Regressão Linear BoxCox

Rogério Coelho

2025-02-11

Exercício **Auto MPG - uso de BOXCOX**

Leitura do arquivo: auto-mpg.data

- Atribuição dos nomes das colunas e ajustes nos tipos de variáveis (categóricas e numéricas)
- A variável horsepower possui informações ausentes “?”

```
df_auto <- read.table("auto-mpg.data", quote="\"", comment.char="")
# Renomeando as colunas
colnames(df_auto) <- c("mpg", "cylinders", "displacement", "horsepower",
                       "weight", "acceleration", "model_year", "origin", "car_name")
# Definindo as colunas numéricas e categóricas
colunas_num <- c("mpg", "displacement", "horsepower", "weight", "acceleration")
colunas_categoricas <- c("cylinders", "model_year", "origin", "car_name")
# Convertendo colunas categóricas
df_auto[colunas_categoricas] <- lapply(df_auto[colunas_categoricas], as.factor)
# Convertendo colunas numéricas
df_auto[colunas_num] <- lapply(df_auto[colunas_num], as.numeric)
# separa marca do modelo
library(tidyverse)
df_auto <- df_auto %>%
  separate(car_name, into = c("marca", "modelo"), sep = " ", extra = "merge") %>%
  mutate(marca = as.factor(marca), modelo = as.factor(modelo))
summary(df_auto)
```

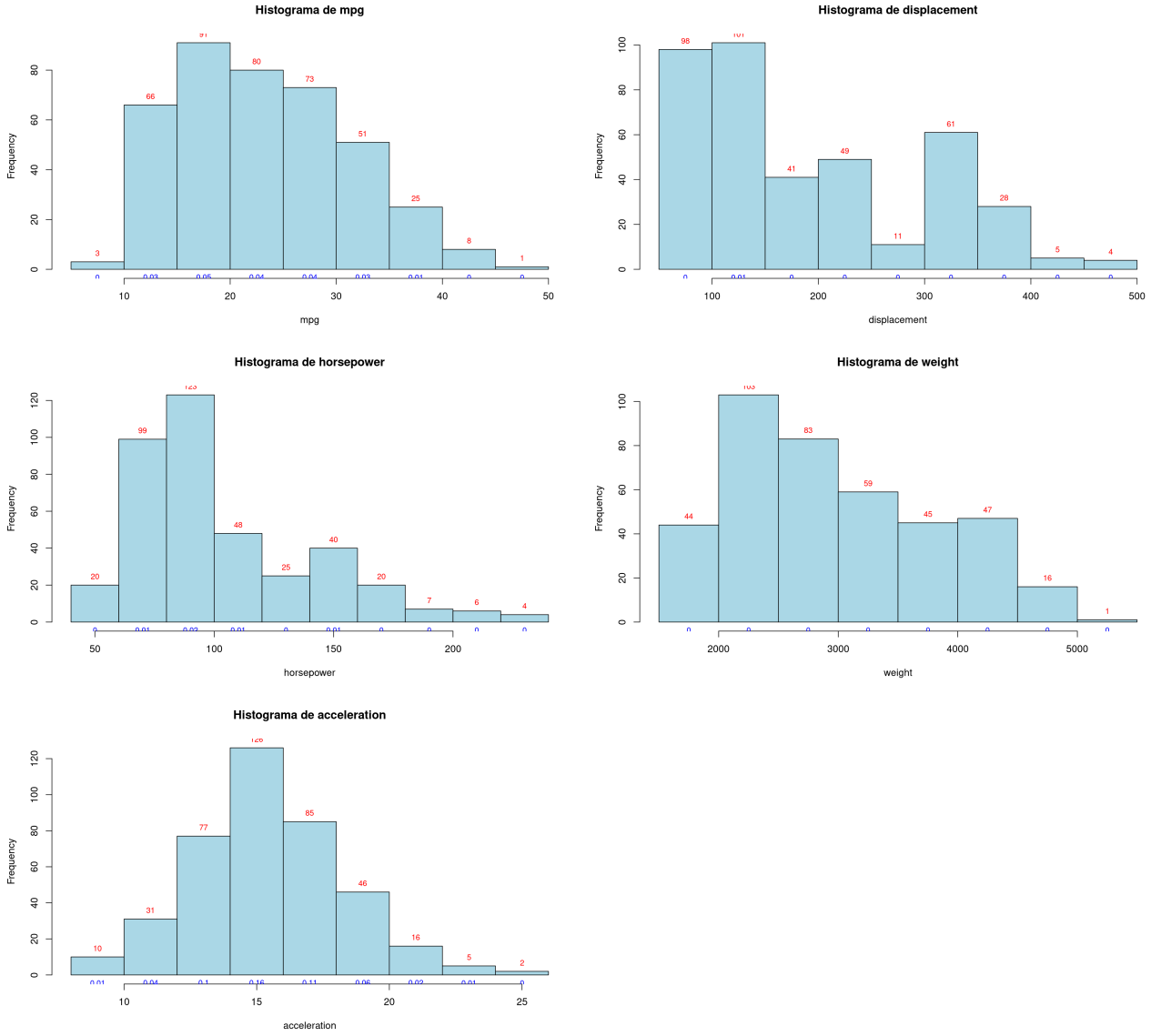
```
##       mpg        cylinders  displacement     horsepower        weight
##  Min.   : 9.00   3:  4     Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.50   4:204     1st Qu.:104.2   1st Qu.: 75.0   1st Qu.:2224
##  Median :23.00   5:  3     Median :148.5   Median : 93.5   Median :2804
##  Mean   :23.51   6: 84     Mean   :193.4   Mean   :104.5   Mean   :2970
##  3rd Qu.:29.00   8:103     3rd Qu.:262.0   3rd Qu.:126.0   3rd Qu.:3608
##  Max.   :46.60             Max.   :455.0   Max.   :230.0   Max.   :5140
##                                            NA's   :6
##   acceleration     model_year  origin        marca          modelo
##  Min.   : 8.00   73     : 40   1:249   ford     : 51   pinto   :  6
##  1st Qu.:13.82   78     : 36   2: 70   chevrolet: 43   corolla :  5
##  Median :15.50   76     : 34   3: 79   plymouth : 31   matador :  5
##  Mean   :15.57   82     : 31           amc      : 28   maverick:  5
##  3rd Qu.:17.18   75     : 30           dodge    : 28   rabbit  :  5
##  Max.   :24.80   70     : 29           toyota   : 25   (Other) :370
##                  (Other):198           (Other)  :192   NA's    :  2
```

```
colunas_categoricas <- c("cylinders", "model_year", "origin", "marca", "modelo")
```

