Exploratory analysis and predictive model development for inventory demand forecast.

Mario Peres

October 20, 2021

```
knitr::opts_chunk$set(echo = TRUE)
```

This project was developed as part of my studies at the course Big Data Analytics with R and Azure ML offerec by Data Science Academy. It was developed a predictive model to support the inventory demand forecast, based on the XGBoost algorithm.

File descriptions:

train.csv — the training set test.csv — the test set sample_submission.csv — a sample submission file in the correct format cliente_tabla.csv — client names (can be joined with train/test on Cliente_ID) producto_tabla.csv — product names (can be joined with train/test on Producto_ID) town_state.csv — town and state (can be joined with train/test on Agencia_ID)

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)

Let's get started!!!

First of all, let's load the packages that were used in this project and set seed, so someone can reproduce the results observed in this project.

```
# Loading packages
library(data.table)
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(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(ggplot2)

library(caret)
```

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

```
library(e1071)
library(xgboost)
```

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

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

```
# Setting seed
set.seed(123)
```

Let's load the test data set (test.csv) and visualize it.

```
# Loading data set test_data
test_data <- fread('grupo-bimbo-inventory-demand/test.csv')

# Visualizing the test_data
head(test_data)
```

```
##    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
```

Based on the predictor variables present in the test data, the same variables were selected in the train data, as they will be the ones to be presented to our model.

Loading train data and selecting the chosen variables.

```
# Loading data set train_data
train_data <- fread('grupo-bimbo-inventory-demand/train.csv') %>%
  select(Semana, Agencia_ID, Canal_ID, Ruta_SAK, Cliente_ID, Producto_ID, Demanda_uni_equil)
```

```
# Visualizing the train_data
head(train_data)
```

```
##    Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID Demanda_uni_equil
## 1:      3       1110        7     3301      15766        1212                 3
## 2:      3       1110        7     3301      15766        1216                 4
## 3:      3       1110        7     3301      15766        1238                 4
## 4:      3       1110        7     3301      15766        1240                 4
## 5:      3       1110        7     3301      15766        1242                 3
## 6:      3       1110        7     3301      15766        1250                 5
```

```
# Checking the dimension of the data set train_data
dim(train_data)
```

```
## [1] 74180464        7
```

As there are too many registers (> 74 million) in train_data, before we start doing the analysis and create the first version of the models, let's collect a sample of the train_data for fast processing and testing.

```
# Creating the variable index
train_data$index = c(1:length(train_data$Semana))

# Sampling 100000 indexes to build the train_sample
train_sample_index = sample(train_data$index, 100000, replace = FALSE)

# Creating train_sample with all the registers contained the train_sample_index (Client_IDs)
train_sample = train_data[train_sample_index,]

# Removing the variable index, which was only used for sampling
train_sample$index = NULL
```

So, the data set has reduced from about 74 million to 100 thousands observations. It will make the work faster with the smaller data set and draw the first conclusions. Later on, a larger part (10 million registers) of the original train data set was used to create the final version of the model.

PRELIMINARY EXPLORATORY ANALYSIS

Let's perform a preliminary exploratory analysis to check the presence of NA values, to check data types of variables and transform them to the right type if necessary, to check unique values for each categorical variables, to check some descriptive statistics and to understand how the variables are distributed.

Checking the presence of NA values.

```
# Checking NA values
sapply(colnames(train_sample), function(x) {sum(is.na(train_sample[,..x]))})
```

```
##            Semana        Agencia_ID          Canal_ID          Ruta_SAK
##                 0                 0                 0                 0
##        Cliente_ID       Producto_ID Demanda_uni_equil
##                 0                 0                 0
```

There are no NA values in our data set.

Checking the data type for each variable in the train_data set.

```
# Checking data type
glimpse(train_sample)
```

```
## Rows: 100,000
## Columns: 7
## $ Semana            <int> 5, 4, 4, 6, 9, 8, 9, 8, 6, 4, 6, 5, 5, 8, 9, 8, 3, 6~
## $ Agencia_ID        <int> 1552, 1121, 1215, 1974, 1259, 2263, 1331, 1219, 1150~
## $ Canal_ID          <int> 1, 1, 1, 5, 7, 1, 1, 1, 4, 1, 4, 1, 1, 1, 1, 1, 1, 1~
## $ Ruta_SAK          <int> 2806, 1455, 1413, 3003, 3345, 2807, 1204, 1203, 6616~
## $ Cliente_ID        <int> 471946, 162516, 371402, 653378, 1128918, 4486429, 19~
## $ Producto_ID       <int> 35456, 1240, 1250, 30552, 1242, 37057, 1146, 1232, 3~
## $ Demanda_uni_equil <int> 2, 7, 4, 34, 4, 1, 1, 1, 0, 1, 10, 2, 3, 6, 1, 2, 2,~
```

All variables were recognized as numeric variables but a few of them should be transformed to categorical type.

Variables that were transformed to categorical include:

- Semana;
- Agencia_ID;
- Canal_ID;
- Ruta_SAK;
- Cliente_ID;
- Producto_ID.

```
# Transforming categorical variables
train_sample = train_sample %>%
  mutate(Semana = as.factor(Semana),
         Agencia_ID = as.factor(Agencia_ID),
         Canal_ID = as.factor(Canal_ID),
         Ruta_SAK = as.factor(Ruta_SAK),
         Cliente_ID = as.factor(Cliente_ID),
         Producto_ID = as.factor(Producto_ID))
```

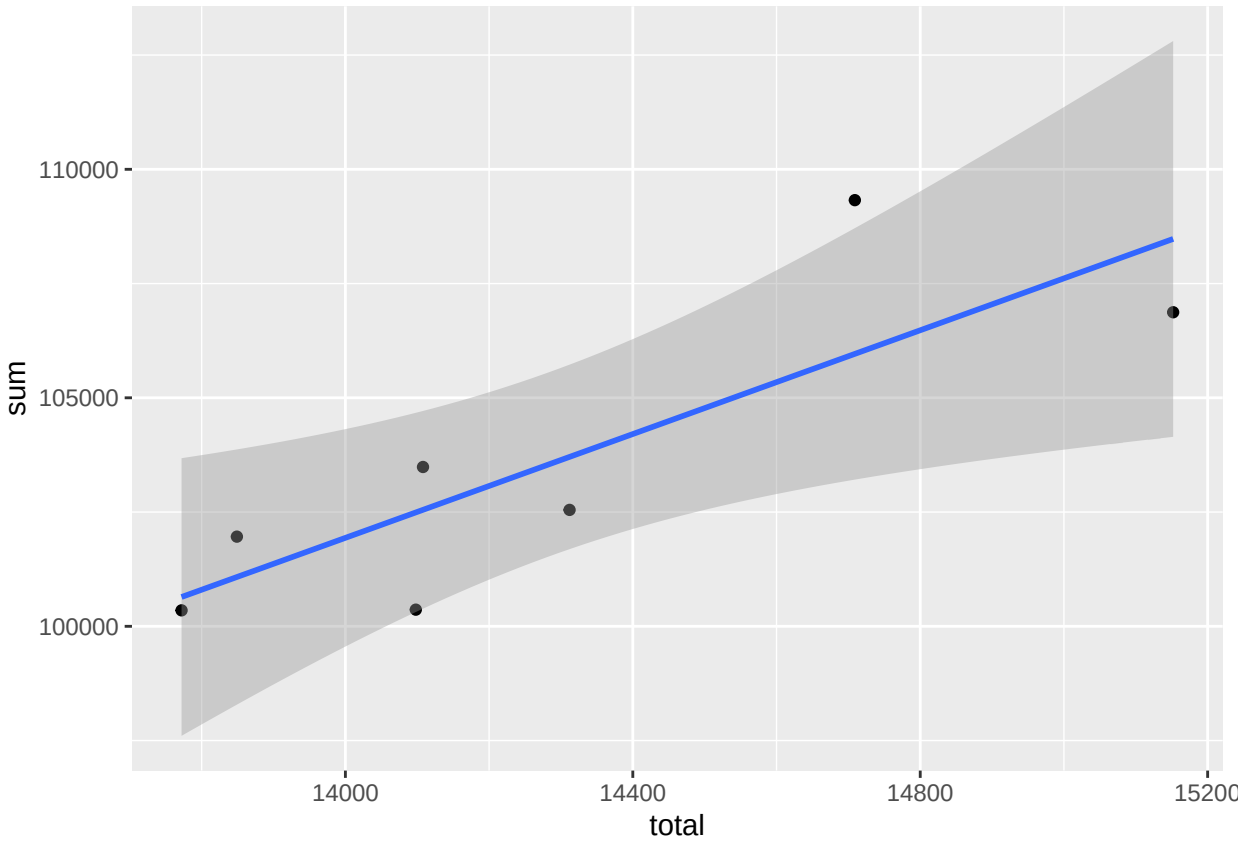
Distribution of variables.

Checking distribution of the sum of Demanda_uni_equil by week and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Semana) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(data = temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

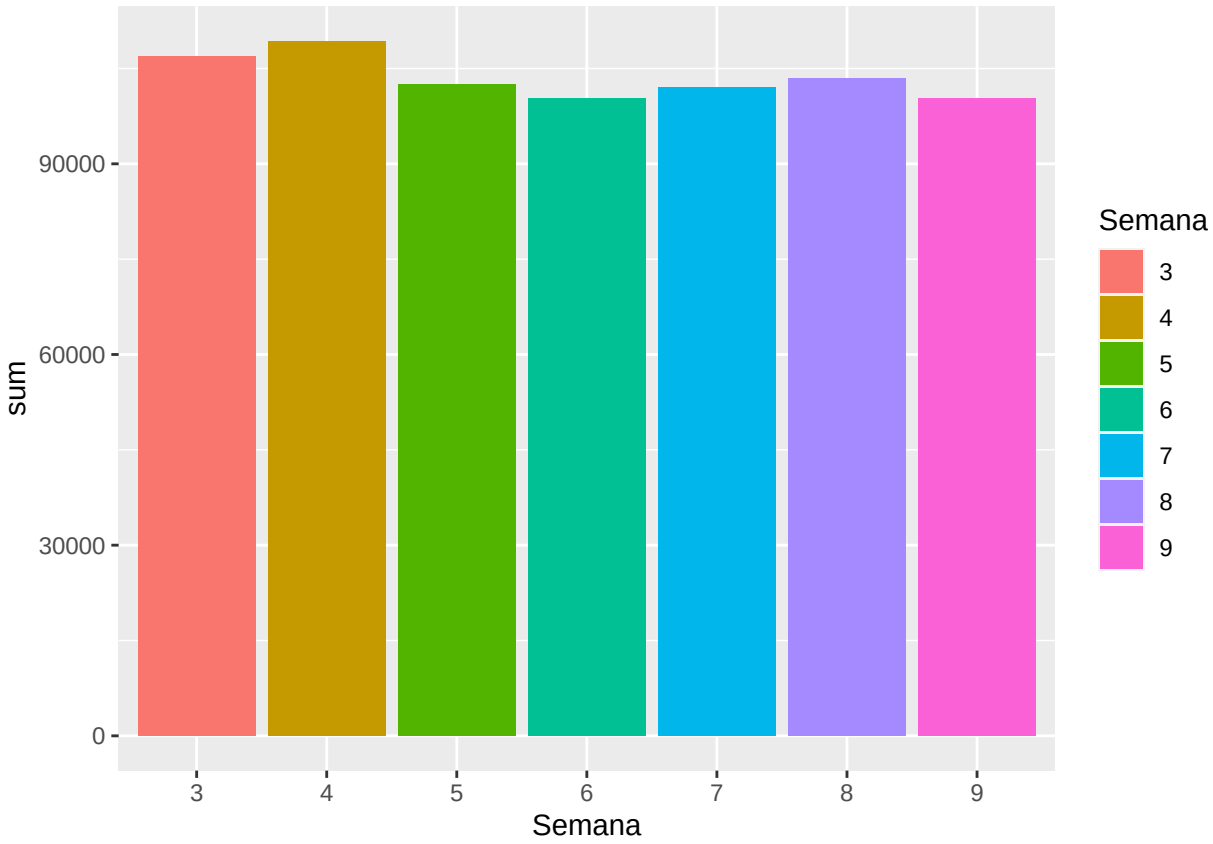
```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.8277413
```

```
train_sample %>%
  group_by(Semana) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil)) %>%
  ggplot() +
  geom_bar(aes(x = Semana, y = sum, fill = Semana), stat = 'identity')
```

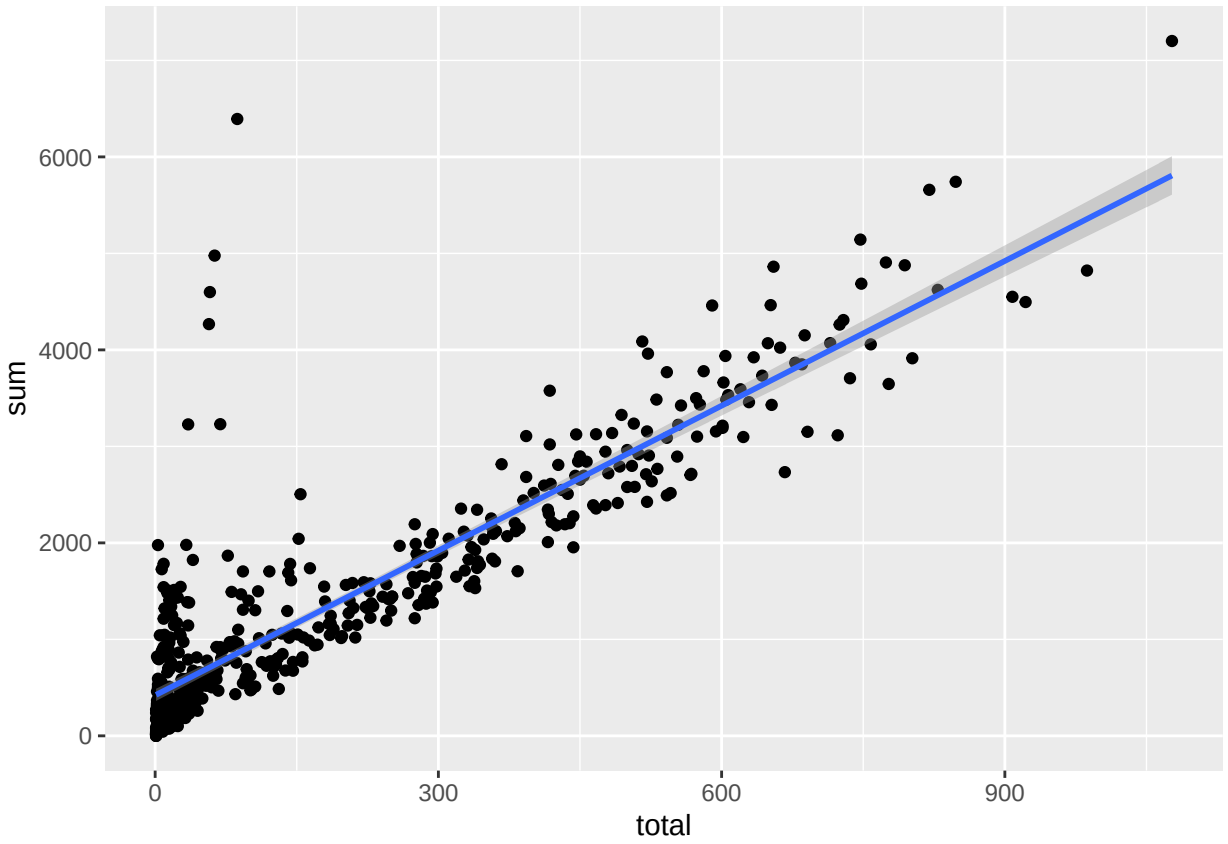


Checking distribution for Agencia_ID and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
  geom_point(aes(total, sum)) +
  geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

