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)
```

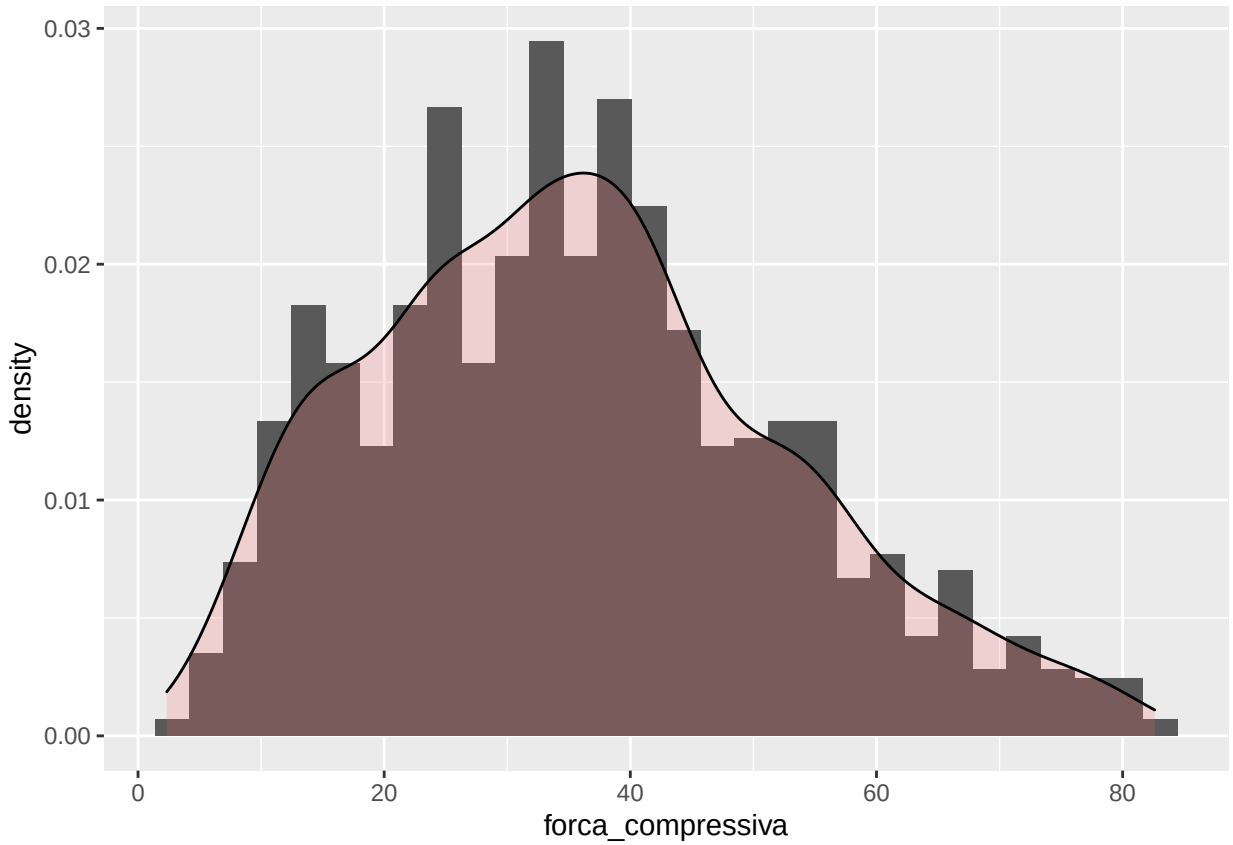
Descrição básica dos dados

```
## 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
```

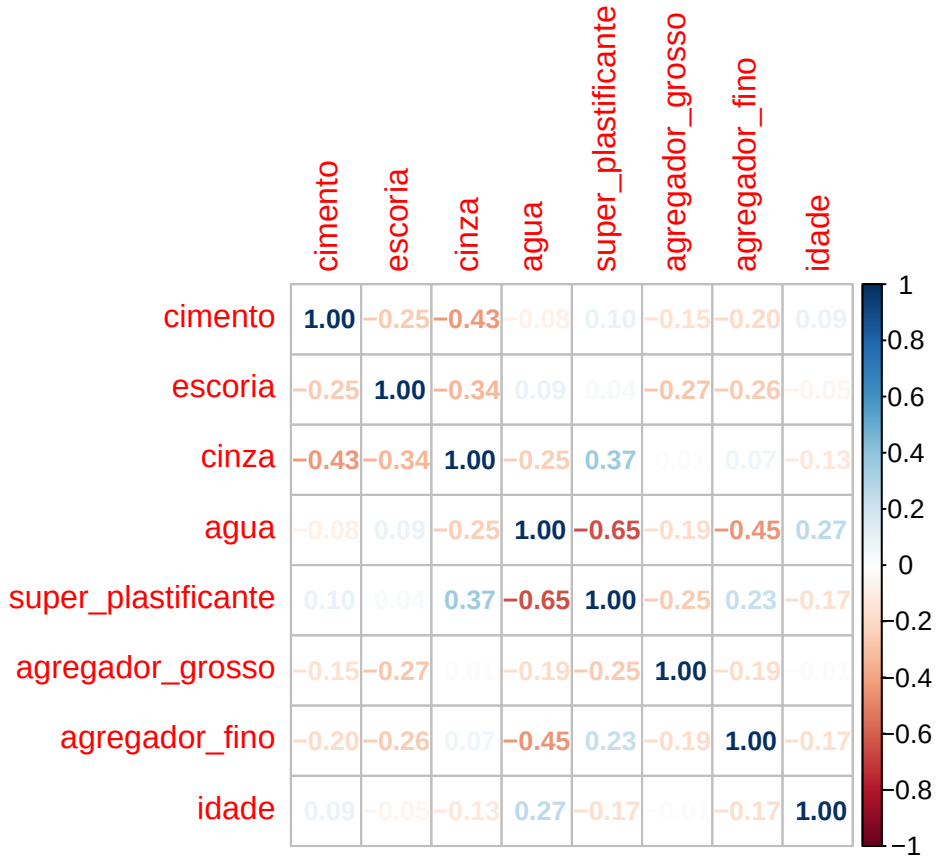
```r
require(GGally)
```

```
## 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)
```



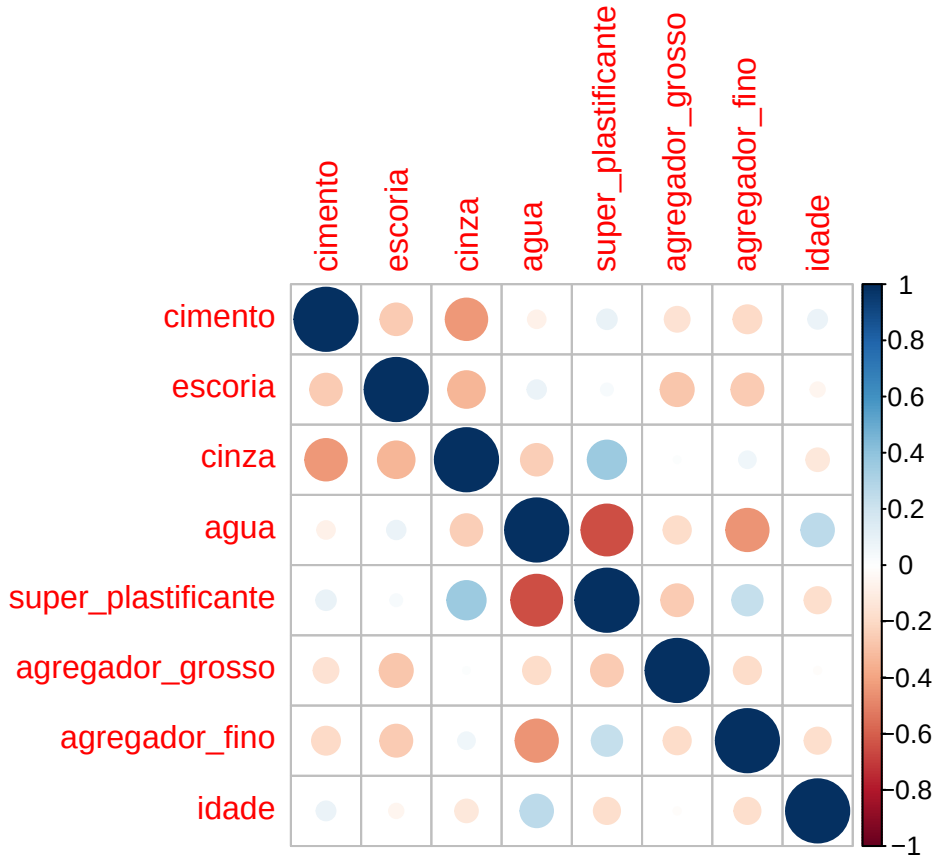