Análise Univariada

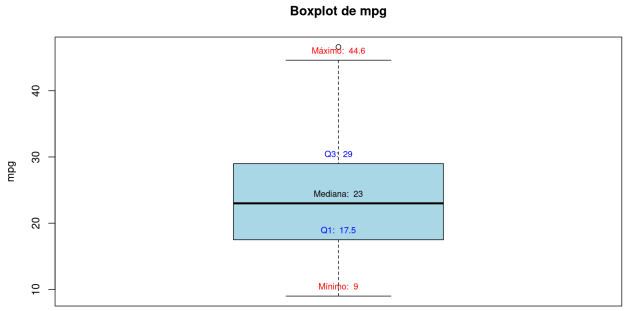
Histogramas e boxplots

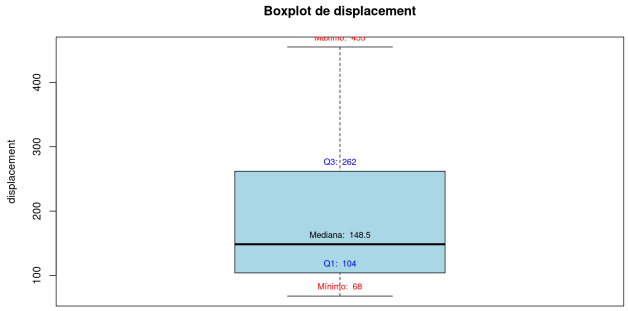
```
# Usando lapply para gerar os histogramas
#lapply(colunas_num, function(col) {
#  hist(df_auto[[col]], main = paste("Histograma de", col), xlab = col, col = "lightblue", border = "black")
#})
# Usando lapply para gerar os histogramas com informações no topo das colunas
invisible(lapply(colunas_num, function(col) {
  # Gerando o histograma e armazenando o objeto
  h <- hist(df_auto[[col]], main = paste("Histograma de", col),
            xlab = col, col = "lightblue", border = "black", plot = FALSE)

  # Desenhando o histograma
  hist(df_auto[[col]], main = paste("Histograma de", col),
       xlab = col, col = "lightblue", border = "black")

  # Adicionando as informações no topo das colunas
  text(h$mids, h$counts, labels = h$counts, pos = 3, cex = 0.8, col = "red")  # Mostra os counts
  text(h$mids, h$density, labels = round(h$density, 2), pos = 1, cex = 0.8, col = "blue")  # Mostra a densidade
}))
```

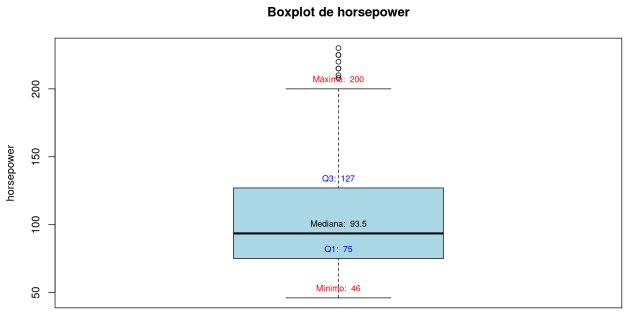


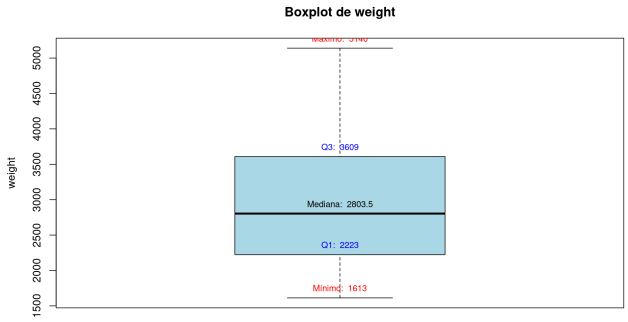
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

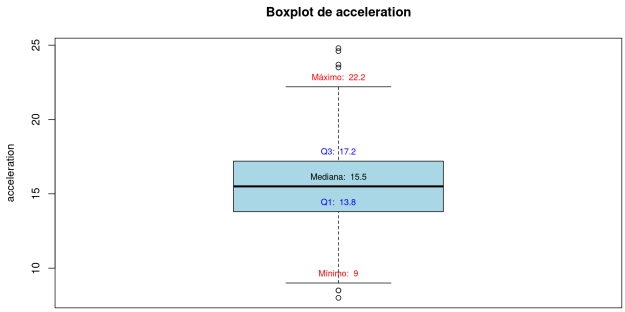
```
# Usando lapply para gerar os boxplots
#lapply(colunas_num, function(col) {
#  boxplot(df_auto[[col]], main = paste("Boxplot de", col), xlab = col, col = "lightblue", border = "black")
#})
# Usando lapply para gerar os boxplots com informações de quartis e outros dados
invisible(lapply(colunas_num, function(col) {
  # Calculando as estatísticas do boxplot
  box_stats <- boxplot(df_auto[[col]], plot = FALSE)

  # Gerando o boxplot
  boxplot(df_auto[[col]], main = paste("Boxplot de", col),
          ylab = col, col = "lightblue", border = "black")

  # Adicionando as informações no gráfico
  # Exibindo o mínimo, Q1, mediana, Q3, e máximo
  text(1, box_stats$stats[1], labels = paste("Mínimo: ", round(box_stats$stats[1], 2)), pos = 3, cex = 0.8, col =
"red")
  text(1, box_stats$stats[2], labels = paste("Q1: ", round(box_stats$stats[2], 2)), pos = 3, cex = 0.8, col = "bl
ue")
  text(1, box_stats$stats[3], labels = paste("Mediana: ", round(box_stats$stats[3], 2)), pos = 3, cex = 0.8, col
= "black")
  text(1, box_stats$stats[4], labels = paste("Q3: ", round(box_stats$stats[4], 2)), pos = 3, cex = 0.8, col = "bl
ue")
  text(1, box_stats$stats[5], labels = paste("Máximo: ", round(box_stats$stats[5], 2)), pos = 3, cex = 0.8, col =
"red")
}))
```











Resumo estatístico das colunas numéricas

```
# Usando um loop for para gerar os resumos das colunas numéricas com seus nomes
for (col in colunas_num) {
  cat("\nResumo da coluna:", col, "\n")  # Exibe o nome da coluna
  print(summary(df_auto[[col]]))  # Exibe o resumo estatístico da coluna
}
```

```
##
## Resumo da coluna: mpg
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   17.50   23.00   23.51   29.00   46.60
##
## Resumo da coluna: displacement
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    68.0   104.2   148.5   193.4   262.0   455.0
##
## Resumo da coluna: horsepower
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    46.0    75.0    93.5   104.5   126.0   230.0       6
##
## Resumo da coluna: weight
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1613    2224    2804    2970    3608    5140
##
## Resumo da coluna: acceleration
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   13.82   15.50   15.57   17.18   24.80
```

