Relatório 1 - Regressão

Flavio Margarito Martins de Barros Gabriel Tupinamba da Cunha Leandro
Gustavo Leite Machado

14/05/2022

```
## Carregando os pacotes
require(readxl)
require(corrplot)
require(psych)
require(kableExtra)
require(caret)
require(GGally)
require(Hmisc)
```

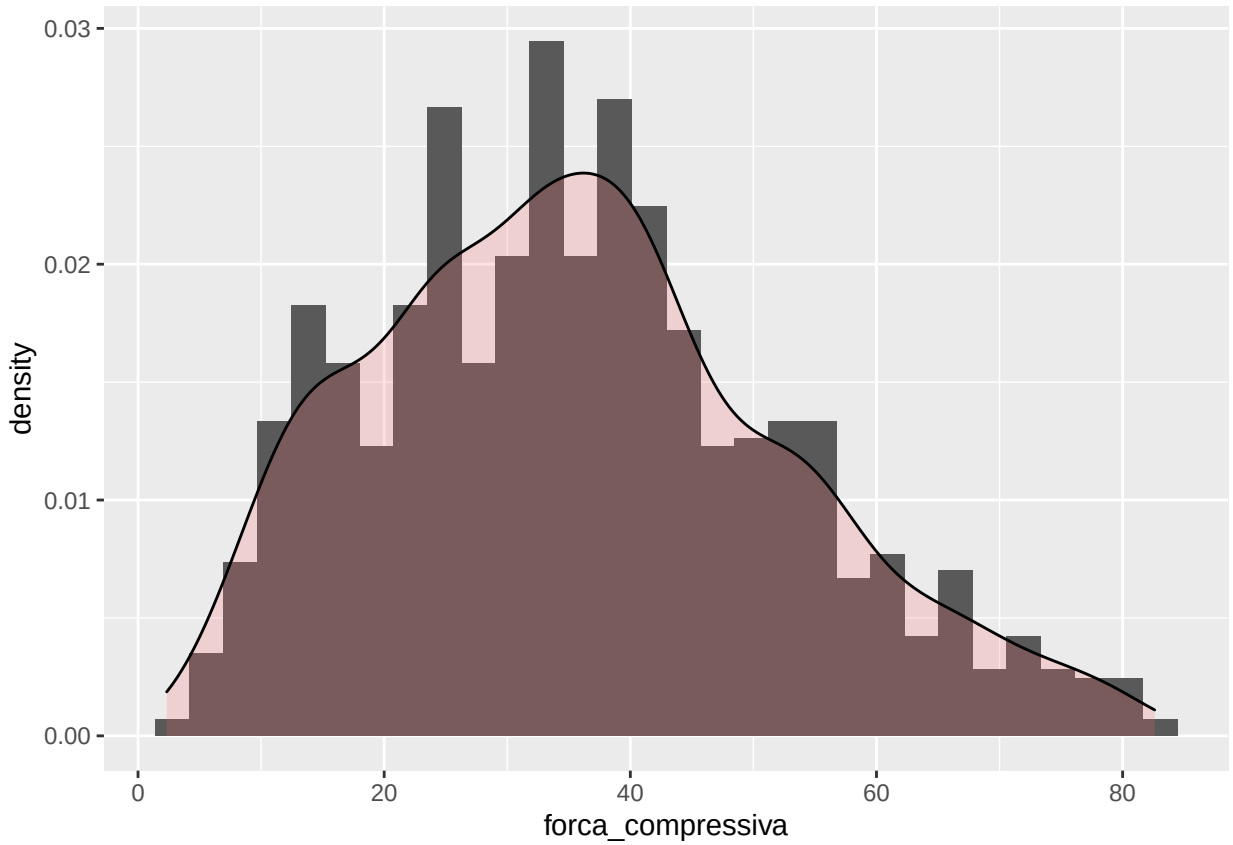
Descrição básica dos dados

```
## Lendo o banco de dados
## Fonte: https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
dados <- read_excel(path = "Concrete_Data.xls", sheet = 1)

## Trocando os nomes das variáveis para o português
colnames(dados) <-
  c(
    "cimento",
    "escoria",
    "cinza",
    "agua",
    "super_plastificante",
    "agregador_grosso",
    "agregador_fino",
    "idade",
    "forca_compressiva"
  )

sum(is.na(data.frame(dados)))
```

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

```
ggplot(dados, aes(x = forca_compressiva)) +
  geom_histogram(aes(y=..density..)) +
  geom_density(alpha=.2, fill="#FF6666")
```

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



- Cimento (kg/m^3)
- Escoria (kg/m^3)
- Cinza (kg/m^3)
- Agua (kg/m^3)
- Super plastificante (kg/m^3)
- Agregadro grosso (kg/m^3)
- Agregador fino (kg/m^3)
- Idade (Dias 1 a 365)
- Força compressiva (Target) (MPa)

Como podemos notar, temos 1030 observações, 8 variáveis explicativas e nossa variável de interesse (força compressiva), e nenhum dado faltante nas observações. Pela descrição básica dos dados, não temos nenhum dado que parece fugir dos valores esperados (por exemplo: não temos valores negativos).

Analisando a nossa variável resposta podemos notar que sua distribuição se assemelha a uma normal

```
## Sumario dos dados
d <- Hmisc::describe(dados)
```

dados

9 Variables **1030 Observations**

cimento

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	280	1	281.2	118.5	143.7	153.5	192.4	272.9	350.0	425.0	480.0

```
lowest : 102.0 108.3 116.0 122.6 132.0, highest: 522.0 525.0 528.0 531.3 540.0
```

escoria

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	187	0.907	73.9	91.71	0.0	0.0	0.0	22.0	142.9	192.0	236.0

```
lowest :   0.00   0.02  11.00  13.61  15.00, highest: 290.20 305.30 316.10 342.10 359.40
```

cinza

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	163	0.834	54.19	67.08	0.0	0.0	0.0	0.0	118.3	141.1	167.0

```
lowest :   0.00  24.46  24.51  24.52  59.00, highest: 194.00 194.90 195.00 200.00 200.10
```

agua

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	205	0.998	181.6	23.82	146.1	154.6	164.9	185.0	192.0	203.5	228.0

```
lowest : 121.75 126.60 127.00 127.30 137.80, highest: 228.00 236.70 237.00 246.90 247.00
```

super_plastificante

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	155	0.95	6.203	6.426	0.00	0.00	0.00	6.35	10.16	12.21	16.05

```
lowest :  0.00  1.72  1.90  2.00  2.20, highest: 22.00 22.10 23.40 28.20 32.20
```

agregador_grosso

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	284	1	972.9	88.55	842.0	852.1	932.0	968.0	1029.4	1076.5	1104.0

```
lowest :  801.0  801.1  801.4  811.0  814.0, highest: 1124.4 1125.0 1130.0 1134.3 1145.0
```

agregador_fino

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	304	1	773.6	89.87	613.0	664.1	730.9	779.5	824.0	880.8	898.1

```
lowest : 594.0 605.0 611.8 612.0 613.0, highest: 925.7 942.0 943.1 945.0 992.6
```

idade

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	14	0.925	45.66	50.89	3	3	7	28	56	100	180

```
lowest :   1   3   7  14  28, highest: 120 180 270 360 365
```

