 1.6541 -0.0742 -0.0742 ...
##   ..- attr(*, "scaled:center")= num 0.0858
##   ..- attr(*, "scaled:scale")= num 1.16
##  $ Ret_Venda        : num [1:48117, 1] -0.0927 -0.0927 2.9777 -0.0927 -0.0927 ...
##   ..- attr(*, "scaled:center")= num 0.845
```

```
##   ..- attr(*, "scaled:scale")= num 9.12
##  $ Avg_Venda        : num [1:48117, 1] -0.221 -0.356 0.669 1.711 -0.102 ...
##   ..- attr(*, "scaled:center")= num 10.2
##   ..- attr(*, "scaled:scale")= num 5.7
##  $ Taxa_Ret         : num [1:48117, 1] -0.161 -0.161 6.443 -0.161 -0.161 ...
##   ..- attr(*, "scaled:center")= num 0.0122
##   ..- attr(*, "scaled:scale")= num 0.0757
##  - attr(*, "groups")=Classes 'tbl_df', 'tbl' and 'data.frame':   48117 obs. of  7 variables:
##   ..$ Semana     : int  3 3 3 3 3 3 3 3 3 3 ...
##   ..$ Agencia_ID : int  1110 1110 1110 1111 1111 1111 1111 1111 1111 1111 ...
##   ..$ Canal_ID   : int  7 7 11 1 1 1 1 1 1 1 ...
##   ..$ Ruta_SAK   : int  3305 3316 3501 1003 1003 1003 1013 1013 1019 1020 ...
##   ..$ Cliente_ID : int  2340436 2025225 4552320 49753 49920 1872816 1174627 1446502 1120431 67808 ..
##   ..$ Producto_ID: int  46772 325 1700 2233 693 1109 35727 2233 1160 1150 ...
##   ..$ .rows      :List of 48117
##   .. ..$ : int 1
##   .. ..$ : int 2
##   .. ..$ : int 3
##   .. ..$ : int 4
##   .. ..$ : int 5
##   .. ..$ : int 6
##   .. ..$ : int 7
##   .. ..$ : int 8
##   .. ..$ : int 9
##   .. ..$ : int 10
##   .. ..$ : int 11
##   .. ..$ : int 12
##   .. ..$ : int 13
##   .. ..$ : int 14
##   .. ..$ : int 15
##   .. ..$ : int 16
##   .. ..$ : int 17
##   .. ..$ : int 18
##   .. ..$ : int 19
##   .. ..$ : int 20
##   .. ..$ : int 21
##   .. ..$ : int 22
##   .. ..$ : int 23
##   .. ..$ : int 24
##   .. ..$ : int 25
##   .. ..$ : int 26
##   .. ..$ : int 27
##   .. ..$ : int 28
##   .. ..$ : int 29
##   .. ..$ : int 30
##   .. ..$ : int 31
##   .. ..$ : int 32
##   .. ..$ : int 33
##   .. ..$ : int 34
##   .. ..$ : int 35
##   .. ..$ : int 36
##   .. ..$ : int 37
##   .. ..$ : int 38
##   .. ..$ : int 39
```

```
##   .. ..$ : int 40
##   .. ..$ : int 41
##   .. ..$ : int 42
##   .. ..$ : int 43
##   .. ..$ : int 44
##   .. ..$ : int 45
##   .. ..$ : int 46
##   .. ..$ : int 47
##   .. ..$ : int 48
##   .. ..$ : int 49
##   .. ..$ : int 50
##   .. ..$ : int 51
##   .. ..$ : int 52
##   .. ..$ : int 53
##   .. ..$ : int 54
##   .. ..$ : int 55
##   .. ..$ : int 56
##   .. ..$ : int 57
##   .. ..$ : int 58
##   .. ..$ : int 59
##   .. ..$ : int 60
##   .. ..$ : int 61
##   .. ..$ : int 62
##   .. ..$ : int 63
##   .. ..$ : int 64
##   .. ..$ : int 65
##   .. ..$ : int 66
##   .. ..$ : int 67
##   .. ..$ : int 68
##   .. ..$ : int 69
##   .. ..$ : int 70
##   .. ..$ : int 71
##   .. ..$ : int 72
##   .. ..$ : int 73
##   .. ..$ : int 74
##   .. ..$ : int 75
##   .. ..$ : int 76
##   .. ..$ : int 77
##   .. ..$ : int 78
##   .. ..$ : int 79
##   .. ..$ : int 80
##   .. ..$ : int 81
##   .. ..$ : int 82
##   .. ..$ : int 83
##   .. ..$ : int 84
##   .. ..$ : int 85
##   .. ..$ : int 86
##   .. ..$ : int 87
##   .. ..$ : int 88
##   .. ..$ : int 89
##   .. ..$ : int 90
##   .. ..$ : int 91
##   .. ..$ : int 92
##   .. ..$ : int 93
```