```
summary(M[upper.tri(M)])
```

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

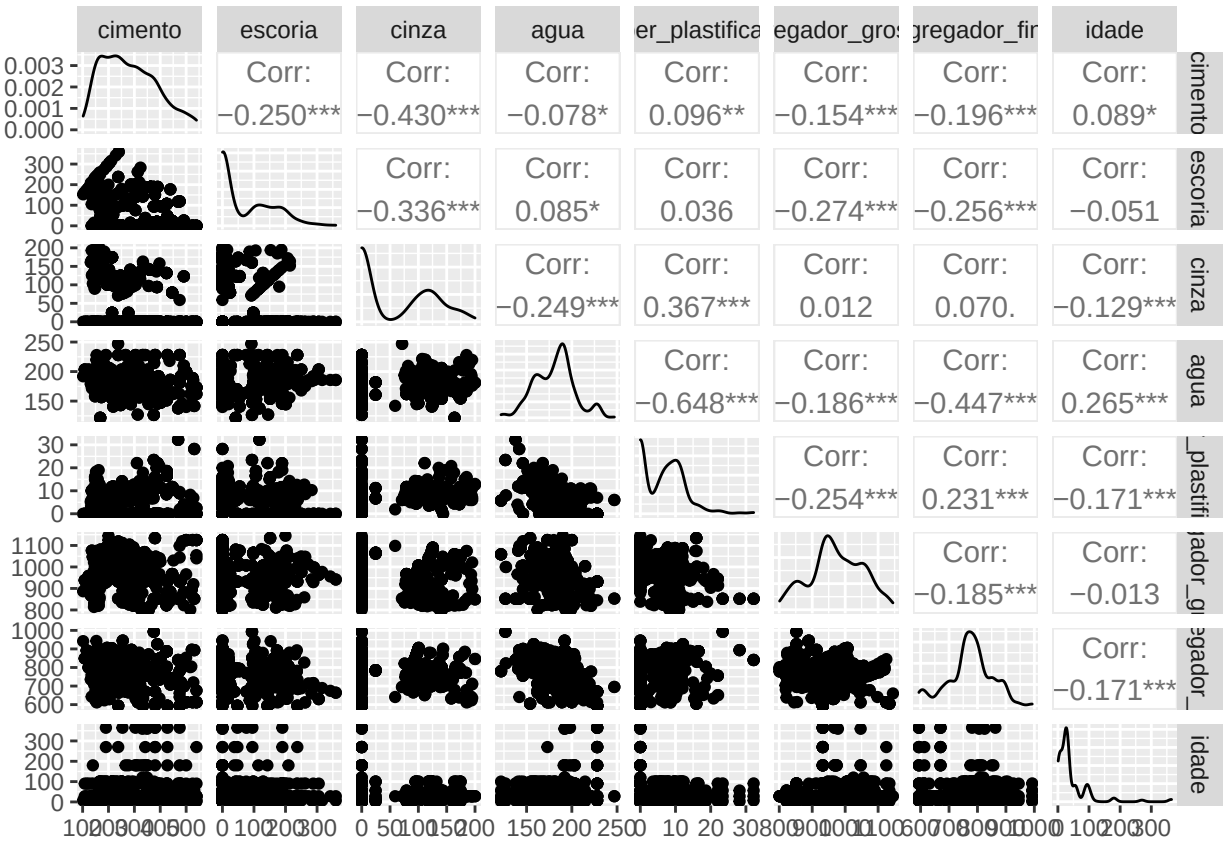
```
## Imprimindo as correlações na forma de circulos
M <- cor(numericas, use = 'complete.obs')
summary(M[upper.tri(M)])
```

```
##      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)
```

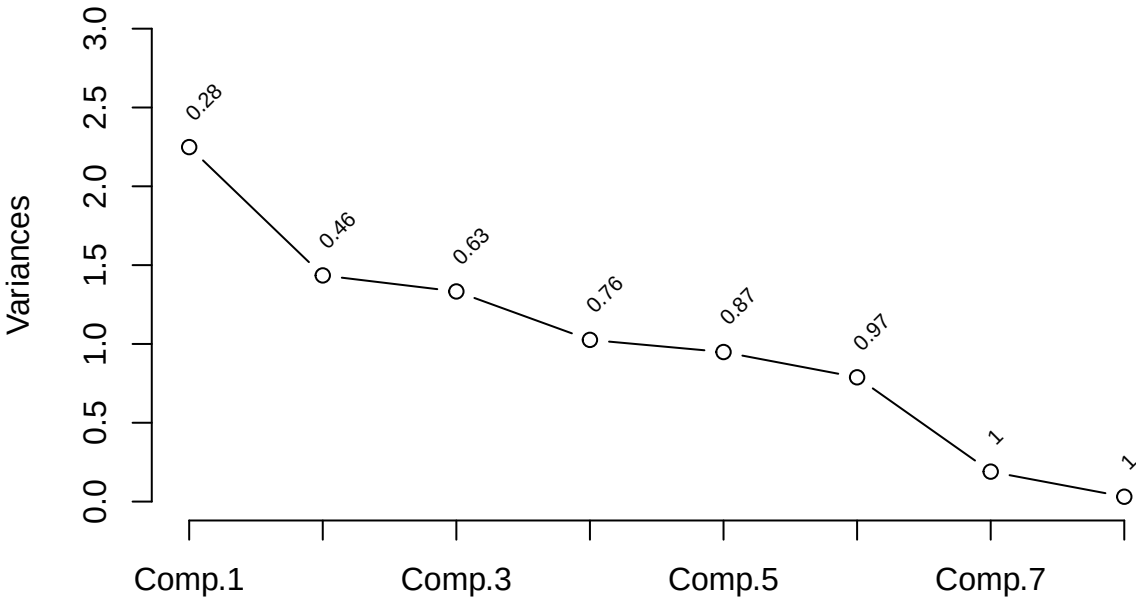
```
##
## Redundancy Analysis
##
## redun(formula = ~., data = numericas, r2 = 0.8, type = "adjusted")
##
## n: 722   p: 8    nk: 3
##
## Number of NAs:    0
##
## Transformation of target variables forced to be linear
##
```

```
## 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 poderiam ser facilmente preditas a partir das outras, portanto seriam excluídas. Entretando são variáveis fundamentais na produção do concreto e o $R^2 < 0,9$. Portanto não excluimos nenhuma variável.