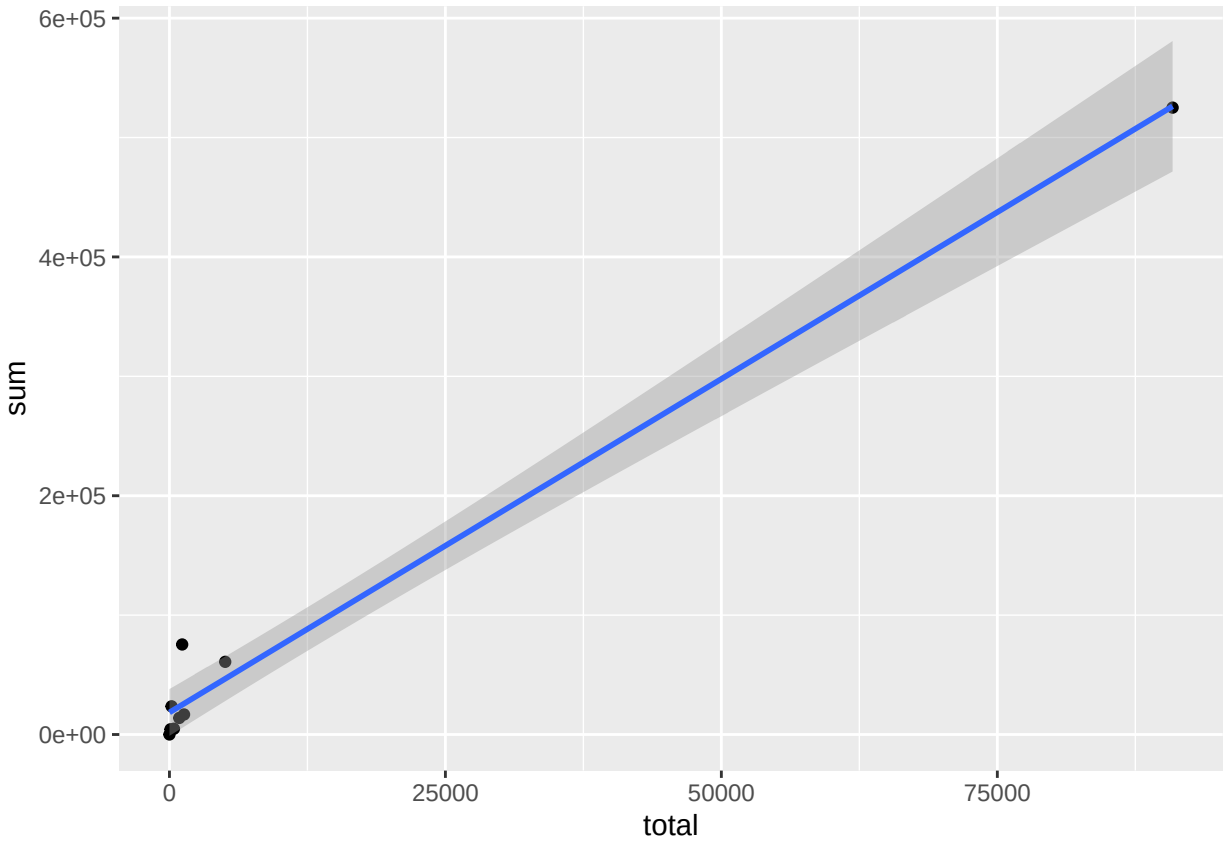
```
## [1] 0.8916681
```

Checking the distribution for Canal_ID and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Canal_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
  geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

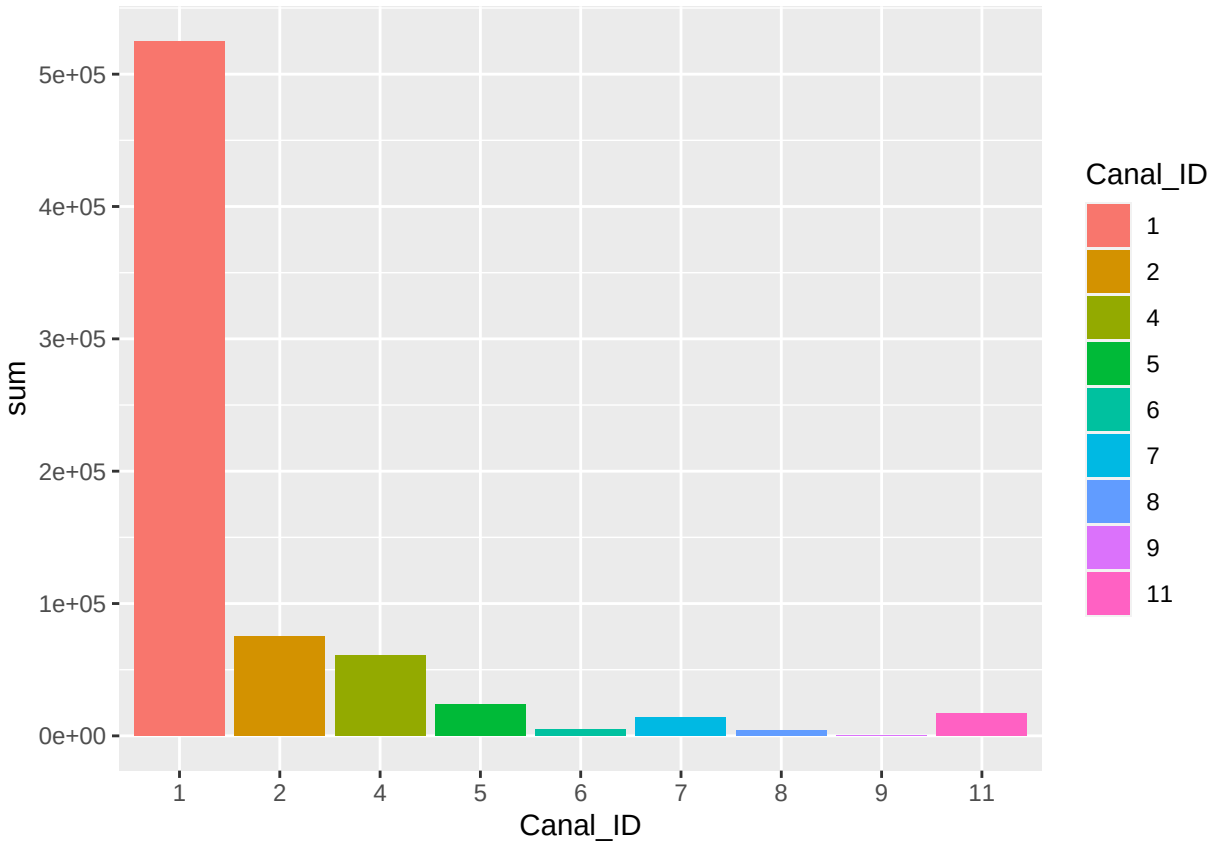
```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.9917343
```

```
train_sample %>%
  group_by(Canal_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil)) %>%
  ggplot() +
  geom_bar(aes(x = Canal_ID, y = sum, fill = Canal_ID), stat = 'identity')
```

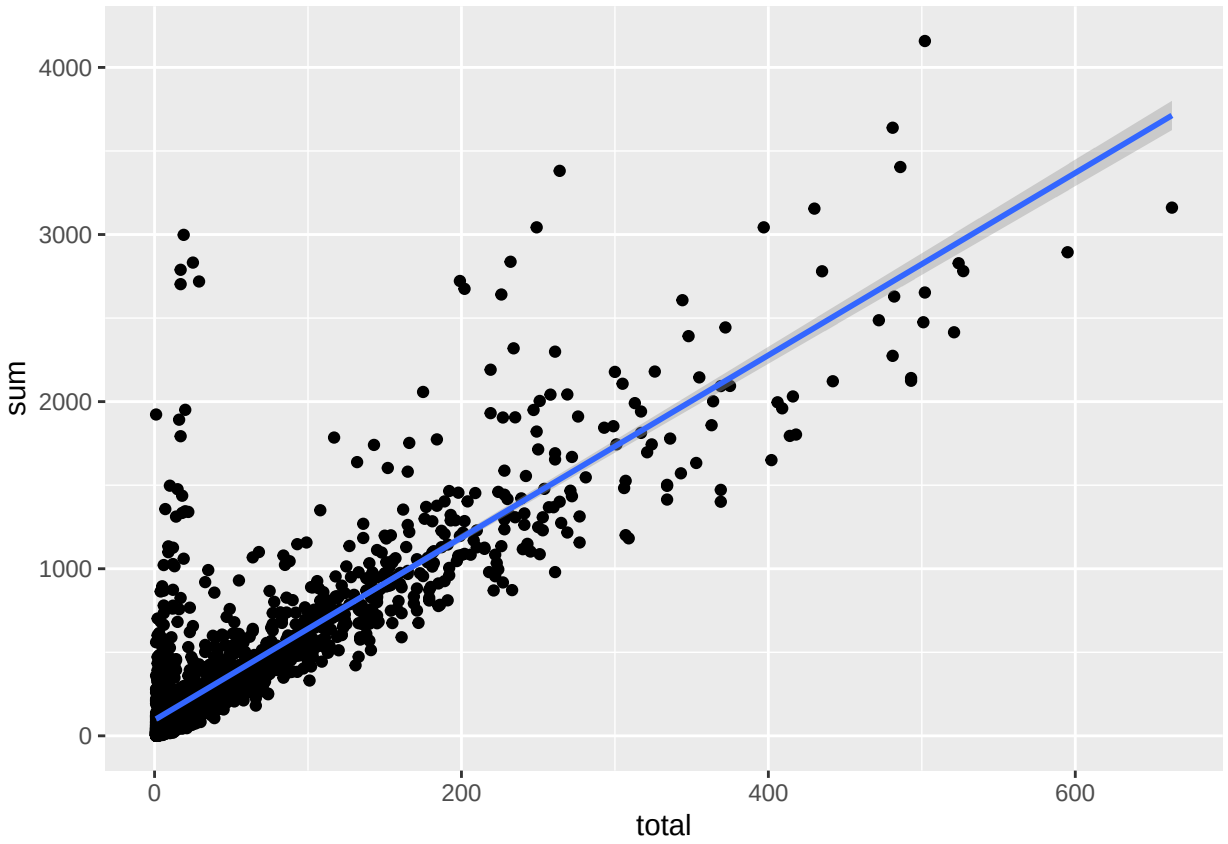


Checking distribution for Ruta_SAK and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

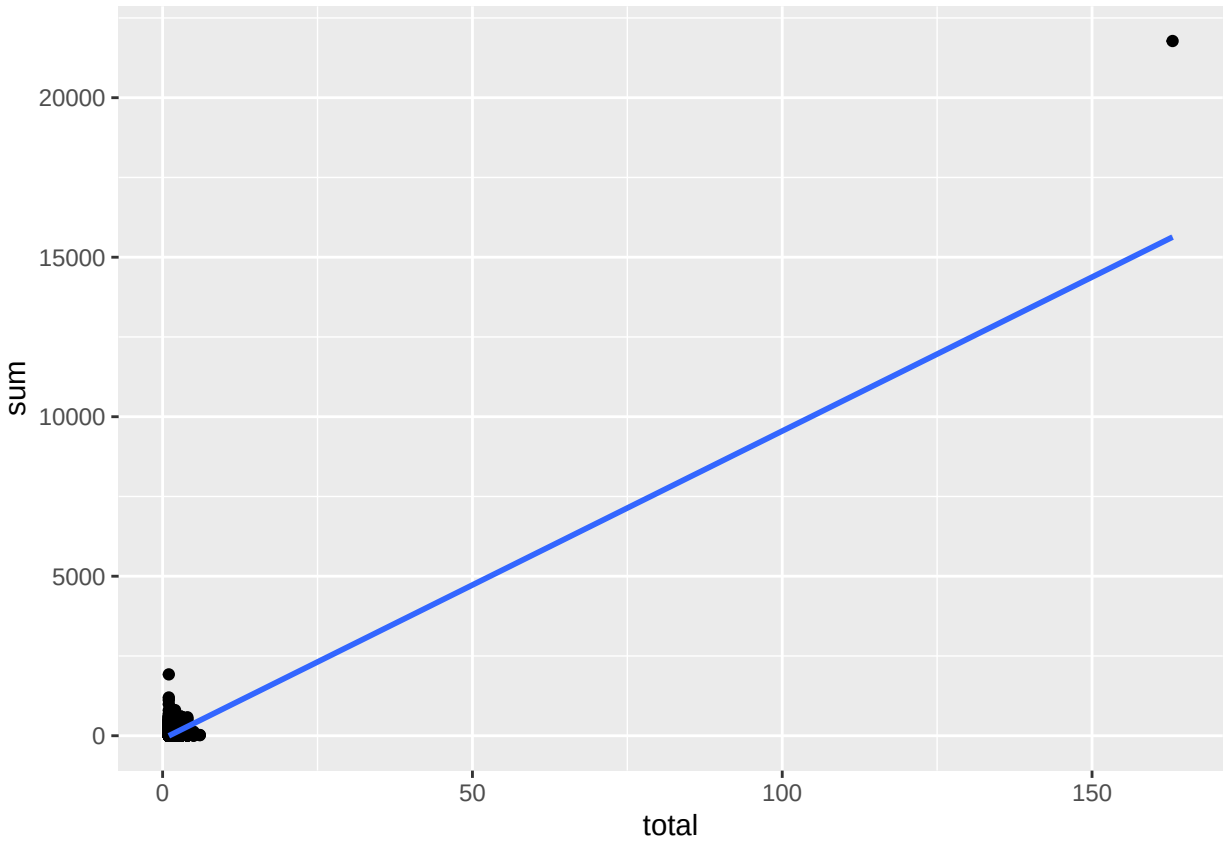
```
## [1] 0.8658028
```

Checking distribution for Cliente_ID and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Cliente_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