```
##   .. ..$ : int 94
##   .. ..$ : int 95
##   .. ..$ : int 96
##   .. ..$ : int 97
##   .. ..$ : int 98
##   .. ..$ : int 99
##   .. .. [list output truncated]
##   ..- attr(*, ".drop")= logi TRUE
```

```
# Removendo funcoes e variaveis da memoria
rm('to.factors')
rm('scale.features')
rm('categorical.vars')
rm('numeric.vars')
```

```
### Criando e Avaliando o modelo ###

# Carregando os Pacotes
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```
# Gerando dados de treino e de teste
splits <- createDataPartition(dfSelected$Demanda_uni_equil, p=0.7, list=FALSE)

# Separando os dados
dados_treino <- dfSelected[ splits,]
dados_teste <- dfSelected[-splits,]
```

```
# Construindo um modelo Linear Model
controlLM <- trainControl(method="cv", number=5)
modeloLM <- train(Demanda_uni_equil ~ ., data = dados_treino, method = "lm", metric="RMSE", trControl=c

# Resumo do Modelo Linear Model
print(modeloLM)
```

```
## Linear Regression
##
## 33684 samples
##    11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 26946, 26948, 26947, 26947, 26948
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5335065  0.7215858  0.3233651
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Construindo um modelo Random Forest
# controlRF <- trainControl(method="cv", number=4)
# modeloRF <- train(Demanda_uni_equil ~ ., data = dados_treino, method = "rf", metric="RMSE", trControl
#
# # Resumo do Modelo Random Forest
# print(modeloRF)
```

```r
# Construindo um modelo Generalized Boosted Regression Modeling (GBM)
controlGBM <- trainControl(method="cv", number=4)
modeloGBM <- train(Demanda_uni_equil ~ ., data=dados_treino, method="gbm", verbose=FALSE, metric="RMSE"

# Resumo do Modelo GBM
print(modeloGBM)
```

```
## Stochastic Gradient Boosting
##
## 33684 samples
##    11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 25263, 25263, 25263, 25263
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE       Rsquared   MAE
##   1                   50      0.4781084  0.8135838  0.26373335
##   1                  100      0.3980477  0.8545766  0.21899272
##   1                  150      0.3703684  0.8688952  0.21315169
##   2                   50      0.2797400  0.9418170  0.13775859
##   2                  100      0.1925619  0.9672273  0.09391987
##   2                  150      0.1596792  0.9760971  0.08272904
##   3                   50      0.1950551  0.9700377  0.09990949
##   3                  100      0.1225166  0.9858702  0.06159847
##   3                  150      0.1020913  0.9898506  0.05351892
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
### Otimizando o modelo Generalized Boosted Regression Modeling (GBM) ###

