André Campos da Silva

06 de Abril, 2021

Projeto - Modelagem Preditiva em IoT - Previsão de Uso de Energia

Este projeto de IoT tem como objetivo a criação de modelos preditivos para a previsão de consumo de energia de eletrodomésticos. Os dados utilizados incluem medições de sensores de temperatura e umidade de uma rede sem fio, previsão do tempo de uma estação de um aeroporto e uso de energia utilizada por luminárias.

Cada nó sem fio transmitia as condições de temperatura e umidade em torno de 3 min. Em seguida, a média dos dados foi calculada para períodos de 10 minutos. Os dados de energia foram registrados a cada 10 minutos com medidores de energia de barramento m.

Este Projeto visa construir um modelo preditivo que possa prever o consumo de energia com base nos dados de sensores IoT coletados.

Dcionario dos dados

date: Tempo de coleta dos dados pelos sensores.

Appliances: Uso de energia (em W).

lights: Potência de energia de eletrodomésticos na casa (em W).

TX: Temperatura em um lugar da casa (em Celsius).

RH_X: Umidade relativa em algum ponto da casa (em %).

Press_mm_hg: Não foi informado.

Windspeed: Velocidade do vento (em m/s).

Visibility: Visibilidade (em Km).

Tdewpoint: Não foi informado.

rv1: Variável randômica adicional.

rv2: Variável randômica adicional.

WeekStatus: Indica se é dia de semana ou final de semana.

Day_of_week: Dia da semana.

NSM: Medida de tempo (em s).

Carregando pacotes

```
# Instalando os pacotes

#install.packages('tidyverse')
#install.packages('caret')
#install.packages('ROSE')
#install.packages('data.table')
#install.packages('gridExtra')
#install.packages('randomForest')
#install.packages('DMwR')
```

```
#install.packages('e1071')
#install.packages('rpart')
#install.packages('caTools')
#install.packages('kernlab')
#install.packages('xgboost')

# Carregando pacotes

library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library('caret')
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library('ROSE')
```

```
## Loaded ROSE 0.0-3
```

```
library('data.table')
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library('gridExtra')
```

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

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

```
library('randomForest')
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

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

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library('DMwR')
```

```
## Loading required package: grid

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

```
library('e1071')
library('rpart')
library('caTools')
library('corrplot')
```

```
## corrplot 0.84 loaded
```

```
library('kernlab')
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library('xgboost')
```

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

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

Carregando os Dados

```
# Carrego os datasets para análise.

df_train <- read_csv('Dados/projeto8-training.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   date = col_datetime(format = ""),
##   WeekStatus = col_character(),
##   Day_of_week = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
df_train <- as.data.frame(df_train)
df_test <- read_csv('Dados/projeto8-testing.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   date = col_datetime(format = ""),
##   WeekStatus = col_character(),
##   Day_of_week = col_character()
## )
## i Use `spec()` for the full column specifications.
```

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

# Imprimo as primeiras linhas
head(df_train)
```

```
##                  date Appliances lights       T1     RH_1   T2     RH_2    T3
## 1 2016-01-11 17:00:00         60     30 19.89000 47.59667 19.2 44.79000 19.79
## 2 2016-01-11 17:10:00         60     30 19.89000 46.69333 19.2 44.72250 19.79
## 3 2016-01-11 17:20:00         50     30 19.89000 46.30000 19.2 44.62667 19.79
## 4 2016-01-11 17:40:00         60     40 19.89000 46.33333 19.2 44.53000 19.79
## 5 2016-01-11 17:50:00         50     40 19.89000 46.02667 19.2 44.50000 19.79
## 6 2016-01-11 18:10:00         60     50 19.85667 45.56000 19.2 44.50000 19.73
##       RH_3       T4     RH_4       T5  RH_5       T6     RH_6       T7     RH_7
## 1 44.73000 19.00000 45.56667 17.16667 55.20 7.026667 84.25667 17.20000 41.62667
## 2 44.79000 19.00000 45.99250 17.16667 55.20 6.833333 84.06333 17.20000 41.56000
## 3 44.93333 18.92667 45.89000 17.16667 55.09 6.560000 83.15667 17.20000 41.43333
## 4 45.00000 18.89000 45.53000 17.20000 55.09 6.366667 84.89333 17.20000 41.23000
## 5 44.93333 18.89000 45.73000 17.13333 55.03 6.300000 85.76667 17.13333 41.26000
## 6 44.90000 18.89000 45.86333 17.10000 54.90 6.190000 86.42333 17.10000 41.20000
##     T8     RH_8       T9  RH_9    T_out Press_mm_hg   RH_out Windspeed
## 1 18.2 48.90000 17.03333 45.53 6.600000    733.5000 92.00000  7.000000
## 2 18.2 48.86333 17.06667 45.56 6.483333    733.6000 92.00000  6.666667
## 3 18.2 48.73000 17.00000 45.50 6.366667    733.7000 92.00000  6.333333
## 4 18.1 48.59000 17.00000 45.40 6.133333    733.9000 92.00000  5.666667
## 5 18.1 48.59000 17.00000 45.29 6.016667    734.0000 92.00000  5.333333
## 6 18.1 48.59000 17.00000 45.29 5.916667    734.1667 91.83333  5.166667
##   Visibility Tdewpoint      rv1      rv2   NSM WeekStatus Day_of_week
## 1   63.00000  5.300000 13.27543 13.27543 61200    Weekday      Monday
## 2   59.16667  5.200000 18.60619 18.60619 61800    Weekday      Monday
## 3   55.33333  5.100000 28.64267 28.64267 62400    Weekday      Monday
## 4   47.66667  4.900000 10.08410 10.08410 63600    Weekday      Monday
## 5   43.83333  4.800000 44.91948 44.91948 64200    Weekday      Monday
## 6   40.00000  4.683333 33.03989 33.03989 65400    Weekday      Monday
```

```
# Imprimo as primeiras linhas
head(df_test)
```

```
##                  date Appliances lights       T1     RH_1       T2     RH_2
## 1 2016-01-11 17:30:00         50     40 19.89000 46.06667 19.20000 44.59000
## 2 2016-01-11 18:00:00         60     50 19.89000 45.76667 19.20000 44.50000
## 3 2016-01-11 18:40:00        230     70 19.92667 45.86333 19.35667 44.40000
## 4 2016-01-11 18:50:00        580     60 20.06667 46.39667 19.42667 44.40000
## 5 2016-01-11 19:30:00        100     10 20.56667 53.89333 20.03333 46.75667
## 6 2016-01-11 19:50:00         70     30 20.85667 53.66000 20.20000 47.05667
##      T3     RH_3    T4     RH_4       T5     RH_5       T6     RH_6       T7
## 1 19.79 45.00000 18.89 45.72333 17.16667 55.09000 6.433333 83.42333 17.13333
## 2 19.79 44.90000 18.89 45.79000 17.10000 54.96667 6.263333 86.09000 17.13333
## 3 19.79 44.90000 18.89 46.43000 17.10000 55.00000 6.190000 87.86667 17.24750
## 4 19.79 44.82667 19.00 46.43000 17.10000 55.00000 6.123333 87.99333 17.53000
## 5 20.10 48.46667 19.00 48.49000 17.15000 56.04250 5.800000 88.36667 17.89000
## 6 20.20 48.44750 18.89 47.96333 17.20000 56.93333 5.526667 87.30000 17.70000
##       RH_7       T8     RH_8    T9     RH_9    T_out Press_mm_hg   RH_out
## 1 41.29000 18.10000 48.59000 17.00 45.40000 6.250000    733.8000 92.00000
## 2 41.20000 18.10000 48.59000 17.00 45.29000 5.900000    734.1000 92.00000
## 3 42.71750 18.10000 48.59000 17.00 45.29000 5.966667    734.3667 91.33333
## 4 44.26333 18.06667 48.63333 16.89 45.29000 5.983333    734.4333 91.16667
## 5 44.92667 18.15000 49.20000 16.89 45.32667 6.000000    734.8500 89.50000
## 6 43.72667 18.35667 50.02667 16.89 45.29000 6.000000    735.0833 88.50000
##   Windspeed Visibility Tdewpoint       rv1       rv2   NSM WeekStatus
```

```
## 1  6.000000       51.5  5.000000 45.410389 45.410389 63000    Weekday
## 2  5.000000       40.0  4.700000 47.233763 47.233763 64800    Weekday
## 3  5.666667       40.0  4.633333 10.298729 10.298729 67200    Weekday
## 4  5.833333       40.0  4.616667  8.827838  8.827838 67800    Weekday
## 5  6.000000       40.0  4.350000 24.884962 24.884962 70200    Weekday
## 6  6.000000       40.0  4.183333 49.595305 49.595305 71400    Weekday
##   Day_of_week
## 1      Monday
## 2      Monday
## 3      Monday
## 4      Monday
## 5      Monday
## 6      Monday
```

Análise Exploratória de Dados

```
# Crio novas variáveis(mês,dia, hora e minuto) que farão parte da analise e seleção de variáveis.
df_train$Month <- sapply(df_train$date, month)
#df_train$Month <- as.factor(df_train$Month)

df_train$Day <- sapply(df_train$date, mday)
#df_train$Day <- as.factor(df_train$Day)

df_train$Hour <- sapply(df_train$date,hour )
#df_train$Hour <-as.factor(df_train$Hour)

df_train$Minute <- sapply(df_train$date, minute)
#df_train$Minu <-as.factor(df_train$Minu)

# Verifico os formatos dos dados
glimpse(df_train)
```

```
## Rows: 14,803
## Columns: 36
## $ date        <dttm> 2016-01-11 17:00:00, 2016-01-11 17:10:00, 2016-01-11 1...
## $ Appliances  <dbl> 60, 60, 50, 60, 50, 60, 60, 70, 430, 250, 100, 90, 80, ...
## $ lights      <dbl> 30, 30, 30, 40, 40, 50, 40, 40, 50, 40, 10, 10, 30, 40,...
## $ T1          <dbl> 19.89000, 19.89000, 19.89000, 19.89000, 19.89000, 19.85...
## $ RH_1        <dbl> 47.59667, 46.69333, 46.30000, 46.33333, 46.02667, 45.56...
## $ T2          <dbl> 19.20000, 19.20000, 19.20000, 19.20000, 19.20000, 19.20...
## $ RH_2        <dbl> 44.79000, 44.72250, 44.62667, 44.53000, 44.50000, 44.50...
## $ T3          <dbl> 19.79000, 19.79000, 19.79000, 19.79000, 19.79000, 19.73...
## $ RH_3        <dbl> 44.73000, 44.79000, 44.93333, 45.00000, 44.93333, 44.90...
## $ T4          <dbl> 19.00000, 19.00000, 18.92667, 18.89000, 18.89000, 18.89...
## $ RH_4        <dbl> 45.56667, 45.99250, 45.89000, 45.53000, 45.73000, 45.86...
## $ T5          <dbl> 17.16667, 17.16667, 17.16667, 17.20000, 17.13333, 17.10...
## $ RH_5        <dbl> 55.20000, 55.20000, 55.09000, 55.09000, 55.03000, 54.90...
## $ T6          <dbl> 7.026667, 6.833333, 6.560000, 6.366667, 6.300000, 6.190...
## $ RH_6        <dbl> 84.25667, 84.06333, 83.15667, 84.89333, 85.76667, 86.42...
## $ T7          <dbl> 17.20000, 17.20000, 17.20000, 17.20000, 17.13333, 17.10...
## $ RH_7        <dbl> 41.62667, 41.56000, 41.43333, 41.23000, 41.26000, 41.20...
## $ T8          <dbl> 18.20000, 18.20000, 18.20000, 18.10000, 18.10000, 18.10...
```

```
## $ RH_8        <dbl> 48.90000, 48.86333, 48.73000, 48.59000, 48.59000, 48.59...
## $ T9          <dbl> 17.03333, 17.06667, 17.00000, 17.00000, 17.00000, 17.00...
## $ RH_9        <dbl> 45.53000, 45.56000, 45.50000, 45.40000, 45.29000, 45.29...
## $ T_out       <dbl> 6.600000, 6.483333, 6.366667, 6.133333, 6.016667, 5.916...
## $ Press_mm_hg <dbl> 733.5000, 733.6000, 733.7000, 733.9000, 734.0000, 734.1...
## $ RH_out      <dbl> 92.00000, 92.00000, 92.00000, 92.00000, 92.00000, 91.83...
## $ Windspeed   <dbl> 7.000000, 6.666667, 6.333333, 5.666667, 5.333333, 5.166...
## $ Visibility  <dbl> 63.00000, 59.16667, 55.33333, 47.66667, 43.83333, 40.00...
## $ Tdewpoint   <dbl> 5.300000, 5.200000, 5.100000, 4.900000, 4.800000, 4.683...
## $ rv1         <dbl> 13.2754332, 18.6061950, 28.6426682, 10.0840966, 44.9194...
## $ rv2         <dbl> 13.2754332, 18.6061950, 28.6426682, 10.0840966, 44.9194...
## $ NSM         <dbl> 61200, 61800, 62400, 63600, 64200, 65400, 66000, 66600,...
## $ WeekStatus  <chr> "Weekday", "Weekday", "Weekday", "Weekday", "Weekday", ...
## $ Day_of_week <chr> "Monday", "Monday", "Monday", "Monday", "Monday", "Mond...
## $ Month       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Day         <int> 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,...
## $ Hour        <int> 17, 17, 17, 17, 17, 18, 18, 18, 19, 19, 19, 19, 20, 20,...
## $ Minute      <int> 0, 10, 20, 40, 50, 10, 20, 30, 0, 10, 20, 40, 0, 10, 20...
```

```
# Faço um resumo dos dados
summary(df_train)
```

```
##       date                          Appliances          lights
##  Min.   :2016-01-11 17:00:00   Min.   :  10.00   Min.   : 0.000
##  1st Qu.:2016-02-14 22:55:00   1st Qu.:  50.00   1st Qu.: 0.000
##  Median :2016-03-19 20:20:00   Median :  60.00   Median : 0.000
##  Mean   :2016-03-20 02:37:28   Mean   :  98.01   Mean   : 3.803
##  3rd Qu.:2016-04-23 06:55:00   3rd Qu.: 100.00   3rd Qu.: 0.000
##  Max.   :2016-05-27 18:00:00   Max.   :1080.00   Max.   :50.000
##        T1              RH_1             T2              RH_2
##  Min.   :16.79   Min.   :27.02   Min.   :16.10   Min.   :20.89
##  1st Qu.:20.73   1st Qu.:37.36   1st Qu.:18.82   1st Qu.:37.90
##  Median :21.60   Median :39.66   Median :20.00   Median :40.50
##  Mean   :21.68   Mean   :40.27   Mean   :20.34   Mean   :40.42
##  3rd Qu.:22.60   3rd Qu.:43.09   3rd Qu.:21.50   3rd Qu.:43.29
##  Max.   :26.26   Max.   :63.36   Max.   :29.86   Max.   :56.03
##        T3              RH_3             T4              RH_4
##  Min.   :17.20   Min.   :28.77   Min.   :15.10   Min.   :27.66
##  1st Qu.:20.79   1st Qu.:36.90   1st Qu.:19.50   1st Qu.:35.53
##  Median :22.10   Median :38.53   Median :20.67   Median :38.40
##  Mean   :22.26   Mean   :39.25   Mean   :20.86   Mean   :39.03
##  3rd Qu.:23.29   3rd Qu.:41.76   3rd Qu.:22.10   3rd Qu.:42.13
##  Max.   :29.24   Max.   :50.16   Max.   :26.20   Max.   :51.06
##        T5              RH_5             T6              RH_6
##  Min.   :15.33   Min.   :29.86   Min.   :-6.065   Min.   : 1.00
##  1st Qu.:18.27   1st Qu.:45.40   1st Qu.: 3.657   1st Qu.:30.10
##  Median :19.39   Median :49.09   Median : 7.295   Median :55.30
##  Mean   :19.59   Mean   :50.96   Mean   : 7.921   Mean   :54.62
##  3rd Qu.:20.60   3rd Qu.:53.66   3rd Qu.:11.245   3rd Qu.:83.33
##  Max.   :25.75   Max.   :95.95   Max.   :28.290   Max.   :99.90
##        T7              RH_7             T8              RH_8
##  Min.   :15.39   Min.   :23.20   Min.   :16.31   Min.   :29.60
##  1st Qu.:18.70   1st Qu.:31.50   1st Qu.:20.79   1st Qu.:39.06
##  Median :20.03   Median :34.82   Median :22.13   Median :42.36
```

```
##  Mean   :20.26   Mean   :35.39   Mean   :22.03   Mean   :42.92
##  3rd Qu.:21.60   3rd Qu.:39.00   3rd Qu.:23.39   3rd Qu.:46.56
##  Max.   :26.00   Max.   :51.40   Max.   :27.23   Max.   :58.78
##        T9              RH_9           T_out         Press_mm_hg
##  Min.   :14.89   Min.   :29.17   Min.   :-5.000   Min.   :729.3
##  1st Qu.:18.00   1st Qu.:38.50   1st Qu.: 3.667   1st Qu.:750.9
##  Median :19.39   Median :40.86   Median : 6.900   Median :756.1
##  Mean   :19.48   Mean   :41.54   Mean   : 7.413   Mean   :755.5
##  3rd Qu.:20.60   3rd Qu.:44.36   3rd Qu.:10.400   3rd Qu.:760.9
##  Max.   :24.50   Max.   :53.33   Max.   :25.967   Max.   :772.3
##      RH_out         Windspeed        Visibility      Tdewpoint
##  Min.   : 24.00   Min.   : 0.000   Min.   : 1.00   Min.   :-6.600
##  1st Qu.: 70.00   1st Qu.: 2.000   1st Qu.:29.00   1st Qu.: 0.900
##  Median : 83.67   Median : 3.667   Median :40.00   Median : 3.450
##  Mean   : 79.73   Mean   : 4.034   Mean   :38.33   Mean   : 3.757
##  3rd Qu.: 91.67   3rd Qu.: 5.500   3rd Qu.:40.00   3rd Qu.: 6.533
##  Max.   :100.00   Max.   :13.500   Max.   :66.00   Max.   :15.500
##       rv1                rv2                NSM         WeekStatus
##  Min.   : 0.00532   Min.   : 0.00532   Min.   :    0   Length:14803
##  1st Qu.:12.58042   1st Qu.:12.58042   1st Qu.:21600   Class :character
##  Median :25.04399   Median :25.04399   Median :43200   Mode  :character
##  Mean   :25.07809   Mean   :25.07809   Mean   :42986
##  3rd Qu.:37.66591   3rd Qu.:37.66591   3rd Qu.:64800
##  Max.   :49.99653   Max.   :49.99653   Max.   :85800
##  Day_of_week            Month             Day              Hour
##  Length:14803       Min.   :1.000   Min.   : 1.00   Min.   : 0.00
##  Class :character   1st Qu.:2.000   1st Qu.: 9.00   1st Qu.: 6.00
##  Mode  :character   Median :3.000   Median :16.00   Median :12.00
##                     Mean   :3.097   Mean   :16.08   Mean   :11.52
##                     3rd Qu.:4.000   3rd Qu.:23.00   3rd Qu.:18.00
##                     Max.   :5.000   Max.   :31.00   Max.   :23.00
##      Minute
##  Min.   : 0.00
##  1st Qu.:10.00
##  Median :20.00
##  Mean   :24.99
##  3rd Qu.:40.00
##  Max.   :50.00
```

```
# Verifico se existe valores nulos nos dados
sum(is.na(df_train))
```

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

Analise em grafico de cada varável

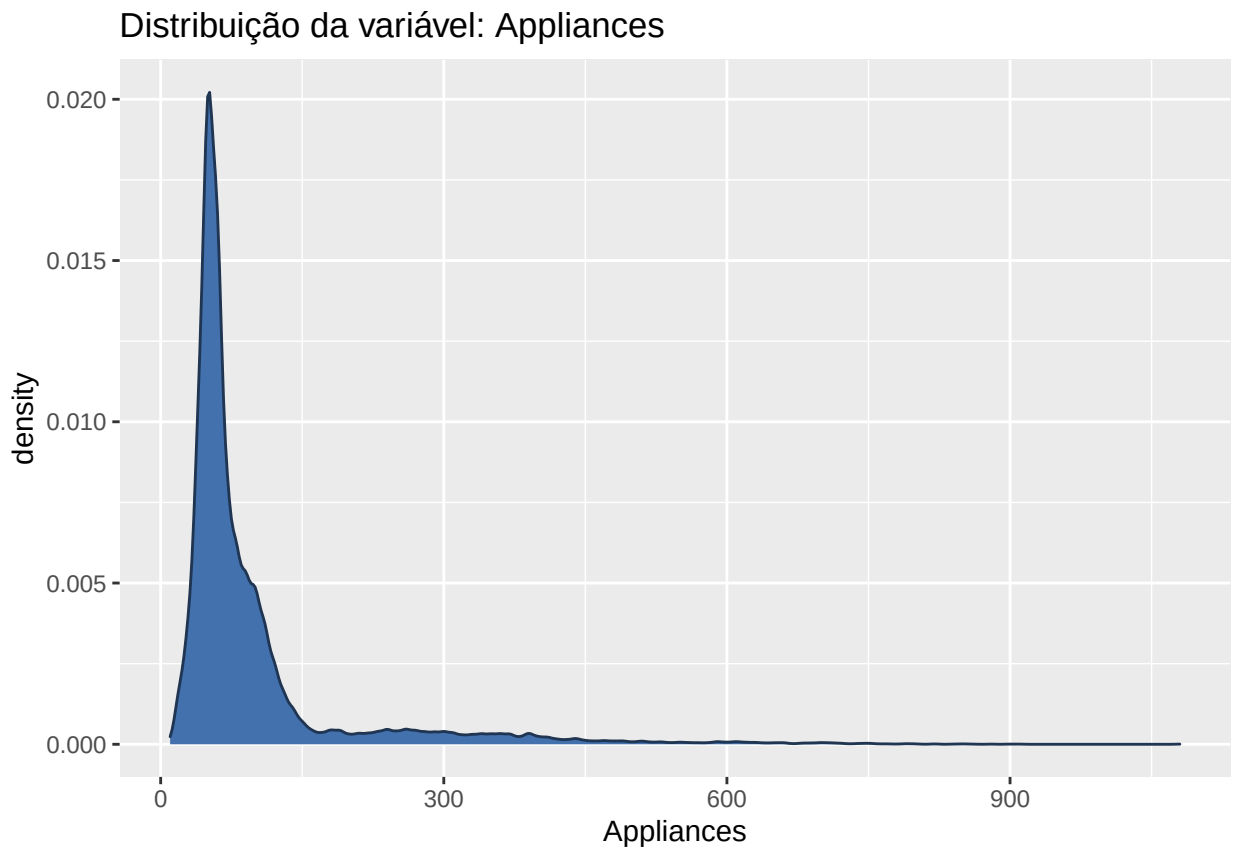
Funções auxiliares

```
dist_plot <- function(data, col) {

  ggplot() +
```

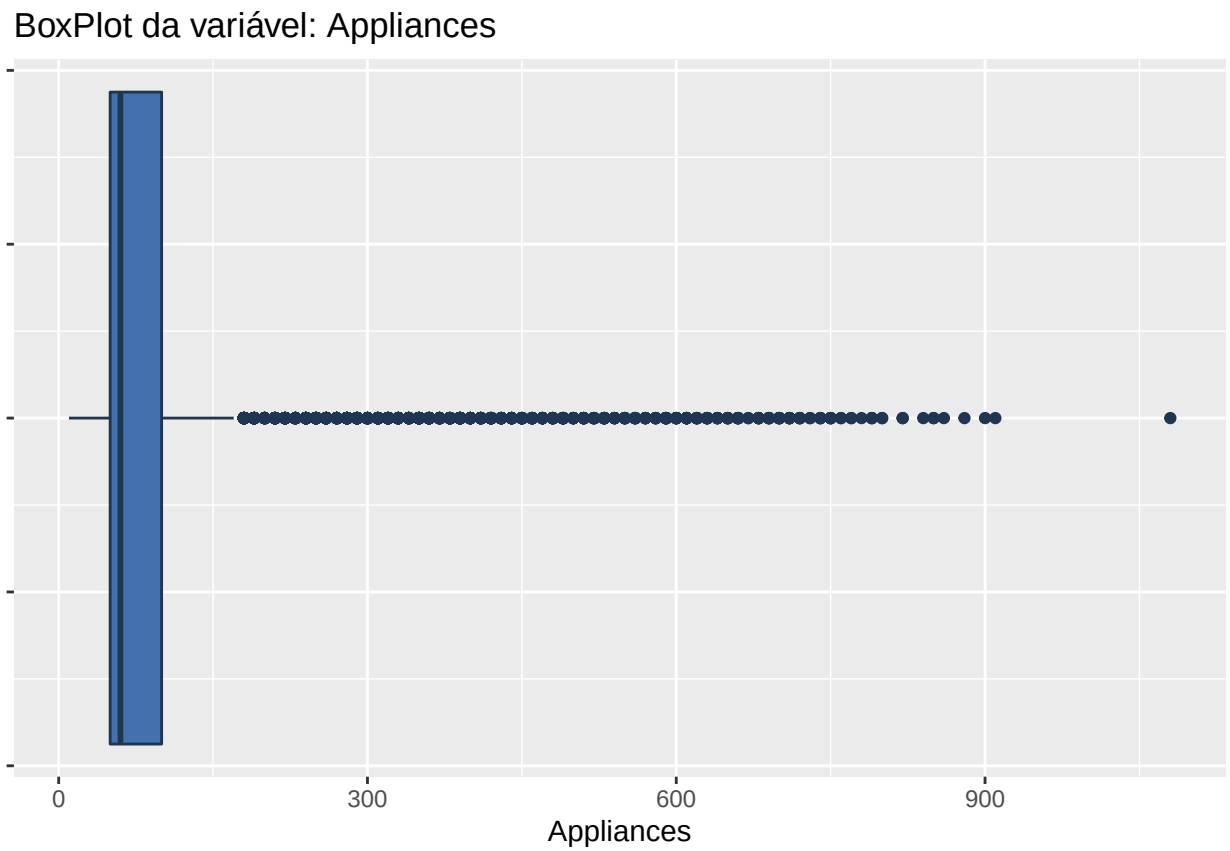
```
    geom_density(aes(data[,col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('Distribuição da variável:',col), x = col)

}

box_plot <- function(data, col, title, xlab) {

  ggplot() +
    geom_boxplot(aes(x = data[, col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('BoxPlot da variável:',col), x = col) +
    theme(axis.text.y = element_blank())
}

bar_plot <- function(data, col, title, xlab) {

  ggplot() +
    geom_bar(aes(x = data[, col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('Grafico de barra da variável: ',col), x = col)

}
```

Appliances

```
dist_plot(data = df_train, col = 'Appliances')
```



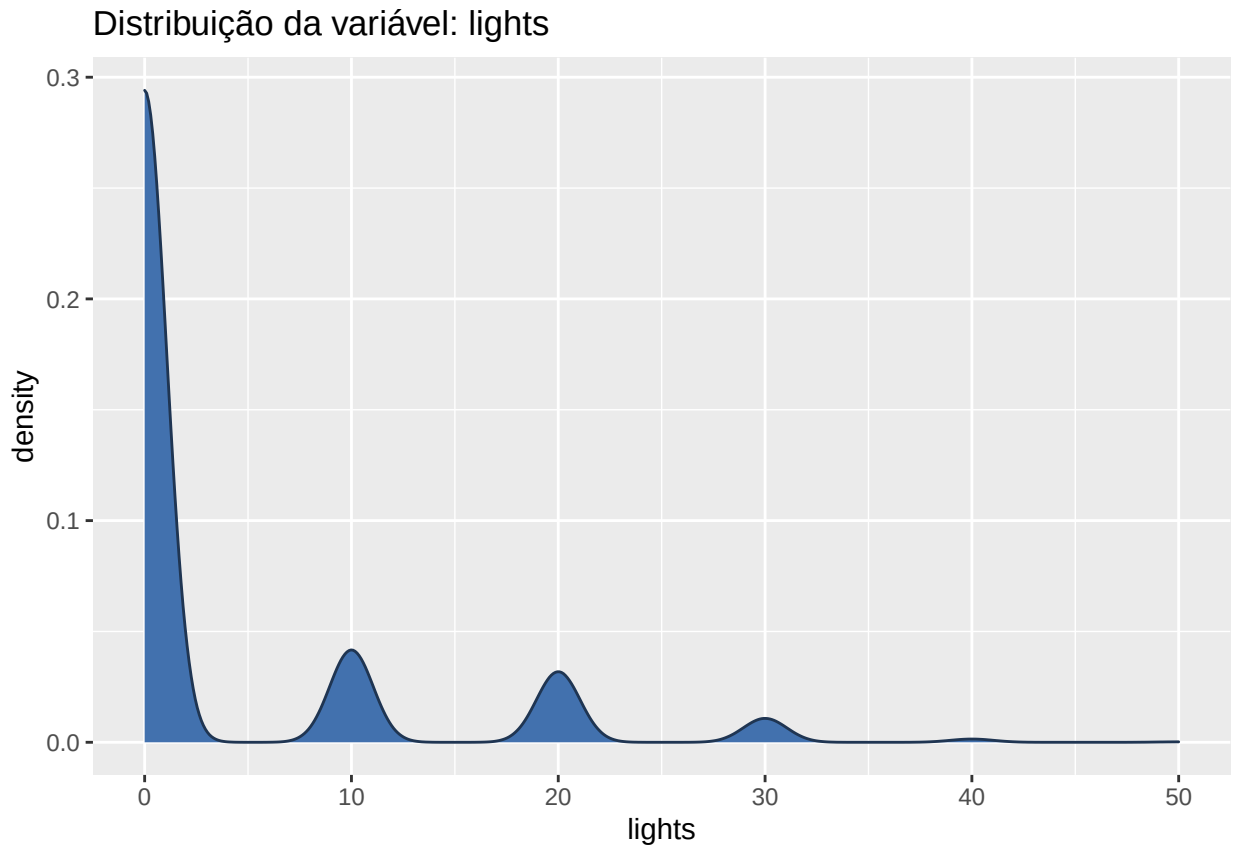
```
box_plot(data = df_train,col = 'Appliances')
```



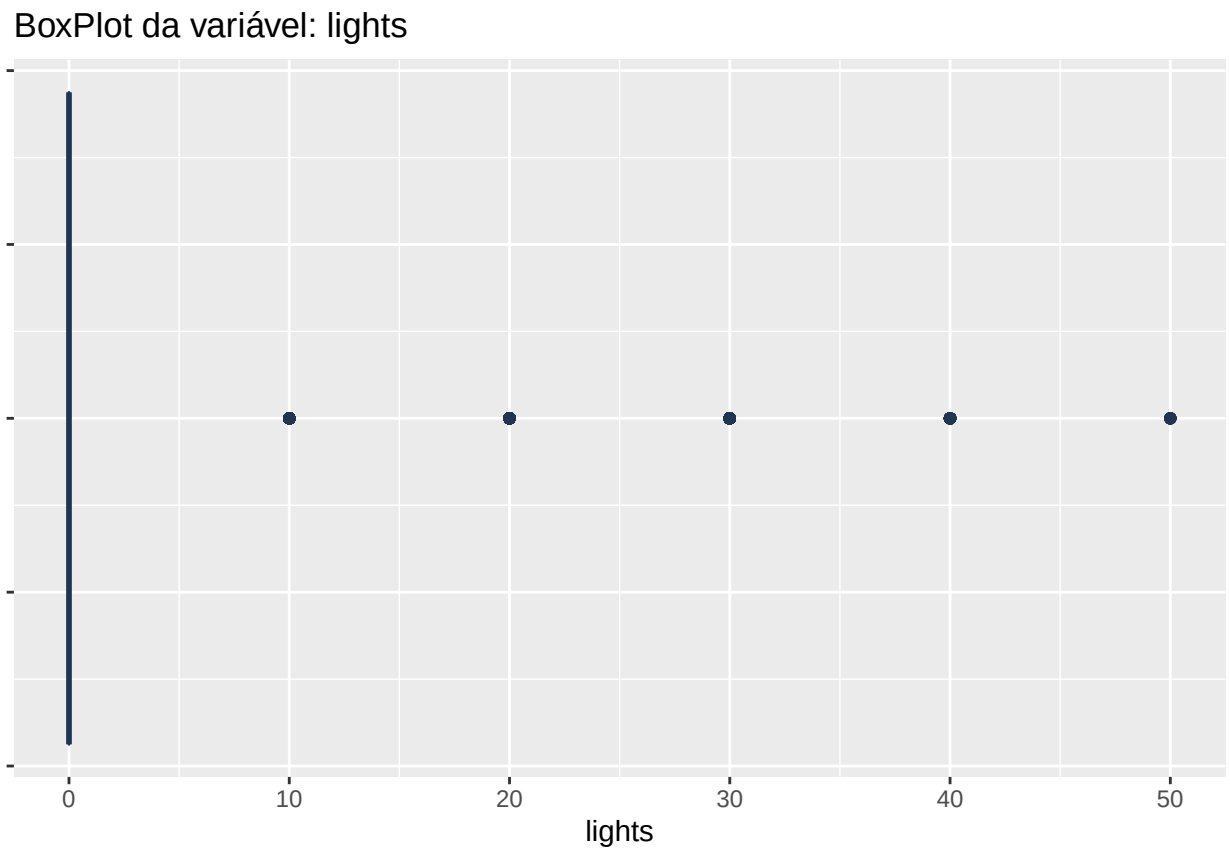
Com o distplot podemos ver uma assimetria nos dados, onde se encontram mais na parte esquerda, e com a ajuda do boxplot podemos também ver isso assim como a identificação de valores outliers que deveram ser tratados antos da aplicação dos modelos.

lights

```
dist_plot(data = df_train, col = 'lights')
```



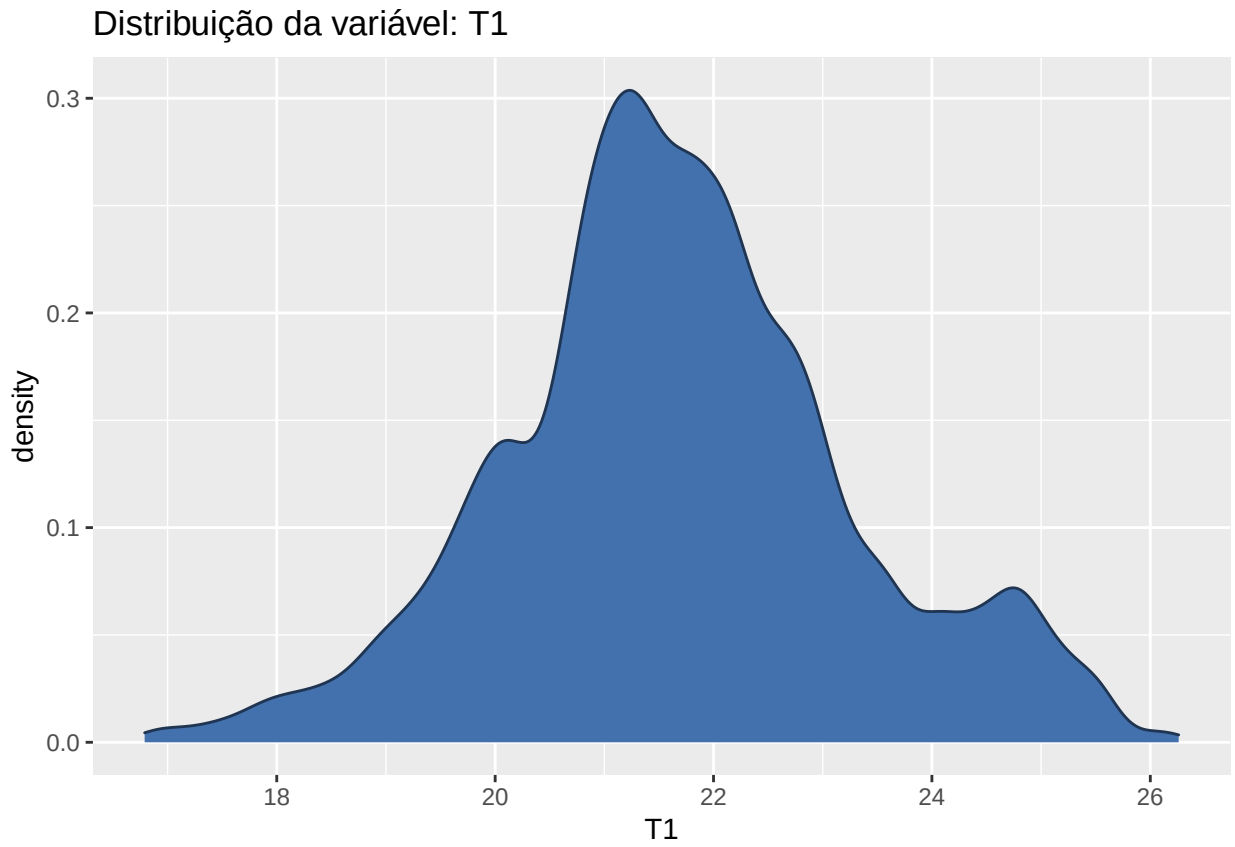

```
box_plot(data = df_train,col = 'lights')
```



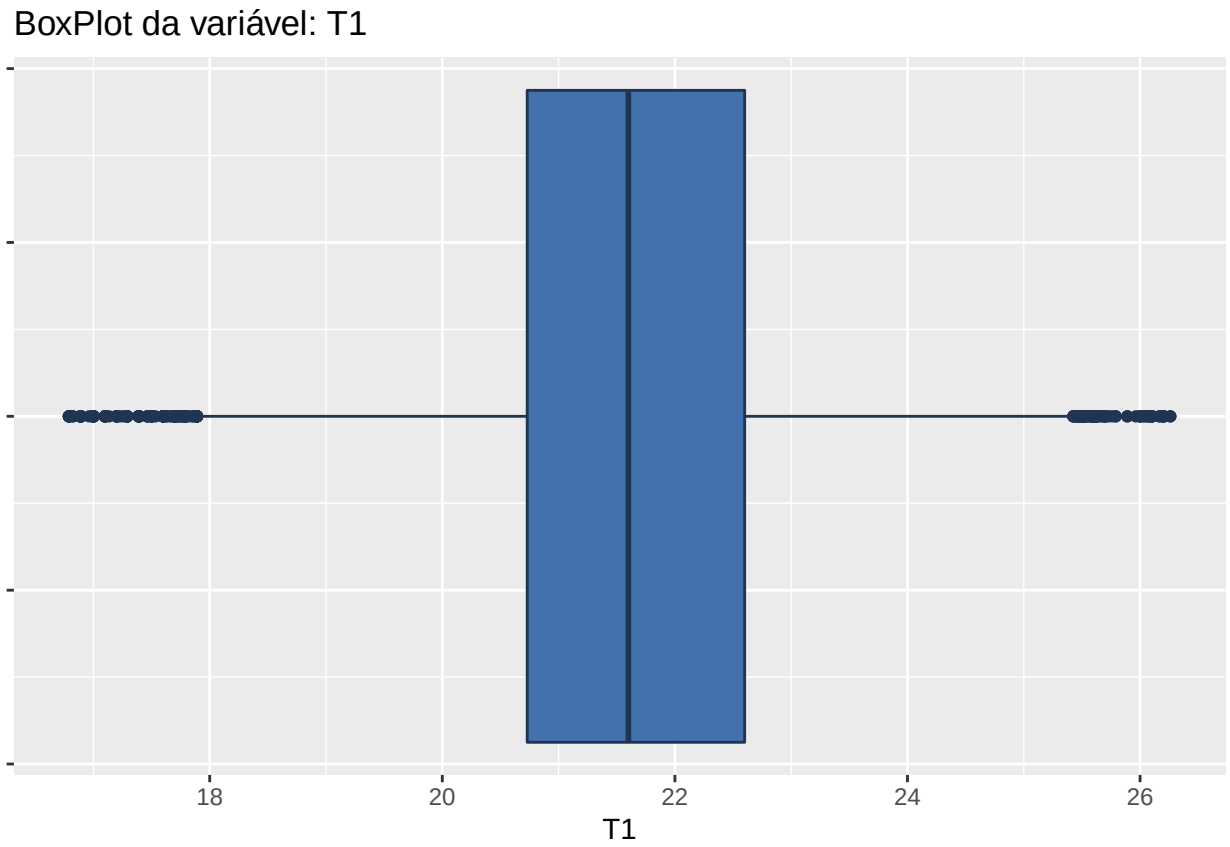
Com o distplot podemos ver uma assimetria nos dados, onde basicamente todos os valores estão próximos de 0 , e com a ajuda do boxplot podemos constatar que não possuem muitos valores outliers.

