

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
```

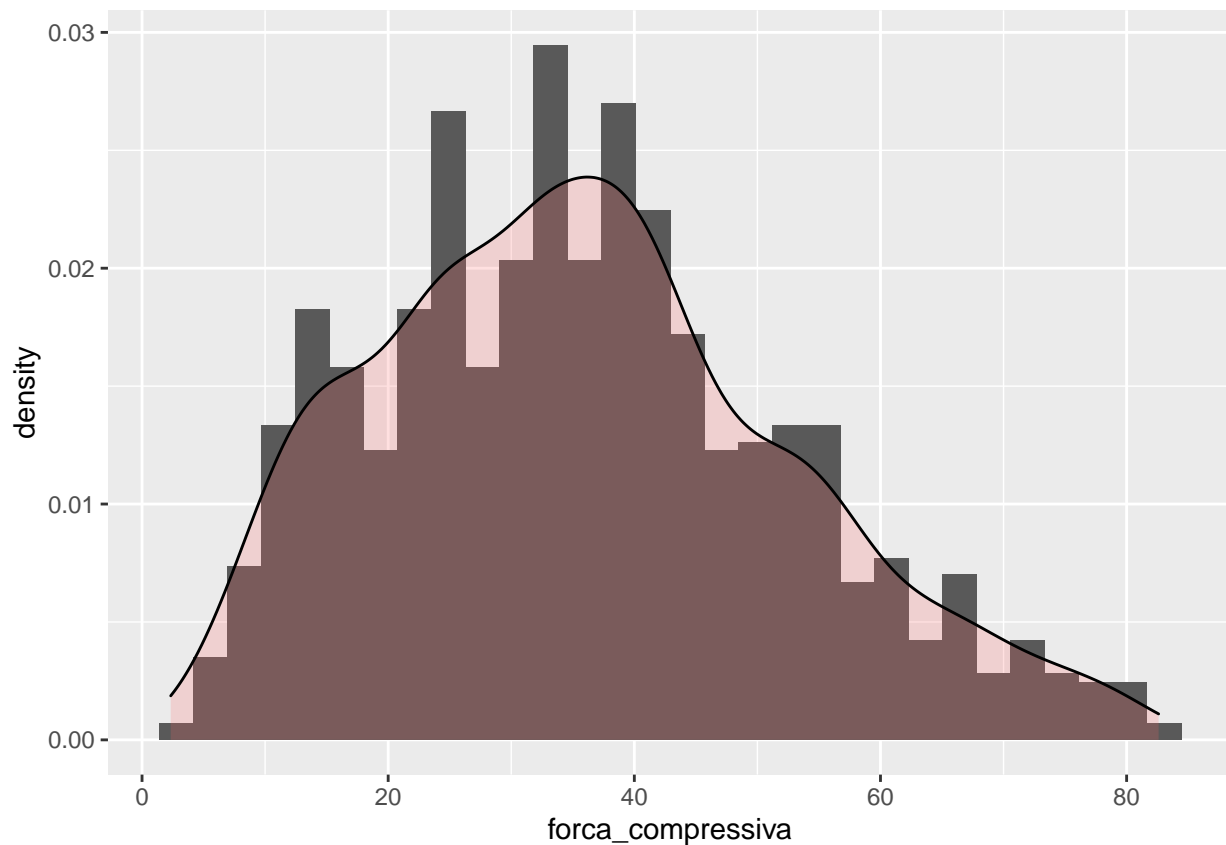
```
colnames(dados) <-
  c(
    "cimento",
    "escoria",
    "cinza",
    "agua",
    "super_plastificante",
    "agregador_grosso",
    "agregador_fino",
    "idade",
    "forca_compressiva"
  )

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)
- Água (kg/m^3)
- Super plastificante (kg/m^3)
- Agregado grosso (kg/m^3)
- Agregado 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
```

```

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

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.numeric(x))]
numericas <- treino[, Ind_numericas]
categoricas <- treino[, Ind_categoricas]

```

Redução de dimensionalidade

Estrutura de correlações

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