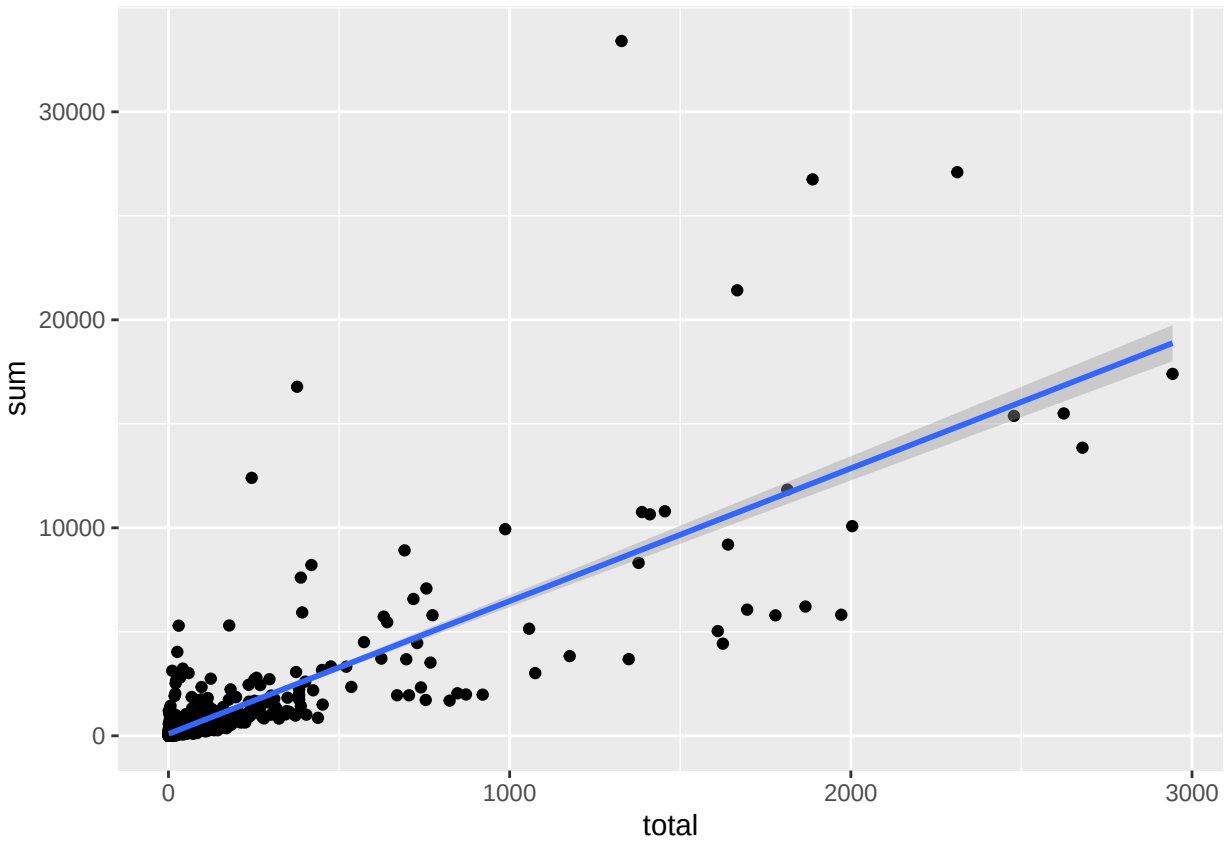
```
## [1] 0.8282168
```

Checking distribution for Producto_ID and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Producto_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.8029309
```

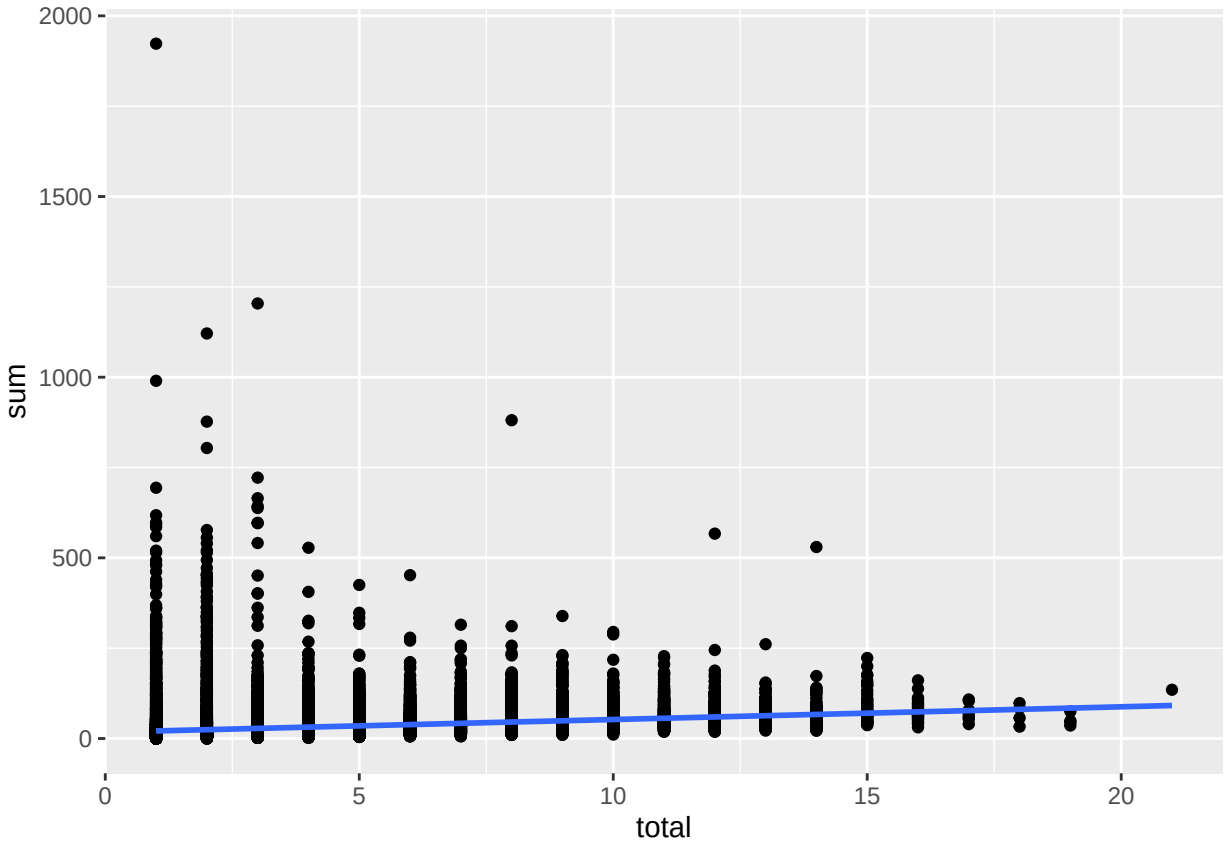
Checking distribution for Cliente_ID and Producto_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Ruta_SAK) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.2170783
```

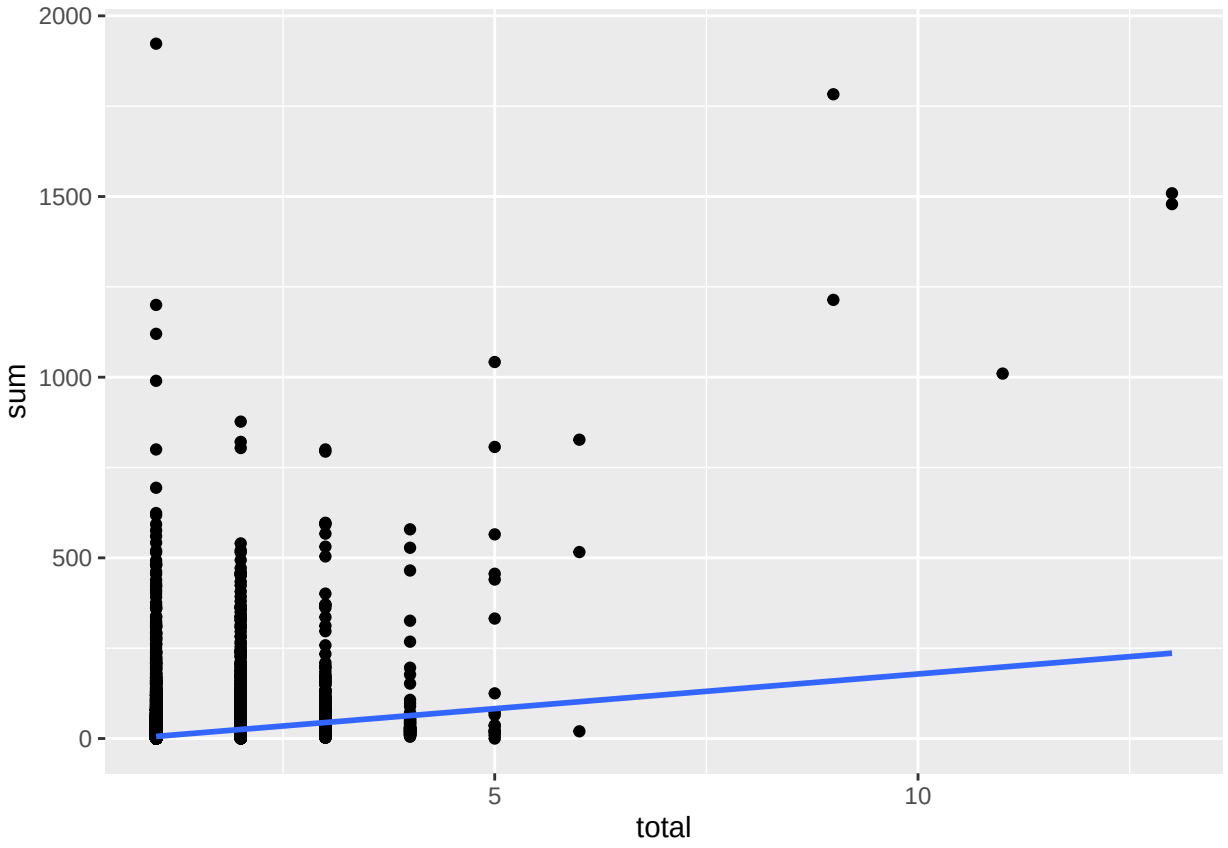
Checking distribution for Cliente_ID and Producto_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Cliente_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.2732448
```

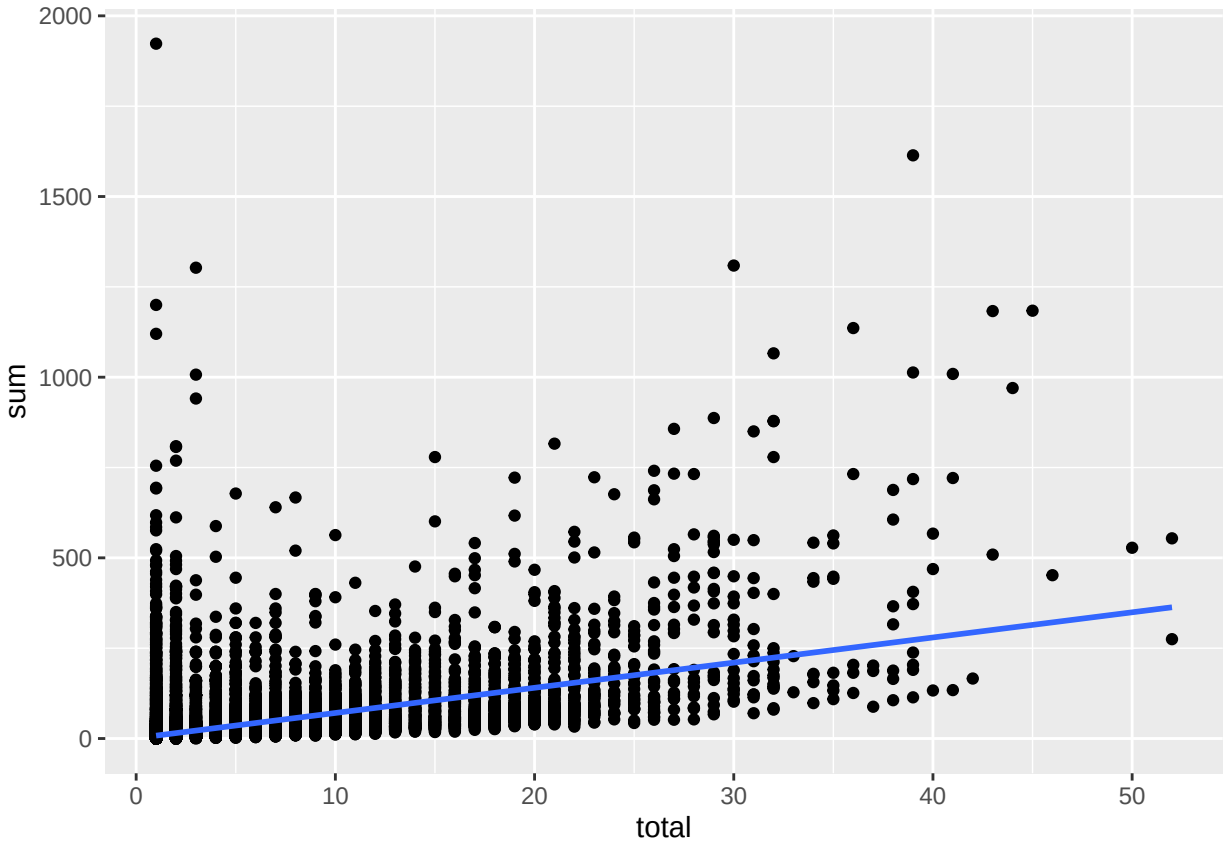
Checking distribution for Agencia_ID and Producto_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Producto_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.5215244
```

