# Projeto8-PrevendoUsoEnergiaIoT

Eduardo de Souza Dias

2/24/2020

```
# Definindo diretório de trabalho
setwd("C:/Cursos/FCD/04-Machine-Learning/21-Projetos_com_feedback/Previsao_Uso_de_Energia_IoT")
getwd()
```

```
## [1] "C:/Cursos/FCD/04-Machine-Learning/21-Projetos_com_feedback/Previsao_Uso_de_Energia_IoT"
```

```
library(data.table)
library(ggplot2)
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(grid)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(caret)
```

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

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

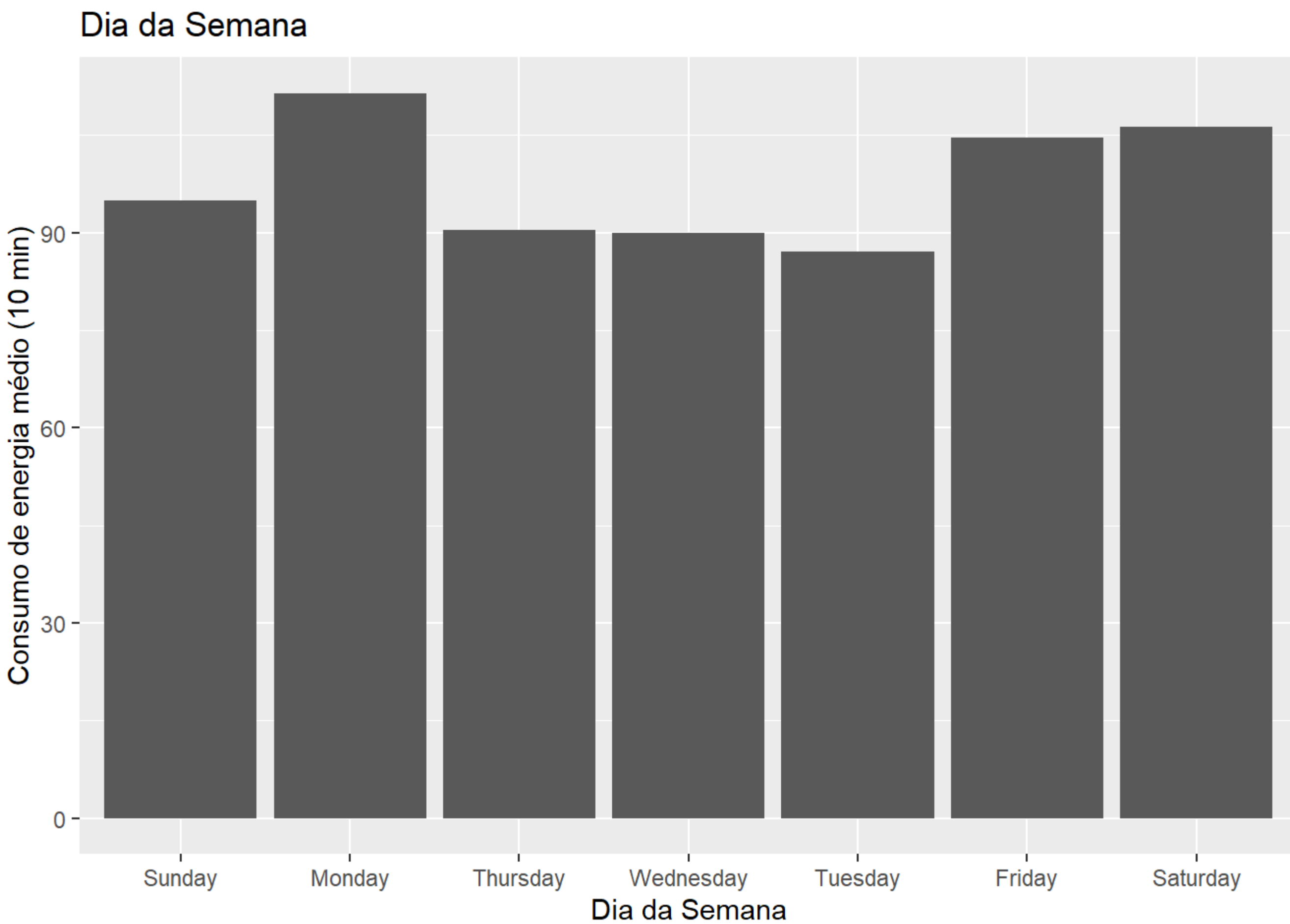
```
## The following object is masked from 'package:base':
##
##     date
```

```
# Base de Dados
dfTrain <- fread("projeto8-training.csv")
dfTest <- fread("projeto8-testing.csv")
df <- rbind(dfTrain, dfTest)

# Criando variaveis novas
df$Dia <- day(df$date)
df$Mes <- month(df$date)
df$Hora <- hour(df$date)
df$Min <- minute(df$date)

# Convertendo as variáveis para os formatos corretos
df$date <- as_datetime(df$date)
df$dateOnly <- date(df$date)
df$Appliances <- as.integer(df$Appliances)
df$lights <- as.integer(df$lights)
df$T6 <- as.numeric(df$T6)
df$RH_6 <- as.numeric(df$RH_6)
df$T_out <- as.numeric(df$T_out)
df$RH_out <- as.numeric(df$RH_out)
df$Windspeed <- as.numeric(df$Windspeed)
df$Visibility <- as.numeric(df$Visibility)
df$Tdewpoint <- as.numeric(df$Tdewpoint)
df$rv1 <- as.numeric(df$rv1)
df$rv2 <- as.numeric(df$rv2)
df$NSM <- as.integer(df$NSM)
df$Dia <- as.factor(df$Dia)
df$Mes <- as.factor(df$Mes)
df$Hora <- as.factor(df$Hora)
df$Min <- as.factor(df$Min)

str(df)
```