Value	1	3	7	14	28	56	90	91	100	120	180	270	360	365
Frequency	2	134	126	62	425	91	54	22	52	3	26	13	6	14
Proportion	0.002	0.130	0.122	0.060	0.413	0.088	0.052	0.021	0.050	0.003	0.025	0.013	0.006	0.014

forca_compressiva

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25	.50	.75	.90	.95
1030	0	938	1	35.82	18.92	10.96	14.20	23.71	34.44	46.14	58.82	66.80

```
lowest :  2.331808  3.319827  4.565021  4.782206  4.827711
highest: 79.400056 79.986111 80.199848 81.751169 82.599225
```

Preparação dos dados

```
## Separando o conjunto de dados em treino e teste
set.seed(2)
inTrain <- createDataPartition(dados$forca_compressiva, p = 7/10)[[1]]
treino <- dados[inTrain,]
teste <- dados[-inTrain,]

## Mantendo casos completos em treino e teste
treino <- treino[complete.cases(treino),]
teste <- teste[complete.cases(teste),]

## Separando a variavel resposta, categóricas e numericas
resposta <- treino$forca_compressiva
resposta_teste <- teste$forca_compressiva

## Criando dataset normalizado para avaliar diferença de resultado
normalized_train <- treino
normalized_teste <- teste

maxTrainFeatures <- apply(normalized_train[,1:8], 2, max) #max of each feature
```

```r
minTrainFeatures <- apply(normalized_train[,1:8], 2, min) #min of each feature


minMaxDiffTrain <- (maxTrainFeatures - minTrainFeatures)
minMaxDiffTrain
```

```
##             cimento             escoria               cinza               agua
##              438.00              359.40              200.10              125.25
## super_plastificante    agregador_grosso      agregador_fino               idade
##               32.20              344.00              398.60              364.00
```

```r
normalized_train[,1:8] <- sweep(normalized_train[,1:8], 2, minTrainFeatures, "-")
normalized_train[,1:8] <- sweep(normalized_train[,1:8], 2, minMaxDiffTrain, "/")

normalized_teste[,1:8] <- sweep(normalized_teste[,1:8], 2, minTrainFeatures, "-")
normalized_teste[,1:8] <- sweep(normalized_teste[,1:8], 2, minMaxDiffTrain, "/")

## Retendo as numéricas
Ind_numericas <- colnames(treino[,-ncol(treino)])[sapply(treino[,-ncol(treino)], is.numeric)]
Ind_categoricas <- colnames(treino[,-ncol(treino)])[sapply(treino[,-ncol(treino)], function(x) !is.nume
numericas <- treino[,Ind_numericas]
categorias <- treino[,Ind_categoricas]
```

Redução de dimensionalidade

Estrutura de correlações

Como são todas variáveis numéricas inicialmente veremos na matriz de correlação se há algumas relação mais forte entre pares de variáveis. Se houver poderemos escolher somente uma das variáveis pois adiconar outra variável fortemente correlacionanda não adicionaria novas informações e traria dificuldades no processo de estimação em virtude de possível multicolinearidade.

```r
## Adicionando pacote corrplot
require(corrplot)
```

```
## Carregando pacotes exigidos: corrplot
```

```
## corrplot 0.92 loaded
```

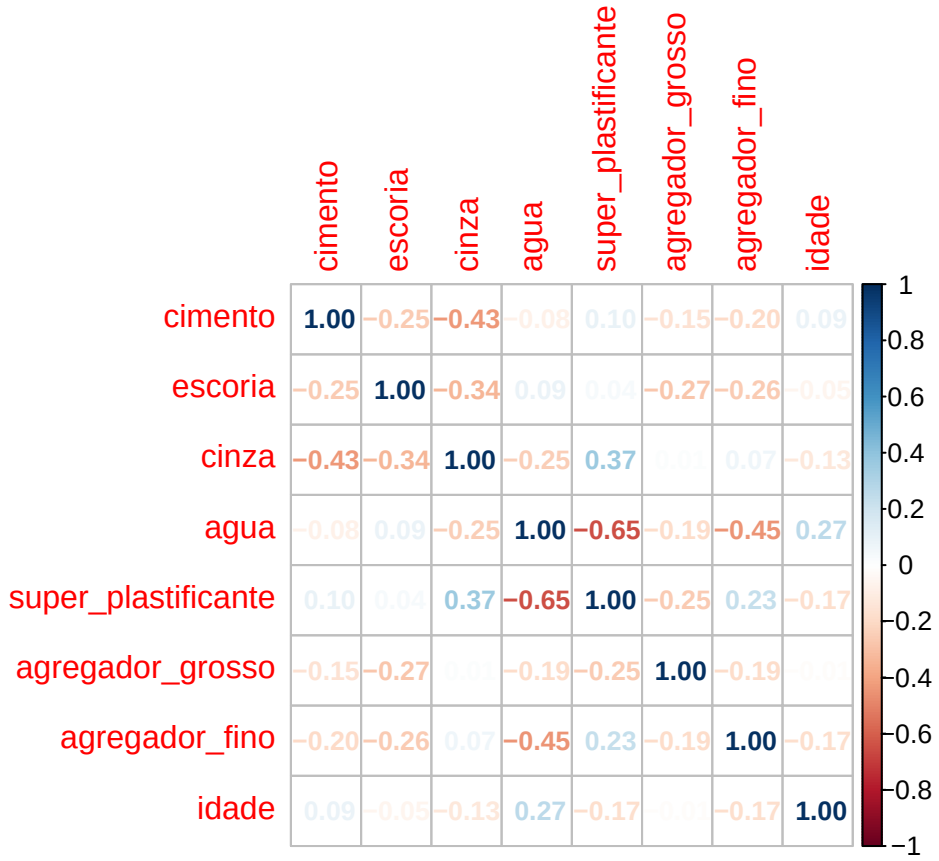
```r
require(GGally)
```

```
## Carregando pacotes exigidos: GGally
```

```
## Carregando pacotes exigidos: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
## Analisando as correlações
M <- cor(numericas, use = 'complete.obs')
corrplot(M, method='number', diag = T, number.cex = 0.8)
```