```

## Adicionando pacote corrplot
require(corrplot)

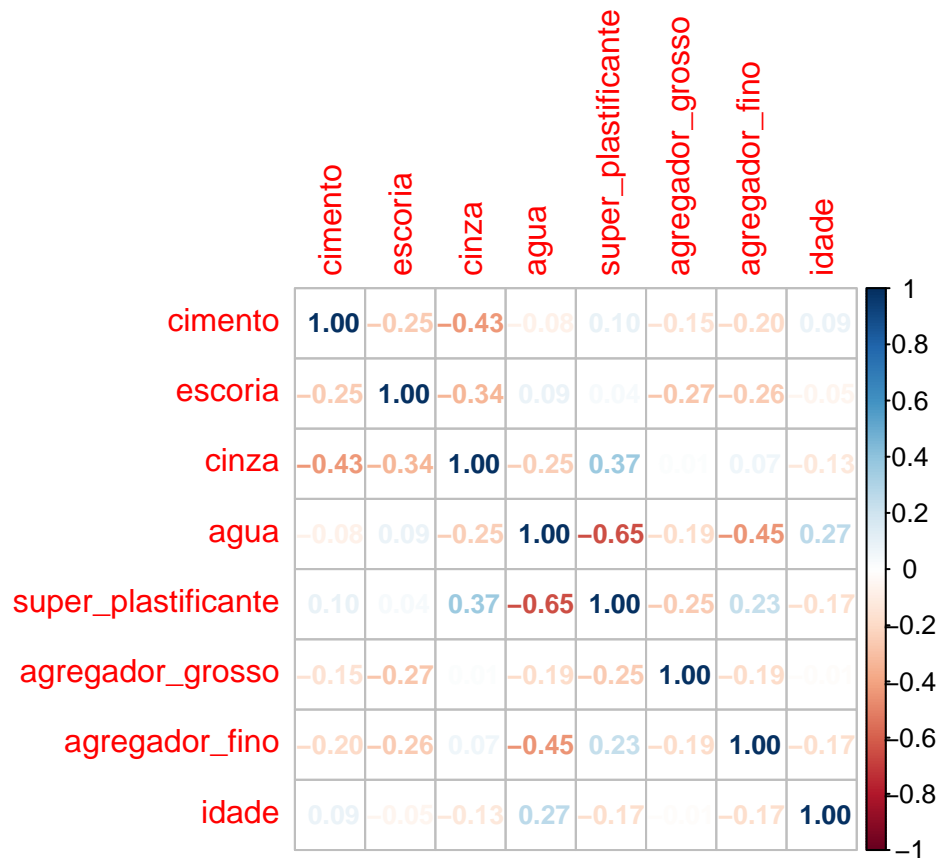
## Carregando pacotes exigidos: corrplot
## corrplot 0.92 loaded

require(GGally)

## Carregando pacotes exigidos: GGally
## Carregando pacotes exigidos: ggplot2
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg      ggplot2

## 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
```

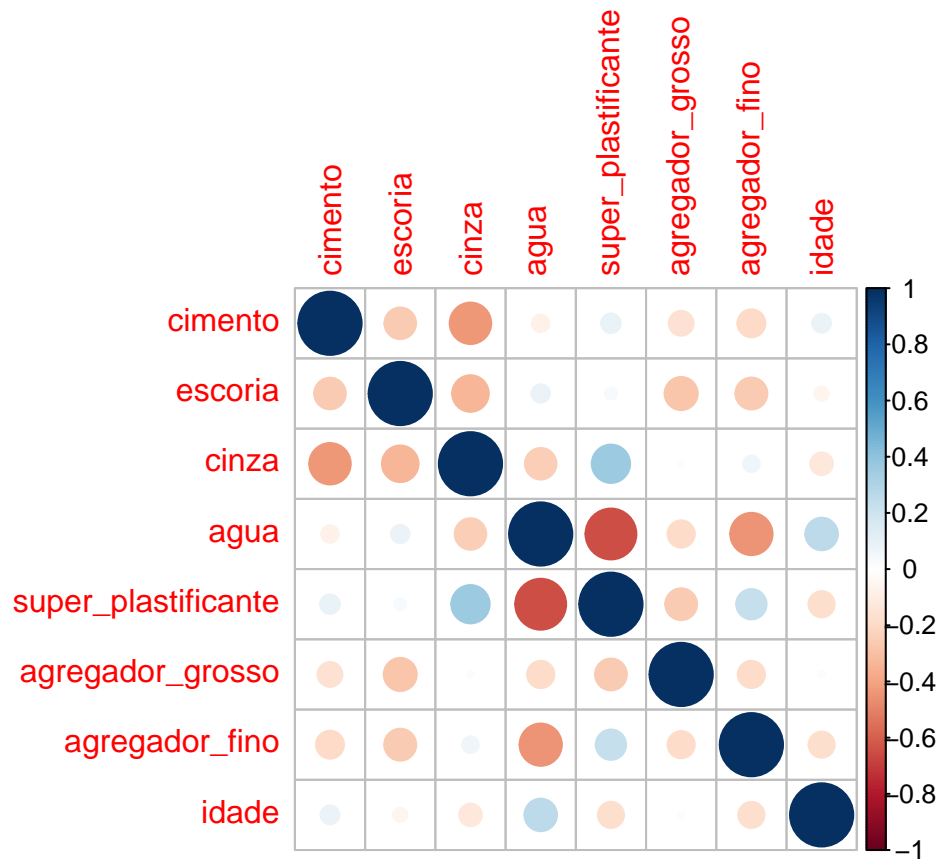
```
## Imprimindo as correlações na forma de círculos
```

