# Demanda de Estoque com Base em Vendas

Lucas Kitano

10/07/2021

## Projeto com Feedback 2 do curso Big Data Analytics com R e Microsoft Azure Machine Learning

O objetivo deste projeto é criar um modelo que possa prever a demanda de estoque com base nos dados históricos de vendas. Este projeto foi uma competição do Kaggle ("https://www.kaggle.com/c/grupo-bimbo-inventory-demand") proposto pela empresa Grupo Bimbo ("https://www.grupobimbo.com/") que é uma das maiores empresas do mundo em panificação. Atendendo mais de 1 milhão de lojas no México.

Atualmente, os cálculos diários de estoque são realizados por funcionários de vendas de entregas diretas, que devem, sozinhos, prever a necessidade de estoque dos produtos e demanda com base em suas experiências pessoais em cada loja. Como alguns pães têm uma vida útil de uma semana, a margem aceitável para o erro é pequena. Assim sendo vamos utilizar os dados históricos fornecidos para tentar criar um modelo que possa prever a reposição destes produtos nas prateleiras dos clientes do grupo.

## Pacotes necessários

Aqui estão os pacotes necessários para a execução/reprodução deste script:

```
# Carrega Pacotes Necessários
# Use o comando "install.packages("nome_do_pacote") para instalar algum dos pacotes abaixo,
# caso não o possua.

#library(data.table)
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
```

```
library("stringr")
library("ggplot2")
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

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

```
##
##     slice
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
options(warn=-1)
```

# Carga dos dados

O dataset de treinamento possui 74.180.464, para fluídez do processo, construímos uma amostra de treimamento aleatória com 1.000.000 de registros. Entretanto caso queira trabalhar com a amostra de treinamento completa, verifique o script "PrevendoDemandaEstoque.R".

```
# Lê o arquivo da amostra de treinamento:
train_sample <- read.csv("datasets/train_sample.csv")

# Verifica o formato dos dados
str(train_sample)
```

```
## 'data.frame':    1000000 obs. of  11 variables:
##  $ Semana           : int  6 9 6 7 6 7 9 7 8 3 ...
##  $ Agencia_ID       : int  1315 4051 2217 2087 1347 2229 1312 1112 3213 3214 ...
##  $ Canal_ID         : int  1 1 1 2 1 1 1 1 1 1 ...
##  $ Ruta_SAK         : int  2053 1222 2826 7221 1181 2136 2860 1414 1030 2151 ...
##  $ Cliente_ID       : int  62043 4348343 316511 1586190 139794 4402092 4659932 339119 20332 2115031
##  $ Producto_ID      : int  43147 36711 5355 45567 8940 31466 37361 1220 1150 30532 ...
##  $ Venta_uni_hoy    : int  10 3 2 48 7 3 2 3 7 4 ...
##  $ Venta_hoy        : num  45.4 22.5 10.4 1418.9 56 ...
##  $ 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  10 3 2 48 7 3 2 3 7 4 ...
```

```
head(train_sample)
```

```
##   Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID Venta_uni_hoy
## 1      6       1315        1     2053      62043       43147            10
## 2      9       4051        1     1222    4348343       36711             3
## 3      6       2217        1     2826     316511        5355             2
## 4      7       2087        2     7221    1586190       45567            48
## 5      6       1347        1     1181     139794        8940             7
## 6      7       2229        1     2136    4402092       31466             3
##   Venta_hoy Dev_uni_proxima Dev_proxima Demanda_uni_equil
## 1     45.40               0           0               10
## 2     22.50               0           0                3
## 3     10.38               0           0                2
## 4   1418.88               0           0               48
## 5     56.00               0           0                7
## 6     29.97               0           0                3
```

## Colunas do dataset:

Entenda-se "nesta semana" a semana de referência da coluna "Semana".

Semana — Número que identifica a semana (de Quinta para Quarta);
Agencia_ID — ID do depósito de vendas;

Canal_ID — ID do canal de vendas;
Ruta_SAK — ID da rota (Várias rotas = Depósito de vendas);
Cliente_ID — ID do Cliente;
Producto_ID — Product ID;
Venta_uni_hoy — Vendas unitárias (nesta semana);
Venta_hoy — Vendas nesta semana (unidade: pesos);
Dev_uni_proxima — Retorno por unidade nesta semana;
Dev_proxima — Retorno na próxima semana (unidade: pesos);
Demanda_uni_equil — Demanda Ajustada (Variável Target);

Apesar do dataset de treino conter as variáveis numericas relacionadas as vendas estas não poderão ser utilizadas no modelo, pois na competição do Kaggle, apenas as variáveis de ID estão no dataset de teste, de fato a previsão deve ocorrer rm cima das variáveis ID, uma vez que a demanda ajustada (variável alvo) é dada pelas colunas "Venta_uni_hoy" - "Dev_uni_proxima".

# Análise Exploratória e Engenharia de Atríbutos

Conforme análise preliminar dos arquivos disponíveis, o dataset de treinamento contém o registro das semanas 3 a 9, e o dataset de teste o registro das semanas 10 e 11 - que devem ter a demanda ajustada preditas pelo algoritimo.

Incialmente, vamos a obter a quantidade de valores únicos para cada variável ID considerando a nossa amostra de treinamento:

```
length(unique(train_sample$Agencia_ID))
```

```
## [1] 551
```

```
length(unique(train_sample$Canal_ID))
```

```
## [1] 9
```

```
length(unique(train_sample$Ruta_SAK))
```

```
## [1] 2259
```

```
length(unique(train_sample$Cliente_ID))
```

```
## [1] 458457
```

```
length(unique(train_sample$Producto_ID))
```

```
## [1] 1302
```

Baseado em nossa amostra de treinamento que possui 1.000.000 registros, a variável "Cliente_ID", possui 458.457 registros únicos, enquanto a variável "Producto_ID" possui 1.302 registros, podemos começar a análise por estas variáveis.

## Análise de Clientes e Produtos

Vamos carregar o dataset "Cliente_tabla" que possui a descrição dos clientes identificados pela chave "Cliente_ID":

```
cliente <- read.csv("datasets/cliente_tabla.csv", stringsAsFactors = FALSE)
str(cliente)
```

```
## 'data.frame':    935362 obs. of  2 variables:
##  $ Cliente_ID   : int  0 1 2 3 4 4 5 6 7 8 ...
##  $ NombreCliente: chr  "SIN NOMBRE" "OXXO XINANTECATL" "SIN NOMBRE" "EL MORENO" ...
```

```
nrow(cliente)
```

```
## [1] 935362
```

```
length(unique(cliente$Cliente_ID))
```

```
## [1] 930500
```

```
length(unique(cliente$NombreCliente))
```

```
## [1] 311155
```

O dataset possui 935.362 regsitros, no entanto temos 930.500 valores únicos de ID, isto indica que podem haver valores duplicados de ID no dataset, vamos removê-los:

```
cliente <- cliente %>% distinct(Cliente_ID, .keep_all= TRUE)

nrow(cliente)
```

```
## [1] 930500
```

```
length(unique(cliente$Cliente_ID))
```

```
## [1] 930500
```

```
length(unique(cliente$NombreCliente))
```

```
## [1] 307009
```

Agora sim, temos a mesma quantidade de valores únicos de ID, e de números de linhas na tabela, entretanto, para o nome do cliente, temos 307.009 registros, isto pode indicar que o mesmo nome de cliente possa estar sendo identificado por 2 IDs diferentes:

```
head(cliente)
```

```
##   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          5                             LA VAQUITA
```

Por exemplo, o nome de cliente sem nome "SIN NOMBRE" está com os ID's 0 e 2, outros casos semelhantes devem estar ocorrendo neste dataset, vamos obter uma tabela de frequências com os nomes de clientes:

```
cliente %>%
  group_by(NombreCliente) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n)) %>%
  head(20)
```

```
## # A tibble: 20 x 3
##    NombreCliente        n    freq
##    <chr>            <int>   <dbl>
##  1 NO IDENTIFICADO 281670 30.3
##  2 LUPITA            4863  0.523
##  3 MARY              3016  0.324
##  4 LA PASADITA       2426  0.261
##  5 LA VENTANITA      2267  0.244
```