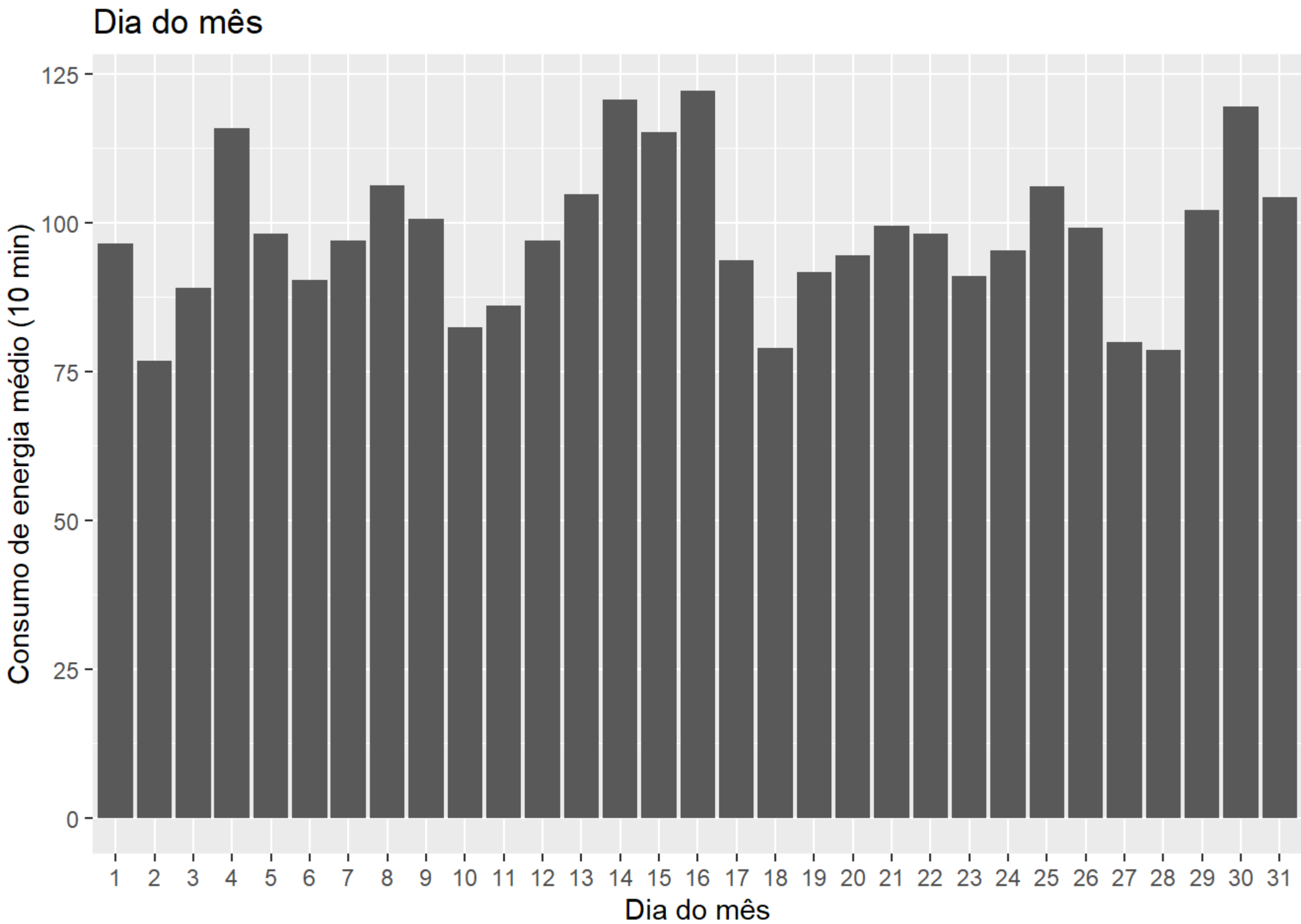
```
## Classes 'data.table' and 'data.frame':   19735 obs. of  37 variables:
##  $ date       : POSIXct, format: "2016-01-11 17:00:00" "2016-01-11 17:10:00" ...
##  $ Appliances : int  60 60 50 60 50 60 60 70 430 250 ...
##  $ lights     : int  30 30 30 40 40 50 40 40 50 40 ...
##  $ T1         : num  19.9 19.9 19.9 19.9 19.9 ...
##  $ RH_1       : num  47.6 46.7 46.3 46.3 46 ...
##  $ T2         : num  19.2 19.2 19.2 19.2 19.2 ...
##  $ RH_2       : num  44.8 44.7 44.6 44.5 44.5 ...
##  $ T3         : num  19.8 19.8 19.8 19.8 19.8 ...
##  $ RH_3       : num  44.7 44.8 44.9 45 44.9 ...
##  $ T4         : num  19 19 18.9 18.9 18.9 ...
##  $ RH_4       : num  45.6 46 45.9 45.5 45.7 ...
##  $ T5         : num  17.2 17.2 17.2 17.2 17.1 ...
##  $ RH_5       : num  55.2 55.2 55.1 55.1 55 ...
##  $ T6         : num  7.03 6.83 6.56 6.37 6.3 ...
##  $ RH_6       : num  84.3 84.1 83.2 84.9 85.8 ...
##  $ T7         : num  17.2 17.2 17.2 17.2 17.1 ...
##  $ RH_7       : num  41.6 41.6 41.4 41.2 41.3 ...
##  $ T8         : num  18.2 18.2 18.2 18.1 18.1 ...
##  $ RH_8       : num  48.9 48.9 48.7 48.6 48.6 ...
##  $ T9         : num  17 17.1 17 17 17 ...
##  $ RH_9       : num  45.5 45.6 45.5 45.4 45.3 ...
##  $ T_out      : num  6.6 6.48 6.37 6.13 6.02 ...
##  $ Press_mm_hg: num  734 734 734 734 734 ...
##  $ RH_out     : num  92 92 92 92 92 ...
##  $ Windspeed  : num  7 6.67 6.33 5.67 5.33 ...
##  $ Visibility : num  63 59.2 55.3 47.7 43.8 ...
##  $ Tdewpoint  : num  5.3 5.2 5.1 4.9 4.8 ...
##  $ rv1        : num  13.3 18.6 28.6 10.1 44.9 ...
##  $ rv2        : num  13.3 18.6 28.6 10.1 44.9 ...
##  $ NSM        : int  61200 61800 62400 63600 64200 65400 66000 66600 68400 69000 ...
##  $ WeekStatus : chr  "Weekday" "Weekday" "Weekday" "Weekday" ...
##  $ Day_of_week: chr  "Monday" "Monday" "Monday" "Monday" ...
##  $ Dia        : Factor w/ 31 levels "1","2","3","4",..: 11 11 11 11 11 11 11 11 11 11 ...
##  $ Mes        : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Hora       : Factor w/ 24 levels "0","1","2","3",..: 18 18 18 18 18 19 19 19 20 20 ...
##  $ Min        : Factor w/ 6 levels "0","10","20",..: 1 2 3 5 6 2 3 4 1 2 ...
##  $ dateOnly   : Date, format: "2016-01-11" "2016-01-11" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Análise exploratória
g <- ggplot(df)

# Datas
df %>%
  group_by(WeekStatus) %>%
  summarize(Appliances = mean(Appliances)) %>%
  ggplot + geom_bar(aes(x=WeekStatus, y=Appliances), stat="identity")  + labs(title="WeekStatus"
, x="WeekStatus", y="Consumo de energia médio (10 min)")
```

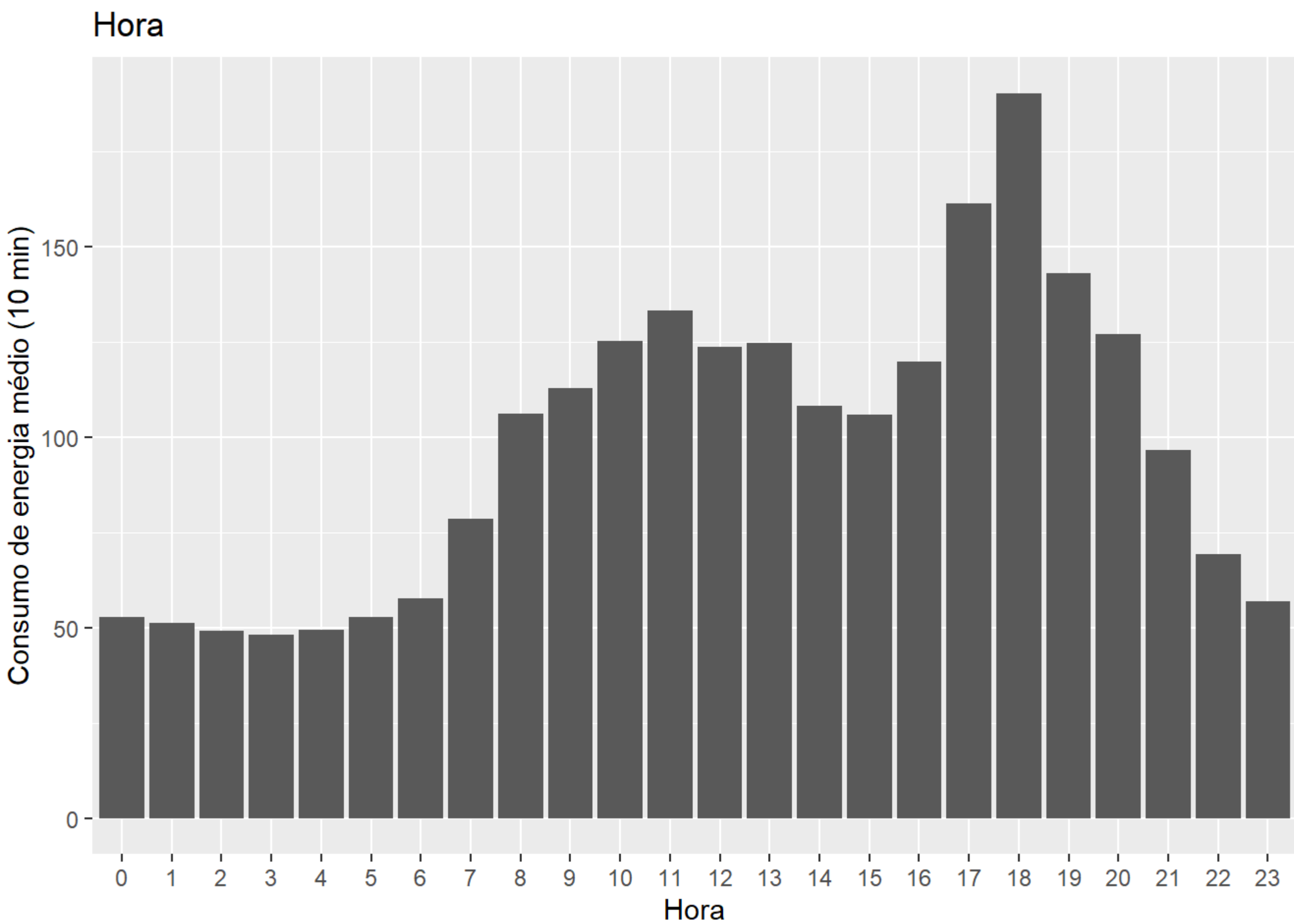

```
df %>%
  group_by(Day_of_week) %>%
  summarize(Appliances = mean(Appliances)) %>%
  ggplot + geom_bar(aes(x=Day_of_week, y=Appliances), stat="identity") + scale_x_discrete(limits
=c('Sunday', 'Monday','Thursday','Wednesday','Tuesday','Friday', 'Saturday')) + labs(title="Dia
 da Semana", x="Dia da Semana", y="Consumo de energia médio (10 min)")
```