T1

```
dist_plot(data = df_train, col = 'T1')
```



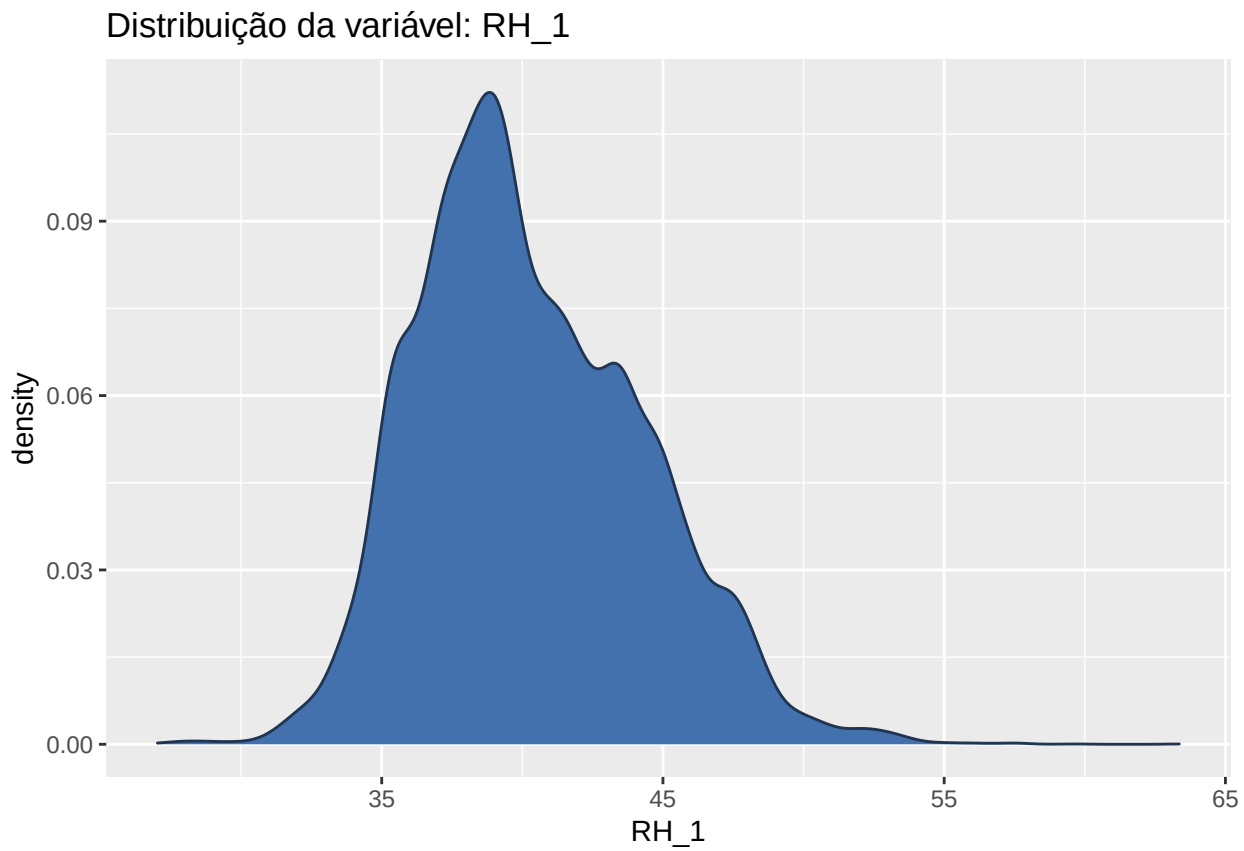

```
box_plot(data = df_train,col = 'T1')
```



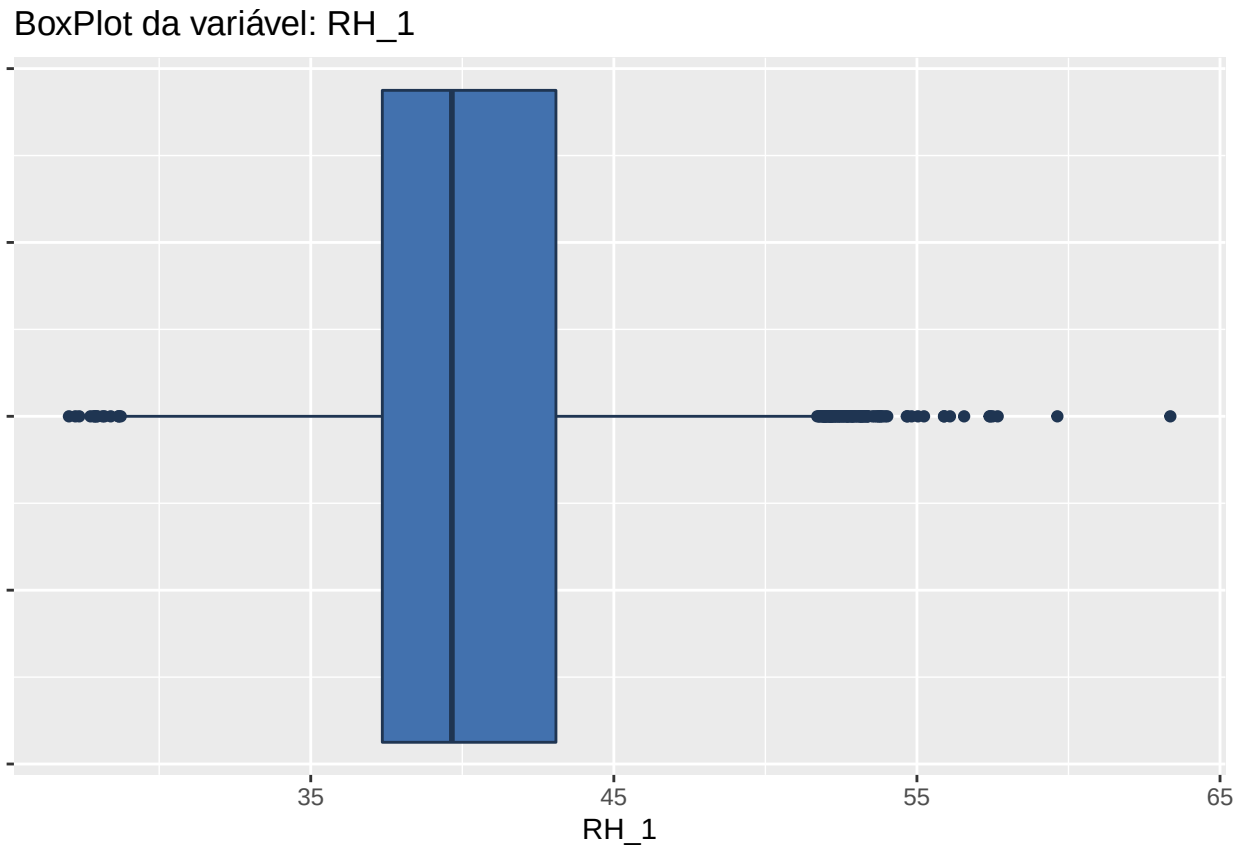
Com o distplot podemos ver que os dados estão quase simétricos apenas com algumas variações, e podemos ver com o boxplot outliers tanto na borda esquerda quando na direita.

RH_1

```
dist_plot(data = df_train, col = 'RH_1')
```



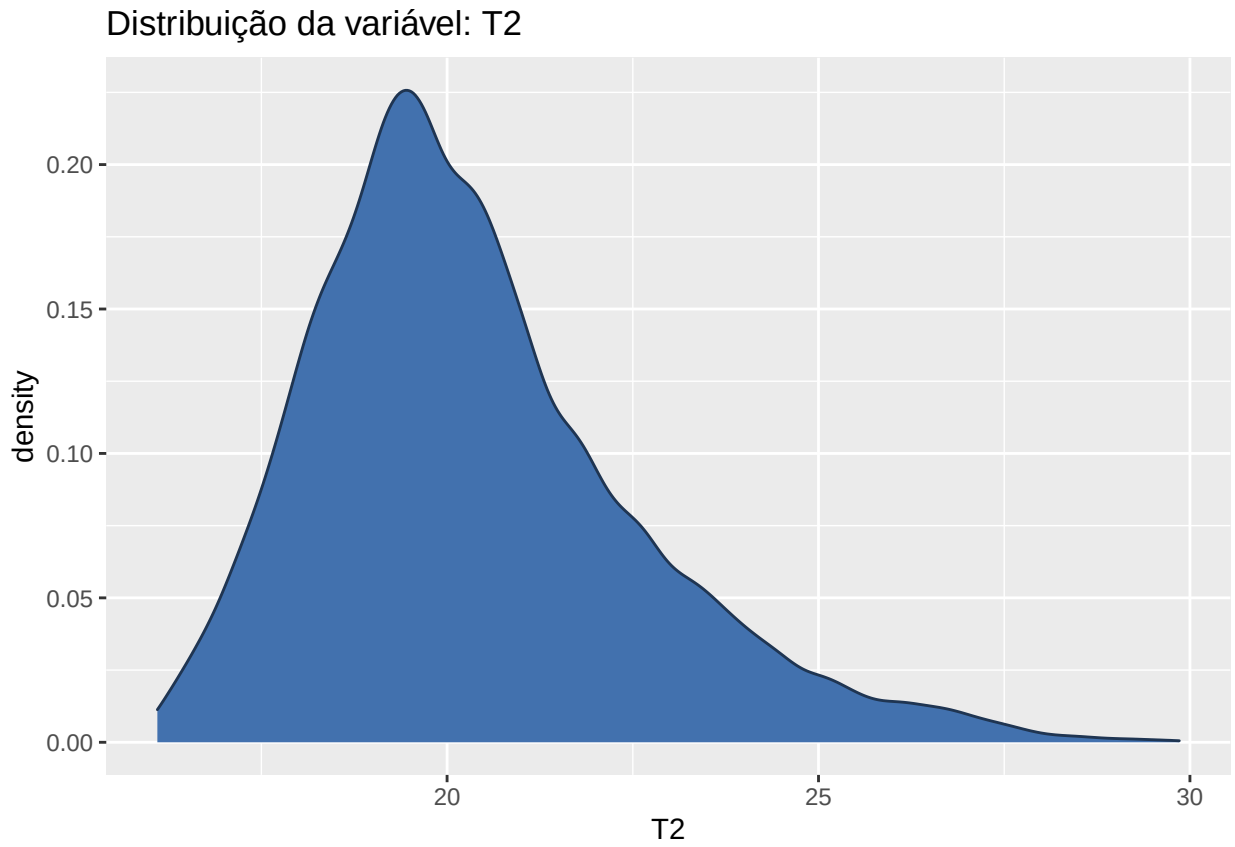

```
box_plot(data = df_train,col = 'RH_1')
```



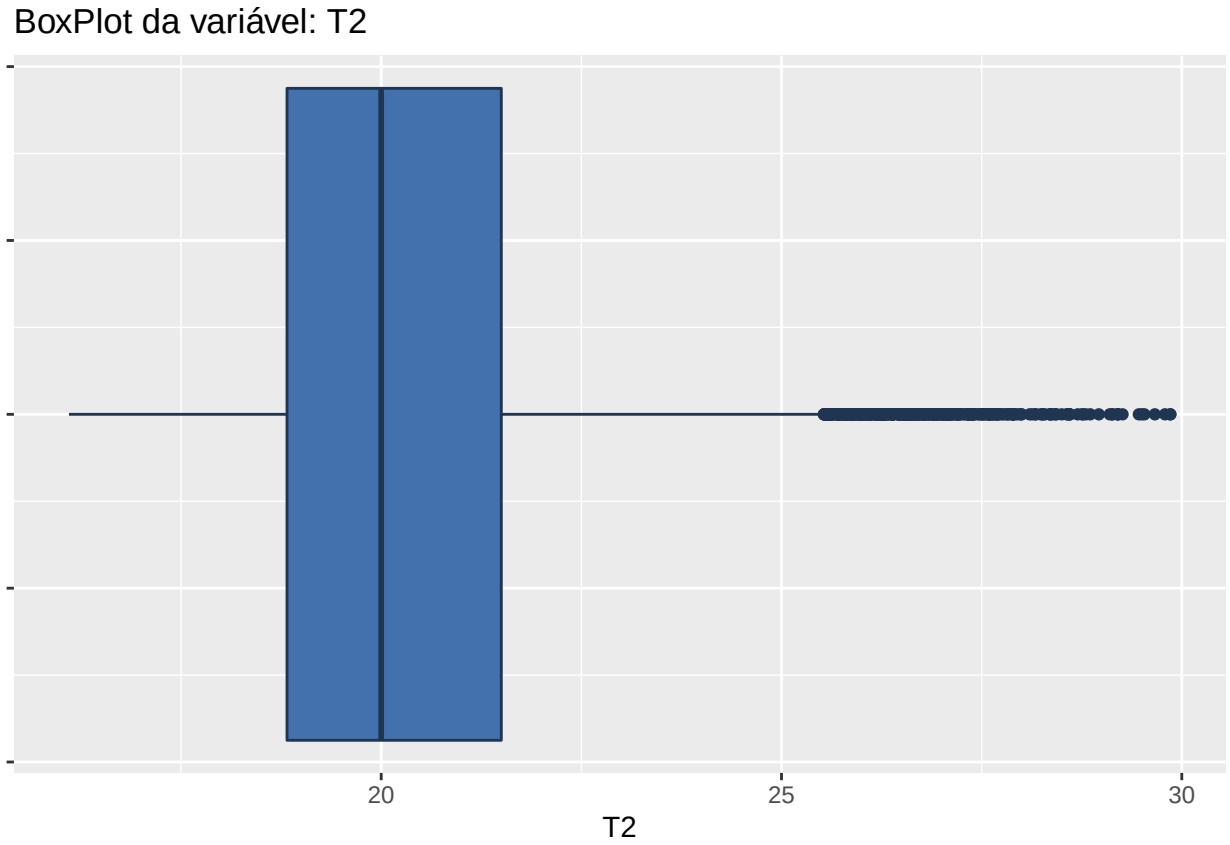
Com o distplot podemos ver que os dados estão quase simétricos apenas com algumas variações, e podemos ver com o boxplot outliers tanto na borda esquerda quando na direita.

T2

```
dist_plot(data = df_train, col = 'T2')
```



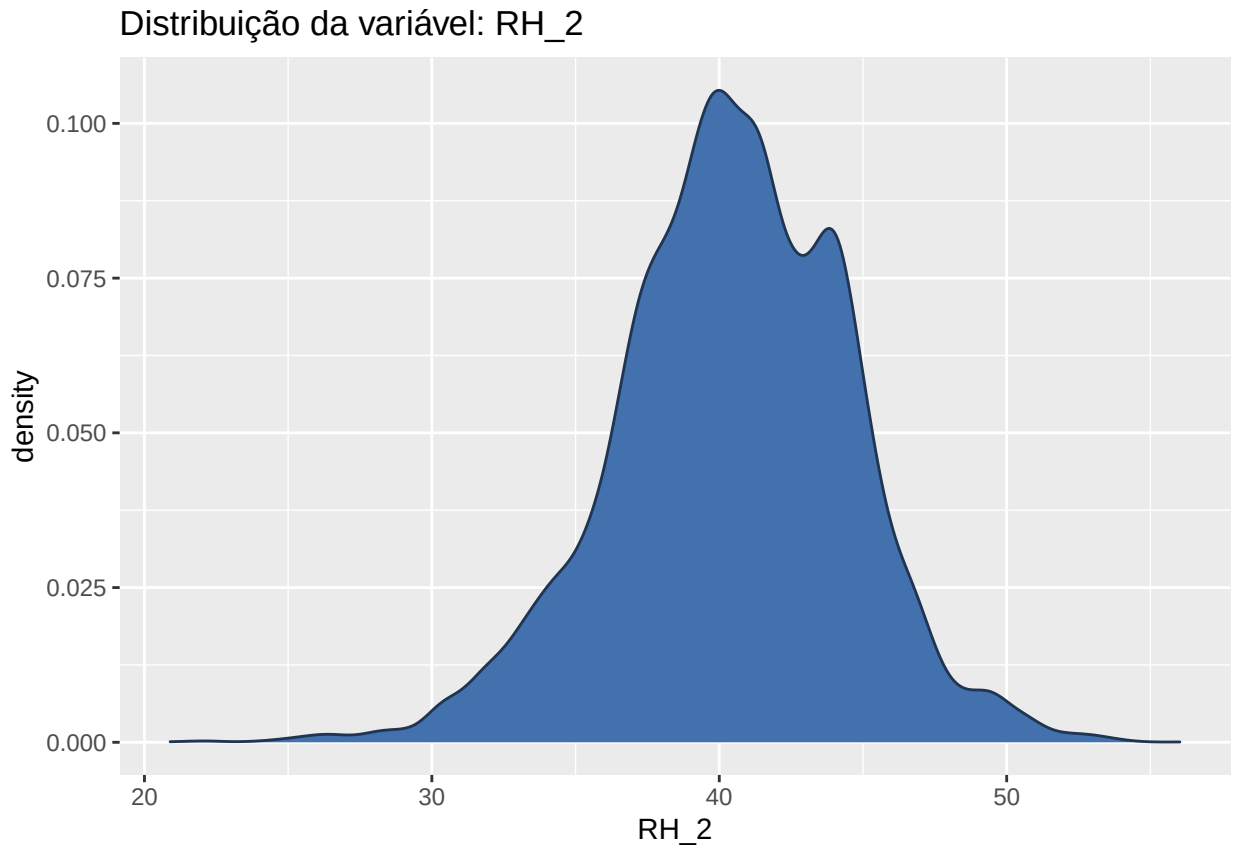

```
box_plot(data = df_train,col = 'T2')
```



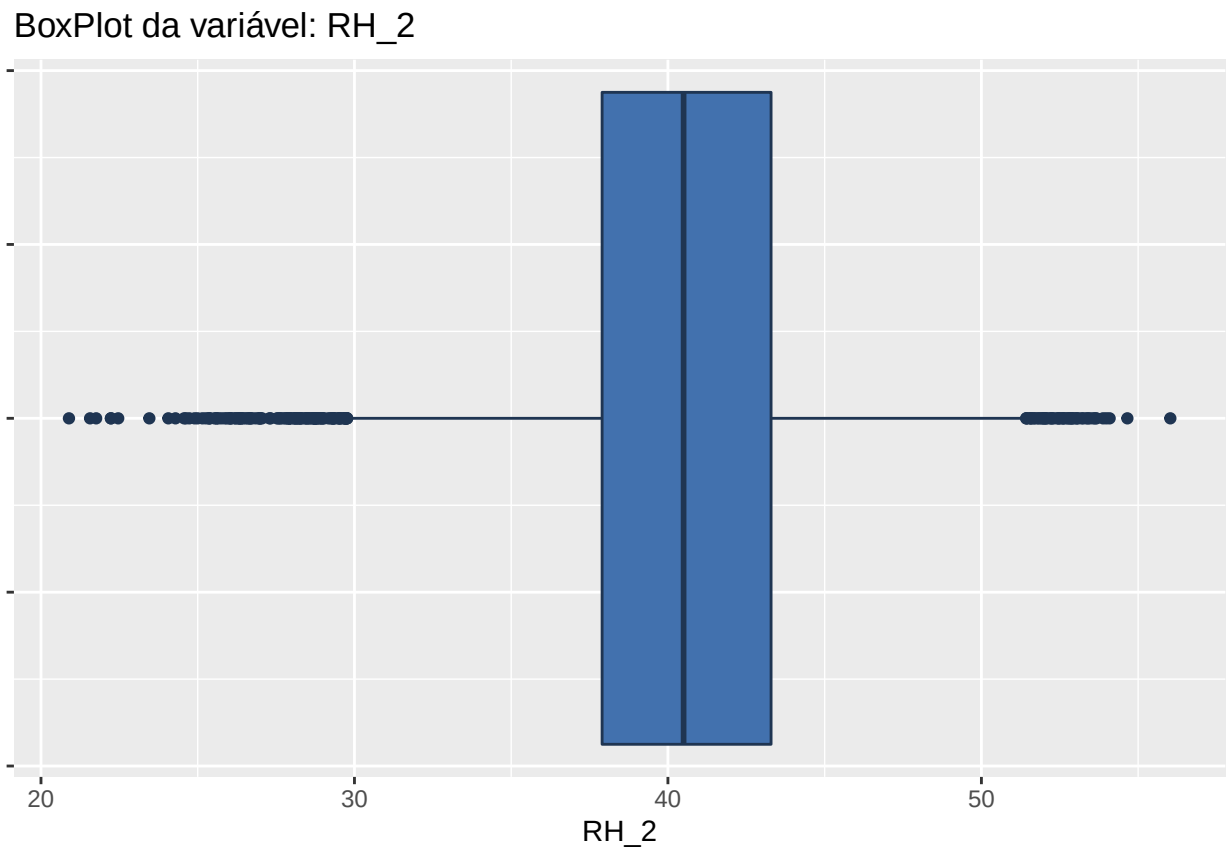
Com o distplot podemos ver que os dados um pouco mais concentrados a esquerda, porem quase simétrico, e com o boxplot constatamos alguns outliers na borda direita.

RH_2

```
dist_plot(data = df_train, col = 'RH_2')
```



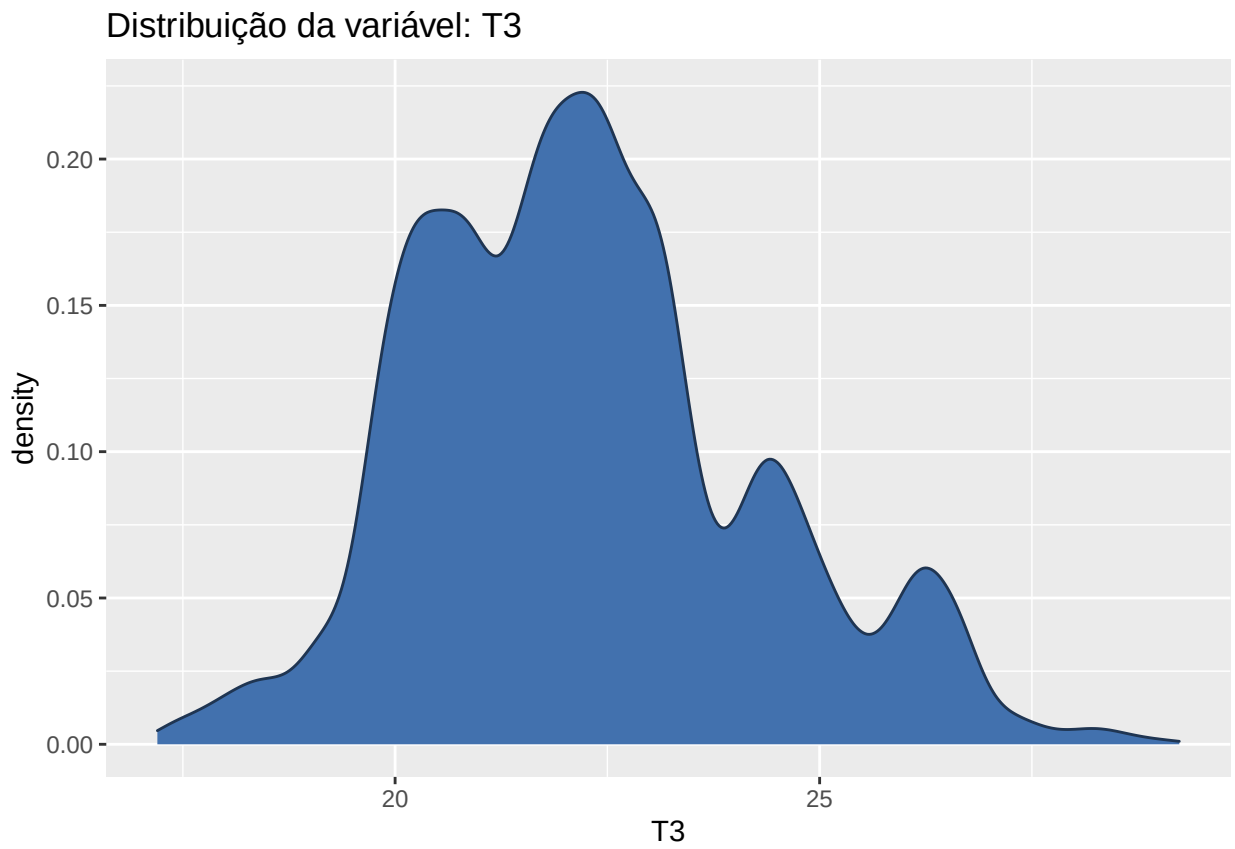

```
box_plot(data = df_train,col = 'RH_2')
```



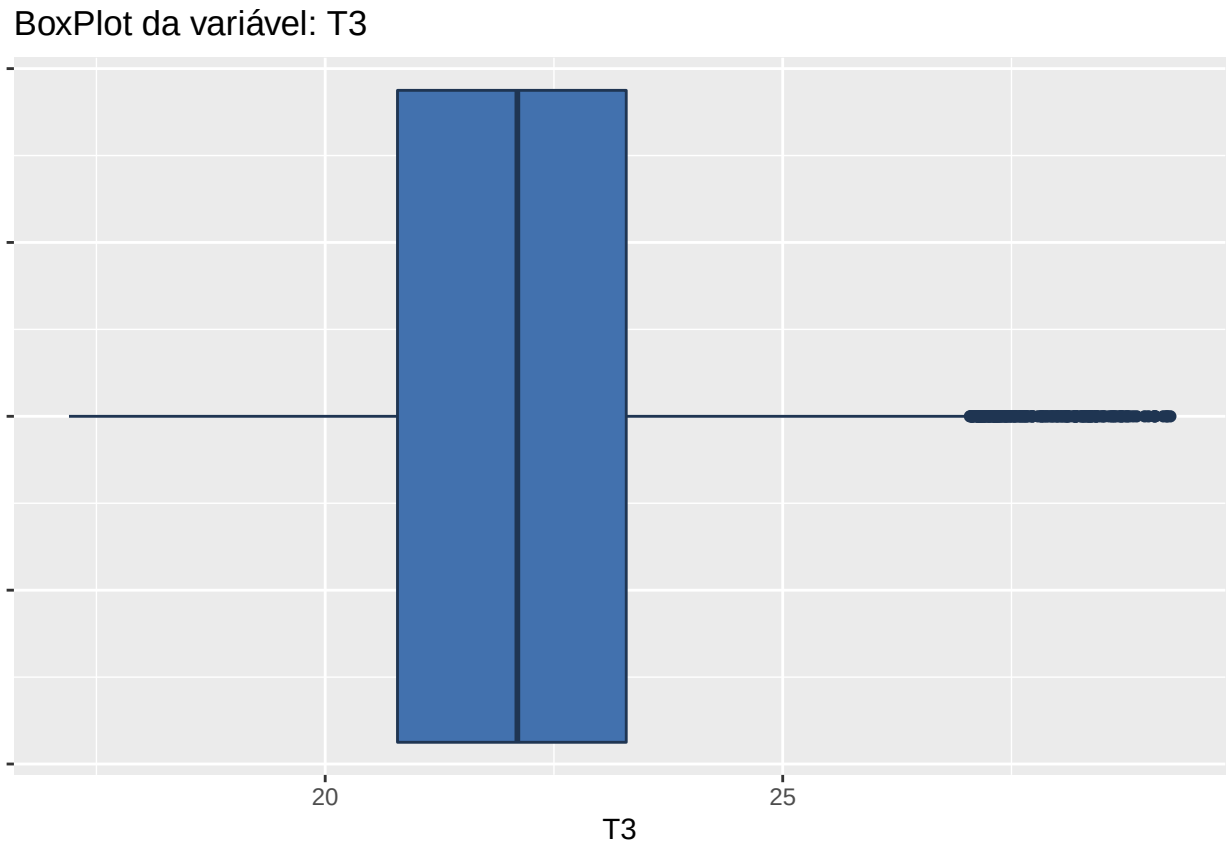
Com o distplot podemos ver que os dados estão quase simétricos apenas com algumas variações, e podemos ver com o boxplot outliers tanto na borda esquerda quando na direita.

T3

```
dist_plot(data = df_train, col = 'T3')
```



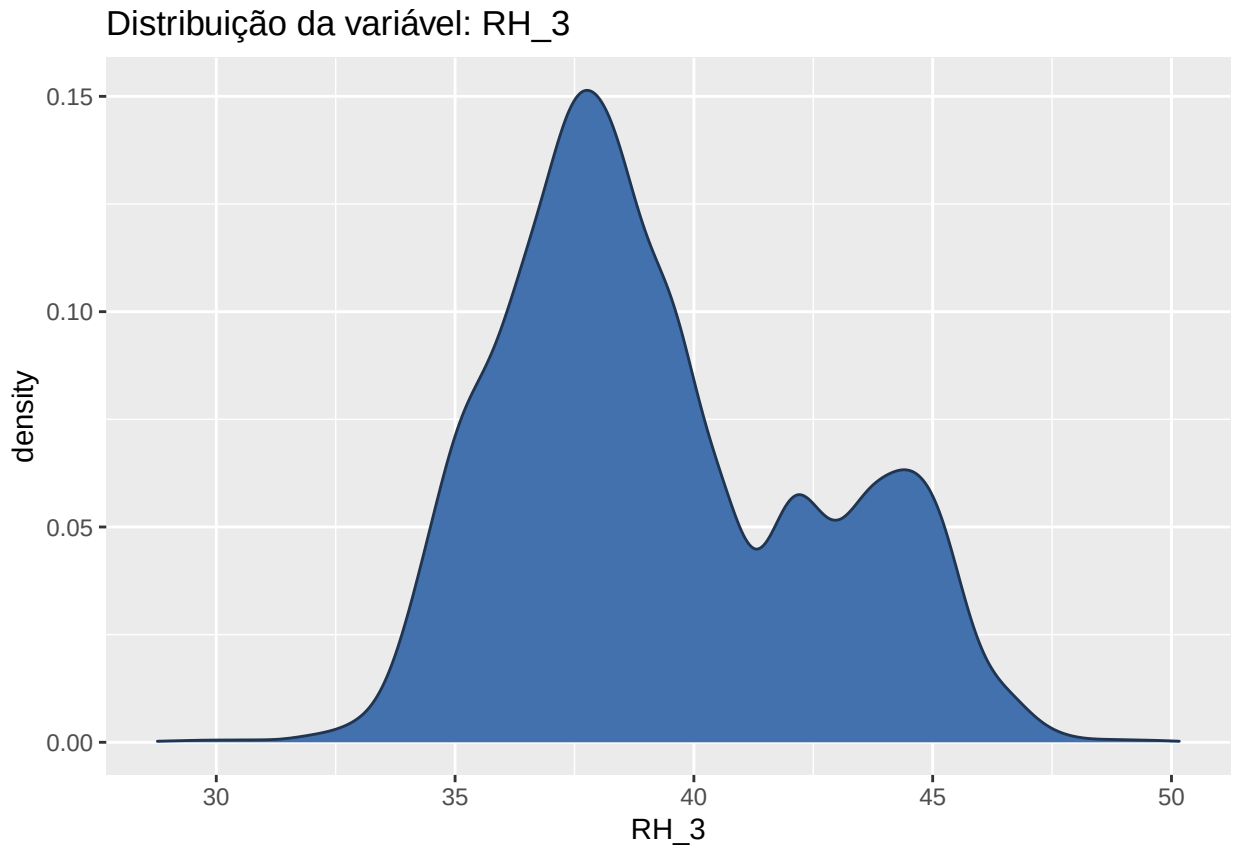

```
box_plot(data = df_train,col = 'T3')
```



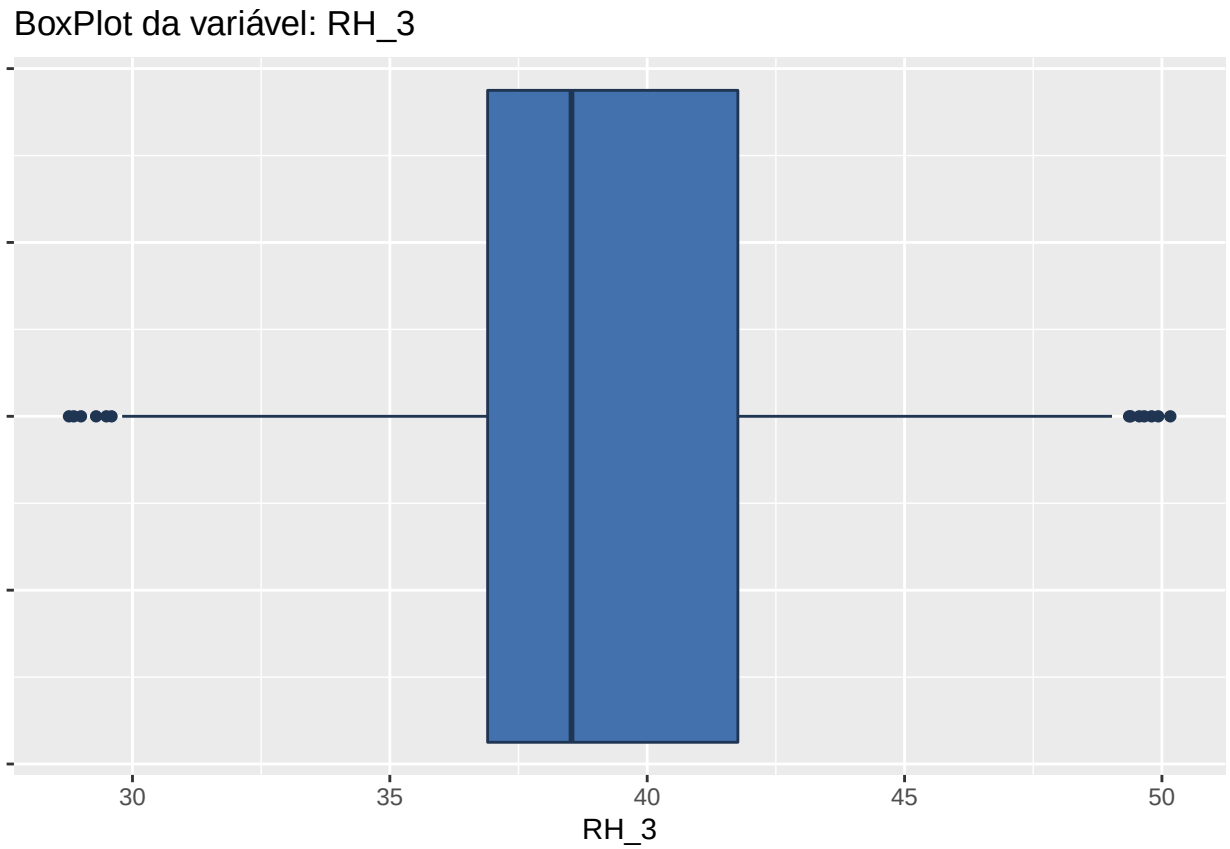
Com o distplot podemos ver que os dados estão bem distribuídos e com o boxplot temos alguns outliers na borda direita.