```
##  6 LA GUADALUPANA    1299  0.140
##  7 ROSY              1245  0.134
##  8 ALEX              1242  0.133
##  9 GABY              1238  0.133
## 10 LA ESCONDIDA      1216  0.131
## 11 PATY              1145  0.123
## 12 LA ESPERANZA      1139  0.122
## 13 HERNANDEZ         1129  0.121
## 14 LA CHIQUITA       1116  0.120
## 15 DANY              1082  0.116
## 16 GARCIA            1062  0.114
## 17 JUQUILITA         1022  0.110
## 18 MARTINEZ           987  0.106
## 19 LA TIENDITA        914  0.0982
## 20 3 HERMANOS         913  0.0981
```

O nome de cliente "NO INDENTIFICADO", ocorre em mais de 30% no dataset, outros nomes comuns são nomes coloquiais como Lupita, Mary, Rosy, Alex... Já nomes de clientes contém artigos definidos em espanhol como "EL" e "LA". A grande questão é que estes nomes podem ser agrupados, por exemplo, "TIENDITA" em português significa "pequena loja", vamos pesquisar quantos registros possui este termo:

```
cliente %>%
  filter(grepl('TIENDITA', NombreCliente)) %>%
  group_by(NombreCliente) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 441 x 3
##    NombreCliente                n   freq
##    <chr>                    <int>  <dbl>
##  1 LA TIENDITA                914 36.5
##  2 MI TIENDITA                881 35.2
##  3 TIENDITA                    43  1.72
##  4 ABARROTES LA TIENDITA       39  1.56
##  5 ABARROTES MI TIENDITA       37  1.48
##  6 MINI TIENDITA               28  1.12
##  7 MISCELANEA LA TIENDITA      24  0.958
##  8 LA TIENDITA DE LA ESQUINA   19  0.758
##  9 MISCELANEA MI TIENDITA      18  0.719
## 10 LA NUEVA TIENDITA            7  0.279
## # ... with 431 more rows
```

O termo TIENDITA possui 441 ocorrências, destas 36,5% são referidas como "LA TIENDITA" e 35,2% como "MI TIENDITA". Há outros termos no dataset que referem se a estabelecimentos em geral, por exemplo, "oxxo" é uma rede de lojas de conveniência no México, "abarrotes" significa Mercearia, "super" é uma palavra que pode se referir a um supermercado, há até palavras mais comuns como farmácia:

```
cliente %>%
  filter(grepl('OXXO', NombreCliente)) %>%
  group_by(NombreCliente) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 6,353 x 3
```

```
##    NombreCliente          n  freq
##    <chr>              <int> <dbl>
##  1 OXXO MORELOS          26 0.290
##  2 OXXO TECNOLOGICO      22 0.245
##  3 OXXO HIDALGO          20 0.223
##  4 OXXO REFORMA          20 0.223
##  5 OXXO JUAREZ           19 0.212
##  6 OXXO UNIVERSIDAD      19 0.212
##  7 OXXO INDEPENDENCIA    18 0.201
##  8 OXXO ZARAGOZA         18 0.201
##  9 OXXO AEROPUERTO       16 0.178
## 10 OXXO ALAMEDA          16 0.178
## # ... with 6,343 more rows
```

```
cliente %>%
  filter(grepl('ABARROTES', NombreCliente)) %>%
  group_by(NombreCliente) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 12,325 x 3
##    NombreCliente           n  freq
##    <chr>               <int> <dbl>
##  1 ABARROTES LUPITA      464 1.41
##  2 ABARROTES MARY        337 1.03
##  3 ABARROTES GARCIA      179 0.545
##  4 ABARROTES MARTINEZ    168 0.511
##  5 ABARROTES ALEX        162 0.493
##  6 ABARROTES HERNANDEZ   149 0.453
##  7 ABARROTES RODRIGUEZ   140 0.426
##  8 ABARROTES DANY        135 0.411
##  9 ABARROTES ROSY        124 0.377
## 10 ABARROTES GONZALEZ    122 0.371
## # ... with 12,315 more rows
```

```
cliente %>%
  filter(grepl('SUPER', NombreCliente)) %>%
  group_by(NombreCliente) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 11,608 x 3
##    NombreCliente         n  freq
##    <chr>             <int> <dbl>
##  1 MINI SUPER          143 0.857
##  2 EL SUPERCITO         64 0.383
##  3 SUPERCITO            55 0.330
##  4 MINI SUPER LUPITA    47 0.282
##  5 SUPER SIX            47 0.282
##  6 SUPER UNO            41 0.246
##  7 MI SUPER             40 0.240
##  8 SUPER 7              40 0.240
##  9 MINISUPER            38 0.228
```

```
## 10 SUPER CENTRO         37 0.222
## # ... with 11,598 more rows
```

Vamos criar agrupamentos com estes termos, estes agrupamentos foram baseados no competidor Kaggle ("https://www.kaggle.com/abbysobh/classifying-client-type-using-client-names") com algumas modificações pontuais.

```
mercado <- c("ABARROTES", "TIENDITA", "COMERCIAL", "BODEGA", "DEPOSITO", "MERCADO", "CAMBIO", "MARKET",
             "MARKET", "MART", "MINI", "PLAZA", "MISC", "MINI", "PLAZA", "MISC", "ELEVEN", "EXP",
             "SNACK", "PAPELERIA", "CARNICERIA", "LOCAL", "COMODIN", "PROVIDENCIA")
escola <- c("ESCOLA", "COLEG", "UNIV", "ESCU", "INSTI", "PREPAR", "INSTITUTO", "C E S U", "CESU")
restaurante <- c("CASA", "CAFE", "CREMERIA", "DULCERIA", "REST", "BURGER", "TACO", "TORTA", "TAQUER", "I
saude <- c("FARMA", "HOSPITAL", "CLINI")
fresco <-c("VERDU", "FRUT")
hotel <- c("HOTEL", "MOTEL")
super_mercados <- c("WALL MART", "SAMS CLUB", "SUPER")
pequenos <- c('LA','EL','DE','LOS','DEL','Y', 'SAN', 'SANTA',
              'AG','LAS','MI','MA', 'II')

governo <- c('POLICIA','CONASUPO')

todos <- c(mercado, escola, restaurante, saude, fresco, hotel, pequenos, governo, "POSTO", "OXXO",
           "REMISION", "BIMBO")

cliente_new <- cliente %>%
  mutate(local_grup = NombreCliente) %>%

  mutate(local_grup = replace(local_grup, grepl(paste(escola, collapse = "|"), local_grup), "EDUCACAO"))
  mutate(local_grup = replace(local_grup, grepl("POSTO", local_grup, ignore.case = FALSE), "POSTO")) %>%
  mutate(local_grup = replace(local_grup, grepl(paste(saude, collapse = "|"), local_grup), "HOSPITAL"))
  mutate(local_grup = replace(local_grup, grepl(paste(restaurante, collapse = "|"), local_grup), "RESTAU
  mutate(local_grup = replace(local_grup, grepl(paste(mercado, collapse = "|"), local_grup), "MERCADOS/C
  mutate(local_grup = replace(local_grup, grepl(paste(super_mercados, collapse = "|"), local_grup), "SU
  mutate(local_grup = replace(local_grup, grepl(paste(fresco, collapse = "|"), local_grup), "MERCADO FR
  mutate(local_grup = replace(local_grup, grepl(paste(hotel, collapse = "|"), local_grup), "SERVICOS"))
  mutate(local_grup = replace(local_grup, grepl("OXXO", local_grup, ignore.case = FALSE), "LOJA OXXO"))
  mutate(local_grup = replace(local_grup, grepl("REMISION", local_grup, ignore.case = FALSE), "CORREIOS
  mutate(local_grup = replace(local_grup, grepl(paste(governo, collapse = "|"), local_grup), "GOVERNO"))
  mutate(local_grup = replace(local_grup, grepl("BIMBO", local_grup, ignore.case = FALSE), "LOJA BIMBO"

  mutate(local_grup = replace(local_grup, grepl(paste(pequenos, collapse = "|"), local_grup), "FRANQUIA
  mutate(local_grup = ifelse(str_detect(NombreCliente, paste(todos, collapse = "|"), negate = TRUE), "SI
  group_by(NombreCliente, local_grup)

length(unique(cliente_new$local_grup))
```

```
## [1] 14
```

```
cliente_new %>%
  group_by(local_grup) %>%
  summarise(n = n()) %>%
  mutate(freq = 100*n / sum(n)) %>%
  arrange(desc(n)) %>%
  head(15)
```