```
M <- cor(numericas, use = 'complete.obs')
```

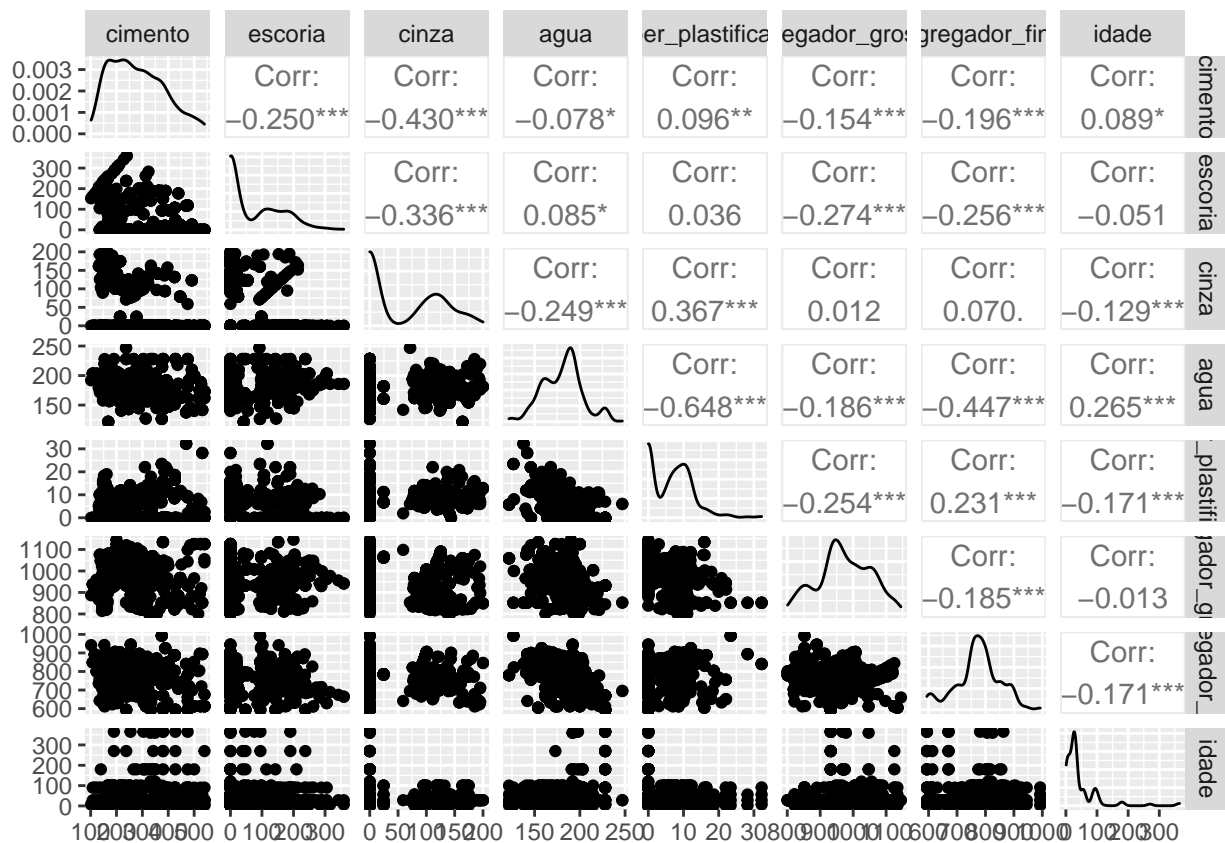
```
summary(M[upper.tri(M)])
```

```
##      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 tivéssemos, provavelmente não poderíamos remover covariáveis da análise, entretanto, vamos verificar mesmo assim.