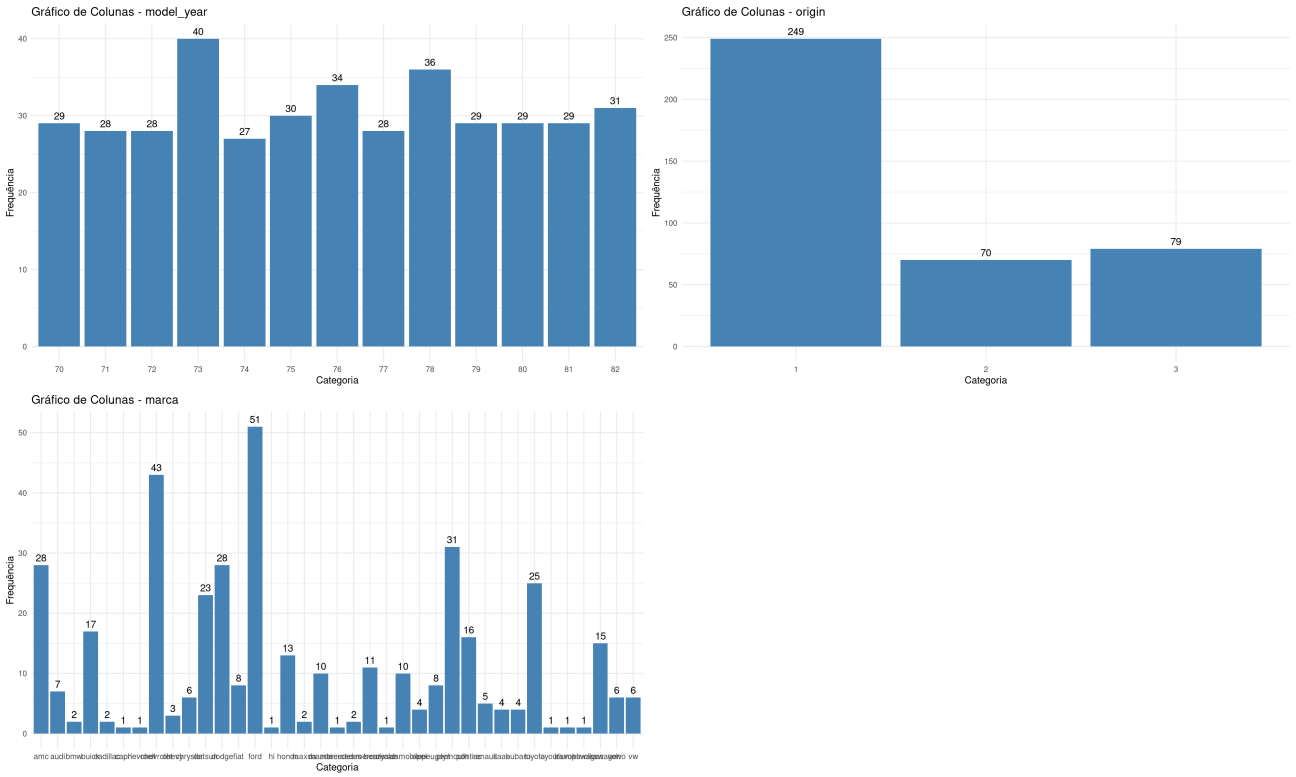
Resumo

- mpg: distribuição simétrica

- displacement: simetria positiva
- horsepower: simetria positiva e presença de outliers
- weight: leve assimetria positiva
- acceleration: distribuição simétrica

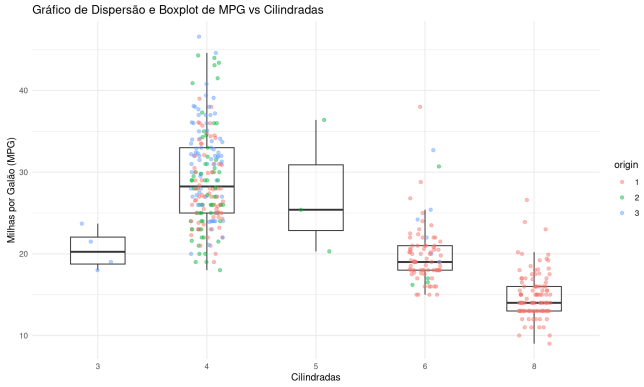
Variáveis Categoricas

```
# Loop para calcular frequências e criar gráficos para cada variável categórica
colunas_categoricas <- c("model_year", "origin", "marca")
for (var in colunas_categoricas) {
  # Calcular as frequências
  frequencias <- as.data.frame(table(df_auto[[var]]))
  colnames(frequencias) <- c("categoria", "frequencia")

  # Criar o gráfico de colunas
  plot <- ggplot(frequencias, aes(x = categoria, y = frequencia)) +
    geom_col(fill = "steelblue") +
    geom_text(aes(label = frequencia), vjust = -0.5) +  # Adiciona rótulos acima das barras
    labs(
      title = paste("Gráfico de Colunas -", var),  # Nome da variável no título
      x = "Categoria",
      y = "Frequência"
    ) +
    theme_minimal()

  # Exibir o gráfico
  print(plot)
}
```



Entendendo mpg x cilindradas

```
# Converter a coluna 'cylinders' para numérica
df_auto$cylinders <- as.numeric(as.character(df_auto$cylinders))

# Criar o gráfico de dispersão com jitter e boxplots
ggplot(df_auto, aes(x = as.factor(cylinders), y = mpg)) +  # Usar as.factor para boxplot
  geom_boxplot(width = 0.5, alpha = 0.5, outlier.shape = NA) +  # Boxplot sem outliers
  geom_jitter(aes(color = origin),width = 0.15, height = 0, alpha = 0.45) +  # Pontos de dispersão
  labs(
    title = "Gráfico de Dispersão e Boxplot de MPG vs Cilindradas",
    x = "Cilindradas",
    y = "Milhas por Galão (MPG)"
  ) +
  theme_minimal()
```




```
df_auto$cylinders <- as.factor(df_auto$cylinders)
```

Análise do gráfico

- Veículos com 4 cilindros são a maioria na amostra
  - maior amplitude em comparação com os demais
- 3 e 5 cilindros não possuem um quantidade significativa na amostra
- 6 e 8 cilindros possuem um consumo maior de combustível, sendo que 8 são os

Dispersão de MPG vs weight

- É possível perceber a presença de heterocedasticidade

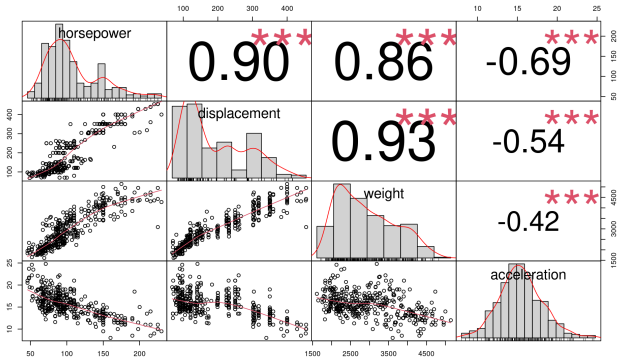
```
library(ggplot2)
library(rlang)

colunas <- colunas_num[-1]

# Loop para criar gráficos de dispersão
for (col in colunas) {
  # Criar o gráfico
  plot <- ggplot(df_auto, aes(x = .data[[col]], y = mpg, color = origin)) +
    geom_point() +  # Adiciona os pontos ao gráfico
    labs(
      title = paste("Gráfico de Dispersão de MPG vs", col),
      x = col,
      y = "Milhas por Galão (MPG)"
    ) +
    theme_minimal()

  # Exibir o gráfico
  print(plot)  # Corrige o erro chamando explicitamente o objeto do gráfico
}
```

Heterocedasticidade

- Presença de heterocedasticidade em todos os gráficos de dispersão
- Os gráficos apresentam uma leve curvatura na dispersão

Verificando a multicolinearidade

```
library(PerformanceAnalytics)
analise_correl <- select(df_auto, horsepower, displacement, weight, acceleration)
chart.Correlation((analise_correl),histogram=TRUE)
```



Resulado

- Existe uma correlação muito forte entre as variáveis horsepower, cylinders, displacement e weight
- Optei por realizar uma regressão usando step wise para saber qual variável seria a melhor escolha para o modelo.
- Em relação a explicabilidade seria mais simples explicar a relação entre mpg e peso. Quanto mais pesado mais combustível o veículo consome. Potência também seria uma alternativa para explicação. Já displacement e cilindros nem tanto.