```
## # A tibble: 14 x 3
```

```
##    local_grup                     n     freq
##    <chr>                      <int>    <dbl>
##  1 FRANQUIA PEQUENA          593224 63.8
##  2 SEM IDENTIFICACAO         181348 19.5
##  3 MERCADOS/COMERCIOS GERAIS  87532  9.41
##  4 RESTAURANTE                41335  4.44
##  5 LOJA OXXO                   7945  0.854
##  6 EDUCACAO                    5820  0.625
##  7 HOSPITAL                    5730  0.616
##  8 SUPER MERCADO               4431  0.476
##  9 SERVICOS                    1074  0.115
## 10 MERCADO FRESCO              1032  0.111
## 11 GOVERNO                      958  0.103
## 12 LOJA BIMBO                    33  0.00355
## 13 POSTO                         24  0.00258
## 14 CORREIOS                      14  0.00150
```

Agora temos 14 categorias, a única ressalva fica pela alta concentração da chamada Pequenas Franquias - 64% dos dados, esta categoria junto da Sem Identificação está concentrando a maior parte dos clientes individuais (com um só nome), não é o ideal, mas já está bem melhor do que ter mais de 300 mil registros únicos para o nome do cliente.

Vamos estender a análise agora para a variável "Producto_ID", para a nossa amostra de treinamento ela possui mais de 1000 valores unicos, além disso o produto pode estar diretamente relacionado com a necessidade de demanda, seja por consumo, seja por armazenamento. Alguns produtos possuem um tempo limitado dearmazenamento em estoque, por isto está variável pode ser relevante para o algoritmo.

Vamos carregar inicialmente o dataset fornecido que identifica os produtos:

```
produtos <- read.csv("datasets/producto_tabla.csv", stringsAsFactors = FALSE)
str(produtos)
```

```
## 'data.frame':    2592 obs. of  2 variables:
##  $ Producto_ID   : int  0 9 41 53 72 73 98 99 100 106 ...
##  $ NombreProducto: chr  "NO IDENTIFICADO 0" "Capuccino Moka 750g NES 9" "Bimbollos Ext sAjonjoli 6p 4
```

```
head(produtos)
```

```
##   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
```

A variável "NombreProducto" segue um padrão de nomenclatura (Nome do Produto, Quantidade, Peso, Marca). Quem percebeu esta divisão foi o competidor Kaggle: ("https://www.kaggle.com/vykhand/exploring-products"). Vamos dividir estes campos em 3 novos campos representando (Nome do Produto, Quantidade, Peso):

```
produtos_new <- produtos %>%
  mutate(short_product_name = str_extract(NombreProducto, regex("^\\D*"))) %>%
  mutate(pieces = as.numeric(gsub("p","",str_extract(NombreProducto, regex("(\\d+)p "))))) %>%
  mutate(weights = str_extract(NombreProducto, regex("(\\d+)(Kg|g) "))) %>%
  mutate(weight = ifelse(str_detect(weights, "Kg"), 1000*as.numeric(gsub("Kg", "", weights)), as.numeri

str(produtos_new)
```

```
## 'data.frame':    2592 obs. of  6 variables:
##  $ Producto_ID       : int  0 9 41 53 72 73 98 99 100 106 ...
##  $ NombreProducto    : chr  "NO IDENTIFICADO 0" "Capuccino Moka 750g NES 9" "Bimbollos Ext sAjonjoli
##  $ short_product_name: chr  "NO IDENTIFICADO " "Capuccino Moka " "Bimbollos Ext sAjonjoli " "Burritos
##  $ pieces            : num  NA NA 6 NA 4 NA NA NA NA NA ...
##  $ weights           : chr  NA "750g " "480g " "170g " ...
##  $ weight            : num  NA 750 480 170 45 540 180 567 680 475 ...
```

```
head(produtos_new)
```

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

Agora vamos juntar estas informações obtidas com o dataset principal (chave: Producto_ID e Cliente_ID).

```
train_sample <- train_sample %>%
  left_join(select(produtos_new, short_product_name, pieces, weight, Producto_ID), by = c("Producto_ID"

train_sample <- train_sample %>%
  left_join(select(cliente_new, local_grup, Cliente_ID), by = c("Cliente_ID" = "Cliente_ID"))
```

```
## Adding missing grouping variables: `NombreCliente`
```

```
head(train_sample)
```

```
##   Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID Venta_uni_hoy
## 1      6       1315        1     2053      62043       43147            10
## 2      9       4051        1     1222    4348343       36711             3
## 3      6       2217        1     2826     316511        5355             2
## 4      7       2087        2     7221    1586190       45567            48
## 5      6       1347        1     1181     139794        8940             7
## 6      7       2229        1     2136    4402092       31466             3
##   Venta_hoy Dev_uni_proxima Dev_proxima Demanda_uni_equil
## 1     45.40               0           0                10
## 2     22.50               0           0                 3
## 3     10.38               0           0                 2
## 4   1418.88               0           0                48
## 5     56.00               0           0                 7
## 6     29.97               0           0                 3
##          short_product_name pieces weight             NombreCliente
## 1                   Dalmata       1     55 MA DE LOS ANGELES ZAMORA G
## 2                 Madalenas       3     93            NO IDENTIFICADO
## 3                Napolitano       1     70                   FRANTERA
## 4 Pan Whole grain and flax      NA    680           SUPERAMA MILENIO
## 5      Tortilla Hna RC SLP      10    260                    AVENIDA
```

```
## 6            Principe Tubo      14    152            NO IDENTIFICADO
##          local_grup
## 1  FRANQUIA PEQUENA
## 2  FRANQUIA PEQUENA
## 3 SEM IDENTIFICACAO
## 4     SUPER MERCADO
## 5 SEM IDENTIFICACAO
## 6  FRANQUIA PEQUENA
```

Com as informações mais consolidadas dos produtos e estabelecimentos, podemos fazer algums análises, como por exemplo quais produtos possuem maior demanda:

```
train_sample %>%
  group_by(short_product_name) %>%
  summarise(sum_Demanda_uni_equil = sum(Demanda_uni_equil)) %>%
  mutate(freq = 100*sum_Demanda_uni_equil/sum(sum_Demanda_uni_equil)) %>%
  arrange(desc(sum_Demanda_uni_equil)) %>%
  head(15)
```

```
## # A tibble: 15 x 3
##    short_product_name       sum_Demanda_uni_equil  freq
##    <chr>                                   <int> <dbl>
##  1 "Nito "                                773974 10.8
##  2 "Rebanada "                            348701  4.85
##  3 "Gansito "                             346199  4.82
##  4 "Pan Blanco "                          294804  4.10
##  5 "Tortillinas "                         249549  3.47
##  6 "Pan Integral "                        223287  3.11
##  7 "Barritas Fresa "                      198697  2.76
##  8 "Mantecadas Vainilla "                 188009  2.62
##  9 "Donas Azucar "                        181135  2.52
## 10 "Donitas Espolvoreadas "               171165  2.38
## 11 "Bolsa Mini Rocko "                    166091  2.31
## 12 "Polvoroncitos Panera "                165330  2.30
## 13 "Madalenas "                           156669  2.18
## 14 "Medias Noches "                       152817  2.13
## 15 "Principe "                            134848  1.88
```

Os produtos com o termo “Nito” são os mais vendidos para este conjunto de dados (10,8% do total), “Nito” é uma marca bastante conhecida no México, principalmente em sorvetes. Vamos analisar um pouco mais desta marca:

```
train_sample %>%
  filter(str_detect(short_product_name,"Nito")) %>%
  select(Producto_ID, short_product_name, pieces, weight, local_grup) %>%
  head(20)
```

```
##    Producto_ID short_product_name pieces weight              local_grup
## 1         2425              Nito       1     62          FRANQUIA PEQUENA
## 2        34054              Nito       1     62                 LOJA OXXO
## 3         1278              Nito       1     62          FRANQUIA PEQUENA
## 4         2425              Nito       1     62          FRANQUIA PEQUENA
## 5        34053              Nito       1     62                 LOJA OXXO
## 6         1278              Nito       1     62          FRANQUIA PEQUENA
## 7         1278              Nito       1     62          FRANQUIA PEQUENA
## 8         2425              Nito       1     62          FRANQUIA PEQUENA
## 9         2425              Nito       1     62         SEM IDENTIFICACAO
```