Dependendo do objetivo da análise, se queremos acertar o valor da força compressiva 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
##
```

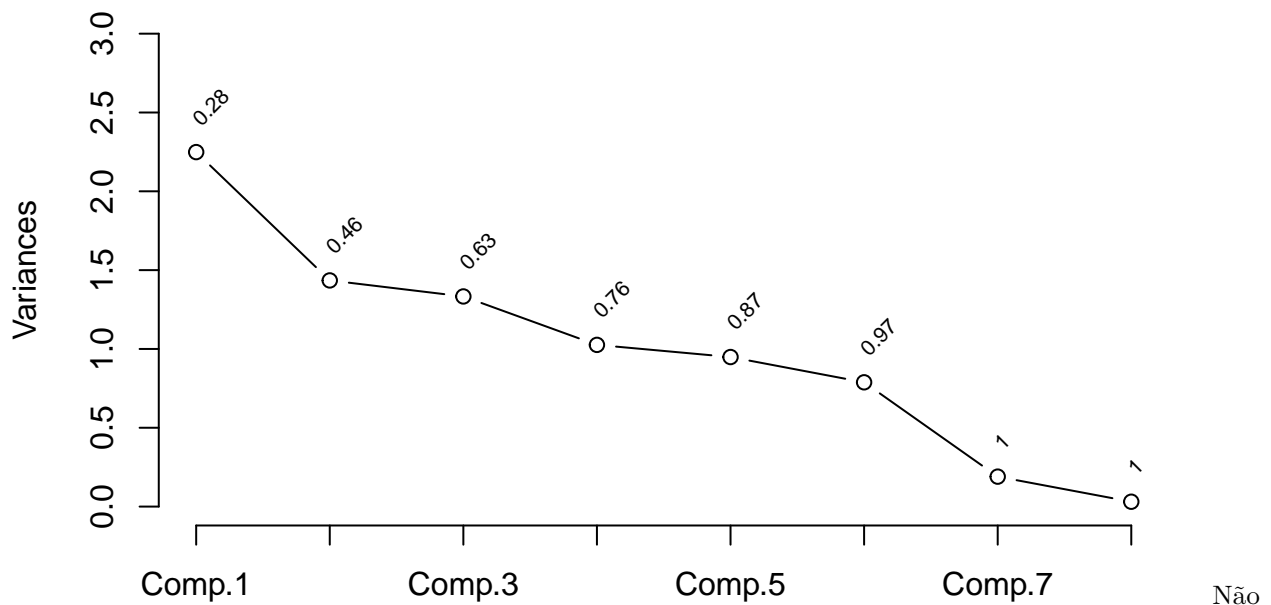
```
## R-squared cutoff: 0.8      Type: adjusted
##
## R^2 with which each variable can be predicted from all other variables:
##
##          cimento          escoria          cinza          agua
##          0.876          0.863          0.874          0.857
## super_plastificante  agregador_grosso  agregador_fino  idade
##          0.669          0.814          0.854          0.161
##
## Rerundant 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. Entretanto 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)
```

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
```

```

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_treino))
  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_treino))
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.

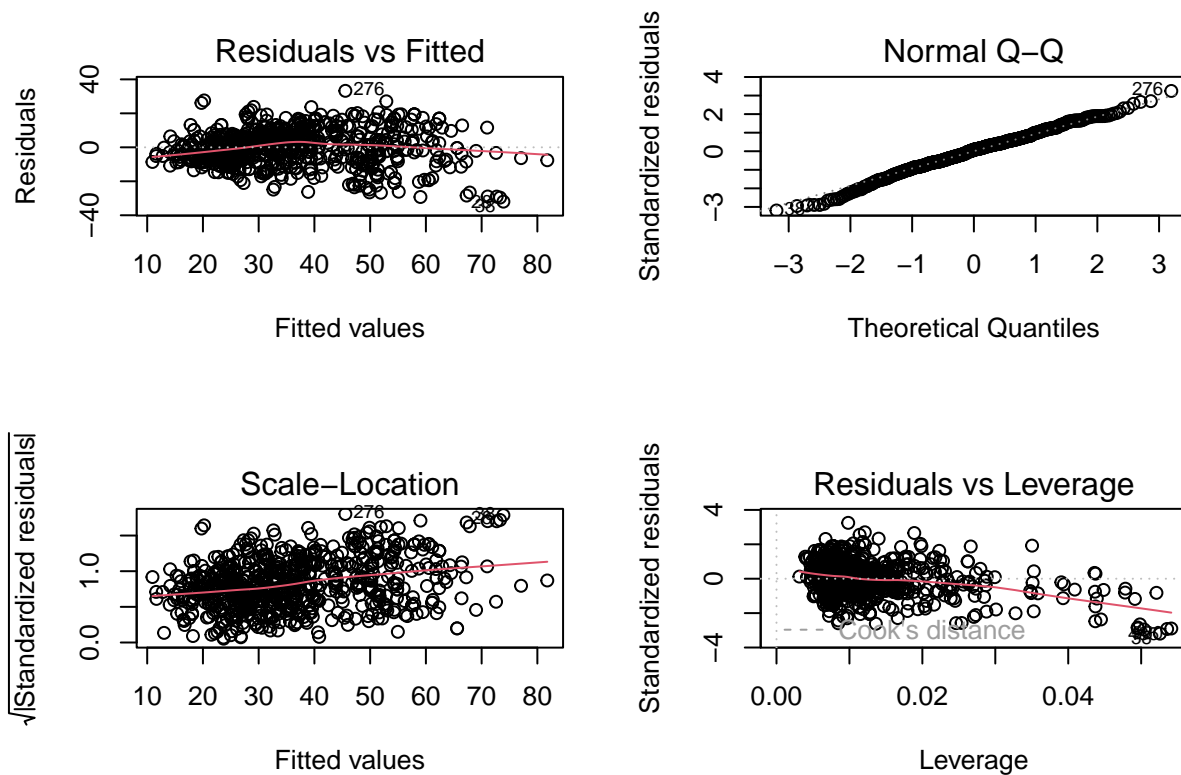
Análise de Resíduos