Dada a multicolinearidade existente uma opção seria utilizar PCA para construção do modelo, no entando não seria nada fácil explicar o comportamento.

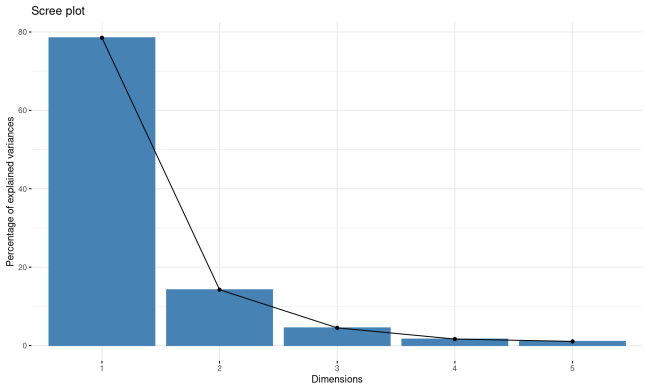
Análise PCA

- A Análise de Componentes Principais (PCA) tem como principal objetivo reduzir a dimensionalidade de um conjunto de dados, preservando ao máximo a variabilidade presente nas variáveis originais. Isso é feito transformando as variáveis correlacionadas em um novo conjunto de variáveis não correlacionadas, chamadas de componentes principais (PCs). Caso o modelo de regressão não fique bom, podemos testar um outro modelo usando os componentes principais calculados.

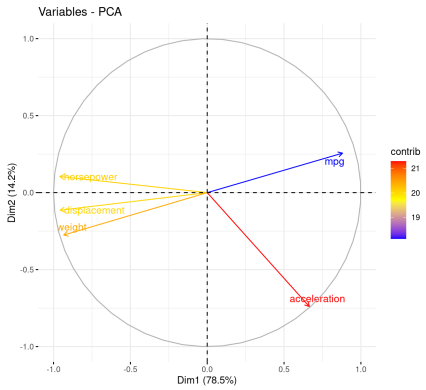
```
library(ggplot2)
library(FactoMineR)
library(factoextra)

# Selecionar as variáveis de interesse
df_pca <- df_auto[, c("horsepower", "mpg", "displacement", "weight", "acceleration")]

# Remover linhas com valores ausentes (se necessário)
df_pca <- na.omit(df_pca)

# Padronizar as variáveis (opcional, mas recomendado)
df_pca <- scale(df_pca)

# Aplicar a PCA
pca_result <- prcomp(df_pca, scale = TRUE)

# Resumo dos resultados
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5
## Standard deviation     1.9816 0.8438 0.47500 0.28788 0.22966
## Proportion of Variance 0.7853 0.1424 0.04512 0.01658 0.01055
## Cumulative Proportion  0.7853 0.9277 0.97288 0.98945 1.00000
```

```
# Gráfico de scree plot
fviz_eig(pca_result)
```




```
# Gráfico das variáveis
fviz_pca_var(pca_result, col.var = "contrib", gradient.cols = c("blue", "yellow", "red"), repel = TRUE)
```



Transformação da Y usando boxcox

```
library(car)
df_auto_modelo <- df_auto %>%
  select(mpg, weight, acceleration, model_year)
# Remover linhas com valores ausentes (se necessário)
df_auto_modelo <- na.omit(df_auto_modelo)
df_auto_modelo$weight <- df_auto_modelo$weight / 1000

# Calcula o lambda ótimo usando powerTransform (modelo sem covariáveis: mpg ~ 1)
lambda <- powerTransform(mpg ~ 1, data = df_auto_modelo)
print(summary(lambda))  # Verifique o resultado e o valor de lambda
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.1974           0      -0.0907       0.4854
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                            LRT df    pval
## LR test, lambda = (0) 1.809026  1 0.17863
##
## Likelihood ratio test that no transformation is needed
##                            LRT df       pval
## LR test, lambda = (1) 29.35683  1 6.0204e-08
```

```
# Extrai o lambda ótimo
lambda_opt <- lambda$lambda
cat("Lambda ótimo:", lambda_opt, "\n")
```

```
## Lambda ótimo: 0.1973548
```

```
# Aplica a transformação Box-Cox na variável mpg usando bcPower
df_auto_modelo$mpg_boxcox <- bcPower(df_auto_modelo$mpg, lambda_opt)
```

Criação do modelo

- df_auto_modelo dataframe com as variáveis
  - Devido a multicolinearidade existente entre as variáveis
  - Decidi por: weight, acceleration, model_year (Mais fácil explicar influência do peso no comportamento do modelo)
- df_auto_modelo_dummies: dataframe com as dummies

```
## Define o dataframe e variáveis que farão parte do modelo
df_auto_modelo <- df_auto_modelo %>%
  select(mpg_boxcox, weight, acceleration, model_year)

# Criar variáveis dummy para as variáveis categóricas
#reordenando para remover a dummy de maior volume
df_auto_modelo$model_year <- factor(df_auto_modelo$model_year, levels = c("73", "70", "71", "72", "74", "75", "7
6", "77", "78", "79", "80", "81", "82"))

# Criar variáveis dummy removendo uma categoria de referência "73"
df_auto_modelo_dummies <- df_auto_modelo %>%
  model.matrix(~ . , data = .) %>%
  as.data.frame()
# Remove a coluna intercept
df_auto_modelo_dummies <- df_auto_modelo_dummies %>% select(-`(Intercept)`)
```

observação

Poderia usar as variáveis categóricas diretamente sem criar as dummies, pois a step (chamada para Step Wise faz isso automaticamente) Para escolher a casela de referência: - df_auto_modelo$model_year < -relevel(df_auto_modelomodel_year, ref = “73”) # Define “73” como referência

Modelo nulo

```
#modelo nulo
lm_mpg_nulo <- lm(mpg_boxcox ~ 1, data=df_auto_modelo_dummies) # modelo nulo
summary(lm_mpg_nulo)
```

```
##
## Call:
## lm(formula = mpg_boxcox ~ 1, data = df_auto_modelo_dummies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.54809 -0.45176  0.04222  0.48261  1.44900
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.29873    0.03133   137.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.625 on 397 degrees of freedom
```

Modelo completo

```
lm_mpg_full <- lm(mpg_boxcox ~ weight + acceleration + model_year70 + model_year71 + model_year72 +
                    model_year74 + model_year75 + model_year76 + model_year77 + model_year78 +
                    model_year79 + model_year80 + model_year81 + model_year82,
                    data = df_auto_modelo_dummies) # modelo com todas as variáveis
summary(lm_mpg_full)
```

```
##
## Call:
## lm(formula = mpg_boxcox ~ weight + acceleration + model_year70 +
##     model_year71 + model_year72 + model_year74 + model_year75 +
##     model_year76 + model_year77 + model_year78 + model_year79 +
##     model_year80 + model_year81 + model_year82, data = df_auto_modelo_dummies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70385 -0.13739  0.01174  0.14606  0.69736
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.486629   0.101068  54.286  < 2e-16 ***
## weight       -0.536962   0.014758 -36.385  < 2e-16 ***
## acceleration  0.007426   0.004518   1.644 0.101086
## model_year70  0.030500   0.053045   0.575 0.565640
## model_year71  0.135483   0.053537   2.531 0.011785 *
## model_year72  0.046502   0.053324   0.872 0.383715
## model_year74  0.193994   0.054586   3.554 0.000427 ***
## model_year75  0.168823   0.052661   3.206 0.001460 **
## model_year76  0.216178   0.050883   4.249 2.70e-05 ***
## model_year77  0.320349   0.053589   5.978 5.18e-09 ***
## model_year78  0.304585   0.050334   6.051 3.42e-09 ***
## model_year79  0.486691   0.053116   9.163  < 2e-16 ***
## model_year80  0.738068   0.054778  13.474  < 2e-16 ***
## model_year81  0.588091   0.054255  10.839  < 2e-16 ***
## model_year82  0.641131   0.053624  11.956  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2159 on 383 degrees of freedom
## Multiple R-squared:  0.8849, Adjusted R-squared:  0.8807
## F-statistic: 210.3 on 14 and 383 DF,  p-value: < 2.2e-16
```