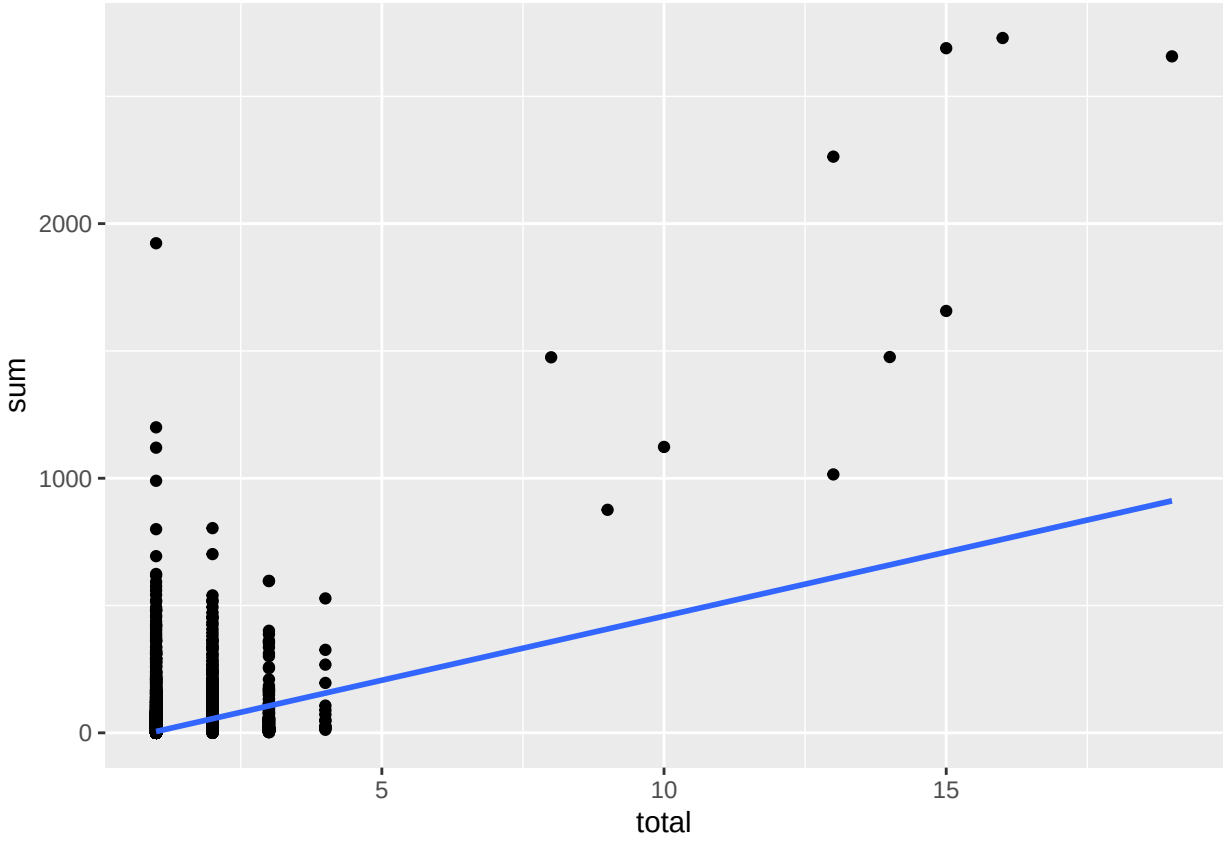
Checking distribution for Ruta_SAK and Cliente_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK, Cliente_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Ruta_SAK'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.4525071
```

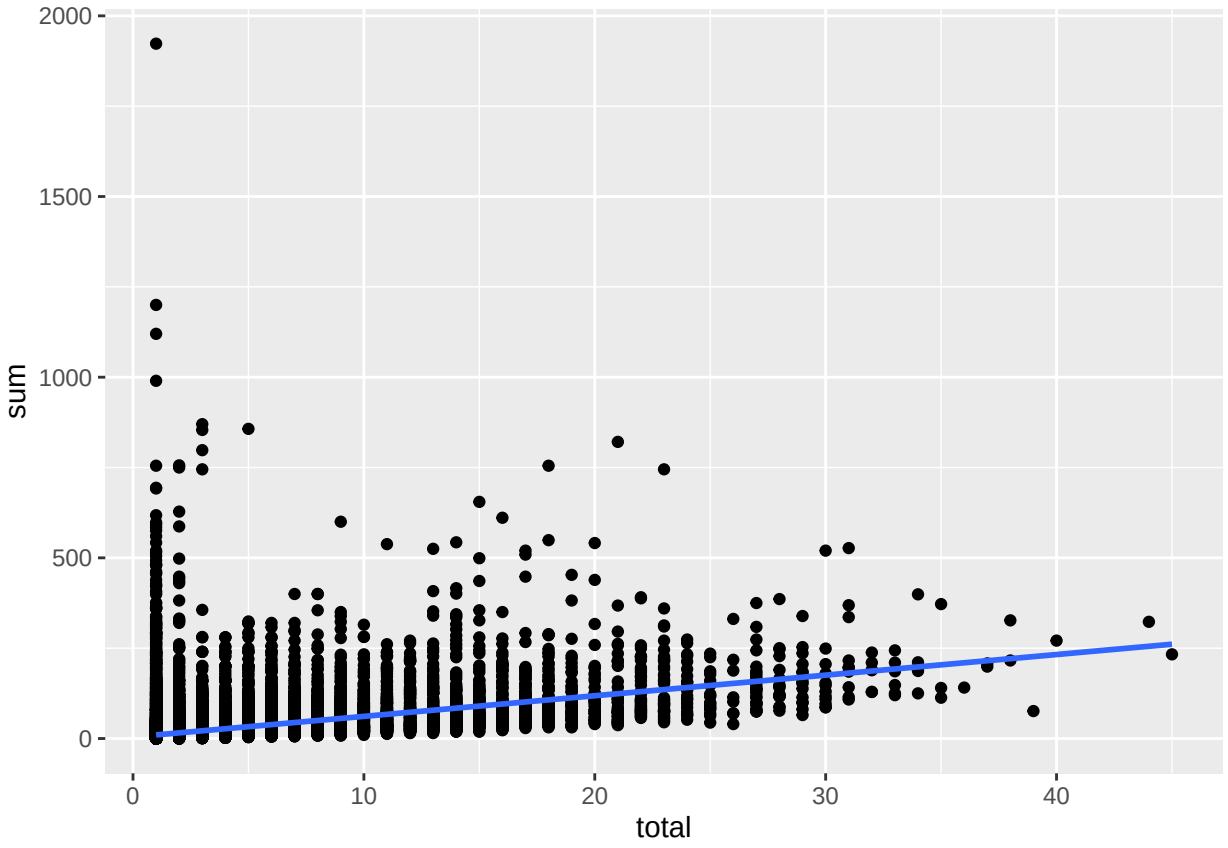
Checking distribution for Ruta_SAK and Producto_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK, Producto_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Ruta_SAK'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
  cor(temp$total, temp$sum)
```

```
## [1] 0.4562532
```

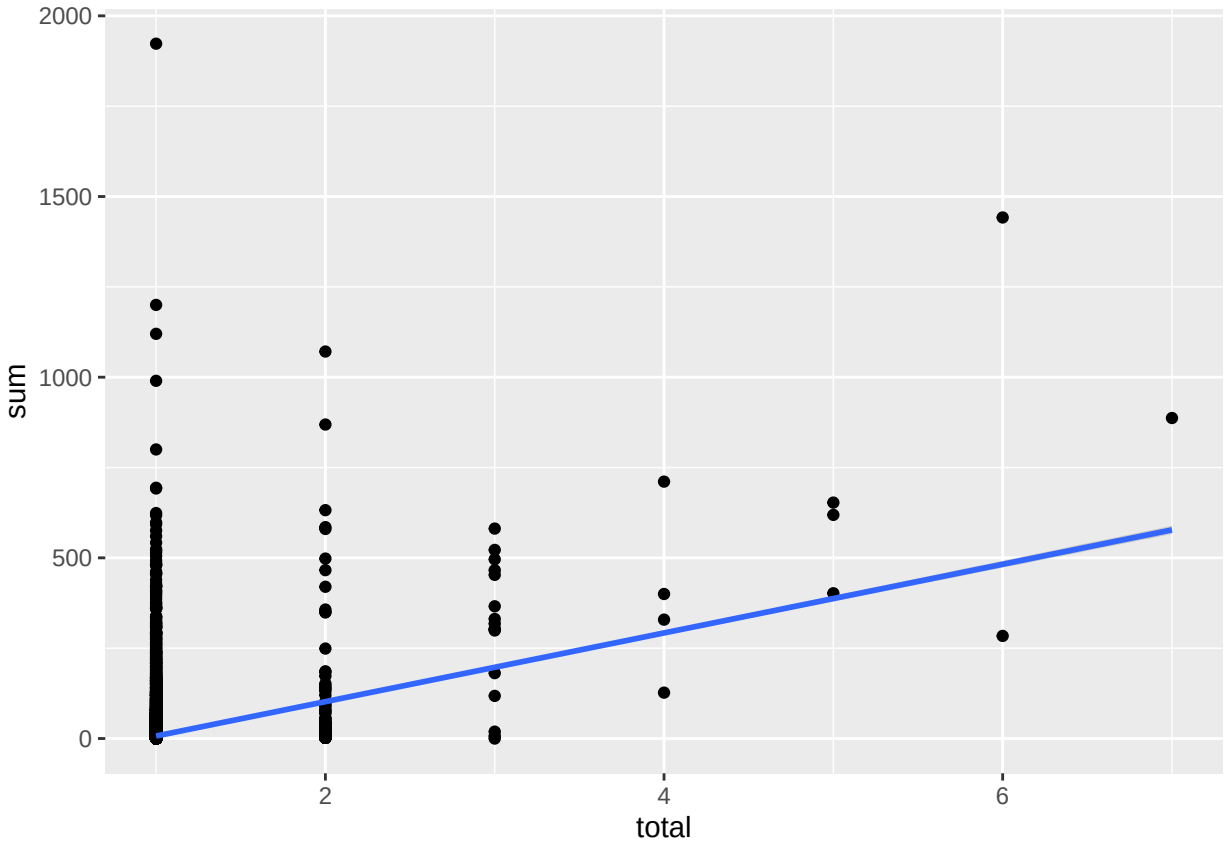
Checking distribution for Cliente_ID and Producto_ID combined and correlation.

```
temp = train_sample %>%
  dplyr::group_by(Cliente_ID, Producto_ID) %>%
  dplyr::summarise(sum = sum(Demanda_uni_equil),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Cliente_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



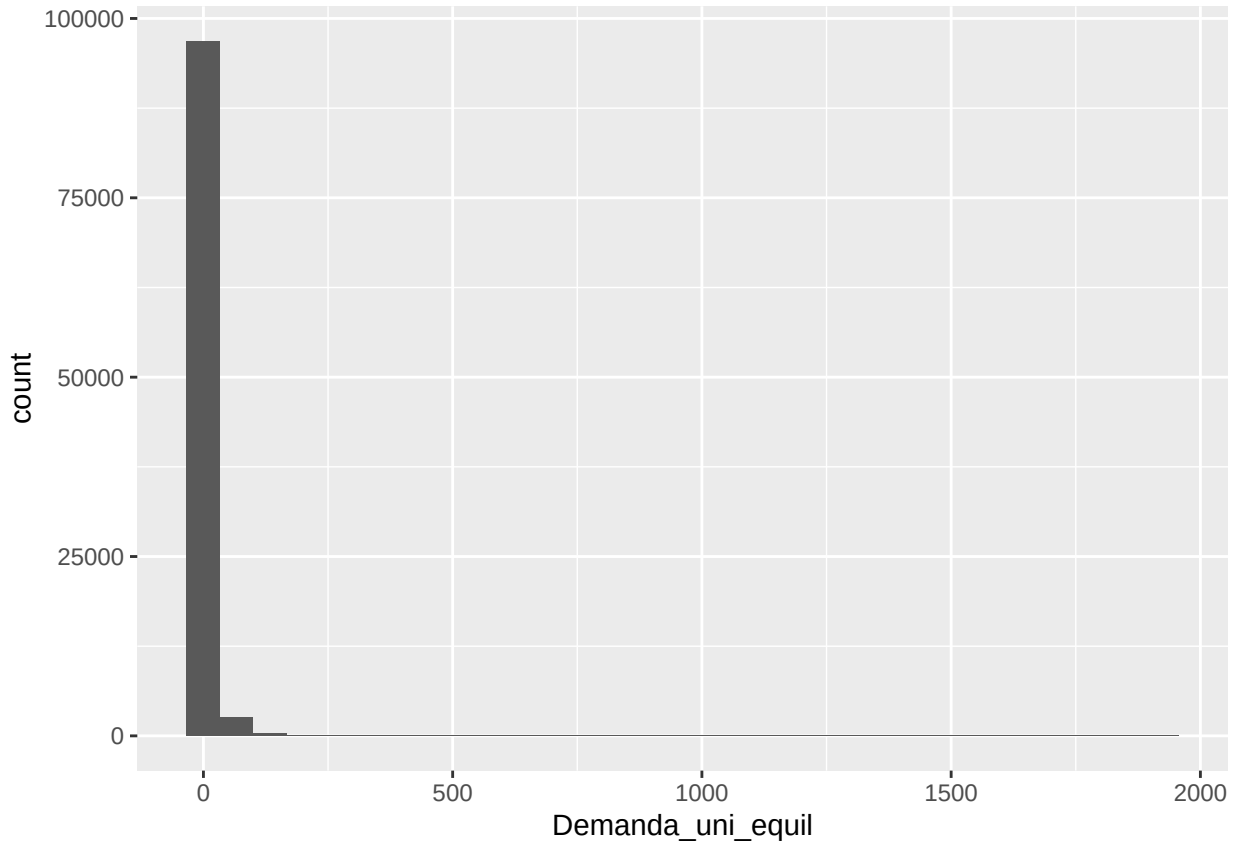

```
  cor(temp$total, temp$sum,)
```

```
## [1] 0.297021
```

Checking distribution for Demanda_uni_equil using original and log10 scales.

