Projeto 1 - R com Azure Data Science Academy

Carlos Augusto Schneider

04/10/2022

ETAPA 1 - Carregamento e preparação do dataframe

Inicialmente, as linhas do script para definição do diretório de trabalho e carregar os pacotes de funções que serão utilizados.

```
## [1] "/home/carlos/FCD/BigDataRAzure/projetos/projeto1"
```

Para carregar o dataframe, optou-se por utilizar o pacote **Haven**, para leitura do arquivo SPSS (formato do arquivo pacote) relativo ao pacote estatístico da IBM.

Abaixo, segue a descrição de cada uma das variáveis do dataframe:

1. **CAR:** nome do carro;
2. **Make:** Fabricante;
3. **Model:** Modelo do carro;
4. **Price:** Preço;
5. **Engine:** Potência do motor;
6. **Torque;** Força de rotação gerada pelo motor;
7. **Brakes:** Tipo do freio;
8. **Drive:** Tipo de tração;
9. **Battery:** Capacidade da bateria;
10. **Range:** Alcance do carro em kilômetros;
11. **Wheelbase:** distância entre os eixos;
12. **Length:** Comprimento;
13. **Width:** Largura;
14. **Height:** Altura;
15. **Weight:** Peso;
16. **Permissible Weight:** Peso máximo permitido, incluindo a carga;
17. **Capacity:** Capacidade de carga;
18. **Seats:** Número de assentos;
19. **Doors:** Número de portas;
20. **Tire Size:** Tamanho do pneu;
21. **Max Speed:** Velocidade máxima
22. **Boot capacity:** Capacidade do porta-malas;

23. **Acceleration:** Aceleração;
24. **DC:** Potência máxima de carregamento;
25. **Energy_consumption:** Consumo de energia. É a variável target (que pretende-se prever).

```
df <- read_spss("FEV-dataset-SPSS.sav")
dim(df)
```

```
## [1] 53 25
```

```
Variaveis <- colnames(df)
Tipos <- sapply(df, mode)
tabela <- as.data.frame(Variaveis,Tipos, colnames = c("Variaveis","Tipos"))
kable(tabela,caption = "Tipos das variáveis",format = "pipe")
```

Table 1: Tipos das variáveis

	Variaveis
character	CAR
character	Make
character	Model
numeric	Price
numeric	Power
numeric	Torque
character	Brakes
character	Drive
numeric	Battery
numeric	Range
numeric	Wheelbase
numeric	Length
numeric	Width
numeric	Height
numeric	Weight
numeric	Permissible_weight
numeric	Capacity
numeric	Seats
numeric	Doors
numeric	Tire_size
numeric	Max_speed
numeric	Boot_capacity
numeric	Acceleration
numeric	DC
numeric	Energy_consumption

```
rm(tabela,Tipos,Variaveis)
```

É necessário converter variáveis do tipo caractere em fator. Há ainda algumas variáveis numéricas com poucos valores, que na realidade também se enquadram na categoria fator, conforme código abaixo:

```
z <- c("Seats","Doors","Tire_size","DC")
lapply(df[z], unique)
```

```
## $Seats
## [1] 5 4 2 7 8 6
##
```

```
## $Doors
## [1] 5 3 4
##
## $Tire_size
## [1] 19 20 16 17 18 21 15 14
##
## $DC
##  [1] 150  50 100  37 110 225 270  40  22 125
```

O código abaixo faz a conversão dessas variáveis para o tipo Fator.

```
df1 <- df %>%
  mutate_if(is.character, factor)

df1[,z] <- lapply(df1[,z],factor)
rm(z)
```

O passo seguinte é o tratamento dos valores missing (**NA**) presentes no dataframe.

```
colSums(is.na(df))
```

```
##                CAR               Make              Model              Price
##                  0                  0                  0                  0
##              Power             Torque             Brakes              Drive
##                  0                  0                  1                  0
##            Battery              Range          Wheelbase             Length
##                  0                  0                  0                  0
##              Width             Height             Weight Permissible_weight
##                  0                  0                  0                  8
##           Capacity              Seats              Doors          Tire_size
##                  8                  0                  0                  0
##          Max_speed      Boot_capacity       Acceleration                 DC
##                  0                  1                  3                  0
## Energy_consumption
##                  9
```

Os modelos de veículos sem dados para a variável target (**Energy Comsuption**) não serão úteis para elaboração do modelo preditivo, uma vez que não há como verificar o desempenho do modelo preditivo. Optou-se por excluír essas observações.

```
df1 <- filter(df1, !is.na(Energy_consumption))

which(is.na(df1), arr.ind = TRUE)
```

```
##      row col
## [1,]  43   7
## [2,]  43  22
## [3,]  43  23
## [4,]  44  23
```

```
colnames(df1[c(7,22,23)])
```

```
## [1] "Brakes"        "Boot_capacity" "Acceleration"
```

Restaram apenas dois modelos de veículos com valores NA, nas variáveis **Brakes**, **Boot_capacity** (capacidade do porta-malas) e **Acceleration**. Optou-se por excluí-las do dataset.

```
df1 <- na.omit(df1)
```

```
any(is.na(df1))
```

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

```
dim(df1)
```

```
## [1] 42 25
```

Por fim, optou-se por excluir do dataframe as variáveis **Car**, **Make (fabricante)** e **Model**. Achamos que os modelos preditivos serão mais relevantes se descobrirem o consumo de energia com base em características do veículo, e não em função do modelo específico ou mesmo do fabricante.

```
df1 <- select(df1,-c("CAR","Make","Model"))
```

ETAPA 2 - Análise Exploratória

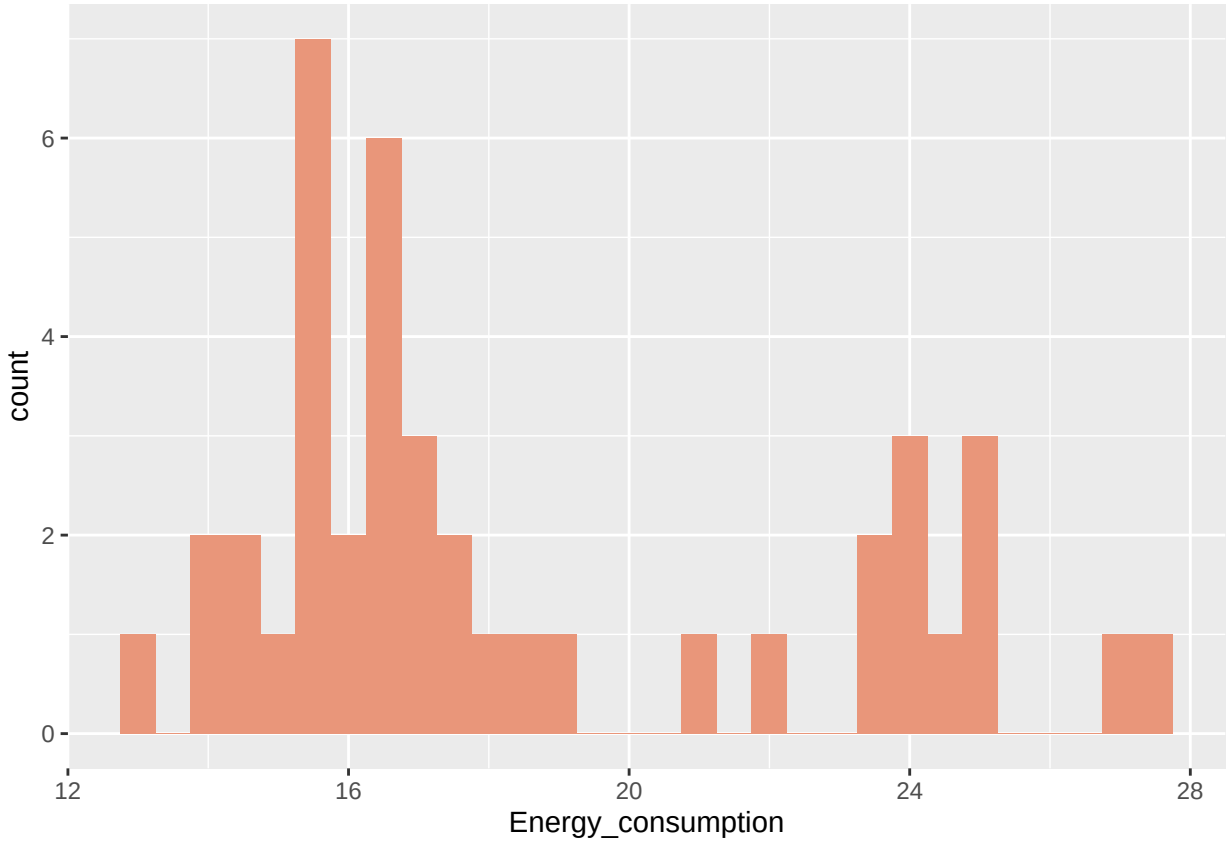
Nessa etapa vamos gerar alguns gráficos para observação da relação das variáveis independentes entre si e com a variável Target.

Inicialmente um histograma da distribuição de frequências da variável target (**Energy_consumption**):

```
summary(df1$Energy_consumption)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.10   15.60   16.88   18.61   22.94   27.55
```

```
ggplot(df1, aes(x = Energy_consumption)) +
  geom_histogram(binwidth = 0.5, fill = "darksalmon")
```



Optamos por utilizar gráficos do tipo boxplot para análise das variáveis do tipo fator com a variável Target e

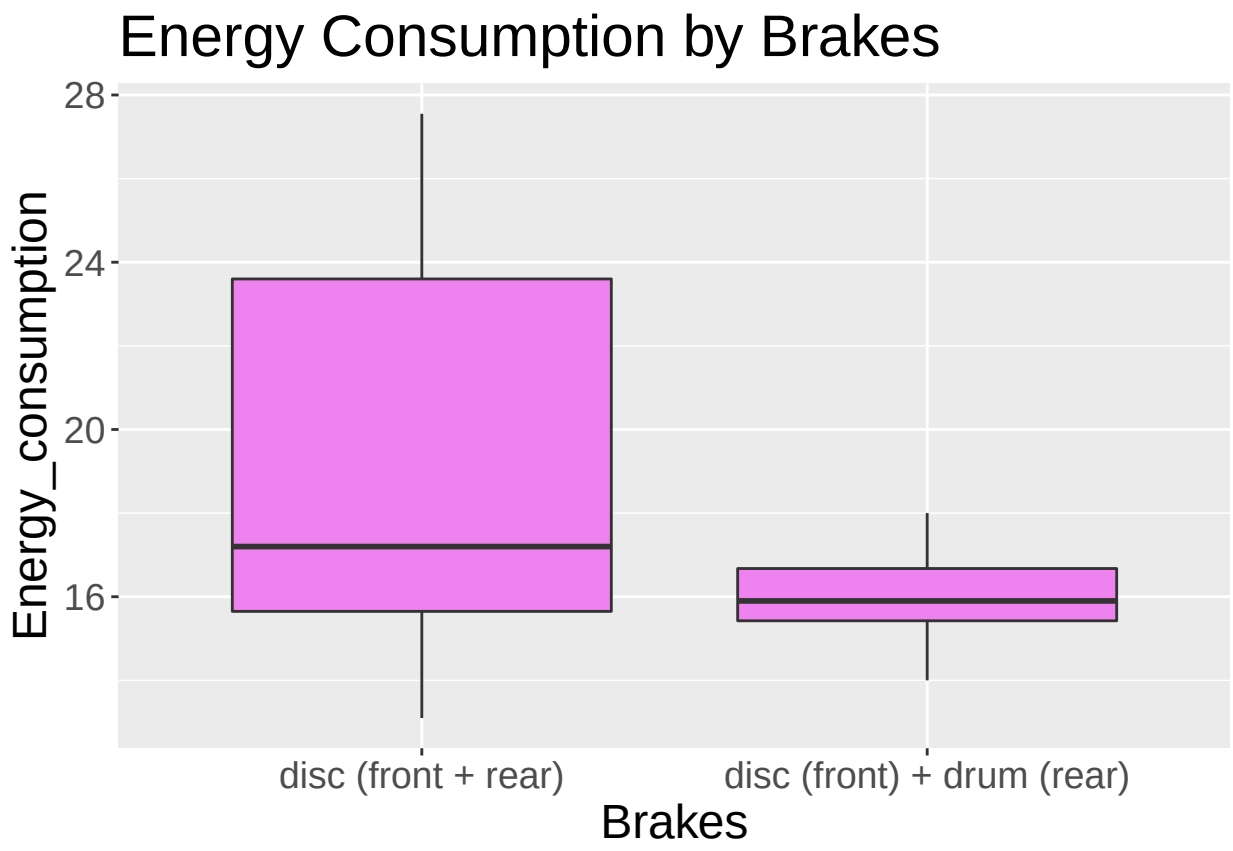
gráficos de pontos (scatterplots) para analisar a relação das variáveis numéricas com a variável Target. Então, vamos primeiro fazer a divisão do dataframe:

```
fator <- sapply(df1, is.factor)

fatores <- colnames(select(df1, which(fator)))
numericas <- colnames(select(df1, which(!fator)))
numericas <- numericas[! numericas %in% c("Energy_consumption")]
```

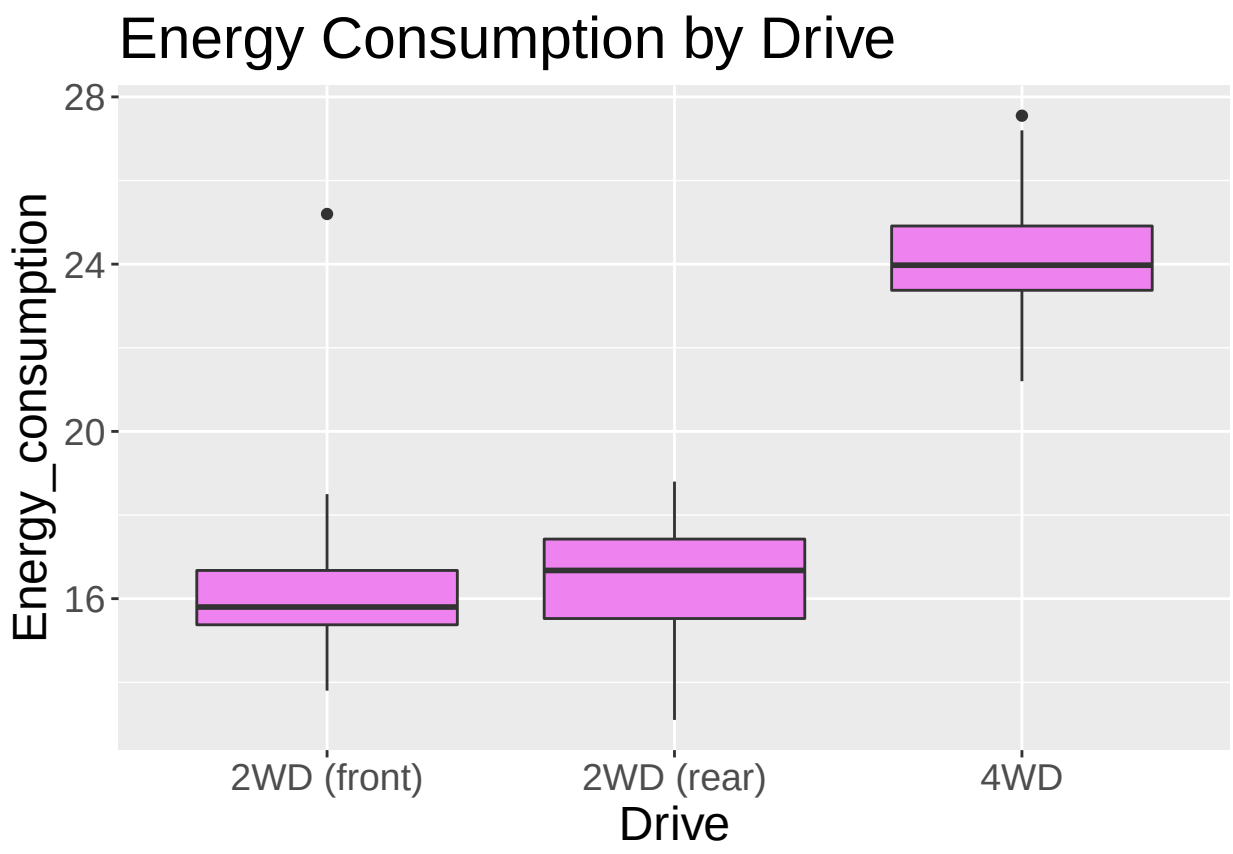
Agora segue o bloco de instruções para a geração dos boxplots:

```
labels_boxplot = list()

for (n in 1:(length(fatores))) {
  labels_boxplot[[n]] <- paste("Energy Consumption by",fatores[n])
  }

plot.boxes  <- function(X, label){
  ggplot(df1, aes_string(x = X, y = "Energy_consumption", group = X)) +
    geom_boxplot(fill = "violet") +
    ggtitle(label) +
    theme(text = element_text(size = 18))
}

Map(plot.boxes, fatores, labels_boxplot)
```