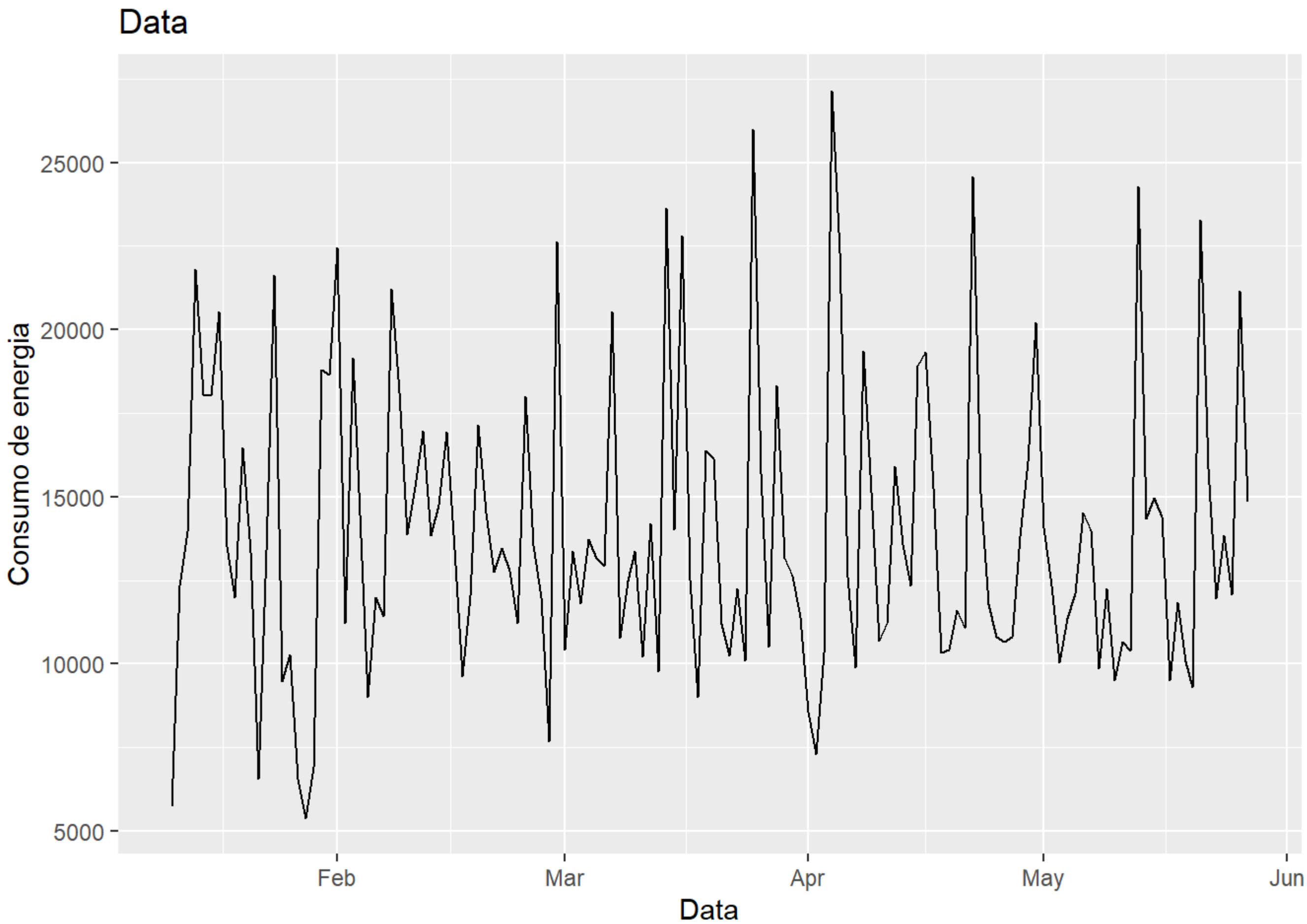

```
df %>%
  group_by(Dia) %>%
  summarize(Appliances = mean(Appliances)) %>%
  ggplot + geom_bar(aes(x=Dia, y=Appliances), stat="identity")  + labs(title="Dia do mês", x="Di
a do mês", y="Consumo de energia médio (10 min)")
```



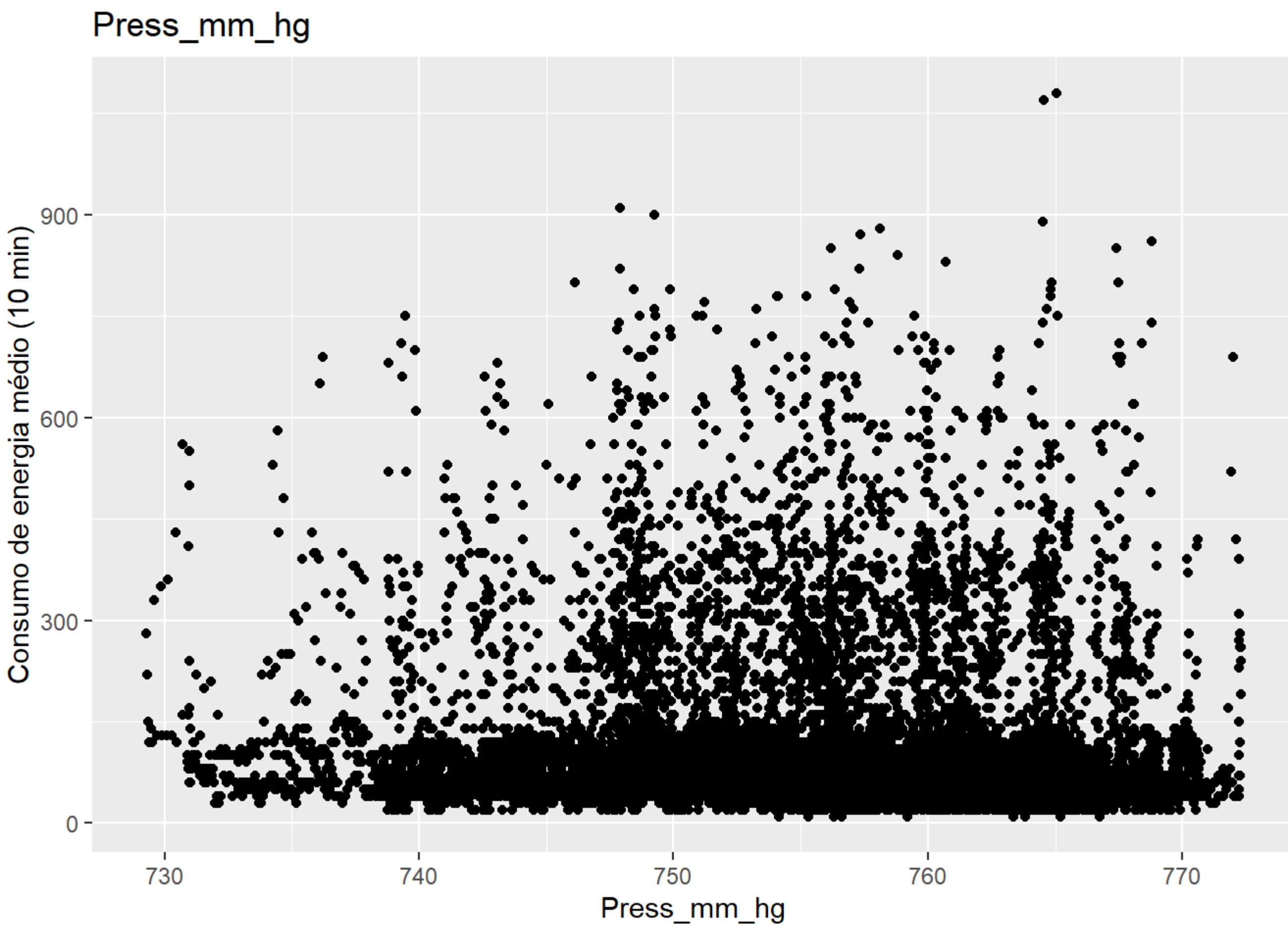

```
df %>%
  group_by(Hora) %>%
  summarize(Appliances = mean(Appliances)) %>%
  ggplot + geom_bar(aes(x=Hora, y=Appliances), stat="identity")  + labs(title="Hora", x="Hora",
 y="Consumo de energia médio (10 min)")
```