RH_3

```
dist_plot(data = df_train, col = 'RH_3')
```



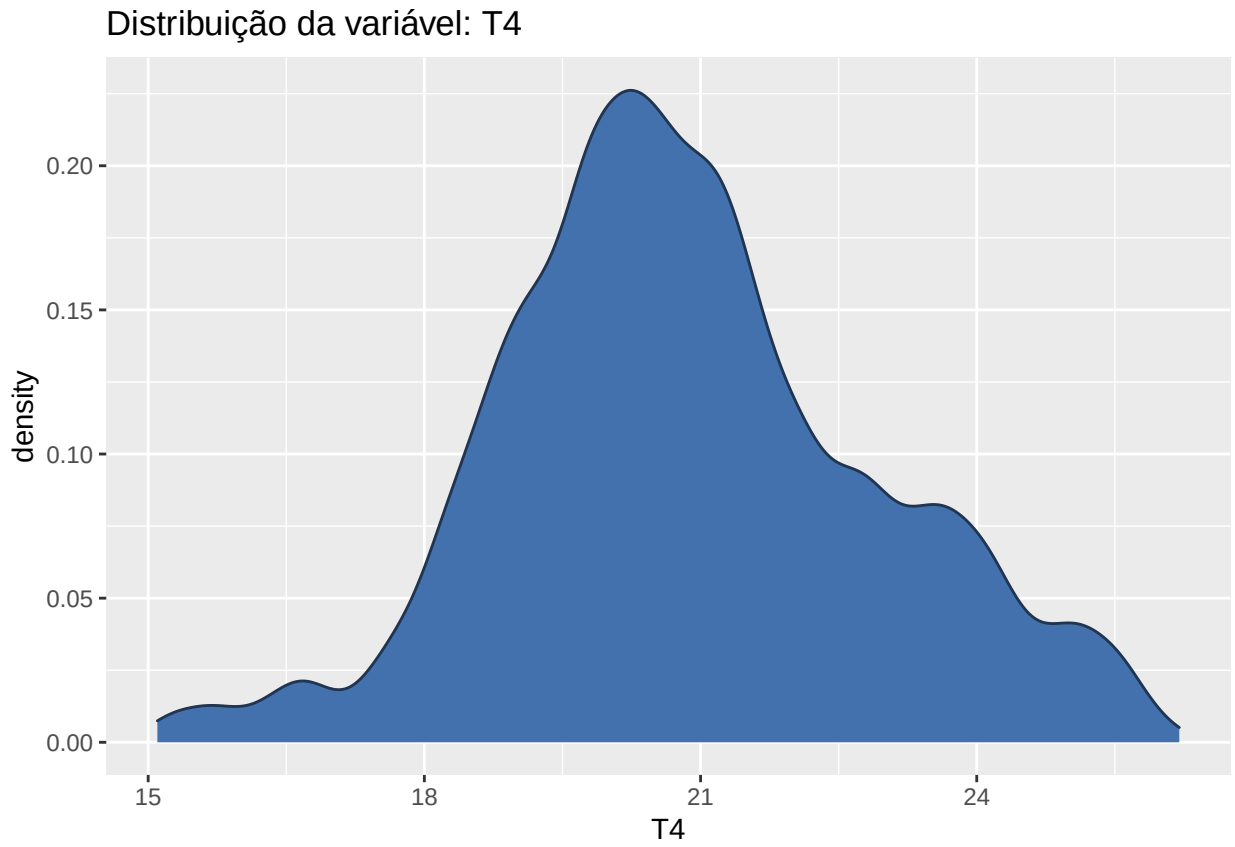

```
box_plot(data = df_train,col = 'RH_3')
```



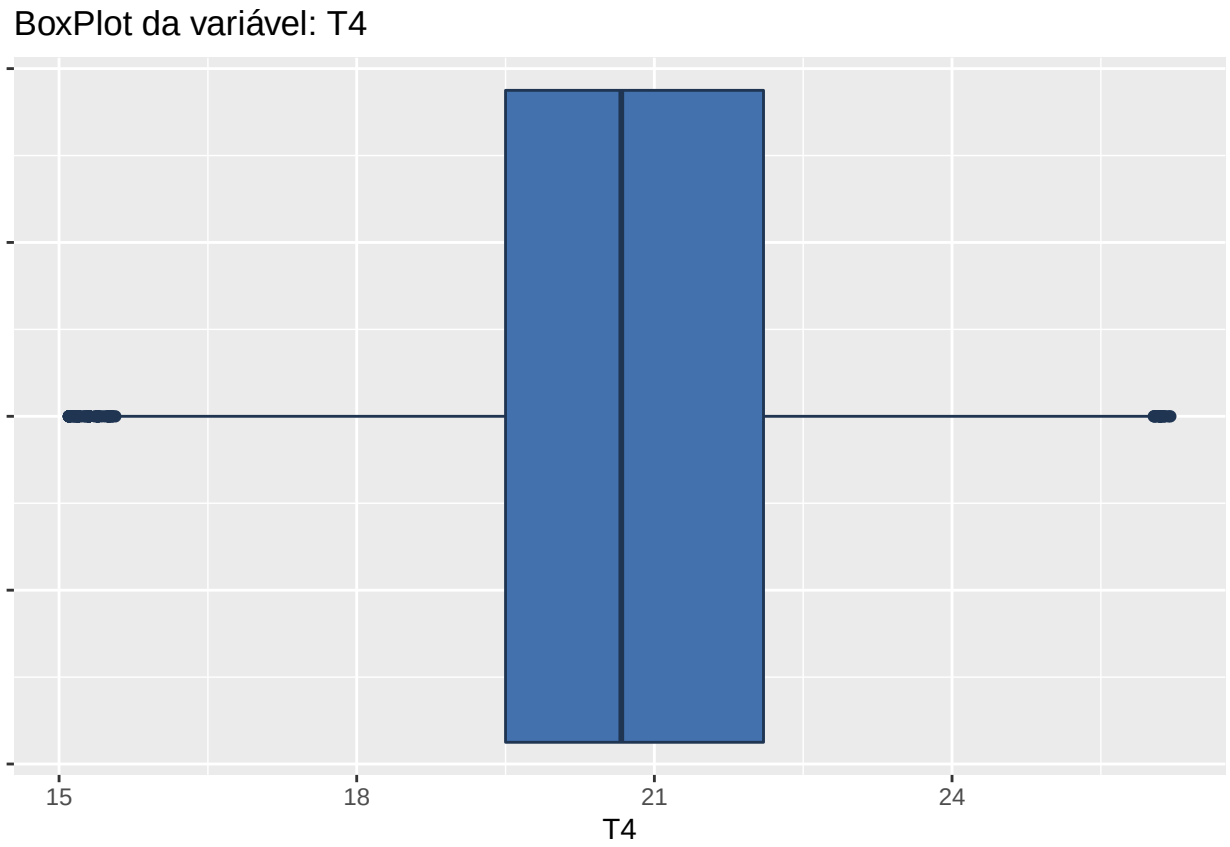
Com o distplot podemos ver que duas caudas nos dados uma um pouco mais acentuada e outra um pouco menor e com o boxplot podemos ver que existe poucos outliers

T4

```
dist_plot(data = df_train, col = 'T4')
```



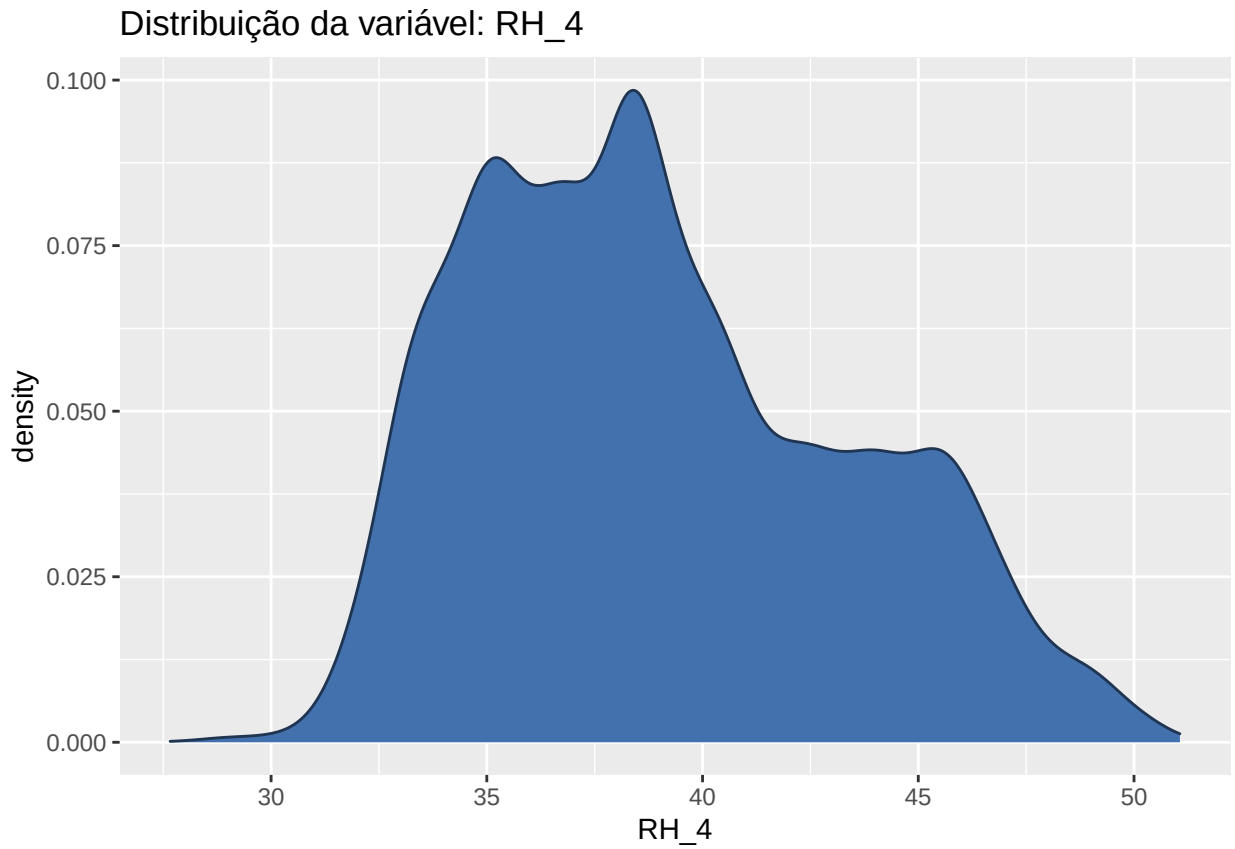

```
box_plot(data = df_train,col = 'T4')
```



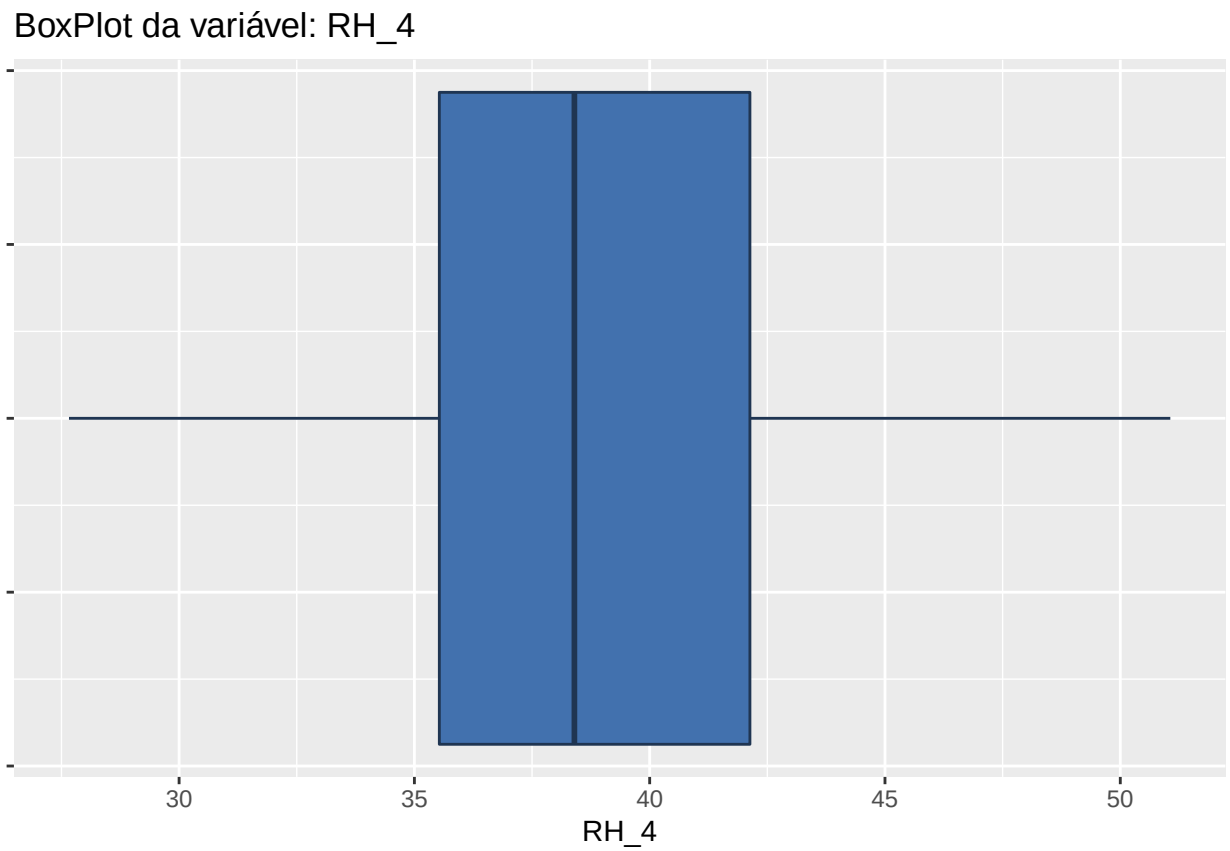
Com o distplot podemos ver que os dados estão quase simétricos apenas com algumas variações, e podemos ver com o boxplot alguns outliers tanto na borda esquerda quando na direita.

RH_4

```
dist_plot(data = df_train, col = 'RH_4')
```



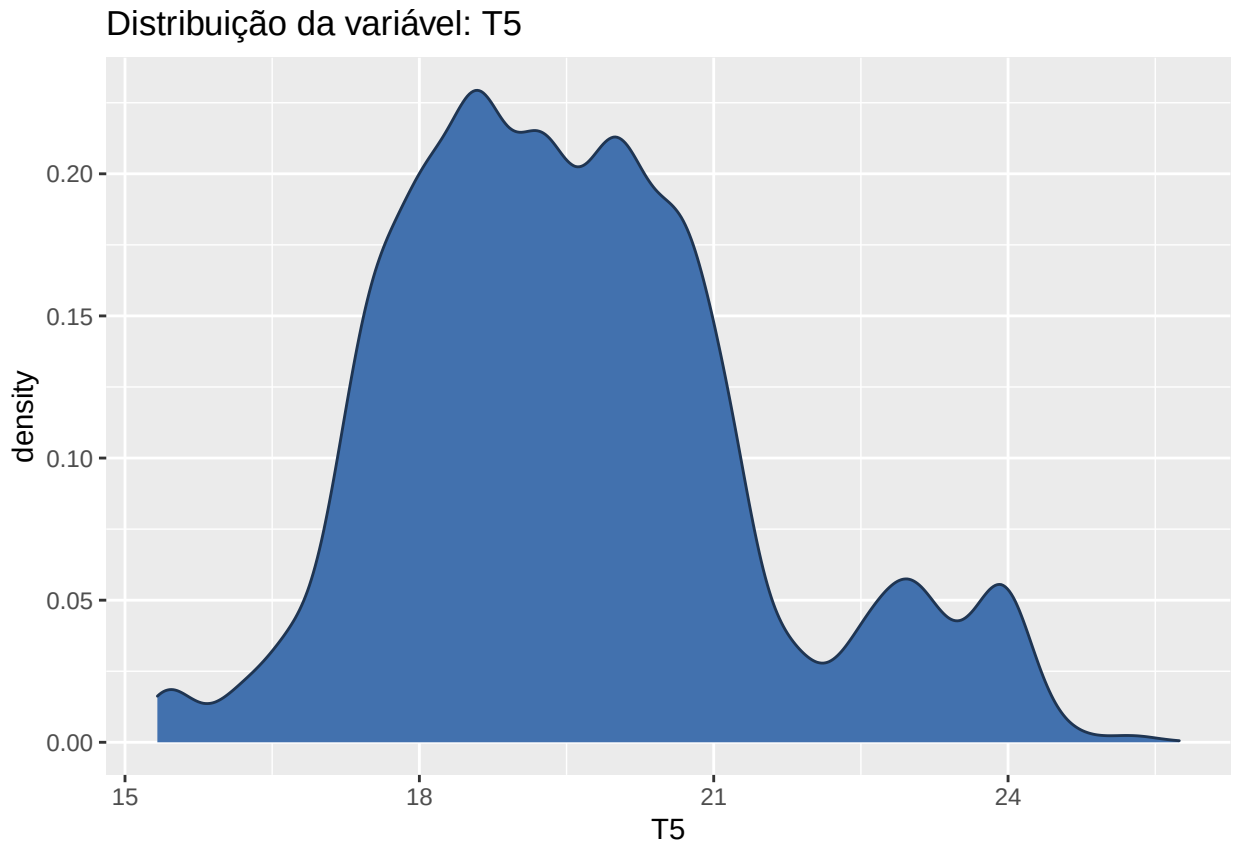

```
box_plot(data = df_train,col = 'RH_4')
```



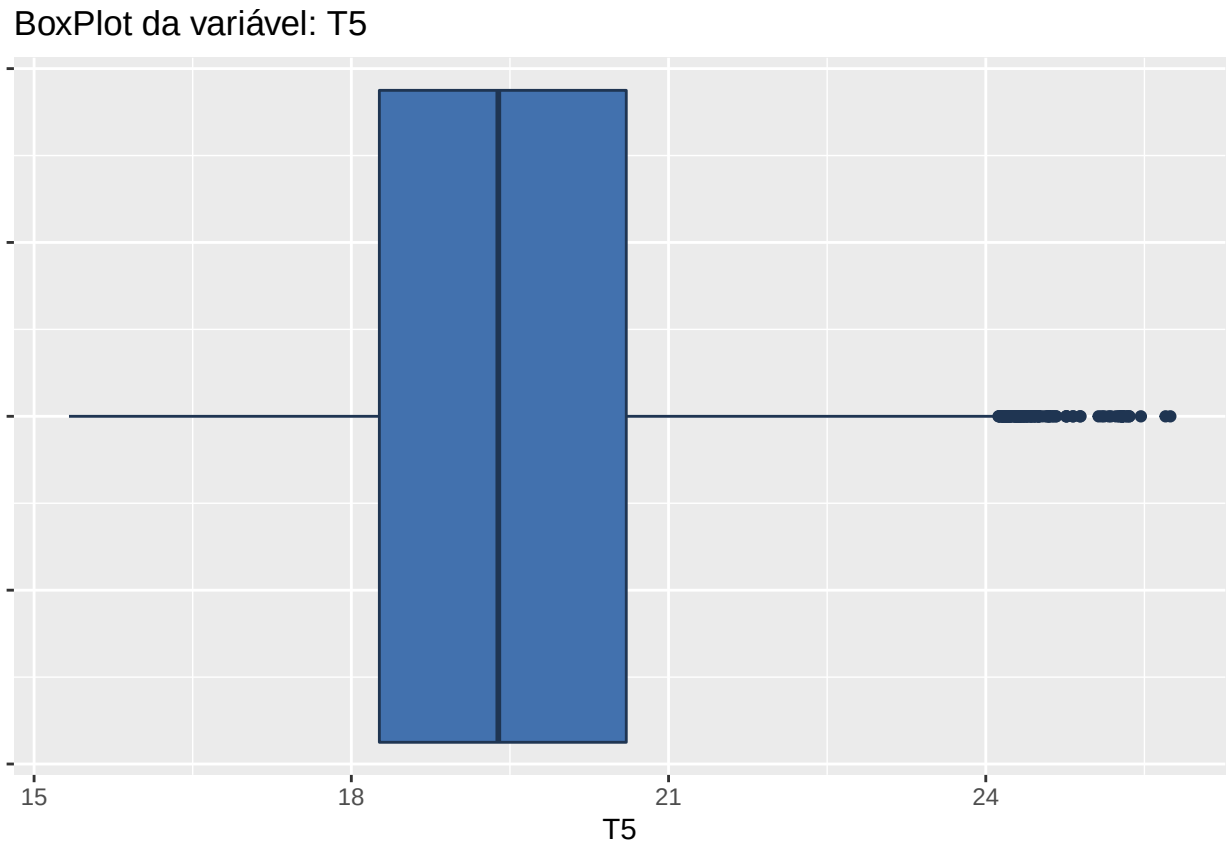
Com o distplot podemos ver que os dados estão bem distribuídos e por isso não á nenhum outliers como visto no boxplot.

T5

```
dist_plot(data = df_train, col = 'T5')
```



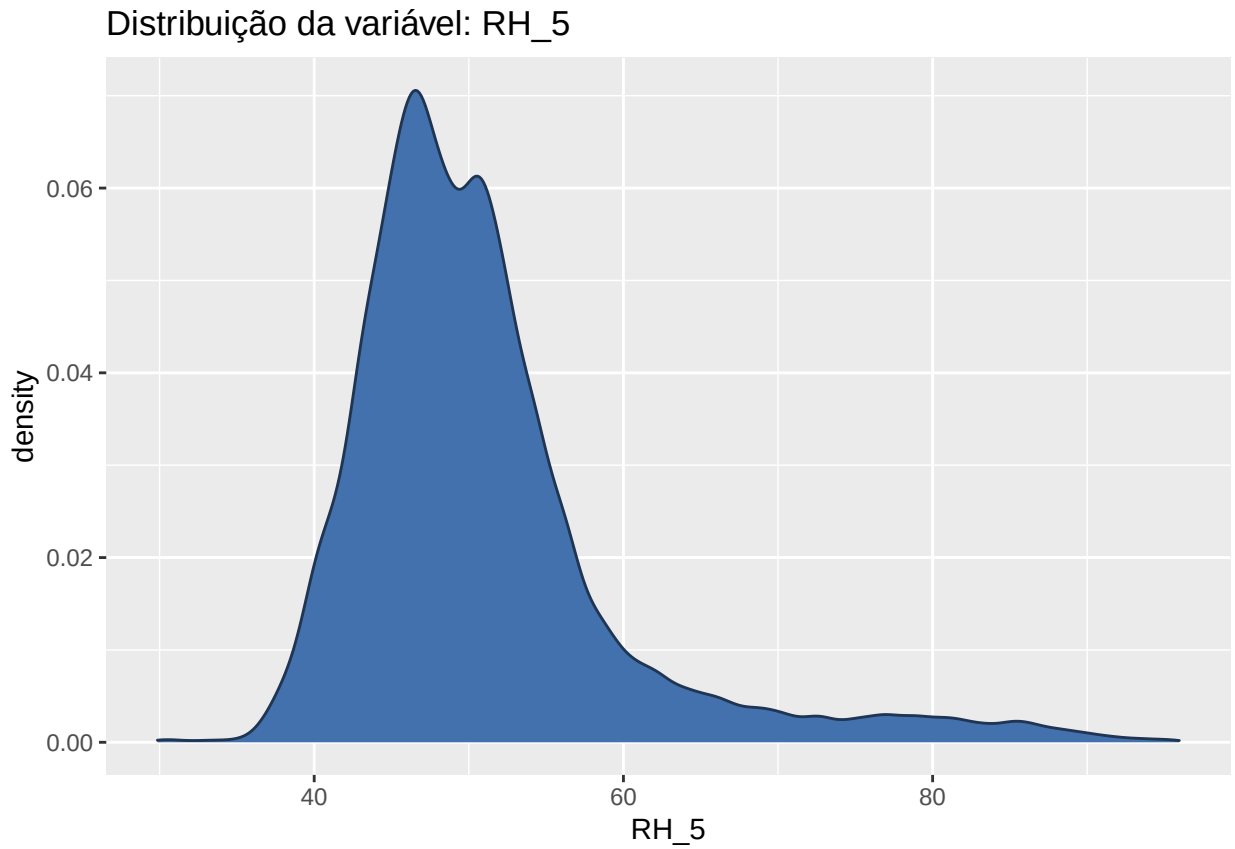

```
box_plot(data = df_train,col = 'T5')
```



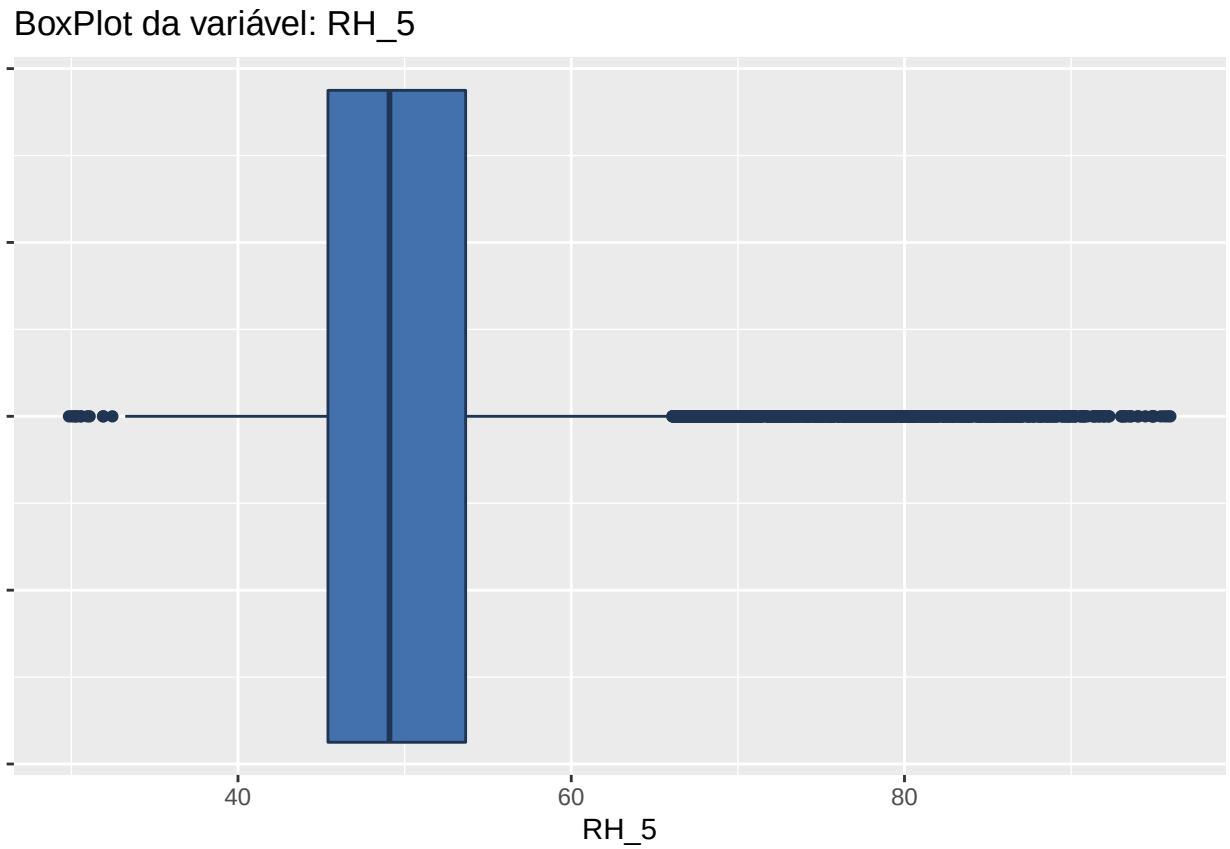
Com o distplot podemos ver que os dados estão quase simétricos e poucos outliers na borda direita como visto no boxplot.

RH_5

```
dist_plot(data = df_train, col = 'RH_5')
```



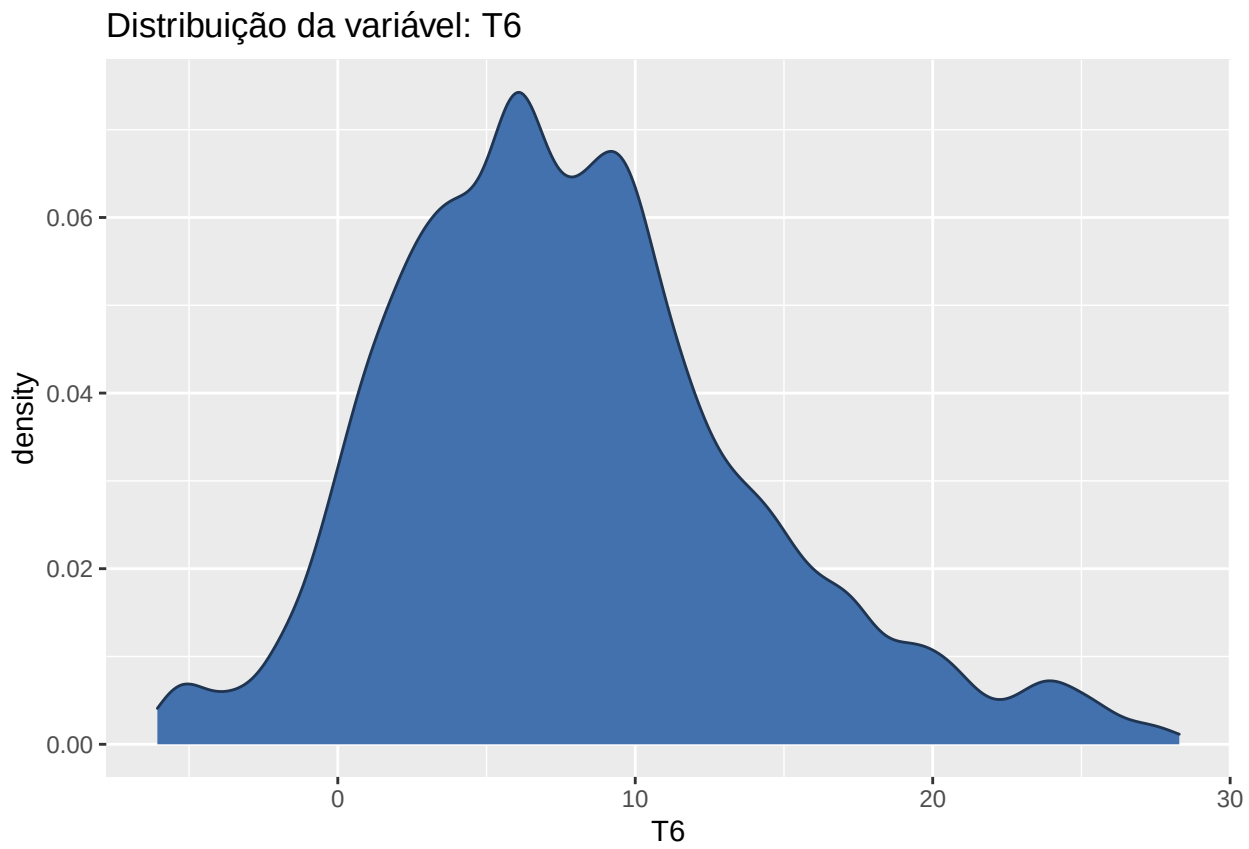

```
box_plot(data = df_train,col = 'RH_5')
```



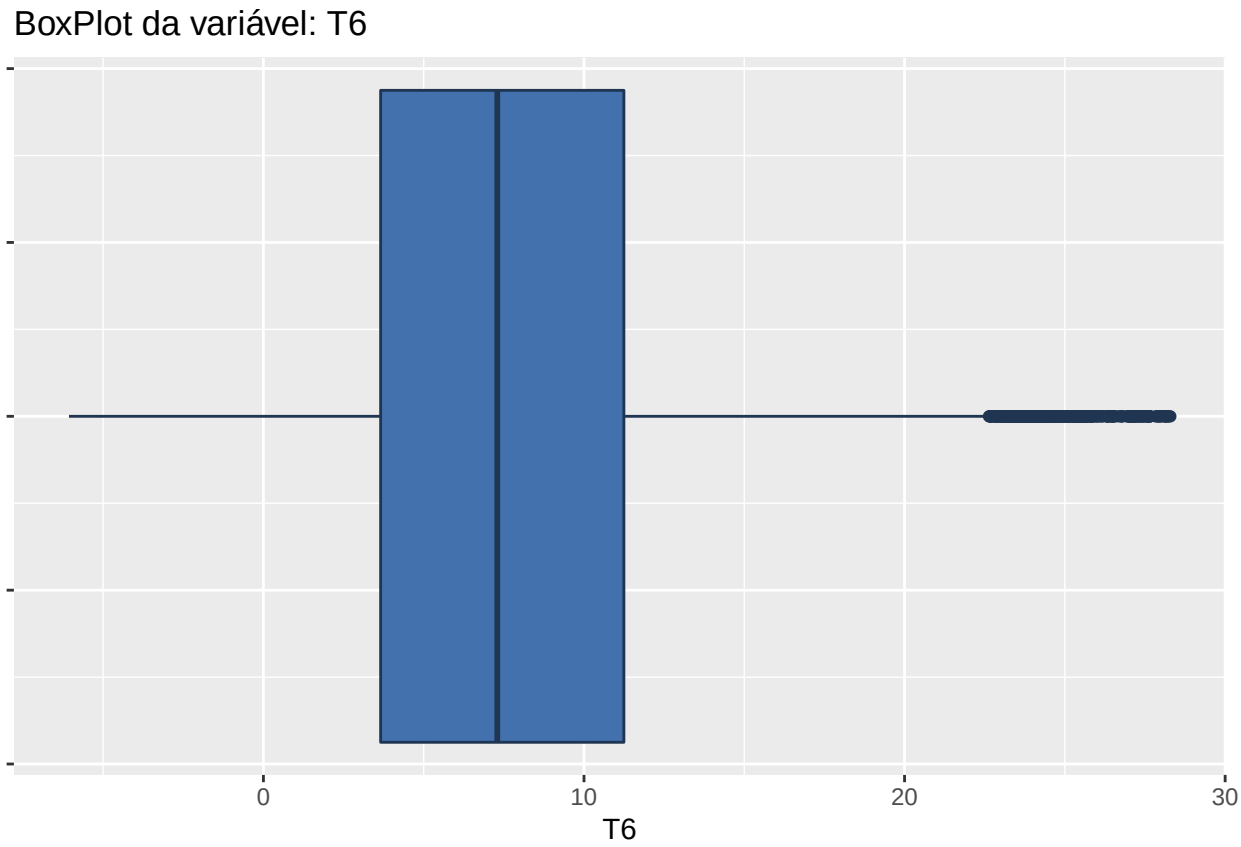
Com o distplot podemos ver que os dados tendem um pouco a esquerda, e tempos alguns outliers como visto com o boxplot, sobretudo na boda direita.