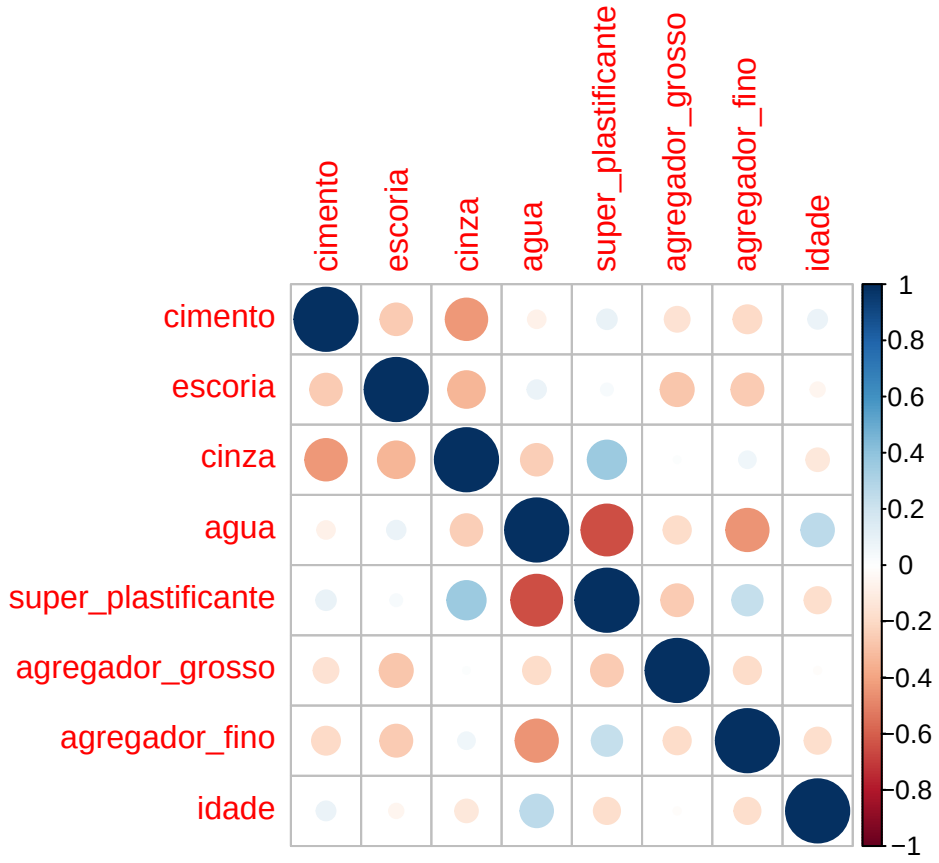

```
summary(M[upper.tri(M)])
```

```
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.64810 -0.25116 -0.16258 -0.11522  0.04417  0.36742
```

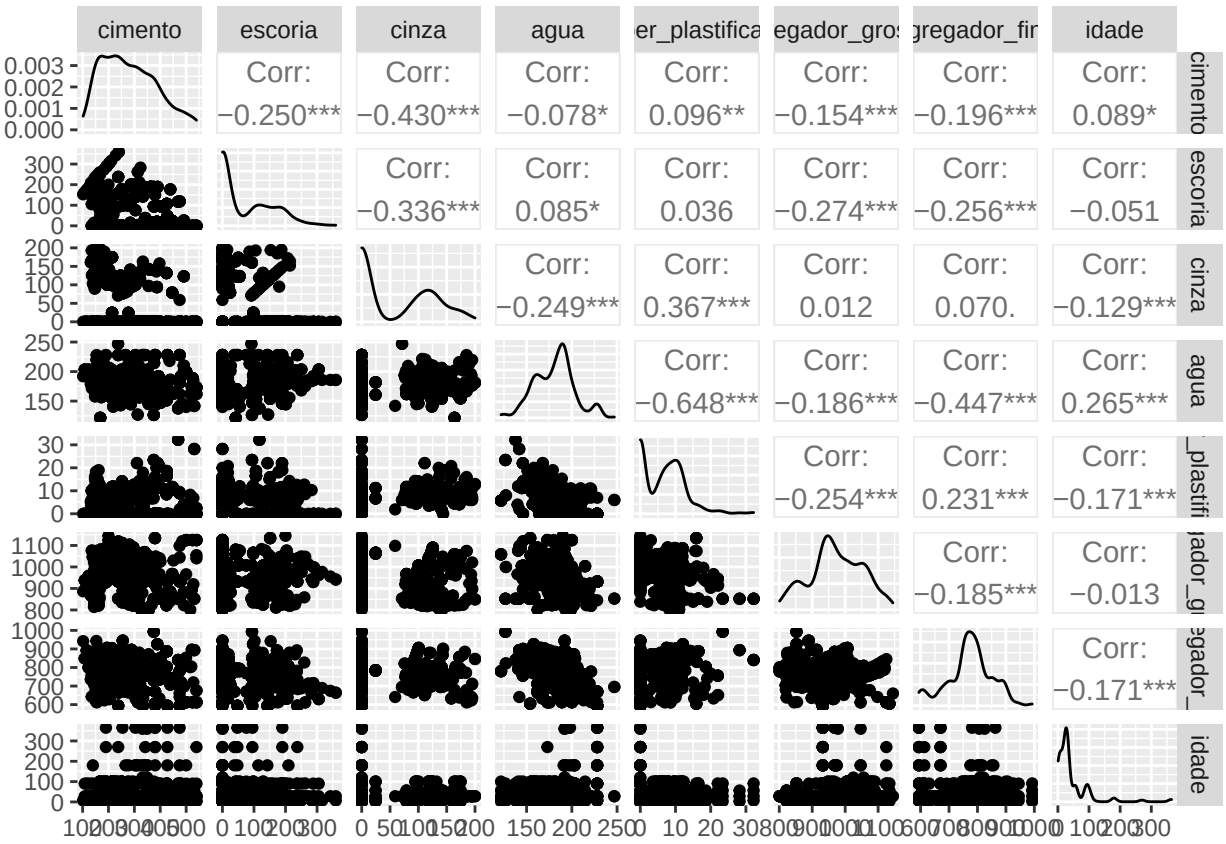
```
## Imprimindo as correlações na forma de circulos
M <- cor(numericas, use = 'complete.obs')
summary(M[upper.tri(M)])
```

```
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.64810 -0.25116 -0.16258 -0.11522  0.04417  0.36742
```

```
corrplot(M, method='circle')
```




```
## Visualizando as correlações
ggpairs(numericas)
```



Como as maiores correlações foram de $0,65$, não podemos afirmar que há pares de variáveis redundantes. Portanto optamos por não retirar nenhuma variável nessa etapa. Como não temos variáveis altamente correlacionadas, provavelmente não teremos multicolinearidade, e ainda que tívessemos, provavelmente não poderíamos remover covariáveis da análise, entretando, vamos verificar mesmo assim.