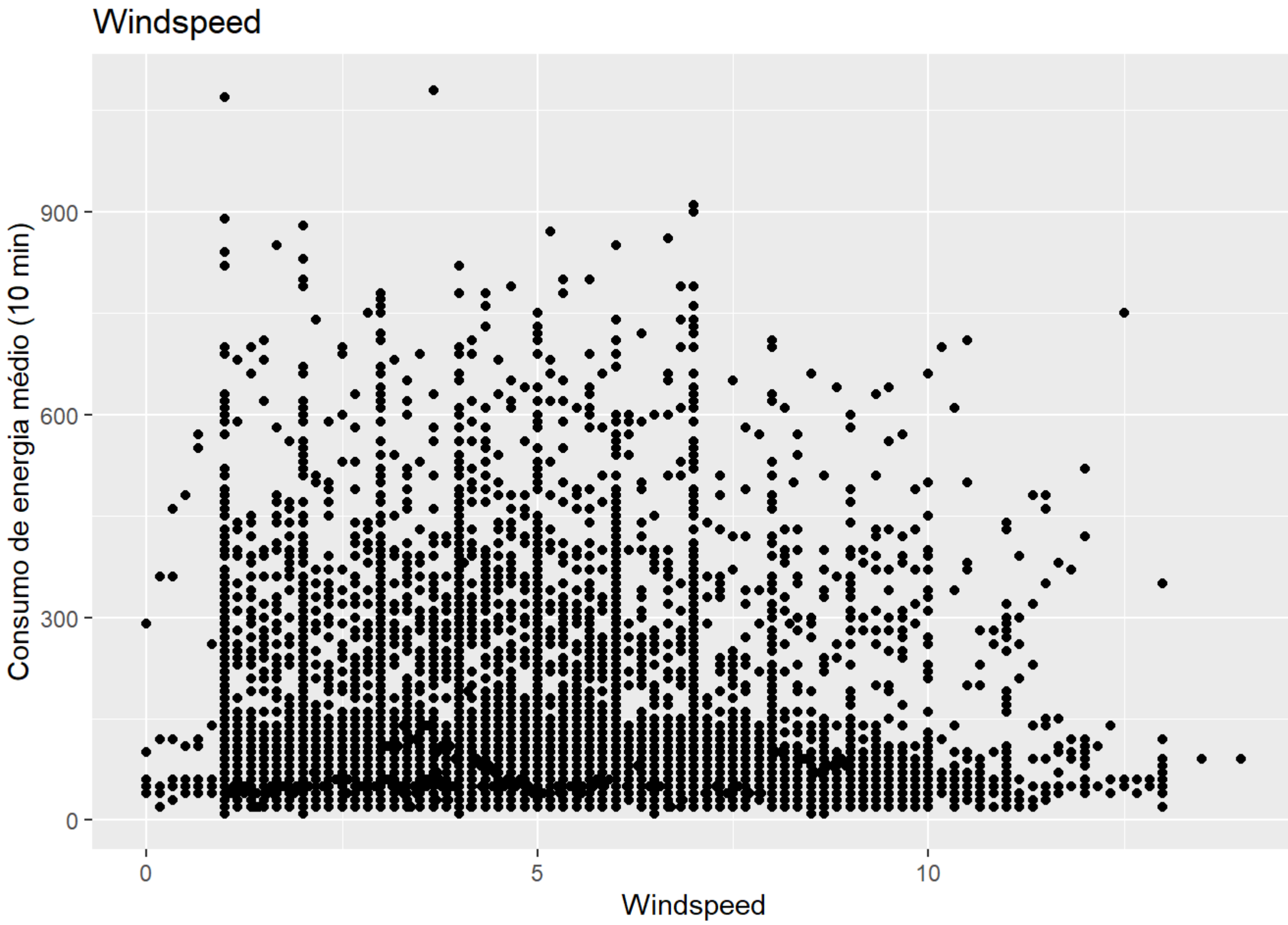

```
# É possível verificar que a diferença no consumo de energia entre dias úteis e finais de semana
é pequena
# Já para o dia da semana, mês e hora, parece existir grande variação no consumo, indicando que
 devem ser boas
# variáveis preditoras

# Transformando as variáveis de dia da semana em fator
df$Day_of_week[df$Day_of_week == 'Sunday'] <- 1
df$Day_of_week[df$Day_of_week == 'Monday'] <- 2
df$Day_of_week[df$Day_of_week == 'Thursday'] <- 3
df$Day_of_week[df$Day_of_week == 'Wednesday'] <- 4
df$Day_of_week[df$Day_of_week == 'Tuesday'] <- 5
df$Day_of_week[df$Day_of_week == 'Friday'] <- 6
df$Day_of_week[df$Day_of_week == 'Saturday'] <- 7
df$Day_of_week <- as.factor(df$Day_of_week)

# Date
df %>%
  group_by(dateOnly) %>%
  summarize(Appliances = sum(Appliances)) %>%
  ggplot + geom_line(aes(x=dateOnly, y=Appliances), stat="identity") + labs(title="Data", x="Dat
a", y="Consumo de energia")
```



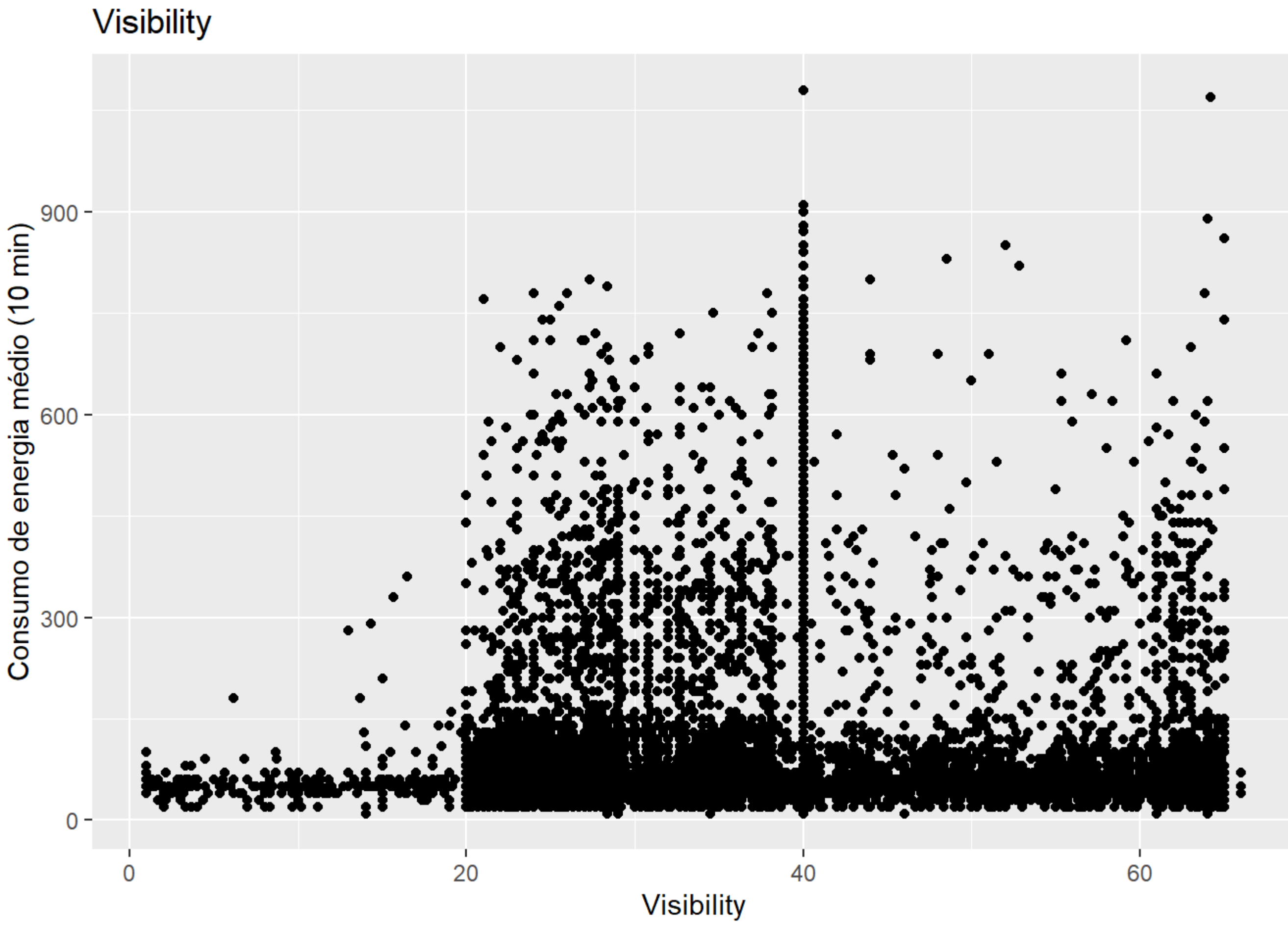

```
# O consumo parece variar de forma relevante ao longo do tempo

# Relações variáveis numéricas
g + geom_point(aes(x=Press_mm_hg, y=Appliances)) + labs(title="Press_mm_hg",x="Press_mm_hg", y=
"Consumo de energia médio (10 min)")
```