VIF para verificar a multicolinearidade

```
# Calcular o VIF
vif_valores <- vif(lm_mpg_full)
print(vif_valores)
```

```
##       weight acceleration model_year70 model_year71 model_year72 model_year74
##     1.330361     1.322372     1.623221     1.600767     1.588023     1.609009
## model_year75 model_year76 model_year77 model_year78 model_year79 model_year80
##     1.650488     1.727367     1.603861     1.779884     1.627523     1.731008
## model_year81 model_year82
##     1.698115     1.763645
```

Step Wise

```
# Step partindo do modelo nulo até o modelo completo
forw <- step(lm_mpg_nulo, scope=list(lower=lm_mpg_nulo, upper=lm_mpg_full), direction = "forward")
```

```
## Start:  AIC=-373.14
## mpg_boxcox ~ 1
##
##                Df Sum of Sq     RSS     AIC
## + weight        1   117.209  37.864 -932.28
## + acceleration  1    30.260 124.813 -457.54
## + model_year80  1    17.283 137.790 -418.17
## + model_year82  1    13.561 141.511 -407.56
## + model_year81  1     9.201 145.872 -395.48
## + model_year70  1     7.711 147.362 -391.44
## + model_year72  1     4.624 150.449 -383.19
## + model_year75  1     1.715 153.358 -375.56
## + model_year71  1     0.897 154.176 -373.45
## <none>                      155.073 -373.14
## + model_year79  1     0.717 154.356 -372.98
## + model_year76  1     0.635 154.438 -372.77
## + model_year78  1     0.212 154.861 -371.68
## + model_year74  1     0.054 155.019 -371.28
## + model_year77  1     0.005 155.067 -371.15
##
## Step:  AIC=-932.28
## mpg_boxcox ~ weight
##
##                Df Sum of Sq    RSS     AIC
## + model_year80  1    5.1884 32.675 -988.93
## + model_year82  1    3.1973 34.666 -965.39
## + model_year81  1    2.0837 35.780 -952.81
## + model_year70  1    1.8090 36.055 -949.76
## + model_year72  1    1.4730 36.391 -946.07
## + model_year79  1    1.3272 36.536 -944.48
## + acceleration  1    1.1664 36.697 -942.73
## + model_year71  1    0.7382 37.125 -938.12
## + model_year75  1    0.3099 37.554 -933.55
## + model_year74  1    0.3044 37.559 -933.49
## <none>                      37.864 -932.28
## + model_year76  1    0.1395 37.724 -931.75
## + model_year77  1    0.0284 37.835 -930.58
## + model_year78  1    0.0005 37.863 -930.28
##
## Step:  AIC=-988.93
## mpg_boxcox ~ weight + model_year80
##
##                Df Sum of Sq    RSS      AIC
## + model_year82  1    4.2754 28.400 -1042.75
## + model_year81  1    2.8856 29.790 -1023.73
## + model_year79  1    1.7588 30.916 -1008.95
## + model_year70  1    1.4847 31.191 -1005.44
## + model_year72  1    1.1500 31.525 -1001.19
## + acceleration  1    0.8397 31.836  -997.30
## + model_year71  1    0.4707 32.205  -992.71
## <none>                      32.675  -988.93
## + model_year75  1    0.1604 32.515  -988.89
## + model_year74  1    0.1338 32.542  -988.57
## + model_year77  1    0.1186 32.557  -988.38
## + model_year78  1    0.0598 32.616  -987.66
## + model_year76  1    0.0368 32.639  -987.38
##
## Step:  AIC=-1042.75
## mpg_boxcox ~ weight + model_year80 + model_year82
##
##                Df Sum of Sq    RSS     AIC
## + model_year81  1    3.8905 24.509 -1099.4
## + model_year79  1    2.2876 26.112 -1074.2
## + model_year70  1    1.1742 27.226 -1057.5
## + model_year72  1    0.8506 27.549 -1052.8
## + acceleration  1    0.6546 27.745 -1050.0
## + model_year77  1    0.2825 28.117 -1044.7
## + model_year71  1    0.2528 28.147 -1044.3
## + model_year78  1    0.2319 28.168 -1044.0
## <none>                      28.400 -1042.8
## + model_year75  1    0.0550 28.345 -1041.5
## + model_year74  1    0.0285 28.371 -1041.2
## + model_year76  1    0.0000 28.400 -1040.8
##
## Step:  AIC=-1099.38
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81
##
##                Df Sum of Sq    RSS     AIC
```

```
## + model_year79  1   2.92381 21.586 -1147.9
## + model_year70  1   0.87826 23.631 -1111.9
## + model_year72  1   0.58054 23.929 -1106.9
## + acceleration  1   0.56389 23.945 -1106.7
## + model_year78  1   0.53250 23.977 -1106.1
## + model_year77  1   0.53053 23.979 -1106.1
## <none>                      24.509 -1099.4
## + model_year71  1   0.09654 24.413 -1099.0
## + model_year76  1   0.04258 24.467 -1098.1
## + model_year75  1   0.00331 24.506 -1097.4
## + model_year74  1   0.00131 24.508 -1097.4
##
## Step:  AIC=-1147.94
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79
##
##                Df Sum of Sq    RSS     AIC
## + model_year78  1   0.88547 20.700 -1162.6
## + model_year77  1   0.82538 20.760 -1161.5
## + model_year70  1   0.58761 20.998 -1156.9
## + acceleration  1   0.43469 21.151 -1154.0
## + model_year72  1   0.35115 21.234 -1152.5
## + model_year76  1   0.16242 21.423 -1149.0
## <none>                      21.586 -1147.9
## + model_year74  1   0.04452 21.541 -1146.8
## + model_year71  1   0.01851 21.567 -1146.3
## + model_year75  1   0.01510 21.570 -1146.2
##
## Step:  AIC=-1162.61
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78
##
##                Df Sum of Sq    RSS     AIC
## + model_year77  1   1.08848 19.612 -1182.1
## + model_year70  1   0.43076 20.269 -1169.0
## + acceleration  1   0.39245 20.308 -1168.2
## + model_year76  1   0.29759 20.402 -1166.4
## + model_year72  1   0.22953 20.471 -1165.0
## + model_year74  1   0.11658 20.584 -1162.9
## <none>                      20.700 -1162.6
## + model_year75  1   0.06199 20.638 -1161.8
## + model_year71  1   0.00011 20.700 -1160.6
##
## Step:  AIC=-1182.11
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77
##
##                Df Sum of Sq    RSS     AIC
## + model_year76  1   0.49920 19.112 -1190.4
## + acceleration  1   0.37558 19.236 -1187.8
## + model_year70  1   0.27742 19.334 -1185.8
## + model_year74  1   0.23517 19.376 -1184.9
## + model_year75  1   0.15447 19.457 -1183.3
## + model_year72  1   0.12119 19.490 -1182.6
## <none>                      19.612 -1182.1
## + model_year71  1   0.01682 19.595 -1180.5
##
## Step:  AIC=-1190.37
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77 + model_year76
##
##                Df Sum of Sq    RSS     AIC
## + model_year74  1   0.37566 18.737 -1196.3
## + acceleration  1   0.30079 18.812 -1194.7
## + model_year75  1   0.27433 18.838 -1194.1
## + model_year70  1   0.17134 18.941 -1192.0
## <none>                      19.112 -1190.4
## + model_year71  1   0.06424 19.048 -1189.7
## + model_year72  1   0.05475 19.058 -1189.5
##
## Step:  AIC=-1196.27
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77 + model_year76 +
##     model_year74
##
##                Df Sum of Sq    RSS     AIC
## + model_year75  1   0.42285 18.314 -1203.4
## + acceleration  1   0.23849 18.498 -1199.4
## + model_year71  1   0.14094 18.596 -1197.3
```

```
## + model_year70  1   0.09672 18.640 -1196.3
## <none>                      18.737 -1196.3
## + model_year72  1   0.01618 18.721 -1194.6
##
## Step:  AIC=-1203.36
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77 + model_year76 +
##     model_year74 + model_year75
##
##               Df Sum of Sq    RSS     AIC
## + model_year71  1  0.294547 18.019 -1207.8
## + acceleration  1  0.152822 18.161 -1204.7
## <none>                      18.314 -1203.4
## + model_year70  1  0.026683 18.287 -1201.9
## + model_year72  1  0.000538 18.313 -1201.4
##
## Step:  AIC=-1207.81
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77 + model_year76 +
##     model_year74 + model_year75 + model_year71
##
##               Df Sum of Sq    RSS     AIC
## + acceleration  1  0.130456 17.889 -1208.7
## <none>                      18.019 -1207.8
## + model_year72  1  0.036013 17.983 -1206.6
## + model_year70  1  0.000020 18.019 -1205.8
##
## Step:  AIC=-1208.7
## mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 +
##     model_year79 + model_year78 + model_year77 + model_year76 +
##     model_year74 + model_year75 + model_year71 + acceleration
##
##               Df Sum of Sq    RSS     AIC
## <none>                      17.889 -1208.7
## + model_year72  1 0.0227542 17.866 -1207.2
## + model_year70  1 0.0027173 17.886 -1206.8
```

```
summary(forw)
```

```
##
## Call:
## lm(formula = mpg_boxcox ~ weight + model_year80 + model_year82 +
##     model_year81 + model_year79 + model_year78 + model_year77 +
##     model_year76 + model_year74 + model_year75 + model_year71 +
##     acceleration, data = df_auto_modelo_dummies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70373 -0.14148  0.01376  0.14510  0.69674
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.510399   0.095274  57.837  < 2e-16 ***
## weight       -0.537463   0.014713 -36.529  < 2e-16 ***
## model_year80  0.714977   0.047945  14.913  < 2e-16 ***
## model_year82  0.618059   0.046607  13.261  < 2e-16 ***
## model_year81  0.565063   0.047327  11.940  < 2e-16 ***
## model_year79  0.463946   0.046182  10.046  < 2e-16 ***
## model_year78  0.281744   0.042882   6.570 1.63e-10 ***
## model_year77  0.297588   0.046666   6.377 5.18e-10 ***
## model_year76  0.193441   0.043626   4.434 1.21e-05 ***
## model_year74  0.171148   0.047829   3.578  0.00039 ***
## model_year75  0.146132   0.045730   3.196  0.00151 **
## model_year71  0.112730   0.046577   2.420  0.01597 *
## acceleration  0.007458   0.004451   1.676  0.09463 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2156 on 385 degrees of freedom
## Multiple R-squared:  0.8846, Adjusted R-squared:  0.881
## F-statistic:   246 on 12 and 385 DF,  p-value: < 2.2e-16
```