```
### 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)
```

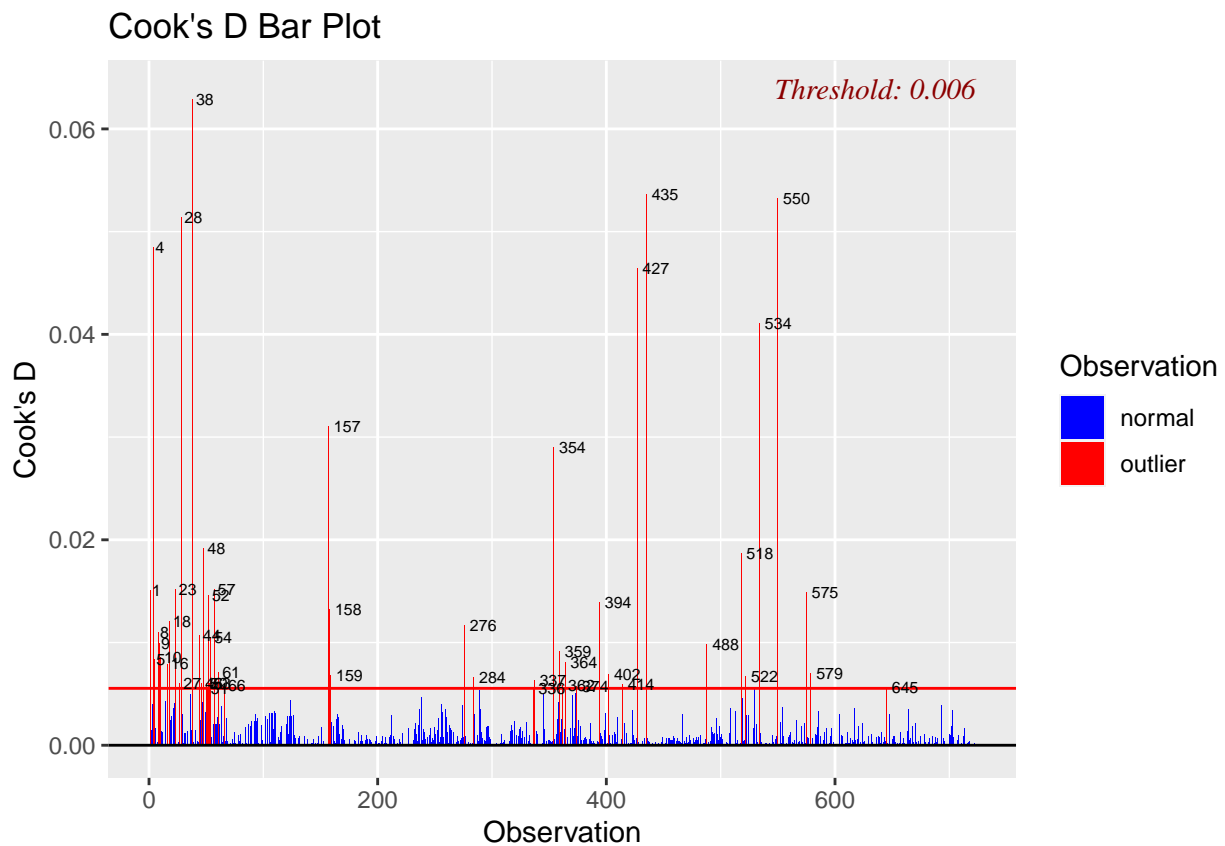


Deteção de outliers e medidas influentes

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)

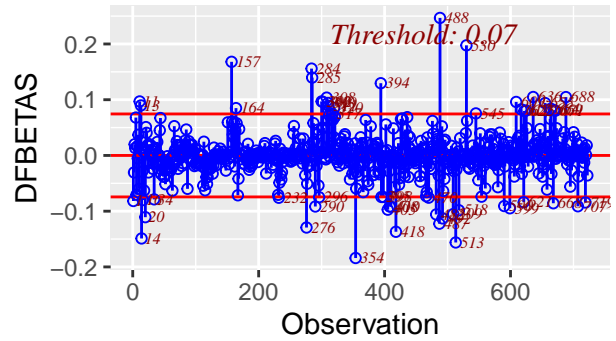
## Carregando pacotes exigidos: olsrr
##
## Attaching package: 'olsrr'
## The following object is masked from 'package:datasets':
##
##     rivers
## Distância de Cook
ols_plot_cooksd_bar(final_model)
```



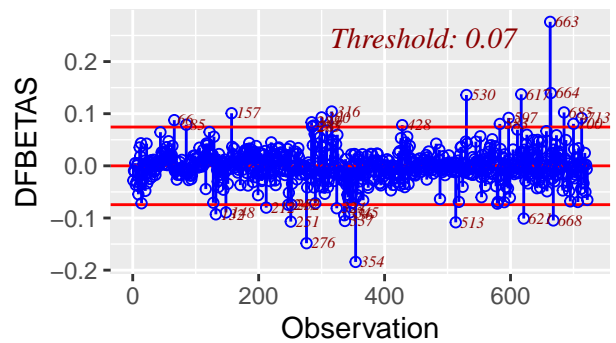
De acordo com esse critério há muitas observações que podem ser classificadas como outliers. Essas observações, nesse critério, tem influência forte sobre os valores ajustados.