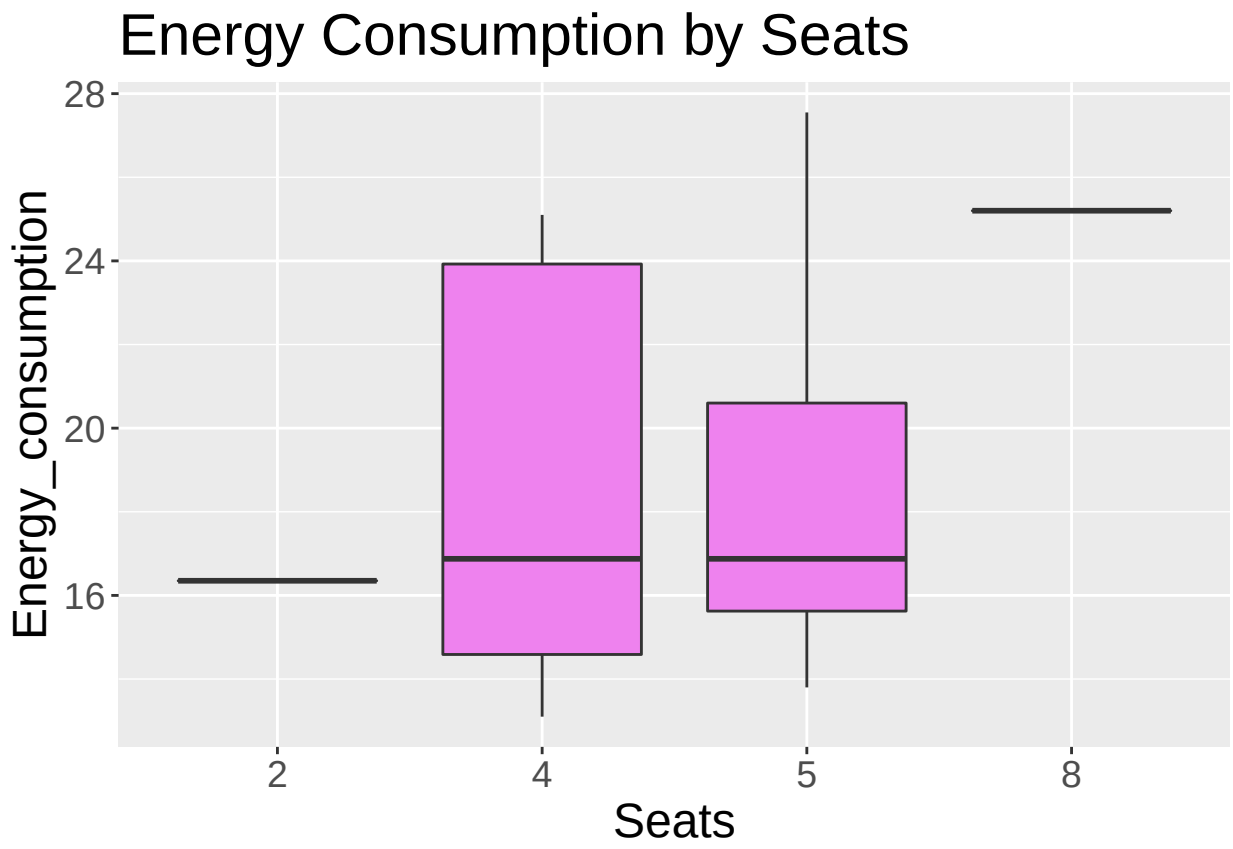
```
## $Brakes
```



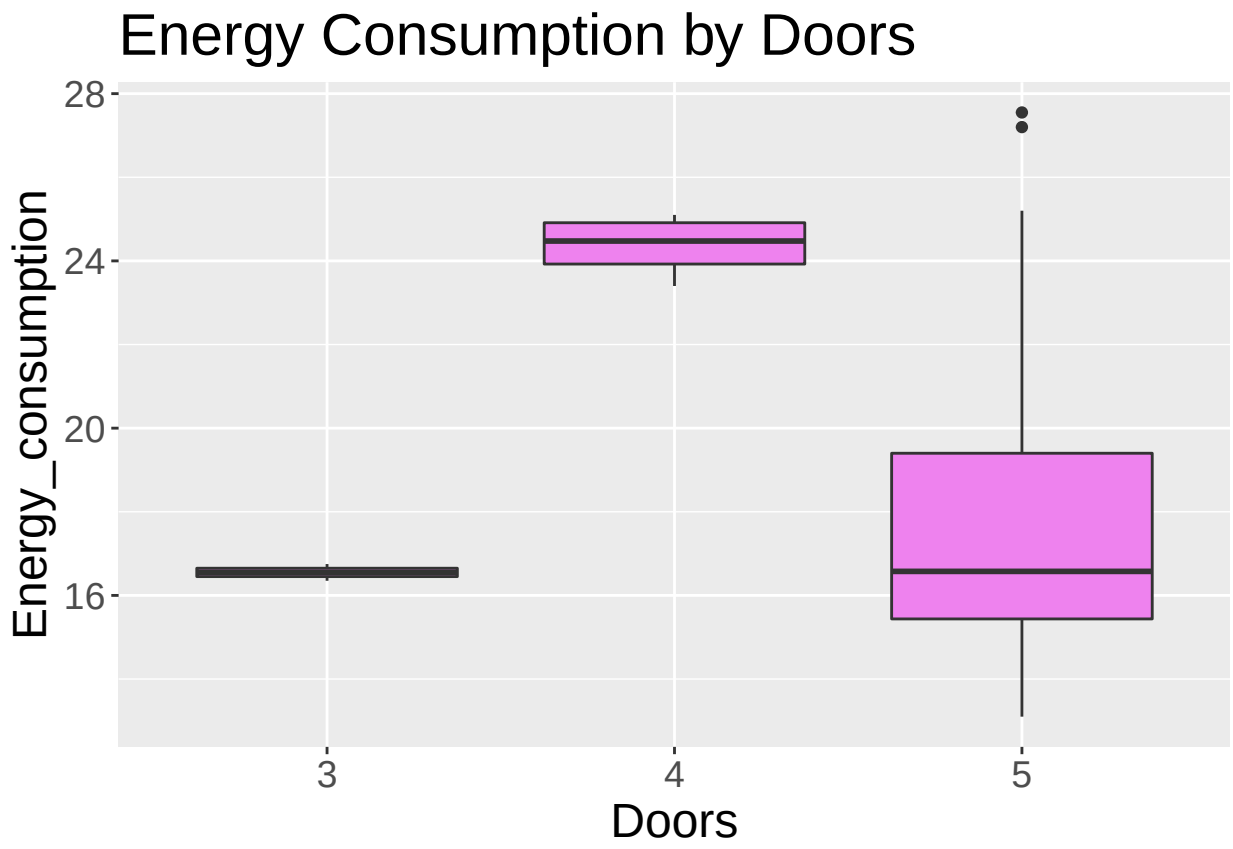
```
##
## $Drive
```


Energy Consumption by Drive
Energy_consumption
28
24
20
16
2WD (front)
2WD (rear)
4WD
Drive

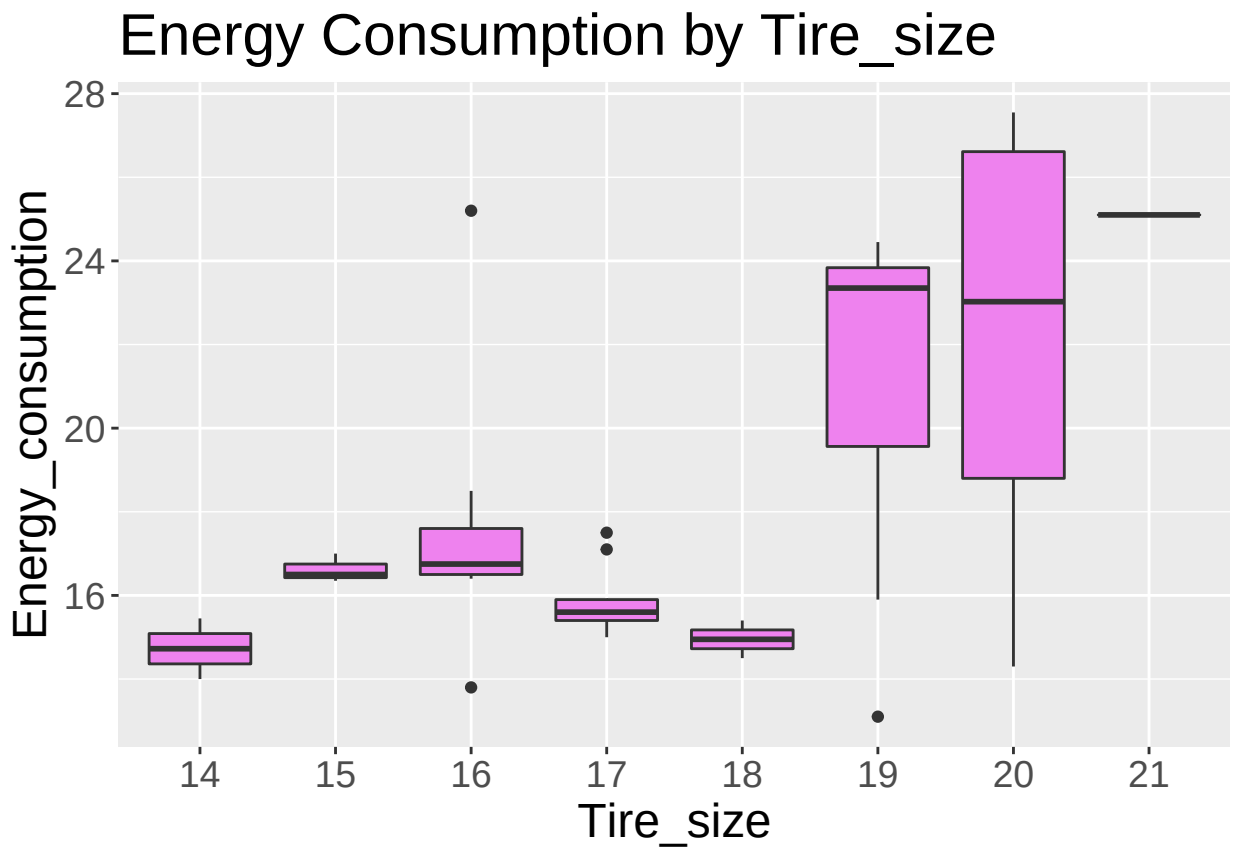
```
##
## $Seats
```


Energy Consumption by Seats
Energy_consumption
28
24
20
16
2
4
5
8
Seats

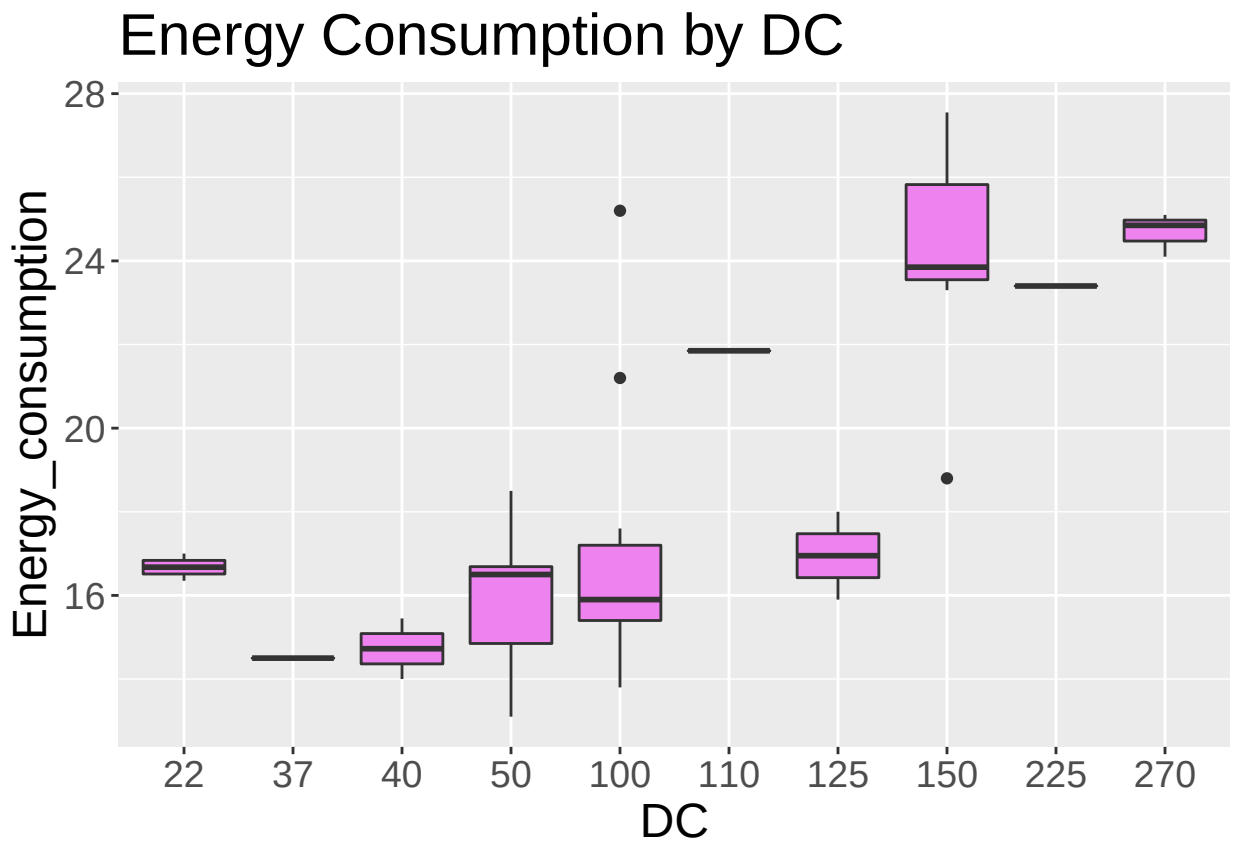
```
##
## $Doors
```


Energy Consumption by Doors
Energy_consumption
28
24
20
16
3
4
5
Doors

```
##
## $Tire_size
```


Energy Consumption by Tire_size
Energy_consumption
28
24
20
16
14
15
16
17
18
19
20
21
Tire_size

```
##
## $DC
```