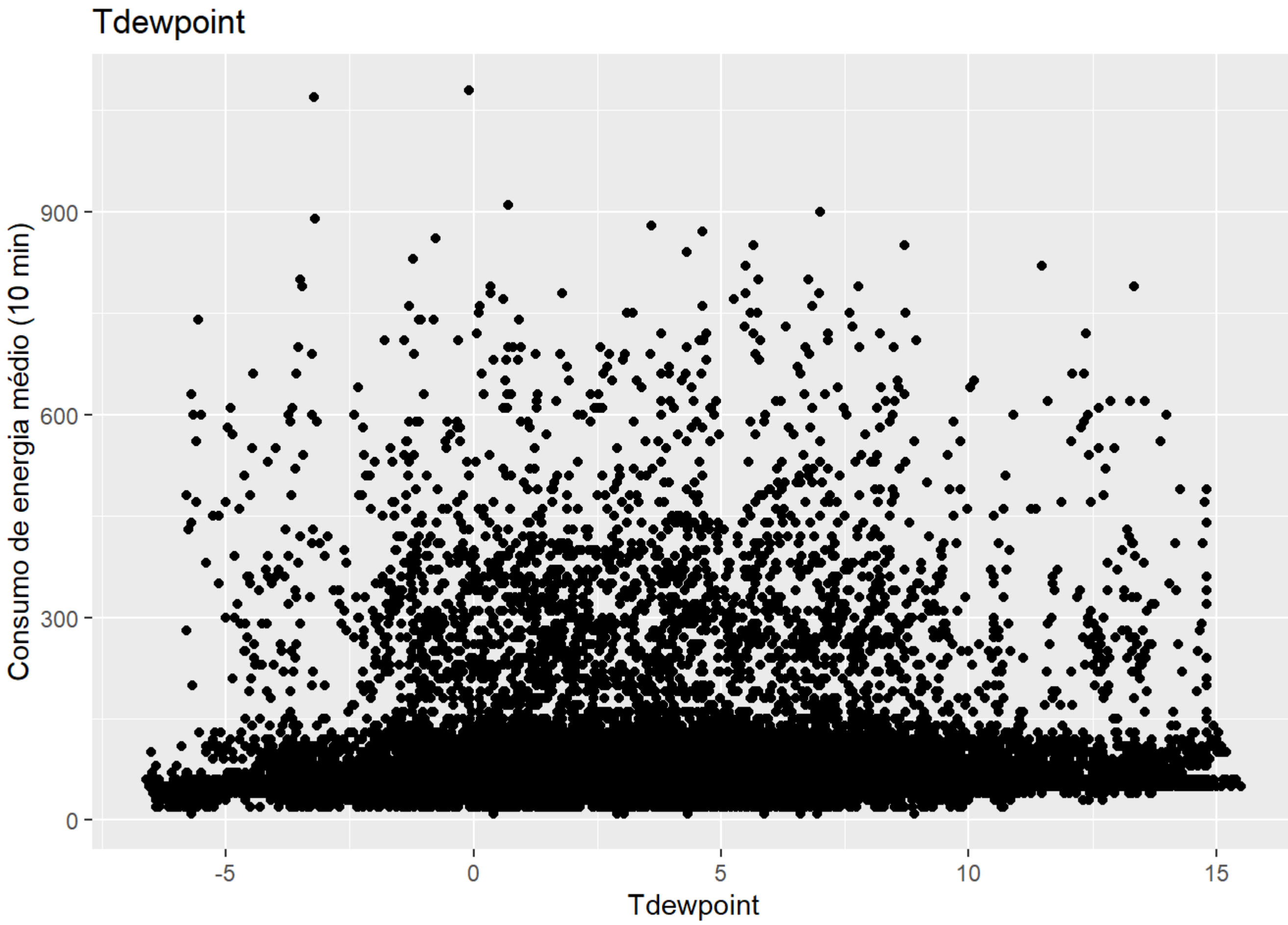

```
g + geom_point(aes(x=Windspeed, y=Appliances)) + labs(title="Windspeed", x="Windspeed", y="Consu
mo de energia médio (10 min)")
```



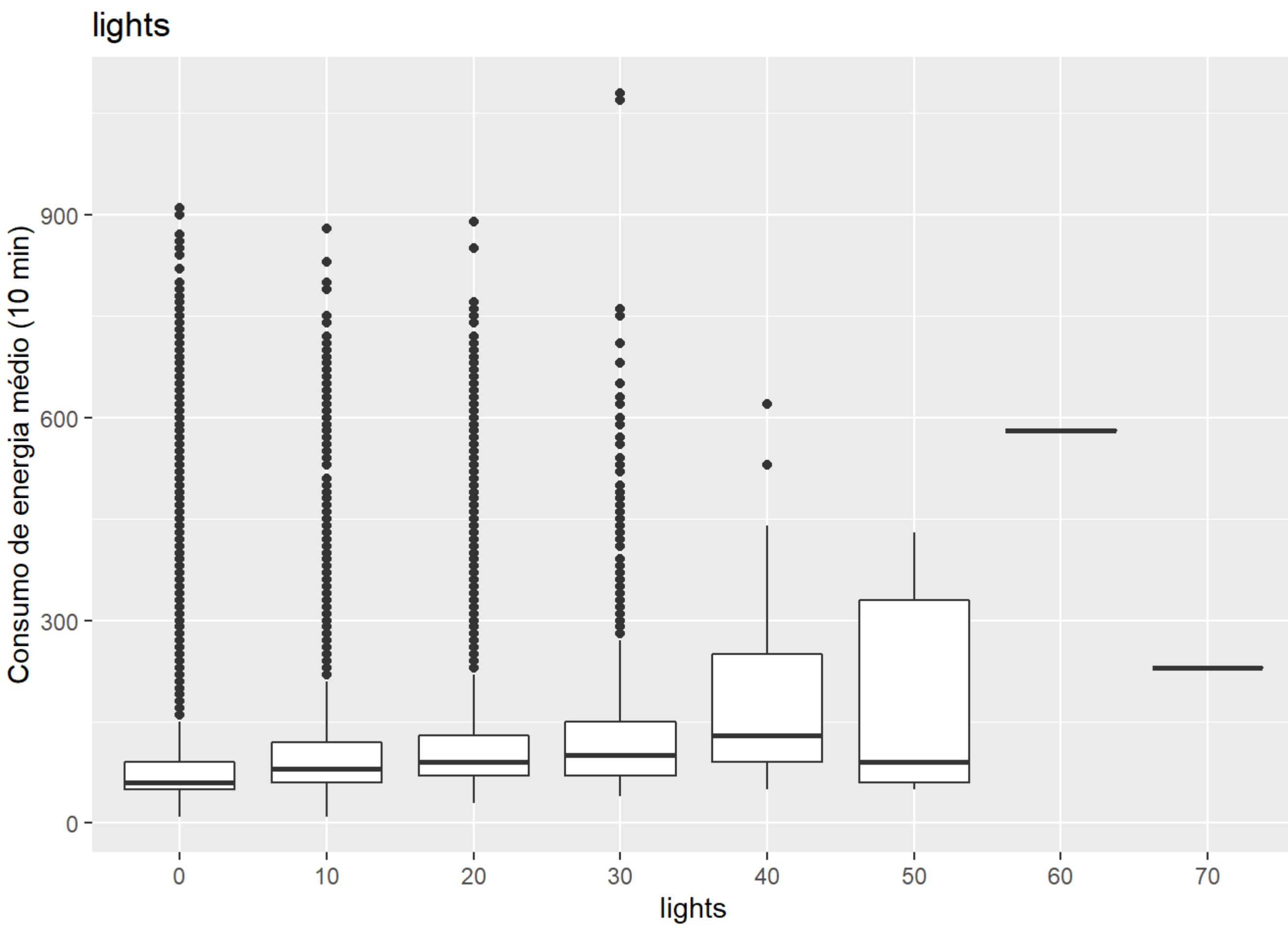

```
g + geom_point(aes(x=Visibility, y=Appliances)) + labs(title="Visibility", x="Visibility", y="Co
nsumo de energia médio (10 min)")
```



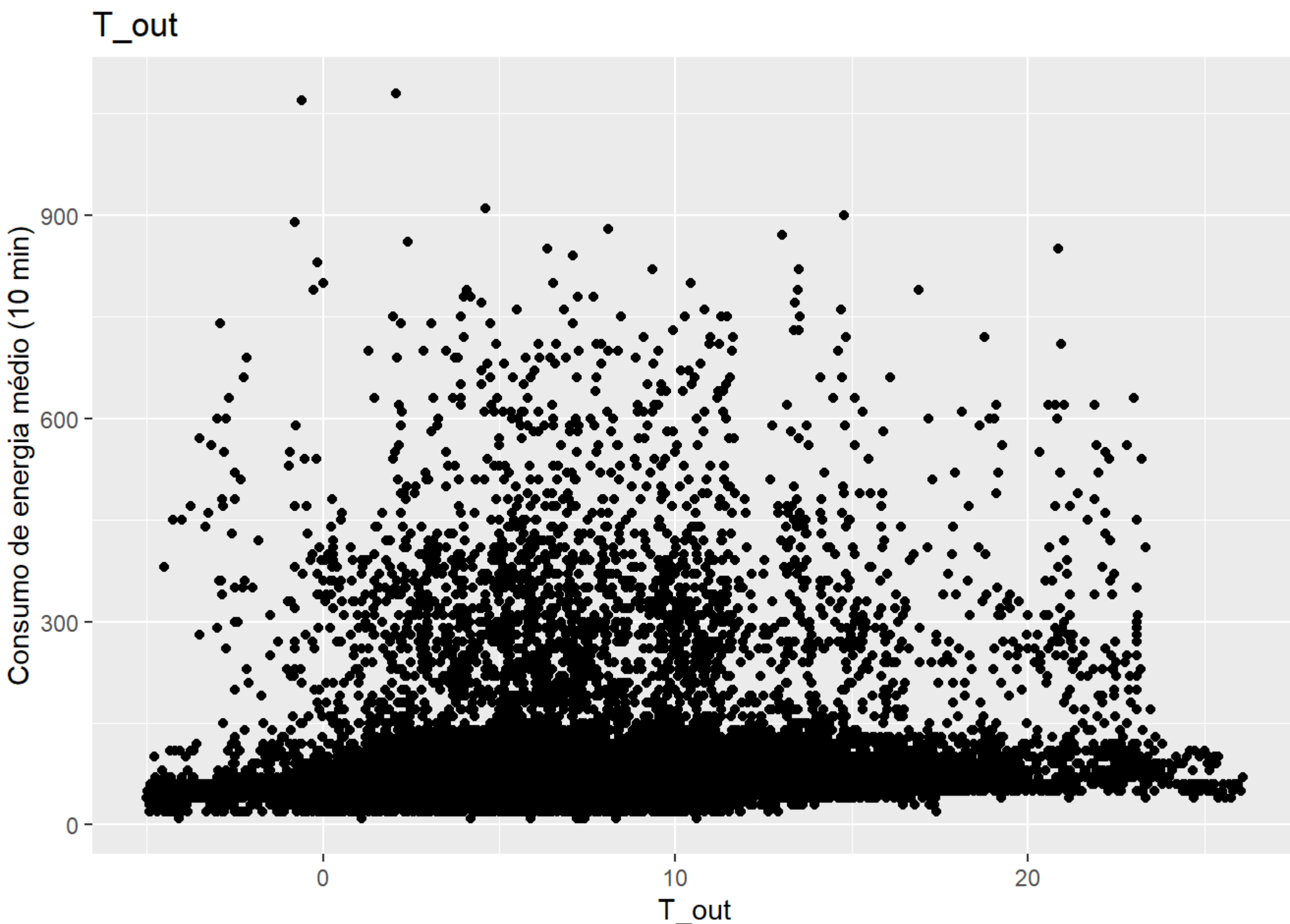

```
g + geom_point(aes(x=Tdewpoint, y=Appliances)) + labs(title="Tdewpoint", x="Tdewpoint", y="Consu
mo de energia médio (10 min)")
```