Dependendo do objetivo da análise, se queremos acertar o valor da força compressevia de uma certa batelagem de cimento, ou do cimento utilizado em uma certa obra, ou se apenas queremos entender como essas variáveis influenciam na força compressiva, temos mais ou menos liberdades para modificar as covariáveis.

Análise de redundância

Na análise de redundância utilizamos regressões de cada variável tendo as outras como suas preditoras, inclusive com componentes não lineares via *splines* cúbicos. Essa análise é superior ao correlograma no sentido de que considera não somente as relações lineares dois a dois, mas também a capacidade das preditoras fornecerem informações sobre as outras preditoras de forma conjunta.

```
redun(~ ., r2 = .8, type = "adjusted", data = numericas)
```

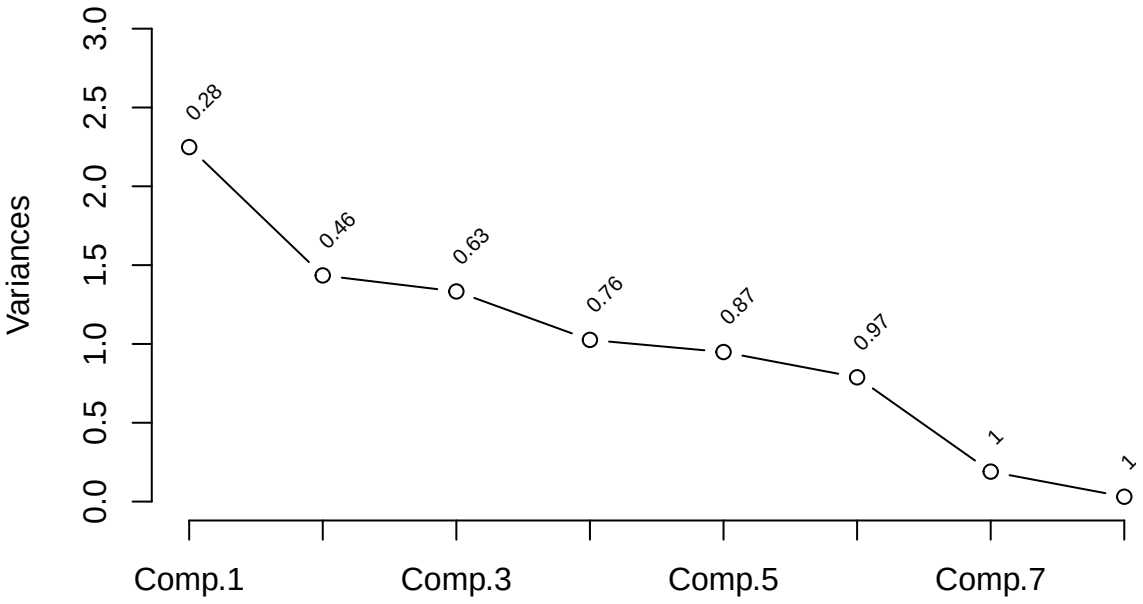
```
##
## Redundancy Analysis
##
## redun(formula = ~., data = numericas, r2 = 0.8, type = "adjusted")
##
## n: 722   p: 8    nk: 3
##
## Number of NAs:    0
##
## Transformation of target variables forced to be linear
##
```

```
## R-squared cutoff: 0.8    Type: adjusted
##
## R^2 with which each variable can be predicted from all other variables:
##
##             cimento             escoria               cinza                agua
##               0.876               0.863               0.874               0.857
## super_plastificante    agregador_grosso      agregador_fino               idade
##               0.669               0.814               0.854               0.161
##
## Rendundant variables:
##
## cimento
##
## Predicted from variables:
##
## escoria cinza agua super_plastificante agregador_grosso agregador_fino idade
##
##   Variable Deleted   R^2 R^2 after later deletions
## 1          cimento 0.876
```

Como resultado dessa análise as variáveis cimento, considerando como critério um $R^2 $ água e cinza podem ser facilmente preditas a partir das outras, portanto serão excluídas.

Estrutura das variáveis com PCA

```
# Calculando o PCA
prin.raw <- princomp (~ ., cor = TRUE , data = numericas)
plot (prin.raw, type = 'lines' , main = ' ' , ylim = c (0 ,3))

# Adicionando a variância cumulativa explicada
addscree <- function (x , npcs = min (10 , length (x$sdev)) ,
                      plotv = FALSE ,
                      col =1 , offset = .8 , adj =0 , pr = FALSE) {
                        vars <- x$sdev^2
                        cumv <- cumsum(vars)/sum(vars)
                        if (pr) print(cumv)
                        text (1: npcs , vars [1: npcs ] + offset*par ('cxy')[2] ,
                        as.character(round(cumv [1: npcs ], 2)),
                        srt =45 , adj = adj , cex = .65 , xpd = NA , col = col)
                        if ( plotv ) lines (1: npcs , vars [1: npcs ], type = ' b ' , col = col )
}
addscree (prin.raw)
```



Não parece haver uma estrutura onde as primeiras componente dominam as outras. Portanto com base nessa análise ainda não teríamos indicação de eliminar variáveis.