T6

```
dist_plot(data = df_train, col = 'T6')
```



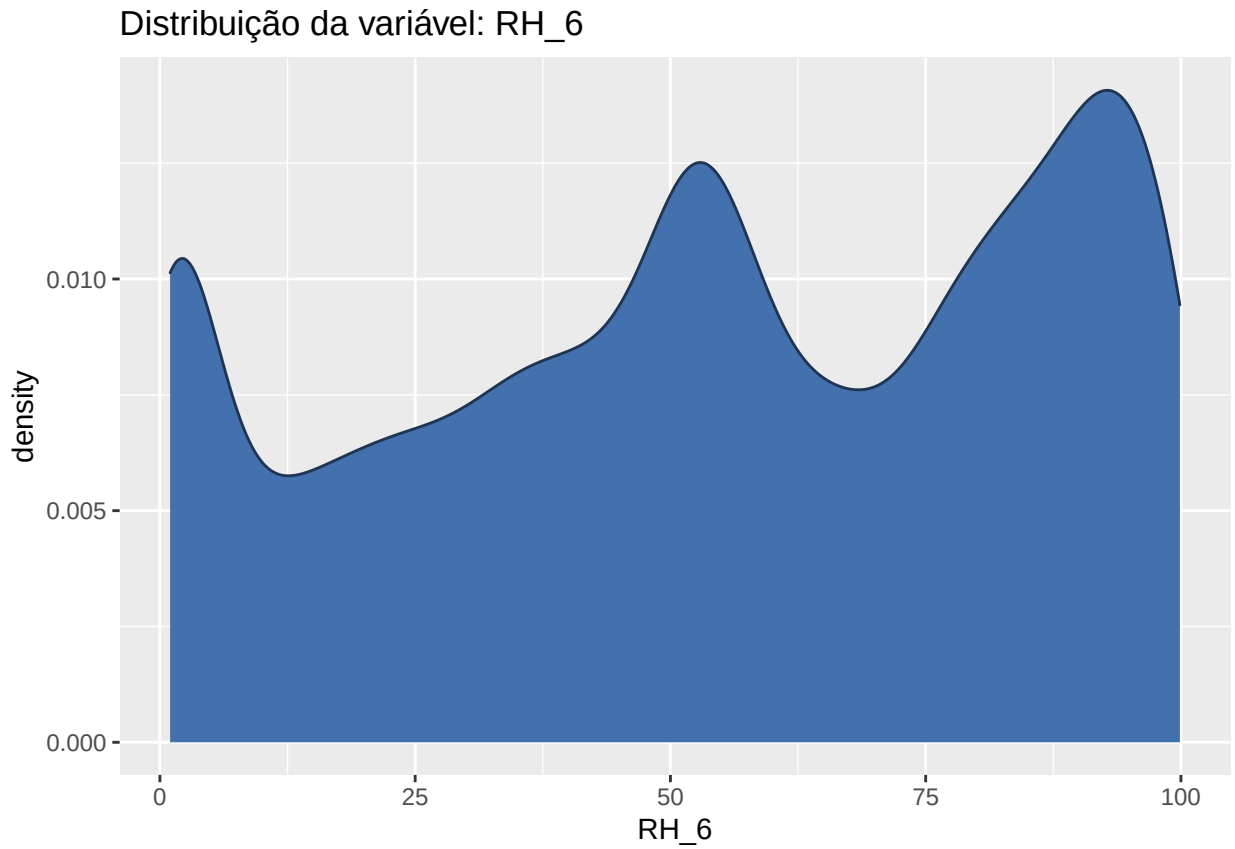

```
box_plot(data = df_train,col = 'T6')
```



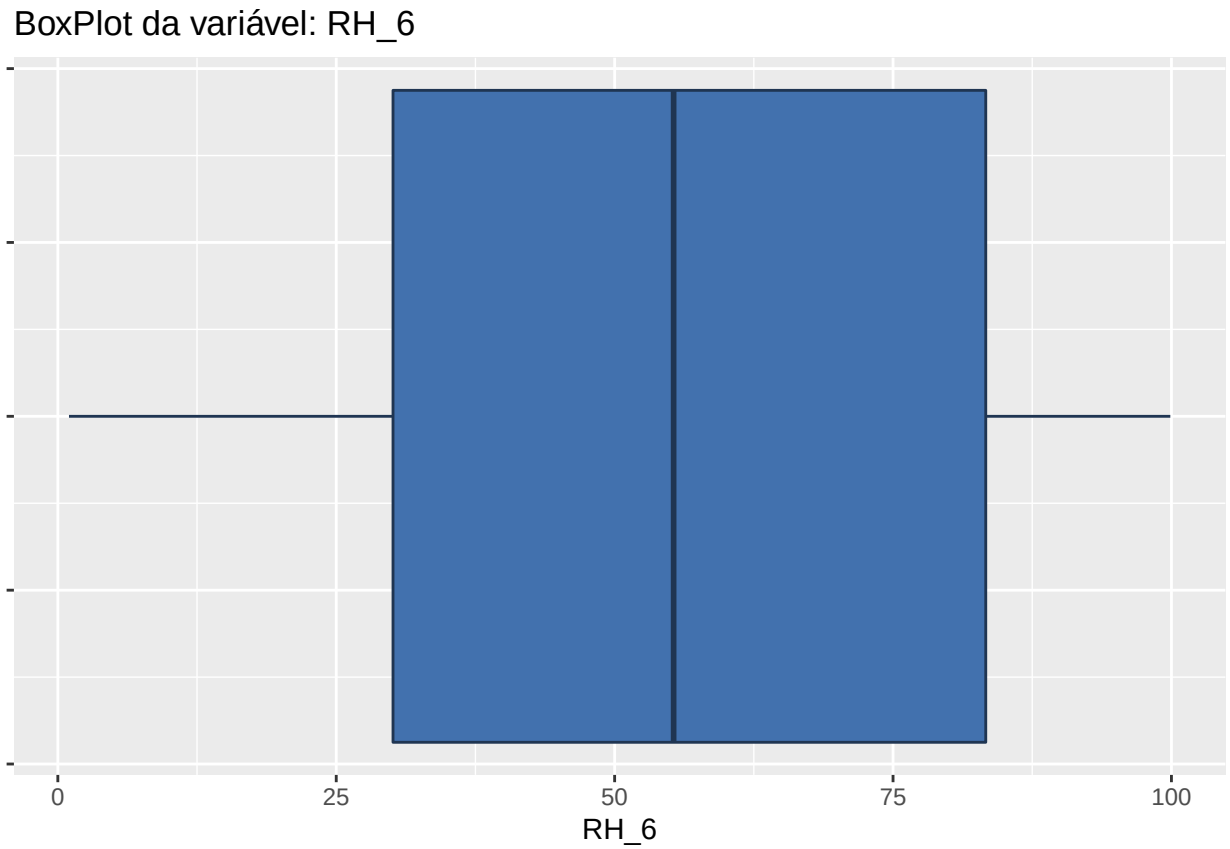
Com o distplot podemos ver que os dados estão bem distribuídos e com o boxplot temos alguns outliers na borda direita.

RH_6

```
dist_plot(data = df_train, col = 'RH_6')
```



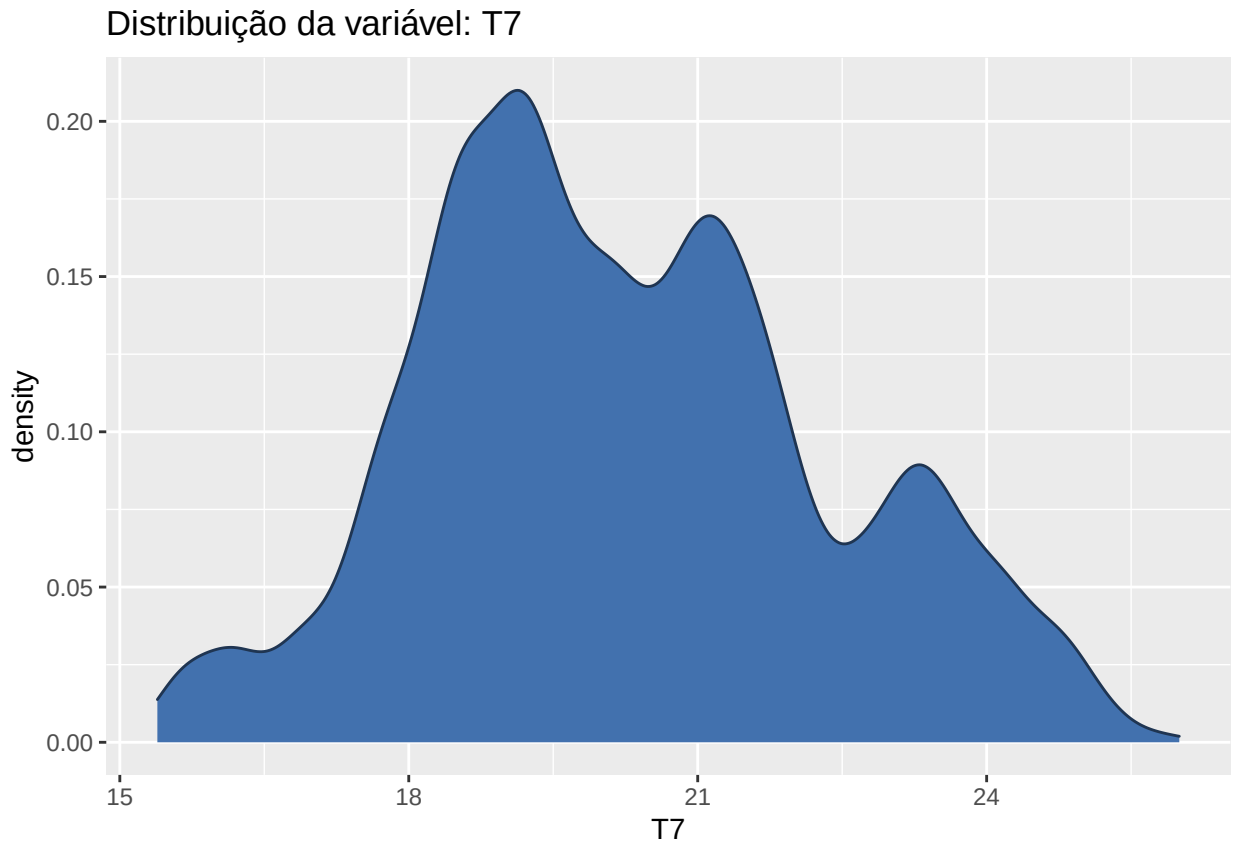

```
box_plot(data = df_train,col = 'RH_6')
```



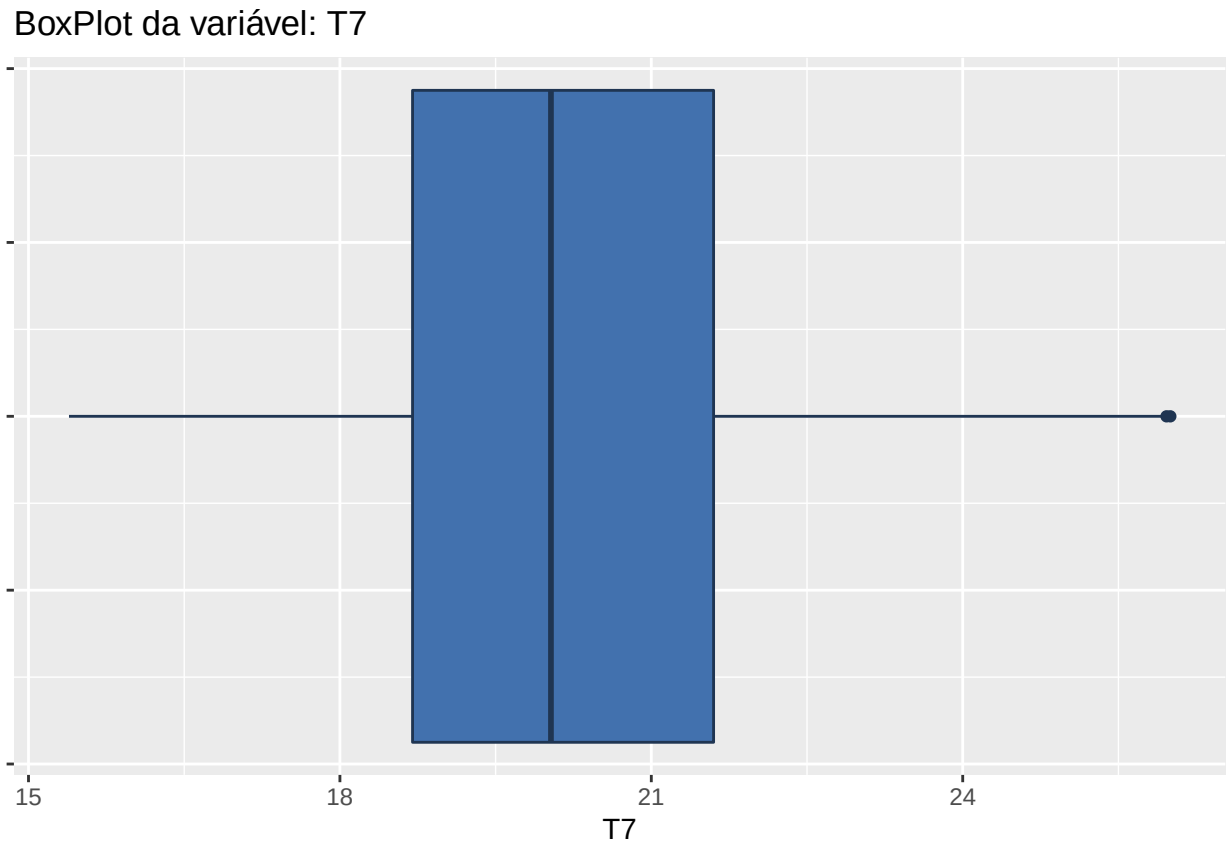
Com o distplot podemos ver que os dados estão totalmente distribuídos e nenhum outlier

T7

```
dist_plot(data = df_train, col = 'T7')
```



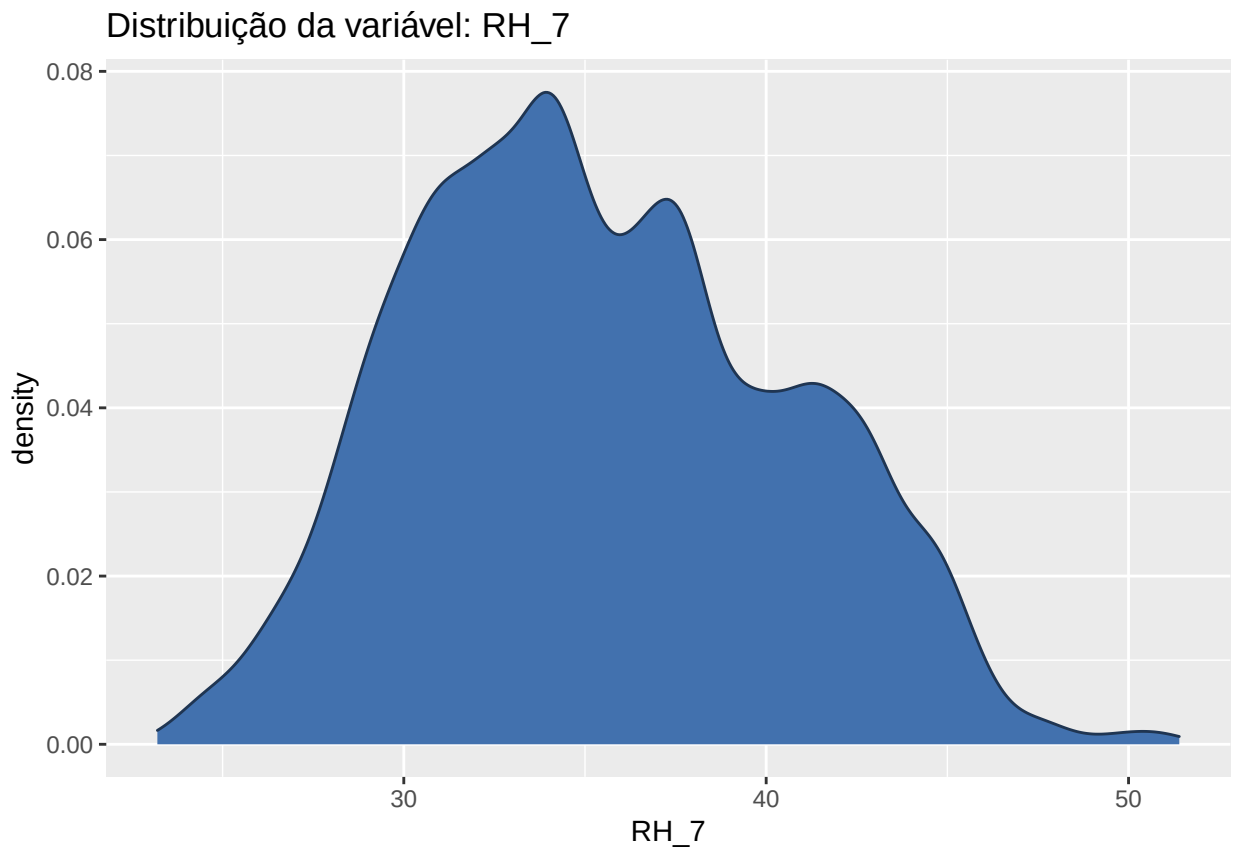

```
box_plot(data = df_train,col = 'T7')
```



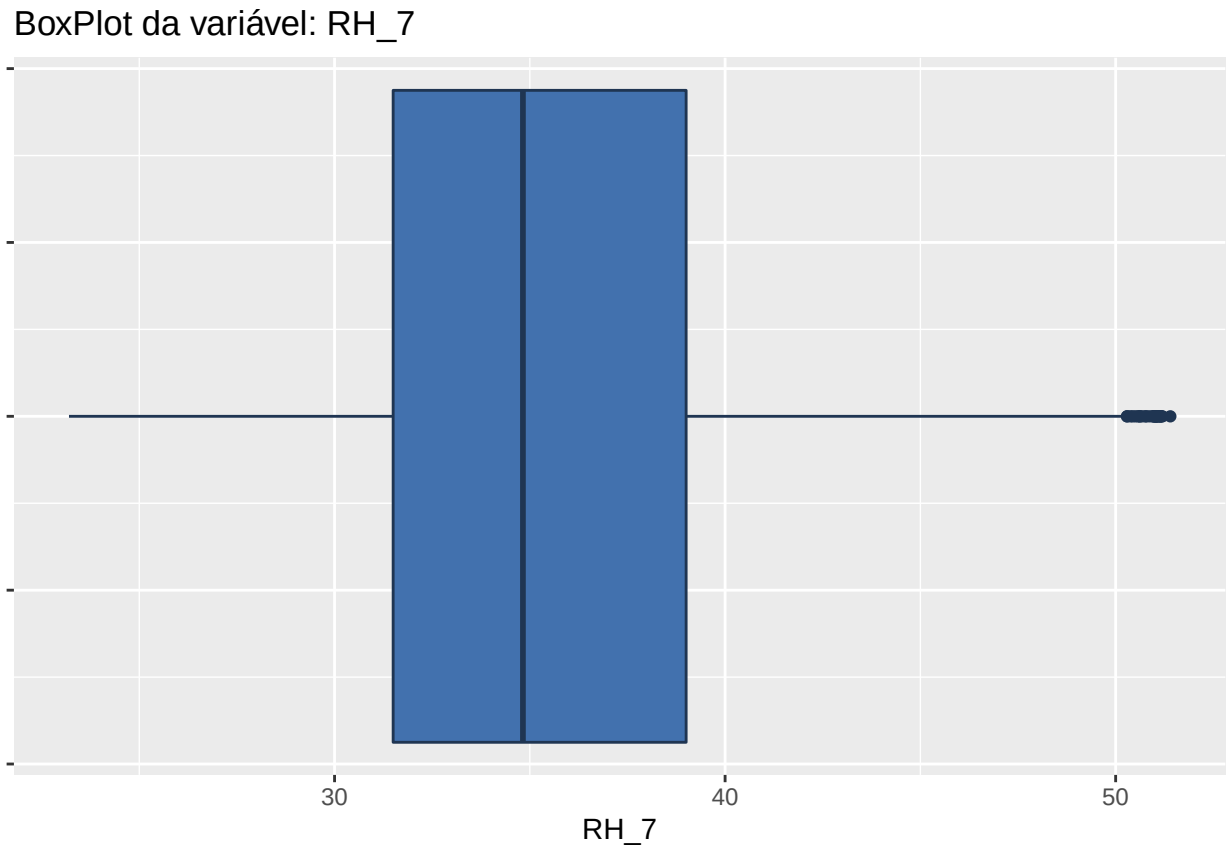
Com o distplot podemos ver que os dados estão bem distribuídos e poucos outliers

RH_7

```
dist_plot(data = df_train, col = 'RH_7')
```



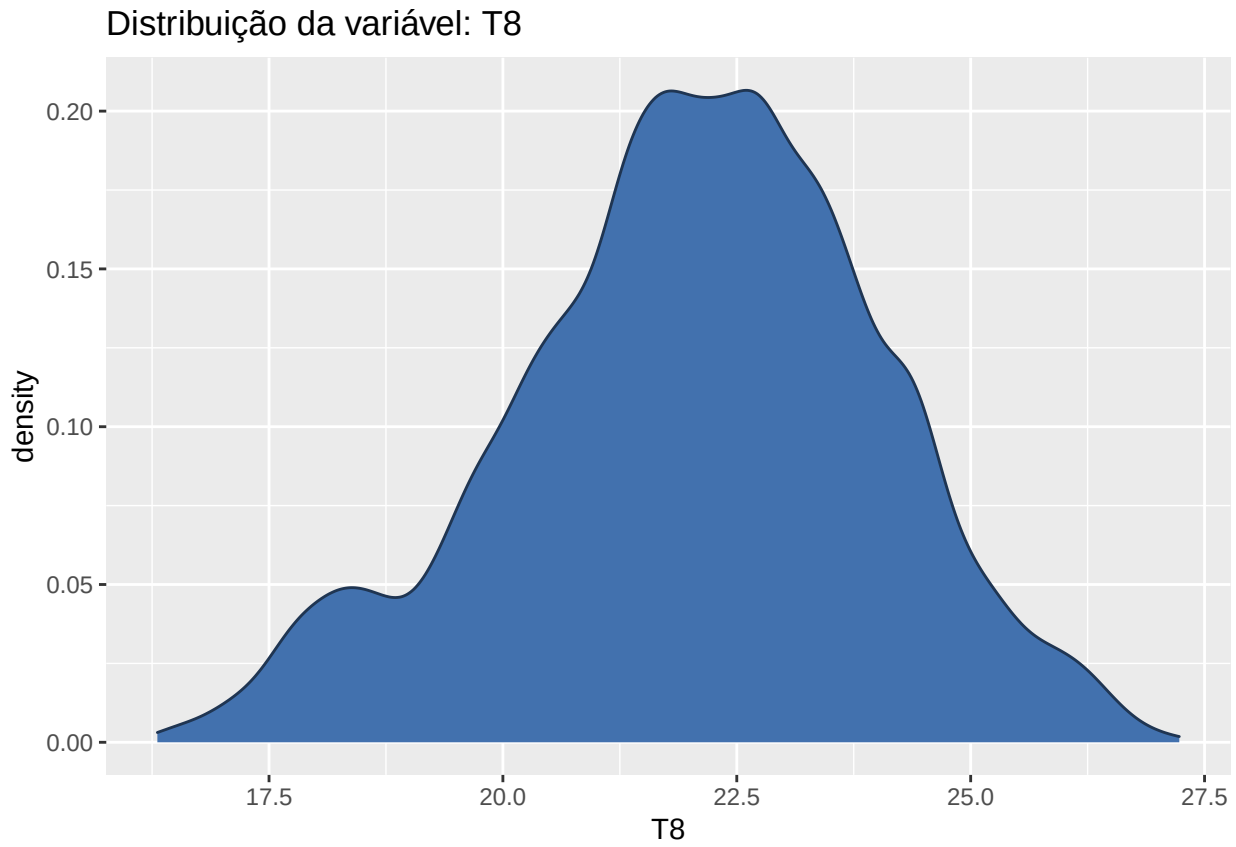

```
box_plot(data = df_train,col = 'RH_7')
```



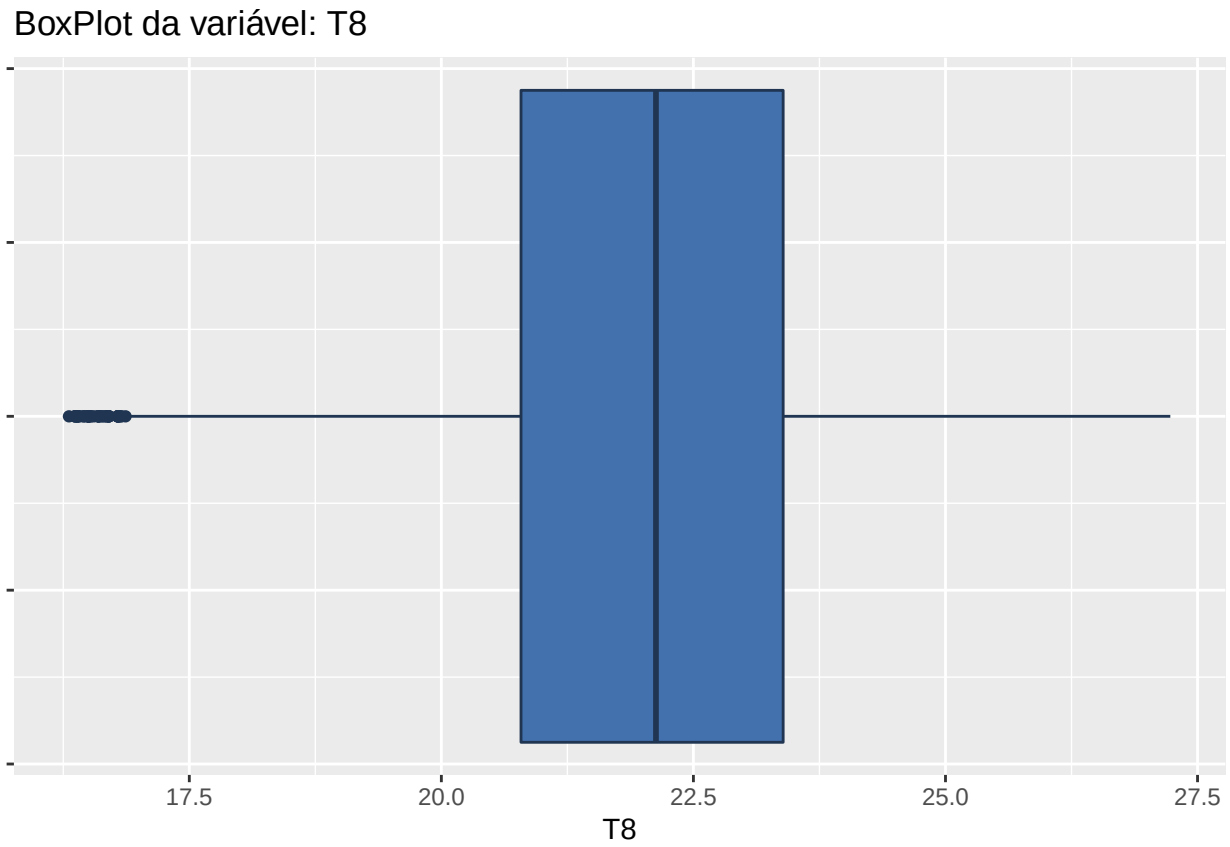
Com o distplot podemos ver que os dados estão bem distribuídos e poucos outliers

T8

```
dist_plot(data = df_train, col = 'T8')
```



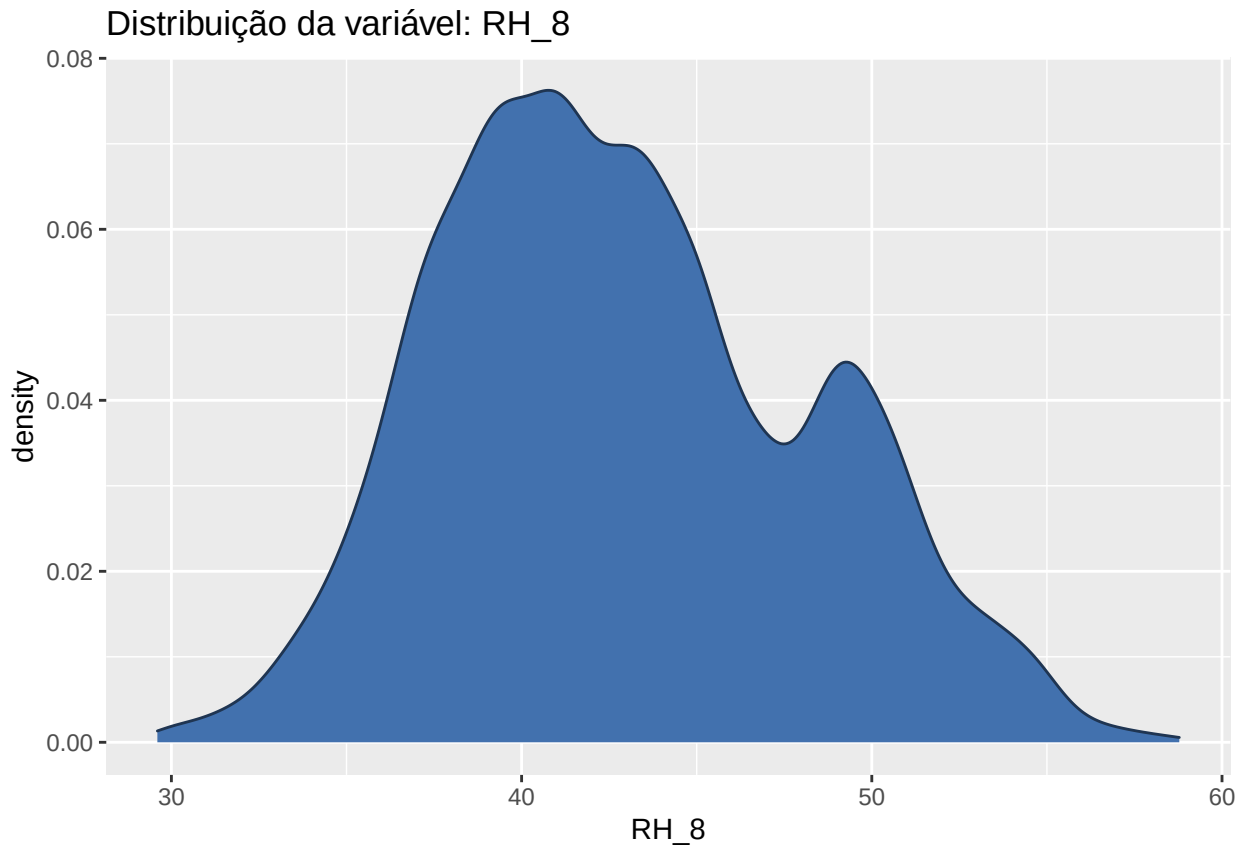

```
box_plot(data = df_train,col = 'T8')
```



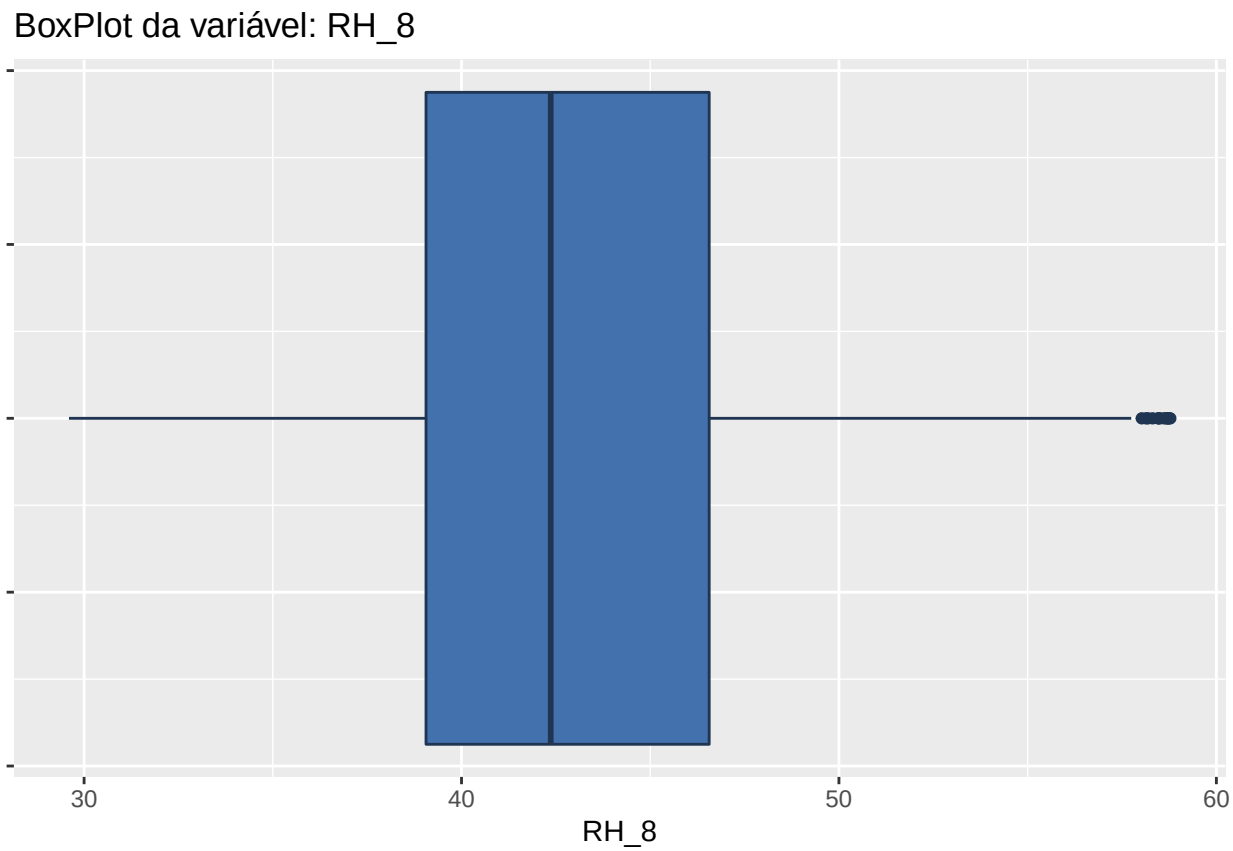
Com o distplot podemos ver que os dados quase simétricos e poucos outliers na borda esquerda como podemos ver com o boxplot.

RH_8

```
dist_plot(data = df_train, col = 'RH_8')
```



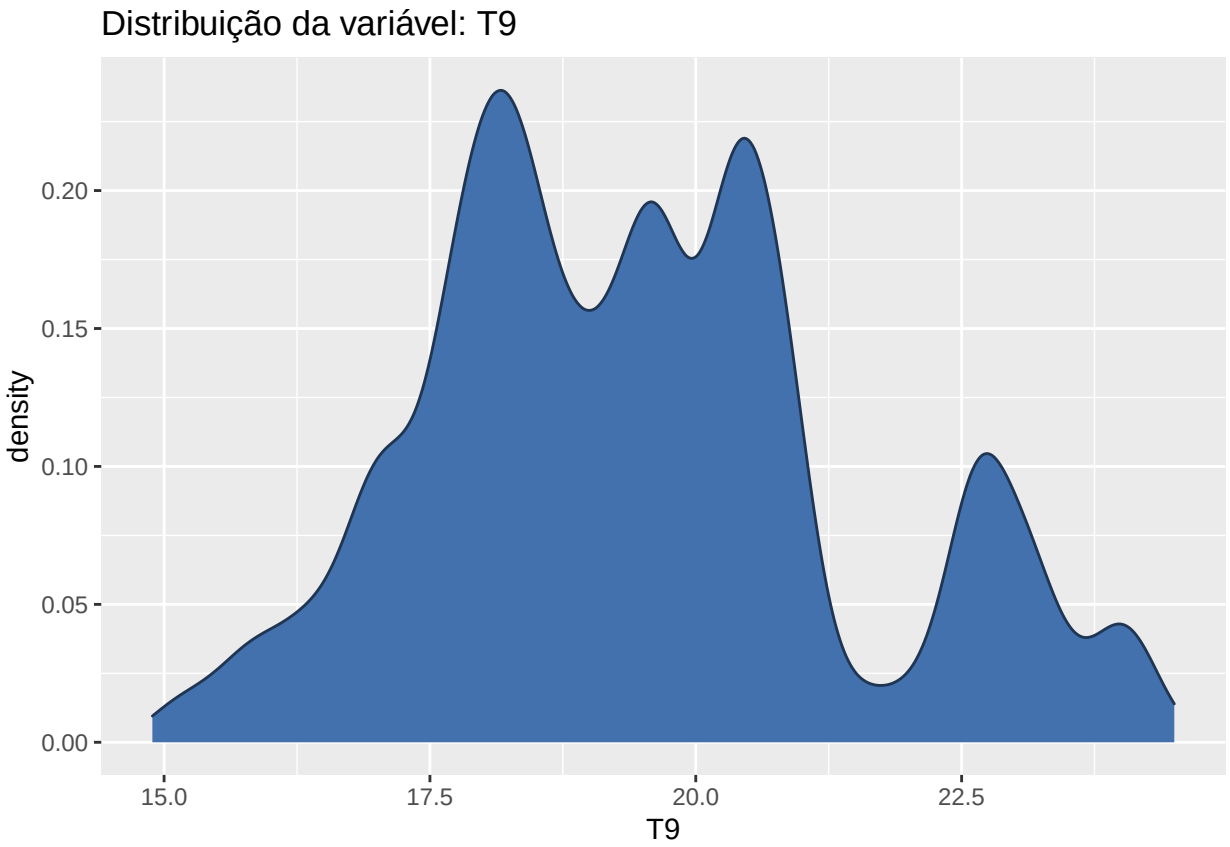

```
box_plot(data = df_train,col = 'RH_8')
```



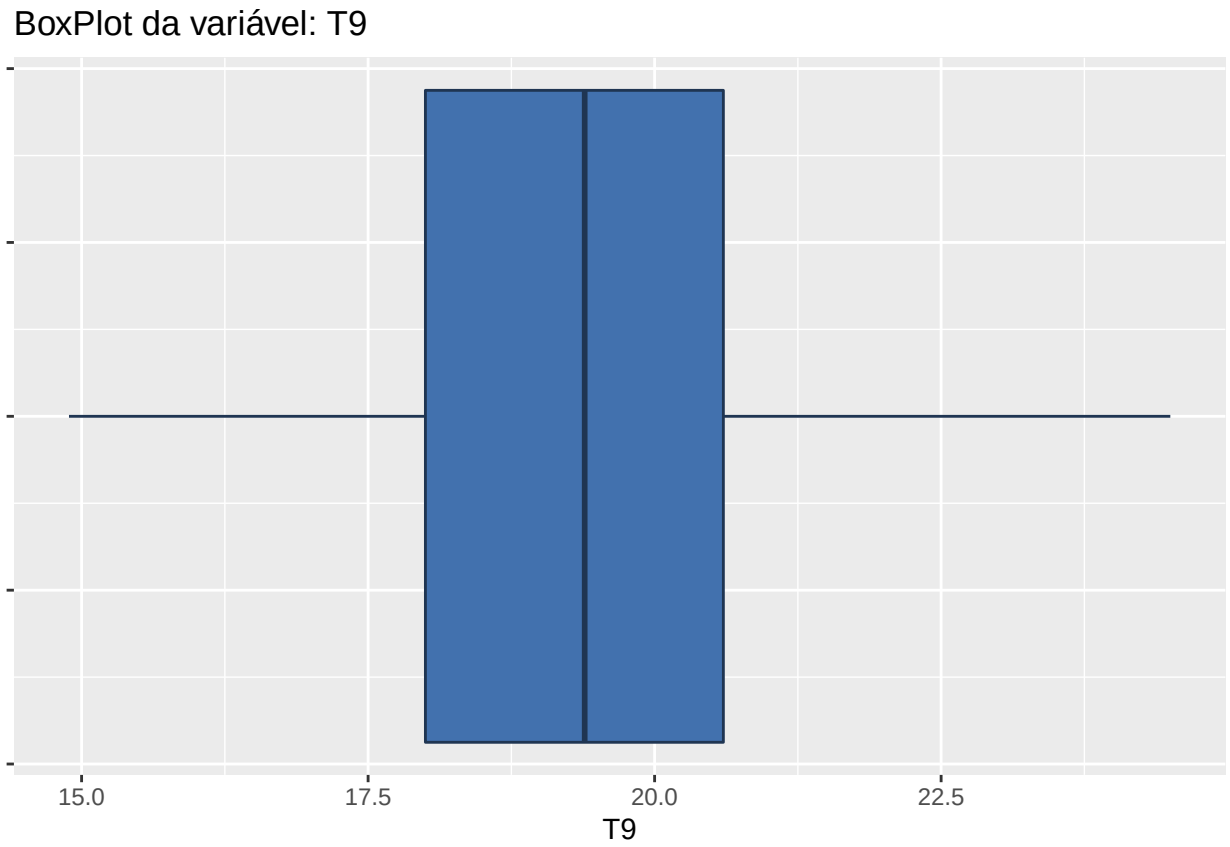
Com o distplot podemos ver que os dados estão bem distribuídos e poucos outliers.