```
## 10       43206              Nito       1     62 MERCADOS/COMERCIOS GERAIS
## 11       41843        Leche Nito      NA     NA          SEM IDENTIFICACAO
## 12        1278              Nito       1     62           FRANQUIA PEQUENA
## 13       43206              Nito       1     62 MERCADOS/COMERCIOS GERAIS
## 14        2425              Nito       1     62           FRANQUIA PEQUENA
## 15        2425              Nito       1     62                RESTAURANTE
## 16        2425              Nito       1     62           FRANQUIA PEQUENA
## 17        1278              Nito       1     62           FRANQUIA PEQUENA
## 18       43342           Nito ME       4    248           FRANQUIA PEQUENA
## 19        1278              Nito       1     62                RESTAURANTE
## 20       41843        Leche Nito      NA     NA           FRANQUIA PEQUENA
```

Muitos destes produtos possuem as mesmas quantidades e peso, ou seja são os mesmos produtos vendidos em lojas diferentes, além disso a variável "Producto_ID" possui mais de uma representação para estes mesmos registros, de fato, este agrupamento que fizemos permitiu reduzir as informações que estão duplicadas.

Vamos obter a mesma visualização, mas agora para os tipos de cliente:

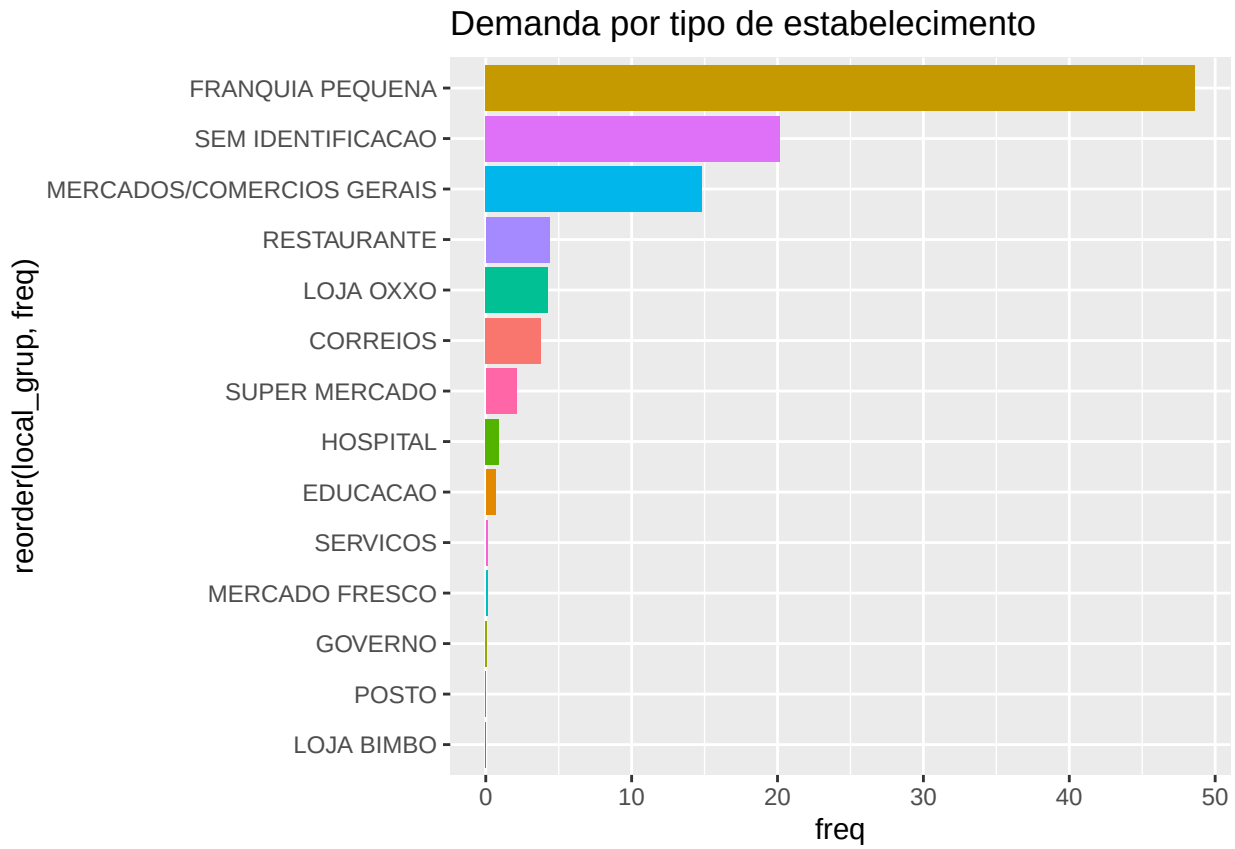
```
local <- train_sample %>%
  group_by(local_grup) %>%
  summarise(sum_Demanda_uni_equil = sum(Demanda_uni_equil)) %>%
  mutate(freq = 100*sum_Demanda_uni_equil/sum(sum_Demanda_uni_equil)) %>%
  arrange(desc(sum_Demanda_uni_equil))

head(local,15)
```

```
## # A tibble: 14 x 3
##    local_grup                sum_Demanda_uni_equil     freq
##    <chr>                                     <int>    <dbl>
##  1 FRANQUIA PEQUENA                        3494571 48.6
##  2 SEM IDENTIFICACAO                       1449692 20.2
##  3 MERCADOS/COMERCIOS GERAIS               1065389 14.8
##  4 RESTAURANTE                              313780  4.36
##  5 LOJA OXXO                                305830  4.25
##  6 CORREIOS                                 271915  3.78
##  7 SUPER MERCADO                            150972  2.10
##  8 HOSPITAL                                  65828  0.916
##  9 EDUCACAO                                  47141  0.656
## 10 SERVICOS                                   8729  0.121
## 11 MERCADO FRESCO                             7565  0.105
## 12 GOVERNO                                    6779  0.0943
## 13 POSTO                                       231  0.00321
## 14 LOJA BIMBO                                  126  0.00175
```

Aqui temos que as pequenas franquias representando quase a metade da demanda ajustada, muito dos dados ficaram sem identificação, mas é preferível te-los agrupado em uma categoria assim, do que te-los em muitos grupos separados.

```
ggplot(local, aes(y = reorder(local_grup, freq), x = freq, fill = local_grup)) + geom_bar(stat = "identi
  theme(legend.position = "none") + ggtitle("Demanda por tipo de estabelecimento")
```

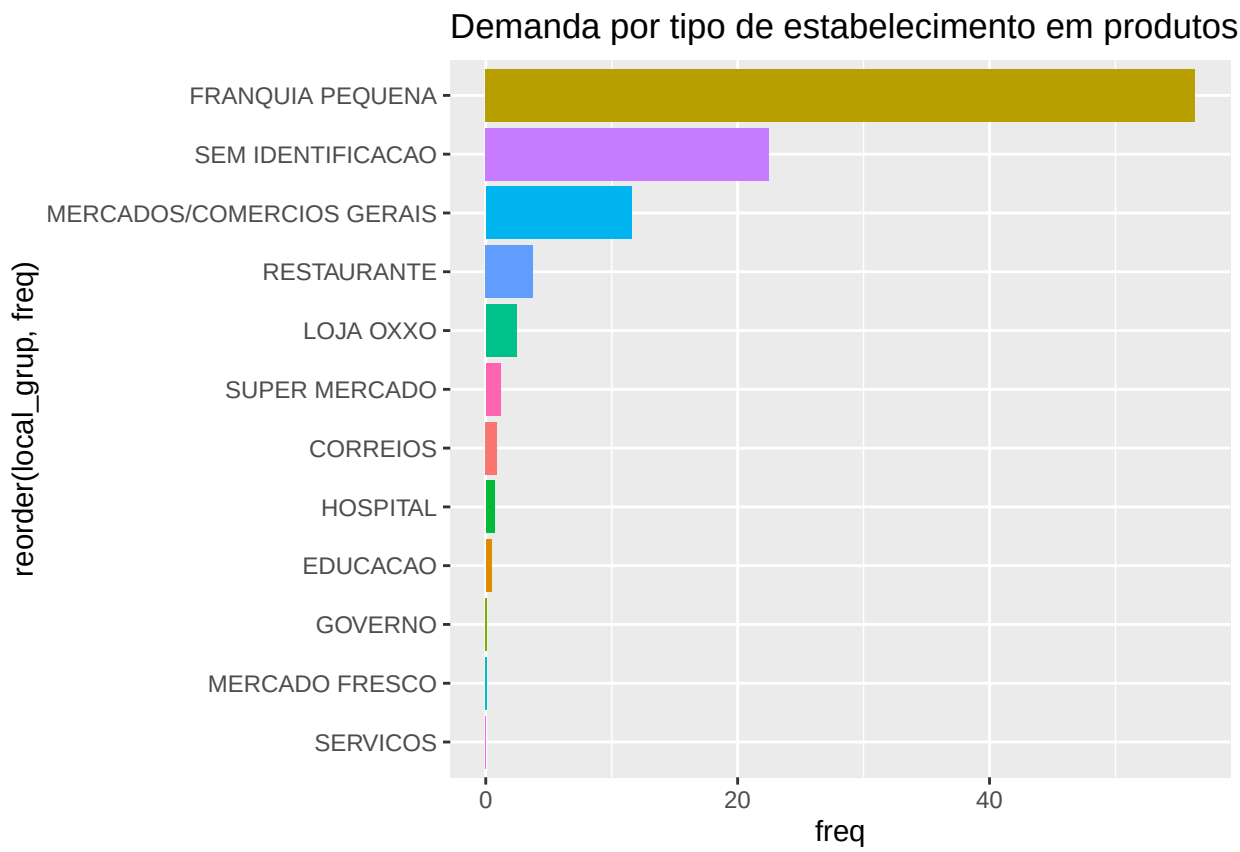