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('')

}
```

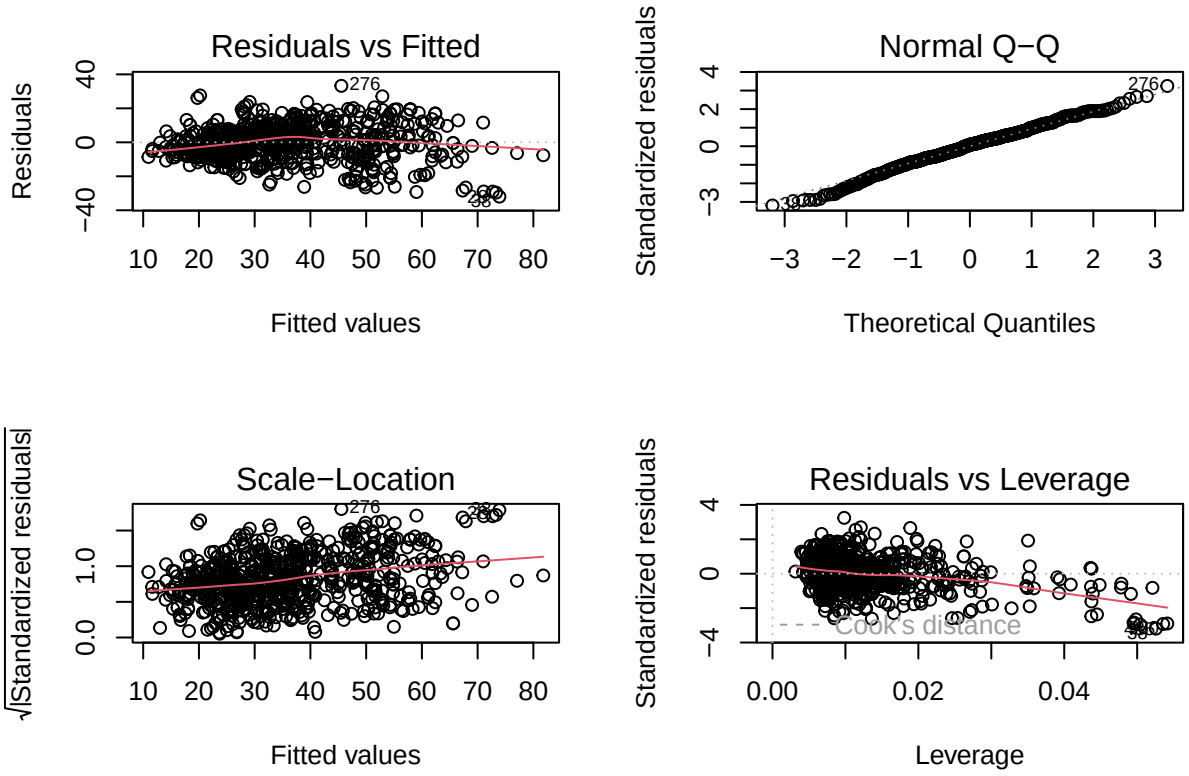
```
## log(forca_compressiva) ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.282 --  0.307"
## [1] "RMSE :0.359 -- 0.39"
## [1] "DATASET NORMALIZADO (TREINO -- TESTE): \n"
## [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)
```



Detecção de outliers e medidas influentes

Com base nos gráficos do modelo final podemos ainda nos aprofundar nas medidas influentes. Vemos que estão indicadas nos gráficos algumas observações influentes baseadas em *Leverage* e *Cook's distance*. Vamos então analisar alguns gráficos.

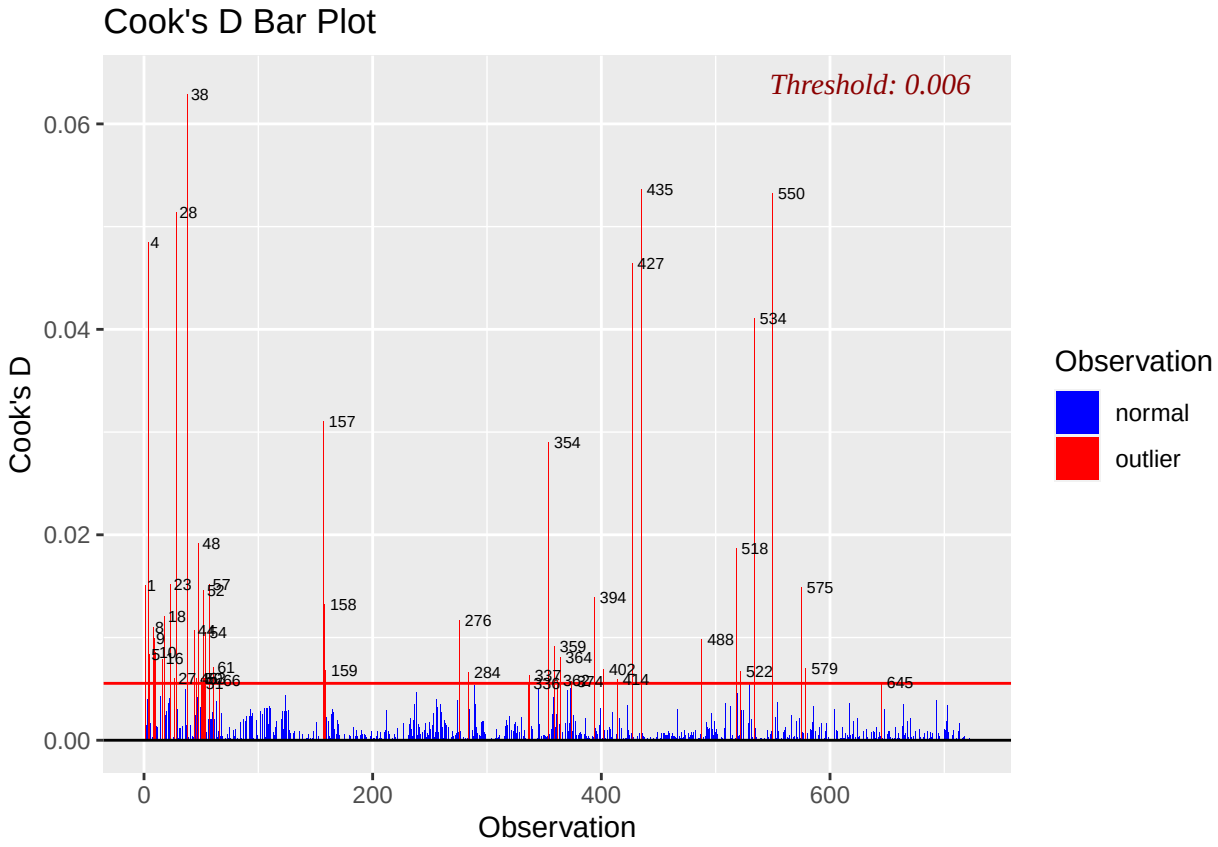
```
## Carregando os pacotes necessários
require(olsrr)
```

```
## Carregando pacotes exigidos: olsrr
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

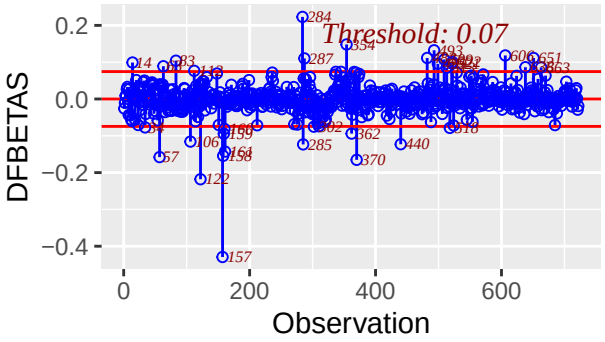
```
## Distância de Cook
ols_plot_cooksd_bar(final_model)
```