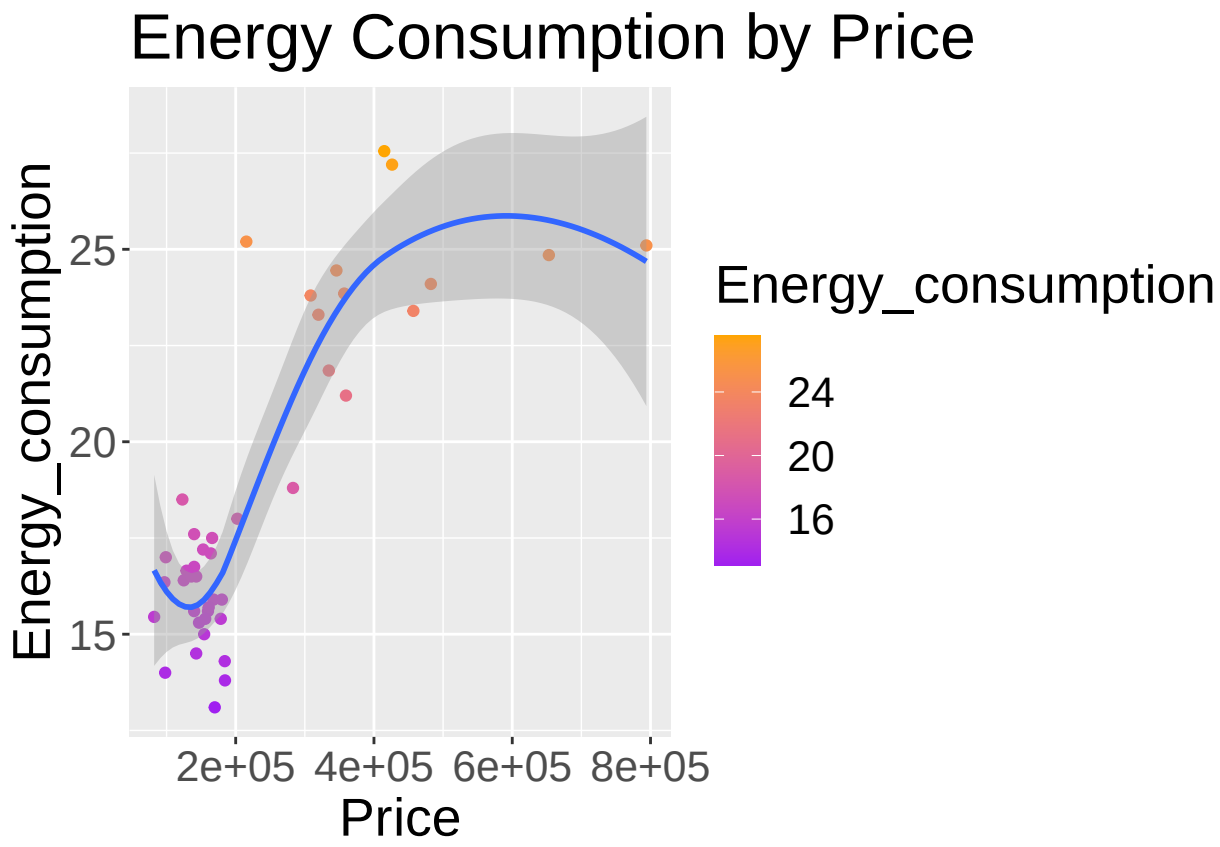

```
rm(labels_boxplot)
rm(plot.boxes)
rm(n)
```

Em seguida, as instruções para a geração dos gráficos de pontos:

```
labels_scatter = list()

for (n in 1:(length(numericas))) {
  labels_scatter[[n]] <- paste("Energy Consumption by",numericas[n])
  }

plot.scatter <- function(X, label){
  ggplot(df1, aes_string(x = X, y = "Energy_consumption")) +
    geom_point(aes_string(colour = "Energy_consumption"), alpha = 2.0) +
    scale_colour_gradient(low = "purple", high = "orange") +
    geom_smooth(method = "loess") +
    ggtitle(label) +
    theme(text = element_text(size = 20))
}

Map(plot.scatter, numericas, labels_scatter)
```