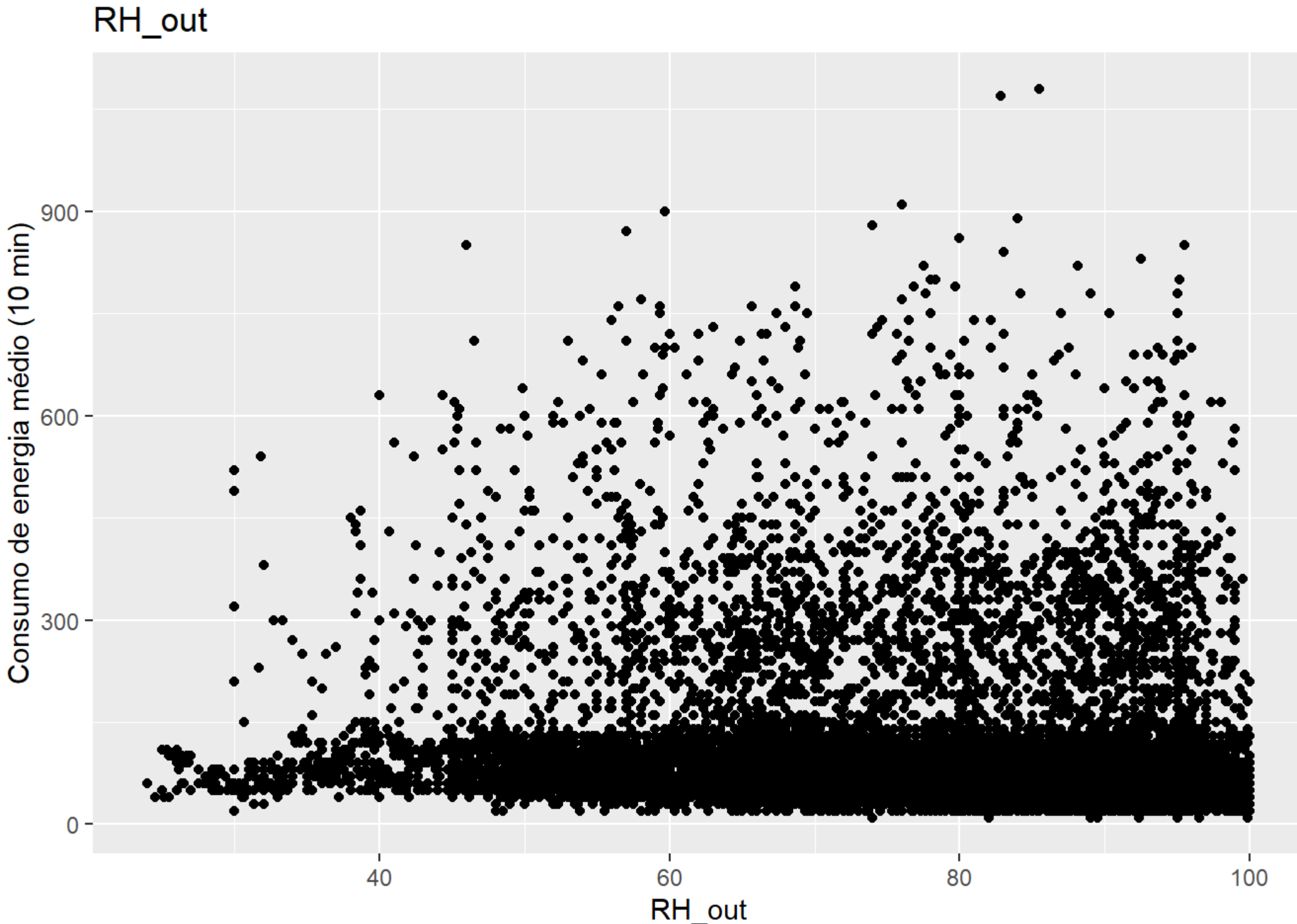

```
# A relaçoes entre as variáveis não parecem relevantes, o que pode indicar que o modelo não cons
iga grande
# acurácia na previsão

# Mais Relações variáveis numéricas
g + geom_boxplot(aes(x=as.factor(lights), y=Appliances)) + labs(title="lights",x="lights", y="Co
nsumo de energia médio (10 min)")
```



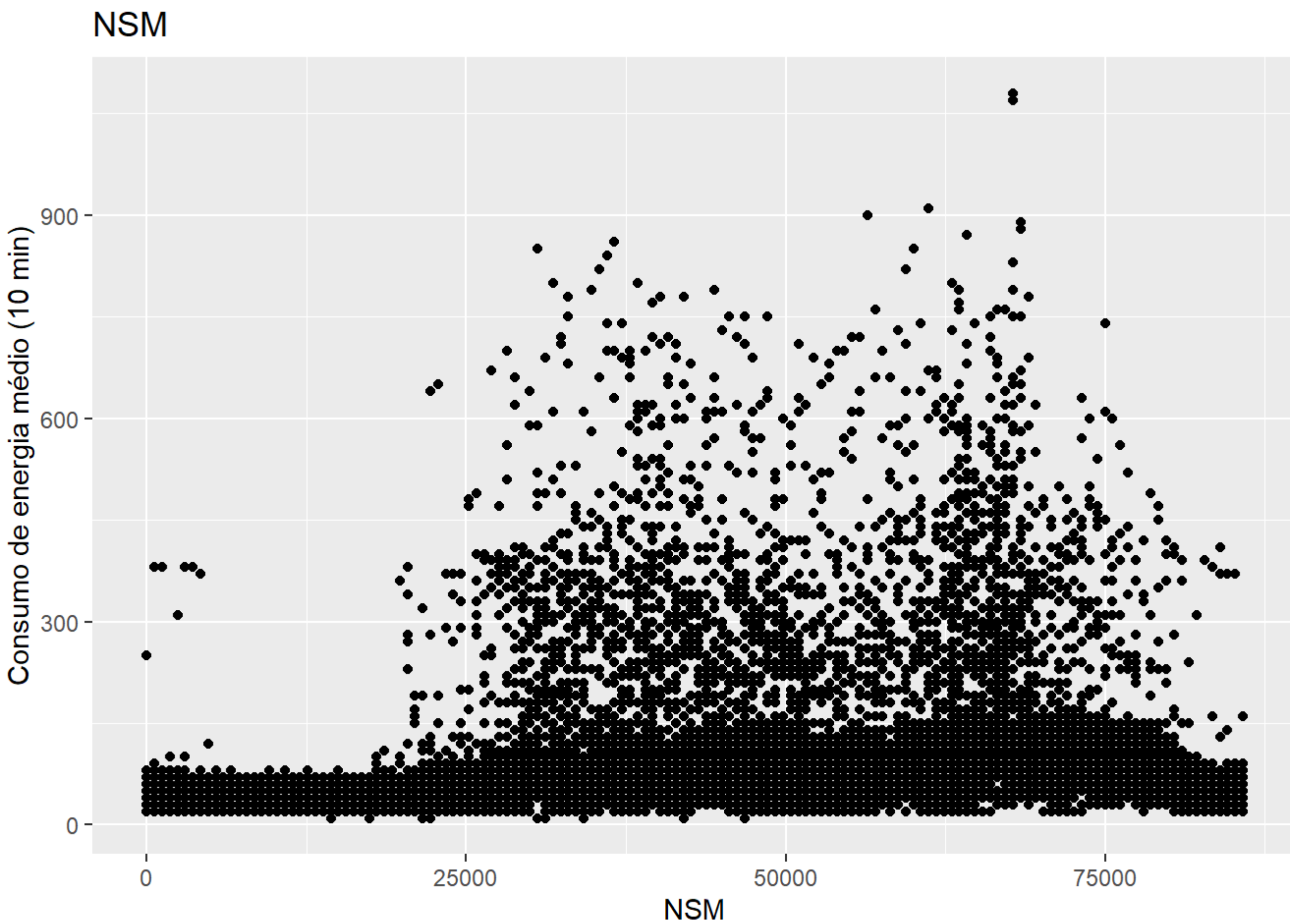

```
g + geom_point(aes(x=T_out, y=Appliances)) + labs(title="T_out", x="T_out", y="Consumo de energi
a médio (10 min)")
```