```
# Filtrando os dados considerando apenas o produto Nito:
nito <- train_sample %>%
  filter(str_detect(short_product_name,"Nito")) %>%
  group_by(local_grup) %>%
  summarise(sum_Demanda_uni_equil = sum(Demanda_uni_equil)) %>%
  mutate(freq = 100*sum_Demanda_uni_equil/sum(sum_Demanda_uni_equil)) %>%
  arrange(desc(sum_Demanda_uni_equil))

head(nito, 15)
```

```
## # A tibble: 12 x 3
##    local_grup                sum_Demanda_uni_equil    freq
##    <chr>                                     <int>   <dbl>
##  1 FRANQUIA PEQUENA                         462921 56.3
##  2 SEM IDENTIFICACAO                        184769 22.5
##  3 MERCADOS/COMERCIOS GERAIS                 95269 11.6
##  4 RESTAURANTE                               30867  3.75
##  5 LOJA OXXO                                 20063  2.44
##  6 SUPER MERCADO                              9546  1.16
##  7 CORREIOS                                   7308  0.889
##  8 HOSPITAL                                   5746  0.699
##  9 EDUCACAO                                   3799  0.462
## 10 GOVERNO                                     863  0.105
## 11 MERCADO FRESCO                              857  0.104
## 12 SERVICOS                                    103  0.0125
```

```
ggplot(nito, aes(y = reorder(local_grup, freq), x = freq, fill = local_grup)) + geom_bar(stat = "identi
  theme(legend.position = "none") + ggtitle("Demanda por tipo de estabelecimento em produtos Nito")
```




Não há muita diferença entre os produtos "Nito" e os dados no geral, no que tange ao local.

Antes de prosseguirmos, precisamos ajustar as variáveis peças e peso, pois há produtos que possuem mais peças, consequentemente o peso será maior, uma alternativa a isto é incluir uma variável que represente o peso por peça.

```
train_sample <- train_sample %>%
  mutate(piece_per_weight = round(weight/pieces,1)) %>%
  arrange(Semana)
```

Já que o dataset se trata de dados temporais (identificados pela variável Semana), vamos obter a média da demanda por semana, para verificar que faixa de valor ela ocupa no dataset de amostra.

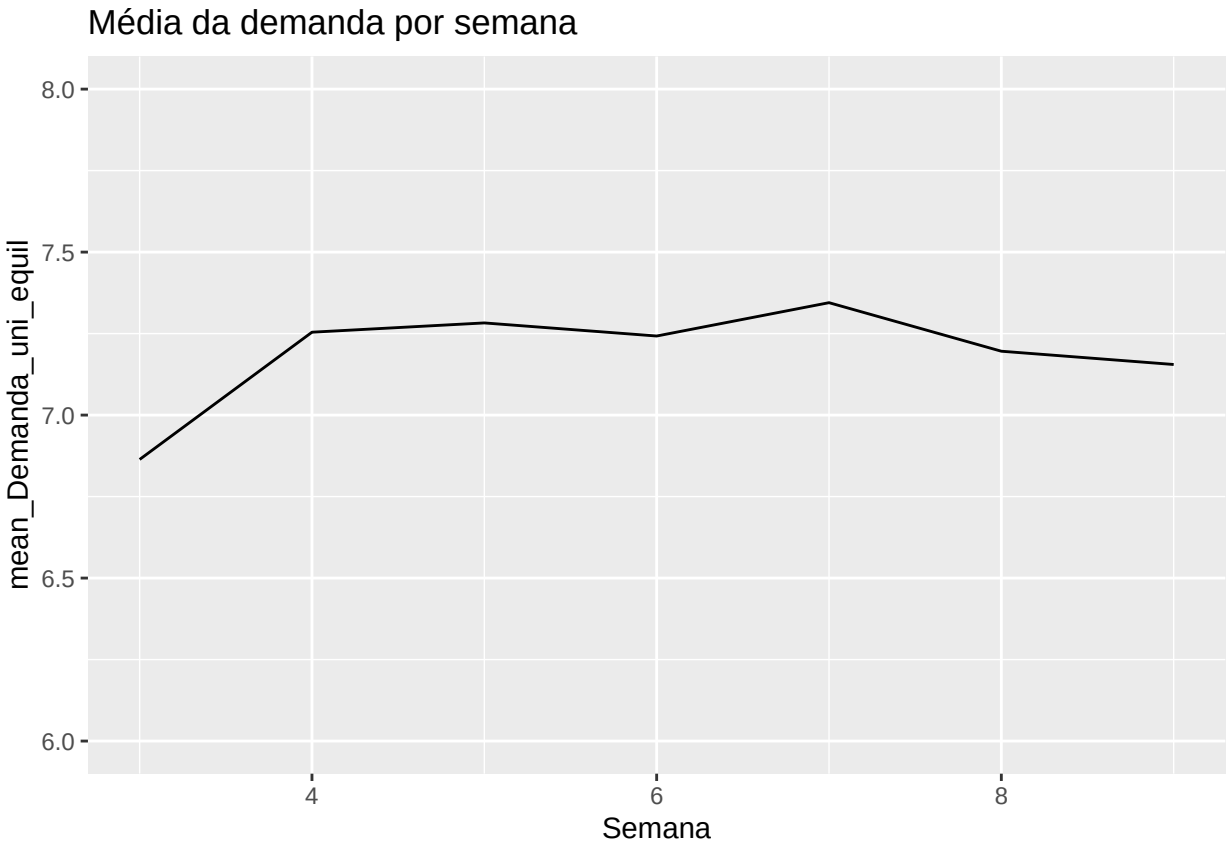
```
semana <- train_sample %>%
  group_by(Semana) %>%
  summarise(mean_Demanda_uni_equil = mean(Demanda_uni_equil))

head(semana)
```

```
## # A tibble: 6 x 2
##   Semana mean_Demanda_uni_equil
##    <int>                  <dbl>
## 1      3                   6.86
## 2      4                   7.25
## 3      5                   7.28
```

```
## 4      6                  7.24
## 5      7                  7.34
## 6      8                  7.20
```

```
ggplot(semana, aes(y = mean_Demanda_uni_equil, x = Semana)) + geom_line(stat = "identity") +
  theme(legend.position = "none") + ggtitle("Média da demanda por semana") +
  ylim(6,8)
```




A média pouco varia nas semanas do dataset de amostra.

### Demanda dos Estados

Uma última tabela fornecida foi a tabela com o local dos produtos, vamos investigá-la:

```
cidades <- read.csv("datasets/town_state.csv", stringsAsFactors = FALSE)
head(cidades)
```