Modelo Step Wise Final

- lm(formula = mpg_boxcox ~ weight + model_year80 + model_year82 + model_year81 + model_year79 + model_year78 + model_year77 + model_year76 + model_year74 + model_year75 + model_year71 + acceleration, data = df_auto_modelo_dummies)

Verificações dos resíduos

- Média em torno de zero
- Normalidade
- Variância

```
# melhor AIC

lm_mpg_aic <- lm(mpg_boxcox ~ weight + model_year80 + model_year82 +
                   model_year81 + model_year79 + model_year78 + model_year77 +
                   model_year76 + model_year74 + model_year75 + model_year71 +
                   acceleration, data = df_auto_modelo_dummies)
summary(lm_mpg_aic)
```

```
##
## Call:
## lm(formula = mpg_boxcox ~ weight + model_year80 + model_year82 +
##     model_year81 + model_year79 + model_year78 + model_year77 +
##     model_year76 + model_year74 + model_year75 + model_year71 +
##     acceleration, data = df_auto_modelo_dummies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70373 -0.14148  0.01376  0.14510  0.69674
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.510399   0.095274  57.837  < 2e-16 ***
## weight       -0.537463   0.014713 -36.529  < 2e-16 ***
## model_year80  0.714977   0.047945  14.913  < 2e-16 ***
## model_year82  0.618059   0.046607  13.261  < 2e-16 ***
## model_year81  0.565063   0.047327  11.940  < 2e-16 ***
## model_year79  0.463946   0.046182  10.046  < 2e-16 ***
## model_year78  0.281744   0.042882   6.570 1.63e-10 ***
## model_year77  0.297588   0.046666   6.377 5.18e-10 ***
## model_year76  0.193441   0.043626   4.434 1.21e-05 ***
## model_year74  0.171148   0.047829   3.578  0.00039 ***
## model_year75  0.146132   0.045730   3.196  0.00151 **
## model_year71  0.112730   0.046577   2.420  0.01597 *
## acceleration  0.007458   0.004451   1.676  0.09463 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2156 on 385 degrees of freedom
## Multiple R-squared:  0.8846, Adjusted R-squared:  0.881
## F-statistic:   246 on 12 and 385 DF,  p-value: < 2.2e-16
```

```
library(olsrr)
ols_vif_tol(lm_mpg_aic)
```

```
##       Variables Tolerance      VIF
## 1        weight 0.7539033 1.326430
## 2  model_year80 0.7517985 1.330144
## 3  model_year82 0.7482986 1.336365
## 4  model_year81 0.7715644 1.296068
## 5  model_year79 0.8102729 1.234152
## 6  model_year78 0.7716921 1.295854
## 7  model_year77 0.8196867 1.219978
## 8  model_year76 0.7851067 1.273712
## 9  model_year74 0.8070232 1.239122
## 10 model_year75 0.8010002 1.248439
## 11 model_year71 0.8228216 1.215330
## 12 acceleration 0.7768660 1.287223
```