Modelagem

Modelos lineares com polinômios

```
## Modelo linear sem interações e sem termos polinomiais
f1 <-
  formula(
    "forca_compressiva ~ cimento + escoria + cinza + agua +
      super_plastificante + agregador_grosso +
      agregador_fino + idade"
  )

## Transformando a variável resposta pelo log()
f1_log <-
  formula(
    "log(forca_compressiva) ~ cimento + escoria + cinza + agua +
      super_plastificante + agregador_grosso +
      agregador_fino + idade"
  )

## Modelo com polinômios
f2 <- formula(
  "forca_compressiva ~ cimento + escoria + cinza + agua +
    super_plastificante + agregador_grosso + agregador_fino + idade +
    I(cimento ^ 2) + I(escoria ^ 2) + I(cinza ^ 2) +
    I(agua ^ 2) + I(super_plastificante ^2) +
    I(agregador_grosso ^ 2) + I(agregador_fino ^ 2) + I(idade ^ 2)"
  )

## Modelo com polinômio na variável transformada por log
```

```r
f2_log <- formula(
  "log(forca_compressiva) ~ cimento + escoria + cinza + agua +
    super_plastificante + agregador_grosso + agregador_fino + idade +
    I(cimento ^ 2) + I(escoria ^ 2) + I(cinza ^ 2) +
    I(agua ^ 2) + I(super_plastificante ^2) +
    I(agregador_grosso ^ 2) + I(agregador_fino ^ 2) + I(idade ^ 2)"
  )

## Salvando as fórmulas
formulas <- c(f1, f2)

## Criando os modelos
for (f in formulas) {
  ##model
  model <- lm(formula = f, data=treino)

  model_norm <- lm(formula = f, data=normalized_train)

  ##predicao treino
  treinoPred     <- predict(model, treino)
  treinoPredNorm <- predict(model_norm, normalized_train)

  ##predicao teste
  testePred     <- predict(model, teste)
  testePredNorm <- predict(model_norm, normalized_teste)


  mae_treino  <- round(MAE(treino$forca_compressiva, treinoPred), 3)
  mae_teste   <- round(MAE(teste$forca_compressiva, testePred), 3)


  rmse_treino <- round(RMSE(treino$forca_compressiva, treinoPred), 3)
  rmse_teste  <- round(RMSE(teste$forca_compressiva, testePred), 3)


  mae_norm_treino  <- round(MAE(normalized_train$forca_compressiva, treinoPredNorm), 3)
  mae_norm_teste   <- round(MAE(normalized_teste$forca_compressiva, testePredNorm), 3)

  rmse_norm_treino <- round(RMSE(normalized_train$forca_compressiva, treinoPredNorm), 3)
  rmse_norm_teste  <- round(RMSE(normalized_teste$forca_compressiva, testePredNorm), 3)

  print(f)

  print(paste0('DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n'))
  print(paste0('MAE :', mae_treino, ' --  ', mae_teste))
  print(paste0('RMSE :', rmse_treino, ' -- ', rmse_teste))

  print(paste0('DATASET NORMALIZADO (TREINO -- TESTE): \n'))
  print(paste0('MAE :', mae_norm_teste, ' -- ', mae_norm_teste))
  print(paste0('RMSE :', rmse_norm_treino, ' -- ', rmse_norm_teste))

  print('MUDANDO DE MODELO')
  print('')
```

```
  print('')
  print('')

}
```

```
## forca_compressiva ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :8.091 --  8.238"
## [1] "RMSE :10.223 -- 10.82"
## [1] "DATASET NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :8.238 -- 8.238"
## [1] "RMSE :10.223 -- 10.82"
## [1] "MUDANDO DE MODELO"
## [1] ""
## [1] ""
## [1] ""
## forca_compressiva ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade + I(cimento^2) +
##     I(escoria^2) + I(cinza^2) + I(agua^2) + I(super_plastificante^2) +
##     I(agregador_grosso^2) + I(agregador_fino^2) + I(idade^2)
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :6.062 --  6.161"
## [1] "RMSE :7.895 -- 8.116"
## [1] "DATASET NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :6.161 -- 6.161"
## [1] "RMSE :7.895 -- 8.116"
## [1] "MUDANDO DE MODELO"
## [1] ""
## [1] ""
## [1] ""
```

```
formulas_log <- c(f1_log, f2_log)

for (f in formulas_log) {
  ##model
  model <- lm(formula = f, data=treino)

  model_norm <- lm(formula = f, data=normalized_train)

  ##predicao treino
  treinoPred     <- predict(model, treino)
  treinoPredNorm <- predict(model_norm, normalized_train)

  ##predicao teste
  testePred     <- predict(model, teste)
  testePredNorm <- predict(model_norm, normalized_teste)


  mae_treino  <- round(MAE(log(treino$forca_compressiva), treinoPred), 3)
  mae_teste   <- round(MAE(log(teste$forca_compressiva), testePred), 3)


  rmse_treino <- round(RMSE(log(treino$forca_compressiva), treinoPred), 3)
```

```
  rmse_teste  <- round(RMSE(log(teste$forca_compressiva), testePred), 3)


  mae_norm_treino  <- round(MAE(log(normalized_train$forca_compressiva), treinoPredNorm), 3)
  mae_norm_teste   <- round(MAE(log(normalized_teste$forca_compressiva), testePredNorm), 3)

  rmse_norm_treino <- round(RMSE(log(normalized_train$forca_compressiva), treinoPredNorm), 3)
  rmse_norm_teste  <- round(RMSE(log(normalized_teste$forca_compressiva), testePredNorm), 3)

  print(f)

  print(paste0('DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n'))
  print(paste0('MAE :', mae_treino, ' --  ', mae_teste))
  print(paste0('RMSE :', rmse_treino, ' -- ', rmse_teste))

  print(paste0('DATASET NORMALIZADO (TREINO -- TESTE): \n'))
  print(paste0('MAE :', mae_norm_teste, ' -- ', mae_norm_teste))
  print(paste0('RMSE :', rmse_norm_treino, ' -- ', rmse_norm_teste))

  print('MUDANDO DE MODELO')
  print('')
  print('')
  print('')

}
```

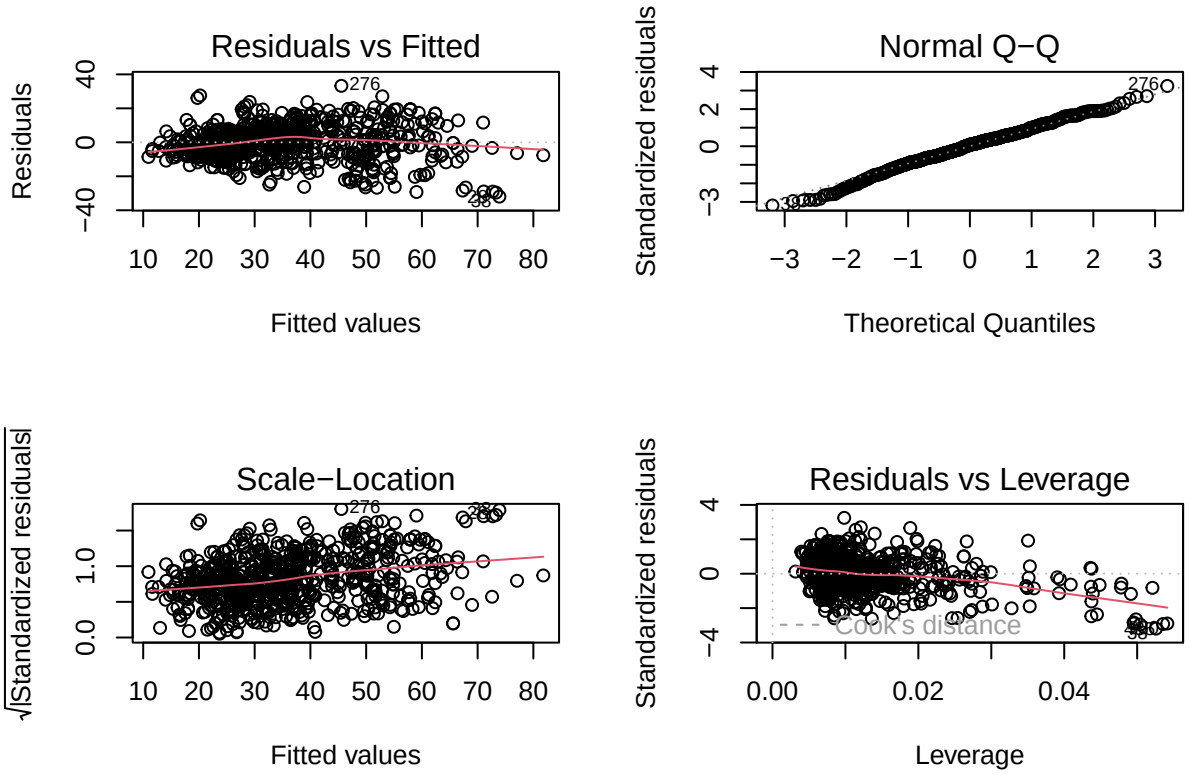
```
## log(forca_compressiva) ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.282 --  0.307"
## [1] "RMSE :0.359 -- 0.39"
## [1] "DATASET NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.307 -- 0.307"
## [1] "RMSE :0.359 -- 0.39"
## [1] "MUDANDO DE MODELO"
## [1] ""
## [1] ""
## [1] ""
## log(forca_compressiva) ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade + I(cimento^2) +
##     I(escoria^2) + I(cinza^2) + I(agua^2) + I(super_plastificante^2) +
##     I(agregador_grosso^2) + I(agregador_fino^2) + I(idade^2)
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.219 --  0.239"
## [1] "RMSE :0.285 -- 0.302"
## [1] "DATASET NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.239 -- 0.239"
## [1] "RMSE :0.285 -- 0.302"
## [1] "MUDANDO DE MODELO"
## [1] ""
## [1] ""
## [1] ""
```