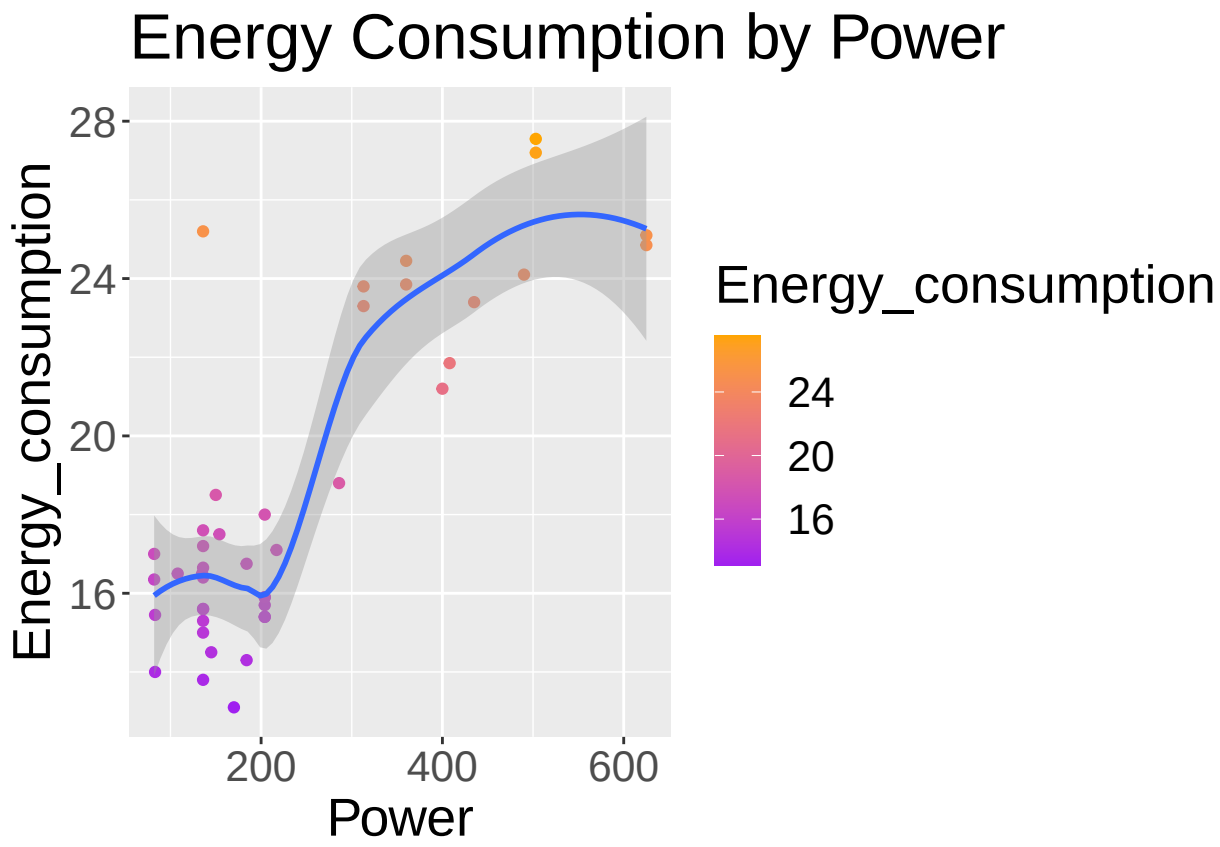
```
## $Price
```

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



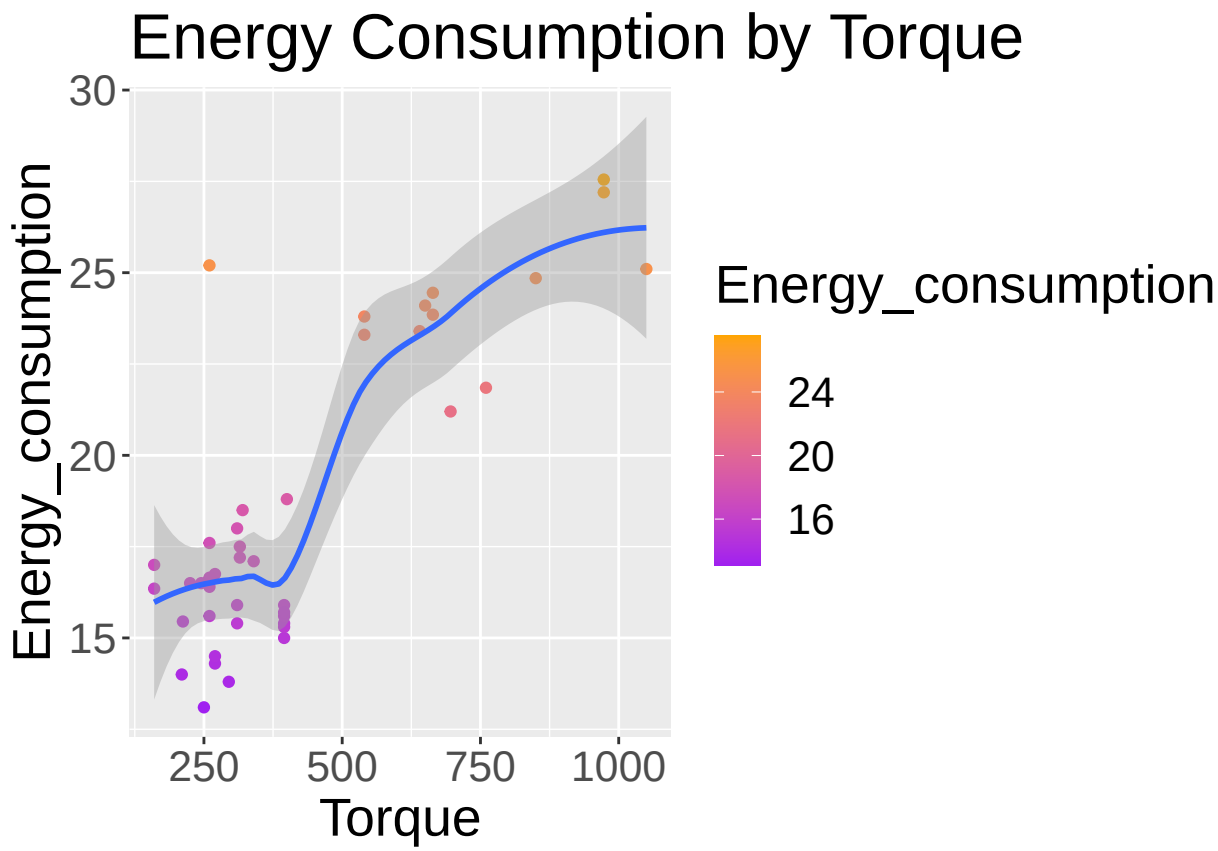

```
##
## $Power
```

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



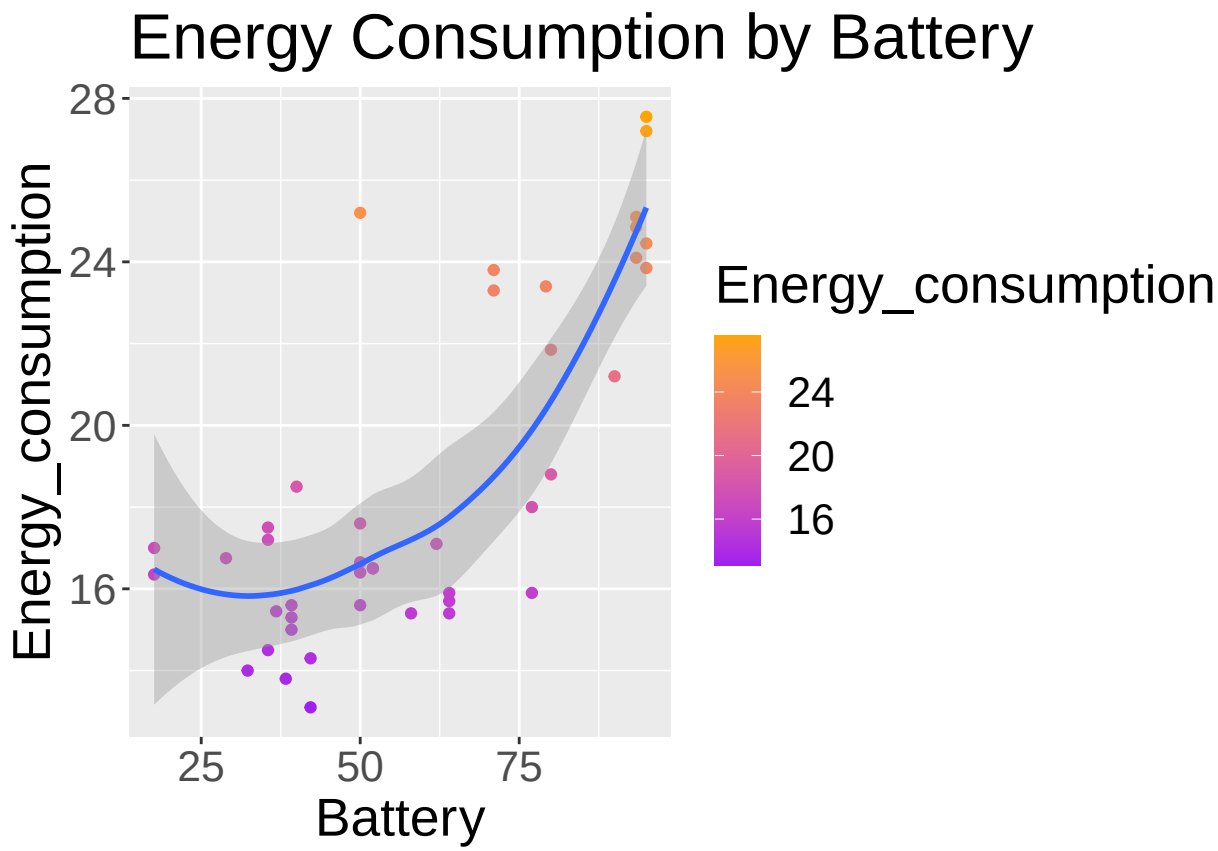

```
##
## $Torque
```

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



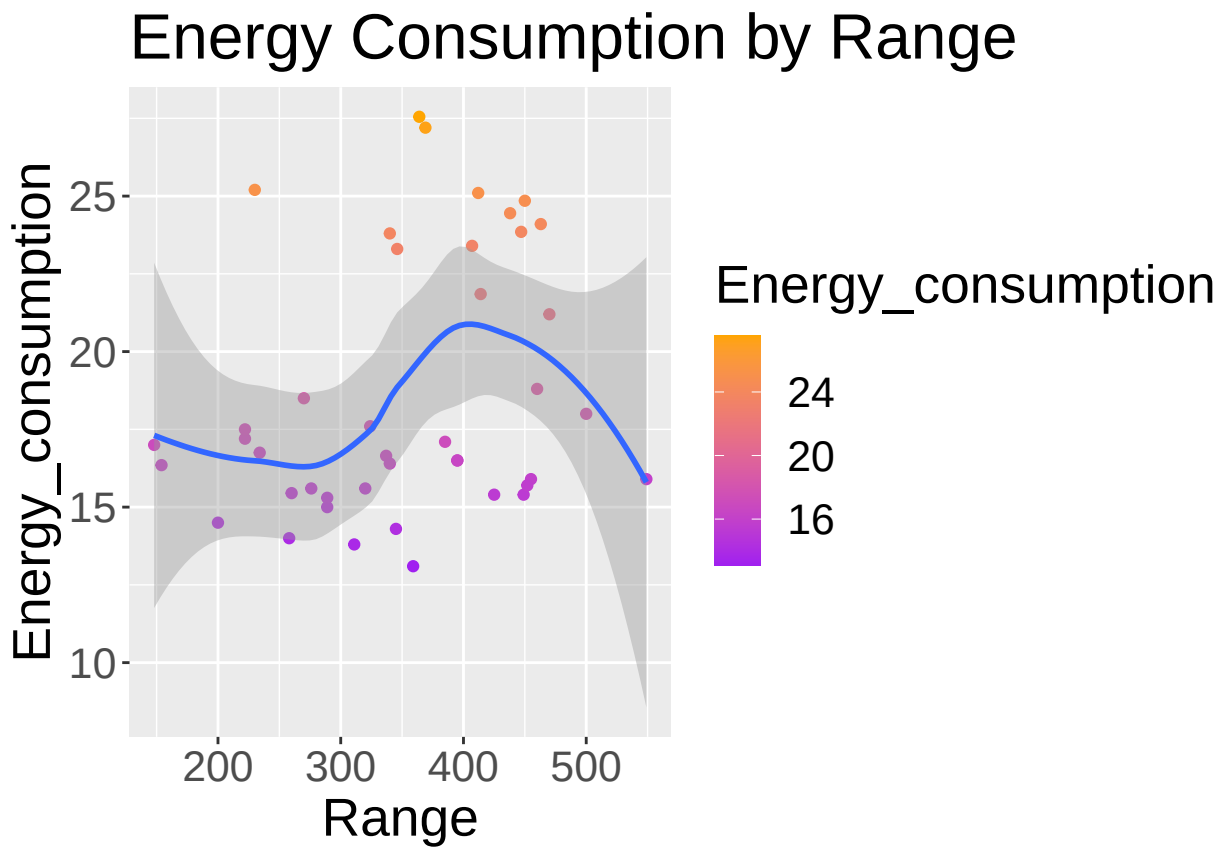

```
##
## $Battery
```

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



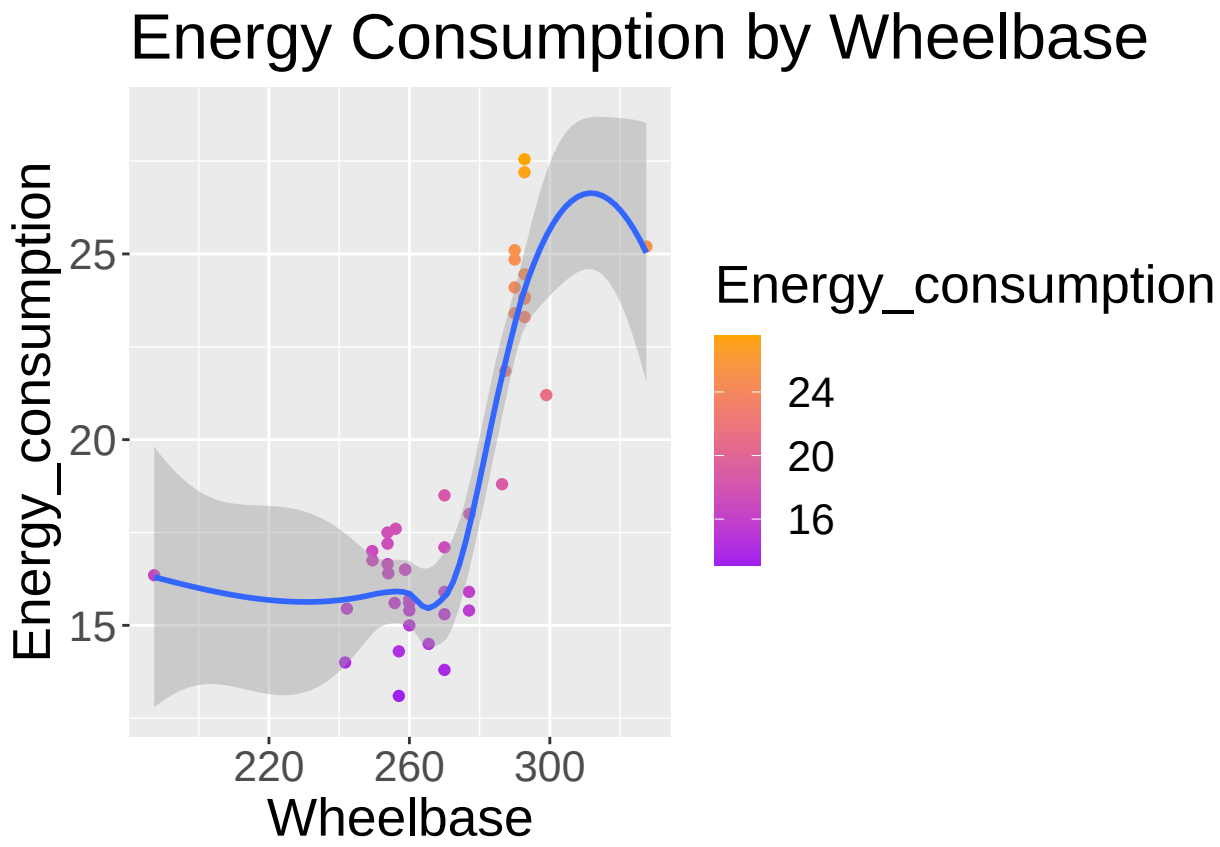

```
##
## $Range
```

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



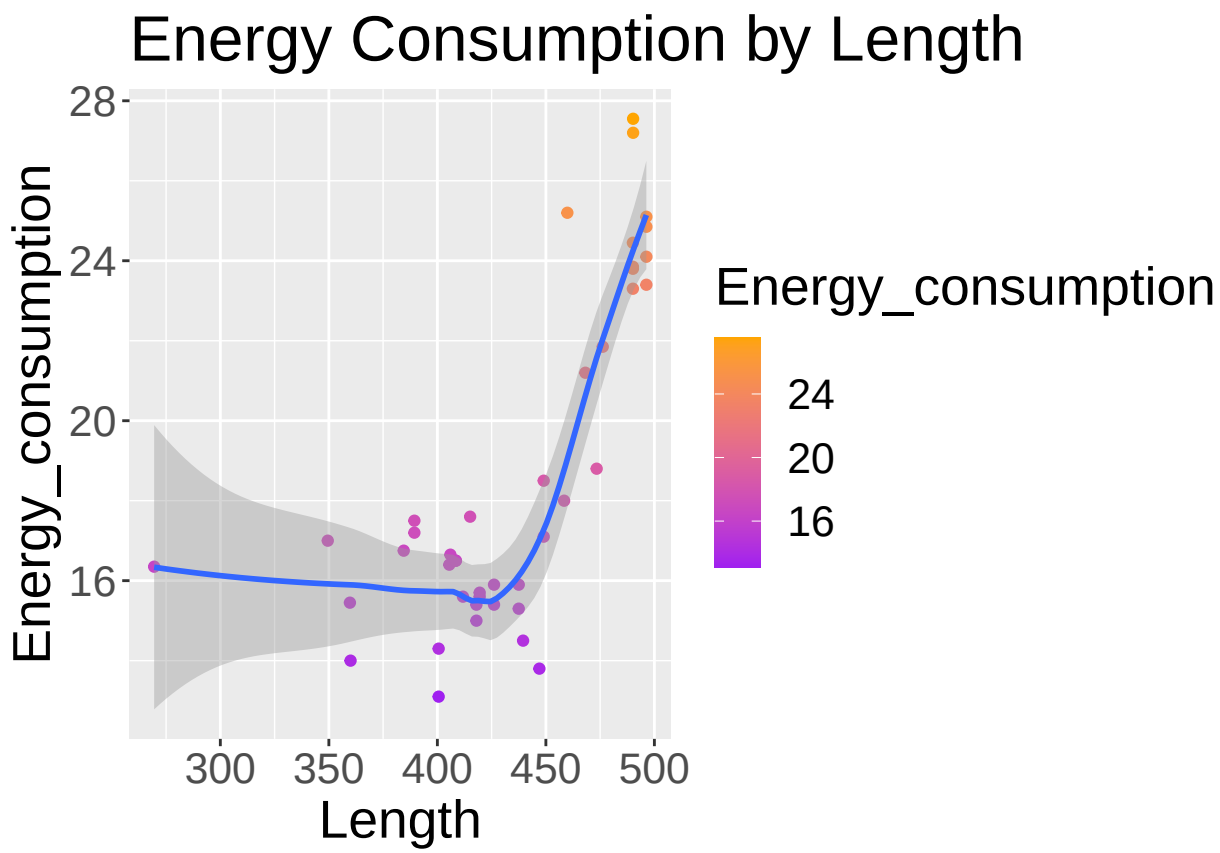

```
##
## $Wheelbase
```

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



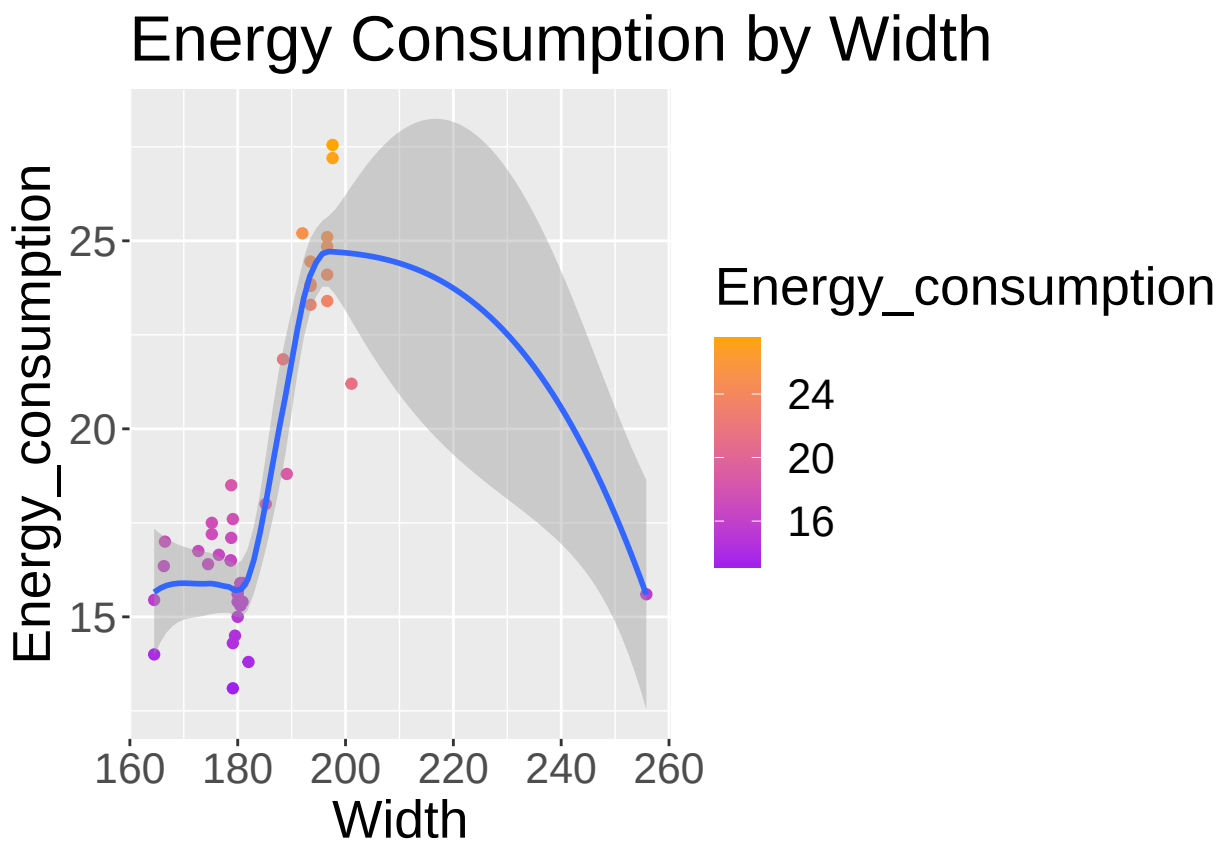

```
##
## $Length
```

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



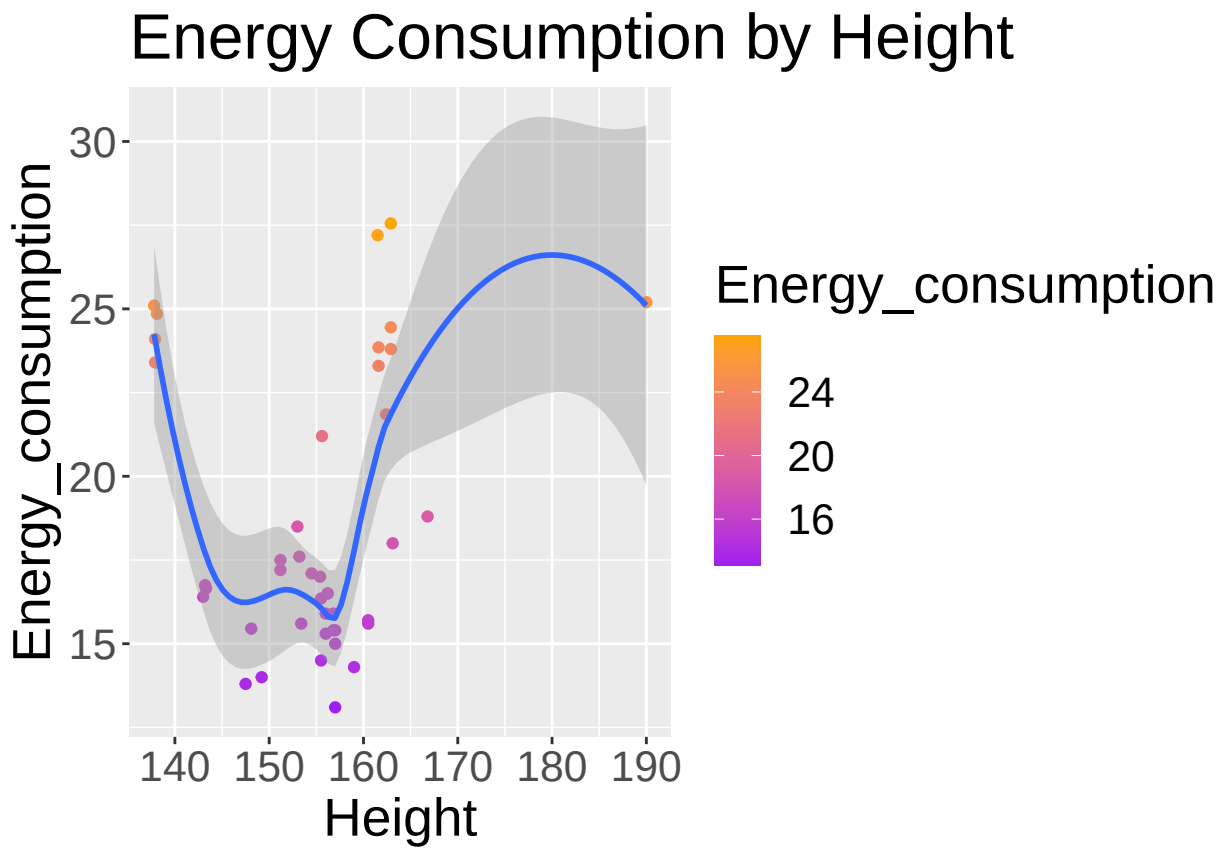