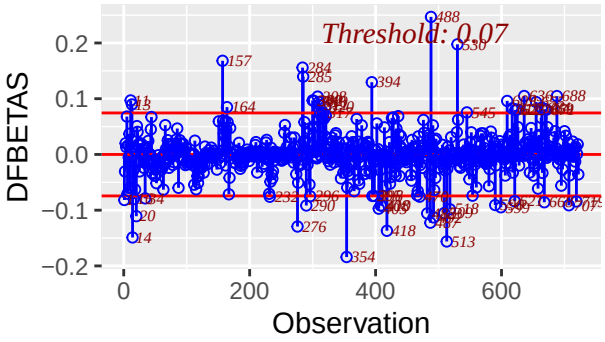
De acordo com esse critério há muitas observações que podem ser classificadas como outliers. Essas observações, nesse critério, tem influência forte sobre os valores ajustados.

```
## Dfbeta
ols_plot_dfbetas(model)
```

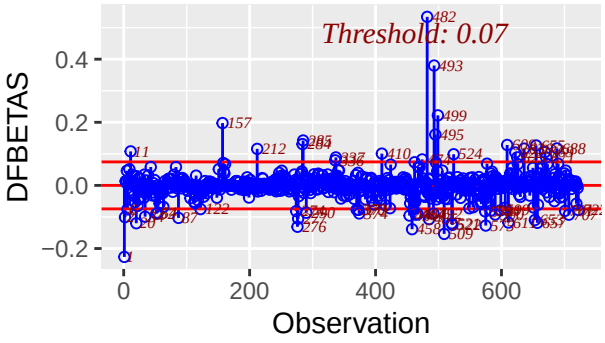
Influence Diagnostics for (Intercep



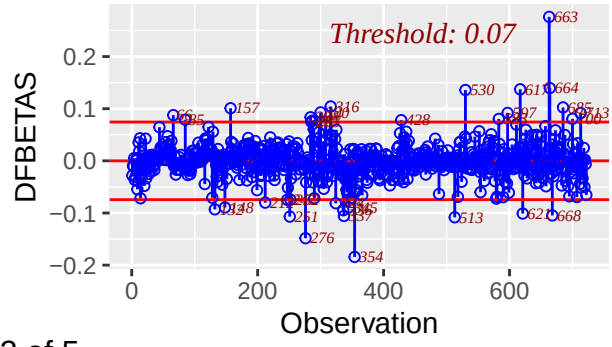
Influence Diagnostics for escoria



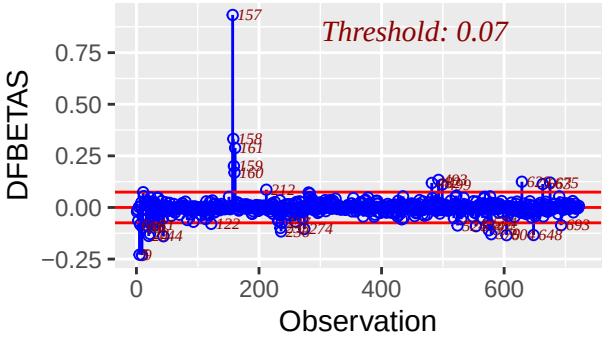
Influence Diagnostics for cimento



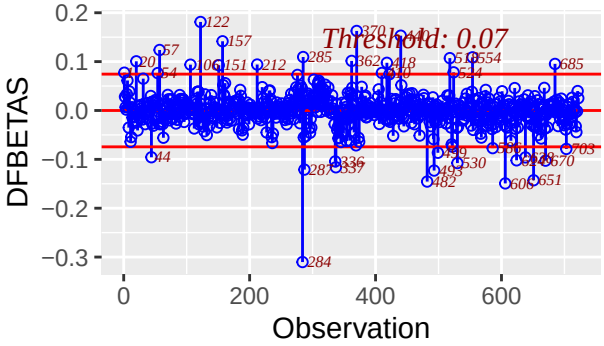
Influence Diagnostics for cinza



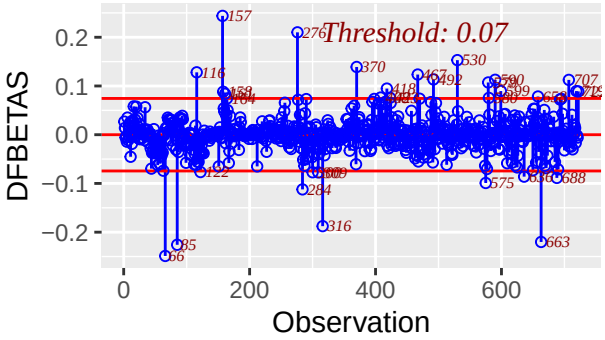
Influence Diagnostics for agua



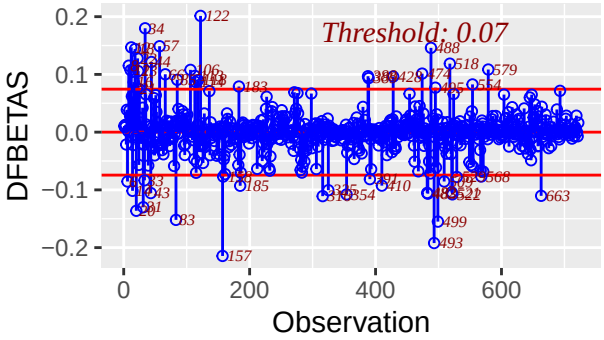
Influence Diagnostics for agregad



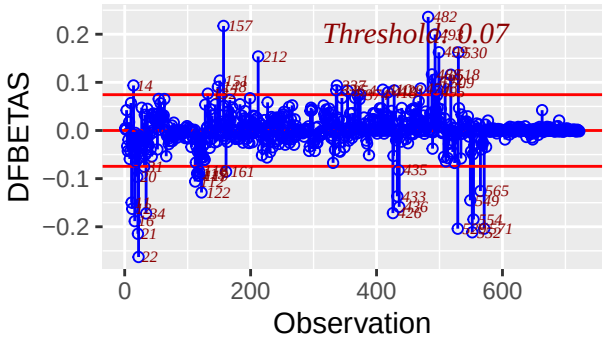
Influence Diagnostics for super_p



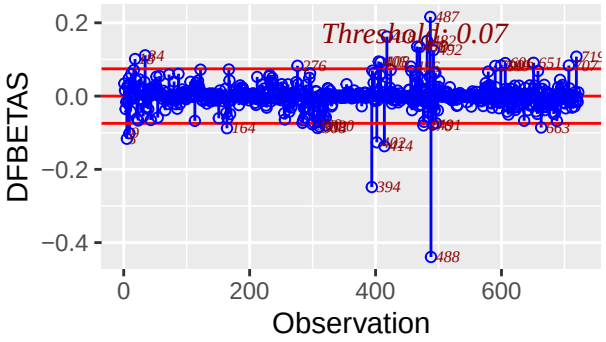
Influence Diagnostics for agregad