Aplicando o $log()$ na variável resposta, analisando MAE e $RMSE$, temos um melhor desempenho. Notamos

também que normalização fez pouca diferença, entretanto os modelos com termos quadráticos tiveram um melhor resultado. Isso já era esperado pois no artigo original dos dados o modelo original foi criado com redes neurais.

Análise de Resíduos

```
### Modelo final
final_model <- lm(formula = f1, data = treino)

## Obtendo os resíduos
e <- resid(final_model)

## Gráfico de resíduos
#plot(log(treino$forca_compressiva), e,
#     ylab="Resíduos", xlab="Valor Observado",
#     main="Análise de Resíduos")
#abline(0, 0)

## Plot automatico
par(mfrow=c(2,2))
plot(final_model)
```



Relaxando a linearidade com splines

```
## Carregando os pacotes
require(ggplot2)
require(rms)
```

```
## Carregando pacotes exigidos: rms
```

```
## Carregando pacotes exigidos: SparseM
```

```
##
## Attaching package: 'SparseM'
```

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

```
## Definindo as estatísticas de resumo para os plots
d <- datadist(dados)
options(datadist = "d")

## Ajustando um modelo com splines cúbicos
mod1 <- ols(forca_compressiva ~ rcs(cimento, 10) + rcs(escoria, 10) + rcs(cinza, 10) +
                  rcs(agua, 10) + rcs(super_plastificante, 10) + rcs(agregador_grosso, 10) +
                  rcs(agregador_fino, 10) + rcs(idade, 10),
              data = dados, x = TRUE, y = TRUE)

## Avaliando o modelo
mod1
```

```
## Linear Regression Model
##
##  ols(formula = forca_compressiva ~ rcs(cimento, 10) + rcs(escoria,
##      10) + rcs(cinza, 10) + rcs(agua, 10) + rcs(super_plastificante,
##      10) + rcs(agregador_grosso, 10) + rcs(agregador_fino, 10) +
##      rcs(idade, 10), data = dados, x = TRUE, y = TRUE)
##
##                    Model Likelihood    Discrimination
##                          Ratio Test           Indexes
##  Obs    1030    LR chi2    2384.45    R2       0.901
##  sigma5.4327    d.f.            68    R2 adj   0.894
##  d.f.    961    Pr(> chi2)  0.0000    g       17.960
##
##  Residuals
##
##        Min         1Q    Median         3Q        Max
##  -28.28060  -3.45019   0.05548   3.44484  20.80976
##
##
##                               Coef      S.E.     t      Pr(>|t|)
##  Intercept                    -261.0151  35.6424  -7.32 <0.0001
##  cimento                         0.0148   0.0425   0.35 0.7285
##  cimento'                       14.5345   3.6336   4.00 <0.0001
##  cimento''                     -40.5433  10.1141  -4.01 <0.0001
##  cimento'''                     49.7643  14.0294   3.55 0.0004
##  cimento''''                   -32.7133  13.0147  -2.51 0.0121
##  cimento'''''                   10.1922   9.7606   1.04 0.2966
##  cimento''''''                  -2.6400   9.4032  -0.28 0.7790
##  cimento'''''''                  4.3489   7.8920   0.55 0.5817
##  cimento''''''''                -4.5610   4.2451  -1.07 0.2829
##  escoria                        -0.2001   0.0861  -2.32 0.0203
##  escoria'                       26.2352   7.4369   3.53 0.0004
##  escoria''                     -51.6859  15.3614  -3.36 0.0008
##  escoria'''                     30.2397  10.6131   2.85 0.0045
```

```
##  escoria''''                 -16.4663  15.4725  -1.06 0.2875
##  escoria'''''                 30.5993  27.9252   1.10 0.2735
##  escoria''''''               -34.3873  25.1431  -1.37 0.1717
##  escoria'''''''               26.5858  17.1638   1.55 0.1217
##  escoria''''''''             -21.9741  13.1474  -1.67 0.0950
##  cinza                         0.2318   0.0634   3.65 0.0003
##  cinza'                       -3.1019   0.8366  -3.71 0.0002
##  cinza''                      25.2546   5.4325   4.65 <0.0001
##  cinza'''                   -476.8807  80.4351  -5.93 <0.0001
##  cinza''''                   679.3583 116.6114   5.83 <0.0001
##  cinza'''''                 -535.1631 142.9617  -3.74 0.0002
##  cinza''''''                 522.7019 311.8302   1.68 0.0940
##  cinza'''''''               -186.7899 235.3608  -0.79 0.4276
##  cinza''''''''               -32.7182  29.8688  -1.10 0.2736
##  agua                          0.1957   0.0854   2.29 0.0221
##  agua'                        -7.5964   1.7878  -4.25 <0.0001
##  agua''                       44.2764  12.7657   3.47 0.0005
##  agua'''                     -69.7609  26.2823  -2.65 0.0081
##  agua''''                     45.6506  45.3754   1.01 0.3146
##  agua'''''                    82.3122  90.4877   0.91 0.3632
##  agua''''''                 -278.9073 124.3037  -2.24 0.0251
##  agua'''''''                1032.9934 405.8051   2.55 0.0111
##  agua''''''''               -863.7800 354.8309  -2.43 0.0151
##  super_plastificante           1.3357   0.4633   2.88 0.0040
##  super_plastificante'         -88.1416  36.3622  -2.42 0.0155
##  super_plastificante''        299.5256 124.5663   2.40 0.0164
##  super_plastificante'''      -824.1545 358.0533  -2.30 0.0216
##  super_plastificante''''     1220.6629 690.5185   1.77 0.0774
##  super_plastificante'''''    -599.5187 736.9680  -0.81 0.4161
##  super_plastificante''''''   -116.2816 831.4576  -0.14 0.8888
##  super_plastificante'''''''  -204.7036 798.0199  -0.26 0.7976
##  super_plastificante''''''''  609.6462 386.1655   1.58 0.1147
##  agregador_grosso              0.0071   0.0330   0.22 0.8295
##  agregador_grosso'            -0.2184   0.5499  -0.40 0.6914
##  agregador_grosso''            1.4233   1.8925   0.75 0.4522
##  agregador_grosso'''         -30.0700  25.5244  -1.18 0.2391
##  agregador_grosso''''         43.0818  46.4942   0.93 0.3544
##  agregador_grosso'''''       -12.5037  51.2061  -0.24 0.8071
##  agregador_grosso''''''        0.3930  35.2133   0.01 0.9911
##  agregador_grosso'''''''     -13.7834  10.2539  -1.34 0.1792
##  agregador_grosso''''''''     32.7950   8.5867   3.82 0.0001
##  agregador_fino                0.3433   0.0294  11.69 <0.0001
##  agregador_fino'              -2.4365   0.2218 -10.98 <0.0001
##  agregador_fino''             15.9036   1.5243  10.43 <0.0001
##  agregador_fino'''           -44.8371   5.3083  -8.45 <0.0001
##  agregador_fino''''          131.5912  28.7074   4.58 <0.0001
##  agregador_fino'''''        -191.0212  61.1805  -3.12 0.0018
##  agregador_fino''''''        124.5494  56.0264   2.22 0.0264
##  agregador_fino'''''''       -42.6687  27.0546  -1.58 0.1151
##  agregador_fino''''''''       11.1750  10.2180   1.09 0.2744
##  idade                         1.8774   0.1282  14.64 <0.0001
##  idade'                     -421.3252  51.6007  -8.17 <0.0001
##  idade''                     508.4502  64.2769   7.91 <0.0001
##  idade'''                    -97.9556  15.5894  -6.28 <0.0001
```

```
##  idade''''                    15.4634   3.8916   3.97 <0.0001
##
```