```
##
## $Width
```

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



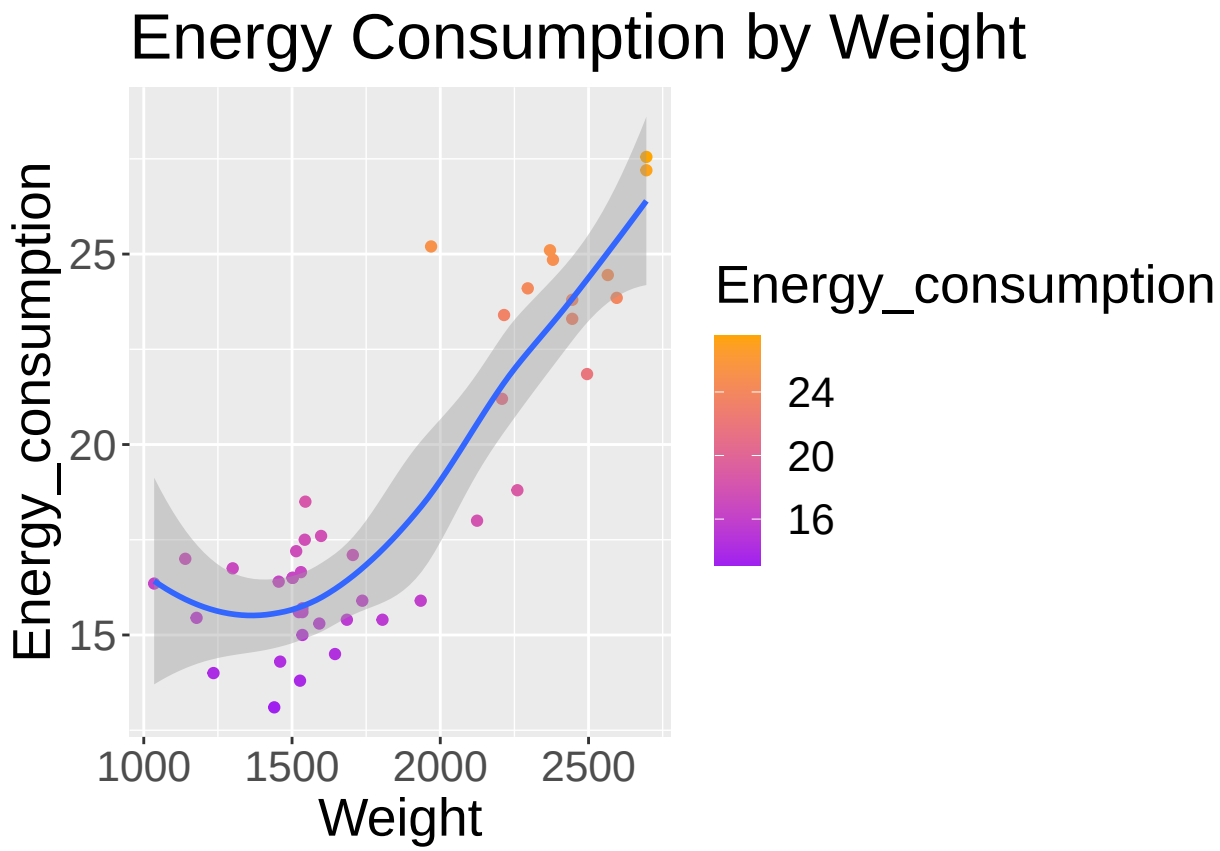

```
##
## $Height
```

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



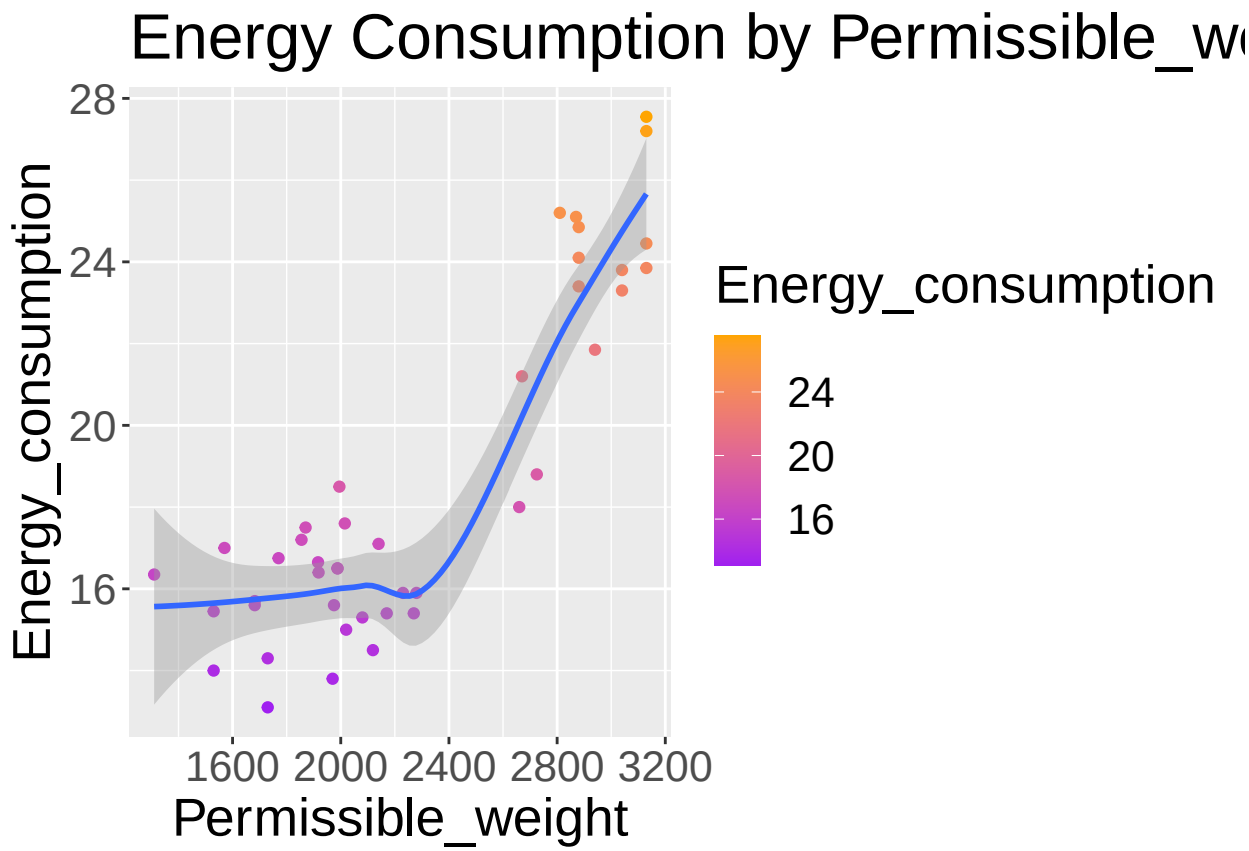

```
##
## $Weight
```

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



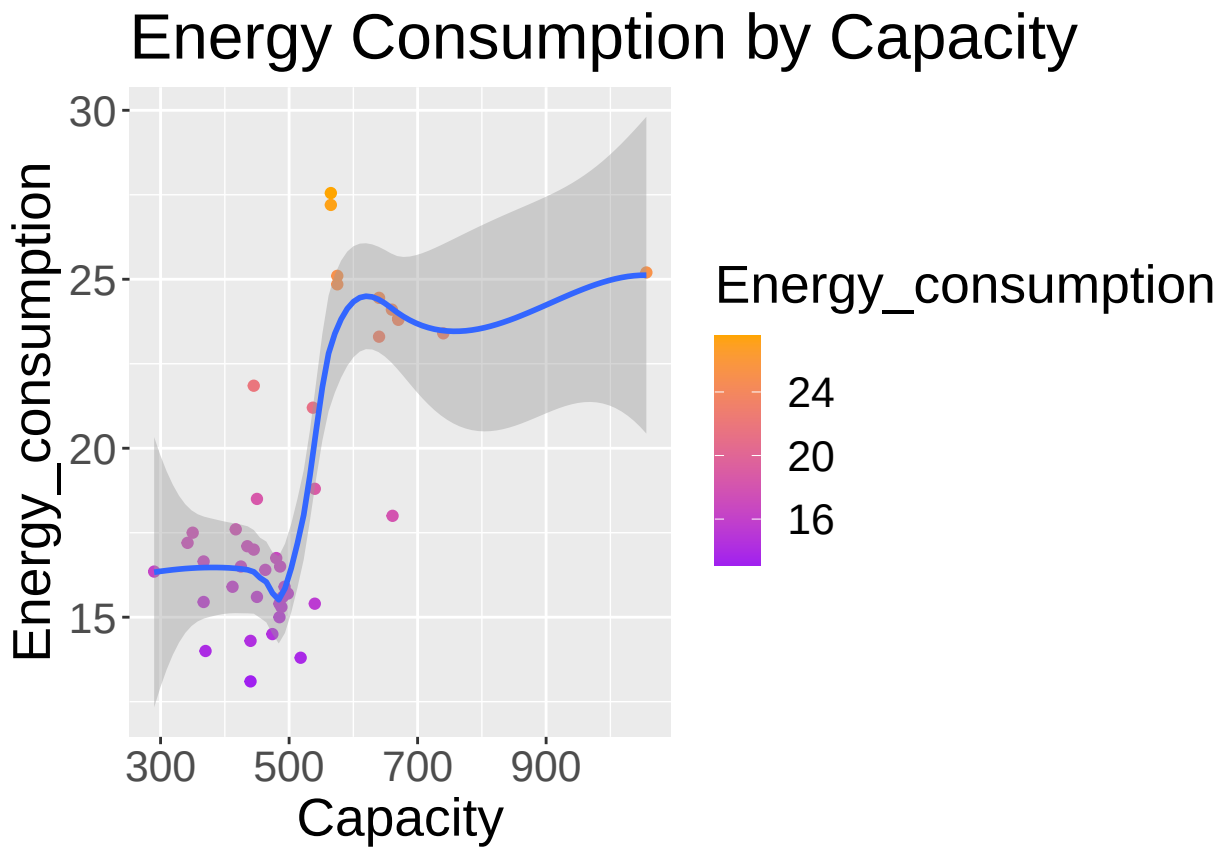

```
##
## $Permissible_weight
```

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



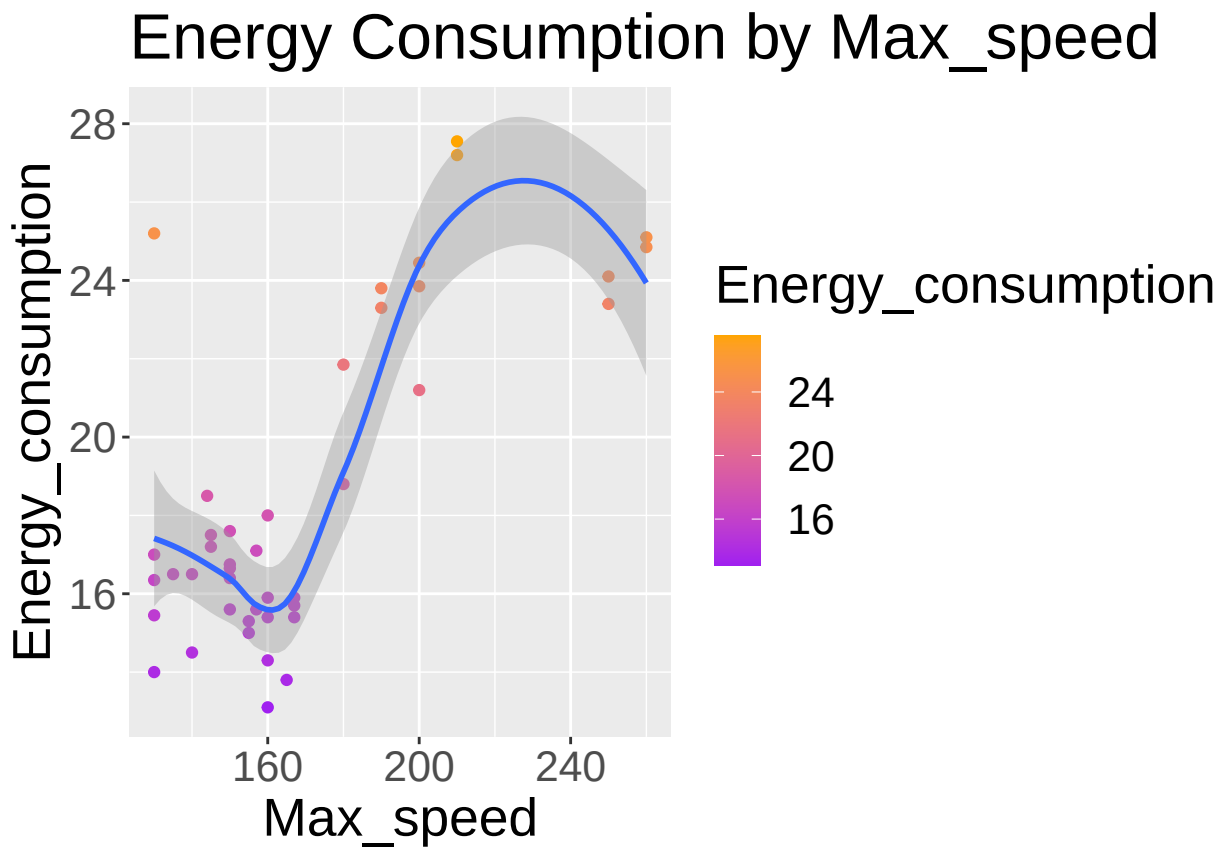

```
##
## $Capacity
```

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



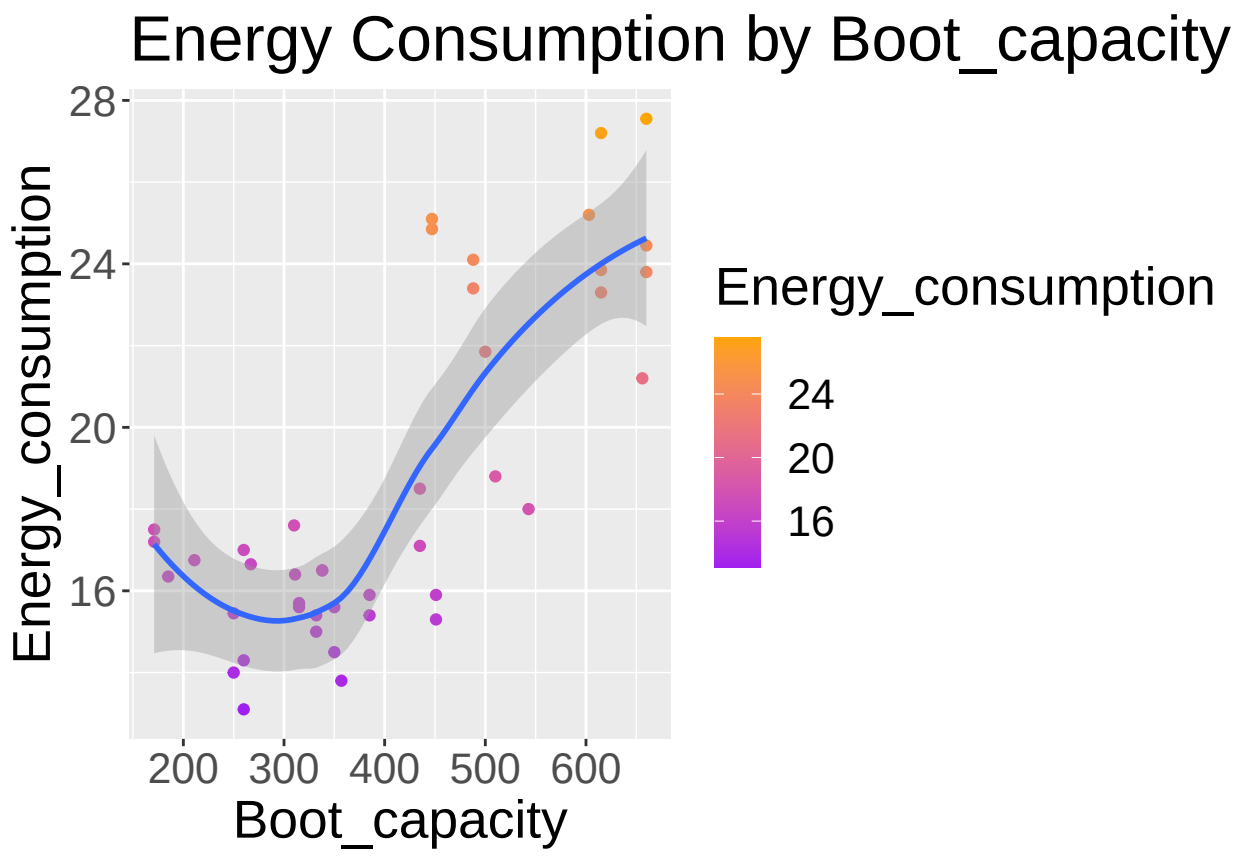

```
##
## $Max_speed
```

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



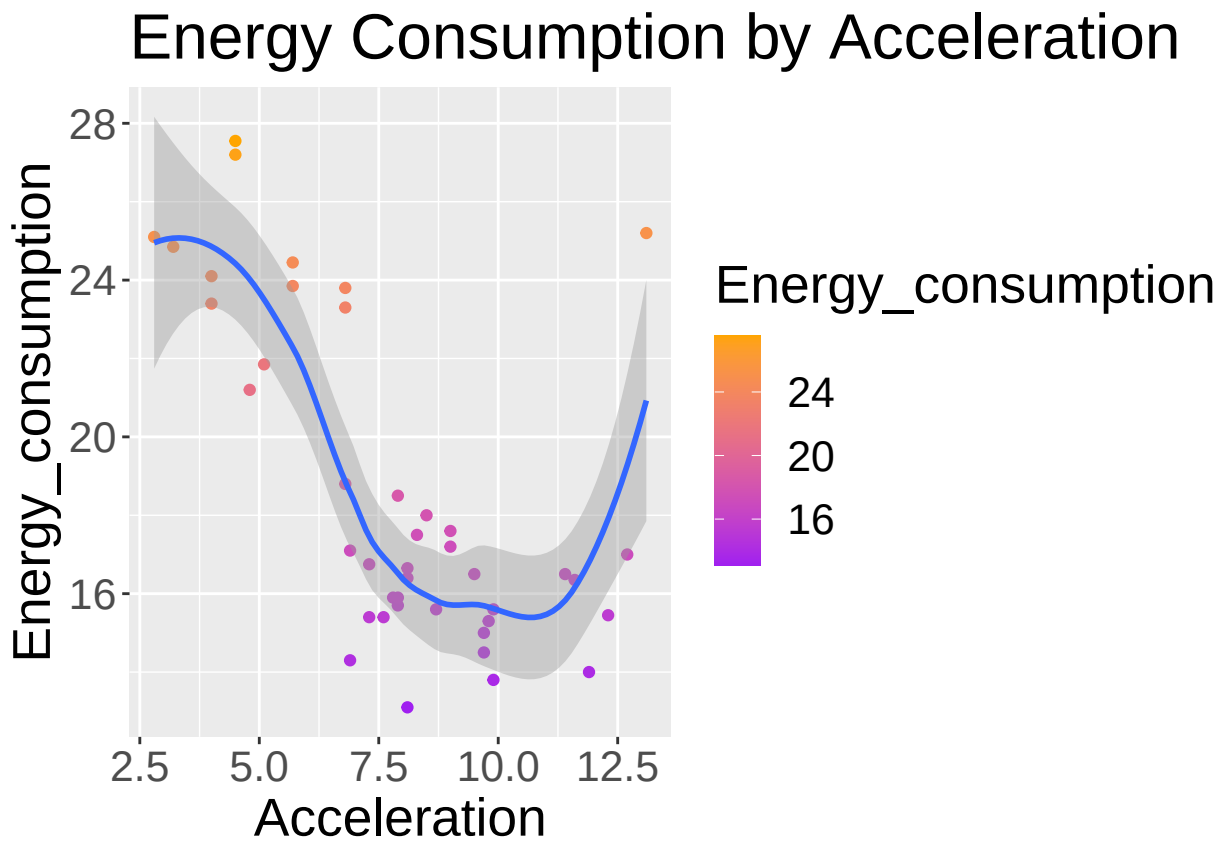

```
##
## $Boot_capacity
```

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




```
##
## $Acceleration
```