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

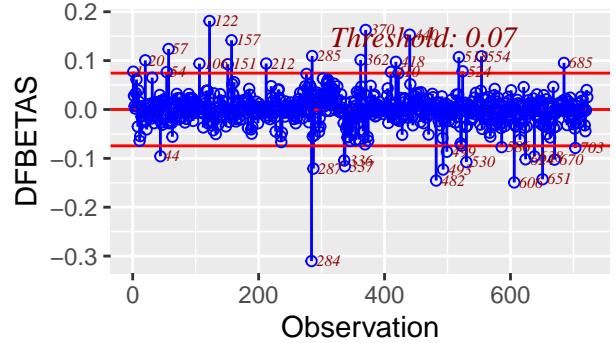
Influence Diagnostics for escoria



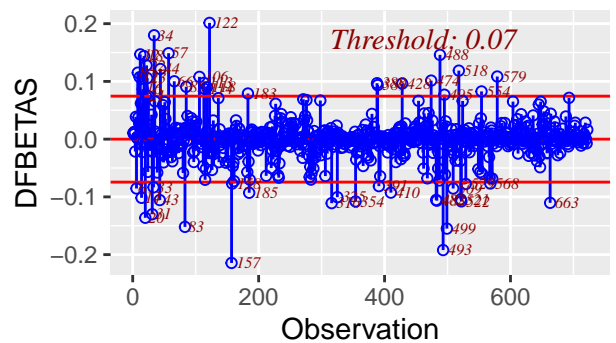
Influence Diagnostics for cinza

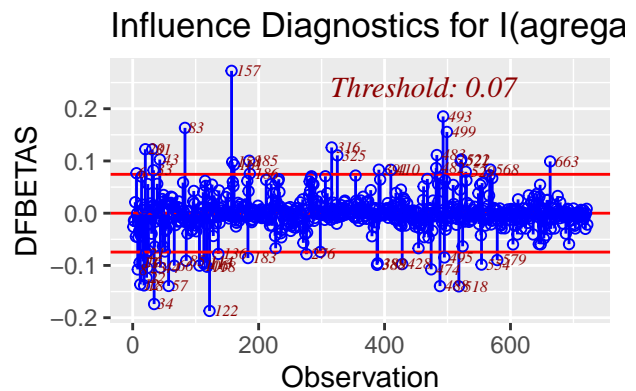