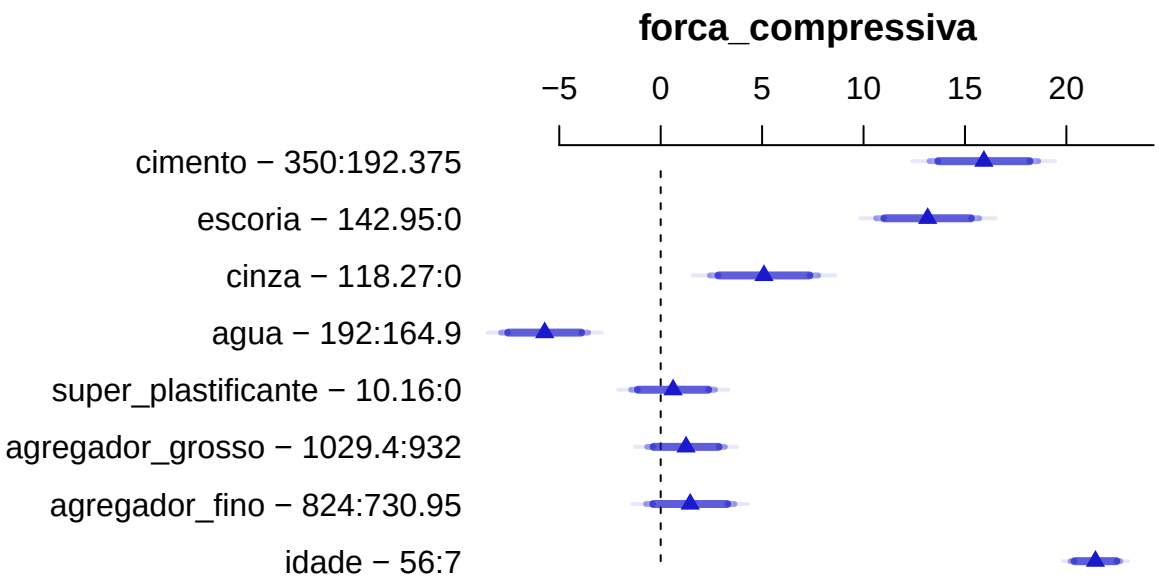
```
anova(mod1)
```

```
##                 Analysis of Variance          Response: forca_compressiva
##
##  Factor              d.f. Partial SS MS          F      P
##  cimento               9   11749.926  1305.54734  44.23 <.0001
##   Nonlinear            8    1219.660   152.45748   5.17 <.0001
##  escoria               9    5853.919   650.43546  22.04 <.0001
##   Nonlinear            8    1458.715   182.33939   6.18 <.0001
##  cinza                 9    3198.296   355.36617  12.04 <.0001
##   Nonlinear            8    1980.667   247.58333   8.39 <.0001
##  agua                  9    3960.857   440.09525  14.91 <.0001
##   Nonlinear            8    2343.119   292.88987   9.92 <.0001
##  super_plastificante   9    2386.342   265.14913   8.98 <.0001
##   Nonlinear            8    2167.901   270.98759   9.18 <.0001
##  agregador_grosso      9    1119.014   124.33485   4.21 <.0001
##   Nonlinear            8    1095.178   136.89730   4.64 <.0001
##  agregador_fino        9    4896.897   544.09961  18.44 <.0001
##   Nonlinear            8    4852.831   606.60393  20.55 <.0001
##  idade                 5   95884.399 19176.87987 649.75 <.0001
##   Nonlinear            4   56813.305 14203.32630 481.24 <.0001
##  TOTAL NONLINEAR      60   82065.079  1367.75131  46.34 <.0001
##  REGRESSION           68  258809.942  3806.02856 128.96 <.0001
##  ERROR               961   28363.079    29.51413
```