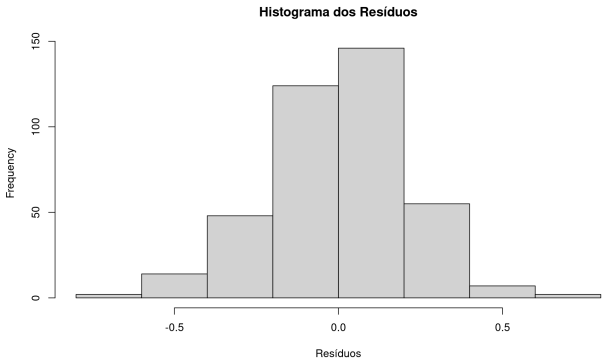
```
media_residuos <- mean(residuals(lm_mpg_aic))
print(media_residuos) # Deve ser próximo de 0
```

```
## [1] -8.529988e-18
```

média próxima de zero … ok

Teste de normalidade

```
hist(residuals(lm_mpg_aic), main = "Histograma dos Resíduos", xlab = "Resíduos")
```




```
# Teste de normalidade (Shapiro-Wilk)
resultado <- shapiro.test(residuals(lm_mpg_aic))
print(resultado$p.value)
```

```
## [1] 0.05280657
```

```
# Não passou no teste de normalidade p-valor > 0.05

# Verificar usando o teste de normalidade do KolmogorovSmirnov

residuos <- residuals(lm_mpg_aic)
desvio_residuos <- sd(residuos)

# Realiza o teste de Kolmogorov-Smirnov para verificar a normalidade
ks_test <- ks.test(residuos, "pnorm", mean = media_residuos, sd = desvio_residuos)

# Exibe o resultado do teste
print(ks_test)
```

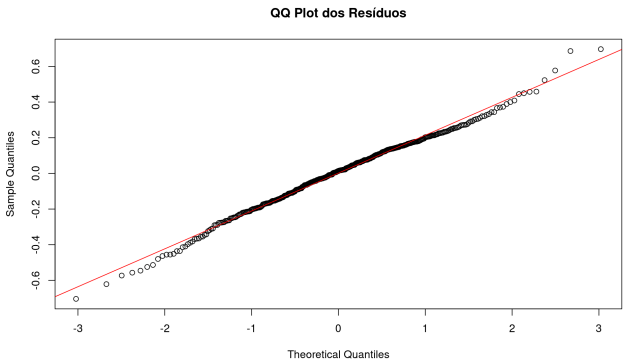
```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuos
## D = 0.031446, p-value = 0.8261
## alternative hypothesis: two-sided
```

Não passou no teste de normalidade ao nível de significância de 5%

Shapiro-Wilk nem Kolmogorov-Smirnov

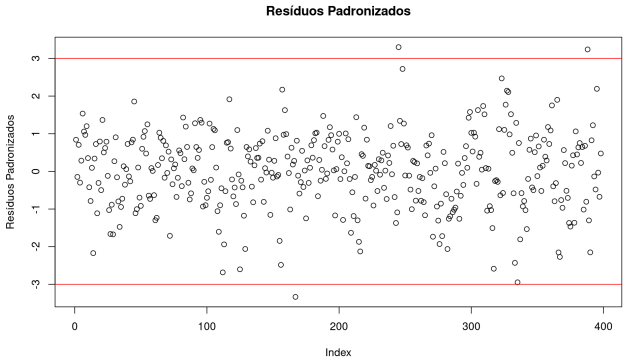
Normalidade dos resíduos

```
qqnorm(residuals(lm_mpg_aic), main = "QQ Plot dos Resíduos")
qqline(residuals(lm_mpg_aic), col = "red")
```



É possivel notar que nas estremidades o modelo não responderá bem, porém para valores mais centrais o modelo tem um bom ajuste

```
# gráfico dos resíduos padronizados
residuos_padronizados <- rstandard(lm_mpg_aic)
plot(residuos_padronizados, main = "Resíduos Padronizados", ylab = "Resíduos Padronizados")
abline(h = c(-3, 3), col = "red") # Limites para outliers
```



Resíduos padronizados devem estar distribuídos aleatoriamente em torno de 0. Valores fora dos limites ±3 (linhas vermelhas) são considerados potenciais outliers.

```
library(lmtest)
bptest(lm_mpg_aic)
```

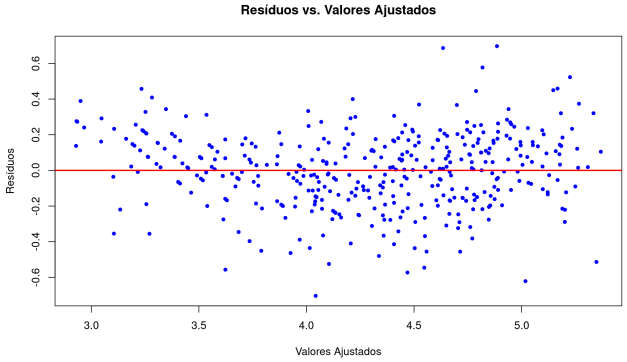
```
##
##  studentized Breusch-Pagan test
##
## data:  lm_mpg_aic
## BP = 25.407, df = 12, p-value = 0.01301
```

Resultado Breusch-Pagan test

- Como o valor-p é menor que 0.05, rejeitamos a hipótese nula (H0) ao nível de significância de 5%.
- Isso indica que há evidências estatísticas de heterocedasticidade nos resíduos e compromete a validade das inferências do modelo linear.
- Os erros padrão das estimativas podem ser incorretos, afetando os testes de significância e intervalos de confiança.

Resíduos vs. Valores Ajustados

```
# Extrair os resíduos e os valores ajustados
residuos <- residuals(lm_mpg_aic)
valores_ajustados <- fitted(lm_mpg_aic)

# Criar o gráfico
plot(valores_ajustados, residuos,
     xlab = "Valores Ajustados",
     ylab = "Resíduos",
     main = "Resíduos vs. Valores Ajustados",
     pch = 20, col = "blue")  # Personalização dos pontos

# Adicionar uma linha horizontal em y = 0
abline(h = 0, col = "red", lwd = 2)
```



Resultado

- Os resíduos não estão distribuídos aleatoriamente em torno de y=0
- Isso indica heterocedasticidade (variância não constante).
- Padrão sistemático pode indicar que o modelo não capturou adequadamente a relação entre as variáveis.

Alternativa Modelo PCA

- Construir um modelo com as variáveis do PCA
- Usar os componentes principais como preditores no modelo, em vez das variáveis originais, e verificar se o desempenho do modelo melhora.