# Carregando os Pacotes
library(gbm)
```

```
## Loaded gbm 2.1.5
```

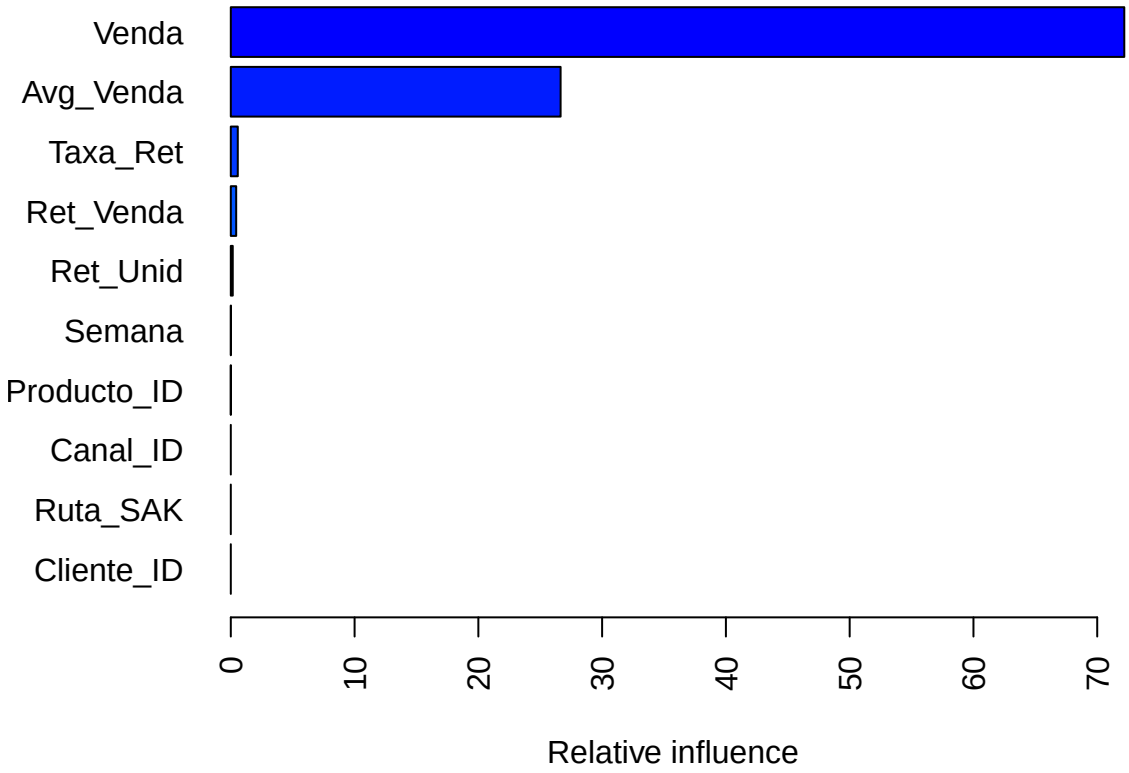
```r
# Modificando os Hyperparametros
hyperParam <- expand.grid(
  shrinkage = c(.01, .05, .1),
  interaction.depth = c(3, 5, 7),
  n.minobsinnode = c(5, 7, 10),
  bag.fraction = c(.65, .8, 1),
```

```
  optimal_trees = 0,
  min_RMSE = 0
)

for(i in 1:nrow(hyperParam)) {
  # Treinando o modelo
  modeloGBM_v2 <- gbm(
    formula = Demanda_uni_equil ~ .,
    distribution = "gaussian",
    data = dados_treino,
    n.trees = 200,
    interaction.depth = hyperParam$interaction.depth[i],
    shrinkage = hyperParam$shrinkage[i],
    n.minobsinnode = hyperParam$n.minobsinnode[i],
    bag.fraction = hyperParam$bag.fraction[i],
    train.fraction = .75,
    n.cores = NULL,
    verbose = FALSE
  )

  hyperParam$optimal_trees[i] <- which.min(modeloGBM_v2$valid.error)
  hyperParam$min_RMSE[i] <- sqrt(min(modeloGBM_v2$valid.error))
}

hyperParam %>%
  arrange(min_RMSE) %>%
  head(10)
```

```
##    shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees
## 1        0.1                 7             10         1.00           199
## 2        0.1                 7              5         1.00           200
## 3        0.1                 7              7         1.00           200
## 4        0.1                 7              5         0.65           200
## 5        0.1                 7              5         0.80           200
## 6        0.1                 7              7         0.65           200
## 7        0.1                 7              7         0.80           199
## 8        0.1                 7             10         0.80           200
## 9        0.1                 7             10         0.65           200
## 10       0.1                 5              7         0.80           200
##      min_RMSE
## 1  0.05040675
## 2  0.05207429
## 3  0.05273225
## 4  0.05312356
## 5  0.05318748
## 6  0.05383118
## 7  0.05421881
## 8  0.05504425
## 9  0.05630654
## 10 0.06405233
```