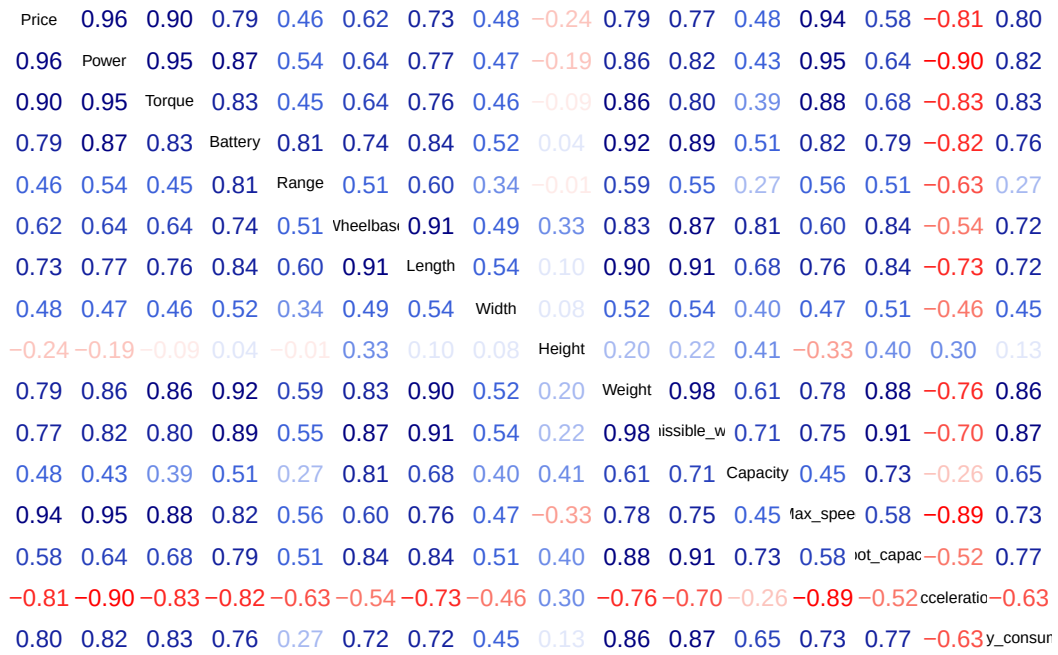
```
## `geom_smooth()` using formula 'y ~ x'
```




```
rm(n)
rm(labels_scatter)
rm(plot.scatter)
rm(numericas)
```

Por fim, vamos verificar a correlação entre as variáveis independentes entre si e com a variável Target:

```
library(corrgram)
corrgram(df1,cor.method = "pearson", panel = panel.cor)
```



Percebe-se que a maioria das variáveis são positivamente correlacionadas com o consumo de energia. Percebe-se ainda que algumas variáveis preditoras tem forte correlação entre si, o que pode representar problemas de multicolinearidade, que afetam negativamente o modelo e terão que ser resolvidos. Um exemplo é o par de variáveis **Wheight**(peso) e **Permissible Weight**(peso permitido), cuja correlação é de 0,98. Aqui é certo que exista colinearidade.

ETAPA 3 - Seleção de variáveis

O objetivo nessa etapa é reduzir o máximo possível o número de variáveis preditoras sem perda significativa de desempenho do modelo preditor. Ao mesmo tempo, buscaremos reduzir a multicolinearidade. Para verificar a colinearidade, extrairemos apenas as variáveis do tipo numérico e usaremos a função **vif** do pacote **car**.

```
df1_numeric <- select(df1,-all_of(fatores))
set.seed(123)
```

Inicialmente, vamos criar um modelo de regressão linear apenas para avaliação das variáveis. O 1º modelo utilizará todas as variáveis numéricas do dataframe.

```
M <- lm(Energy_consumption ~ .,data=df1_numeric)
summary(M)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ ., data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3991 -0.4934  0.1034  0.5112  2.3171
```

```
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.413e+01  1.182e+01   2.041 0.051526 .
## Price               -1.007e-07  6.329e-06  -0.016 0.987427
## Power                7.745e-03  1.061e-02   0.730 0.471884
## Torque              -7.018e-04  3.475e-03  -0.202 0.841509
## Battery              2.131e-01  5.715e-02   3.729 0.000945 ***
## Range               -3.361e-02  6.839e-03  -4.914 4.22e-05 ***
## Wheelbase            1.269e-02  3.178e-02   0.399 0.692994
## Length              -2.432e-02  1.739e-02  -1.399 0.173757
## Width               -1.758e-02  1.609e-02  -1.093 0.284511
## Height              -9.242e-03  5.572e-02  -0.166 0.869539
## Weight              -2.019e-03  4.471e-03  -0.452 0.655340
## Permissible_weight   3.058e-03  3.180e-03   0.962 0.345069
## Capacity             6.226e-03  4.493e-03   1.386 0.177565
## Max_speed           -1.247e-02  3.149e-02  -0.396 0.695361
## Boot_capacity        7.014e-04  4.150e-03   0.169 0.867096
## Acceleration        -7.080e-02  2.528e-01  -0.280 0.781602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.151 on 26 degrees of freedom
## Multiple R-squared:  0.9509, Adjusted R-squared:  0.9225
## F-statistic: 33.55 on 15 and 26 DF,  p-value: 3.325e-13
```

```
vif(M)
```

```
##              Price              Power             Torque            Battery
##          29.448839          77.373089          20.455201          54.841635
##              Range          Wheelbase             Length              Width
##          13.566793          16.231418          22.733347           1.763766
##             Height             Weight Permissible_weight           Capacity
##           8.593031         136.210645          92.574249          11.054942
##          Max_speed      Boot_capacity       Acceleration
##          39.077372          11.453903          13.072575
```

O R2 ajustado foi de **0,9225**. As variáveis mais importantes foram **Battery**(capacidade da bateria) e **Range**(alcance máximo do veículo, sendo portanto um bom avaliador de eficiência energética). Vamos portanto manter essas duas variáveis no modelo. Para avaliar a influência das demais variáveis no modelo, é prudente excluir uma por vez. Assim, teremos uma série de modelos em sequencia. Inicialmente, vamos excluir a variável "Weight", que apresentou o maior valor VIF para medição de colinearidade (valores maiores que 5 são problemáticos).

```
M2 <- lm(Energy_consumption~.-Weight,data=df1_numeric)
summary(M2)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3925 -0.5826  0.0800  0.4717  2.3670
##
## Coefficients:
```

```
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.688e+01  9.983e+00   2.692 0.012039 *
## Price                4.297e-07  6.127e-06   0.070 0.944611
## Power                7.846e-03  1.045e-02   0.751 0.459216
## Torque              -8.721e-04  3.403e-03  -0.256 0.799679
## Battery              2.015e-01  5.035e-02   4.003 0.000439 ***
## Range               -3.254e-02  6.326e-03  -5.145 2.07e-05 ***
## Wheelbase            1.348e-02  3.126e-02   0.431 0.669746
## Length              -2.638e-02  1.654e-02  -1.595 0.122357
## Width               -1.611e-02  1.552e-02  -1.038 0.308505
## Height              -2.718e-02  3.849e-02  -0.706 0.486061
## Permissible_weight   1.803e-03  1.521e-03   1.185 0.246201
## Capacity             7.350e-03  3.685e-03   1.994 0.056291 .
## Max_speed           -1.816e-02  2.843e-02  -0.639 0.528359
## Boot_capacity        1.112e-03  3.989e-03   0.279 0.782605
## Acceleration        -6.305e-02  2.484e-01  -0.254 0.801566
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.134 on 27 degrees of freedom
## Multiple R-squared:  0.9505, Adjusted R-squared:  0.9248
## F-statistic: 37.02 on 14 and 27 DF,  p-value: 5.857e-14
```

```
vif(M2)
```

```
##              Price              Power             Torque            Battery
##          28.434717          77.338734          20.214262          43.863919
##              Range          Wheelbase             Length              Width
##          11.960502          16.181938          21.179295           1.691421
##             Height Permissible_weight           Capacity          Max_speed
##           4.224174          21.818223           7.663851          32.818264
##      Boot_capacity       Acceleration
##          10.904679          13.012314
```

O R2 ajustado foi de **0,9248**. Houve melhora da colinearidade. Excluindo variável **Power** do dataset.

```
M3 <- lm(Energy_consumption~.-Weight -Power,data=df1_numeric)
summary(M3)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2144 -0.6053  0.2321  0.4617  2.3948
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.774e+01  9.839e+00   2.819 0.008750 **
## Price                3.188e-06  4.865e-06   0.655 0.517632
## Torque               9.263e-05  3.126e-03   0.030 0.976572
## Battery              2.117e-01  4.810e-02   4.402 0.000142 ***
## Range               -3.359e-02  6.123e-03  -5.485 7.37e-06 ***
## Wheelbase            1.956e-02  2.995e-02   0.653 0.519107
## Length              -3.088e-02  1.529e-02  -2.019 0.053148 .
```

```
## Width              -1.888e-02  1.496e-02  -1.262 0.217319
## Height             -2.559e-02  3.813e-02  -0.671 0.507574
## Permissible_weight  1.855e-03  1.508e-03   1.231 0.228663
## Capacity            6.974e-03  3.622e-03   1.925 0.064415 .
## Max_speed          -1.190e-02  2.697e-02  -0.441 0.662346
## Boot_capacity       1.417e-03  3.937e-03   0.360 0.721663
## Acceleration       -1.672e-01  2.045e-01  -0.818 0.420522
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.125 on 28 degrees of freedom
## Multiple R-squared:  0.9495, Adjusted R-squared:  0.926
## F-statistic: 40.46 on 13 and 28 DF,  p-value: 1.173e-14
```

```
vif(M3)
```

```
##              Price             Torque            Battery              Range
##          18.212757          17.332764          40.667189          11.383112
##          Wheelbase             Length              Width             Height
##          15.095810          18.397785           1.595905           4.211387
## Permissible_weight           Capacity          Max_speed      Boot_capacity
##          21.772287           7.522486          29.998735          10.791547
##       Acceleration
##           8.957167
```

O R2 ajustado foi de 0,9260. Houve melhora da colinearidade. Excluindo variável **Permissible Weight** do dataset.

```
M4 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight,data=df1_numeric)
summary(M4)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight,
##     data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2691 -0.5829  0.0854  0.5274  2.0442
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.904e+01  9.868e+00   2.943  0.00634 **
## Price          4.580e-06  4.774e-06   0.959  0.34524
## Torque        -8.383e-04  3.060e-03  -0.274  0.78607
## Battery        2.445e-01  4.043e-02   6.047 1.40e-06 ***
## Range         -3.732e-02  5.368e-03  -6.951 1.22e-07 ***
## Wheelbase      1.863e-02  3.021e-02   0.617  0.54225
## Length        -2.532e-02  1.474e-02  -1.718  0.09647 .
## Width         -2.169e-02  1.491e-02  -1.454  0.15664
## Height        -2.002e-02  3.819e-02  -0.524  0.60407
## Capacity       7.597e-03  3.619e-03   2.099  0.04460 *
## Max_speed     -1.615e-02  2.698e-02  -0.598  0.55422
## Boot_capacity  3.151e-03  3.709e-03   0.850  0.40249
## Acceleration  -2.238e-01  2.010e-01  -1.113  0.27478
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.135 on 29 degrees of freedom
## Multiple R-squared:  0.9467, Adjusted R-squared:  0.9247
## F-statistic: 42.94 on 12 and 29 DF,  p-value: 3.573e-15
```

```
vif(M4)
```

```
##         Price        Torque       Battery         Range     Wheelbase
##     17.228123     16.318053     28.232499      8.597425     15.086221
##        Length         Width        Height      Capacity     Max_speed
##     16.796920      1.558779      4.152078      7.375663     29.508229
## Boot_capacity  Acceleration
##      9.409038      8.504400
```

O R2 ajustado foi de 0,9247. Houve melhora da colinearidade. Excluindo variável **Max_speed** do dataset.

```
M5 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed,data=df1_numeric)
summary(M5)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2066 -0.5671  0.0384  0.5221  2.1716
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.451e+01  6.249e+00   3.922 0.000473 ***
## Price          2.979e-06  3.911e-06   0.762 0.452132
## Torque        -1.344e-03  2.910e-03  -0.462 0.647570
## Battery        2.460e-01  3.991e-02   6.164 8.81e-07 ***
## Range         -3.811e-02  5.146e-03  -7.406 2.98e-08 ***
## Wheelbase      2.535e-02  2.774e-02   0.914 0.368019
## Length        -2.719e-02  1.425e-02  -1.908 0.066021 .
## Width         -2.156e-02  1.475e-02  -1.461 0.154309
## Height        -8.439e-03  3.257e-02  -0.259 0.797307
## Capacity       6.430e-03  3.015e-03   2.132 0.041275 *
## Boot_capacity  3.042e-03  3.665e-03   0.830 0.413065
## Acceleration  -1.738e-01  1.809e-01  -0.961 0.344332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.122 on 30 degrees of freedom
## Multiple R-squared:  0.9461, Adjusted R-squared:  0.9263
## F-statistic: 47.84 on 11 and 30 DF,  p-value: 5.949e-16
```

```
vif(M5)
```

```
##         Price        Torque       Battery         Range     Wheelbase
##     11.815220     15.075409     28.114266      8.071707     12.999027
##        Length         Width        Height      Capacity Boot_capacity
##     16.044682      1.558454      3.085312      5.233113      9.386163
##  Acceleration
```

```
##      7.036743
```

O R2 ajustado foi de 0,9263. Houve melhora da colinearidade. Excluindo variável **Torque** do dataset.

```
M6 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque,data=df1_numeric)
summary(M6)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque, data = df1_numeric)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.10283 -0.67413 -0.02417  0.55755  2.21478
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.441e+01  6.166e+00   3.959 0.000409 ***
## Price          1.884e-06  3.070e-06   0.614 0.543882
## Battery        2.391e-01  3.654e-02   6.545 2.63e-07 ***
## Range         -3.692e-02  4.394e-03  -8.402 1.72e-09 ***
## Wheelbase      2.539e-02  2.738e-02   0.927 0.360928
## Length        -2.782e-02  1.400e-02  -1.987 0.055850 .
## Width         -2.026e-02  1.430e-02  -1.417 0.166385
## Height        -1.225e-02  3.110e-02  -0.394 0.696465
## Capacity       7.056e-03  2.658e-03   2.654 0.012423 *
## Boot_capacity  2.733e-03  3.557e-03   0.768 0.448160
## Acceleration  -1.503e-01  1.713e-01  -0.877 0.387258
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.108 on 31 degrees of freedom
## Multiple R-squared:  0.9457, Adjusted R-squared:  0.9282
## F-statistic: 53.97 on 10 and 31 DF,  p-value: < 2.2e-16
```

```
vif(M6)
```

```
##         Price       Battery         Range     Wheelbase        Length
##      7.472469     24.175578      6.037982     12.998891     15.896237
##         Width        Height      Capacity Boot_capacity  Acceleration
##      1.502250      2.887513      4.173292      9.072560      6.477846
```