```
ggplot(train_sample) +
  geom_histogram(aes(Demanda_uni_equil))
```

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

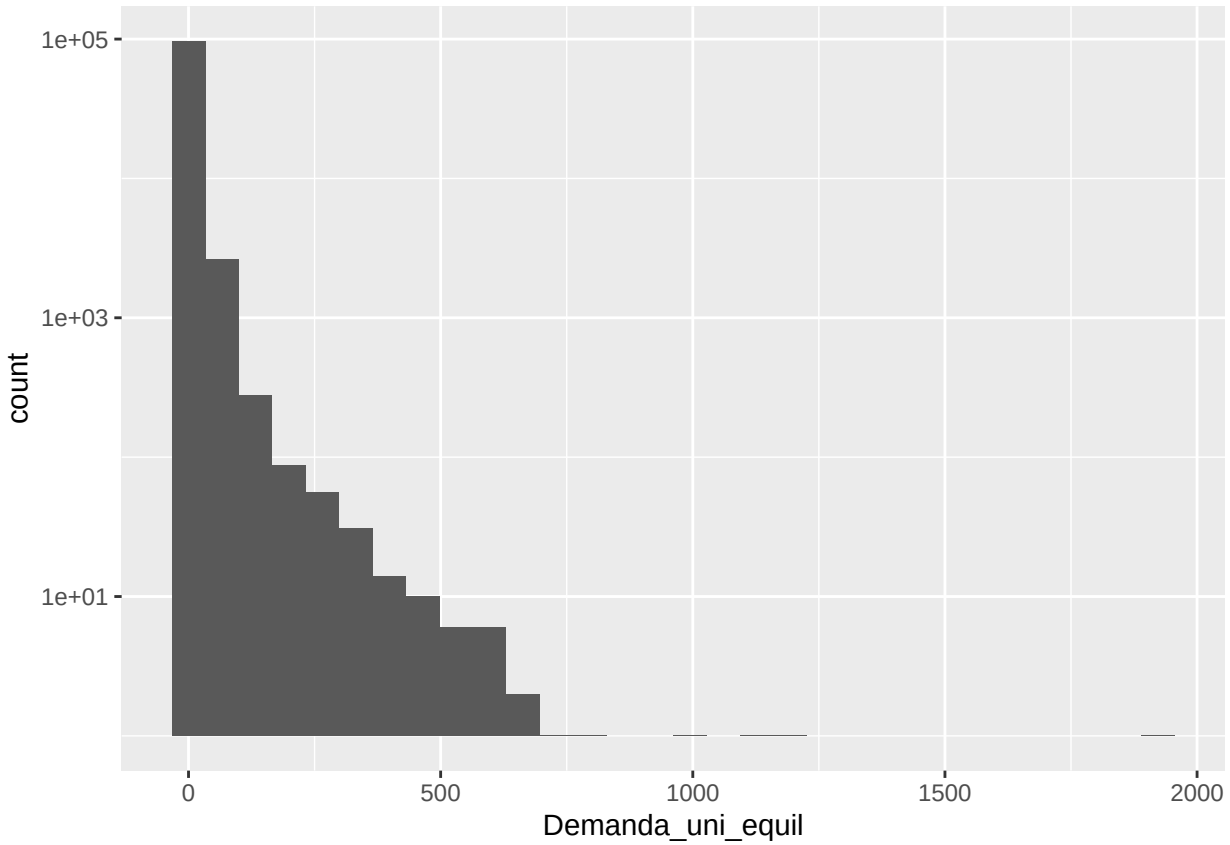



```
ggplot(train_sample) +
  geom_histogram(aes(Demanda_uni_equil)) +
  scale_y_log10()
```

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

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 13 rows containing missing values (geom_bar).
```



Let's try to convert the Demanda_uni_equil to log10 scale and check if the correlations are slightly improved or not.

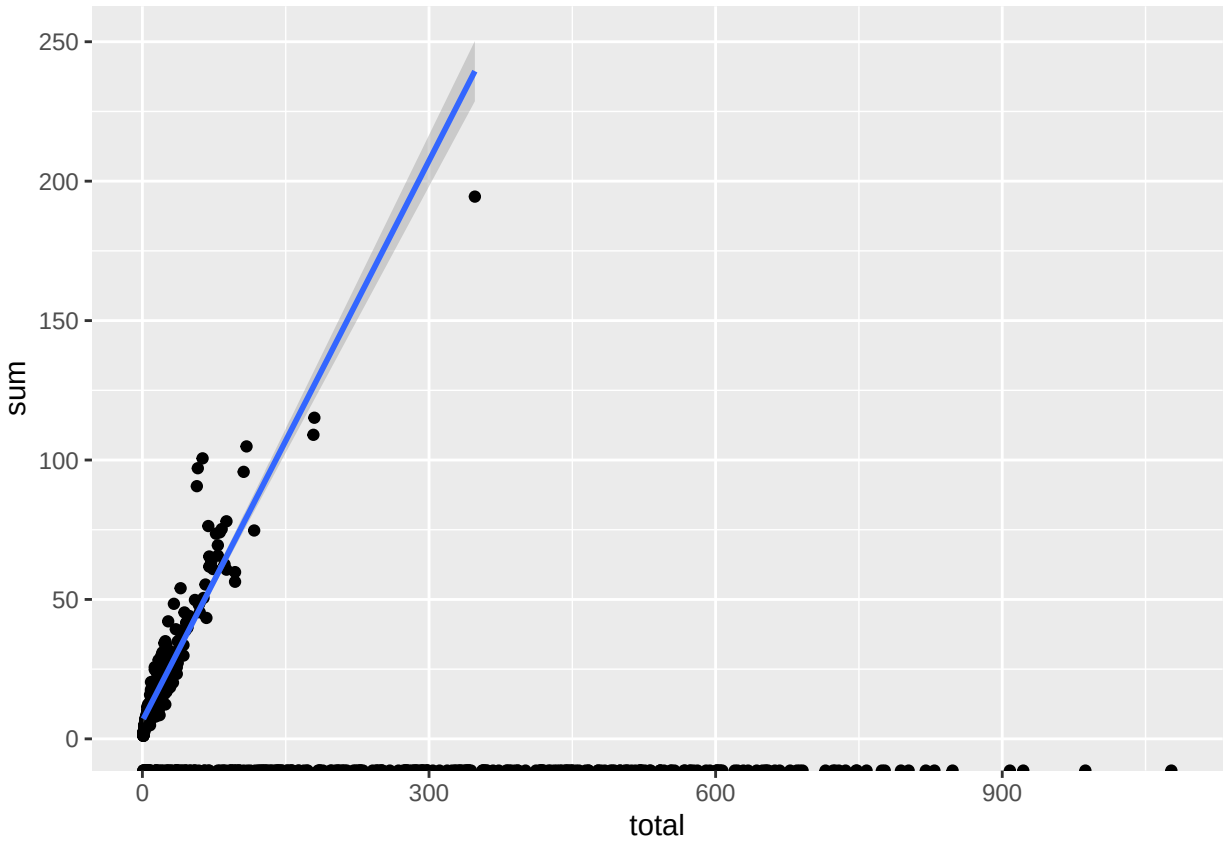
Checking distribution for Agencia_ID and correlation ## Log10.

```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 300 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.9328968
```

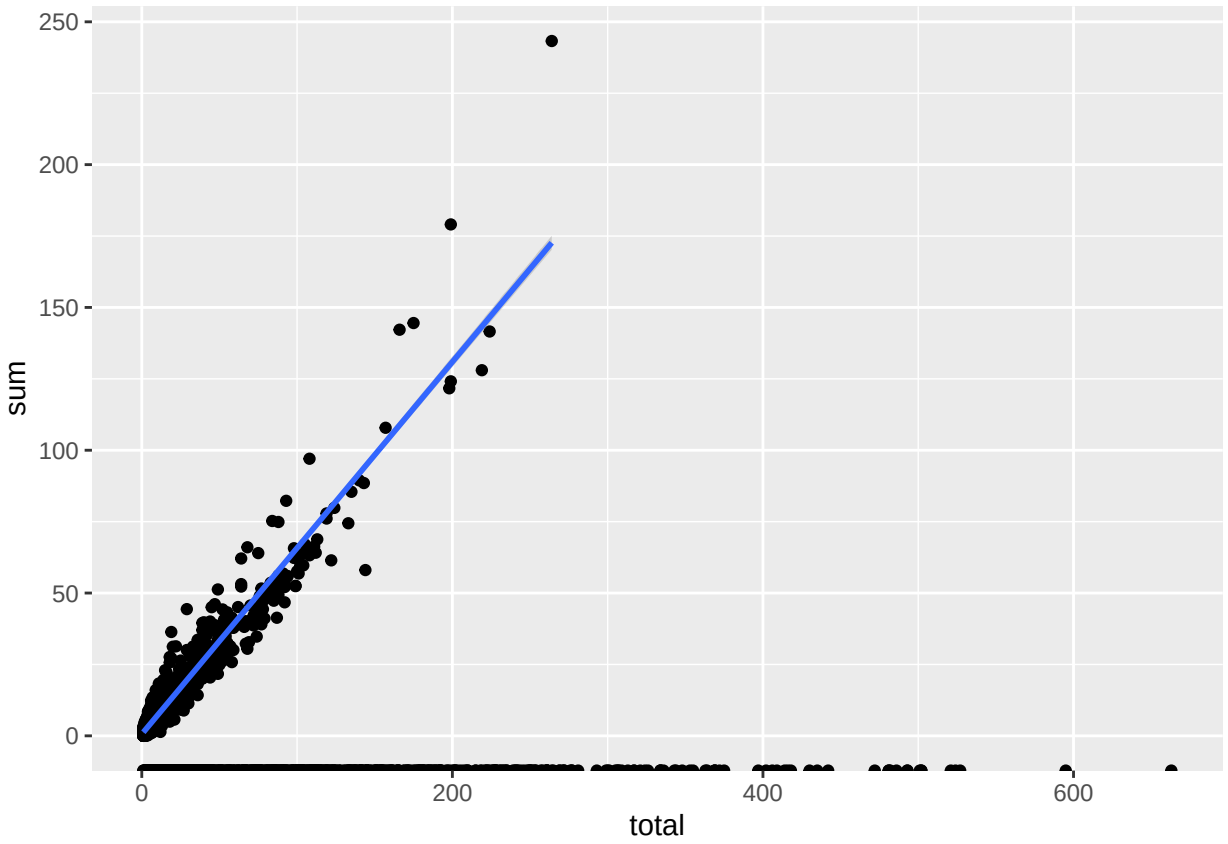
Checking distribution for Ruta_SAK and correlation ## Log10.

```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 650 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.9649856
```

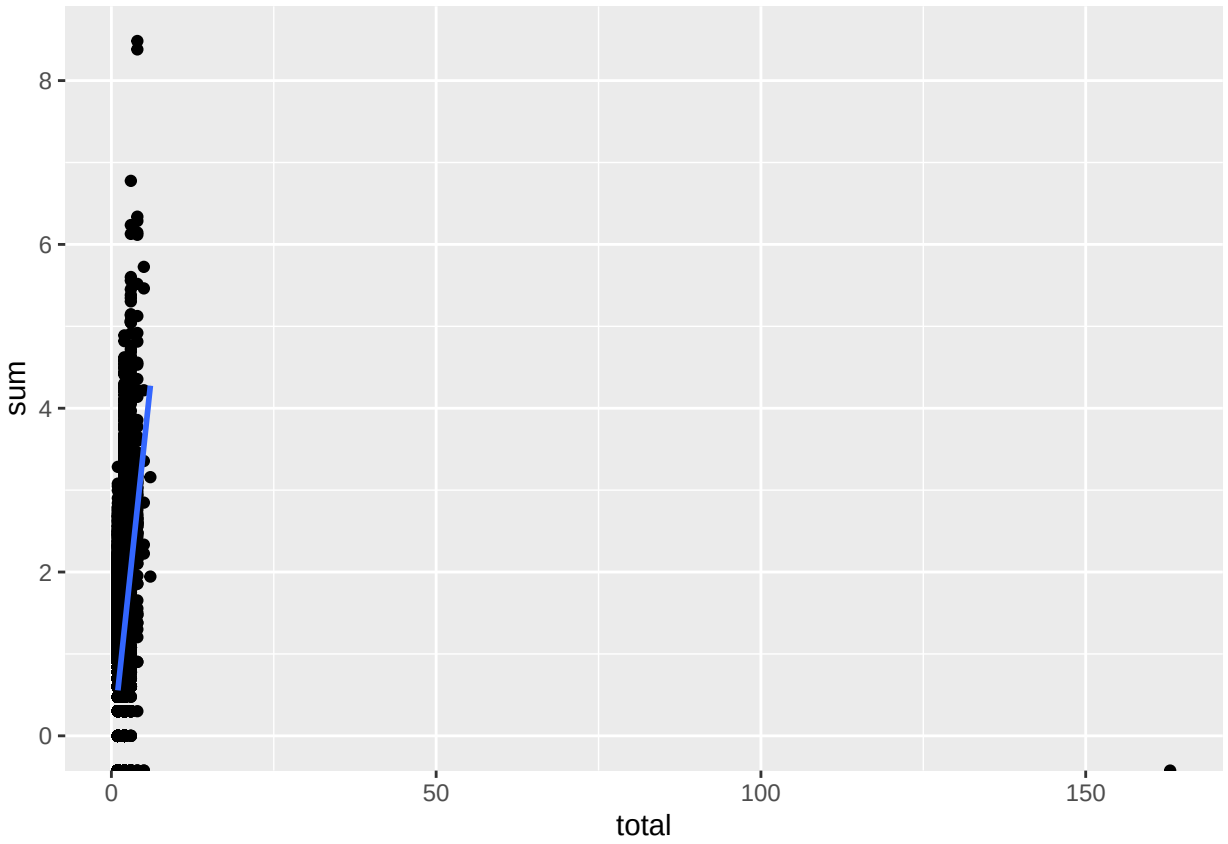
Checking distribution for Cliente_ID and correlation ## Log10.

```
temp = train_sample %>%
  dplyr::group_by(Cliente_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1783 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.4928372
```