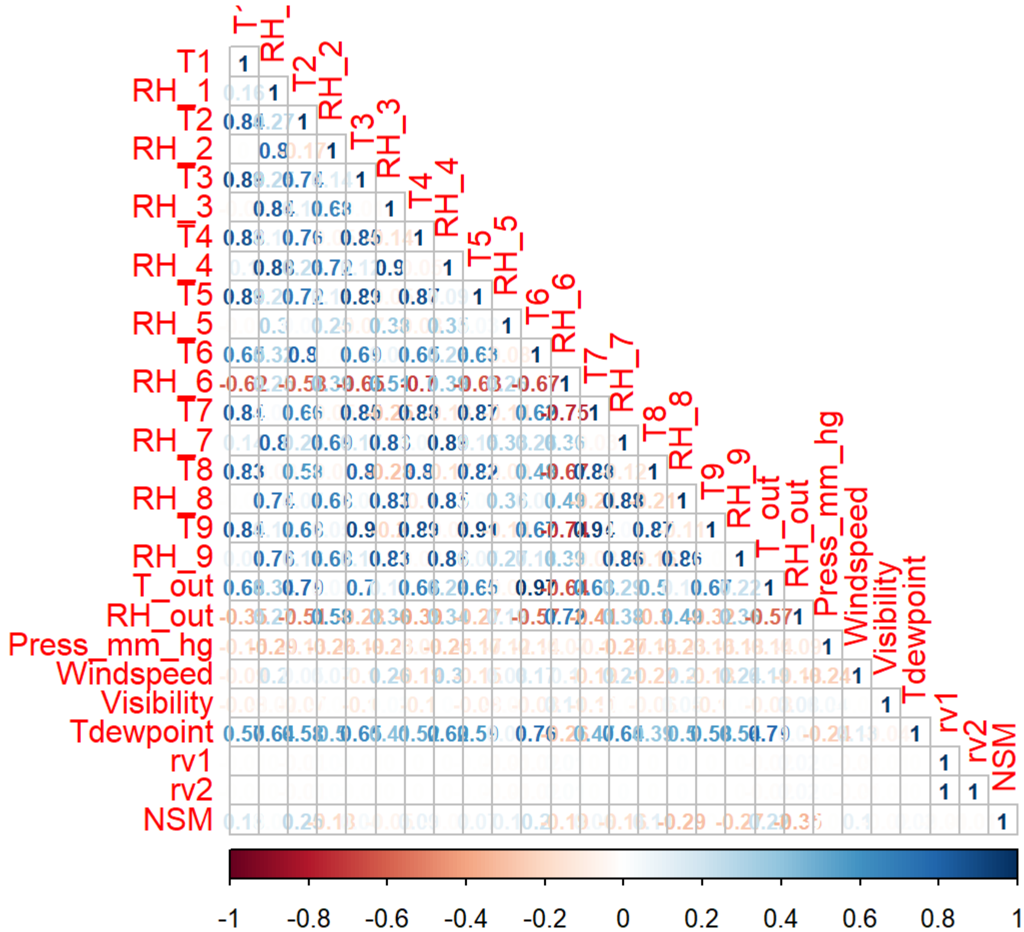

```
g + geom_point(aes(x=RH_out, y=Appliances)) + labs(title="RH_out", x="RH_out", y="Consumo de ene
rgia médio (10 min)")
```




```
g + geom_point(aes(x=NSM, y=Appliances)) + labs(title="NSM", x="NSM", y="Consumo de energia médi
o (10 min)")
```




```
# Mais uma vez as variáveis não demostram fortes relações, com exceção da variável lights, que p
ossui alguma
# variabilidade nos dados

# Correlações
variaveis <- c('T1','RH_1','T2','RH_2','T3','RH_3','T4','RH_4','T5','RH_5','T6','RH_6','T7','RH_
7','T8','RH_8','T9','RH_9','T_out','RH_out','Press_mm_hg','Windspeed','Visibility','Tdewpoint',
'rv1','rv2', 'NSM')
correlacoes = cor(df[,..variaveis], method="pearson")
corrplot(correlacoes, type="lower", method = 'number', number.cex = 0.7)
```




```
# As variáveis T_out com T6 e T7 com T9 possuem grande correlação, não devendo entrar no modelo
 juntas
# Com isso, iremos remove-las

df$T6 <- NULL
df$T9 <- NULL
variaveis <- c('T1','RH_1','T2','RH_2','T3','RH_3','T4','RH_4','T5','RH_5','RH_6','T7','RH_7','T
8','RH_8','RH_9','T_out','RH_out','Press_mm_hg','Windspeed','Visibility','Tdewpoint','rv1','rv2'
, 'NSM')

# Modelo
library(randomForest)
```

```
## randomForest 4.6-14
```

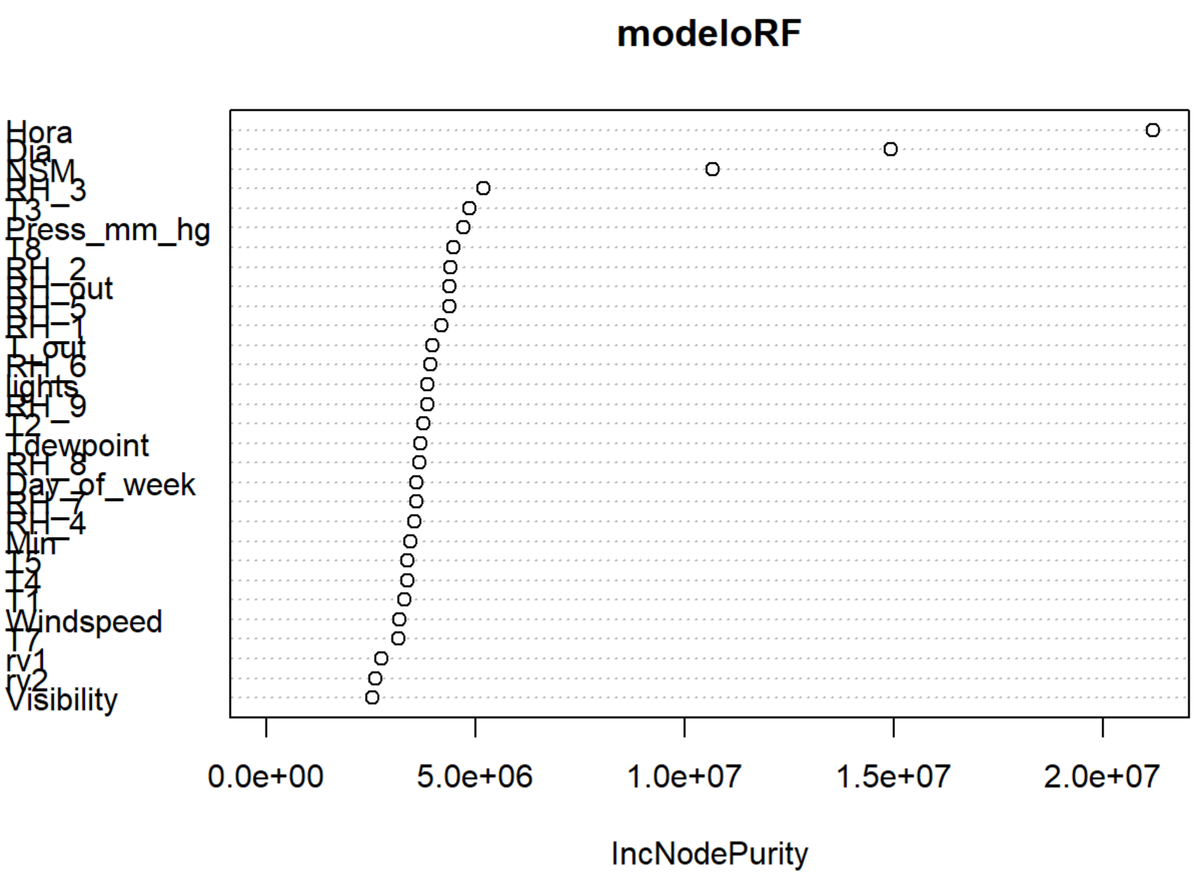
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