```
##   Agencia_ID                  Town             State
## 1       1110    2008 AG. LAGO FILT     MÃ%XICO, D.F.
## 2       1111 2002 AG. AZCAPOTZALCO     MÃ%XICO, D.F.
## 3       1112   2004 AG. CUAUTITLAN ESTADO DE MÃ%XICO
## 4       1113    2008 AG. LAGO FILT     MÃ%XICO, D.F.
## 5       1114  2029 AG.IZTAPALAPA 2     MÃ%XICO, D.F.
## 6       1116  2011 AG. SAN ANTONIO     MÃ%XICO, D.F.
```

```
length(unique(cidades$State))
```

```
## [1] 33
```

A variável estado possui 33 valores únicos, vamos trazê-la ao nosso dataset de treinamento, pela chave Agencia_ID:

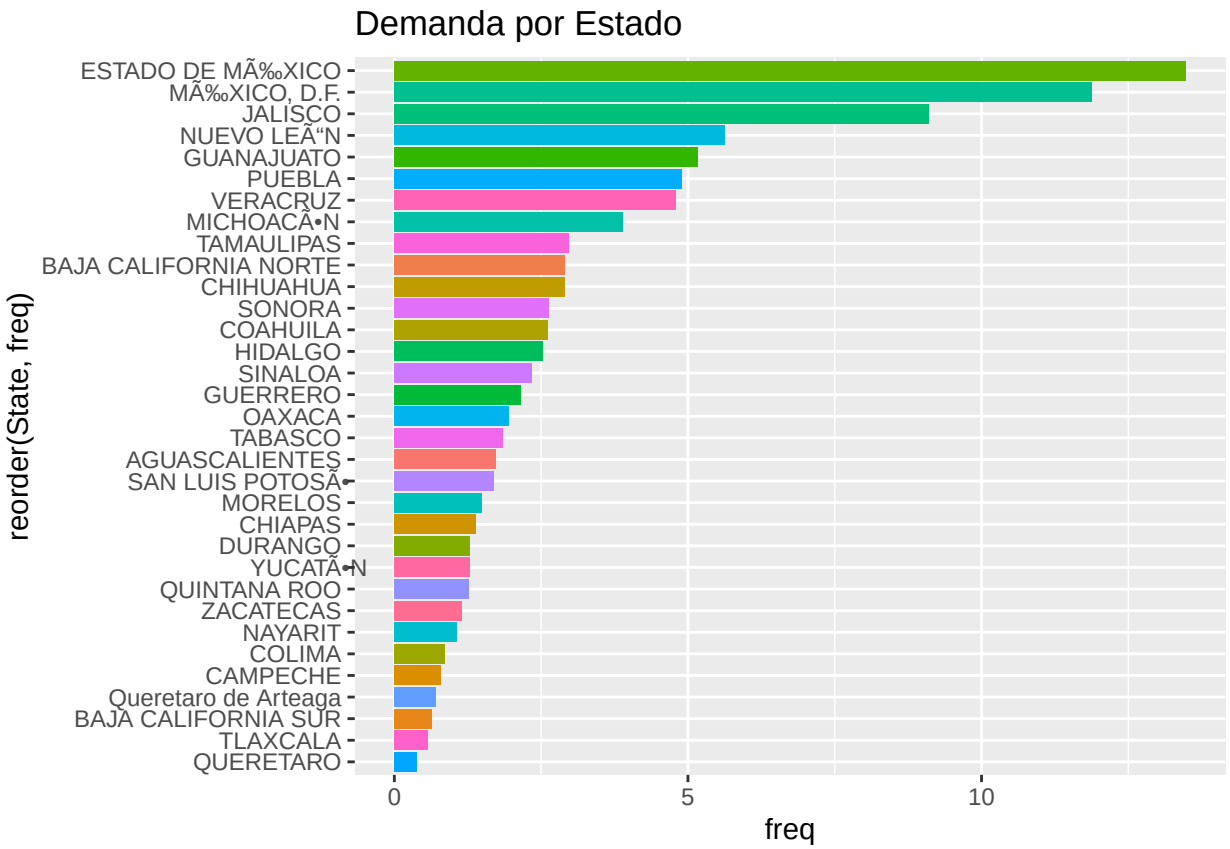
```
train_sample <- train_sample %>%
  left_join(select(cidades, State, Agencia_ID), by = c("Agencia_ID" = "Agencia_ID"))

state_ranking <- train_sample %>%
  group_by(State) %>%
  summarise(sum_Demanda_uni_equil = sum(Demanda_uni_equil)) %>%
  mutate(freq = 100*sum_Demanda_uni_equil/sum(sum_Demanda_uni_equil)) %>%
  arrange(desc(sum_Demanda_uni_equil))

head(state_ranking)
```

```
## # A tibble: 6 x 3
##   State              sum_Demanda_uni_equil  freq
##   <chr>                              <int> <dbl>
## 1 ESTADO DE MÃ‰XICO                 968913 13.5
## 2 MÃ‰XICO, D.F.                     853337 11.9
## 3 JALISCO                          654074  9.10
## 4 NUEVO LEÃ"N                      404829  5.63
## 5 GUANAJUATO                       371564  5.17
## 6 PUEBLA                           351915  4.90
```

```
ggplot(state_ranking, aes(y = reorder(State, freq), x = freq, fill = State)) + geom_bar(stat = "identity
  theme(legend.position = "none") + ggtitle("Demanda por Estado")
```




A demanda é maioria no Estado do Mexico, que é o estado mais populoso do México, e em México D.F que é

aonde está a sede do governo, interessante pois no geral estados mais populosos possuem uma demana maior, o que de fato aparece em nossa amostra de dados. Vamos deixar está variável no nosso modelo.

Vamos criar um dataset de treino apenas com as variáveis que serão usadas nesta primeira versão do modelo, naturalmente iremos excluir as variáveis de ID.

```
train <- train_sample %>%
  select(Semana, short_product_name, local_grup, pieces, weight, piece_per_weight, State, Demanda_uni_e

head(train)
```

```
##   Semana     short_product_name               local_grup pieces weight
## 1      3              Principe          SEM IDENTIFICACAO     10    106
## 2      3 Tostada Ondulada Tubo  MERCADOS/COMERCIOS GERAIS     30    360
## 3      3       Mantecadas Nuez          SEM IDENTIFICACAO     NA    123
## 4      3   Mantecadas Vainilla                LOJA OXXO      4    125
## 5      3          Chocochispas           FRANQUIA PEQUENA     NA     80
## 6      3             Madalenas          SEM IDENTIFICACAO      3     93
##   piece_per_weight             State Demanda_uni_equil
## 1             10.6 ESTADO DE MÃ%XICO                 4
## 2             12.0     MÃ%XICO, D.F.                 2
## 3               NA          GUERRERO                 3
## 4             31.2      QUINTANA ROO                20
## 5               NA        MICHOACÃ\201N                4
## 6             31.0            PUEBLA                 2
```

```
#Verifica valores missings
apply(train, 2, function(x) any(is.na(x)))
```

```
##             Semana short_product_name         local_grup             pieces
##              FALSE              FALSE              FALSE               TRUE
##             weight   piece_per_weight              State  Demanda_uni_equil
##               TRUE               TRUE              FALSE              FALSE
```

O nosso dataset de treino ainda possui alguns valores NA's nas colunas "pieces", "weight" e "piece_per_weight", para a coluna pieces vamos substituir os valores NA por 1. Para representar pelo menos que os produtos que tiveram a sua quantidade omitida tinham no minimo uma peça.

Para a coluna "weight", vamos substituir o peso pela média de pesos do grupo de produtos (short_product_name). E desde que a coluna "piece_per_weight" é dada pelas colunas "weight" e "pieces", basta calcularmos de novo no dataset.

```
train$pieces[is.na(train$pieces)] <- 1

impute.mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))

sum(is.na(train$weight))
```

```
## [1] 9495
```

```
train <- train %>%
  group_by(short_product_name) %>%
  mutate(weight = round(impute.mean(weight),1)) %>%
  mutate(piece_per_weight = round(weight/pieces,1))

sum(is.na(train$weight))
```

```
## [1] 6332
```

Ainda sim sobram 6332 valores NA para a coluna "weights" (eram 9495 anteriormente), para estes valores vamos substituir pela média do tipo de estabelecimento.