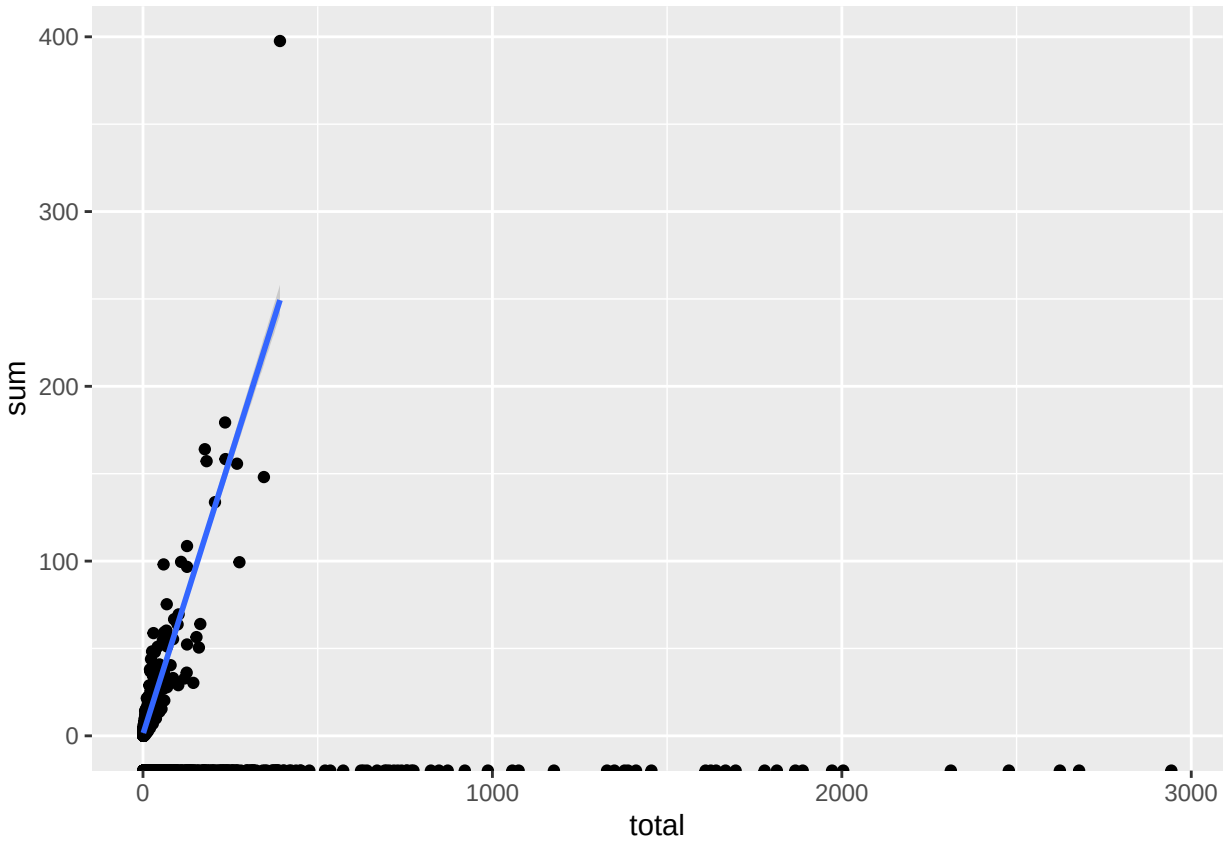
Checking distribution for Producto_ID and correlation ## Log10.

```
temp = train_sample %>%
  dplyr::group_by(Producto_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))

  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 312 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.9070088
```

Checking distribution for Agencia_ID and Ruta_SAK combined and correlation ## Log10.

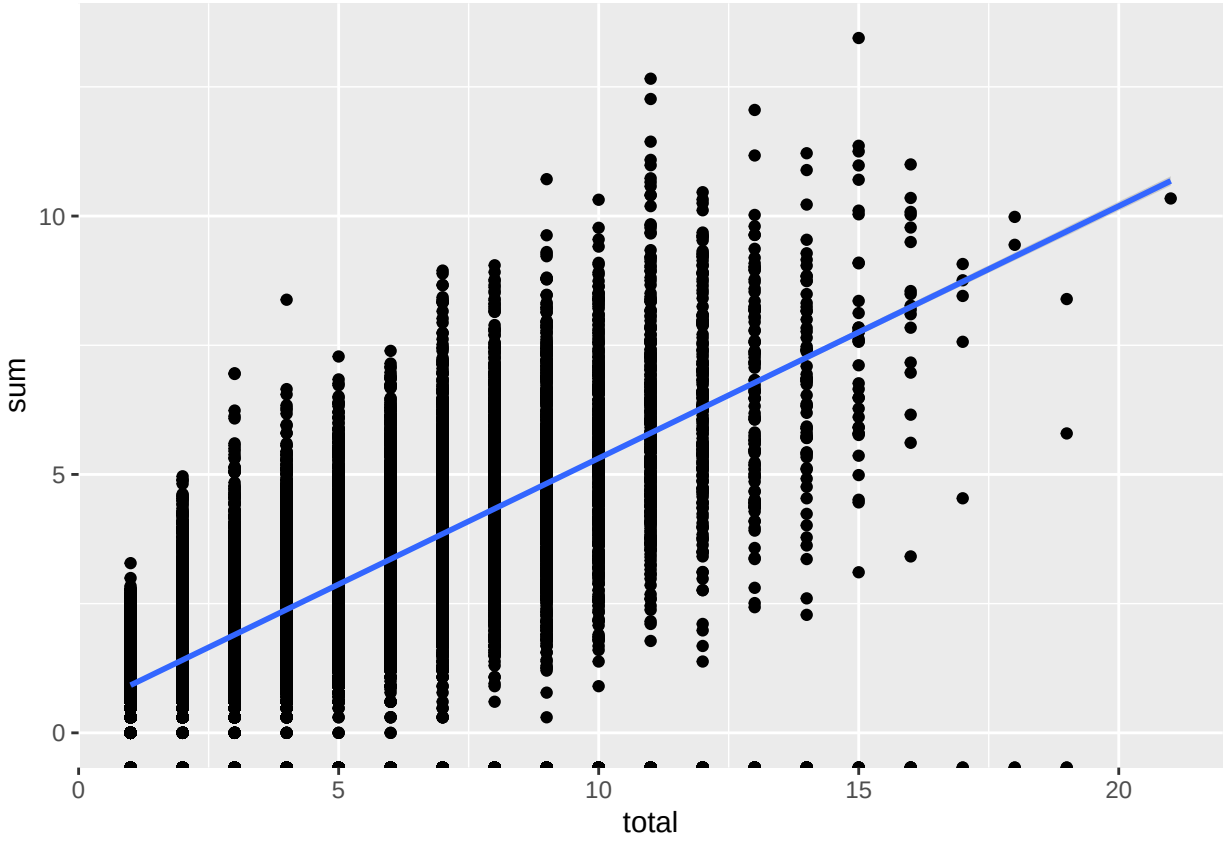
```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Ruta_SAK) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1670 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.7799412
```

Checking distribution for Agencia_ID and Cliente_ID combined and correlation ## Log10.

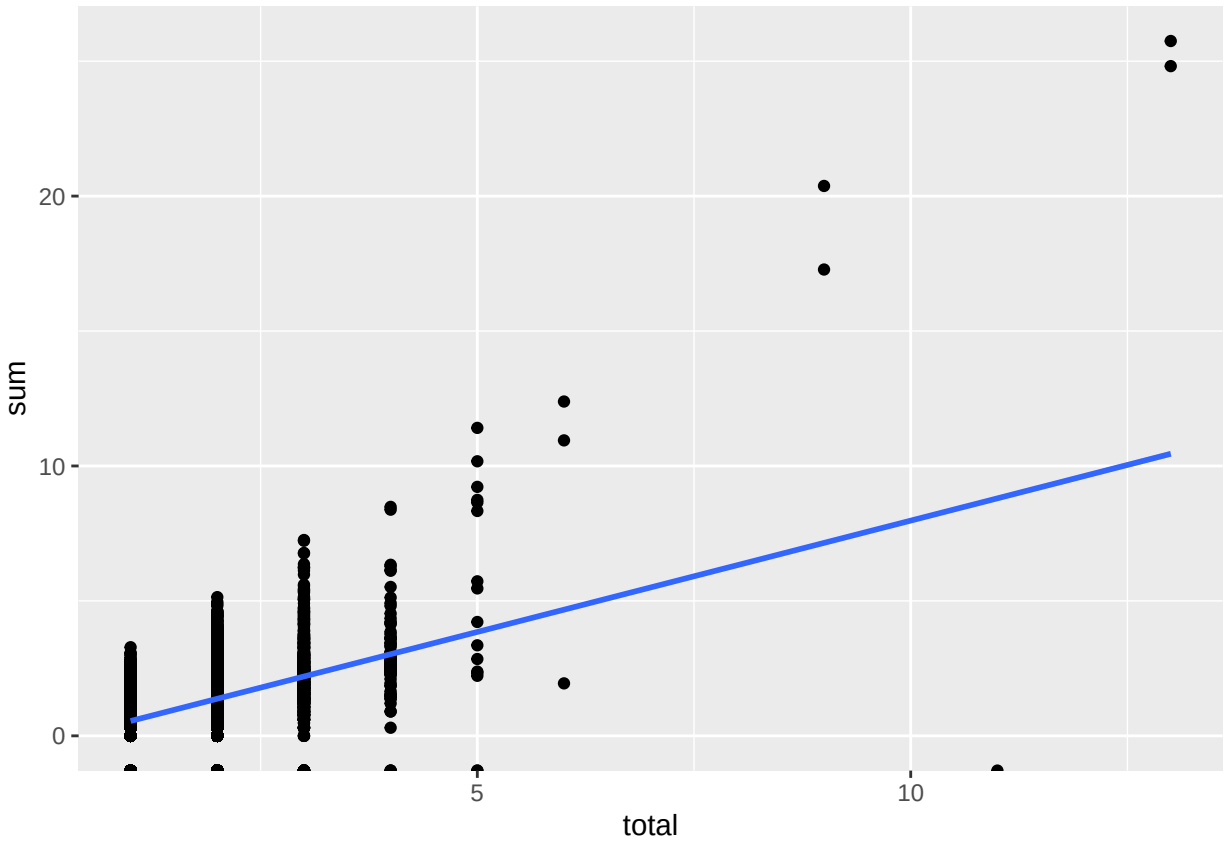
```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Cliente_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1788 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.5209147
```

Checking distribution for Agencia_ID and Producto_ID combined and correlation ## Log10.

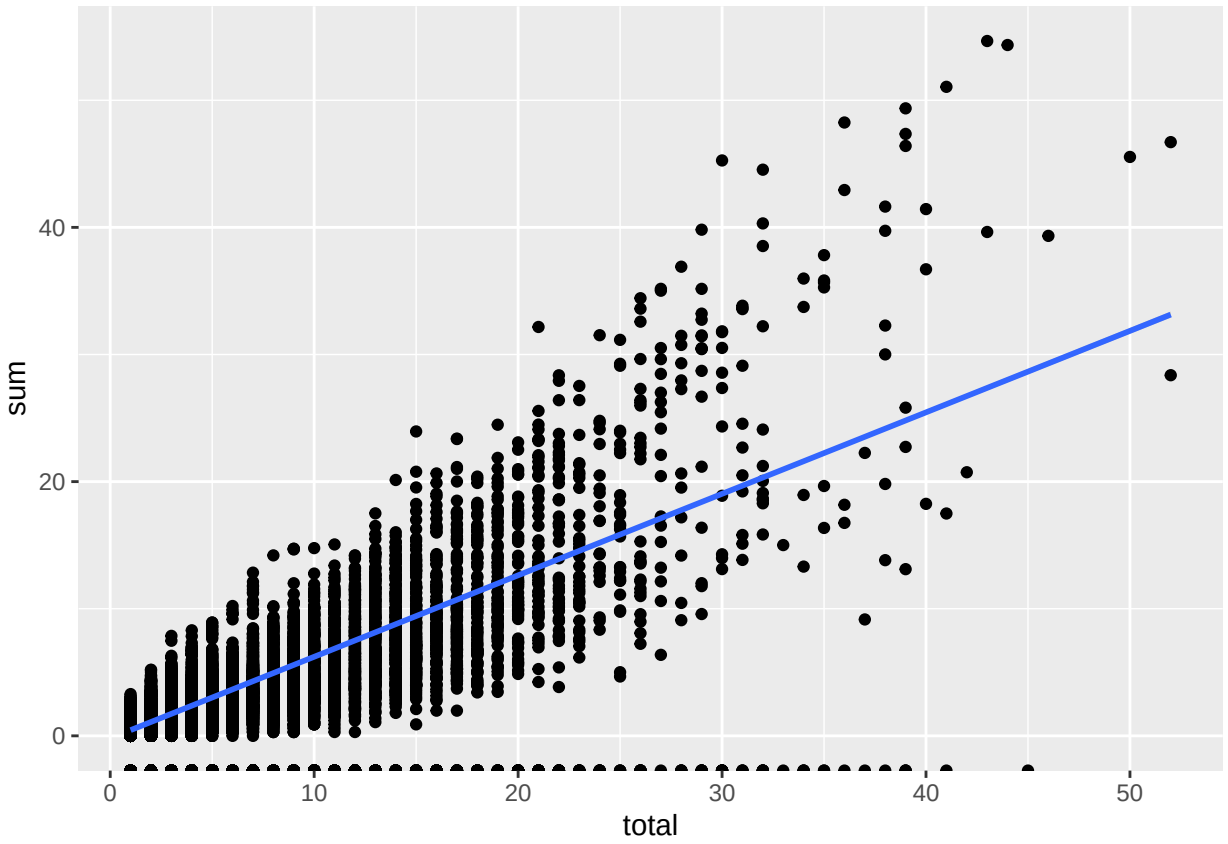
```
temp = train_sample %>%
  dplyr::group_by(Agencia_ID, Producto_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Agencia_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1645 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.8545766
```

Checking distribution for Ruta_SAK and Cliente_ID combined and correlation ## Log10.