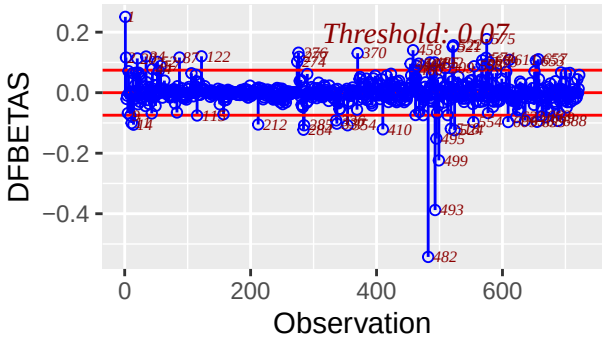
Influence Diagnostics for idade



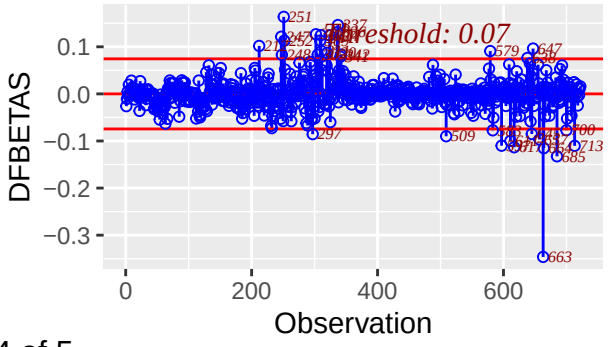
Influence Diagnostics for I(escoria



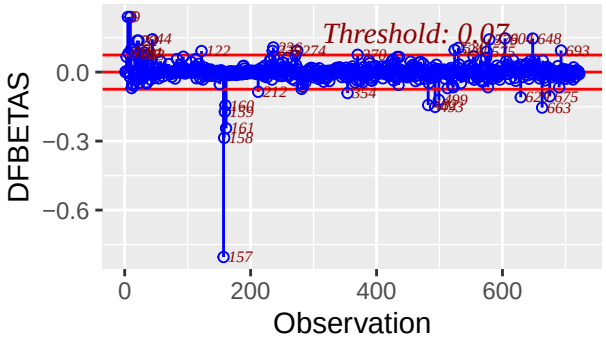
Influence Diagnostics for I(ciment



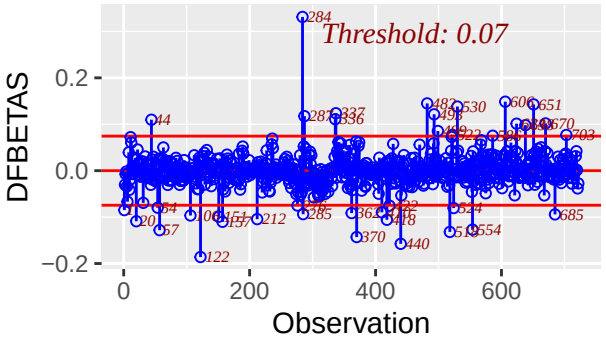
Influence Diagnostics for I(cinza^2



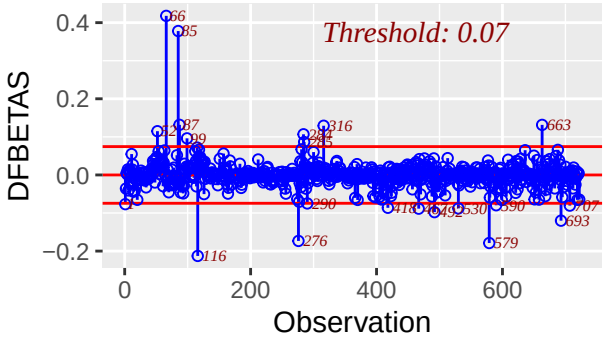
Influence Diagnostics for I(agua^2



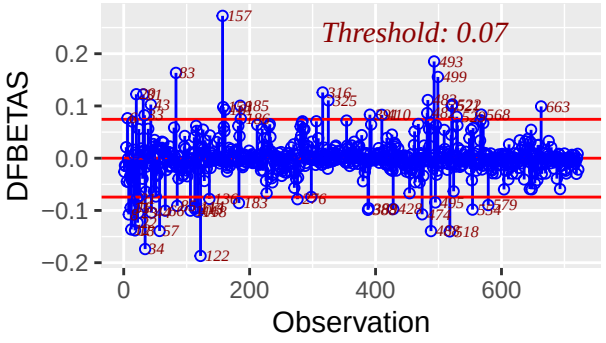
Influence Diagnostics for I(agrega

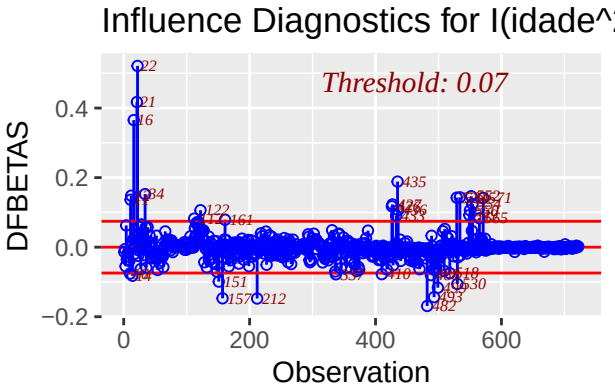


Influence Diagnostics for I(super_



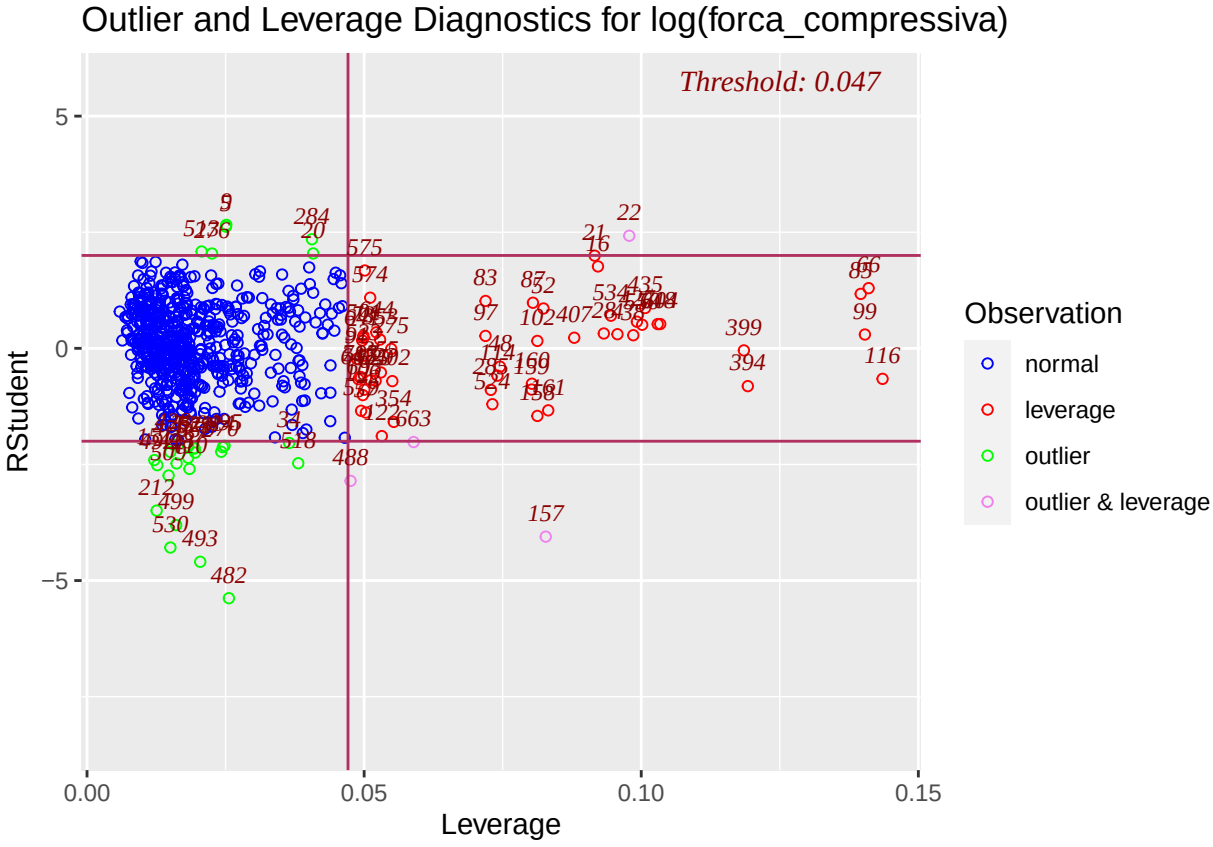
Influence Diagnostics for I(agrega





Com os dfbetas conseguimos entender quais medidas são influentes em um dado β_i. Aqui vemos que em todas as variáveis há medidas muito influentes.

```
## Comparando resíduo e leverage
ols_plot_resid_lev(model)
```