```
train <- train %>%
  group_by(local_grup) %>%
  mutate(weight = round(impute.mean(weight),1)) %>%
  mutate(piece_per_weight = round(weight/pieces,1))

sum(is.na(train$weight))
```

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

```
apply(train, 2, function(x) any(is.na(x)))
```

```
##             Semana short_product_name         local_grup             pieces
##              FALSE              FALSE              FALSE              FALSE
##             weight   piece_per_weight              State  Demanda_uni_equil
##              FALSE              FALSE              FALSE              FALSE
```

Sem valores NA no nosso dataset de treino, podemos seguir adiante.

## Análise de Corrrelação

Uma última ánalise que podemos fazer é a análise de correlação para as variáveis númericas.

```
require("corrplot")
```

```
## Loading required package: corrplot
```

```
## corrplot 0.84 loaded
```

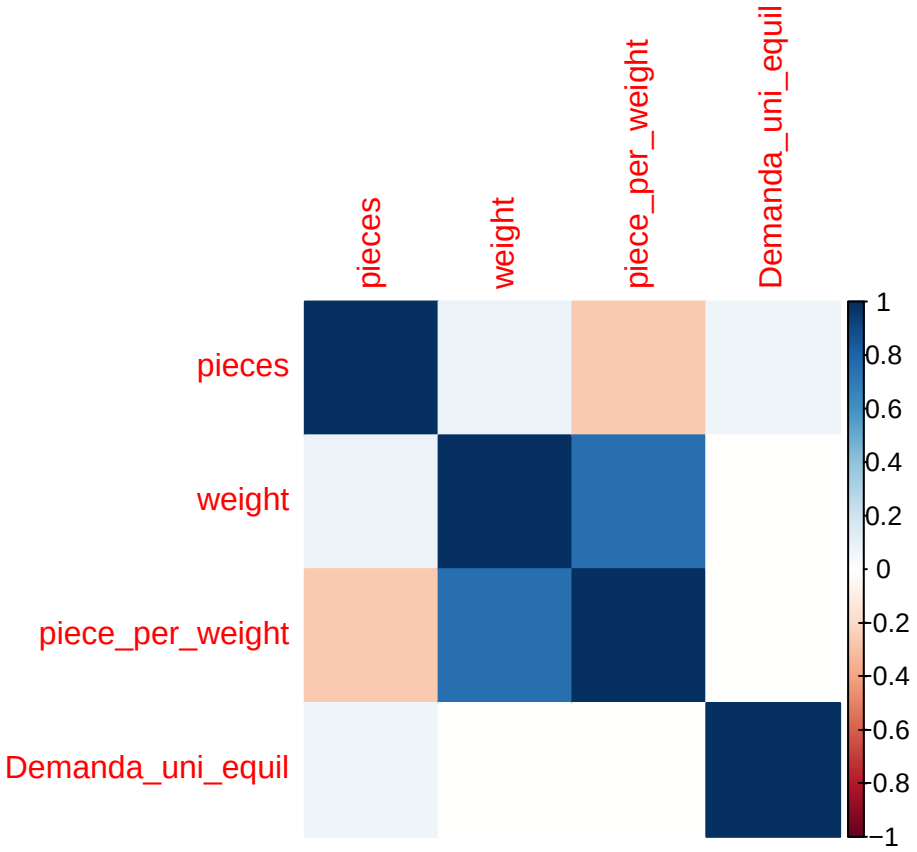
```
train_num <- as.data.frame(train) %>%
  select(pieces, weight, piece_per_weight, Demanda_uni_equil)

str(train_num)
```

```
## 'data.frame':    1000000 obs. of  4 variables:
##  $ pieces           : num  10 30 1 4 1 3 10 8 3 1 ...
##  $ weight           : num  106 360 123 125 80 93 110 120 105 62 ...
##  $ piece_per_weight : num  10.6 12 123 31.2 80 31 11 15 35 62 ...
##  $ Demanda_uni_equil: int  4 2 3 20 4 2 1 4 2 30 ...
```

```
cor <- cor(train_num)

corrplot(cor, method="color")
```




No mapa de correlação, não há nenhuma relação entre as variáveis numericas do dataset e a variável alvo como esperado, se houvesse, o problema seria menos complexo.

## Criação do modelo

Vamos tratar o dataset de treinamento (convertê-lo para dataframe), e transformar as variáveis "char" em categóricas.

```
train <- as.data.frame(train)

train$Semana <- as.factor(train$Semana)
train$short_product_name <- as.factor(train$short_product_name)
train$local_grup <- as.factor(train$local_grup)
train$State <- as.factor(train$State)

str(train)
```

```
## 'data.frame':    1000000 obs. of  8 variables:
##  $ Semana            : Factor w/ 7 levels "3","4","5","6",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ short_product_name: Factor w/ 536 levels "","ActiFresh Menta ",..: 359 497 243 245 123 236 361 41
##  $ local_grup        : Factor w/ 14 levels "CORREIOS","EDUCACAO",..: 12 9 12 7 3 12 3 3 9 7 ...
##  $ pieces            : num  10 30 1 4 1 3 10 8 3 1 ...
##  $ weight            : num  106 360 123 125 80 93 110 120 105 62 ...
##  $ piece_per_weight  : num  10.6 12 123 31.2 80 31 11 15 35 62 ...
##  $ State             : Factor w/ 33 levels "AGUASCALIENTES",..: 10 15 12 24 16 21 26 31 20 2 ...
##  $ Demanda_uni_equil : int  4 2 3 20 4 2 1 4 2 30 ...
```

Vamos usar o algoritmo Extreme Gradient Boosting (XGBoost):

```
# Separando a variável alvo das variáveis independentes:
x_train <- data.matrix(train[, -8])
y_train <- data.matrix(train[, 8])

# Criando a matriz xgb
xgb_train = xgb.DMatrix(data = x_train, label = y_train)

# Treinando o modelo
xgbc1 = xgboost(data = xgb_train, max.depth = 2, nrounds = 50)
```

```
## [1]  train-rmse:20.641714
## [2]  train-rmse:20.033825
## [3]  train-rmse:19.701946
## [4]  train-rmse:19.531042
## [5]  train-rmse:19.401762
## [6]  train-rmse:19.326359
## [7]  train-rmse:19.258226
## [8]  train-rmse:19.185581
## [9]  train-rmse:19.149822
## [10] train-rmse:19.133110
## [11] train-rmse:19.116362
## [12] train-rmse:19.092798
## [13] train-rmse:19.066952
## [14] train-rmse:19.047033
## [15] train-rmse:19.032650
## [16] train-rmse:19.022703
## [17] train-rmse:19.014273
## [18] train-rmse:18.986214
## [19] train-rmse:18.978588
## [20] train-rmse:18.959351
## [21] train-rmse:18.939425
## [22] train-rmse:18.925829
## [23] train-rmse:18.918150
## [24] train-rmse:18.909027
## [25] train-rmse:18.876278
## [26] train-rmse:18.870501
## [27] train-rmse:18.861162
## [28] train-rmse:18.848799
## [29] train-rmse:18.837172
## [30] train-rmse:18.832478
## [31] train-rmse:18.819220
## [32] train-rmse:18.815290
## [33] train-rmse:18.808292
## [34] train-rmse:18.800171
## [35] train-rmse:18.783175
## [36] train-rmse:18.780264
## [37] train-rmse:18.776779
## [38] train-rmse:18.772865
## [39] train-rmse:18.766829
## [40] train-rmse:18.762932
## [41] train-rmse:18.753632
## [42] train-rmse:18.738729
## [43] train-rmse:18.735975
```

```
## [44] train-rmse:18.730604
## [45] train-rmse:18.724573
## [46] train-rmse:18.719423
## [47] train-rmse:18.712957
## [48] train-rmse:18.692993
## [49] train-rmse:18.684408
## [50] train-rmse:18.679106
```

```
print(xgbc1)
```

```
## ##### xgb.Booster
## raw: 39.2 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, max.depth = 2)
## params (as set within xgb.train):
##   max_depth = "2", validate_parameters = "1"
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 7
## niter: 50
## nfeatures : 7
## evaluation_log:
##     iter train_rmse
##        1   20.64171
##        2   20.03383
## ---
##       49   18.68441
##       50   18.67911
```

```
# Usando como parâmetro o RMSE do treinamento, o valor é alto 18.6791. Antes de tentar otimizar o model

pred_y = predict(xgbc1, xgb_train)

y_train_mean = mean(y_train)