T9

```
dist_plot(data = df_train, col = 'T9')
```



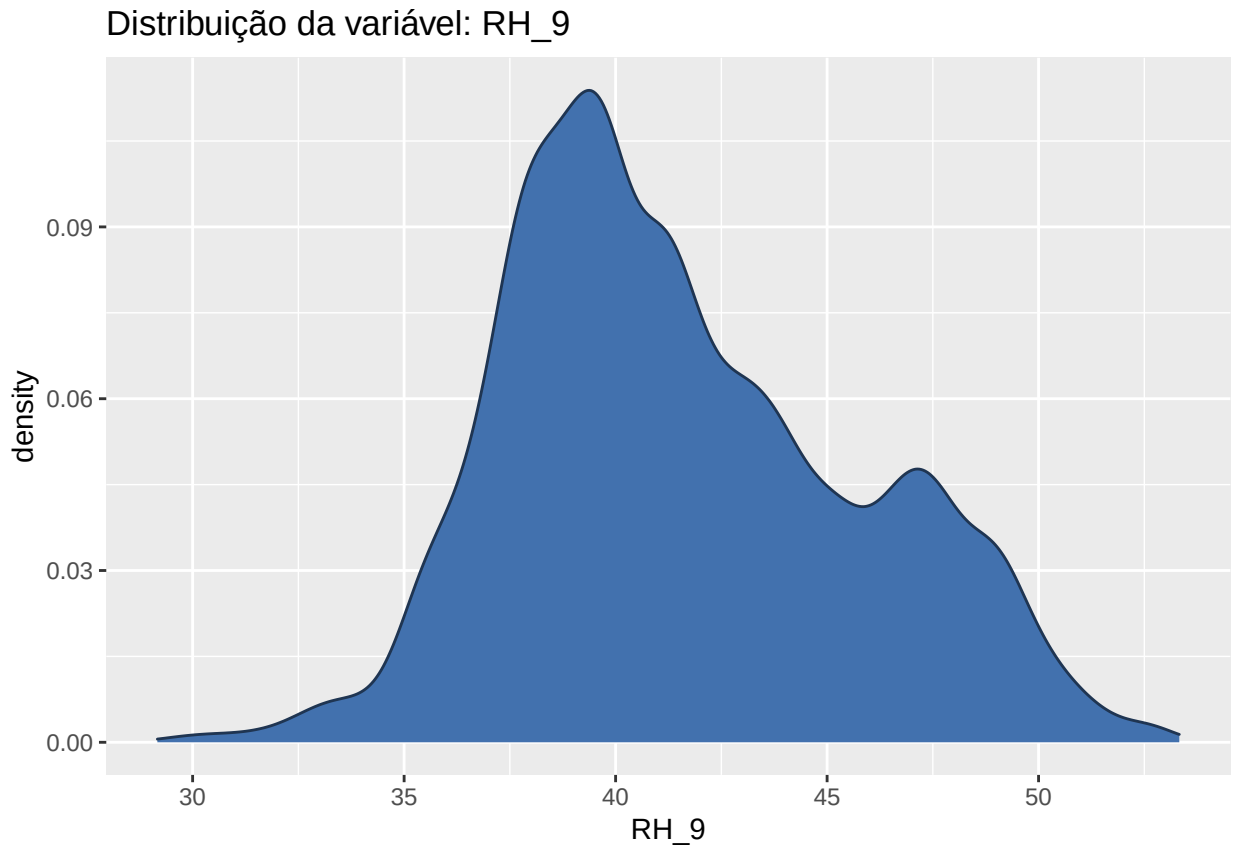

```
box_plot(data = df_train,col = 'T9')
```



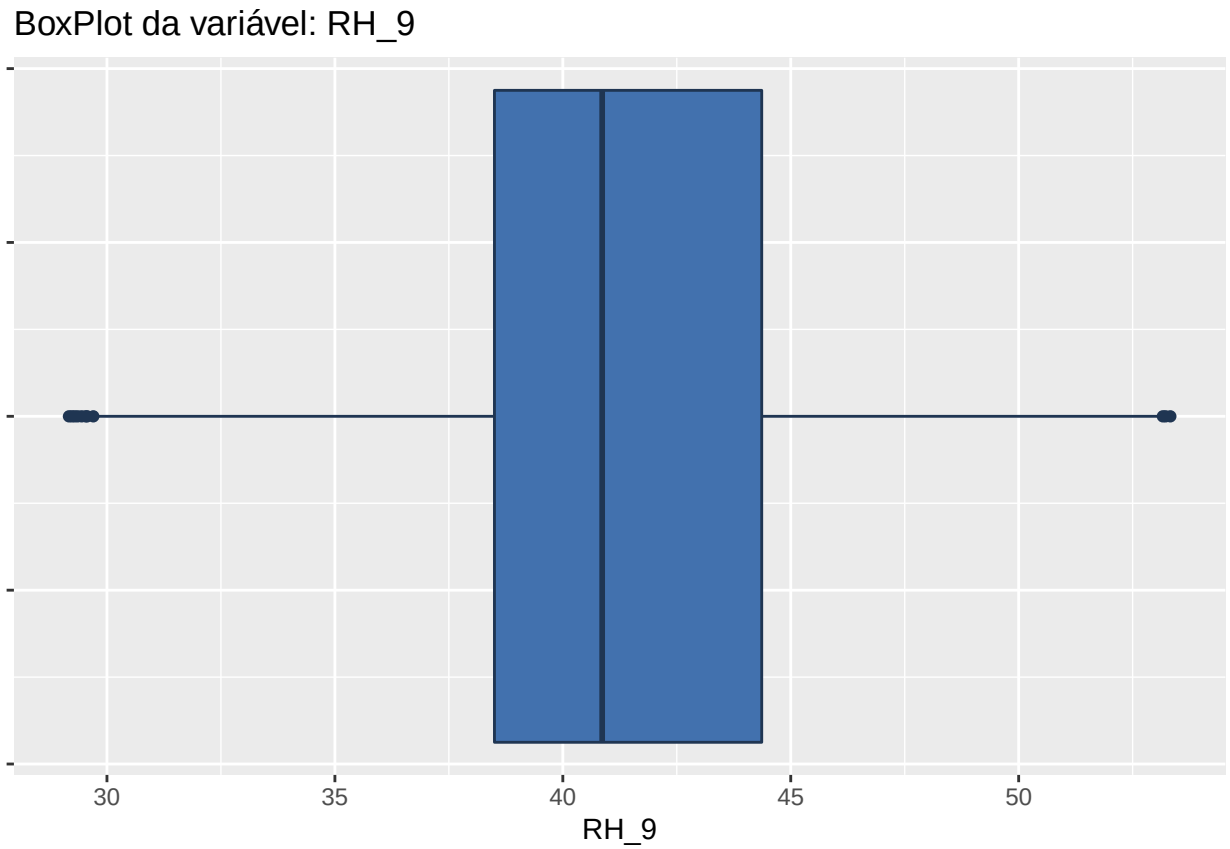
Com o distplot podemos ver que os dados estão bem distribuídos e nenhum outlier visto com o boxplot.

RH_9

```
dist_plot(data = df_train, col = 'RH_9')
```



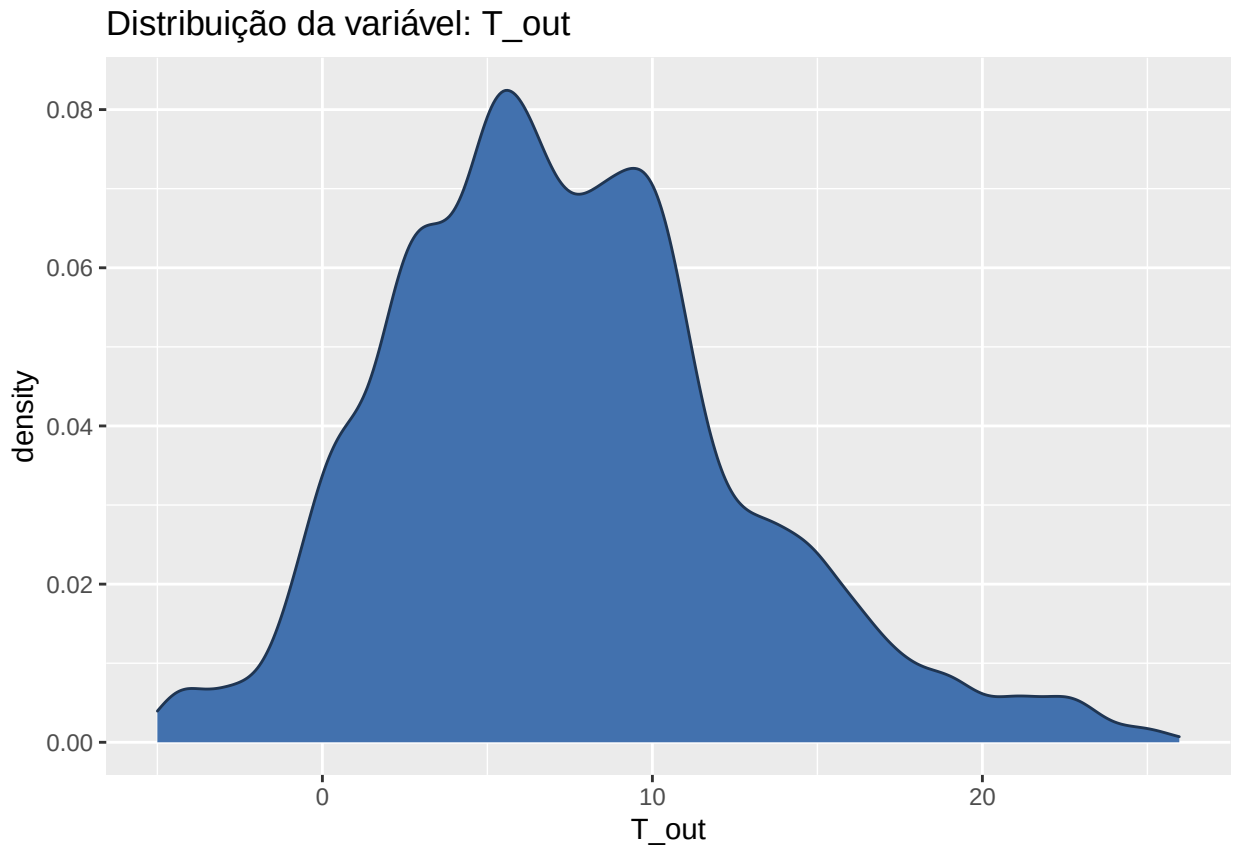

```
box_plot(data = df_train,col = 'RH_9')
```



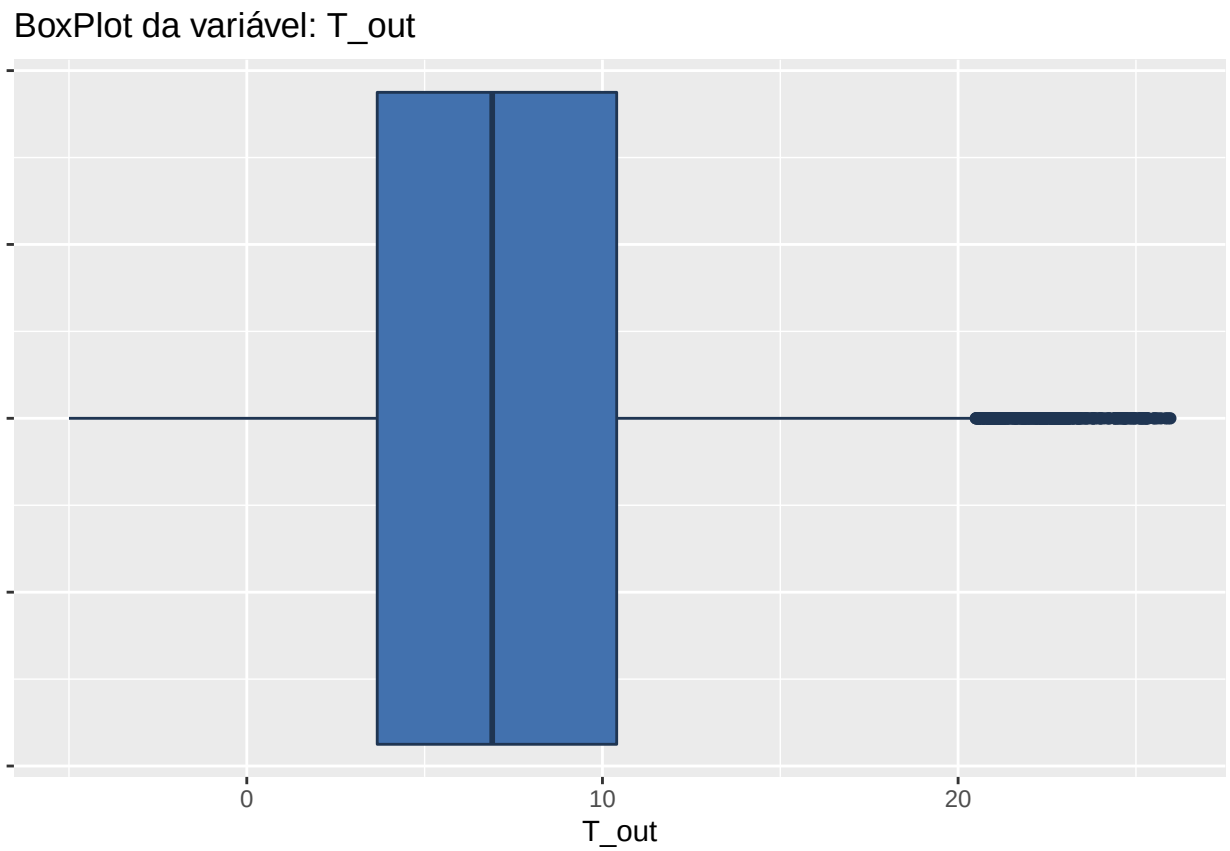
Com o distplot podemos ver que os dados tendem um pouco a direita, e tempos alguns outliers como visto com o boxplot.

T_out

```
dist_plot(data = df_train, col = 'T_out')
```



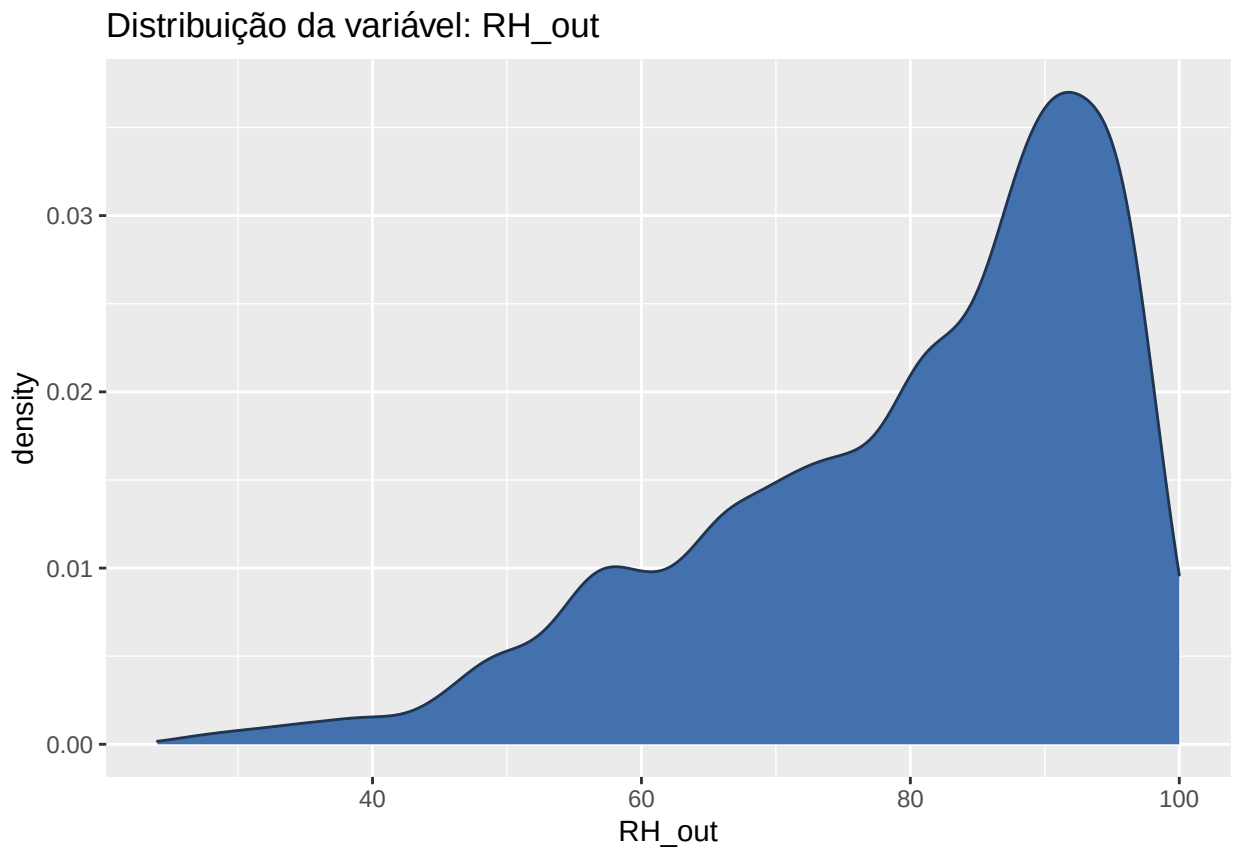

```
box_plot(data = df_train,col = 'T_out')
```



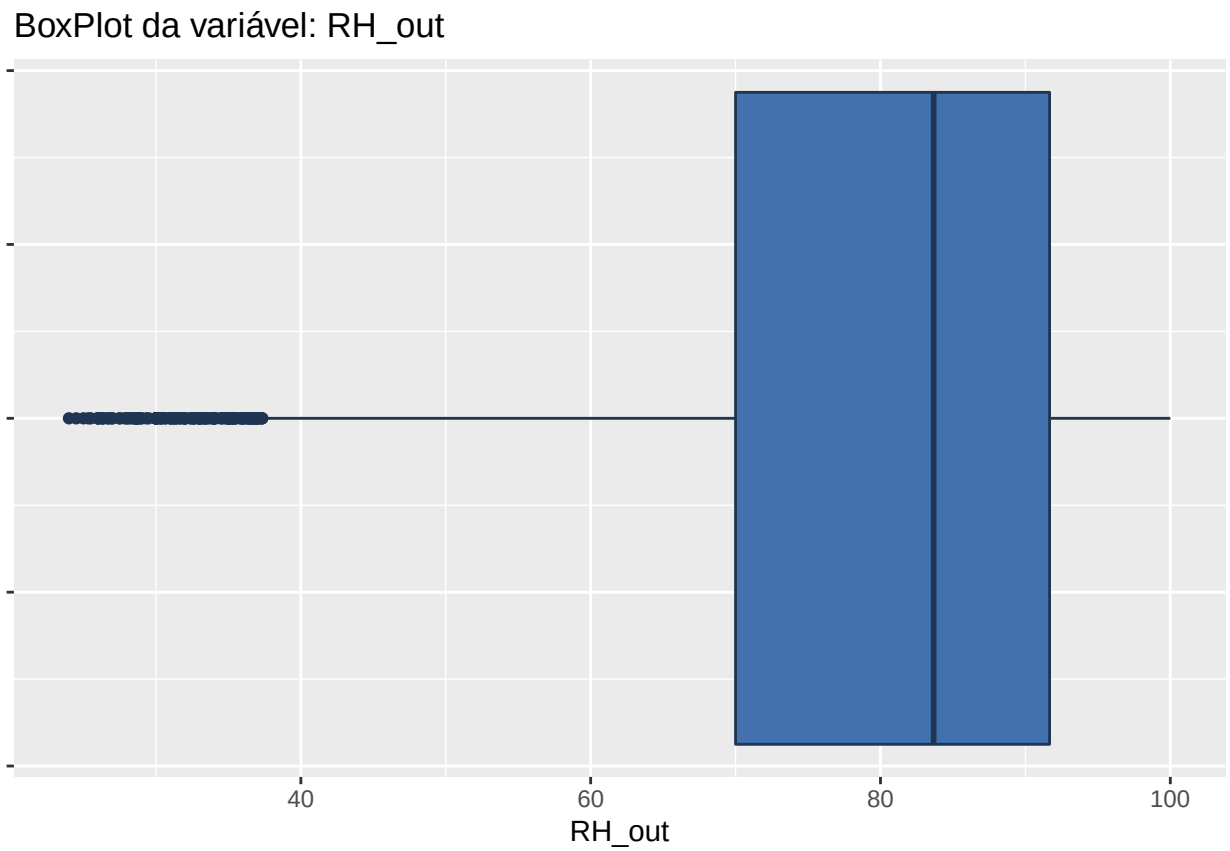
Com o distplot podemos ver que os dados tendem um pouco a esquerda, e tempos alguns outliers como visto com o boxplot na borda direita.

RH_out

```
dist_plot(data = df_train, col = 'RH_out')
```



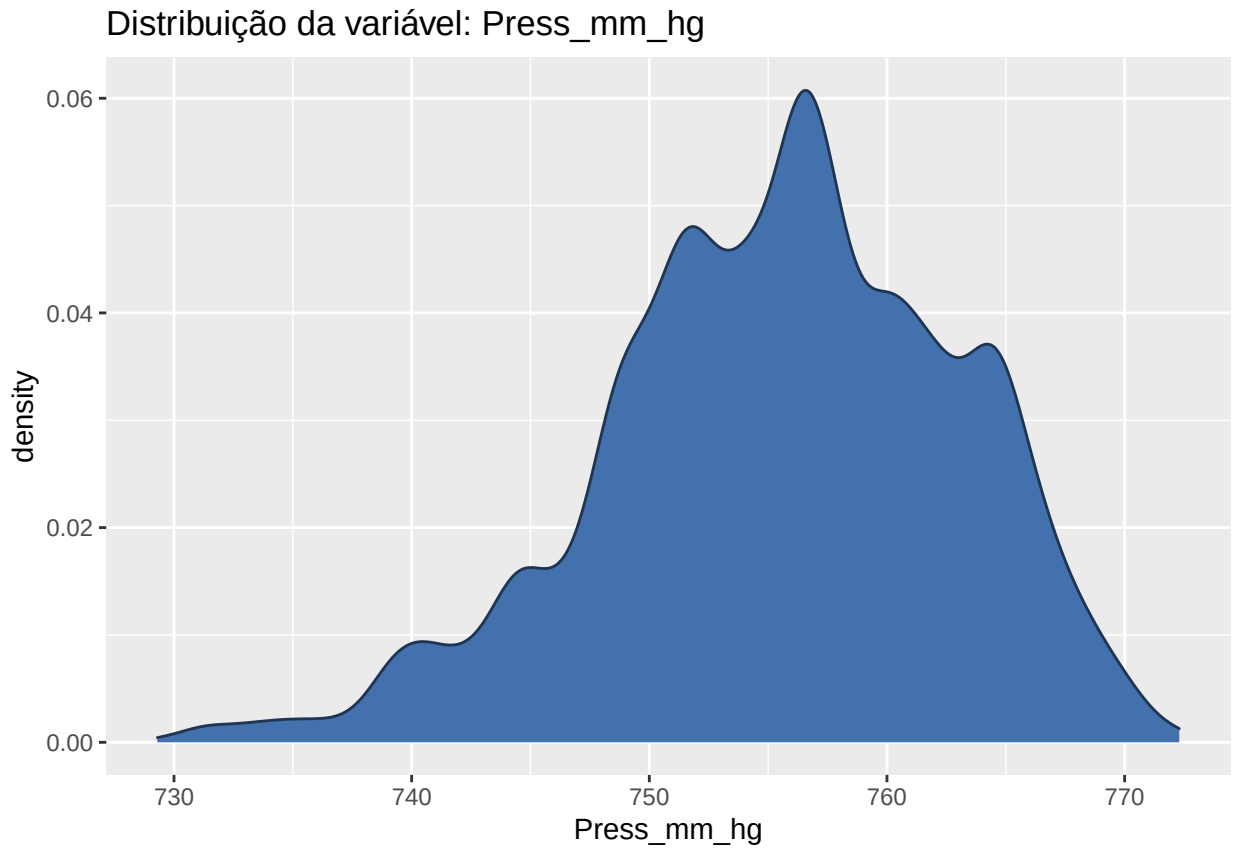

```
box_plot(data = df_train,col = 'RH_out')
```



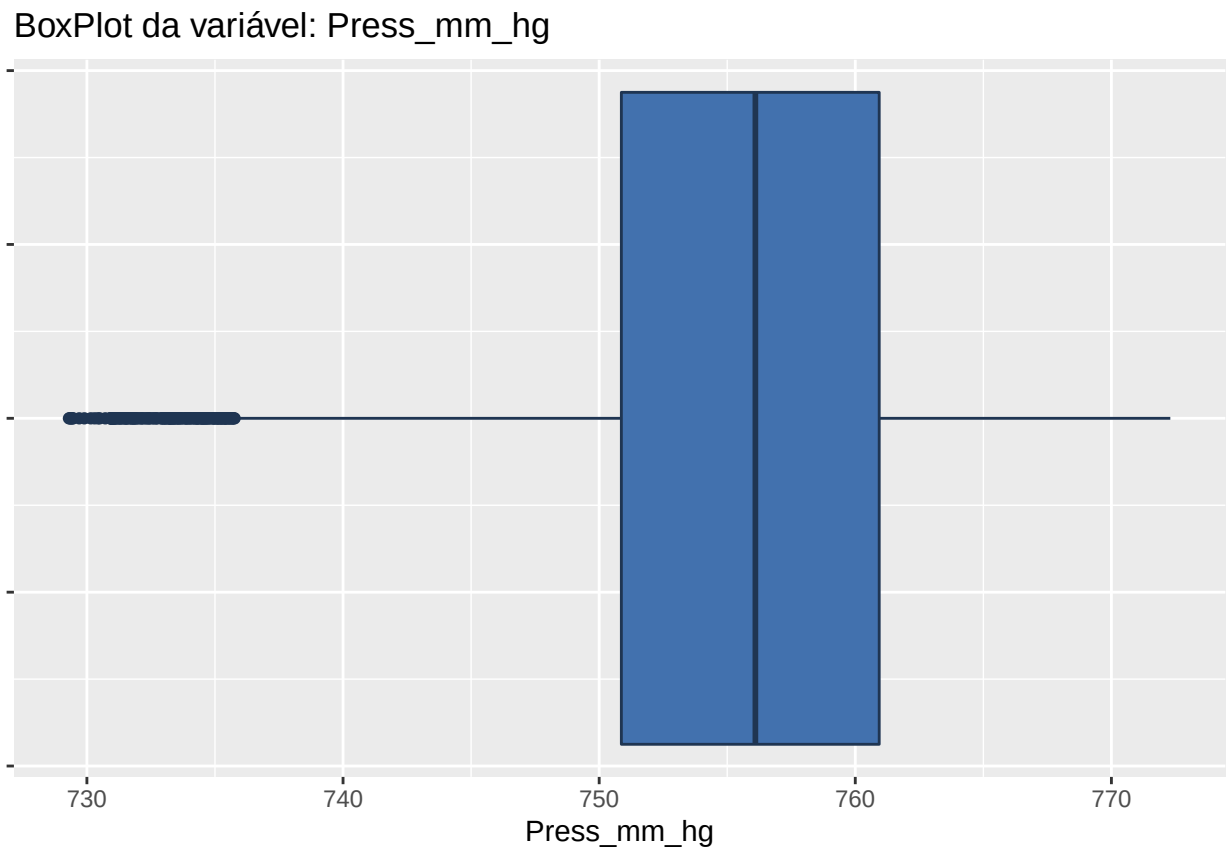
Com o distplot podemos ver uma assimetria nos dados, onde se encontram mais na parte direita, e com a ajuda do boxplot podemos ver outliers a na borda esquerda.

Press_mm_hg

```
dist_plot(data = df_train, col = 'Press_mm_hg')
```



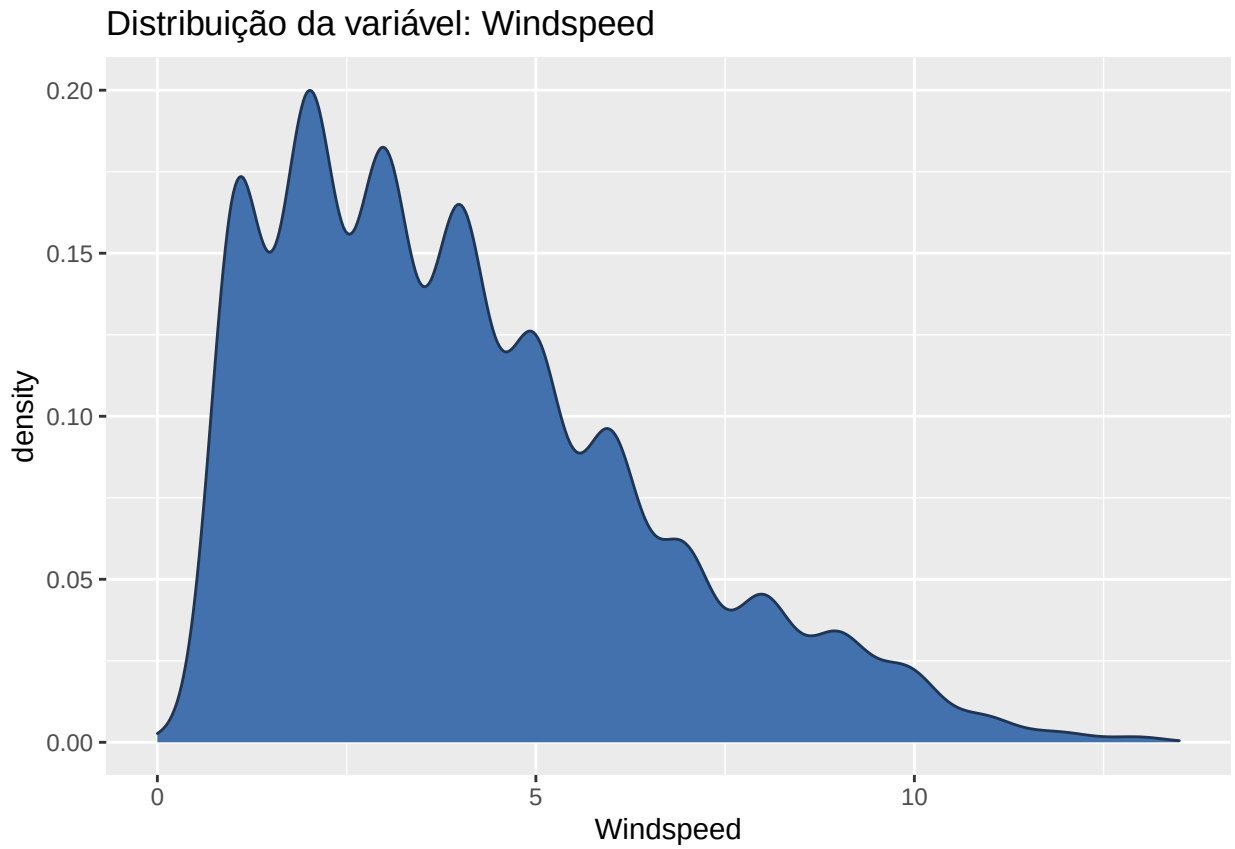

```
box_plot(data = df_train,col = 'Press_mm_hg')
```



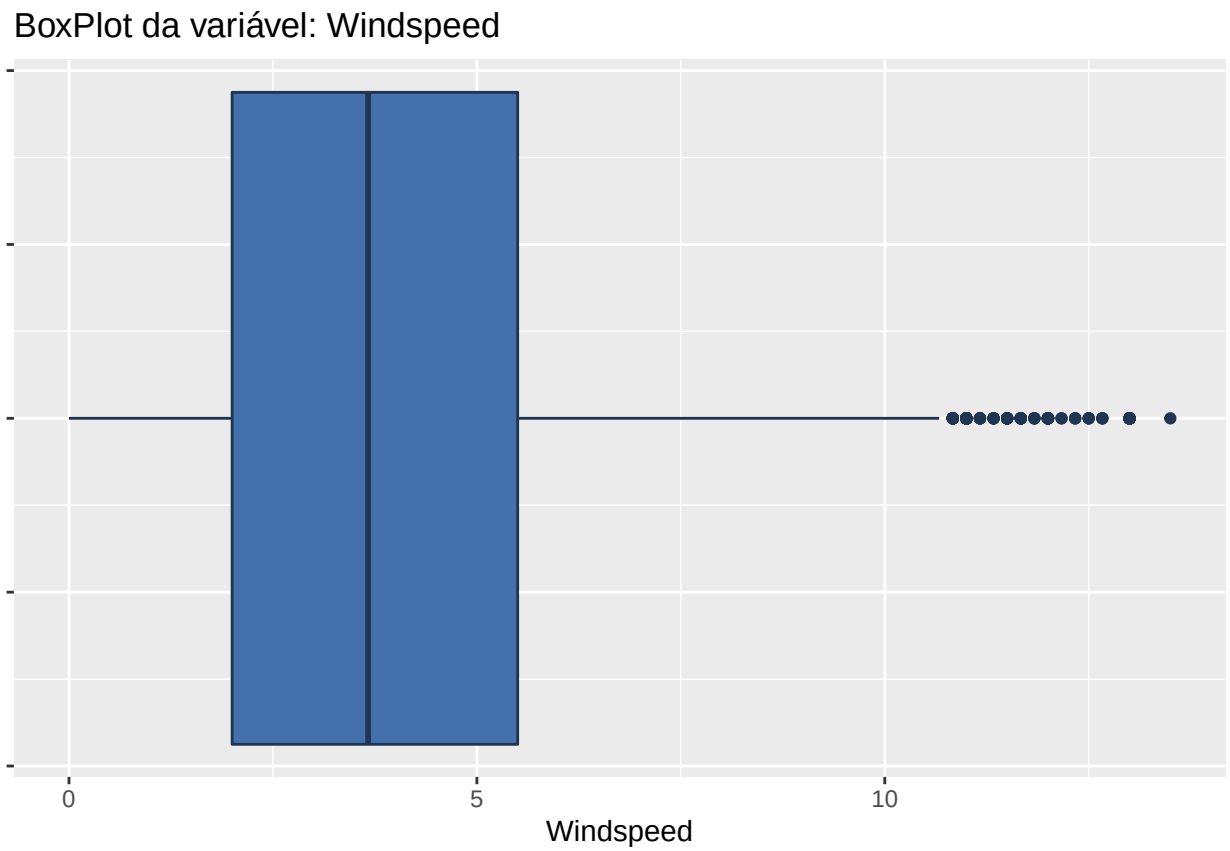
Com o distplot podemos ver uma assimetria nos dados, onde se encontram mais na parte direita, e com a ajuda do boxplot podemos ver outliers a na borda esquerda.