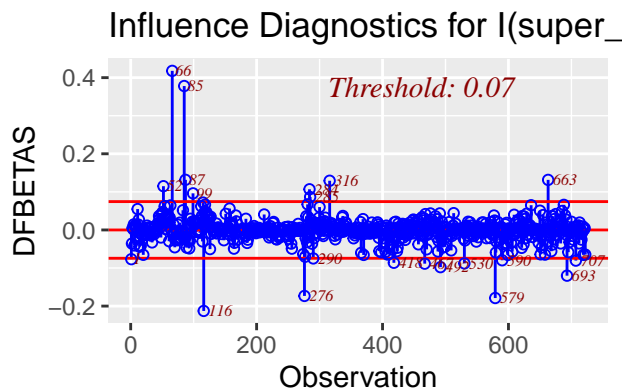
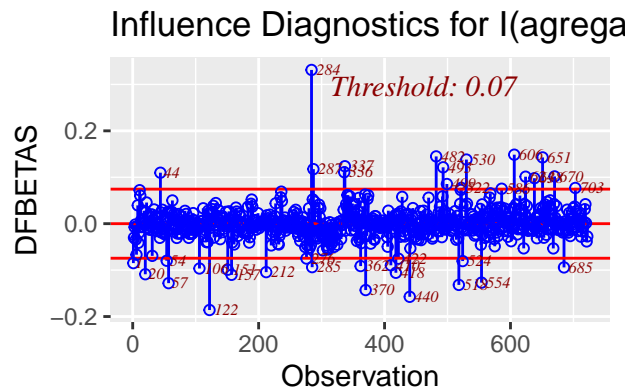
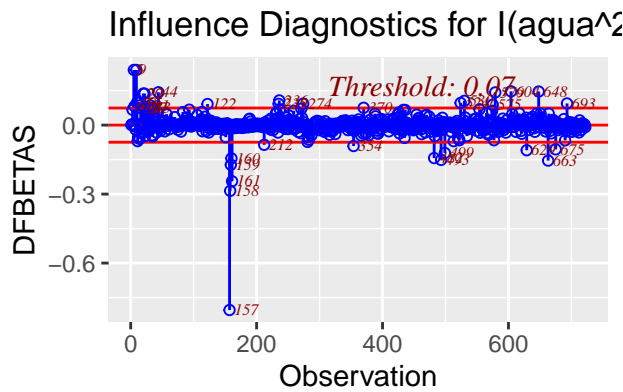
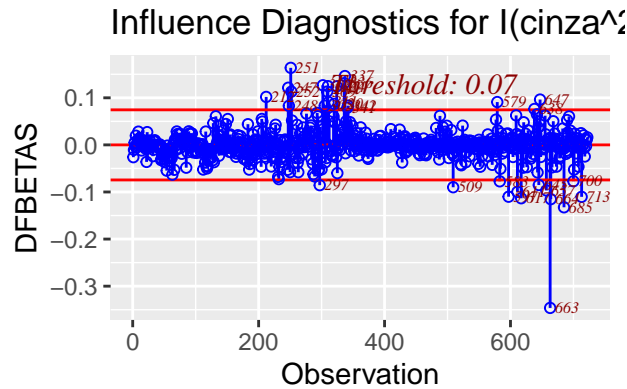
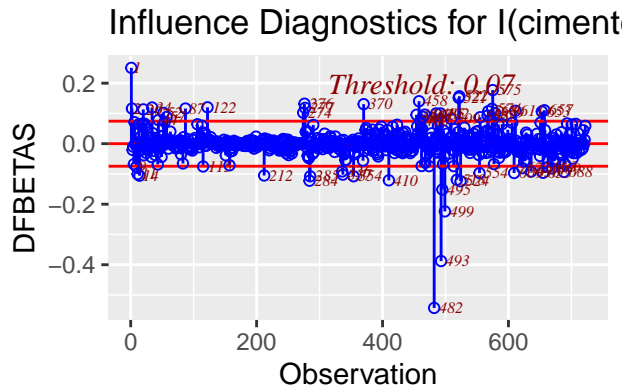
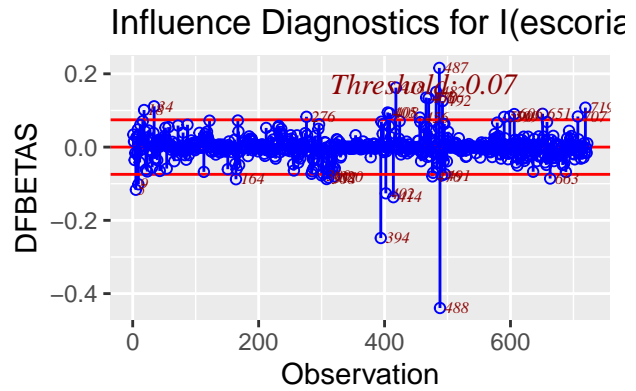
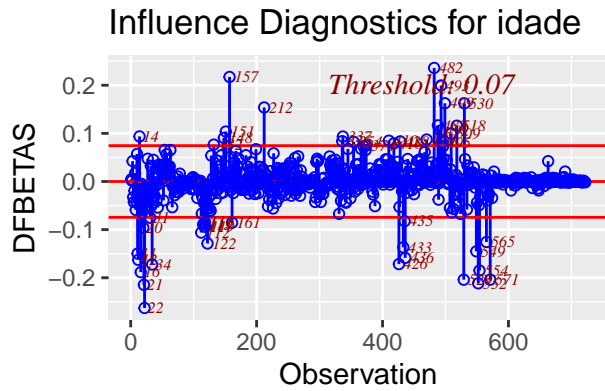