# Cálculo do R^2:
tss =  sum((y_train - y_train_mean)^2 )

residuals = y_train - pred_y

rss =  sum(residuals^2)

rsq  =  1 - (rss/tss)
rsq
```

```
## [1] 0.1883533
```

O valor de R^2 obtido foi de 0,1883, está muito longe de 1 (ideal), vamos tentar otimizar o modelo para aumentá-lo e diminuir o RMSE.

## Otimização do Modelo

Vamos aumentar a profundidade da árvore por trás do XGBoost:

```
xgbc2 = xgboost(data = xgb_train, max.depth = 4, nrounds = 50)
```

```
## [1]  train-rmse:20.385553
## [2]  train-rmse:19.639103
## [3]  train-rmse:19.257822
## [4]  train-rmse:19.032948
## [5]  train-rmse:18.772871
## [6]  train-rmse:18.593491
## [7]  train-rmse:18.541651
## [8]  train-rmse:18.491594
## [9]  train-rmse:18.439495
## [10] train-rmse:18.395559
## [11] train-rmse:18.344591
## [12] train-rmse:18.318340
## [13] train-rmse:18.238930
## [14] train-rmse:18.218000
## [15] train-rmse:18.202160
## [16] train-rmse:18.132095
## [17] train-rmse:18.078592
## [18] train-rmse:18.051985
## [19] train-rmse:18.035843
## [20] train-rmse:18.008257
## [21] train-rmse:17.979954
## [22] train-rmse:17.951136
## [23] train-rmse:17.931561
## [24] train-rmse:17.922274
## [25] train-rmse:17.909678
## [26] train-rmse:17.878319
## [27] train-rmse:17.861681
## [28] train-rmse:17.850929
## [29] train-rmse:17.839546
## [30] train-rmse:17.835913
## [31] train-rmse:17.807135
## [32] train-rmse:17.795153
## [33] train-rmse:17.789139
## [34] train-rmse:17.776159
## [35] train-rmse:17.642704
## [36] train-rmse:17.632828
## [37] train-rmse:17.616594
## [38] train-rmse:17.527084
## [39] train-rmse:17.516876
## [40] train-rmse:17.513790
## [41] train-rmse:17.449697
## [42] train-rmse:17.444273
## [43] train-rmse:17.418394
## [44] train-rmse:17.414593
## [45] train-rmse:17.394264
## [46] train-rmse:17.391228
## [47] train-rmse:17.386127
## [48] train-rmse:17.365173
## [49] train-rmse:17.360109
```

```
## [50] train-rmse:17.352262
```

```
print(xgbc2)
```

```
## ##### xgb.Booster
## raw: 101.1 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, max.depth = 4)
## params (as set within xgb.train):
##   max_depth = "4", validate_parameters = "1"
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 7
## niter: 50
## nfeatures : 7
## evaluation_log:
##     iter train_rmse
##        1   20.38555
##        2   19.63910
## ---
##       49   17.36011
##       50   17.35226
```

Dobrando a profundidade, o RMSE foi para 17.3523, podemos extrapolar e aumentar muito a profundidade:

```
xgbc3 = xgboost(data = xgb_train, max.depth = 32, nrounds = 50)
```

```
## [1]  train-rmse:18.687149
## [2]  train-rmse:16.652542
## [3]  train-rmse:15.267508
## [4]  train-rmse:14.339376
## [5]  train-rmse:13.710159
## [6]  train-rmse:13.274789
## [7]  train-rmse:12.974125
## [8]  train-rmse:12.768022
## [9]  train-rmse:12.622147
## [10] train-rmse:12.518668
## [11] train-rmse:12.445380
## [12] train-rmse:12.393955
## [13] train-rmse:12.357283
## [14] train-rmse:12.331034
## [15] train-rmse:12.312365
## [16] train-rmse:12.299113
## [17] train-rmse:12.289682
## [18] train-rmse:12.282924
## [19] train-rmse:12.278101
## [20] train-rmse:12.274650
## [21] train-rmse:12.272038
## [22] train-rmse:12.270201
```

```
## [23] train-rmse:12.268869
## [24] train-rmse:12.267931
## [25] train-rmse:12.267232
## [26] train-rmse:12.266747
## [27] train-rmse:12.266390
## [28] train-rmse:12.266150
## [29] train-rmse:12.265975
## [30] train-rmse:12.265844
## [31] train-rmse:12.265758
## [32] train-rmse:12.265699
## [33] train-rmse:12.265653
## [34] train-rmse:12.265626
## [35] train-rmse:12.265606
## [36] train-rmse:12.265587
## [37] train-rmse:12.265579
## [38] train-rmse:12.265573
## [39] train-rmse:12.265568
## [40] train-rmse:12.265564
## [41] train-rmse:12.265558
## [42] train-rmse:12.265556
## [43] train-rmse:12.265559
## [44] train-rmse:12.265560
## [45] train-rmse:12.265560
## [46] train-rmse:12.265558
## [47] train-rmse:12.265561
## [48] train-rmse:12.265564
## [49] train-rmse:12.265563
## [50] train-rmse:12.265567
```

```
print(xgbc3)
```

```
## ##### xgb.Booster
## raw: 418.6 Mb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, max.depth = 32)
## params (as set within xgb.train):
##   max_depth = "32", validate_parameters = "1"
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 7
## niter: 50
## nfeatures : 7
## evaluation_log:
##     iter train_rmse
##        1   18.68715
##        2   16.65254
## ---
##       49   12.26556
```

```
##       50   12.26557
```

Desta vez o RMSE diminuiu bem - 12.26, vamos verificar o R^2:

```
pred_y = predict(xgbc3, xgb_train)

tss =  sum((y_train - y_train_mean)^2 )

residuals = y_train - pred_y

rss =  sum(residuals^2)

rsq  =  1 - (rss/tss)
rsq
```

```
## [1] 0.6500165
```

O R^2 foi para 0.65, considerando que em alguns outros modelos que foram omtidos neste documento o maior R^2 estava sendo por volta de 10, e na nossa primeira versão foi de 0,1883 não é algo ruim, tentamos extraploar ainda mais a profunidade da rede para 128, com 200 iterações, o tempo de treinamento do modelo auemntou muutio, mas a smétricas RMSE e R^2 foram praticamente as mesmas, assim sendo vamos manter os parâmetros do modelo 3.

```
final <- train %>%
  select(Semana, Demanda_uni_equil)

final <- cbind(final, pred_train = round(pred_y,1))

View(final)
str(final)
```

```
## 'data.frame':    1000000 obs. of  3 variables:
##  $ Semana           : Factor w/ 7 levels "3","4","5","6",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Demanda_uni_equil: int  4 2 3 20 4 2 1 4 2 30 ...
##  $ pred_train       : num  5.7 4.4 2.9 30 2.6 3.5 3.4 6.9 2.6 26.2 ...
```

```
final_med <- final %>%
  group_by(Semana) %>%
  summarise_at(vars(Demanda_uni_equil,pred_train), mean)

head(final_med)
```

```
## # A tibble: 6 x 3
##   Semana Demanda_uni_equil pred_train
##   <fct>              <dbl>      <dbl>
## 1 3                   6.86       6.86
## 2 4                   7.25       7.25
## 3 5                   7.28       7.28
## 4 6                   7.24       7.24
## 5 7                   7.34       7.34
## 6 8                   7.20       7.19
```

# Conclusão

As médias das demandas ficaram identicas entre as reais e previstas, tudo bem que estes são dados de treinamento, e os dados de teste disponibilizados não possuem os valores da variável alvo para comparação, assim vamos terminar a nossa análise por aqui. O Valor do RMSE final de 12.26 é muito alto, principalmente

considerando que são dados de treinamento, mas parece ser o minimo valor que obteremos com astransformações realizadas, o valor de R^2 também está longe do ideal, mas apresentou uma significativa melhora em relação a primeira versão do modelo.

É provável que para se melhorar este número, sejam necessárias novas mudanças nos dados, e uma análise exploratória mais profunda, analisando os trabalhos dos competidores do Kaggle, vejo que como Cientista de Dado tenho muito o que melhorar ainda, mas este é o caminho. Há de se destacar a complexidade do problema que basicamente fornece variáveis de ID ou categóricas para regressão, entretanto, vale destacar que este é um problema real, proposto por uma empresa real, e naturalmente os problemas "reais" são tão ou mais dificeis quanto este.