Aqui podemos observar que outliers e medidas influentes são bastante diferentes. Há medidas que são fortemente influentes e ainda são outliers, o que representa o pior caso. Entretanto simples outliers podem não ser tão prejudiciais para o modelo. Neste caso em específico decidimos não remover nenhuma observação devido a natureza não linear dos dados. No fim das contas estamos ajustando um modelo de regressão linear a um conjunto de dados ajustado originalmente para uma rede neural.

Relaxando a linearidade com splines

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

## 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, 5) + rcs(escoria, 5) + rcs(cinza, 5) +
                  rcs(agua, 5) + rcs(super_plastificante, 5) + rcs(agregador_grosso, 5) +
                  rcs(agregador_fino, 5) + rcs(idade, 5),
              data = treino, x = TRUE, y = TRUE)

## Avaliando o modelo
mod1
```

```
## Linear Regression Model
##
##  ols(formula = forca_compressiva ~ rcs(cimento, 5) + rcs(escoria,
##      5) + rcs(cinza, 5) + rcs(agua, 5) + rcs(super_plastificante,
##      5) + rcs(agregador_grosso, 5) + rcs(agregador_fino, 5) +
##      rcs(idade, 5), data = treino, x = TRUE, y = TRUE)
##
##                    Model Likelihood    Discrimination
##                          Ratio Test           Indexes
##  Obs     722    LR chi2    1466.52    R2       0.869
##  sigma6.1935    d.f.            31    R2 adj   0.863
##  d.f.    690    Pr(> chi2)  0.0000    g       17.601
##
##  Residuals
##
##       Min       1Q   Median       3Q      Max
##  -19.6499  -4.5567  -0.1728   4.0809  23.3571
##
##
##                          Coef      S.E.    t     Pr(>|t|)
##  Intercept               -157.9173 28.3133 -5.58 <0.0001
##  cimento                    0.1800  0.0221  8.13 <0.0001
##  cimento'                  -0.0278  0.2180 -0.13 0.8987
##  cimento''                 -0.3384  0.5692 -0.59 0.5523
##  cimento'''                 0.9235  0.5546  1.67 0.0963
##  escoria                    0.1418  0.0478  2.97 0.0031
##  escoria'                  -0.2739  0.9829 -0.28 0.7806
##  escoria''                  0.3670  1.3057  0.28 0.7787
##  escoria'''                -0.3129  0.5854 -0.53 0.5932
##  cinza                      0.0375  0.0357  1.05 0.2944
##  cinza'                     0.0707  0.2052  0.34 0.7304
##  cinza''                    0.3612  1.2499  0.29 0.7727
##  cinza'''                  -3.7028  3.6928 -1.00 0.3164
##  agua                       0.0720  0.0570  1.26 0.2068
##  agua'                     -1.8963  0.4408 -4.30 <0.0001
```

```
##  agua''                   6.3580  1.7076  3.72 0.0002
##  agua'''                 -7.4829  3.6190 -2.07 0.0390
##  super_plastificante      0.7100  0.3209  2.21 0.0272
##  super_plastificante'    -4.5418  5.0646 -0.90 0.3702
##  super_plastificante''    6.8835 11.8234  0.58 0.5606
##  super_plastificante'''  -1.5677 12.3325 -0.13 0.8989
##  agregador_grosso         0.0422  0.0186  2.27 0.0234
##  agregador_grosso'       -0.0123  0.0766 -0.16 0.8728
##  agregador_grosso''      -0.0634  0.6002 -0.11 0.9159
##  agregador_grosso'''      0.1612  0.8488  0.19 0.8494
##  agregador_fino           0.0894  0.0196  4.55 <0.0001
##  agregador_fino'         -0.0349  0.0472 -0.74 0.4601
##  agregador_fino''        -0.9008  0.8700 -1.04 0.3008
##  agregador_fino'''        2.5856  1.8264  1.42 0.1573
##  idade                    1.2607  0.0827 15.25 <0.0001
##  idade'                  -9.2802  1.2777 -7.26 <0.0001
##  idade''                 15.4344  2.3706  6.51 <0.0001
##
```

```
anova(mod1)
```

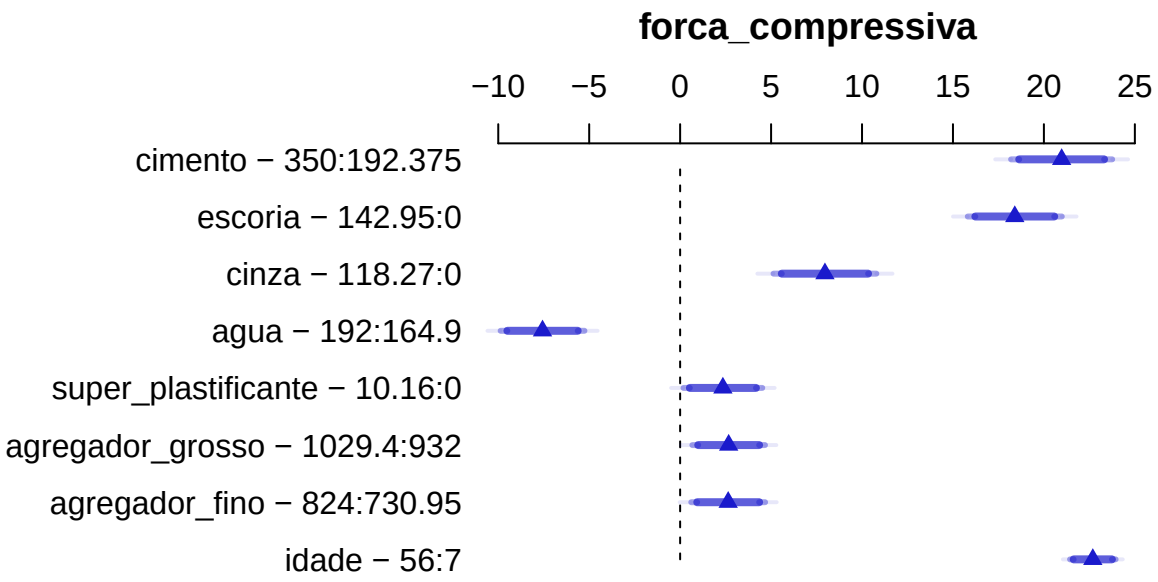
```
##                 Analysis of Variance          Response: forca_compressiva
##
##  Factor              d.f. Partial SS   MS          F      P
##  cimento               4   15562.69761  3890.67440 101.43 <.0001
##   Nonlinear            3     853.09202   284.36401   7.41 0.0001
##  escoria               4    7952.19369  1988.04842  51.83 <.0001
##   Nonlinear            3     461.66975   153.88992   4.01 0.0076
##  cinza                 4    1519.02447   379.75612   9.90 <.0001
##   Nonlinear            3     336.10157   112.03386   2.92 0.0334
##  agua                  4    2541.54767   635.38692  16.56 <.0001
##   Nonlinear            3    2241.27070   747.09023  19.48 <.0001
##  super_plastificante   4     770.35217   192.58804   5.02 0.0005
##   Nonlinear            3     712.51490   237.50497   6.19 0.0004
##  agregador_grosso      4     619.19390   154.79848   4.04 0.0030
##   Nonlinear            3      67.38951    22.46317   0.59 0.6246
##  agregador_fino        4    1782.66317   445.66579  11.62 <.0001
##   Nonlinear            3    1009.41467   336.47156   8.77 <.0001
##  idade                 3   70062.86549 23354.28850 608.84 <.0001
##   Nonlinear            2   38723.41824 19361.70912 504.75 <.0001
##  TOTAL NONLINEAR      23   48993.62883  2130.15778  55.53 <.0001
##  REGRESSION           31  175300.56120  5654.85681 147.42 <.0001
##  ERROR               690   26467.65408    38.35892
```