Windspeed

```
dist_plot(data = df_train, col = 'Windspeed')
```



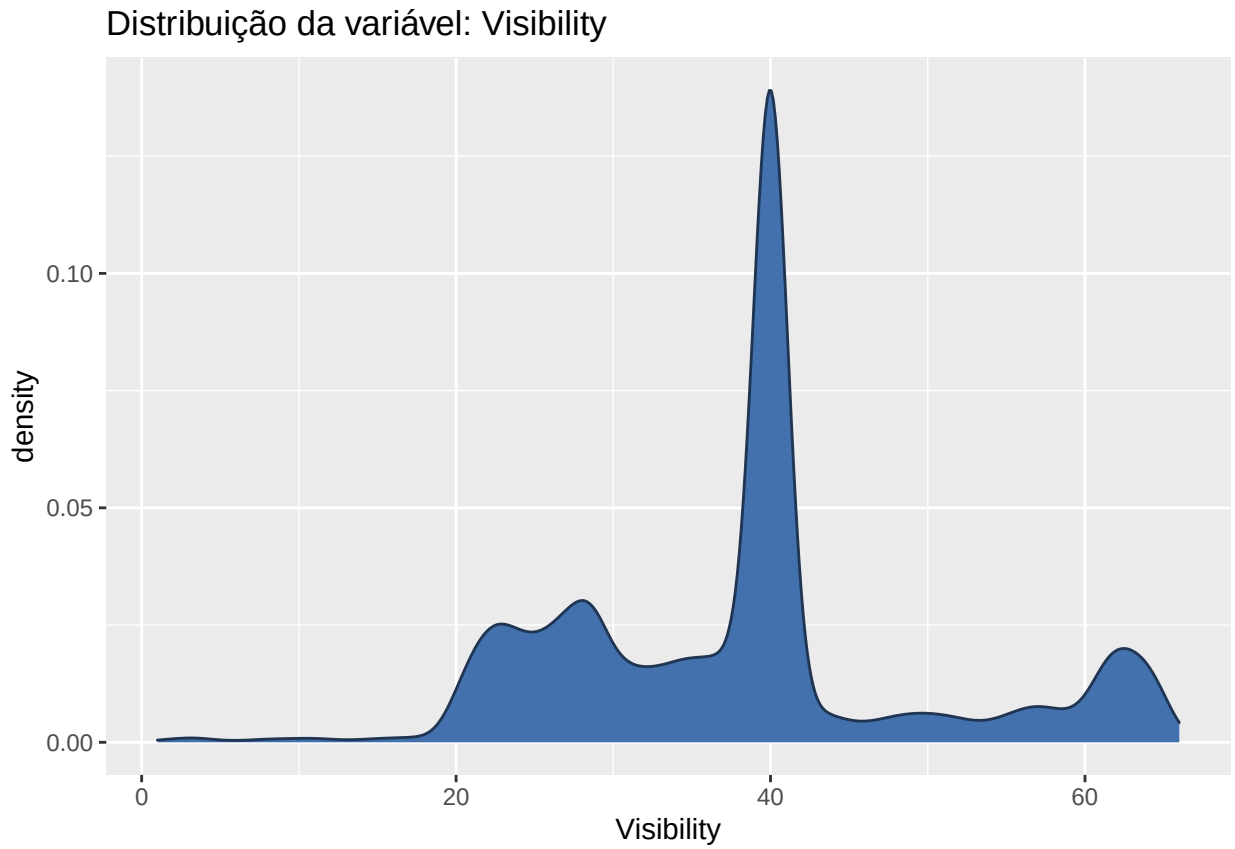

```
box_plot(data = df_train,col = 'Windspeed')
```



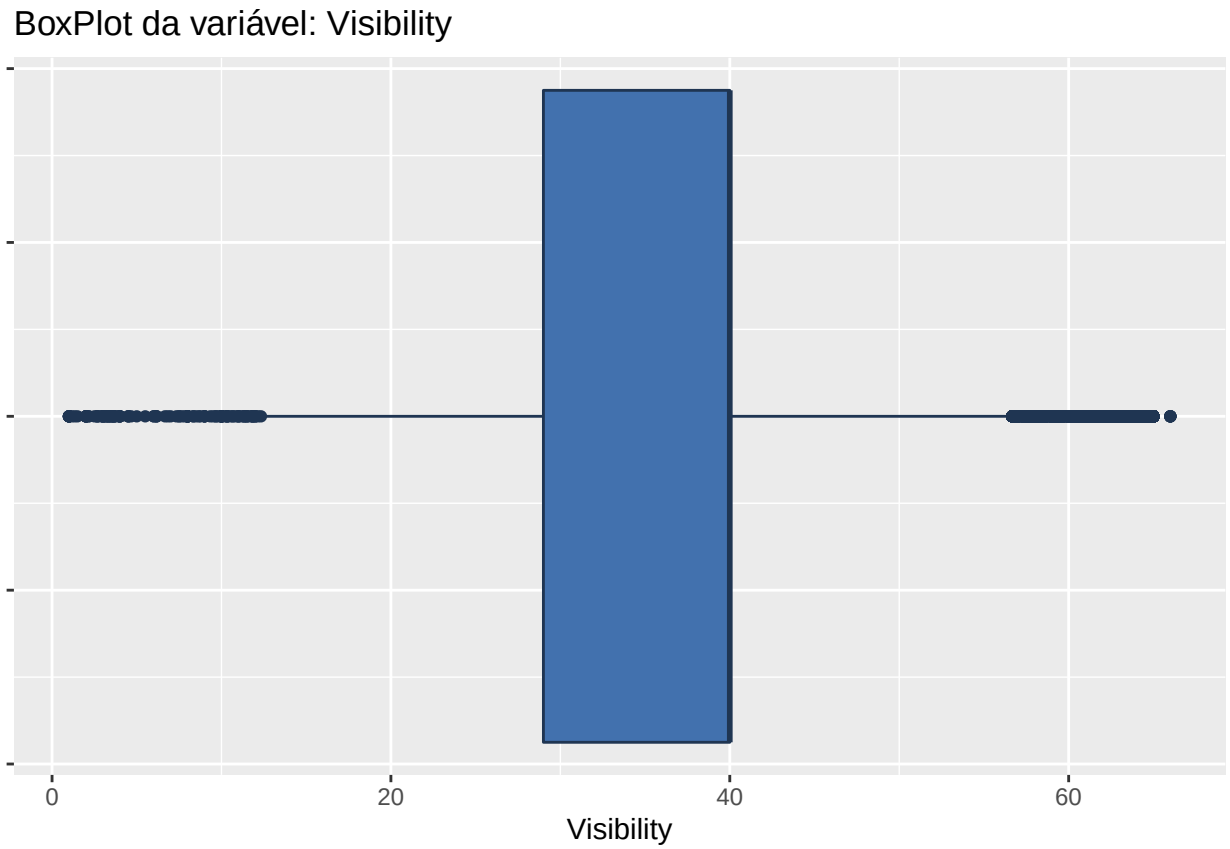
Com o distplot podemos ver que os dados tendem um pouco a esquerda, e tempos alguns outliers como visto com o boxplot na borda direita.

Visibility

```
dist_plot(data = df_train, col = 'Visibility')
```



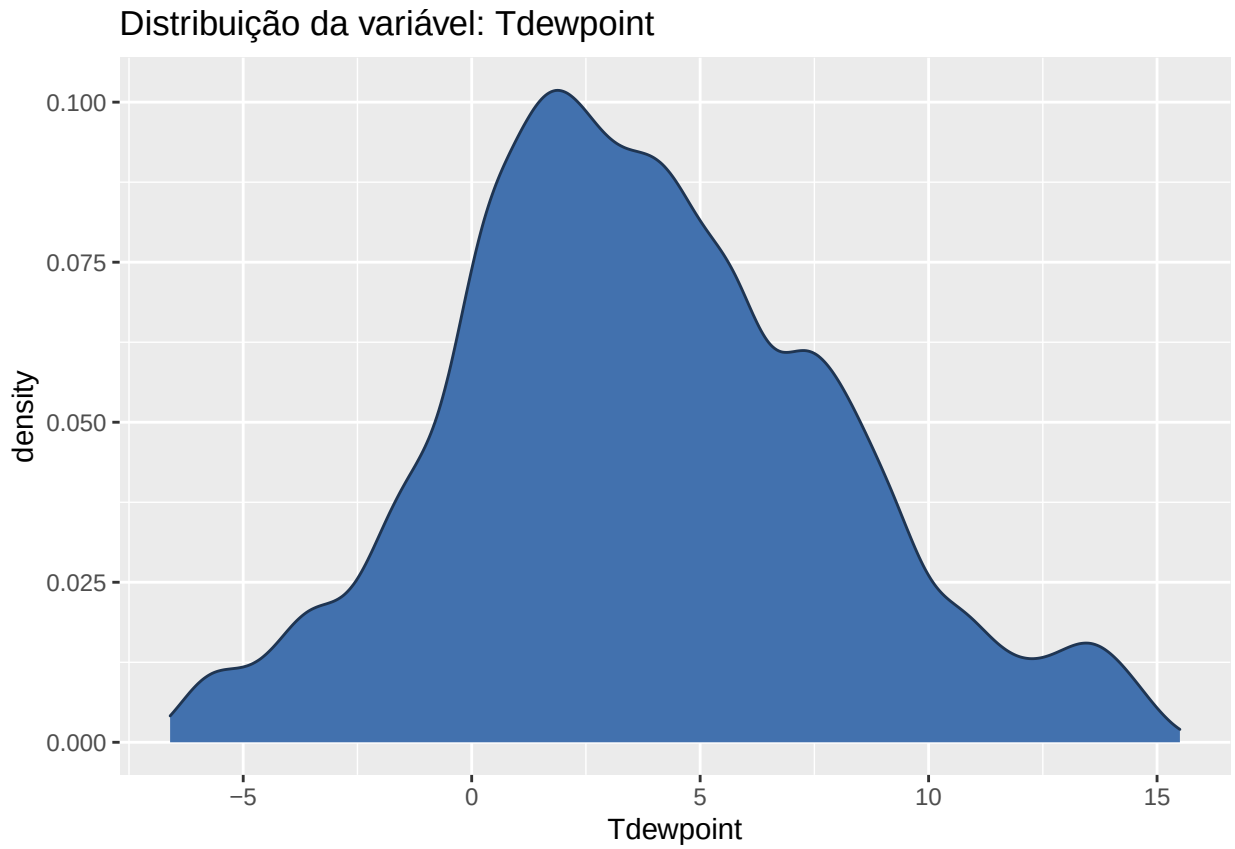

```
box_plot(data = df_train,col = 'Visibility')
```



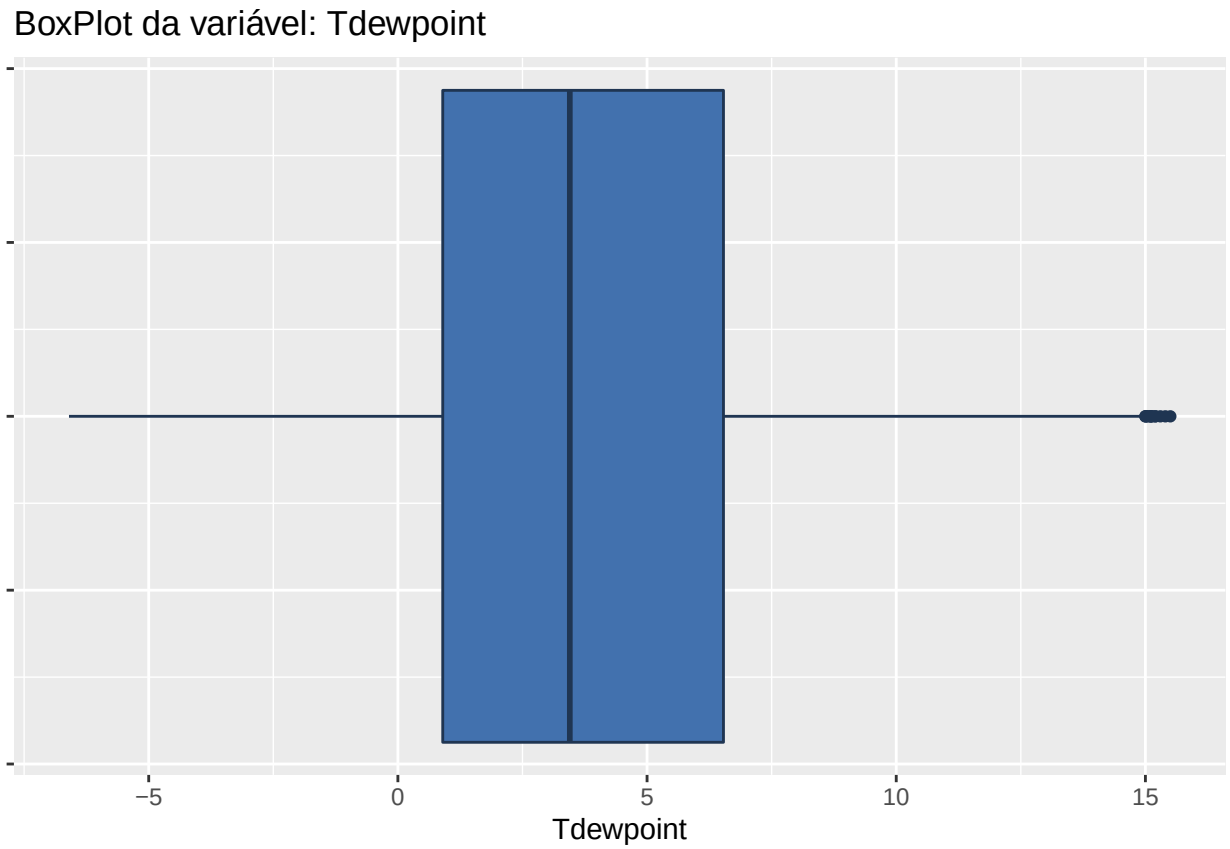
Com o distplot podemos ver que os dados estão bem distribuídos com uma pequena acentuação perto do valor 40 e com o boxplot temos outliers tanto na borda esquerda quanto a direita.

Tdewpoint

```
dist_plot(data = df_train, col = 'Tdewpoint')
```



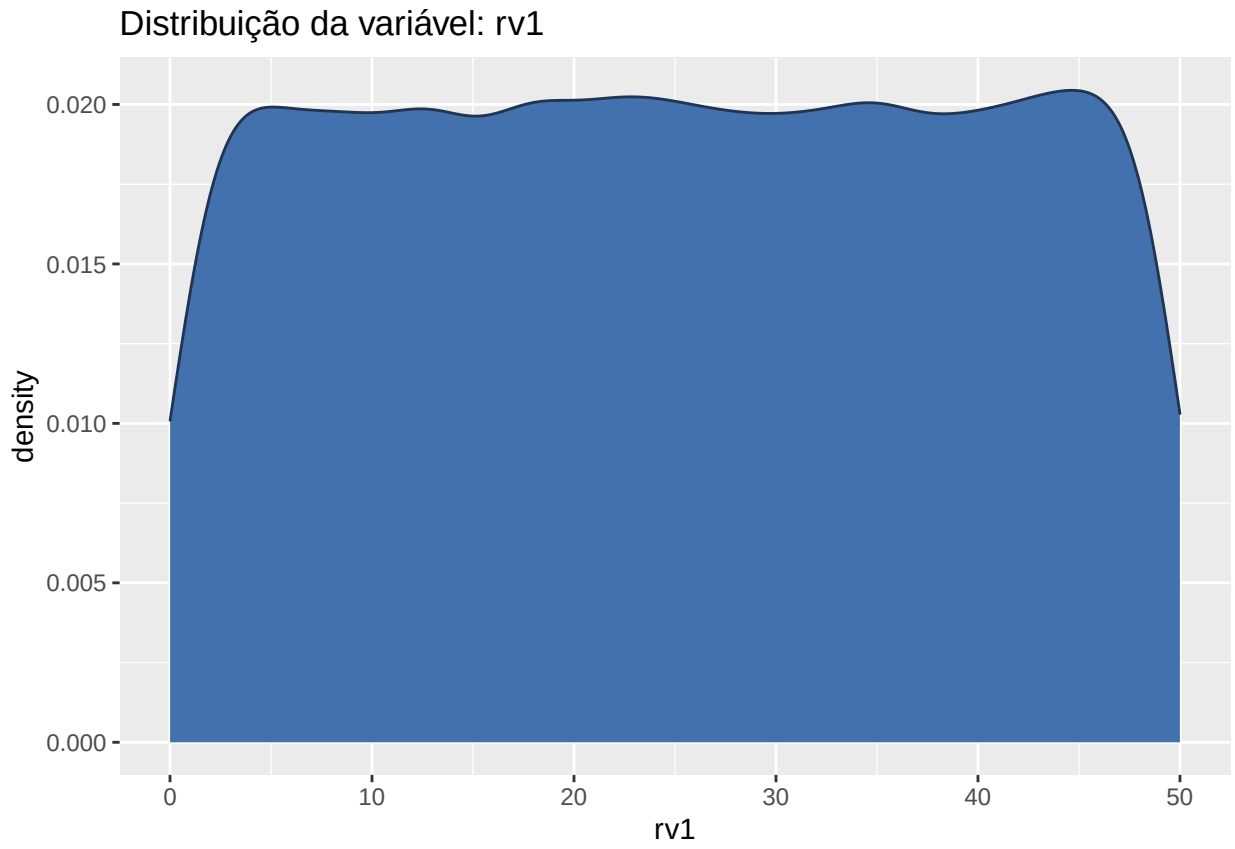

```
box_plot(data = df_train,col = 'Tdewpoint')
```



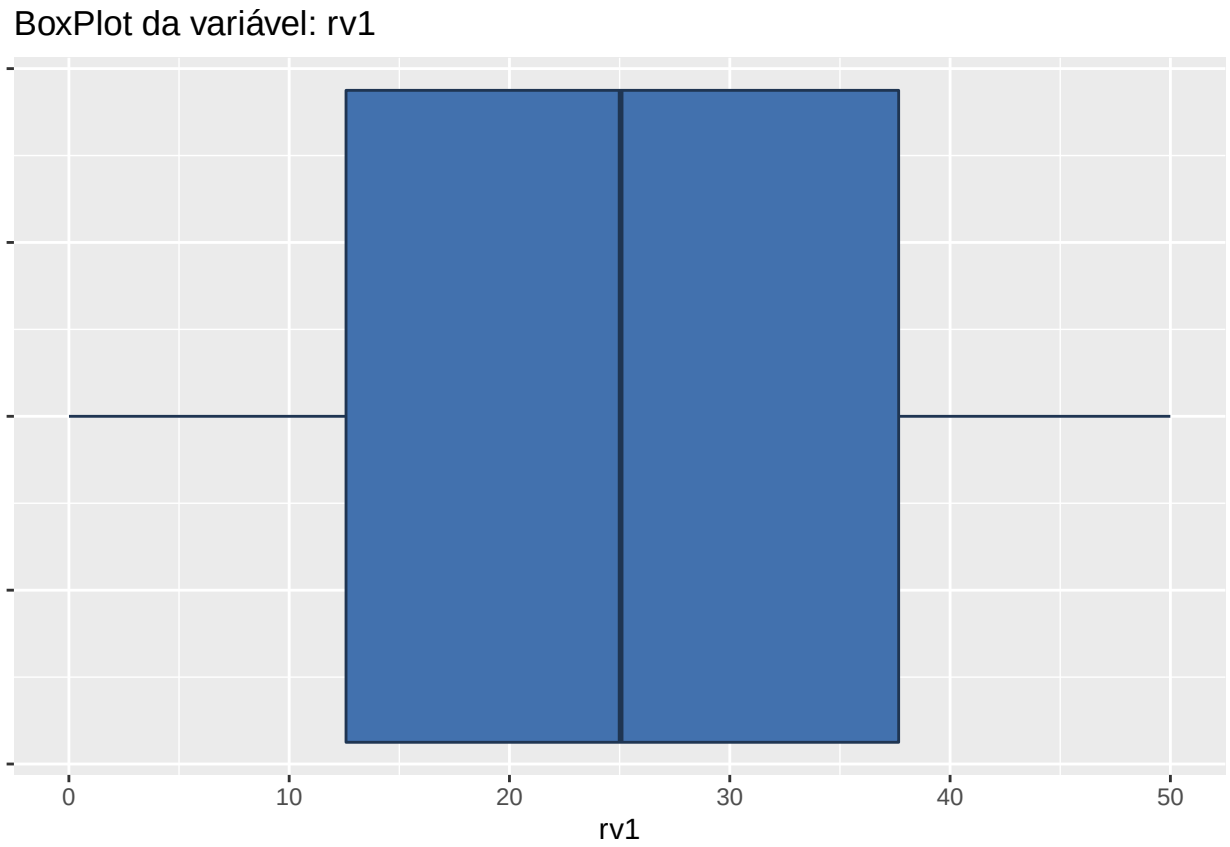
Com o distplot podemos ver que os dados quase simétricos e poucos outliers na borda esquerda como podemos ver com o boxplot.

rv1

```
dist_plot(data = df_train, col = 'rv1')
```



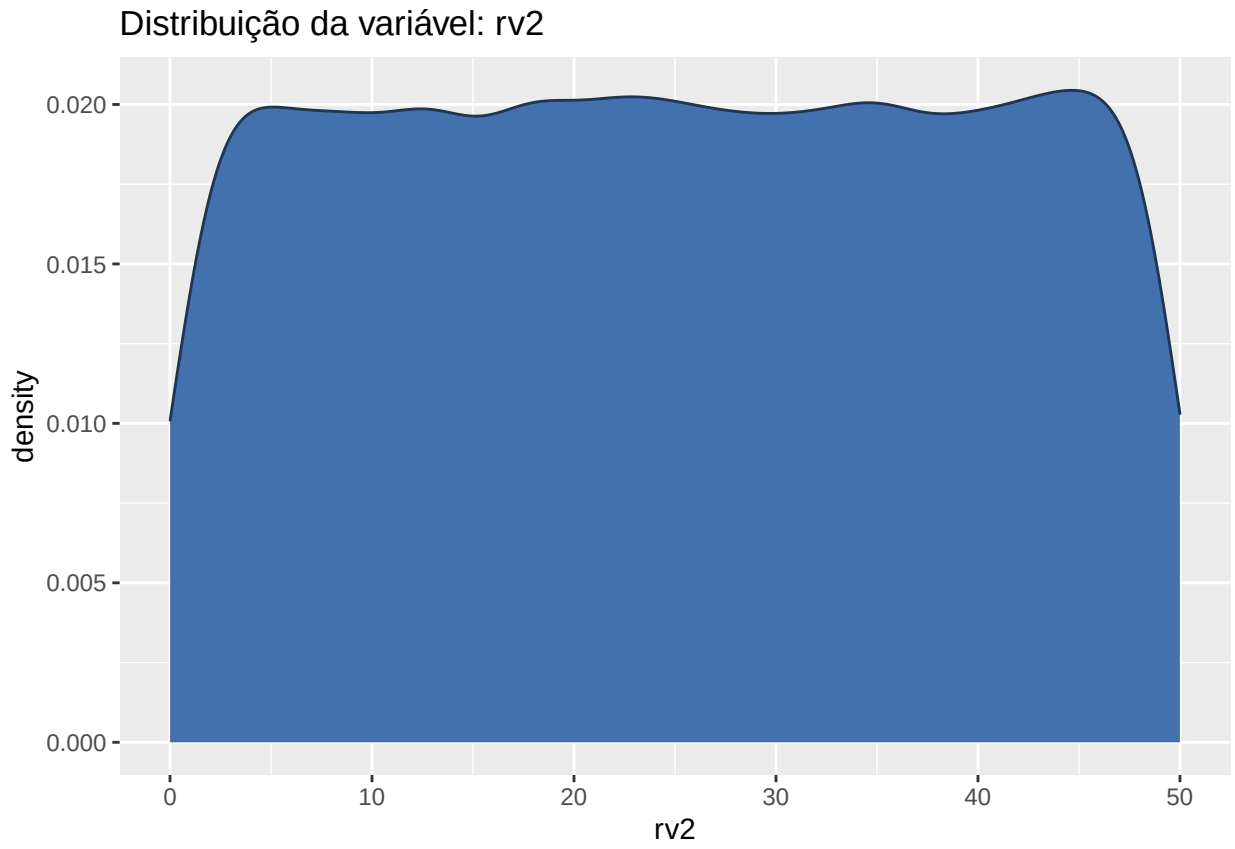

```
box_plot(data = df_train,col = 'rv1')
```



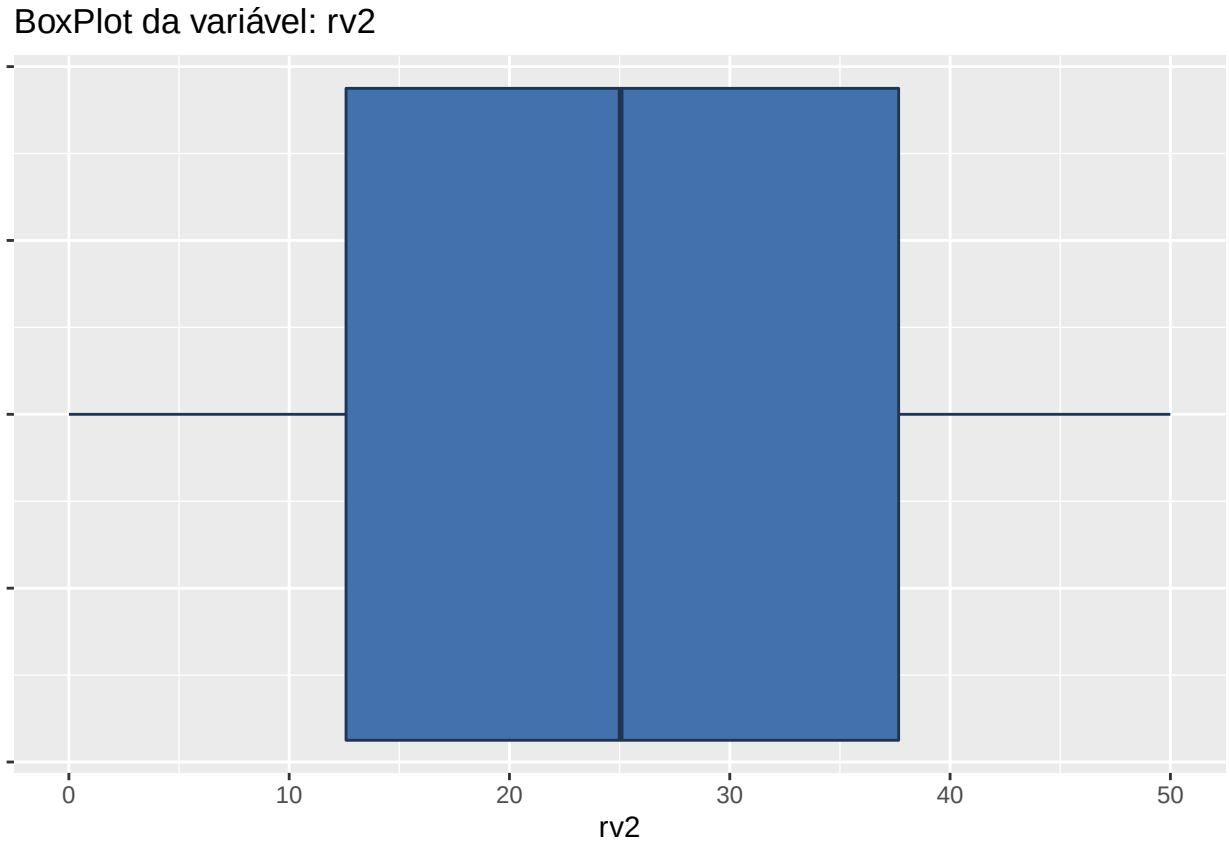
Com o distplot podemos ver que os dados estão completamente distribuídos boxplot vemos que não tem outliers.

rv2

```
dist_plot(data = df_train, col = 'rv2')
```



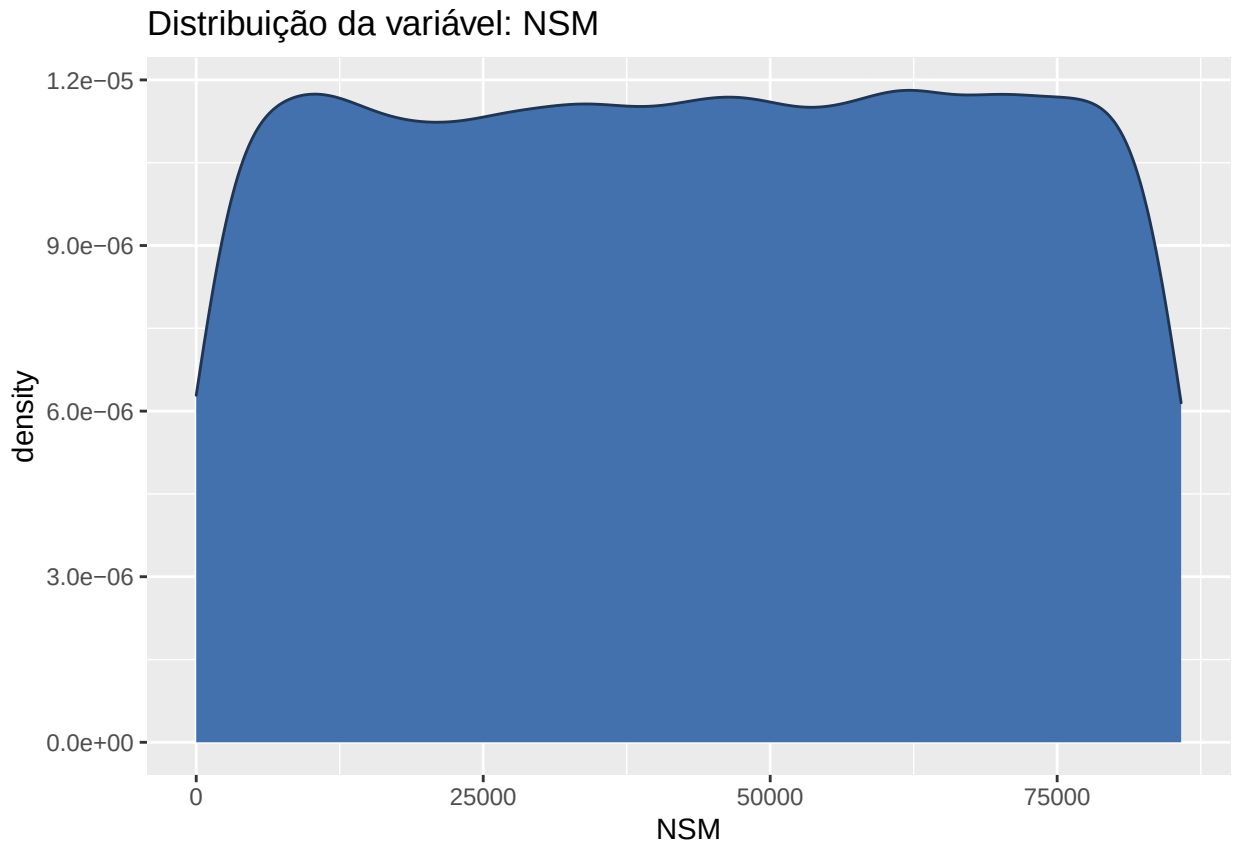

```
box_plot(data = df_train,col = 'rv2')
```



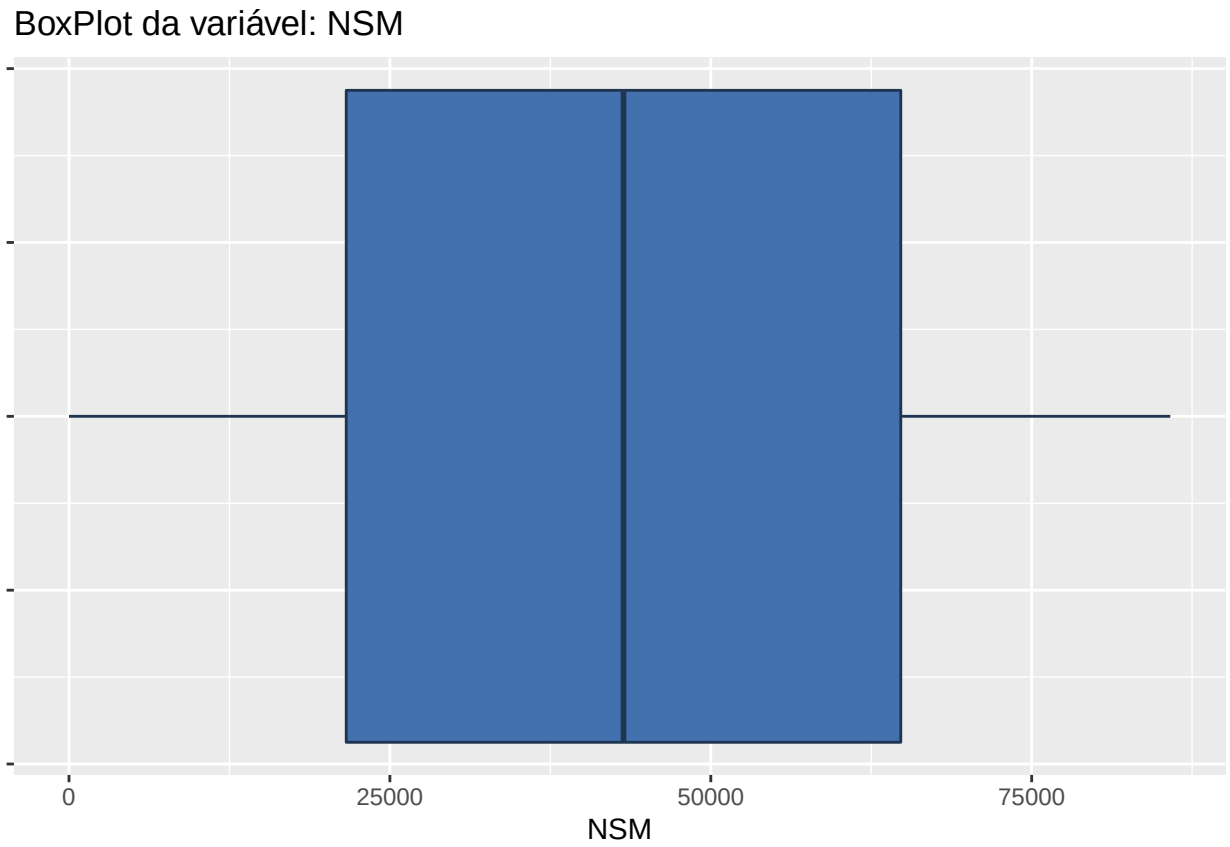
Com o distplot podemos ver que os dados estão completamente distribuídos boxplot vemos que não tem outliers.

NSM

```
dist_plot(data = df_train, col = 'NSM')
```



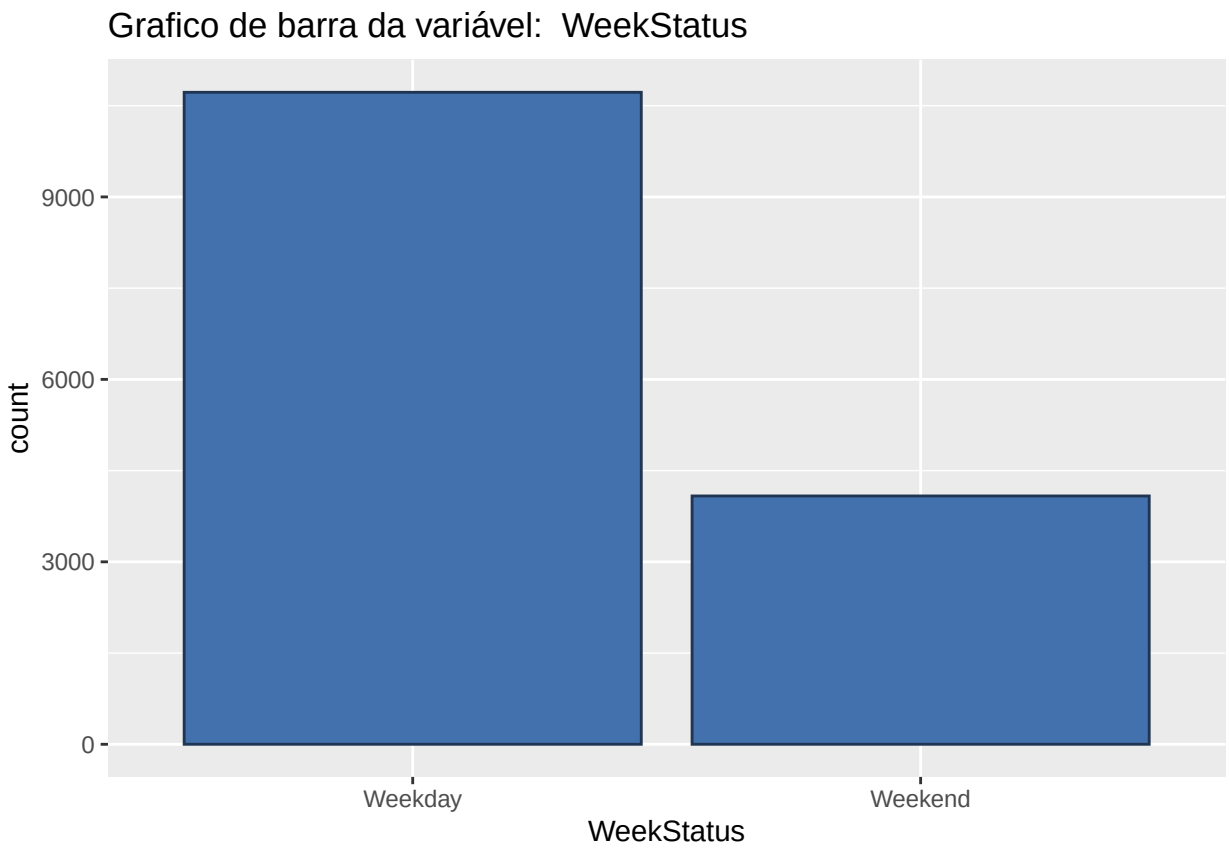

```
box_plot(data = df_train,col = 'NSM')
```



Com o distplot podemos ver que os dados estão completamente distribuídos boxplot vemos que não tem outliers.