```
# Treinando o modelo final com os melhores parametros
modeloGBM_final <- gbm(
  formula = Demanda_uni_equil ~ .,
```

```
  distribution = "gaussian",
  data = dados_treino,
  n.trees = 180,
  interaction.depth = 7,
  shrinkage = 0.1,
  n.minobsinnode = 5,
  bag.fraction = 1,
  train.fraction = 1,
  n.cores = NULL,
  verbose = FALSE
)

# Visualizando as variaveis de influencia
par(mar = c(5, 8, 1, 1))
summary(
  modeloGBM_final,
  cBars = 10,
  method = relative.influence,
  las = 2
)
```

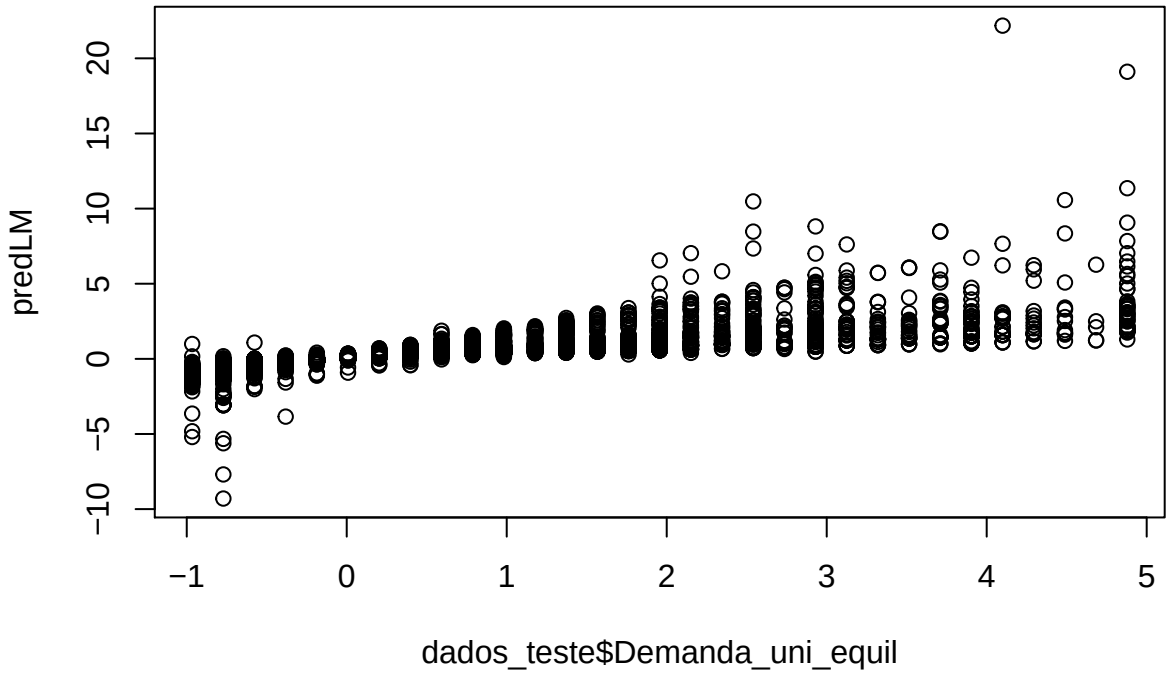