```
summary(mod1)
```

```
##              Effects              Response : forca_compressiva
##
##  Factor              Low    High    Diff.  Effect  S.E.    Lower 0.95
##  cimento             192.38  350.00 157.62 20.9750 1.41810 18.19100
##  escoria               0.00  142.95 142.95 18.3980 1.31860 15.80900
##  cinza                 0.00  118.27 118.27  7.9610 1.44230  5.12920
##  agua                164.90  192.00  27.10 -7.5657 1.17560 -9.87390
##  super_plastificante   0.00   10.16  10.16  2.3505 1.10520  0.18052
##  agregador_grosso    932.00 1029.40  97.40  2.6679 1.01860  0.66808
```

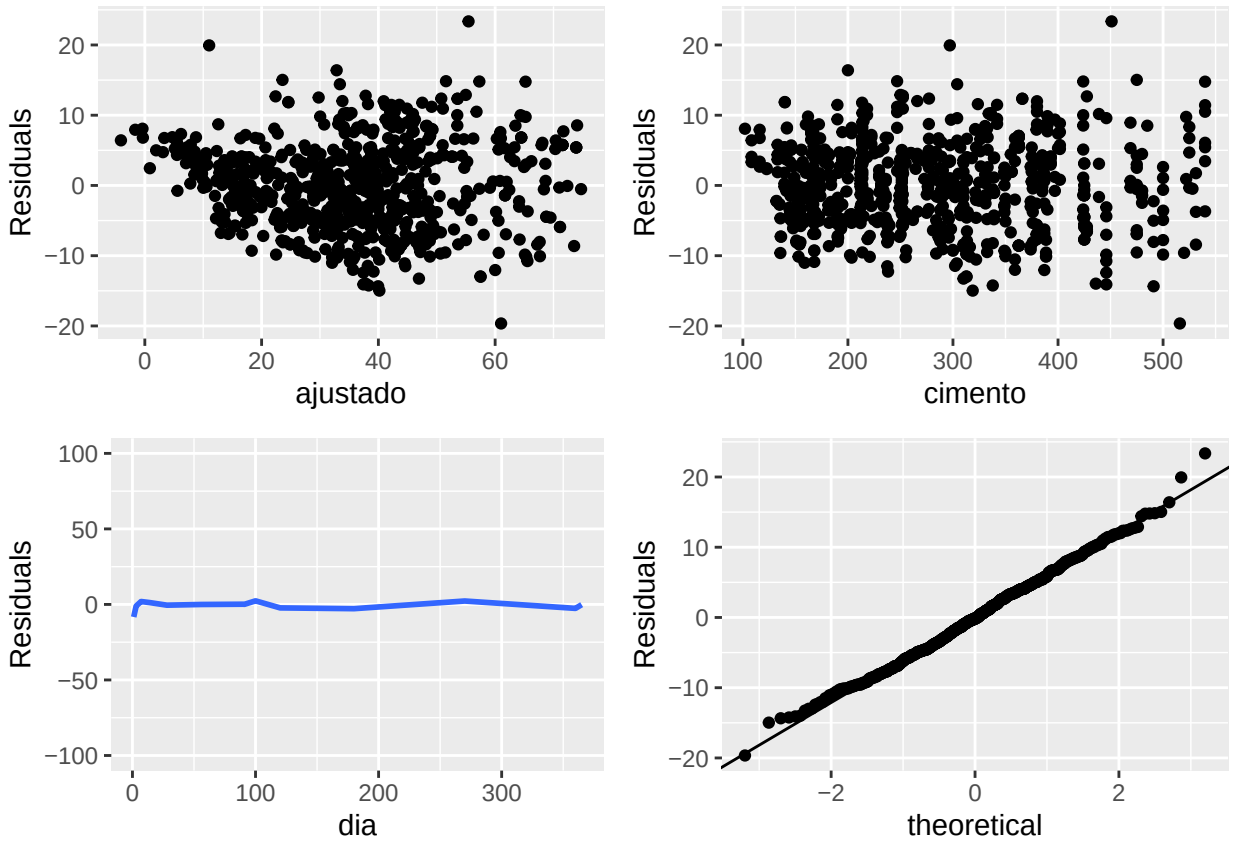
```
##  agregador_fino      730.95  824.00  93.05  2.6415 1.03620  0.60707
##  idade                 7.00   56.00  49.00 22.6890 0.63916 21.43400
##  Upper 0.95
##  23.7600
##  20.9870
##  10.7930
##  -5.2574
##   4.5204
##   4.6678
##   4.6760
##  23.9440
```

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



De acordo com o modelo é possível entender o impacto de cada variável na força compressiva. Nesse gráfico está registrada a variação interquartílica e o impacto na variável resposta. É fácil observar que a idade, cimento, cinza e escória, componentes fundamentais na produção do concreto, são as variáveis mais importantes.

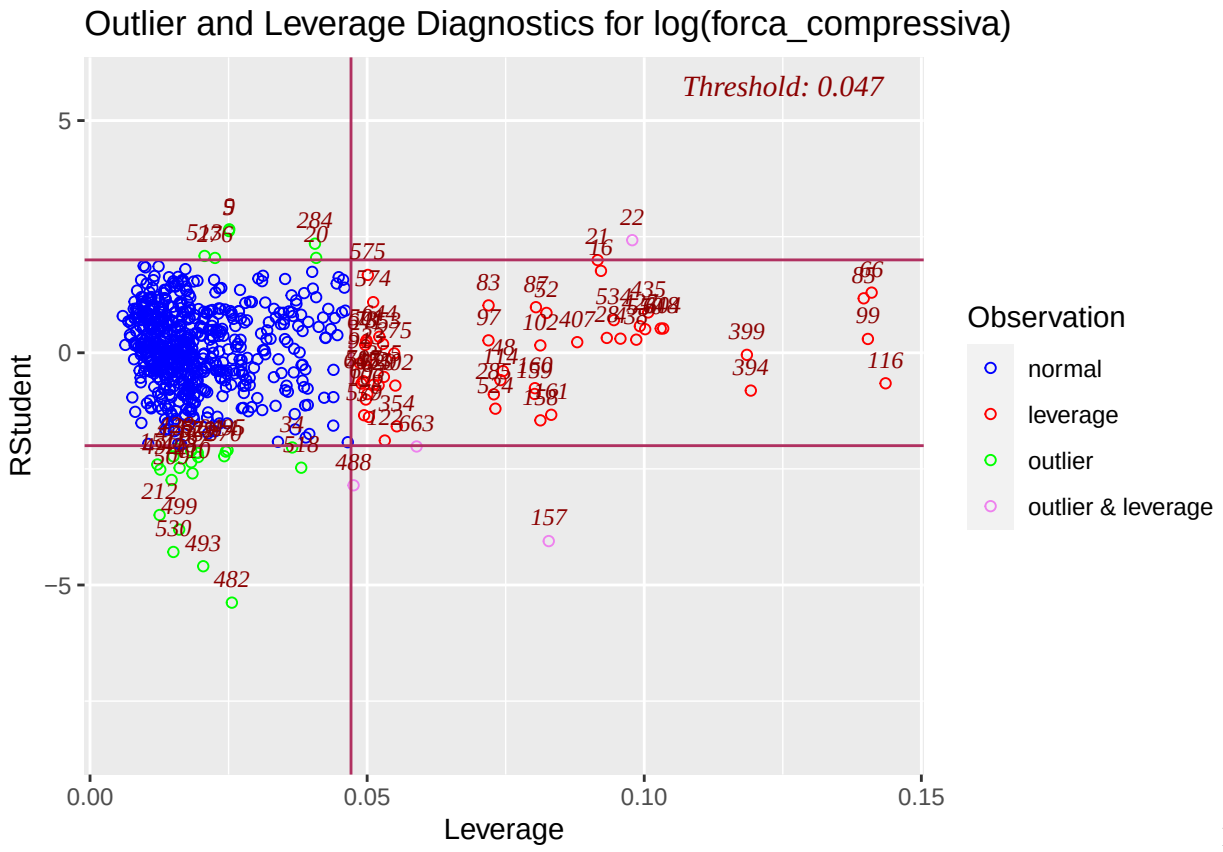
```
## Fazendo os plots dos resíduos
both <- data.frame(residuos = resid(mod1), ajustado = fitted (mod1))
both$cimento <- treino$cimento
both$dia <- treino$idade