WeekStatus

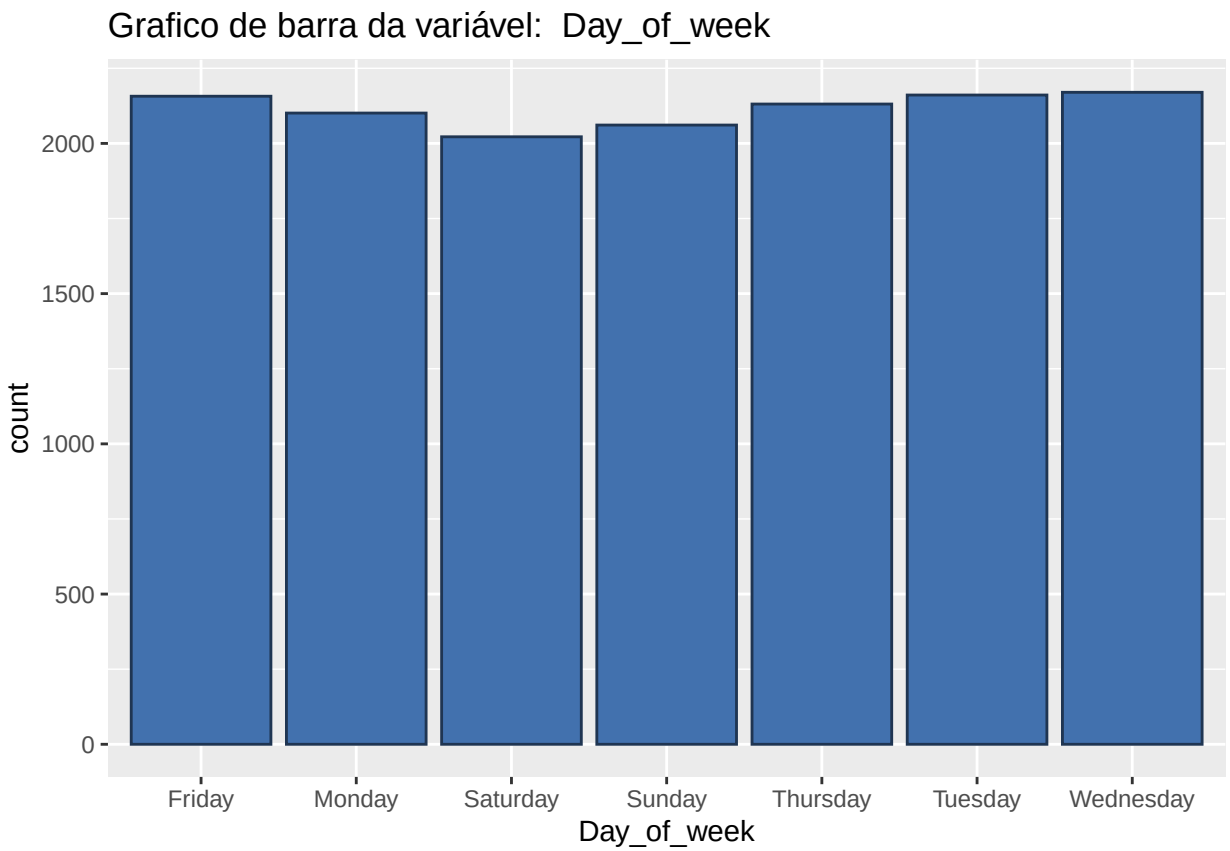
```
bar_plot(df_train, 'WeekStatus')
```



Temos mais dados coletados durante a semana do que final de semana, como já esperado já que temos mais dias durante a semana e foi coletado dados em dias corridos.

Day_of_week

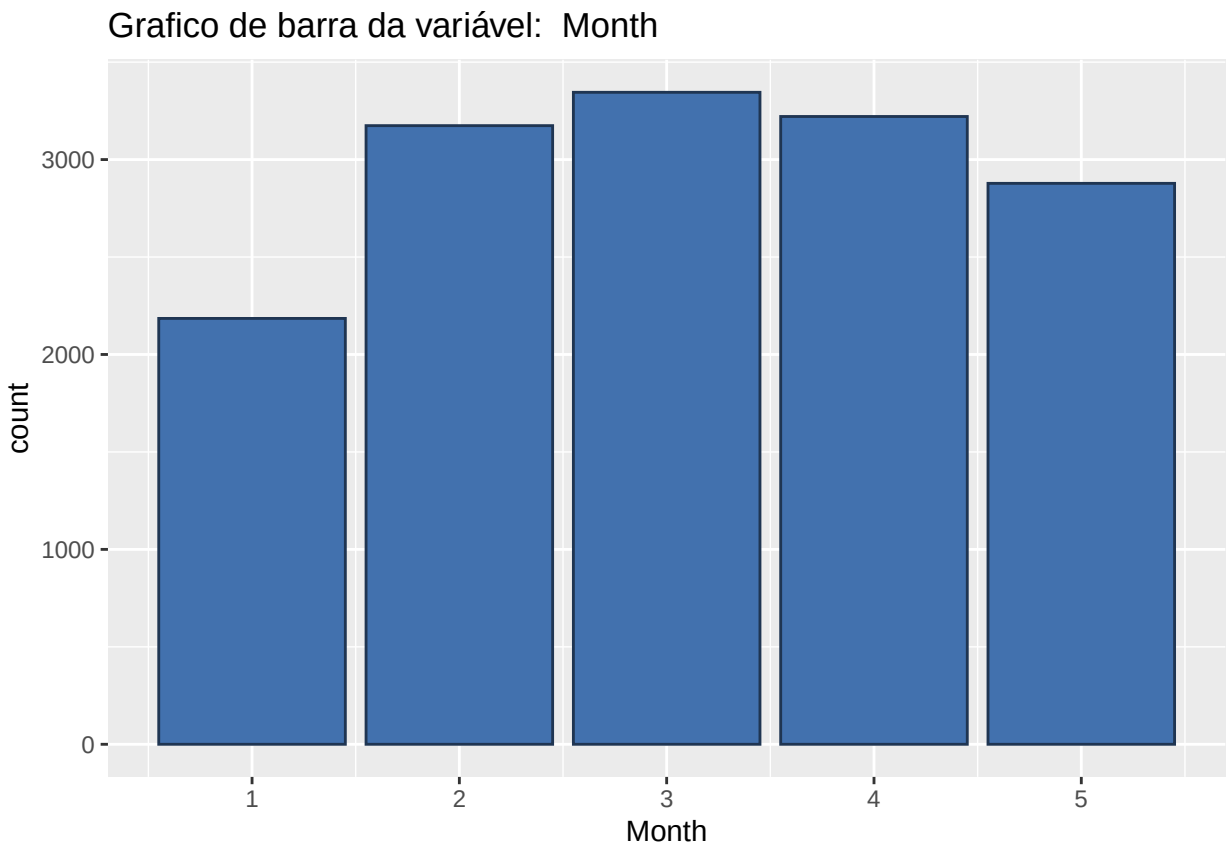
```
bar_plot(df_train, 'Day_of_week')
```



A distribuição dos dados coletados por dia está totalmente equilibrada.

Month

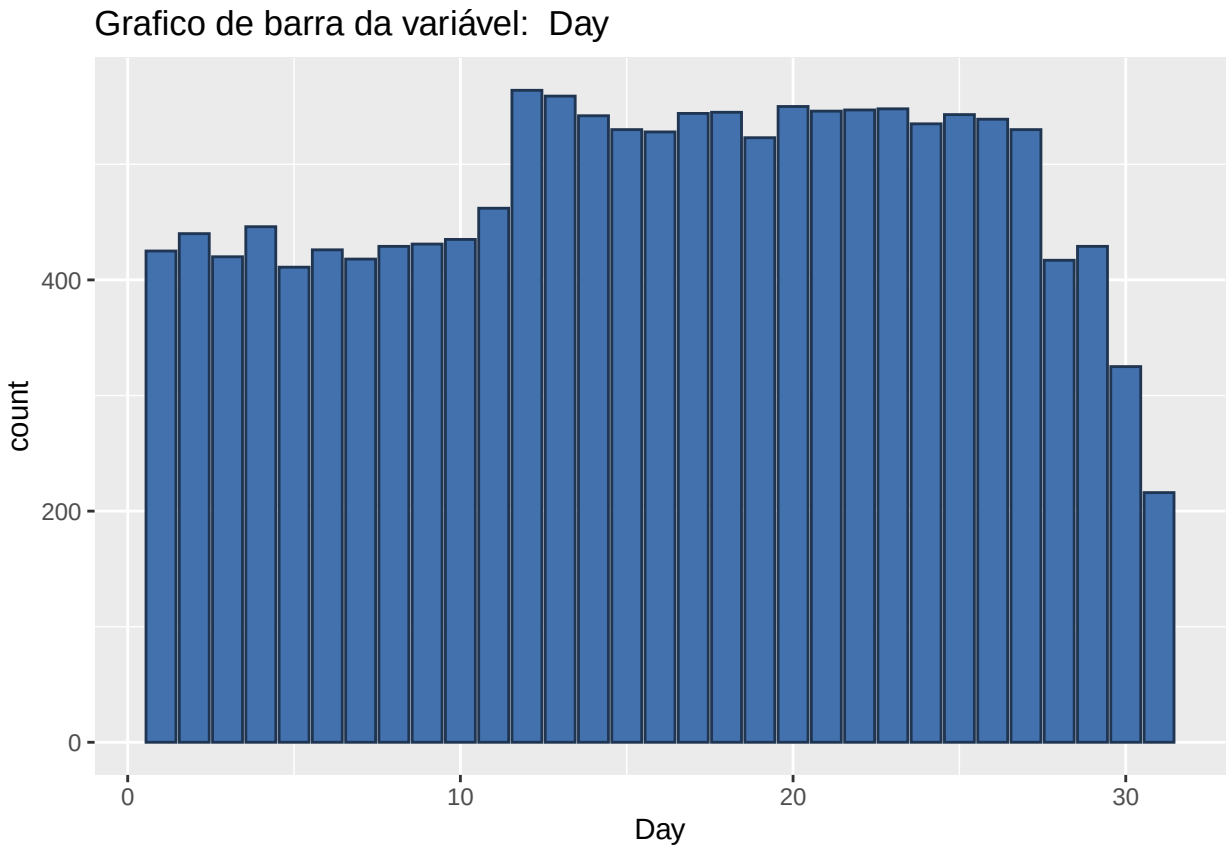
```
bar_plot(df_train, 'Month')
```



A distribuição dos dados coletados por mês mostra que o mês de marco foi o que mais teve dados, e o de janeiro menos.

Day

```
bar_plot(df_train, 'Day')
```



Do meio para o final do mes aumenta a contabilização de dados coletados.

Quantidade total de energia gasta por mês

```
df_train %>%
  select(Appliances,Month)%>%
  group_by(Month)%>%
  summarise(sum(Appliances))
```

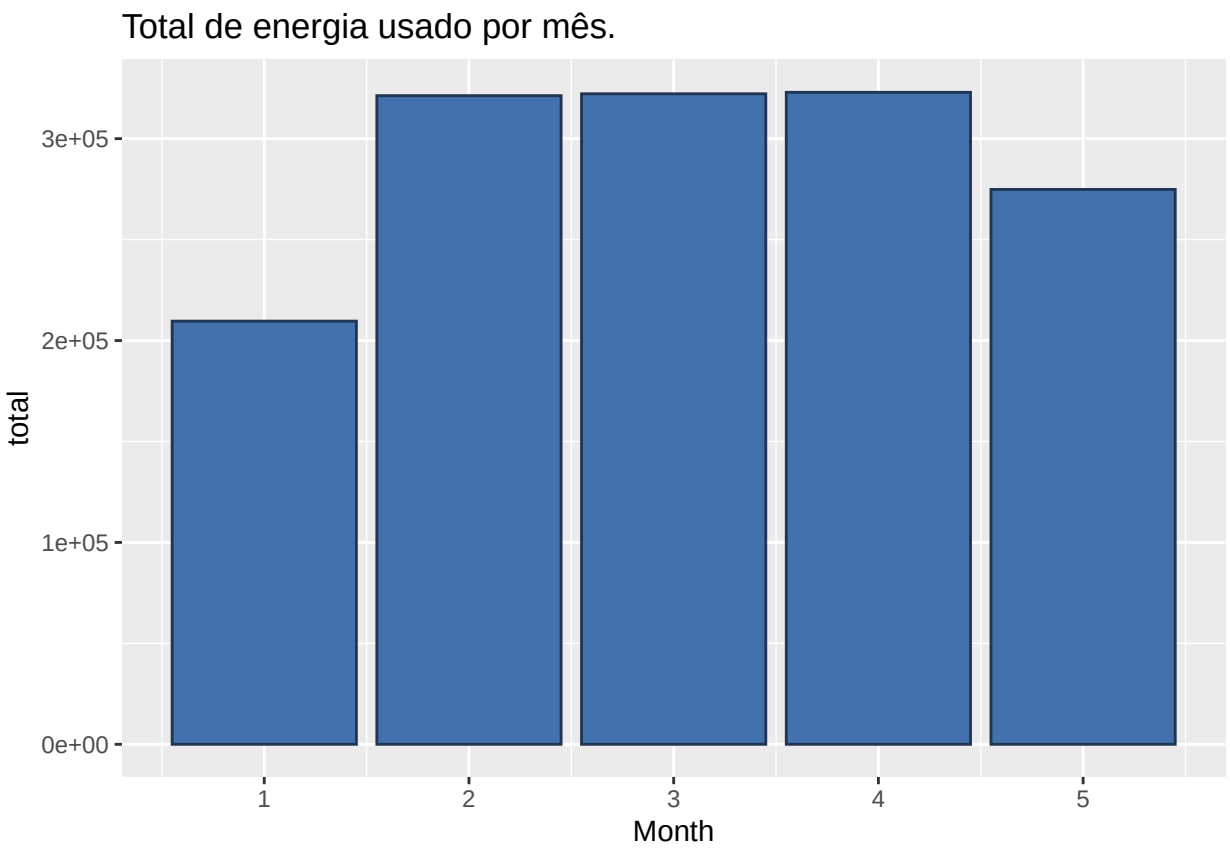
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 2
##   Month `sum(Appliances)`
##   <int>             <dbl>
## 1     1            209600
## 2     2            321280
## 3     3            322210
## 4     4            322930
## 5     5            274840
```

```
df_train %>%
  select(Appliances,Month)%>%
  group_by(Month)%>%
```

```
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = Month, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por mês.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Fevereiro, março e abril teve o mesmo consumo total, seguidos um pouco mais baixo maio e janeiro respectivamente.

Quantidade total de energia gasta por dia

```
df_train %>%
  select(Appliances,Day)%>%
  group_by(Day)%>%
  summarise(sum(Appliances))
```

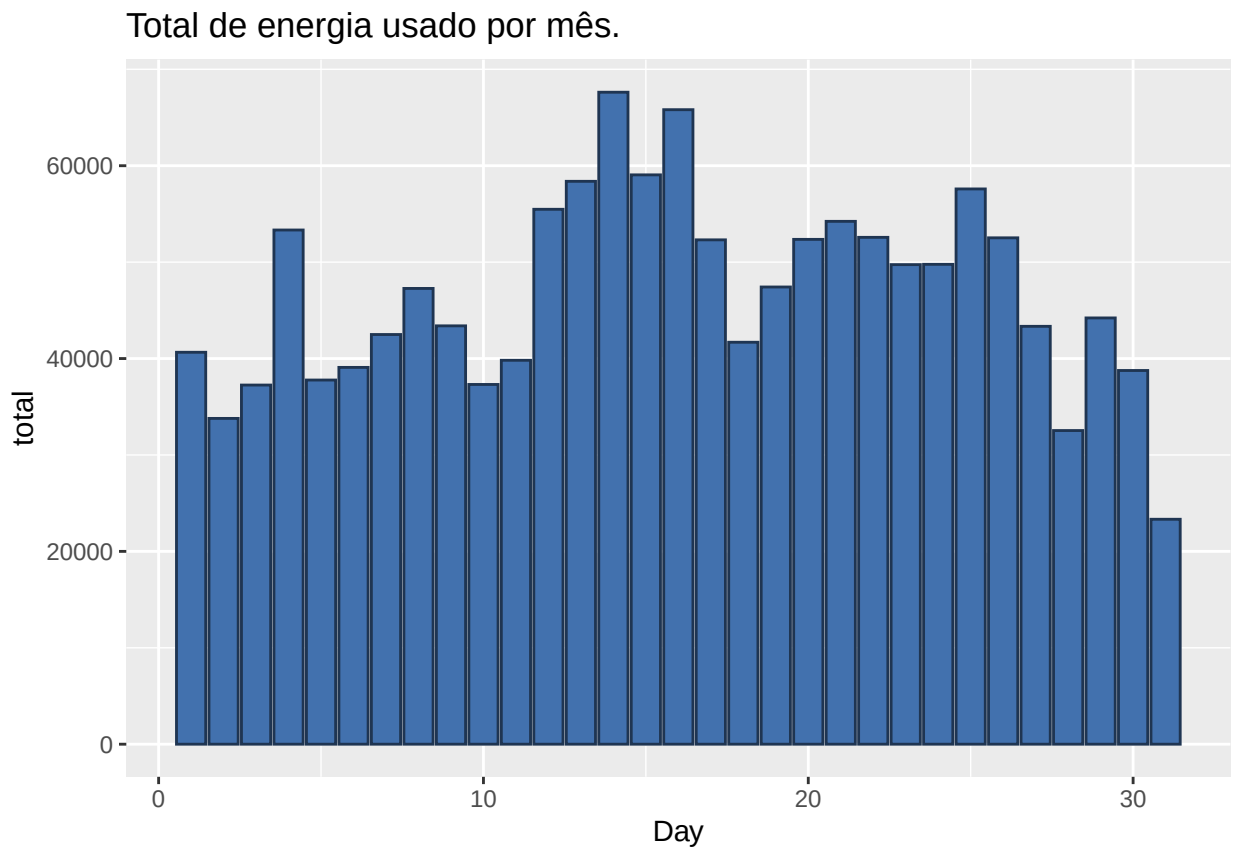
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 31 x 2
##      Day `sum(Appliances)`
##    <int>             <dbl>
```

```
##  1     1              40650
##  2     2              33790
##  3     3              37250
##  4     4              53330
##  5     5              37770
##  6     6              39080
##  7     7              42490
##  8     8              47270
##  9     9              43390
## 10    10              37310
## # ... with 21 more rows
```

```
df_train %>%
  select(Appliances,Day)%>%
  group_by(Day)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = Day, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por mês.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Podemos notar um aumento de consumo de energia no meio do mês, entre os dias 12 a 17.

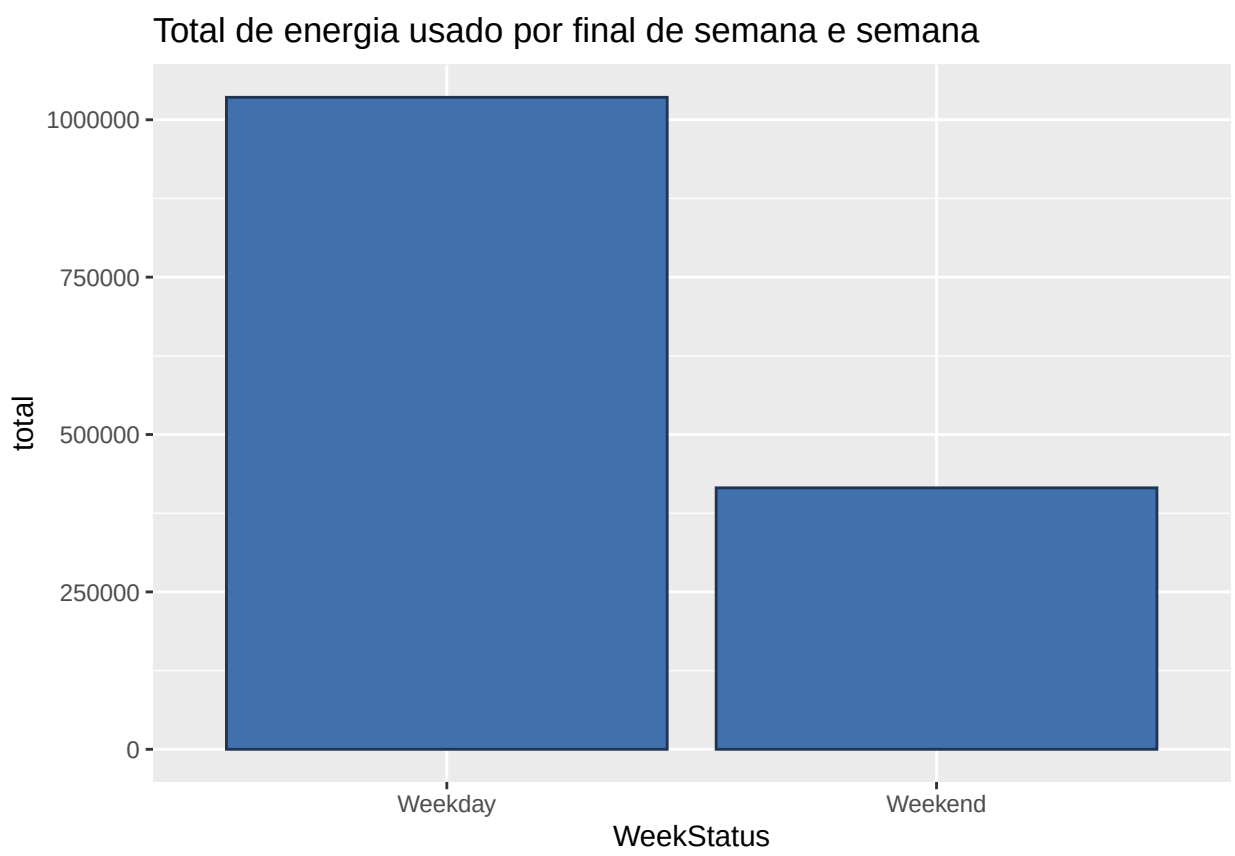
Quantidade total de energia gasta separando por dia da semana e final de semana.

```
df_train %>%
  select(Appliances,WeekStatus)%>%
  group_by(WeekStatus)%>%
  summarise(total = sum(Appliances))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 2
##   WeekStatus   total
##   <chr>        <dbl>
## 1 Weekday    1035590
## 2 Weekend     415270
```

```
df_train %>%
  select(Appliances,WeekStatus)%>%
  group_by(WeekStatus)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = WeekStatus, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por final de semana e semana'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Como já era de se esperava o consumo total durante a semana é o dobro do final de semana.

Quantidade total de energia gasta por dia da semana

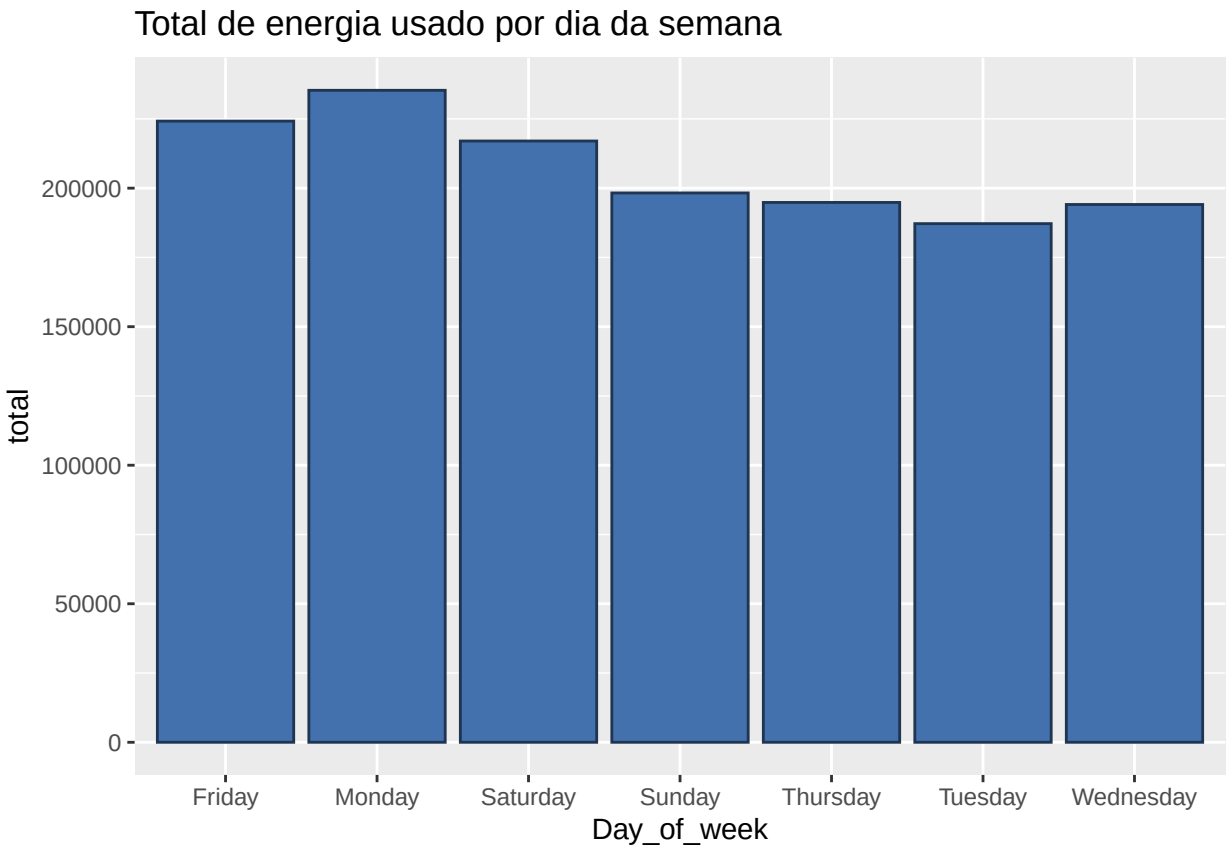
```
df_train %>%
  select(Appliances,Day_of_week)%>%
  group_by(Day_of_week)%>%
  summarise(total = sum(Appliances))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 7 x 2
##   Day_of_week  total
##   <chr>        <dbl>
## 1 Friday      224190
## 2 Monday      235300
## 3 Saturday    217010
## 4 Sunday      198260
## 5 Thursday    194830
## 6 Tuesday     187180
## 7 Wednesday   194090
```

```
df_train %>%
  select(Appliances,Day_of_week)%>%
  group_by(Day_of_week)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = Day_of_week, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por dia da semana'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Segunda, sexta e sábado lideram com o maior consumo, seguido dos demais dias que possuem praticamente o mesmo consumo.

Quantidade total de energia gasta por hora

```
df_train %>%
  select(Appliances,Hour)%>%
  group_by(Hour)%>%
  summarise(total = sum(Appliances))
```

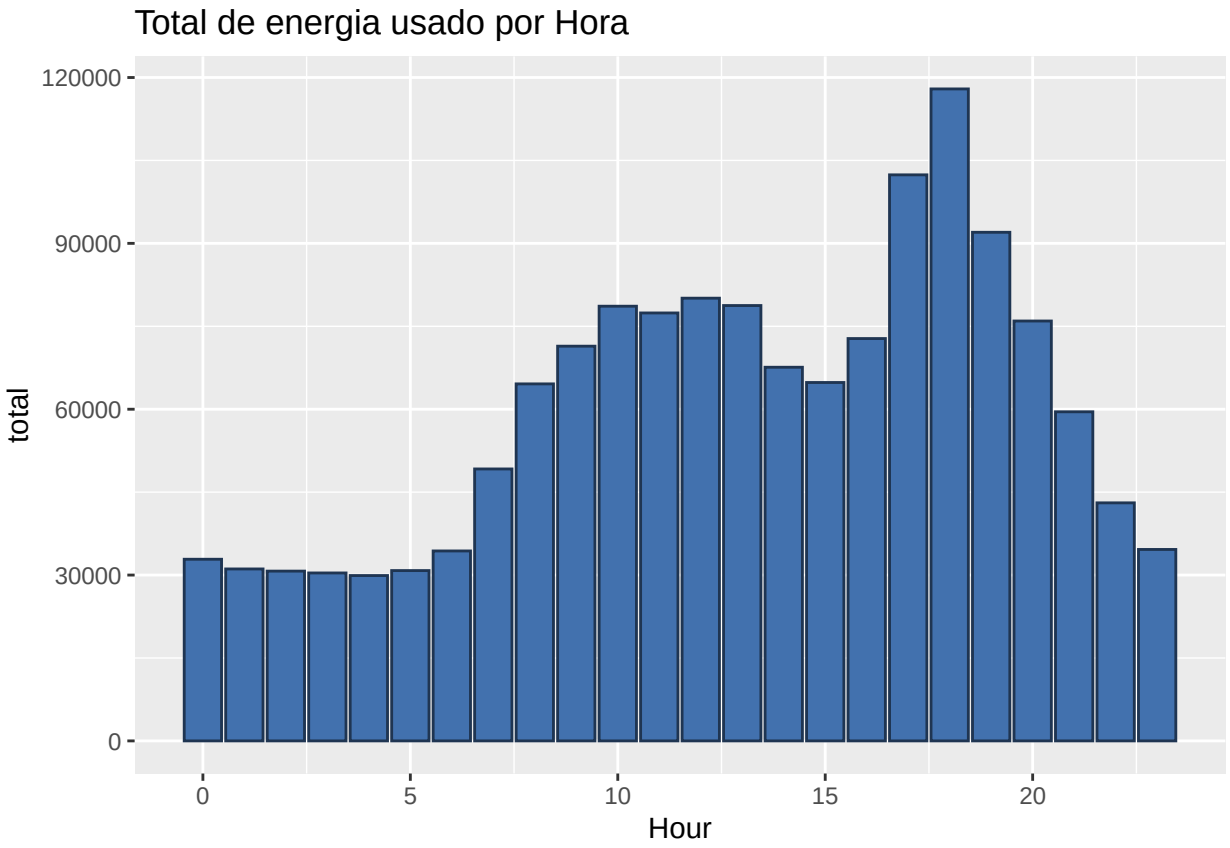
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 24 x 2
##     Hour total
##    <int> <dbl>
##  1     0 32860
##  2     1 31100
##  3     2 30710
##  4     3 30380
##  5     4 29910
##  6     5 30810
##  7     6 34360
##  8     7 49190
##  9     8 64580
```

```
## 10     9 71400
## # ... with 14 more rows
```

```
df_train %>%
  select(Appliances,Hour)%>%
  group_by(Hour)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = Hour, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por Hora'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



A parte da noite entre 17:00 a 19:00 são os horários com maiores consumos, um indicio é o fato de a maioria das pessoas chegarem em casa por essa hora do trabalho, então com mais pessoas na cada o uso aumenta.

Quantidade total de energia gasta por minuto

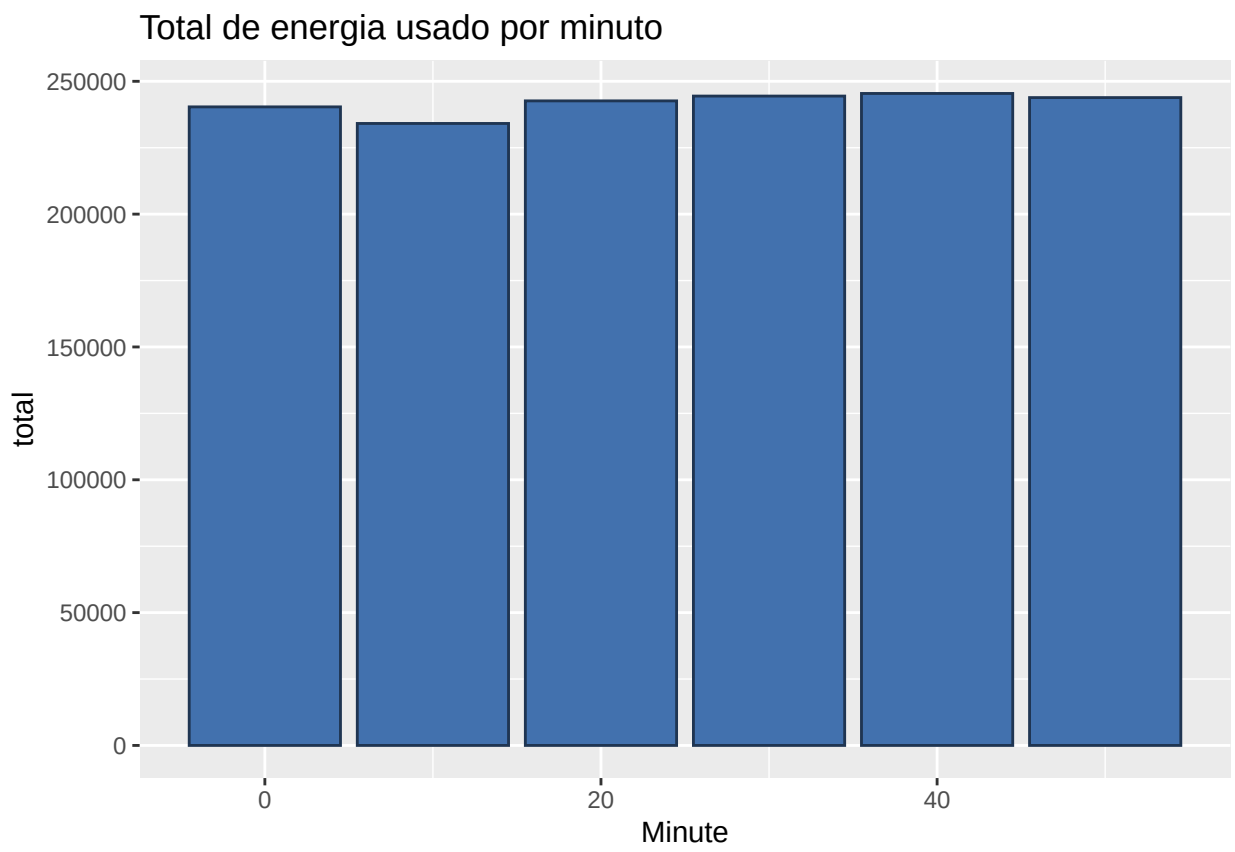
```
df_train %>%
  select(Appliances,Minute)%>%
  group_by(Minute)%>%
  summarise(total = sum(Appliances))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 6 x 2
##   Minute  total
##    <int>  <dbl>
## 1      0 240370
## 2     10 234130
## 3     20 242630
## 4     30 244440
## 5     40 245420
## 6     50 243870
```

```
df_train %>%
  select(Appliances,Minute)%>%
  group_by(Minute)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = Minute, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por minuto'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Não há diferente no uso de energia com relação os minutos que foram feitos o registro.

Quantida total de energia gasta separando por potencia dos eletrodomesticos.

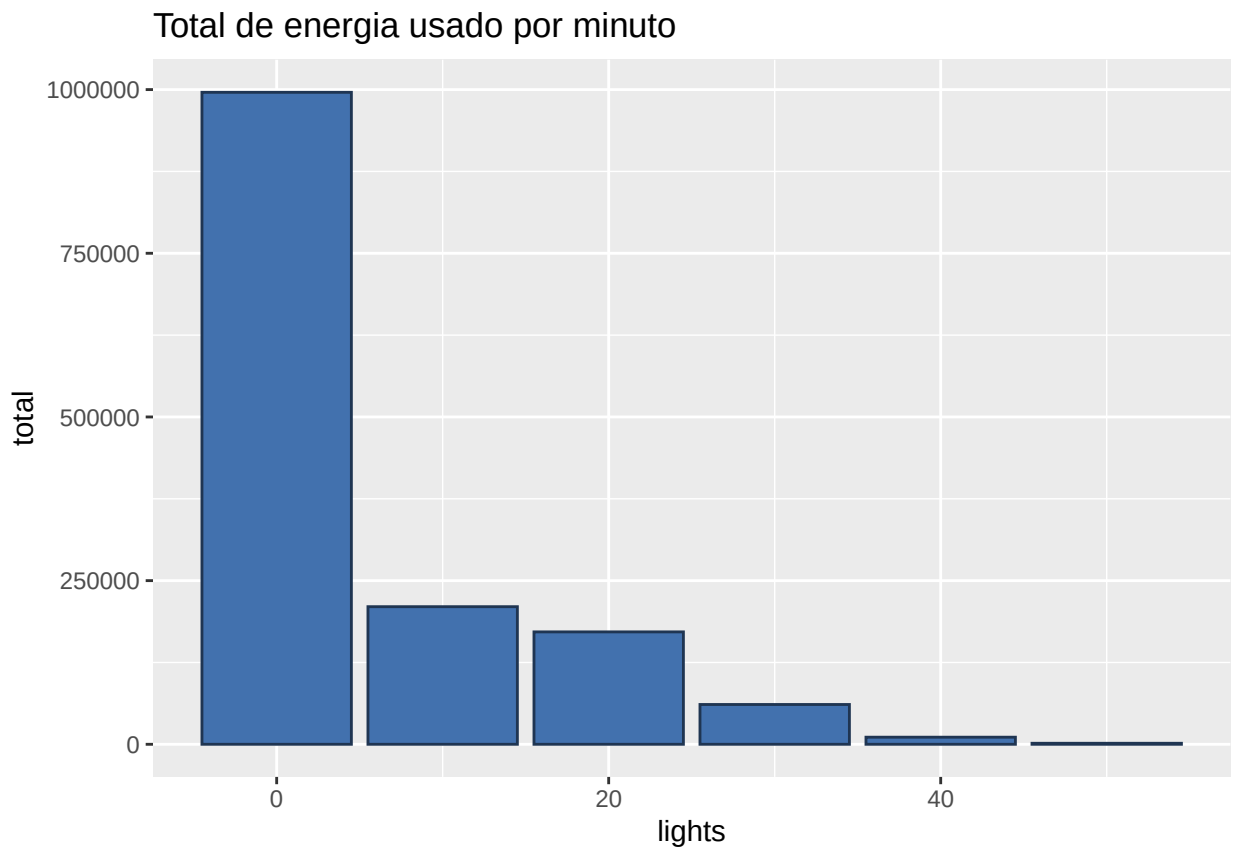
```
df_train %>%
  select(Appliances,lights)%>%
  group_by(lights)%>%
  summarise(total = sum(Appliances))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 6 x 2
##   lights  total
##    <dbl>  <dbl>
## 1      0 995830
## 2     10 210180
## 3     20 171670
## 4     30  60790
## 5     40  10840
## 6     50   1550
```

```
df_train %>%
  select(Appliances,lights)%>%
  group_by(lights)%>%
  summarise(total = sum(Appliances))%>%
  ggplot()+
  geom_bar(aes (x = lights, y= total),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Total de energia usado por minuto'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Podemos notar que o total de consumo não está diretamente ligado com o valor da potência para cada eletrodoméstico pois o com potência com nomeação 0 no DataFrame é o que mais tem consumo.