```
##                     var      rel.inf
## Venda             Venda 7.218630e+01
## Avg_Venda     Avg_Venda 2.664443e+01
## Taxa_Ret       Taxa_Ret 5.637774e-01
## Ret_Venda     Ret_Venda 4.360133e-01
## Ret_Unid       Ret_Unid 1.581468e-01
## Semana           Semana 8.262950e-03
## Producto_ID Producto_ID 1.361254e-03
```

```
## Canal_ID       Canal_ID 1.072541e-03
## Ruta_SAK       Ruta_SAK 5.245131e-04
## Cliente_ID   Cliente_ID 1.085038e-04
## Agencia_ID   Agencia_ID 0.000000e+00
```

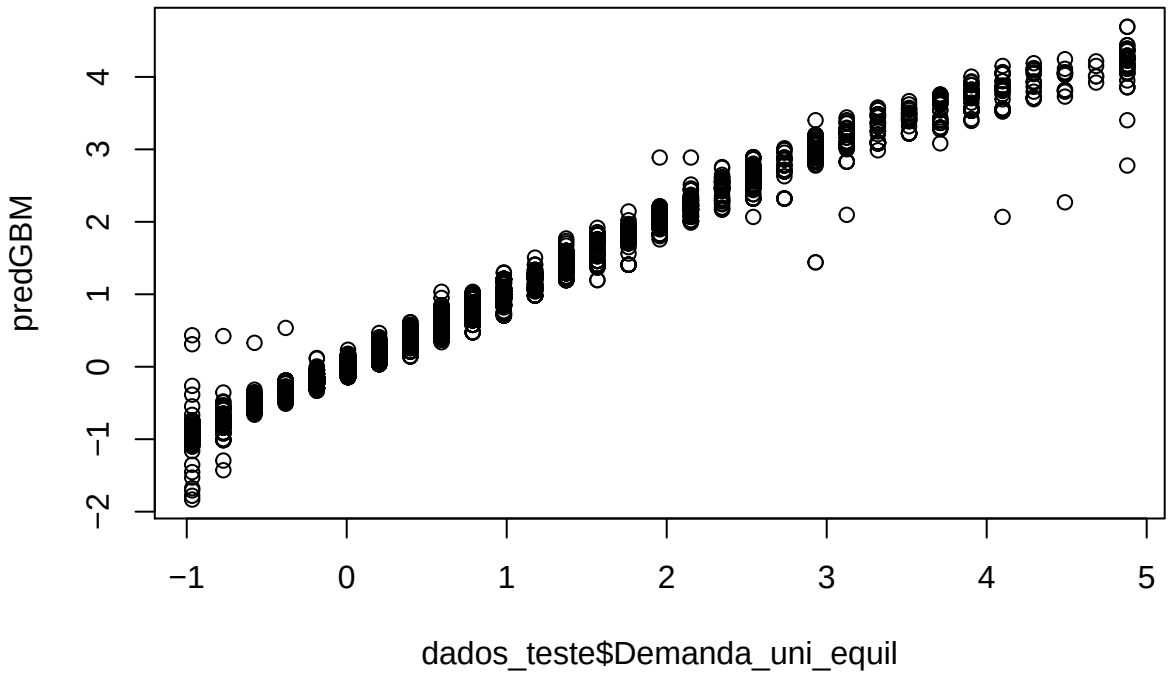
```
### Analisando o resultado atraves de gráficos ###
```

```
# Modelo Linear Model
predLM <- predict(modeloLM, newdata = dados_teste)
plot(dados_teste$Demanda_uni_equil, predLM)
```




```
# Modelo Random Forest
# predRF <- predict(modeloRF, newdata = dados_teste)
# plot(dados_teste$Demanda_uni_equil, predRF)
```

```
# Modelo Generalized Boosted Regression Modeling (GBM)
predGBM <- predict(modeloGBM, newdata = dados_teste)
plot(dados_teste$Demanda_uni_equil, predGBM)
```




```
# Modelo GBM Otimizado
predGBM_final <- predict(modeloGBM_final, n.trees = modeloGBM_final$n.trees, dados_teste)
plot(dados_teste$Demanda_uni_equil, predGBM_final)
```