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)
```



Aqui é importante observar que não há grandes padrões nos resíduos em comparação com as variáveis mais importantes como o número de dias e a quantidade de cimento. Entretanto, especialmente nos valores mais baixos de força do concreto vemos algumas distorções. É possível que serja necessário utilizar modelos com ainda mais linearidade ou apelar para modelos de aprendizado de máquina.

Ouliers e medidas influentes

```
mod1_plot <- mod1
class(mod1_plot) <- "lm"
## Comparando resíduo e leverage
ols_plot_resid_lev(model)
```



Da mesma forma que o modelo anterior, sem splines e com polinômios, há também medidas influentes e outliers.

Comparando os modelos com splines e com polinômios

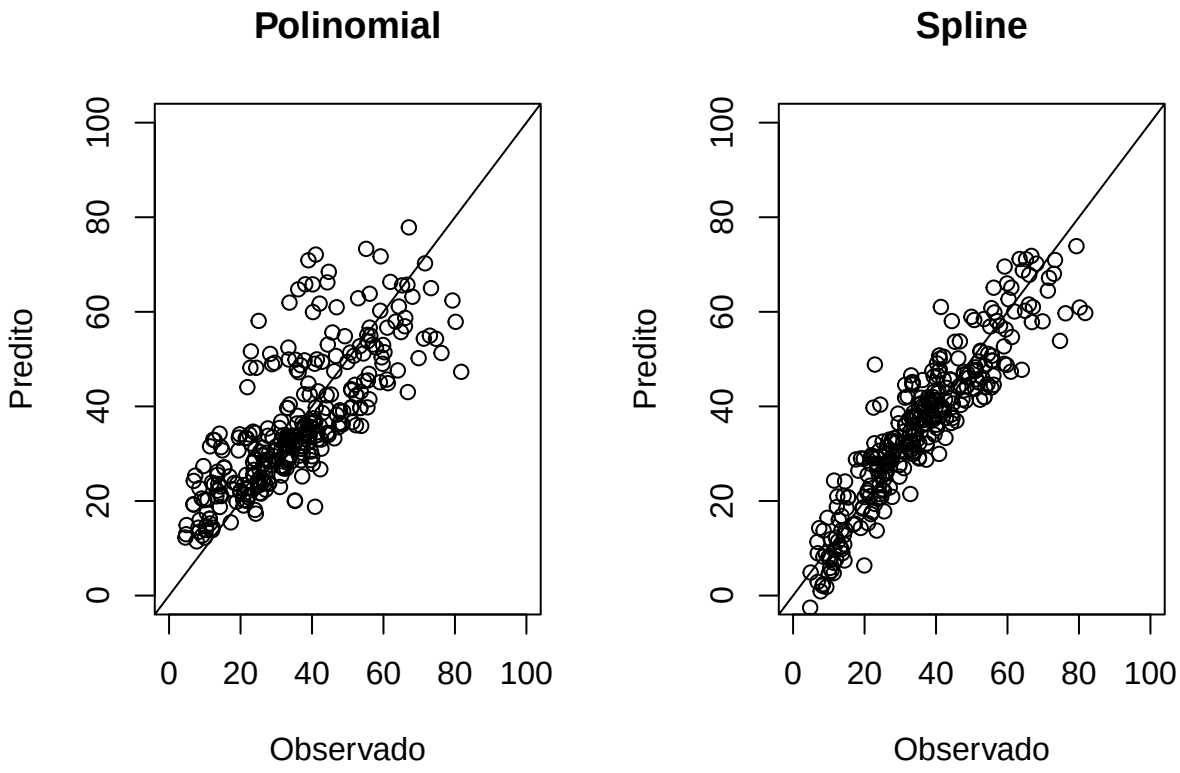
```
##predicao teste
  testePred_pol    <- predict(final_model, teste)
  testePred_spline <- predict(mod1, teste)

## RMSE
  rmse_pol <- round(RMSE(teste$forca_compressiva, testePred_pol), 3)
  rmse_spline <- round(RMSE(teste$forca_compressiva, testePred_spline), 3)

## MAE
  mae_pol <- round(MAE(teste$forca_compressiva, testePred_pol), 3)
  mae_spline <- round(MAE(teste$forca_compressiva, testePred_spline), 3)

## Data.frame com os resultados finais
resultados <- data.frame(MAE = c(mae_spline, mae_pol), RMSE = c(rmse_spline, rmse_pol), row.names = c("S

## Comparando os dois plots
par(mfrow = c(1,2))
plot(x = teste$forca_compressiva, y = testePred_pol, main = "Polinomial",
     ylab = "Predito",
     xlab = "Observado",
     xlim = c(0,100), ylim = c(0,100))
abline(a=0, b=1, "dashed")
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): NAs
```

```
## introduzidos por coerção
```

```
plot(x = teste$forca_compressiva, y = testePred_spline, main = "Spline",
     ylab = "Predito",
     xlab = "Observado",
     xlim = c(0,100), ylim = c(0,100))
abline(a=0, b=1, "dashed")
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): NAs
## introduzidos por coerção
```




```
## Mostrando o resultado
kable(resultados)
```

	MAE	RMSE
Spline	5.049	6.494
Polinômio	8.238	10.820

Conclusão

O modelo com splines é levemente superior. A vantagem se deve a forma mais inteligente de incluir a não-linearidade por meio dos splines que se comportam melhor que os polinômios, especialmente nos limites do range dos dados.