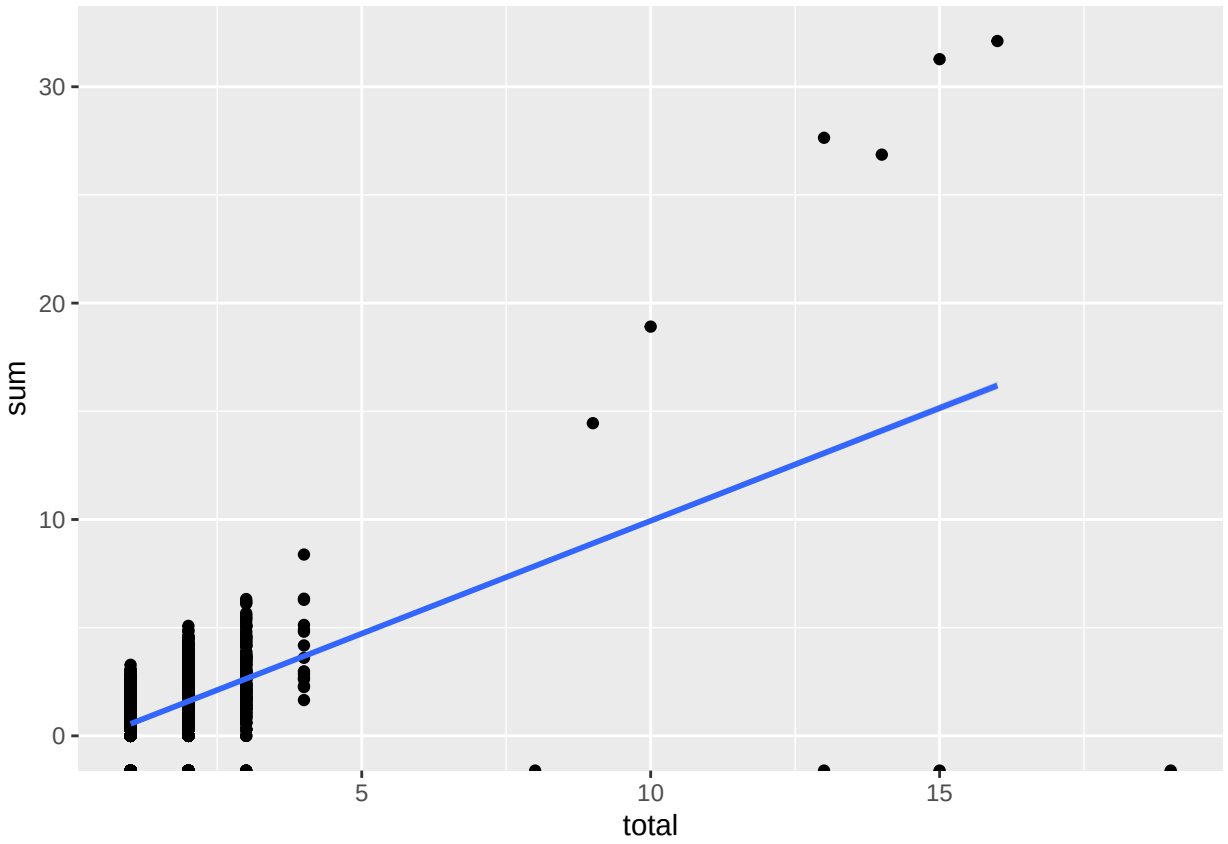
```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK, Cliente_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Ruta_SAK'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1794 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.4723593
```

Checking distribution for Ruta_SAK and Producto_ID combined and correlation ## Log10.

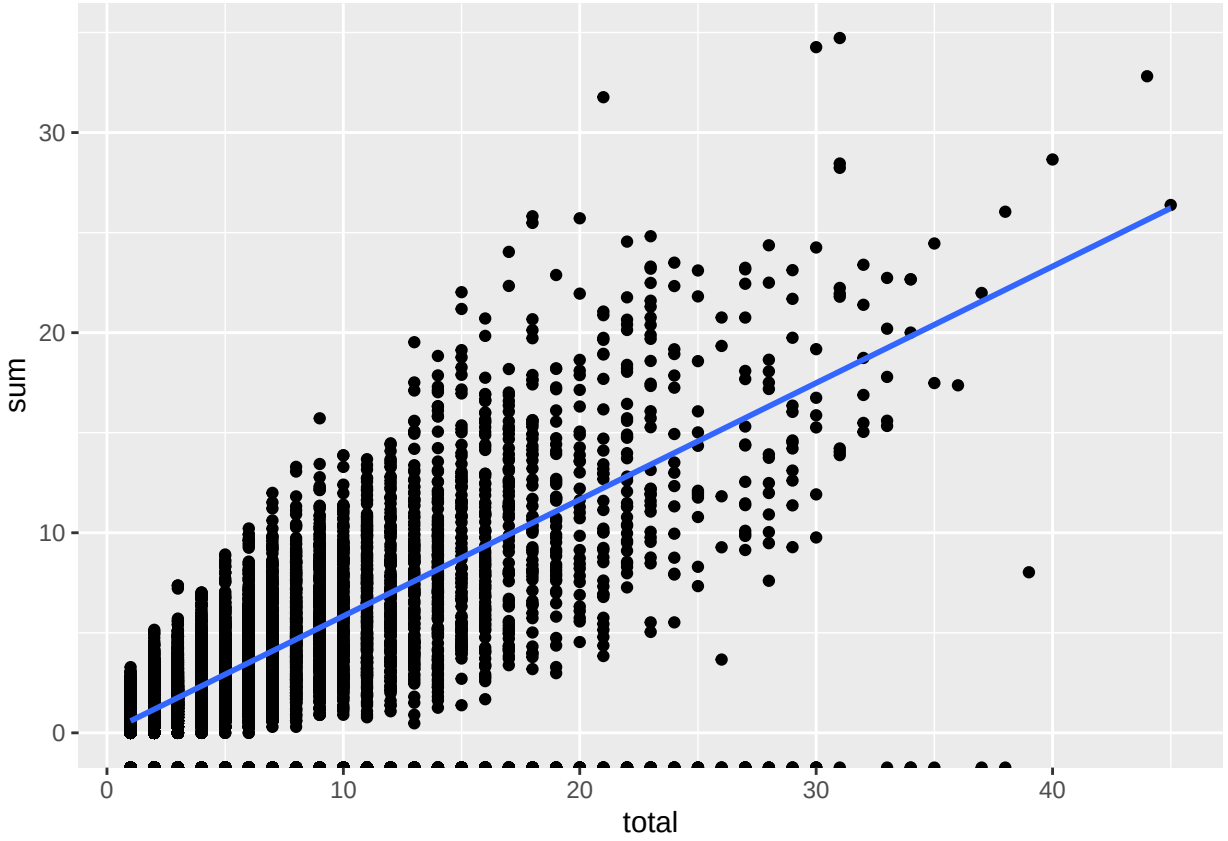
```
temp = train_sample %>%
  dplyr::group_by(Ruta_SAK, Producto_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Ruta_SAK'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1692 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.8477796
```

Checking distribution for Cliente_ID and Producto_ID combined and correlation ## Log10.

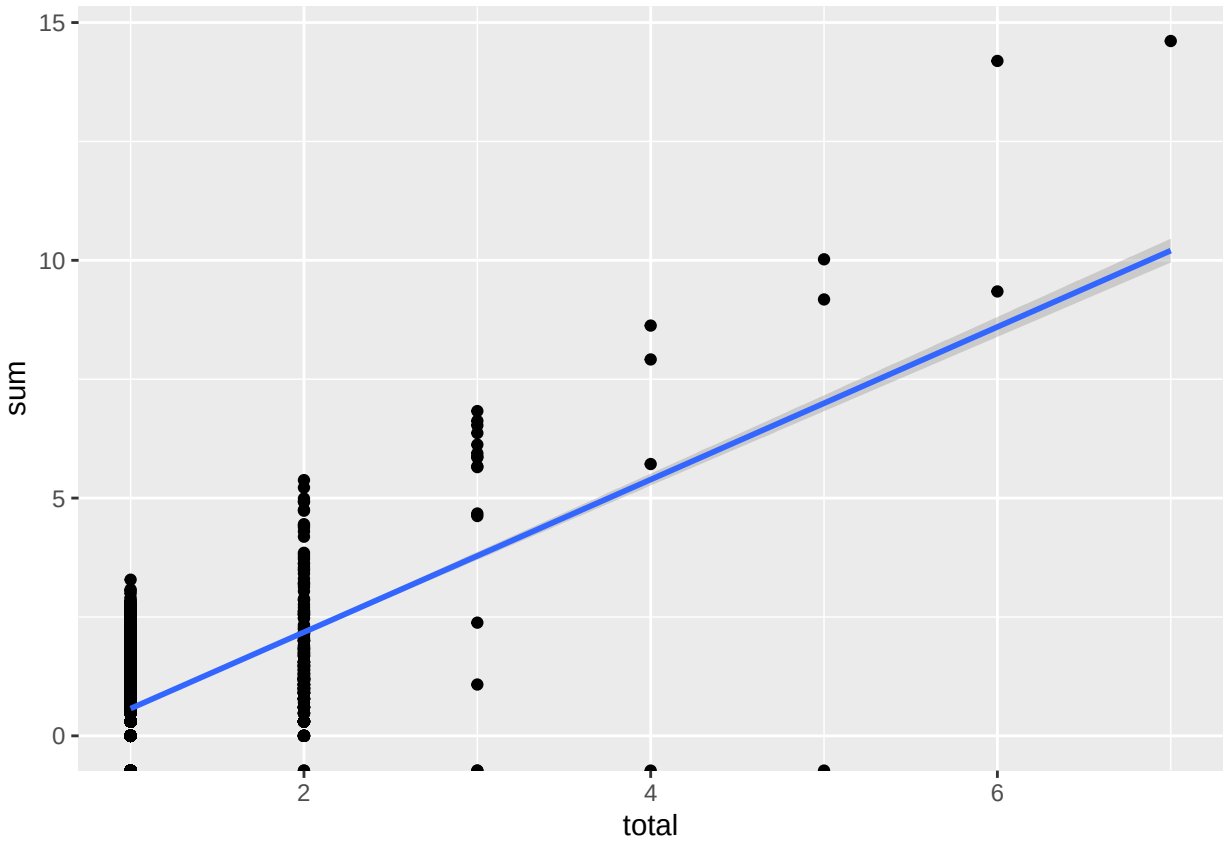
```
temp = train_sample %>%
  dplyr::group_by(Cliente_ID, Producto_ID) %>%
  dplyr::summarise(sum = sum(log10(Demanda_uni_equil)),
                   total = n()) %>%
  dplyr::arrange(desc(sum))
```

```
## `summarise()` has grouped output by 'Cliente_ID'. You can override using the `.groups` argument.
```

```
  ggplot(temp) +
    geom_point(aes(total, sum)) +
    geom_smooth(aes(total, sum), method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1799 rows containing non-finite values (stat_smooth).
```




```
  temp = temp[temp$sum >= 0,]
  cor(temp$total, temp$sum)
```

```
## [1] 0.233954
```

From the analysis above, it is possible to see that the sum of the Demanda_uni_equil is proportional to the total number of times that the Agencia or Producto or Cliente, etc demanded something in the analysed period.

Let's also change the target scale to log10, as it seems to have produced better results.

FEATURE ENGINEERING

Let's create a few new variables that counts the frequency in each category for each specific chosen categorical variable or group of variables.

```
# Adding counts to some variables and group of variables
train_sample = train_sample %>%
  add_count(Semana) %>% dplyr::rename(Semana_n = n) %>%
  add_count(Agencia_ID) %>% dplyr::rename(Agencia_n = n) %>%
  add_count(Cliente_ID) %>% dplyr::rename(Cliente_n = n) %>%
  add_count(Ruta_SAK) %>% dplyr::rename(Ruta_SAK_n = n) %>%
  add_count(Producto_ID) %>% dplyr::rename(Producto_n = n) %>%
  add_count(Ruta_SAK, Cliente_ID) %>% dplyr::rename(Ruta_SAK_Cliente_n = n) %>%
  add_count(Ruta_SAK, Producto_ID) %>% dplyr::rename(Ruta_SAK_Producto_n = n) %>%
  add_count(Agencia_ID, Ruta_SAK) %>% dplyr::rename(Agencia_Ruta_SAK_n = n) %>%
  add_count(Agencia_ID, Cliente_ID) %>% dplyr::rename(Agencia_Cliente_n = n) %>%
  add_count(Agencia_ID, Producto_ID) %>% dplyr::rename(Agencia_Producto_n = n) %>%
```

```
  add_count(Cliente_ID, Producto_ID) %>% dplyr::rename(Cliente_Producto_n = n)
```