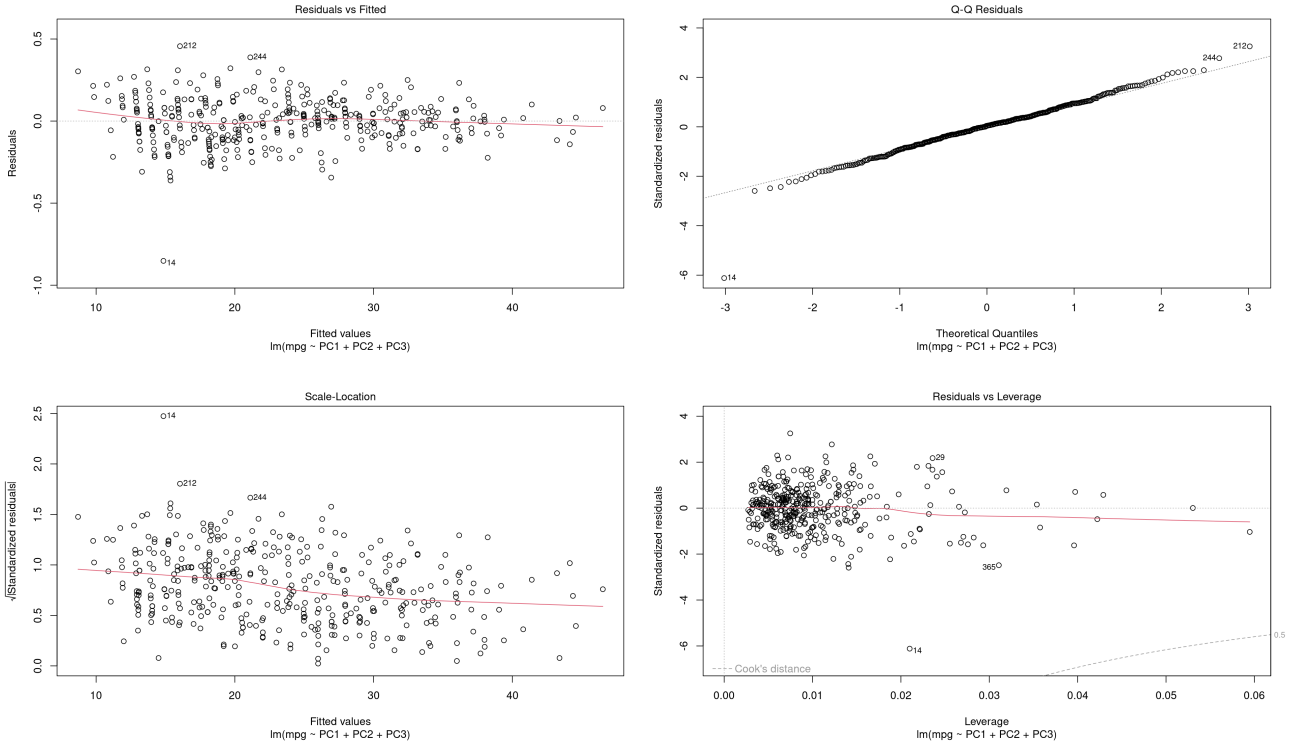
```
# Remover as linhas com valores ausentes (NA) do data frame df_auto
df_auto_limpo <- na.omit(df_auto)

# Selecionar as variáveis numéricas para a PCA
df_pca <- df_auto_limpo[, c("horsepower", "mpg", "displacement", "weight", "acceleration")]

# Padronizar as variáveis (opcional, mas recomendado para PCA)
df_pca <- scale(df_pca)

# Aplicar a PCA
pca_result <- prcomp(df_pca, scale = TRUE)

# Resumo dos resultados da PCA
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5
## Standard deviation     1.981 0.8445 0.47549 0.28799 0.23000
## Proportion of Variance 0.785 0.1426 0.04522 0.01659 0.01058
## Cumulative Proportion  0.785 0.9276 0.97283 0.98942 1.00000
```

```
# Criar um data frame com os componentes principais
df_pca_model <- as.data.frame(pca_result$x[, 1:3])  # Seleciona os 3 primeiros PCs
colnames(df_pca_model) <- c("PC1", "PC2", "PC3")    # Renomeia os PCs para facilitar

# Adicionar a variável dependente (mpg) ao data frame sincronizado
df_pca_model$mpg <- df_auto_limpo$mpg

# Ajustar o modelo de regressão linear usando os PCs como preditores
modelo_pca <- lm(mpg ~ PC1 + PC2 + PC3, data = df_pca_model)

# Resumo do modelo
summary(modelo_pca)
```

```
##
## Call:
## lm(formula = mpg ~ PC1 + PC2 + PC3, data = df_pca_model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85202 -0.08541  0.00587  0.08168  0.45641
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.416667   0.007128  3285.3   <2e-16 ***
## PC1          3.469534   0.003602   963.1   <2e-16 ***
## PC2          2.377900   0.008451   281.4   <2e-16 ***
## PC3         -6.554449   0.015009  -436.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1408 on 386 degrees of freedom
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9997
## F-statistic: 3.992e+05 on 3 and 386 DF,  p-value: < 2.2e-16
```

```
# Diagnóstico do modelo
#par(mfrow = c(2, 2))  # Configura layout para múltiplos gráficos
plot(modelo_pca)
```




```
# Teste de normalidade dos resíduos
shapiro.test(residuals(modelo_pca))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(modelo_pca)
## W = 0.97487, p-value = 2.779e-06
```

```
# Teste de homocedasticidade (Breusch-Pagan)
library(lmtest)
bptest(modelo_pca)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo_pca
## BP = 14.82, df = 3, p-value = 0.001977
```

Qualidade e Acurácia dos modelos

```
## RMSE do modelo com AIC: 20.5461
```

```
## RMSE do modelo com PCA: 0.1400363
```

```
## R² do modelo com AIC: 0.884642
```

```
## R² do modelo com PCA: 0.9996778
```

```
## MAE do modelo com AIC: 19.21585
```

```
## MAE do modelo com PCA: 0.1064248
```

```
## MSE do modelo com AIC: 422.1421
```

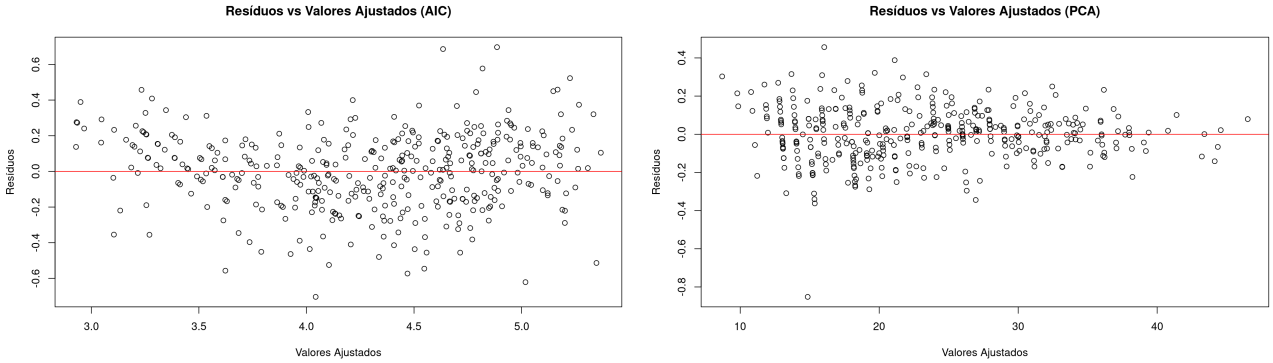
```
## MSE do modelo com PCA: 0.01961015
```

```
## RMSE do modelo com AIC: 20.5461
```

```
## RMSE do modelo com PCA: 0.1400363
```

```
## MAPE do modelo com AIC: 80.41983 %
```

```
## MAPE do modelo com PCA: 0.5516365 %
```



Conclusões

- O modelo baseado na PCA supera amplamente o modelo baseado no AIC em termos de qualidade do ajuste e acurácia preditiva, conforme evidenciado pelas métricas calculadas.
- Os componentes principais (PCs) são combinações lineares das variáveis originais, o que pode dificultar a interpretação direta do modelo.