O R2 ajustado foi de 0,9282. Houve melhora da colinearidade. Excluindo variável **Length** do dataset.

```
M7 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length,data=df1_num
summary(M7)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8080 -0.5701  0.0422  0.7784  1.9802
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.019e+01  6.048e+00   3.338  0.00215 **
## Price          1.939e-06  3.209e-06   0.604  0.54993
## Battery        2.457e-01  3.803e-02   6.461 2.88e-07 ***
## Range         -3.784e-02  4.566e-03  -8.289 1.80e-09 ***
## Wheelbase     -1.444e-02  1.949e-02  -0.741  0.46410
## Width         -2.291e-02  1.488e-02  -1.540  0.13341
## Height         8.537e-03  3.061e-02   0.279  0.78212
## Capacity       7.504e-03  2.768e-03   2.711  0.01069 *
## Boot_capacity -7.120e-06  3.426e-03  -0.002  0.99835
## Acceleration  -2.507e-02  1.665e-01  -0.151  0.88126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.158 on 32 degrees of freedom
## Multiple R-squared:  0.9388, Adjusted R-squared:  0.9215
## F-statistic: 54.51 on 9 and 32 DF,  p-value: < 2.2e-16
```

```
vif(M7)
```

```
##         Price       Battery         Range     Wheelbase         Width
##      7.471874     23.978725      5.969854      6.029797      1.489218
##        Height      Capacity Boot_capacity  Acceleration
##      2.560867      4.143254      7.708939      5.601765
```

O R2 ajustado foi de 0,9215. Houve melhora da colinearidade. Excluindo variável **Price** do dataset.

```
M8 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length -Price,data=
summary(M8)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length - Price, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8623 -0.5884 -0.0287  0.8365  1.9437
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    21.343423   5.680612   3.757 0.000666 ***
## Battery         0.260303   0.029080   8.951 2.41e-10 ***
## Range          -0.039527   0.003584 -11.028 1.32e-12 ***
## Wheelbase      -0.013452   0.019234  -0.699 0.489203
## Width          -0.022742   0.014730  -1.544 0.132153
## Height          0.001462   0.028011   0.052 0.958680
## Capacity        0.008092   0.002566   3.153 0.003430 **
## Boot_capacity  -0.000685   0.003206  -0.214 0.832153
## Acceleration   -0.050128   0.159707  -0.314 0.755592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.147 on 33 degrees of freedom
## Multiple R-squared:  0.9381, Adjusted R-squared:  0.923
```

```
## F-statistic: 62.48 on 8 and 33 DF,  p-value: < 2.2e-16
```

```
vif(M8)
```

```
##       Battery         Range     Wheelbase         Width        Height
##     14.296176      3.751257      5.987176      1.488703      2.186209
##      Capacity Boot_capacity  Acceleration
##      3.631347      6.882693      5.254349
```

O R2 ajustado foi de 0,923. Houve melhora da colinearidade. Excluindo variável **Wheelbase** do dataset.

```
M9 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length -Price
         -Wheelbase,data=df1_numeric)
summary(M9)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length - Price - Wheelbase, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9650 -0.5897  0.0027  0.8218  1.8560
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   18.829847   4.366053   4.313 0.000131 ***
## Battery        0.259309   0.028827   8.995 1.63e-10 ***
## Range         -0.039687   0.003550 -11.180 6.22e-13 ***
## Width         -0.022731   0.014619  -1.555 0.129234
## Height        -0.001153   0.027551  -0.042 0.966873
## Capacity       0.007003   0.002024   3.460 0.001474 **
## Boot_capacity -0.001261   0.003075  -0.410 0.684334
## Acceleration  -0.025916   0.154734  -0.167 0.867978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.138 on 34 degrees of freedom
## Multiple R-squared:  0.9371, Adjusted R-squared:  0.9242
## F-statistic: 72.42 on 7 and 34 DF,  p-value: < 2.2e-16
```

```
vif(M9)
```

```
##       Battery         Range         Width        Height      Capacity
##     14.262077      3.735950      1.488701      2.147258      2.292966
## Boot_capacity  Acceleration
##      6.428422      5.007474
```

O R2 ajustado foi de 0,9242. Houve melhora da colinearidade. Excluindo variável **Acceleration** do dataset.

```
M10 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length -Price
         -Wheelbase -Acceleration,data=df1_numeric)
summary(M10)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length - Price - Wheelbase - Acceleration,
```

```
##     data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9621 -0.5786 -0.0107  0.8396  1.8970
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   18.794403   4.299944   4.371 0.000105 ***
## Battery        0.262233   0.022620  11.593 1.55e-13 ***
## Range         -0.039808   0.003426 -11.619 1.45e-13 ***
## Width         -0.022419   0.014297  -1.568 0.125862
## Height        -0.003254   0.024186  -0.135 0.893759
## Capacity       0.006963   0.001982   3.514 0.001241 **
## Boot_capacity -0.001286   0.003029  -0.424 0.673817
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.122 on 35 degrees of freedom
## Multiple R-squared:  0.9371, Adjusted R-squared:  0.9263
## F-statistic:  86.9 on 6 and 35 DF,  p-value: < 2.2e-16
```

```
vif(M10)
```

```
##       Battery         Range         Width        Height      Capacity
##      9.032077      3.579585      1.464511      1.702119      2.260872
## Boot_capacity
##      6.413748
```

O R2 ajustado foi de 0,9263. Houve melhora da colinearidade. Excluindo variável **Boot_capacity** do dataset.

```
M11 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length -Price
          -Wheelbase -Acceleration -Boot_capacity,data=df1_numeric)
summary(M11)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length - Price - Wheelbase - Acceleration -
##     Boot_capacity, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9902 -0.6250  0.0329  0.8150  1.7754
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.576047   3.841197   5.096 1.12e-05 ***
## Battery      0.255467   0.015865  16.102  < 2e-16 ***
## Range       -0.039330   0.003198 -12.297 1.89e-14 ***
## Width       -0.022910   0.014087  -1.626 0.112614
## Height      -0.008427   0.020652  -0.408 0.685675
## Capacity     0.006611   0.001780   3.715 0.000687 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.109 on 36 degrees of freedom
## Multiple R-squared:  0.9368, Adjusted R-squared:  0.928
## F-statistic: 106.7 on 5 and 36 DF,  p-value: < 2.2e-16
```

```
vif(M11)
```

```
##  Battery    Range    Width   Height Capacity
## 4.546921 3.191990 1.454938 1.269977 1.866493
```

O R2 ajustado foi de 0,928. Houve melhora da colinearidade. Excluindo variável **Width** do dataset.

```
M12 <- lm(Energy_consumption~.-Weight -Power -Permissible_weight -Max_speed -Torque -Length -Price
          -Wheelbase -Acceleration -Boot_capacity -Width,data=df1_numeric)
summary(M12)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Weight - Power - Permissible_weight -
##     Max_speed - Torque - Length - Price - Wheelbase - Acceleration -
##     Boot_capacity - Width, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9865 -0.6411  0.0270  0.7849  1.7895
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.736986   3.096829   5.082 1.10e-05 ***
## Battery      0.246642   0.015236  16.188  < 2e-16 ***
## Range       -0.038644   0.003240 -11.927 3.05e-14 ***
## Height      -0.007867   0.021103  -0.373  0.71144
## Capacity     0.006215   0.001802   3.449  0.00142 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.134 on 37 degrees of freedom
## Multiple R-squared:  0.9321, Adjusted R-squared:  0.9248
## F-statistic:   127 on 4 and 37 DF,  p-value: < 2.2e-16
```

```
vif(M12)
```

```
##  Battery    Range   Height Capacity
## 4.014962 3.136531 1.269624 1.831523
```

O R2 ajustado foi de 0,9248. Houve melhora da colinearidade. Excluindo variável **Height** do dataset.

```
M13 <- lm(Energy_consumption~Battery+Range,data=df1_numeric)
summary(M13)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ Battery + Range, data = df1_numeric)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0408 -0.7108 -0.1345  0.7678  3.9211
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17.144442   0.773499   22.16  < 2e-16 ***
## Battery      0.275598   0.014757   18.68  < 2e-16 ***
## Range       -0.041937   0.003551  -11.81 1.87e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.293 on 39 degrees of freedom
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.9022
## F-statistic: 190.2 on 2 and 39 DF,  p-value: < 2.2e-16
```

```
vif(M13)
```

```
##  Battery    Range
## 2.897661 2.897661
```

O R2 ajustado foi de 0,9022. Houve melhora da colinearidade, mas também alguma perda de precisão do modelo. Como a colinearidade já está resolvida, decidimos continuar com a variável **Heigth** no modelo.

Vamos criar um segundo dataframe, incluindo as variáveis do tipo fator.

```
z <- c(fatores,c("Battery","Range","Height","Energy_consumption"))
df2 <- df1 %>%
  select(all_of(z))
View(df2)
rm(z)
```

Vamos avaliar o modelo com as variáveis do tipo fator incluídas

```
M14 <- lm(Energy_consumption ~ .,data=df2)
summary(M14)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ ., data = df2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0334 -0.1104  0.0000  0.1795  0.8001
##
## Coefficients: (3 not defined because of singularities)
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     13.206499   6.538323   2.020  0.05945 .
## Brakesdisc (front) + drum (rear) -0.515332   2.015278  -0.256  0.80124
## Drive2WD (rear)                   0.651343   0.541732   1.202  0.24571
## Drive4WD                          3.232077   1.077993   2.998  0.00809 **
## Seats4                            0.963611   1.415888   0.681  0.50531
## Seats5                            1.575845   2.363623   0.667  0.51391
## Seats8                            6.660793   3.428905   1.943  0.06882 .
## Doors4                            0.749102   1.128758   0.664  0.51581
## Doors5                           -0.485604   1.163090  -0.418  0.68153
## Tire_size15                       1.562891   0.887733   1.761  0.09629 .
## Tire_size16                       1.934386   1.116530   1.732  0.10129
## Tire_size17                       1.471863   1.276323   1.153  0.26478
## Tire_size18                       1.723978   1.789657   0.963  0.34890
## Tire_size19                      -0.923559   1.855659  -0.498  0.62507
```

```
## Tire_size20                      -0.405598   1.850526  -0.219  0.82912
## Tire_size21                      -1.323960   1.944863  -0.681  0.50520
## DC37                             -5.077253   2.558835  -1.984  0.06362 .
## DC40                                   NA         NA      NA       NA
## DC50                             -0.458800   1.029862  -0.445  0.66158
## DC100                            -2.151612   0.971988  -2.214  0.04082 *
## DC110                            -0.790579   0.677364  -1.167  0.25926
## DC125                             0.960509   2.309011   0.416  0.68263
## DC150                                  NA         NA      NA       NA
## DC225                             0.484590   0.733439   0.661  0.51765
## DC270                                  NA         NA      NA       NA
## Battery                           0.192344   0.030792   6.247 8.86e-06 ***
## Range                            -0.028935   0.005322  -5.437 4.44e-05 ***
## Height                            0.016176   0.043360   0.373  0.71371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5693 on 17 degrees of freedom
## Multiple R-squared:  0.9921, Adjusted R-squared:  0.981
## F-statistic: 89.37 on 24 and 17 DF,  p-value: 1.362e-13
```

O R2 ajustado foi de 0,981. Vamos agora excluir as variáveis fator menos relevantes para o modelo, começando com **Doors**.

```
M15 <- lm(Energy_consumption ~ . -Doors,data=df2)
summary(M15)
```

```
##
## Call:
## lm(formula = Energy_consumption ~ . - Doors, data = df2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0464 -0.1317  0.0000  0.1644  0.7784
##
## Coefficients: (2 not defined because of singularities)
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     14.689826   5.564987   2.640  0.01665 *
## Brakesdisc (front) + drum (rear) -1.806822   1.581927  -1.142  0.26835
## Drive2WD (rear)                   0.652195   0.529158   1.233  0.23361
## Drive4WD                          3.346875   1.018150   3.287  0.00410 **
## Seats4                            0.477527   0.787085   0.607  0.55162
## Seats5                            0.716825   1.136411   0.631  0.53611
## Seats8                            5.832274   2.731442   2.135  0.04674 *
## Tire_size15                       1.593081   0.864252   1.843  0.08181 .
## Tire_size16                       1.968099   1.087765   1.809  0.08714 .
## Tire_size17                       1.488579   1.246093   1.195  0.24775
## Tire_size18                       3.025153   1.260883   2.399  0.02747 *
## Tire_size19                      -1.192616   1.699787  -0.702  0.49188
## Tire_size20                      -0.675740   1.693493  -0.399  0.69457
## Tire_size21                      -1.596274   1.789718  -0.892  0.38421
## DC37                             -7.290991   2.457605  -2.967  0.00826 **
## DC40                                   NA         NA      NA       NA
## DC50                             -1.415753   1.012081  -1.399  0.17885
## DC100                            -3.095747   0.963477  -3.213  0.00482 **
```

```
## DC110                          -1.596226   1.186078  -1.346  0.19507
## DC125                           1.567690   1.751868   0.895  0.38267
## DC150                          -0.798727   1.096436  -0.728  0.47569
## DC225                           0.503441   0.715061   0.704  0.49041
## DC270                                 NA         NA      NA       NA
## Battery                         0.193909   0.029853   6.495 4.15e-06 ***
## Range                          -0.028989   0.005197  -5.578 2.71e-05 ***
## Height                          0.014619   0.042197   0.346  0.73302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5561 on 18 degrees of freedom
## Multiple R-squared:  0.9921, Adjusted R-squared:  0.9819
## F-statistic: 97.73 on 23 and 18 DF,  p-value: 1.463e-14
```

O R2 ajustado foi de 0,9819. Todas as outras variáveis fator apresentam algum nível de importância. Assim, chegou-se ao dataframe final, excluindo a variável **Doors**.

```
df2 <- select(df2, -Doors)
View(df2)
dim(df)
```

```
## [1] 53 25
```

```
dim(df2)
```

```
## [1] 42  9
```

Houve redução significativa do número de variáveis preditoras (de 25 para 9), foram resolvidos os problemas de colinearidade e não houve perda significativa da capacidade de previsão do modelo. Vamos excluir os objetos que não serão mais utilizados.

```
rm(df1,df1_numeric,M,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15,fator,fatores)
```

ETAPA 4 - Treinamento e validação do modelo

Inicialmente, vamos dividir o dataframe em dados de treino e de teste utilizando o pacote **Caret**.

```
set.seed(123)
training.samples <- df2$Energy_consumption %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- df2[training.samples, ]
test.data <- df2[-training.samples, ]
```

O primeiro modelo será o de regressão linear. Vamos usar como métrica principal para comparar modelos de diferentes algoritmos o RMSE (raiz quadrada do erro médio). Consideramos medidas de erro melhores para comparação de eficiência entre modelos que o R2 (coeficiente de determinação). Todos os modelos serão criados com validação cruzada de 5 amostras ($\mathbf{cv = 5}$).

```
m1 <- train(Energy_consumption~.,
            data=train.data,
            method='lm',
            metric='RMSE',
            trControl = trainControl(method = "cv", number = 5)
)
```

```
## Warning in predict.lm(modelFit, newdata): uma predição a partir de um ajuste
## rank-deficient pode ser enganoso
```

```
## Warning in predict.lm(modelFit, newdata): uma predição a partir de um ajuste
## rank-deficient pode ser enganoso

## Warning in predict.lm(modelFit, newdata): uma predição a partir de um ajuste
## rank-deficient pode ser enganoso

## Warning in predict.lm(modelFit, newdata): uma predição a partir de um ajuste
## rank-deficient pode ser enganoso

## Warning in predict.lm(modelFit, newdata): uma predição a partir de um ajuste
## rank-deficient pode ser enganoso
```

Em modelos de regressão linear, esse tipo de aviso costuma ocorrer quando o dataframe tem poucas observações e muitas variáveis, como é o caso em análise. Por isso, vamos excluir algumas variáveis menos importantes antes de gerar o modelo.

```
m1 <- train(Energy_consumption~. -DC -Seats -Tire_size,
            data=train.data,
            method='lm',
            metric='RMSE',
            trControl = trainControl(method = "cv", number = 5)
)
```

A seguir os resultados do modelo de regressão:

```
predictions <- m1 %>% predict(test.data)
data.frame(
  RMSE = RMSE(predictions, test.data$Energy_consumption),
  R2 = R2(predictions, test.data$Energy_consumption)
)
```

```
##       RMSE        R2
## 1 1.096454 0.9554194
```

Agora vamos criar um modelo Random Forest.

```
m2 <- train(Energy_consumption~.,
            data=train.data,
            method='rf',
            metric='RMSE',
            trControl = trainControl(method = "cv", number = 5)
)
```

A seguir os resultados do modelo Random Forest:

```
predictions2 <- m2 %>% predict(test.data)
data.frame(
  RMSE = RMSE(predictions2, test.data$Energy_consumption),
  R2 = R2(predictions2, test.data$Energy_consumption)
)
```

```
##       RMSE       R2
## 1 1.594891 0.895377
```

Observa-se que o entre os dois modelos, o que tem o menor RMSE é o de regressão linear. Por fim, a tabela final com as previsões do modelo de regressão, comparativamente aos modelos observados na amostra de teste.

```
tabelalm <- cbind(pred = predictions,obs = test.data$Energy_consumption)
kable(tabelalm, caption = "Resultados - Modelo de Regressão Linear", format = "pipe")
```

Table 2: Resultados - Modelo de Regressão Linear

pred	obs
27.59953	27.20
14.54250	13.10
22.89190	21.20
15.99402	15.30
17.54713	17.10
16.01880	16.65
23.35584	24.85

```
rm(predictions,predictions2,m1,m2,tabelalm,test.data,train.data,training.samples)
```

ETAPA 5 - Treinamento do Modelo com Auto ML

Agora, para treinamento e avaliação do modelo, vamos usar o pacote Auto ML **H20**.