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

```r
library(caret)

# Normalizando os dados
for (item in variaveis){
  X <- df[[item]]
  df[[item]] <- (X - min(X)) / (max(X) - min(X))
}

#Separando o Dataset
trainSet <- df[1:nrow(dfTrain)]
testSet <- df[(nrow(dfTrain)+1):nrow(df)]

#Treinando o modelo
variaveisModelo <- 'Appliances ~ T1+RH_1+T2+RH_2+T3+RH_3+T4+RH_4+T5+RH_5+RH_6+T7+RH_7+T8+RH_8+RH
_9+T_out+RH_out+Press_mm_hg+Windspeed+Visibility+Tdewpoint+rv1+rv2+NSM+lights+Dia+Hora+Mes+Min+D
ay_of_week'
variaveisModelo <- as.formula(variaveisModelo)
modeloRF <- randomForest(variaveisModelo, data = trainSet)
varImpPlot(modeloRF)
```



```
# Escolhendo as variáveis mais importantes
variaveisModelo <- 'Appliances ~ Hora + Dia + NSM + RH_3 + Press_mm_hg + RH_2 + T3 + RH_5 + T8 +
lights + RH_1 + T2 + RH_out'
variaveisModelo <- as.formula(variaveisModelo)

# Definindo cross validation
ctrl <- trainControl(method = "cv", number=5)

# Treinando o modelo de Regressao Logística Multilinear
modeloRLM <- train(variaveisModelo, data=trainSet, method='glm', metric='Rsquared', trControl=ct
rl)
print(modeloRLM)
```

```
## Generalized Linear Model
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11842, 11843, 11842, 11842, 11843
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   90.69459  0.2222008  52.64719
```

```
# Não apresentou performance muito boa

# Treinando o modelo de Support Vector Machines
modeloSVM <- train(variaveisModelo, data=trainSet, method='svmLinear', metric='Rsquared', trCont
rol=ctrl)
print(modeloSVM)
```

```
## Support Vector Machines with Linear Kernel
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11842, 11844, 11842, 11843, 11841
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   98.62133  0.1805948  42.66441
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# Também não teve uma performance muito boa. Teremos que utilizar modelos não lineares.

#Treinando o modelo de Stochastic Gradient Boosting
modeloSGB <- train(variaveisModelo, data=trainSet, method='gbm', metric='Rsquared', trControl=ct
rl)
```

```
print(modeloSGB)
```

```
## Stochastic Gradient Boosting
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11842, 11843, 11843, 11843, 11841
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      94.39849  0.1712010  52.08393
##   1                  100      93.09455  0.1868089  51.06696
##   1                  150      92.32614  0.1987845  50.73352
##   2                   50      92.47867  0.2013537  50.45018
##   2                  100      90.42620  0.2335160  49.16163
##   2                  150      89.14522  0.2532023  48.53764
##   3                   50      90.94045  0.2281819  49.18563
##   3                  100      88.56505  0.2642953  47.74065
##   3                  150      87.17171  0.2858603  46.89789
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Rsquared was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# A performance já melhorou muito, com diminuição do RMSE e aumento do R-squared

#Treinando o modelo de Extreme Gradient Boosting
modeloEGB <- train(variaveisModelo, data=trainSet, method='xgbLinear', metric='Rsquared', trCont
rol=ctrl)
print(modeloEGB)
```

```
## eXtreme Gradient Boosting
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11843, 11842, 11842, 11843, 11842
## Resampling results across tuning parameters:
##
##   lambda  alpha  nrounds  RMSE      Rsquared   MAE
##   0e+00   0e+00   50      76.64989  0.4437942  39.53082
##   0e+00   0e+00  100      74.69476  0.4727500  38.05926
##   0e+00   0e+00  150      74.02046  0.4838730  37.41458
##   0e+00   1e-04   50      76.64987  0.4437944  39.53059
##   0e+00   1e-04  100      74.69476  0.4727500  38.05904
##   0e+00   1e-04  150      74.02046  0.4838729  37.41453
##   0e+00   1e-01   50      76.72203  0.4427753  39.52944
##   0e+00   1e-01  100      74.55159  0.4748381  37.78579
##   0e+00   1e-01  150      73.59808  0.4898370  36.93168
##   1e-04   0e+00   50      76.74837  0.4424136  39.53376
##   1e-04   0e+00  100      74.33958  0.4774332  37.85589
##   1e-04   0e+00  150      73.79728  0.4867395  37.21331
##   1e-04   1e-04   50      76.74837  0.4424136  39.53376
##   1e-04   1e-04  100      74.33958  0.4774332  37.85589
##   1e-04   1e-04  150      73.79728  0.4867395  37.21331
##   1e-04   1e-01   50      76.83381  0.4411659  39.68656
##   1e-04   1e-01  100      74.44662  0.4763733  37.92700
##   1e-04   1e-01  150      73.69071  0.4889178  37.20708
##   1e-01   0e+00   50      76.32582  0.4490071  39.07332
##   1e-01   0e+00  100      74.51308  0.4747331  37.83607
##   1e-01   0e+00  150      73.49191  0.4898025  37.10640
##   1e-01   1e-04   50      76.32582  0.4490071  39.07332
##   1e-01   1e-04  100      74.51308  0.4747331  37.83607
##   1e-01   1e-04  150      73.49191  0.4898025  37.10641
##   1e-01   1e-01   50      76.52119  0.4461837  39.20424
##   1e-01   1e-01  100      74.46526  0.4753976  37.74890
##   1e-01   1e-01  150      73.86948  0.4847926  37.05803
##
## Tuning parameter 'eta' was held constant at a value of 0.3
## Rsquared was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 150, lambda = 0, alpha =
##  0.1 and eta = 0.3.
```

```r
# Melhor resultado até o momento

#Otimizando o melhor modelo EGB, o qual apresentou melhor resultado
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.6.2
```

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

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

```
ctrl <- trainControl(method = "cv", number=5)
grid <- expand.grid(nrounds = c(100,200),
                    max_depth = c(10,20,30),
                    eta = 0.1,
                    gamma = 0,
                    colsample_bytree = c(0.3,0.5,0.7),
                    min_child_weight = 1,
                    subsample = 1)
modeloEGB <- train(variaveisModelo, data=trainSet, method='xgbTree', metric='Rsquared', trContro
l=ctrl, tuneGrid=grid)
print(modeloEGB)
```

```
## eXtreme Gradient Boosting
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11842, 11843, 11842, 11842, 11843
## Resampling results across tuning parameters:
##
##   max_depth  colsample_bytree  nrounds  RMSE      Rsquared   MAE
##   10         0.3               100      70.88503  0.5337330  35.32244
##   10         0.3               200      69.51511  0.5479335  34.22081
##   10         0.5               100      70.17790  0.5387836  34.16586
##   10         0.5               200      69.02793  0.5516525  33.14530
##   10         0.7               100      70.73663  0.5298340  34.16955
##   10         0.7               200      69.46131  0.5452596  33.18696
##   20         0.3               100      69.41678  0.5533937  33.59665
##   20         0.3               200      69.30347  0.5542881  33.56030
##   20         0.5               100      67.79821  0.5673827  31.65363
##   20         0.5               200      67.76470  0.5676720  31.65864
##   20         0.7               100      68.82208  0.5536678  31.76813
##   20         0.7               200      68.80280  0.5539650  31.77710
##   30         0.3               100      69.74803  0.5519535  33.66504
##   30         0.3               200      69.70564  0.5522365  33.67153
##   30         0.5               100      67.82561  0.5676279  31.93625
##   30         0.5               200      67.81335  0.5677137  31.94677
##   30         0.7               100      68.63146  0.5560588  31.66936
##   30         0.7               200      68.62309  0.5561693  31.68118
##
## Tuning parameter 'eta' was held constant at a value of 0.1
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
##
## Tuning parameter 'subsample' was held constant at a value of 1
## Rsquared was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 200, max_depth = 30, eta
##  = 0.1, gamma = 0, colsample_bytree = 0.5, min_child_weight = 1 and subsample
##  = 1.
```

```
# Melhores parâmetros
modeloEGB$bestTune
```

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 16     200        30 0.1     0              0.5                1         1
```

```
# Vamos treinar o modelo com os melhores parametros
modeloEGB <- train(variaveisModelo, data=trainSet, method='xgbTree', metric='Rsquared', trContro
l=ctrl, tuneGrid=modeloEGB$bestTune)
print(modeloEGB)
```

```
## eXtreme Gradient Boosting
##
## 14803 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11842, 11843, 11844, 11841, 11842
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   68.0726  0.5623604  31.75394
##
## Tuning parameter 'nrounds' was held constant at a value of 200
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
##
## Tuning parameter 'subsample' was held constant at a value of 1
```

```
# Por último, vamos analisar como o modelo se sai com os dados de Teste
previsao <- predict(modeloEGB, testSet)
residuos <- testSet$Appliances - previsao
# RMSE nos dados de Teste
rmse(testSet$Appliances, previsao)
```

```
## [1] 66.66934
```

```
# R-Squared nos dados de Teste
tss <- sum((testSet$Appliances - mean(testSet$Appliances))^2)
rss <- sum(residuos^2)
1-(rss/tss)
```

```
## [1] 0.5694412
```

```
# Concluindo, o modelo de Extreme Gradient Boosting apresentou o melhor resultado, utilizando ap
enas
# 13 das 35 variáveis disponíveis, filtradas pela ordem de importância utilizando um modelo de R
andom Forest.
# O r-squared, porém, não atingiu um valor alto. Isso já era esperado, dado que os atributos pos
suem pouca
# relação com a variável preditora, conforme visto na análise exploratória.
```