Com essas analises podemos tirar alguns insights e conhecimento sobre os dados e seus relacionamentos.

Tratamento dos dados

Aqui faremos alguns tratamentos nos dados, como tratamento dos outliers da target, normalização dos dados, criação de variáveis label, verificar a correlação entre as variáveis.

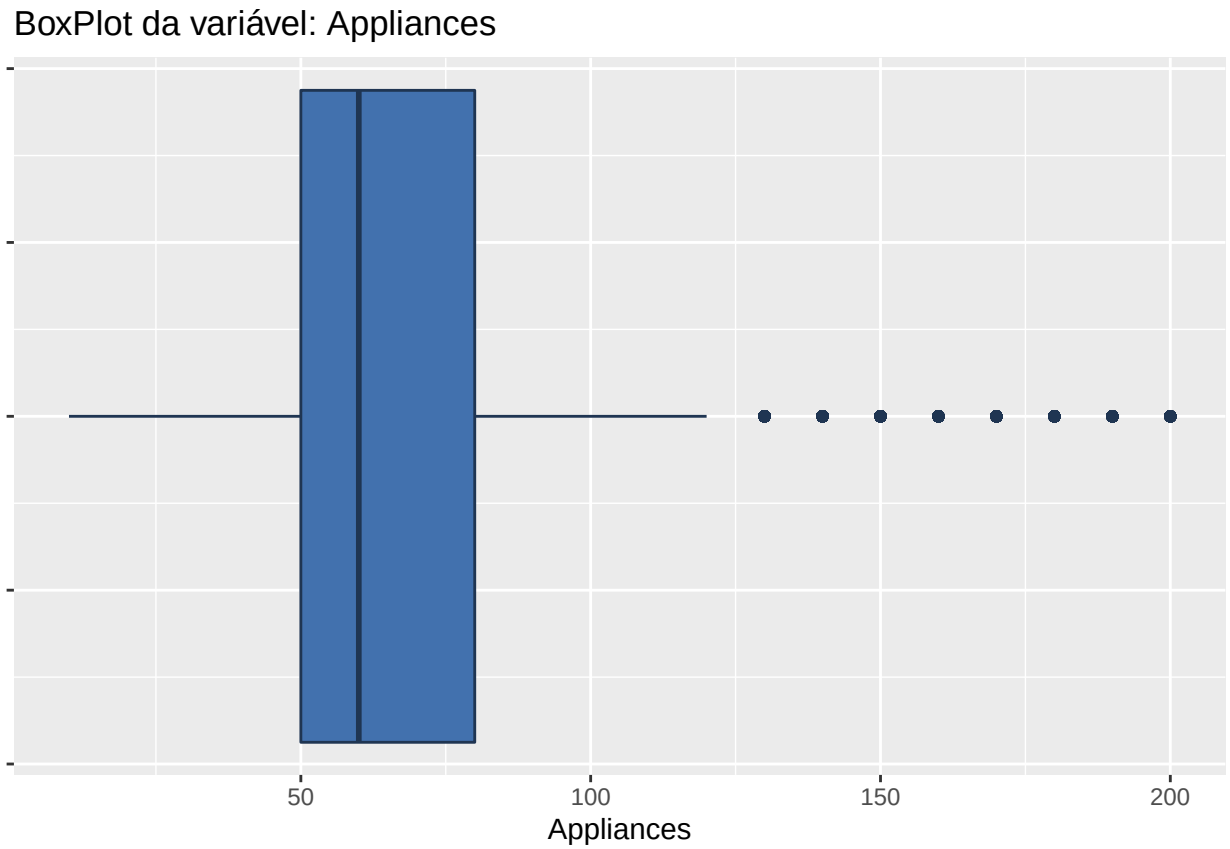
```
# Converto para valor binário (0,1) onde 1 representa dias da semana e 0 dias do final de semana.
df_train$WeekStatus <- ifelse(df_train$WeekStatus == 'Weekend',0,1)

# Função para converter o label dia da semana para numero.
 LabelEncoder<- function(var) {
   if( var == 'Sunday'){
     var = 1
   }
   else if( var == 'Monday'){
     var = 2
   }
   else if( var == 'Tuesday'){
     var = 3
   }
   else if( var == 'Wednesday'){
     var = 4
```

```
   }
   else if( var == 'Thursday'){
     var = 5
   }
   else if( var == 'Friday'){
     var = 6
   }
   else if( var == 'Saturday'){
     var = 7
   }

}
# Aplico convertendo a variável.
df_train$Day_of_week <- sapply(df_train$Day_of_week, LabelEncoder)
# Verifico se foi gerado algum valor nulo.
sum(is.na(df_train))
```

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

Como vimos na analise exploratória, a variável appliances(target) possue valores outliers, e isso pode prejudiar o modelo, então iremos trata-los agora.

```
# Irei  restringir a target em valores menores que 0.90 quartil
quart90 <- quantile(df_train$Appliances, probs = 0.90)

# Aplico o filtro para tirar diminuir os outliers
df_train_out_t <-df_train[df_train$Appliances<=quart90[[1]],]

# Plot do mesmo grafico vis antes agora com menos outliers
box_plot(data = df_train_out_t,col = "Appliances")
```




```
#sumario estatisco da target agora tratada.
summary(df_train_out_t["Appliances"])
```

```
##    Appliances
##  Min.   : 10.00
##  1st Qu.: 50.00
##  Median : 60.00
##  Mean   : 68.64
##  3rd Qu.: 80.00
##  Max.   :200.00
```

```
# Retiro a variável date
df_train_out_t$date <- NULL
```

Normalização dos dados

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# Crio um DataFrame so das variáveis normalizadas
names(df_train_out_t[,1:29])
```

```
##  [1] "Appliances"  "lights"      "T1"          "RH_1"        "T2"
```

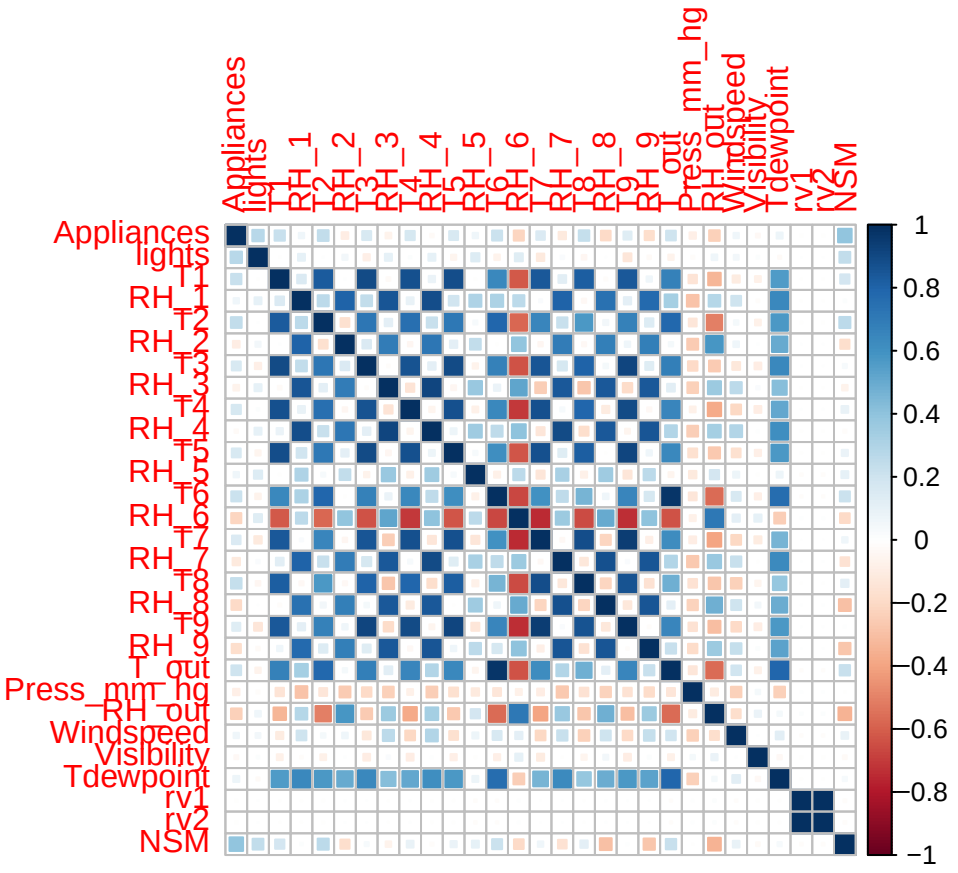
```
##  [6] "RH_2"        "T3"          "RH_3"        "T4"          "RH_4"
## [11] "T5"          "RH_5"        "T6"          "RH_6"        "T7"
## [16] "RH_7"        "T8"          "RH_8"        "T9"          "RH_9"
## [21] "T_out"       "Press_mm_hg" "RH_out"      "Windspeed"   "Visibility"
## [26] "Tdewpoint"   "rv1"         "rv2"         "NSM"
```

```
data_norm <- as.data.frame(sapply(df_train_out_t[,1:29], normalize))
```

```
# Correlação das variáveis
correlacao <- cor(data_norm)
correlacao[1,]
```

```
##   Appliances       lights           T1         RH_1           T2         RH_2
##  1.000000000  0.276933739  0.224515771  0.060656595  0.245819094 -0.093940732
##           T3         RH_3           T4         RH_4           T5         RH_5
##  0.169498390 -0.068901479  0.178547985 -0.025352291  0.168282211  0.061608062
##           T6         RH_6           T7         RH_7           T8         RH_8
##  0.212307645 -0.216713932  0.155394984 -0.115226361  0.235186419 -0.198616453
##           T9         RH_9        T_out  Press_mm_hg       RH_out    Windspeed
##  0.137718546 -0.177750584  0.200937361 -0.086961260 -0.237042661  0.060306495
##   Visibility    Tdewpoint          rv1          rv2          NSM
## -0.032408382  0.077750705 -0.009287457 -0.009287457  0.390787903
```

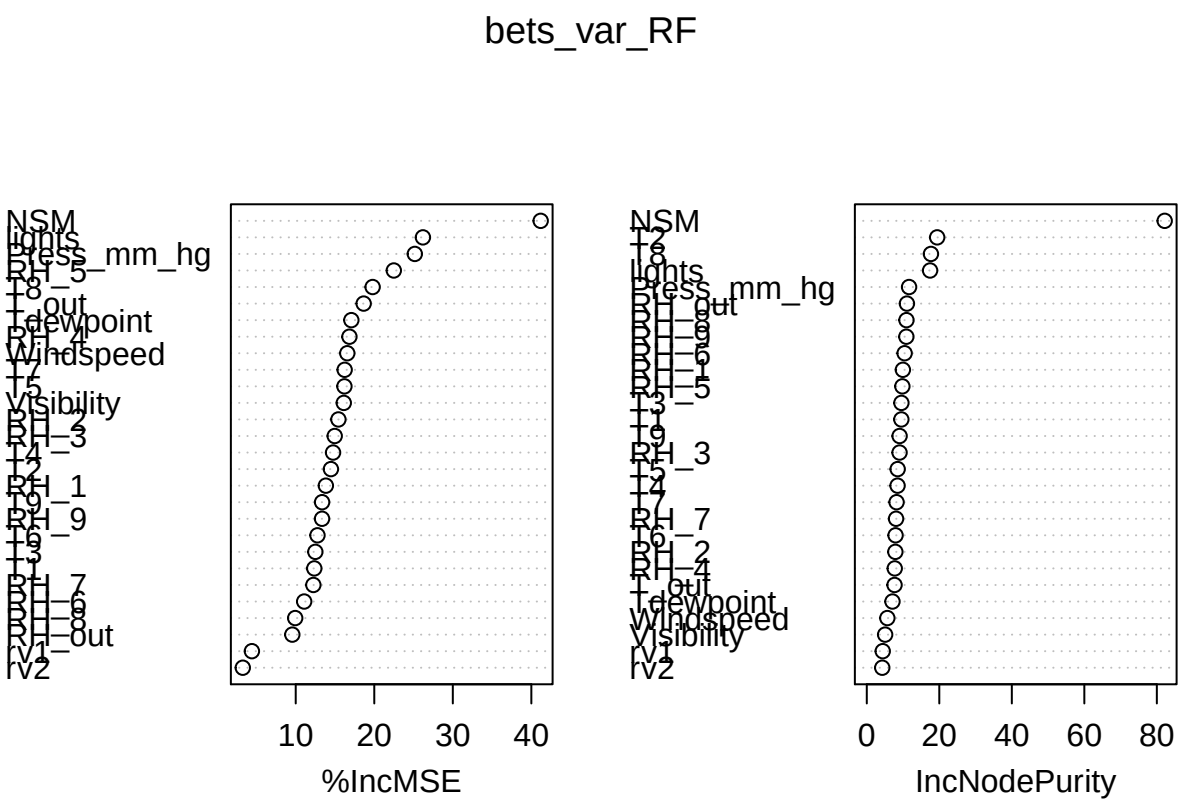
```
corrplot(correlacao,  method = 'square')
```



Feature Selection (Seleção de Variáveis)

Usarei o modelo randomForest e para seleção das melhores variáveis.

```
bets_var_RF <- randomForest(Appliances  ~ . ,data = data_norm,
                            ntree = 100, nodesize = 10, importance = T)

varImpPlot(bets_var_RF)
```



Analisando as importâncias para o modelo e juntamente com a correlação separei as variáveis abaixo para o treinamento dos modelos.

```
# Crio o vetor com as variáveis mais importantes para filtragem na criação do novo DataFrame.
best_var <- c('Appliances','T2','NSM','lights', 'RH_6','RH_out','Tdewpoint')

# Crio o DataFrame final que sera usado nos modelos preditivos.
train_pred <- data_norm[best_var]
head(train_pred)
```

```
##   Appliances        T2       NSM lights      RH_6    RH_out Tdewpoint
## 1  0.2631579 0.2253453 0.7132867    0.6 0.8418268 0.8947368 0.5384615
## 2  0.2631579 0.2253453 0.7202797    0.6 0.8398719 0.8947368 0.5339367
## 3  0.2105263 0.2253453 0.7272727    0.6 0.8307044 0.8947368 0.5294118
## 4  0.2631579 0.2253453 0.7412587    0.8 0.8482642 0.8947368 0.5203620
## 5  0.2105263 0.2253453 0.7482517    0.8 0.8570947 0.8947368 0.5158371
## 6  0.2631579 0.2253453 0.7622378    1.0 0.8637344 0.8925439 0.5105581
```

Split dos dados

```
# Faço uma divisão de 80/20 para dados de treino e teste.
split = sample.split(train_pred$lights, SplitRatio = 0.80)
train = subset(train_pred, split == TRUE)
test = subset(train_pred, split == FALSE)

# imprimo as Dimensões
dim(train)
```

```
## [1] 10676     7
```

```
dim(test)
```

```
## [1] 2669    7
```

Algoritmos de aprendizagem (Regressão)

```
# Modelo svm do pacote library(e1071)
# ?svm

modelo_svm <- svm(Appliances ~ .
                  ,data= train,type = 'eps-regression',kernel = 'radial',
                  cost = 10, scale = FALSE,gamma = 0.1)
previsao_svm <- predict(modelo_svm, test)


# Accuracy
mae = MAE(test$Appliances,previsao_svm)
rmse = RMSE(test$Appliances,previsao_svm)
r2 = R2(test$Appliances,previsao_svm)

cat(" MAE:", mae, "\n",
    "RMSE:", rmse, "\n", "R-squared:", r2)
```

```
##  MAE: 0.09784804
##  RMSE: 0.1388375
##  R-squared: 0.2885339
```

```
#  Modelo com o randomForest
# ?randomForest

modelo_RF <- randomForest(Appliances ~ .
                  ,data= train,ntree = 60,
                  nodesize = 5)
previsao_RF <- predict(modelo_RF, test)

# Accuracy
mae = MAE(test$Appliances,previsao_RF)
```

```
rmse = RMSE(test$Appliances,previsao_RF)
r2 = R2(test$Appliances,previsao_RF)

cat(" MAE:", mae, "\n",
    "RMSE:", rmse, "\n", "R-squared:", r2)
```

```
##  MAE: 0.06866903
##  RMSE: 0.1067149
##  R-squared: 0.5801081
```

```
#  Modelo com o xgboost
# Para o modelo tenho que converter o DF para uma matrix.
trainxb <- as.matrix(train[2:6])
trainl <- as.matrix(train[1])
testxm <- as.matrix(test[2:6])
testl <- as.matrix(test[1])
# ?xgboost
modelo_XB <- xgboost(data = trainxb,
                     label = trainl ,
                     max.depth = 2,
                     eta = 1,
                     nthread = 2,
                     nround = 2
                     )
```

```
## [1]  train-rmse:0.144878
## [2]  train-rmse:0.137439
```

```
previsao_XB <- predict(modelo_XB,testxm)

# Accuracy
mae = MAE(testl,previsao_XB)
rmse = RMSE(testl,previsao_XB)
r2= R2(testl,previsao_XB)

cat(" MAE:", mae, "\n",
    "RMSE:", rmse, "\n", "R-squared:", r2)
```

```
##  MAE: 0.09495496
##  RMSE: 0.1371279
##  R-squared: 0.3047018
```

Como podemos notar para um problema de regressão não tivemos uma boa acurácia, isso pode ser por conta dos poucos dados para treinamento, não permitindo que o modelo aprender o suficiente como também as várias colinearidades entre as variáveis que prejudicam os modelos, embora eu tenha separado as que possuem menos colinearidades possível entre elas.

Algoritmos de aprendizagem (Classificação)

Como foi dito acima, pelos motivos citados e entre outros a regressão não ficou interessante para o projeto nesse panorama, então tentarei agora uma outra abordagem, onde eu irei transformar esse projeto de regressão em um de classificação, criando variáveis target baseados no valor do uso de energia e veremos se assim teremos uma melhor performance.

Criarei 4 variáveis target, com critérios que eu determinarei da seguinte forma: uso de energia baixo de 60 eu considerarei como variável 1 (baixo), acima de 60 até 100 como variável 2 (médio), acima de 100 até 500 como variável 3 (alto), e o que for acima de 500 como variável 4 (muito alto.

Irei carrega dos dados novamente e realizar todo o tratamento feito para que não corra risco de ter algum erro.

```
# Carregano os dados
df_train <- read_csv('Dados/projeto8-training.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   date = col_datetime(format = ""),
##   WeekStatus = col_character(),
##   Day_of_week = col_character()
## )
## i Use `spec()` for the full column specifications.
```

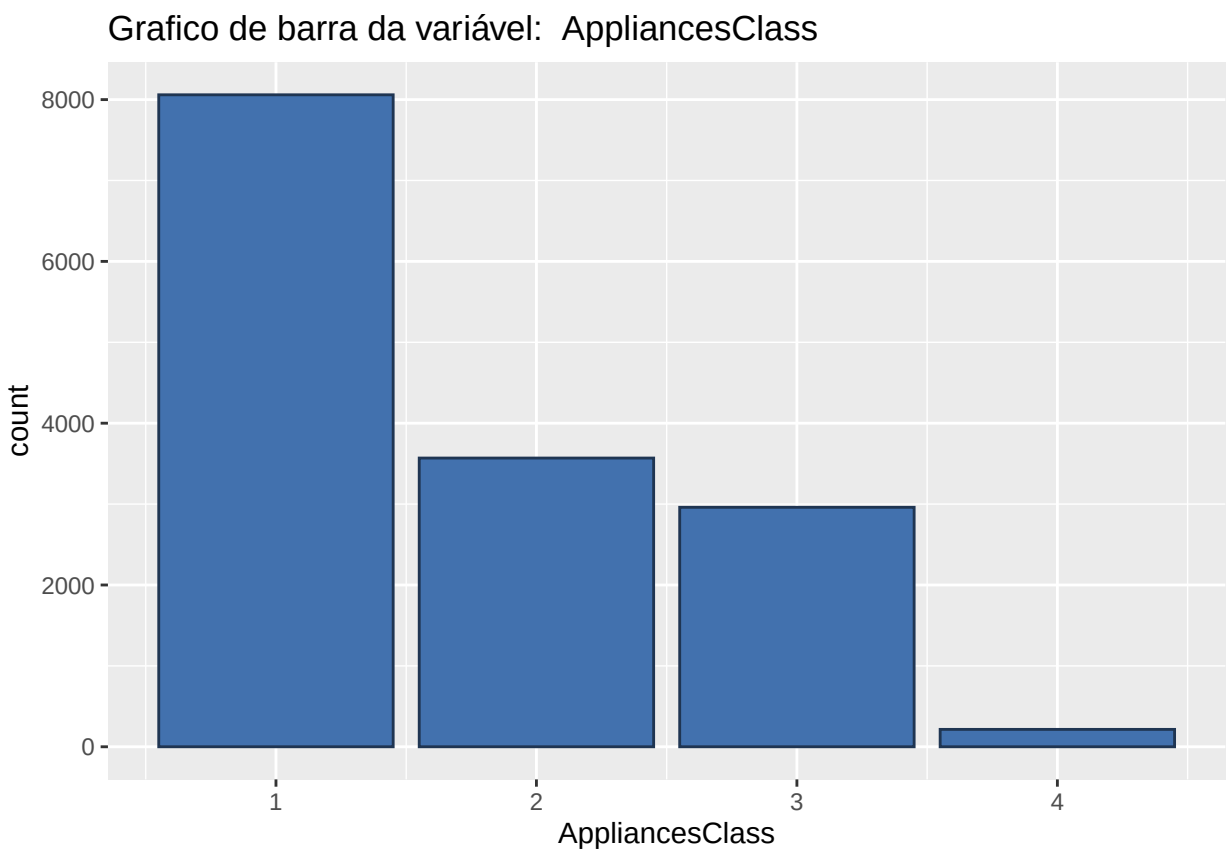
```
df_train <- as.data.frame(df_train)
df_test <- read_csv('Dados/projeto8-testing.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   date = col_datetime(format = ""),
##   WeekStatus = col_character(),
##   Day_of_week = col_character()
## )
## i Use `spec()` for the full column specifications.
```

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

# Criando variáveis (mês,dia, hora e minuto)
# df_train
df_train$Month <- sapply(df_train$date, month)
df_train$Day <- sapply(df_train$date, mday)
df_train$Hour <- sapply(df_train$date,hour)
df_train$Minute <- sapply(df_train$date, minute)
# df_test
df_test$Month <- sapply(df_test$date, month)
df_test$Day <- sapply(df_test$date, mday)
df_test$Hour <- sapply(df_test$date,hour)
df_test$Minute <- sapply(df_test$date, minute)
```

```
# Converto para valor binário (0,1) onde 1 representa dias da semana e 0 dias do final de semana.
df_train$WeekStatus <- ifelse(df_train$WeekStatus == 'Weekend',0,1)
df_test$WeekStatus <- ifelse(df_test$WeekStatus == 'Weekend',0,1)

# Coverto as variáveis dias da semana.
df_train$Day_of_week <- sapply(df_train$Day_of_week, LabelEncoder)
df_test$Day_of_week <- sapply(df_test$Day_of_week, LabelEncoder)
```

```r
# função para a criação do label de classificação
class_appliances  <- function(var) {
  if (var<=60 ){
    var = 1
  }
  else if (var > 60 & var <= 100){
    var = 2
  }
  else if (var > 100 & var <= 500){
    var = 3
  }
  else if (var > 500){
    var = 4
  }

}
# Crio a variável taget de classificação
# No de treino eu deixo junto ao DataFrame em uma nova coluna para o teste eu so vou salvar em uma vari

df_train$AppliancesClass <- sapply(df_train$Appliances, class_appliances)
test_label <- sapply(df_test$Appliances, class_appliances)

# Imprimo a quantidade por classe
bar_plot(df_train, 'AppliancesClass')
```



```r
# Converto para factor a variável target adcionando os labels
df_train$AppliancesClass <- factor(df_train$AppliancesClass,levels = c(1,2,3,4))
test_label <- factor(test_label,levels = c(1,2,3,4))

# Imprimo as categorias
glimpse(df_train$AppliancesClass)
```

```
##  Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 2 3 3 ...
```

```r
glimpse(test_label)
```

```
##  Factor w/ 4 levels "1","2","3","4": 1 1 3 4 2 2 3 3 2 1 ...
```

```r
# Verifico se foi gerado algum valor nulo.
sum(is.na(df_train))
```

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

```r
sum(is.na(df_test))
```

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

```r
# Retiro as colunas que não serão necessárias.
df_train$date <- NULL
df_train$Appliances <- NULL
df_test$date <- NULL
df_test$Appliances <- NULL
```

```r
# Normalizo os dados de treino e teste.

names(df_train[,1:28])
```

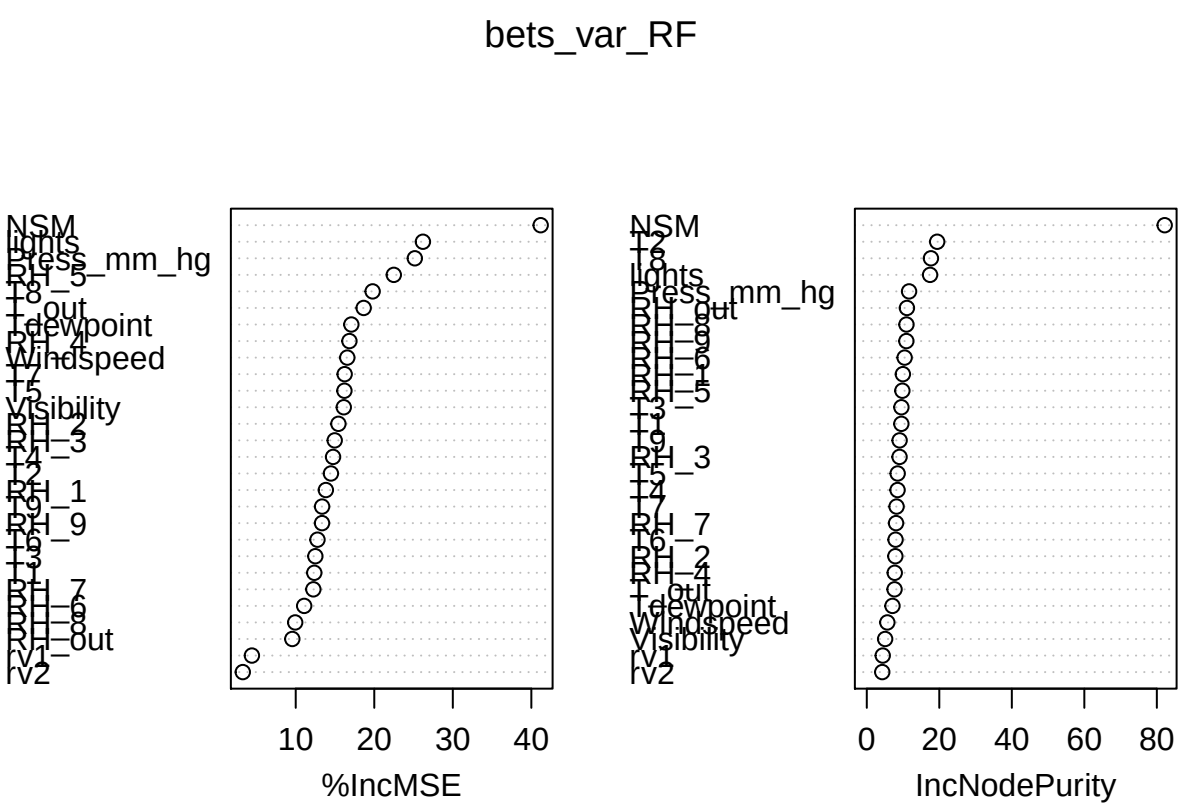
```
##  [1] "lights"      "T1"          "RH_1"        "T2"          "RH_2"
##  [6] "T3"          "RH_3"        "T4"          "RH_4"        "T5"
## [11] "RH_5"        "T6"          "RH_6"        "T7"          "RH_7"
## [16] "T8"          "RH_8"        "T9"          "RH_9"        "T_out"
## [21] "Press_mm_hg" "RH_out"      "Windspeed"   "Visibility"  "Tdewpoint"
## [26] "rv1"         "rv2"         "NSM"
```

```r
names(df_test[,1:28])
```

```
##  [1] "lights"      "T1"          "RH_1"        "T2"          "RH_2"
##  [6] "T3"          "RH_3"        "T4"          "RH_4"        "T5"
## [11] "RH_5"        "T6"          "RH_6"        "T7"          "RH_7"
## [16] "T8"          "RH_8"        "T9"          "RH_9"        "T_out"
## [21] "Press_mm_hg" "RH_out"      "Windspeed"   "Visibility"  "Tdewpoint"
## [26] "rv1"         "rv2"         "NSM"
```

```
df_train_norm <- as.data.frame(sapply(df_train[,1:28], normalize))
df_test_norm <- as.data.frame(sapply(df_test[,1:28], normalize))

# Adciono novamente as outra variáveis
train <- cbind(df_train_norm, df_train[,29:35])
test <- cbind(df_test_norm, df_test[,29:34])
```

```
# Uso o RandomForest para a seleção de variáveis

bets_var <- randomForest(AppliancesClass  ~ . ,data = train,
                            ntree = 100, nodesize = 10, importance = T)

varImpPlot(bets_var_RF)
```




```
# Podemos notar que pouca coisa mudou com relação a impotência então irei utilizar as mesmas variáveis
var_train <- c('AppliancesClass','T2','NSM','lights', 'RH_6','RH_out','Tdewpoint','Day','Hour')
var_test <- c('T2','NSM','lights', 'RH_6','RH_out','Tdewpoint','Day','Hour')
train <- train[var_train]
test <- test[var_test]
```

```
# Modelo  com o KSVM
#??KSVM
modelo_SVM <- ksvm(AppliancesClass ~ .
                   ,data= train,type="C-bsvc", kernel = "rbfdot")
```

```r
previsao_SVM <- predict(modelo_SVM, test)

confusionMatrix(previsao_SVM,test_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4
##          1 2446  539  299   31
##          2  178  411  229    7
##          3   61  239  457   35
##          4    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.6719
##                  95% CI : (0.6586, 0.685)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4159
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.9110  0.34567  0.46396   0.0000
## Specificity            0.6133  0.88939  0.91513   1.0000
## Pos Pred Value         0.7379  0.49818  0.57702      NaN
## Neg Pred Value         0.8522  0.81057  0.87246   0.9852
## Prevalence             0.5444  0.24108  0.19972   0.0148
## Detection Rate         0.4959  0.08333  0.09266   0.0000
## Detection Prevalence   0.6721  0.16727  0.16058   0.0000
## Balanced Accuracy      0.7621  0.61753  0.68954   0.5000
```

```r
# Modelo  com o RandomForest
#?randomForest
modelo_RF <- randomForest(AppliancesClass ~ .
                  ,data= train,ntree = 500,
                  nodesize = 10,method="repeatedcv",
                  number=15, repeats=200)

previsao_RF <- predict(modelo_RF, test)

confusionMatrix(previsao_RF,test_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4
##          1 2494  331   99    3
```

```
##          2  153  676  210    3
##          3   38  182  675   64
##          4    0    0    1    3
##
## Overall Statistics
##
##                Accuracy : 0.7802
##                  95% CI : (0.7684, 0.7917)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6257
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3  Class: 4
## Sensitivity            0.9289   0.5685   0.6853 0.0410959
## Specificity            0.8073   0.9022   0.9280 0.9997942
## Pos Pred Value         0.8521   0.6488   0.7039 0.7500000
## Neg Pred Value         0.9047   0.8681   0.9220 0.9857955
## Prevalence             0.5444   0.2411   0.1997 0.0148013
## Detection Rate         0.5057   0.1371   0.1369 0.0006083
## Detection Prevalence   0.5935   0.2113   0.1944 0.0008110
## Balanced Accuracy      0.8681   0.7354   0.8067 0.5204450
```

```
# Modelo  com o naiveBayes
modelo_NB <- naiveBayes(AppliancesClass ~ .
                 ,data= train,laplace=3)

previsao_NB <- predict(modelo_NB, test)

confusionMatrix(previsao_NB,test_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4
##          1 2093  493  329   31
##          2  403  313  147    9
##          3  189  383  509   33
##          4    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.591
##                  95% CI : (0.5772, 0.6048)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 2.283e-11
##
##                   Kappa : 0.3034
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.7795  0.26325   0.5168   0.0000
## Specificity            0.6204  0.85065   0.8467   1.0000
## Pos Pred Value         0.7105  0.35894   0.4569      NaN
## Neg Pred Value         0.7019  0.78424   0.8753   0.9852
## Prevalence             0.5444  0.24108   0.1997   0.0148
## Detection Rate         0.4244  0.06346   0.1032   0.0000
## Detection Prevalence   0.5973  0.17680   0.2259   0.0000
## Balanced Accuracy      0.6999  0.55695   0.6817   0.5000
```

```
#  Modelo com o xgboost
# Para o modelo tenho que converter o DF para uma matrix.
trainData  <- as.matrix(train[2:9])
trainLabel <-  as.integer(train$AppliancesClass)-1
dtrain <- xgb.DMatrix(data = trainData, label = trainLabel )

testData <- as.matrix(test[1:8])
testLabel <- as.numeric(test_label)
dtest <- xgb.DMatrix(data = testData, label = testLabel )

num_class <- length(unique(trainLabel))

xgb_params <- list(objective="multi:softprob",nfold = 100,max_depth = 6,
                   eval_metric="mlogloss",num_class=num_class,early.stop.round = 10)
# ?xgboost
modelo_XB <- xgb.train(params = xgb_params,
                       data = dtrain,
                       nrounds = 5000,
                       prediction = TRUE,
                       verbose = FALSE
)
```

```
## [12:03:55] WARNING: amalgamation/../src/learner.cc:541:
## Parameters: { early_stop_round, nfold, prediction } might not be used.
##
##   This may not be accurate due to some parameters are only used in language bindings but
##   passed down to XGBoost core.  Or some parameters are not used but slip through this
##   verification. Please open an issue if you find above cases.
```

```
previsao_XB <- predict(modelo_XB,testData,reshape = T)

previsao_XB_label <- factor(max.col(previsao_XB),levels=1:4)

confusionMatrix(previsao_XB_label,test_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4
##          1 2444  298   66    2
```

```
##          2  202  711  201    6
##          3   37  176  700   55
##          4    2    4   18   10
##
## Overall Statistics
##
##                Accuracy : 0.7837
##                  95% CI : (0.7719, 0.7951)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6369
##
##  Mcnemar's Test P-Value : 1.551e-08
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.9102   0.5980   0.7107 0.136986
## Specificity            0.8371   0.8907   0.9321 0.995061
## Pos Pred Value         0.8698   0.6348   0.7231 0.294118
## Neg Pred Value         0.8864   0.8746   0.9281 0.987138
## Prevalence             0.5444   0.2411   0.1997 0.014801
## Detection Rate         0.4955   0.1442   0.1419 0.002028
## Detection Prevalence   0.5697   0.2271   0.1963 0.006894
## Balanced Accuracy      0.8737   0.7444   0.8214 0.566024
```

Considerações Finais

Como podemos ver conseguimos uma melhora significante principalmente para o xgboost para esse problema de classificação, atingindo quase 80% de acurácia, porém, ainda não é o ideal, sobretudo com a pouca quantidade de dados como já mencionados, dificulta atingir melhores resultados, se tivesse mais dados o modelo muito provavelmente iria perfumar melhor.

Temos então agora duas soluções para o problema de negócio, podendo aplicar aquela que e encaixe melhor a necessidade.

Obrigado! Entre em contato comigo acessando meu portifolio (https://campos1989.github.io/) no menu contato!