O H2O requer que os dados estejam no formato específico de dataframe do H2O

```
h2o_frame <- as.h2o(df2)
```

```
##   |                                                                              |
```

```
tabela <- head(h2o_frame)
kable(tabela,caption = "H2O Dataframe",format = "pipe")
```

Table 3: H2O Dataframe

Brakes	Drive	Seats	Tire_size	DC	Battery	Range	Height	Energy_consumption
disc (front + rear)	4WD	5	19	150	95	438	162.9	24.45
disc (front + rear)	4WD	5	19	150	71	340	162.9	23.80
disc (front + rear)	4WD	5	20	150	95	364	162.9	27.55
disc (front + rear)	4WD	5	19	150	71	346	161.6	23.30
disc (front + rear)	4WD	5	19	150	95	447	161.6	23.85
disc (front + rear)	4WD	5	20	150	95	369	161.5	27.20

```
rm(tabela)
```

Utilizando a função de Split do pacote H20 para divisão do dataframe em dados de treino e de teste. Esse split do pacote H20 cria uma lista. O 1º elemento da lista é o dataset de treino e o 2º é o dataset de teste.

```
h2o_frame_split <- h2o.splitFrame(h2o_frame, ratios = 0.80, seed = 123)
head(h2o_frame_split)
```

```
## [[1]]
##                Brakes Drive Seats Tire_size  DC Battery Range Height
## 1 disc (front + rear)   4WD     5        19 150      95   438  162.9
## 2 disc (front + rear)   4WD     5        19 150      71   340  162.9
## 3 disc (front + rear)   4WD     5        20 150      95   364  162.9
## 4 disc (front + rear)   4WD     5        19 150      71   346  161.6
```

```
## 5 disc (front + rear)   4WD     5        19 150      95   447  161.6
## 6 disc (front + rear)   4WD     5        20 150      95   369  161.5
##   Energy_consumption
## 1              24.45
## 2              23.80
## 3              27.55
## 4              23.30
## 5              23.85
## 6              27.20
##
## [36 rows x 9 columns]
##
## [[2]]
##                     Brakes       Drive Seats Tire_size  DC Battery Range
## 1        disc (front + rear)  2WD (rear)     5        19 150    80.0   460
## 2        disc (front + rear) 2WD (front)     5        17 100    50.0   320
## 3        disc (front + rear) 2WD (front)     5        17 100    64.0   449
## 4        disc (front + rear) 2WD (front)     5        16 100    50.0   340
## 5 disc (front) + drum (rear) 2WD (front)     4        14  40    32.3   258
## 6 disc (front) + drum (rear)  2WD (rear)     5        19 125    77.0   549
##   Height Energy_consumption
## 1  166.8               18.8
## 2  153.4               15.6
## 3  157.0               15.4
## 4  143.0               16.4
## 5  149.2               14.0
## 6  156.8               15.9
##
## [6 rows x 9 columns]
```

Na sequência, vamos usar o Auto ML para definir o melhor modelo. Para comparação dos modelos, vamos definir como métrica **RMSE** (raiz quadrada do erro médio). O algoritmo vai avaliar diversos modelos diferentes, alterando automaticamente os parâmetros dos modelos. Como resultado teremos uma tabela ordenada de modelos, em função do RMSE, para escolha do melhor modelo. melhor modelo.

```
modelo_automl <- h2o.automl(y = 'Energy_consumption',
                            training_frame = h2o_frame_split[[1]],
                            leaderboard_frame = h2o_frame_split[[2]],
                            seed = 123,
                            balance_classes = TRUE,
                            max_runtime_secs = 60 * 10,
                            include_algos = c('XGBoost', 'DRF', 'GBM'),
                            sort_metric = "RMSE")
```

```
##   |                                                                      |
## 11:30:53.314: _min_rows param, The dataset size is too small to split for min_rows=100.0: must have a
```

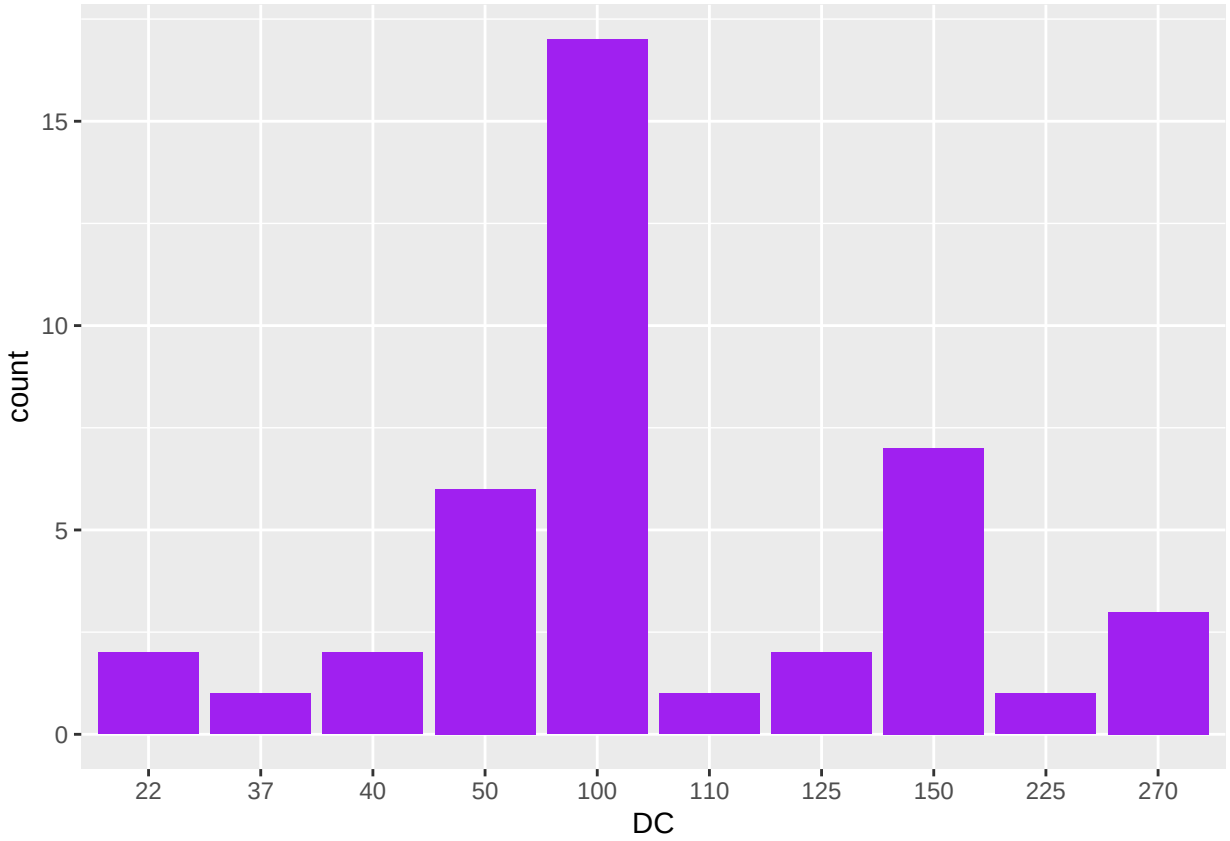
Para avaliação de modelos, o H2O utiliza como padrão a avaliação cruzada com 5 amostras (folds = 5). Tivemos ainda que definir também um parâmetro para limite de tempo de processamento (**max_runtime_secs = 60 * 10**) e um limite no número de algoritmos de Machine Learning para avaliação **(XGBoost, DRF, GBM)**, devido à limitações de capacidade de processamento de dados e memória. Por fim, decidimos fazer um balanceamento de classes, pois o dataframe final tem apenas 42 observações, o que acarreta falta de dados de algumas classes para elaboração do modelo, como pode-se ver no exemplo abaixo:

```
table(df2$DC)
```

```
##
```

```
##  22  37  40  50 100 110 125 150 225 270
##   2   1   2   6  17   1   2   7   1   3
```

```
ggplot(df2, aes(x = DC)) +
  geom_bar(fill = "purple")
```



Extraindo em seguida o leaderboard.

```
leaderboard_automl <- as.data.frame(modelo_automl@leaderboard)
View(leaderboard_automl)
```

Extraindo o líder (modelo com melhor desempenho)

```
lider_automl <- modelo_automl@leader
lider_automl
```

```
## Model Details:
## ==============
##
## H2ORegressionModel: xgboost
## Model ID:  XGBoost_grid_1_AutoML_2_20221023_113051_model_126
## Model Summary:
##   number_of_trees
## 1              63
##
##
## H2ORegressionMetrics: xgboost
## ** Reported on training data. **
##
```

```
## MSE:  0.0740354
## RMSE:  0.2720945
## MAE:  0.1439523
## RMSLE:  0.012121
## Mean Residual Deviance :  0.0740354
##
##
##
## H2ORegressionMetrics: xgboost
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **
##
## MSE:  4.267028
## RMSE:  2.065679
## MAE:  1.304405
## RMSLE:  0.1047962
## Mean Residual Deviance :  4.267028
##
##
## Cross-Validation Metrics Summary:
##                             mean       sd cv_1_valid cv_2_valid cv_3_valid
## mae                     1.272815 0.651457   2.410057   0.837257   1.067713
## mean_residual_deviance 4.033133 4.745959  12.453329   1.722660   1.661577
## mse                     4.033133 4.745959  12.453329   1.722660   1.661577
## r2                      0.765601 0.189724   0.486285   0.901477   0.929705
## residual_deviance       4.033133 4.745959  12.453329   1.722660   1.661577
## rmse                    1.804285 0.985958   3.528927   1.312501   1.289022
## rmsle                   0.093828 0.045384   0.171928   0.082524   0.065388
##                         cv_4_valid cv_5_valid
## mae                       1.181348   0.867698
## mean_residual_deviance    2.953772   1.374328
## mse                       2.953772   1.374328
## r2                        0.654377   0.856160
## residual_deviance         2.953772   1.374328
## rmse                      1.718654   1.172317
## rmsle                     0.089932   0.059368
```

Vamos verificar o desempenho do modelo H2O na previsão dos dados da amostra de teste.

```
predicted <- h2o.predict(lider_automl,h2o_frame_split[[2]]) %>%
  as.data.frame()
```

```
##   |                                                                              |
```

```
df_train <- as.data.frame(h2o_frame_split[[2]])

tabela <- data.frame(
  pred = predicted$predict,
  obs = df_train$Energy_consumption
)
rm(predicted,df_train)
kable(tabela,caption = "Resultados - Modelo AutoML",format = "pipe")
```

Table 4: Resultados - Modelo AutoML

pred	obs
18.74474	18.8
16.15974	15.6
15.86652	15.4
17.16700	16.4
15.45597	14.0
16.53959	15.9

```
R2(tabela$pred,tabela$obs)
```

```
## [1] 0.9591575
```

```
RMSE(tabela$pred,tabela$obs)
```

```
## [1] 0.7800859
```

O desempenho desse modelo foi superior, em termos da métrica que foi adotada (RMSE) em relação ao modelo de regressão linear, criado manualmente na etapa anterior (0,68 X 1,09).

Por fim, vamos verificar o nível de importância de cada variável no modelo líder gerado pelo Auto ML (**análise SHAP**). A função abaixo, do **H2O** traz os valores de contribuição de cada variável, seguindo o modelo **SHAP**, analisando cada observação da amostra de teste:

```
var_contrib <- predict_contributions.H2OModel(lider_automl, h2o_frame_split[[2]])
```

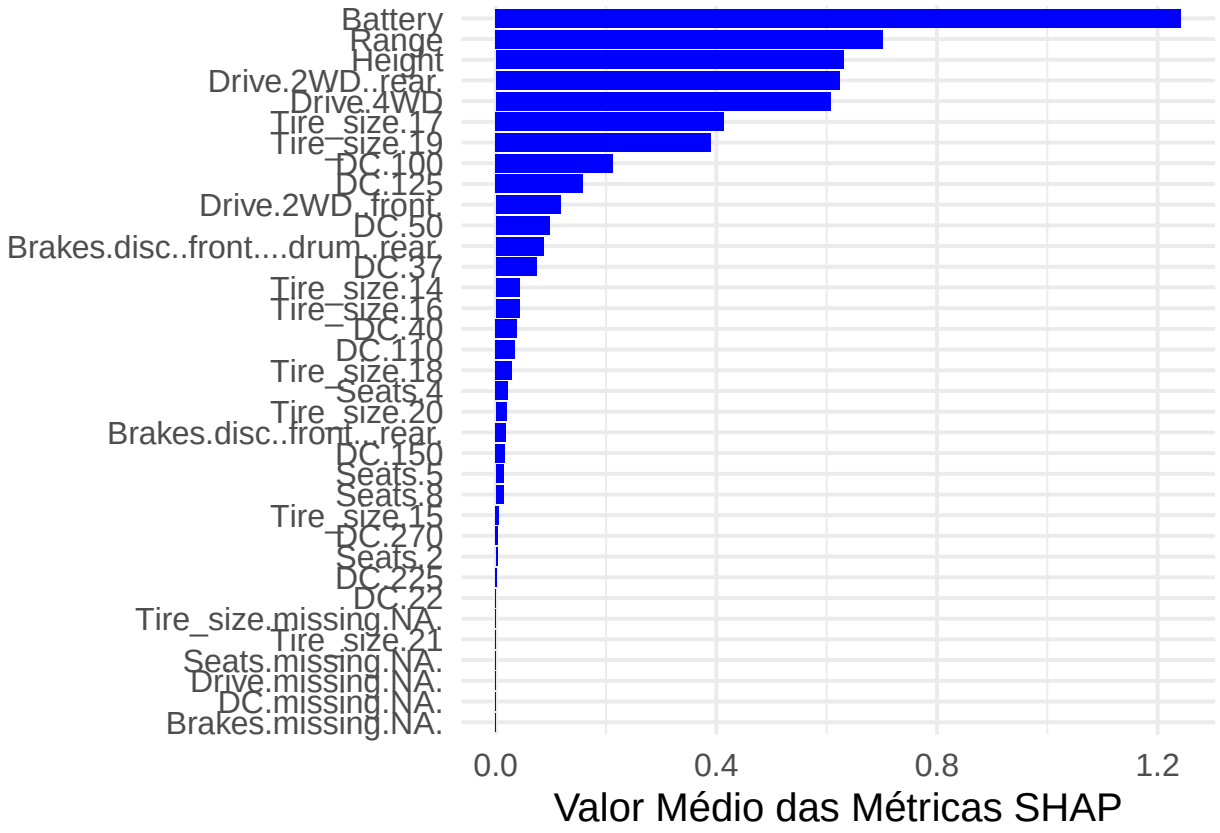
```
##   |                                                                              |
```

Para visualizar o resultado final,vamos preparar um dataframe com os as métricas necessárias

```
df_var_contrib <- var_contrib %>%
  as.data.frame() %>%
  select(-BiasTerm) %>%
  gather(feature, shap_value) %>%
  group_by(feature) %>%
  mutate(shap_importance = mean(abs(shap_value)), shap_force = mean(shap_value)) %>%
  ungroup()

View(df_var_contrib)
```

Agora o gráfico com a importância de cada variável para prever a variável alvo (**Energy__consumption**):

```
df_var_contrib %>%
  select(feature, shap_importance) %>%
  distinct() %>%
  ggplot(aes(x = reorder(feature,shap_importance), y = shap_importance)) +
  geom_col(fill = 'blue') +
  coord_flip() +
  xlab(NULL) +
  ylab("Valor Médio das Métricas SHAP") +
  theme_minimal(base_size = 15)
```



Desligando o H2O

```
h2o.shutdown()
```

```
## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/ (Y/N)?
```