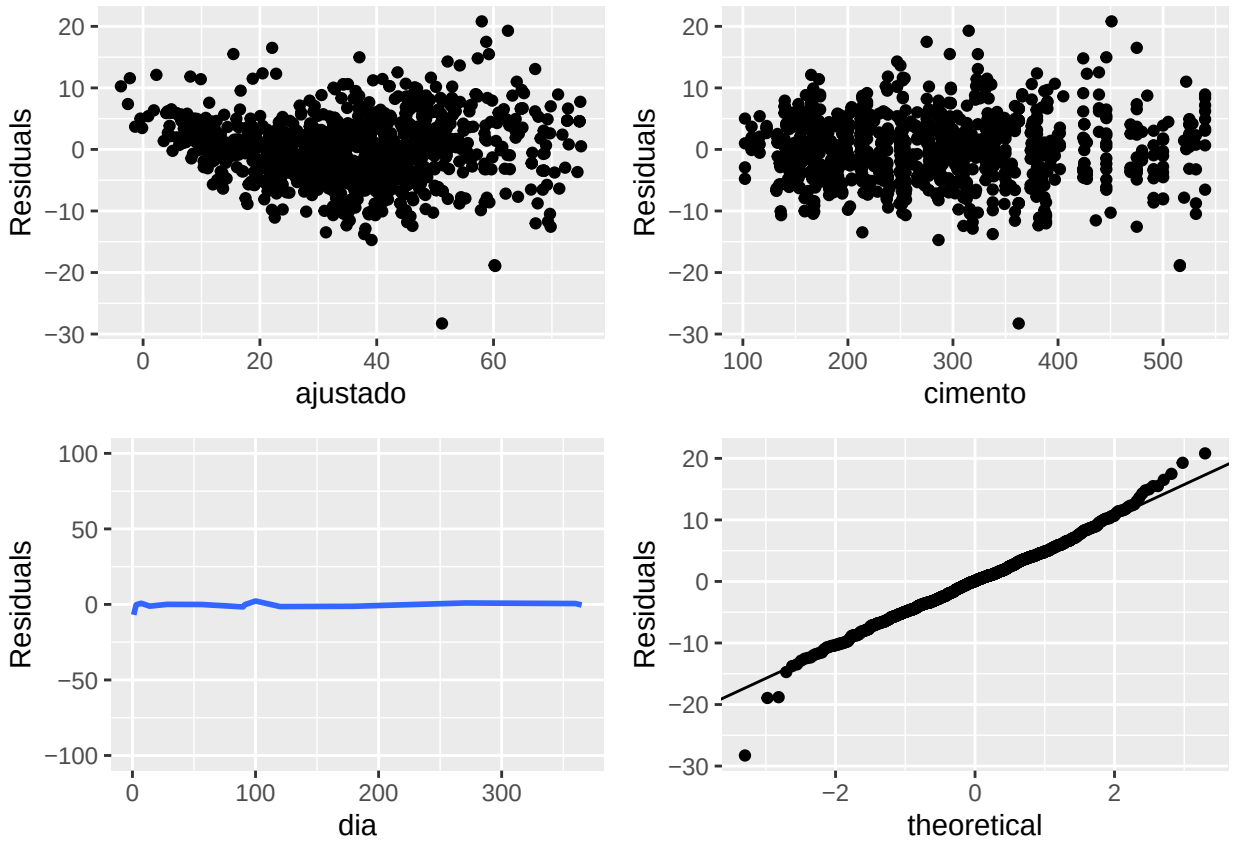
```
summary(mod1)
```

```
##              Effects              Response : forca_compressiva
##
##  Factor              Low    High    Diff.  Effect   S.E.    Lower 0.95
##  cimento             192.38  350.00 157.62 15.93400 1.36590 13.25300
##  escoria               0.00  142.95 142.95 13.16100 1.29670 10.61600
##  cinza                 0.00  118.27 118.27  5.09180 1.36260  2.41770
##  agua                164.90  192.00  27.10 -5.72000 1.09480 -7.86860
##  super_plastificante   0.00   10.16  10.16  0.61137 1.05470 -1.45840
##  agregador_grosso    932.00 1029.40  97.40  1.25140 0.97563 -0.66321
##  agregador_fino      730.95  824.00  93.05  1.45440 1.10970 -0.72326
##  idade                 7.00   56.00  49.00 21.43200 0.62491 20.20600
##  Upper 0.95
##  18.6140
##  15.7050
##   7.7658
##  -3.5715
##   2.6811
##   3.1660
##   3.6320
##  22.6590
```

```
plot(summary(mod1))
```




```
## Fazendo os plots dos resíduos
both <- data.frame(residuos = resid(mod1), ajustado = fitted (mod1))
both$cimento <- dados$cimento
both$dia <- dados$idade

yl <- ylab ('Residuals')
p1 <-  ggplot(both , aes( x = ajustado , y = residuos)) + geom_point() + yl
p2 <- ggplot(both , aes( x = cimento , y = residuos )) + geom_point() + yl
p3 <- ggplot(both , aes( x = dia , y = residuos)) + yl + ylim ( -100 , 100) +
stat_summary(fun.data = "mean_sdl", geom = 'smooth')
p4 <- ggplot(both , aes(sample = residuos)) + stat_qq() +
geom_abline(intercept = mean(resid(mod1)), slope = sd(resid(mod1))) + yl
gridExtra::grid.arrange( p1 , p2 , p3 , p4 , ncol =2)
```



Detecção de Outliers

```
## Carregando os pacotes necessários
require(car)
```

```
## Carregando pacotes exigidos: car
```

```
## Carregando pacotes exigidos: carData
```

```
## Outliers em X
X <- treino[, 1:8] #subset(treino, select = -c("forca_compressiva"))

H <- data.matrix(X) %*%
  solve((t(data.matrix(X)) %*%
           data.matrix(X))) %*%
  t(data.matrix(X))

hbar <- sum(diag(H))/nrow(X)

criterio_oulier <- 2*hbar

sum(diag(H) > criterio_oulier)
```

```
## [1] 55
```

```
sum(diag(H) > 0.5)
```

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

```
sum((diag(H) < criterio_oulier) & ((diag(H) > 0.2)))
```

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

```
sum(diag(H) < 0.2) / nrow(X)
```

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

```
outlierTest(final_model)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 276 3.269637          0.0011287       0.8149
```

Por esse método temos possíveis 55 outliers Entretando, utilizando critérios alternativos:

- h_ii > 0.5 outlier
- 0.2 < h_ii < 0.5 Moderad -> analisar
- h_ii < 0.2 não é outlier

Temos que todos os valores de diag(H) são inferiores a 0.2.

Agora, utilizando a função outliers do pacote car, temos o seguinte:

- Utilizando o mesmo modelo que foi utilizado para a análise de multicolinearidade, não temos nenhum ponto que possa ser considerado um outlier.

  Logo, como em 2 dos 3 testes não temos nenhum ponto identificado como outlier, vamos considerar que não temos nenhuma observação que deveríamos considerar outlier