EXPLORATORY ANALYSIS II

Checking correlation among new variables and target as well as the unique values among categorical variables.

```
# Checking correlation among numerical variables
cor(train_sample[,c(7:18)])
```

```
##                     Demanda_uni_equil      Semana_n    Agencia_n   Cliente_n
## Demanda_uni_equil         1.000000000 -0.0024691403 -0.130851543  0.25739095
## Semana_n                 -0.002469140  1.0000000000  0.009082372 -0.00275652
## Agencia_n                -0.130851543  0.0090823718  1.000000000 -0.07735890
## Cliente_n                 0.257390948 -0.0027565203 -0.077358899  1.00000000
## Ruta_SAK_n               -0.086058947  0.0002840154 -0.041858069 -0.05387363
## Producto_n               -0.046736333  0.0012613285  0.166863016 -0.04412646
## Ruta_SAK_Cliente_n        0.232348761 -0.0021113130 -0.096328930  0.77167249
## Ruta_SAK_Producto_n      -0.048964997  0.0040709592  0.080402667 -0.03602951
## Agencia_Ruta_SAK_n       -0.107606738 -0.0065279419  0.211286308 -0.06143808
## Agencia_Cliente_n         0.126020477 -0.0019827274 -0.016060734  0.41953612
## Agencia_Producto_n       -0.009802749 -0.0034053513  0.527183673 -0.04413388
## Cliente_Producto_n        0.159990230 -0.0008628696 -0.052703336  0.64991381
##                        Ruta_SAK_n   Producto_n Ruta_SAK_Cliente_n
## Demanda_uni_equil   -0.0860589469 -0.046736333       0.2323487606
## Semana_n             0.0002840154  0.001261329      -0.0021113130
## Agencia_n           -0.0418580686  0.166863016      -0.0963289299
## Cliente_n           -0.0538736261 -0.044126455       0.7716724894
## Ruta_SAK_n           1.0000000000  0.222539892      -0.0450228459
## Producto_n           0.2225398921  1.000000000      -0.0400402824
## Ruta_SAK_Cliente_n  -0.0450228459 -0.040040282       1.0000000000
## Ruta_SAK_Producto_n  0.5411589812  0.616737231      -0.0441399343
## Agencia_Ruta_SAK_n   0.1958330839  0.154909409       0.0001706152
## Agencia_Cliente_n   -0.0572415998 -0.064680557       0.5262158215
## Agencia_Producto_n   0.1085233959  0.580539134      -0.0452767868
## Cliente_Producto_n  -0.0308930700 -0.019800511       0.5624546661
##                     Ruta_SAK_Producto_n Agencia_Ruta_SAK_n Agencia_Cliente_n
## Demanda_uni_equil          -0.048964997      -0.1076067379       0.126020477
## Semana_n                    0.004070959      -0.0065279419      -0.001982727
## Agencia_n                   0.080402667       0.2112863076      -0.016060734
## Cliente_n                  -0.036029512      -0.0614380826       0.419536116
## Ruta_SAK_n                  0.541158981       0.1958330839      -0.057241600
## Producto_n                  0.616737231       0.1549094091      -0.064680557
## Ruta_SAK_Cliente_n         -0.044139934       0.0001706152       0.526215822
## Ruta_SAK_Producto_n         1.000000000       0.1036696110      -0.033456908
## Agencia_Ruta_SAK_n          0.103669611       1.0000000000       0.019497967
## Agencia_Cliente_n          -0.033456908       0.0194979673       1.000000000
## Agencia_Producto_n          0.331410430       0.2286663484      -0.054944948
## Cliente_Producto_n         -0.011770893      -0.0314857361       0.276907862
##                     Agencia_Producto_n Cliente_Producto_n
## Demanda_uni_equil          -0.009802749       0.1599902303
## Semana_n                   -0.003405351      -0.0008628696
## Agencia_n                   0.527183673      -0.0527033356
## Cliente_n                  -0.044133882       0.6499138142
```

```
## Ruta_SAK_n                0.108523396      -0.0308930700
## Producto_n                0.580539134      -0.0198005106
## Ruta_SAK_Cliente_n       -0.045276787       0.5624546661
## Ruta_SAK_Producto_n       0.331410430      -0.0117708931
## Agencia_Ruta_SAK_n        0.228666348      -0.0314857361
## Agencia_Cliente_n        -0.054944948       0.2769078615
## Agencia_Producto_n        1.000000000      -0.0167665114
## Cliente_Producto_n       -0.016766511       1.0000000000
```

```
# Checking unique values
#sapply(colnames(train_sample[,c(1:6)]), function(x) {unique(train_sample[,..x])})

# In order to avoid too many pages, this command was commented during the pdf preparation.
```

Most of categorical variables presents a lot of categories, with exception to Semana and Canal_ID, which presents 7 and 9 levels, respectively.

As the mean values for Demanda_uni_equil for weeks 3 to 9 are very close to each other (uniform distribution of the means), this variable was not selected for modeling.

Although there is a very strong correlation between the sum of demanda_uni_equil and Canal_ID, there is one channel that is kind of an outlier, meaning that most of the demand is attended by this canal and the sum of Demanda_uni_equil is much higher compared to the other channels. This variable may not be useful in the modeling either.

According to the analyses above, let's remove one more variable from our train_sample set, which is the variable Cliente_n as it does not seems to be very useful for the model creation, as it is kind of an id for the data set.

From the correlation analysis let's combine some predictors that are not very correlated to each other, but they are correlated to the target variable. In this case, two sets of predictors were chosen:

- { Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n }
- { Semana_n + Ruta_SAK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + Agencia_Ruta_SAK_n }

Before doing the changes, let's split the data set.

SPLIT DATA INTO TRAIN AND TEST SETS

Let's split the train_sample into train and test.

```
# Creating the index column
train_sample$index = 1:length(train_sample$Semana_n)

# Creating the train_index randomly to split the data
train_index = sample(train_sample$index, 0.7 * length(train_sample$index), replace = FALSE)

# Splitting the data into train and test sets
train = train_sample[train_index,]
test = train_sample[-train_index,]

# Removing index column from train and test sets
train$index = NULL
test$index = NULL
```

Creating a new variable, named Demanda_uni_equil_log, holding the transformed values of Demanda_uni_equil to log (base 10) scale.

```
# Creating the new variable Demanda_uni_equil_log based in log (base 10) scale transformation of the De
train= train %>%
  mutate(Demanda_uni_equil_log = log10(Demanda_uni_equil)) %>%
  select(-Demanda_uni_equil)
```

Standardizing the numeric variables of train_sample, with exception to the target variable. Two train sets were used to test the models, being one using the original data and the other with the standardized data.

```
# Filter only the values higher or equal to 0 to remove a few -Inf values that were created during log
train = train %>% filter(Demanda_uni_equil_log >= 0)
train_std = train
train_std[,c(7:17)] = train_std[,c(7:17)] %>% mutate_if(is.numeric,scale)

# Standardize the numeric variables
test_std = test
test_std[,c(8:18)] = test_std[,c(8:18)] %>% mutate_if(is.numeric,scale)
```

As the data is read for modeling, let's first create four version of linear models, based on the two different set of predictors and the two data sets (0riginal and standardized).

```
## Linear Model

# lm1 - target >> Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, using non-scaled numeric variables
# Training the model with lm algorithm and making predictions
lm1 = lm(Demanda_uni_equil_log ~ Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, data = train)
pred_lm1 = predict(lm1, test[,c("Semana_n", "Cliente_n", "Agencia_n", "Ruta_SAK_n")])

# Calculating the RMSE
rmse_lm1 = RMSE(10**(pred_lm1), test$Demanda_uni_equil)
rmse_lm1
```

```
## [1] 18.37735
```

```
# lm2 - target >> Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, using scaled numeric variables
# Training the model with lm algorithm and making predictions
lm2 = lm(Demanda_uni_equil_log ~ Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, data = train_std)
pred_lm2 = predict(lm2, test_std[,c("Semana_n", "Cliente_n", "Agencia_n", "Ruta_SAK_n")])

# Calculating the RMSE
rmse_lm2 = RMSE(10**(pred_lm2), test_std$Demanda_uni_equil)
rmse_lm2
```

```
## [1] 18.40923
```

```
# lm3 - target >> Semana_n + Ruta_saK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + Agencia_Ruta_SA
# Training the model with lm algorithm and making predictions
lm3 = lm(Demanda_uni_equil_log ~ Semana_n + Ruta_SAK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + A
pred_lm3 = predict(lm3, test[,c("Semana_n", "Ruta_SAK_Cliente_n", "Agencia_n", "Ruta_SAK_n", "Producto_

# Calculating the RMSE
rmse_lm3 = RMSE(pred_lm3, test$Demanda_uni_equil)
rmse_lm3
```

```
## [1] 20.13221
```

```
# lm4 - target >> Semana_n + Ruta_saK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + Agencia_Ruta_SA
# Training the model with lm algorithm and making predictions
lm4 = lm(Demanda_uni_equil_log ~ Semana_n + Ruta_SAK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + A
```

```r
pred_lm4 = predict(lm3, test_std[,c("Semana_n", "Ruta_SAK_Cliente_n", "Agencia_n", "Ruta_SAK_n", "Produ

# Calculating the RMSE
rmse_lm4 = RMSE(10**(pred_lm3), test_std$Demanda_uni_equil)
rmse_lm4
```

```
## [1] 18.60033
```

```r
## XGBoost

# xgb1 - target >> Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, using non-scaled numeric variables
# Data preparation for XGBoost - Creating the y_train1 (target) and dtrain1
y_train1 = train$Demanda_uni_equil_log
dtrain1 = xgb.DMatrix(as.matrix(train %>% select(Semana_n, Cliente_n, Agencia_n, Ruta_SAK_n)), label = y

# Training the model with XGBoost algorithm and making predictions
xgb1 = xgb.train(data = dtrain1, nrounds = 200, max_depth = 8, eta = 0.1)
pred_xgb1 = predict(xgb1, as.matrix(test %>% select(Semana_n, Cliente_n, Agencia_n, Ruta_SAK_n)))

# Calculating the RMSE
rmse_xgb1 = RMSE(pred_xgb1, test$Demanda_uni_equil)
rmse_xgb1
```

```
## [1] 20.10277
```

```r
# xgb2 - target >> Semana_n + Cliente_n + Agencia_n + Ruta_SAK_n, using scaled numeric variables
# Data preparation for XGBoost - Creating the y_train1 (target) and dtrain1
y_train2 = train_std$Demanda_uni_equil_log
dtrain2 = xgb.DMatrix(as.matrix(train_std %>% select(Semana_n, Cliente_n, Agencia_n, Ruta_SAK_n)), label

# Training the model with XGBoost algorithm and making predictions
xgb2 = xgb.train(data = dtrain2, nrounds = 200, max_depth = 8, eta = 0.1)
pred_xgb2 = predict(xgb2, as.matrix(test_std %>% select(Semana_n, Cliente_n, Agencia_n, Ruta_SAK_n)))

# Calculating the RMSE
rmse_xgb2 = RMSE(10**(pred_xgb2), test_std$Demanda_uni_equil)
rmse_xgb2
```

```
## [1] 17.76445
```

```r
# xgb3 - target >> Semana_n + Ruta_saK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + Agencia_Ruta_S
# Data preparation for XGBoost - Creating the y_train1 (target) and dtrain1
y_train3 = train$Demanda_uni_equil_log
dtrain3 = xgb.DMatrix(as.matrix(train %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SAK_n, P

# Training the model with XGBoost algorithm and making predictions
xgb3 = xgb.train(data = dtrain3, nrounds = 200, max_depth = 8, eta = 0.1)
pred_xgb3 = predict(xgb3, as.matrix(test %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SAK_n

# Calculating the RMSE
rmse_xgb3 = RMSE(pred_xgb3, test$Demanda_uni_equil)
rmse_xgb3
```

```
## [1] 20.0519
```

```r
# xgb4 - target >> Semana_n + Ruta_saK_Cliente_n + Agencia_n + Ruta_SAK_n + Producto_n + Agencia_Ruta_S
# Data preparation for XGBoost - Creating the y_train1 (target) and dtrain1
```

```
y_train4 = train_std$Demanda_uni_equil_log
dtrain4 = xgb.DMatrix(as.matrix(train_std %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SAK_

# Training the model with XGBoost algorithm and making predictions
xgb4 = xgb.train(data = dtrain4, nrounds = 200, max_depth = 8, eta = 0.1)
pred_xgb4 = predict(xgb4, as.matrix(test_std %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SA

# Calculating the RMSE
rmse_xgb4 = RMSE(10**(pred_xgb4), test_std$Demanda_uni_equil)
rmse_xgb4
```

```
## [1] 17.1698
```

```
# Free unused memory
invisible(gc())
```

The best result, considering the lowest RMSE, was achieved with the scaled train set and considering the target variable in log (base 10) scale. The chosen model was model xgb4.

Let's create the final version of the model as in xgb4, but using 10 million observations this time.

```
# Sampling 10000000 indexes to build the train_sample
train_sample_index = sample(train_data$index, 10000000, replace = FALSE)

# Creating train_sample with all the registers contained the train_sample_index (Client_IDs)
train_sample = train_data[train_sample_index,]

# Removing the variable index, which was only used for sampling
train_sample$index = NULL

# Transforming categorical variables
train_sample = train_sample %>%
  mutate(Semana = as.factor(Semana),
         Agencia_ID = as.factor(Agencia_ID),
         Canal_ID = as.factor(Canal_ID),
         Ruta_SAK = as.factor(Ruta_SAK),
         Cliente_ID = as.factor(Cliente_ID),
         Producto_ID = as.factor(Producto_ID))

# Adding counts to some variables and group of variables
train_sample = train_sample %>%
  add_count(Semana) %>% dplyr::rename(Semana_n = n) %>%
  add_count(Agencia_ID) %>% dplyr::rename(Agencia_n = n) %>%
  add_count(Cliente_ID) %>% dplyr::rename(Cliente_n = n) %>%
  add_count(Ruta_SAK) %>% dplyr::rename(Ruta_SAK_n = n) %>%
  add_count(Producto_ID) %>% dplyr::rename(Producto_n = n) %>%
  add_count(Ruta_SAK, Cliente_ID) %>% dplyr::rename(Ruta_SAK_Cliente_n = n) %>%
  add_count(Agencia_ID, Ruta_SAK) %>% dplyr::rename(Agencia_Ruta_SAK_n = n)

# Creating the index column
train_sample$index = 1:length(train_sample$Semana_n)

# Creating the train_index randomly to split the data
train_index = sample(train_sample$index, 0.7 * length(train_sample$index), replace = FALSE)
```

```r
# Splitting the data into train and test sets
train = train_sample[train_index,]
test = train_sample[-train_index,]

# Removing index column from train and test sets
train$index = NULL
test$index = NULL

# Creating the new variable Demanda_uni_equil_log based in log (base 10) scale transformation of the De
train = train %>%
  mutate(Demanda_uni_equil_log = log10(Demanda_uni_equil)) %>%
  select(-Demanda_uni_equil)

# Filter only the values higher or equal to 0 to remove a few -Inf values that were created during log
train = train %>% filter(Demanda_uni_equil_log >= 0)
train_std = train
train_std[,c(7:13)] = train_std[,c(7:13)] %>% mutate_if(is.numeric,scale)

# Standardize the numeric variables
test_std = test
test_std[,c(7:13)] = test_std[,c(7:13)] %>% mutate_if(is.numeric,scale)

# Creating the y_train1 (target) and dtrain1 for XGBoost algorithm
y_train = train_std$Demanda_uni_equil_log
dtrain = xgb.DMatrix(as.matrix(train_std %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SAK_n

# Training the model with XGBoost algorithm and making predictions
xgb = xgb.train(data = dtrain, nrounds = 25, max_depth = 8, eta = 0.5)
pred_xgb = predict(xgb, as.matrix(test_std %>% select(Semana_n, Ruta_SAK_Cliente_n, Agencia_n, Ruta_SAK_

# Calculating the RMSE
rmse_xgb = RMSE(10**(pred_xgb), test_std$Demanda_uni_equil)
rmse_xgb
```

```
## [1] 4.642203
```