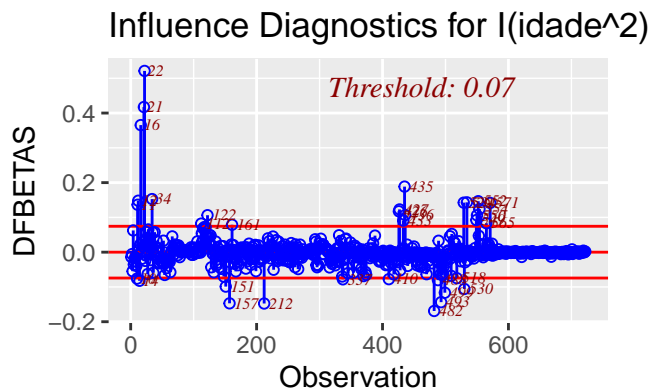
Influence Diagnostics for aggregated



Influence Diagnostics for aggregated

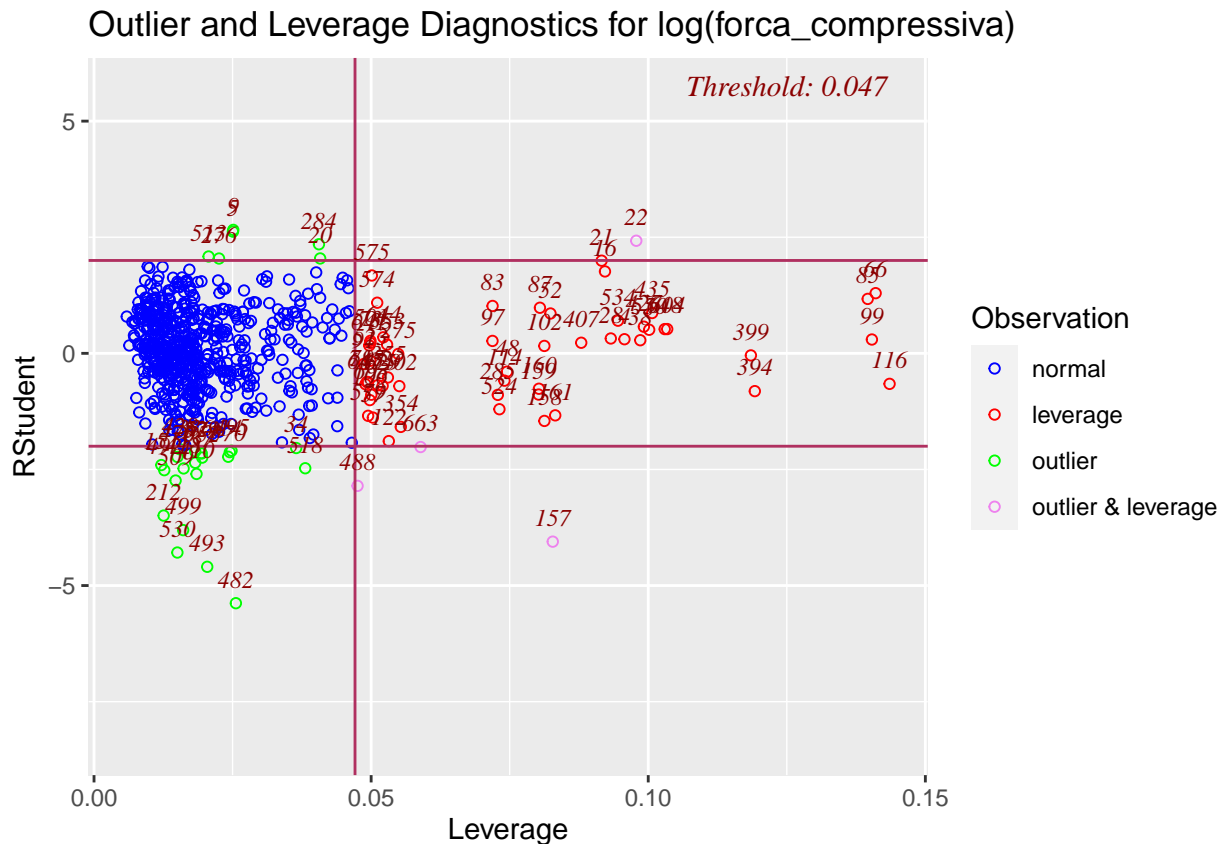






Com os dfbetas conseguimos entender quais medidas são influentes em um dado β_i . Aqui vemos que em todas as variáveis há medidas muito influentes.

```
## 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 = dados, 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 = dados, x = TRUE, y = TRUE)
##
```

		Model Likelihood	Discrimination			
		Ratio Test	Indexes			
##	Obs	1030	LR chi2	2055.19	R2	0.864
##	sigma6	2550	d.f.	31	R2 adj	0.860
##	d.f.	998	Pr(> chi2)	0.0000	g	17.552

```
## Residuals
```

	Min	1Q	Median	3Q	Max
##	-25.7228	-4.3806	-0.1421	4.1052	24.2067

	Coef	S.E.	t	Pr(> t)
## Intercept	-152.1856	24.2620	-6.27	<0.0001
## cimento	0.1564	0.0189	8.28	<0.0001
## cimento'	0.2006	0.1961	1.02	0.3065
## cimento''	-0.8583	0.4836	-1.77	0.0762
## cimento'''	1.4299	0.4907	2.91	0.0036
## escoria	0.1059	0.0400	2.65	0.0083
## escoria'	0.5251	0.8376	0.63	0.5308
## escoria''	-0.7266	1.1268	-0.64	0.5192
## escoria'''	0.2353	0.4971	0.47	0.6360
## cinza	0.0393	0.0312	1.26	0.2073
## cinza'	0.0712	0.1872	0.38	0.7036
## cinza''	0.5009	1.3243	0.38	0.7054
## cinza'''	-3.0202	2.8329	-1.07	0.2866
## agua	0.0091	0.0515	0.18	0.8592
## agua'	-1.5087	0.3412	-4.42	<0.0001

```
## agua'' 5.7260 1.4544 3.94 <0.0001
## agua''' -7.8777 3.6791 -2.14 0.0325
## super_plastificante 0.8849 0.2792 3.17 0.0016
## super_plastificante' -5.8662 4.4538 -1.32 0.1881
## super_plastificante'' 8.5470 9.7217 0.88 0.3795
## super_plastificante''' -1.4248 9.7733 -0.15 0.8841
## agregador_grosso 0.0517 0.0159 3.25 0.0012
## agregador_grosso' -0.0592 0.0723 -0.82 0.4128
## agregador_grosso'' 0.1704 0.5386 0.32 0.7517
## agregador_grosso''' -0.0340 0.7562 -0.04 0.9641
## agregador_fino 0.0899 0.0161 5.58 <0.0001
## agregador_fino' -0.0615 0.0394 -1.56 0.1189
## agregador_fino'' -0.2555 0.5862 -0.44 0.6630
## agregador_fino''' 1.1982 1.1371 1.05 0.2923
## idade 1.3950 0.0608 22.95 <0.0001
## idade' -42.4005 2.7670 -15.32 <0.0001
## idade'' 74.4366 5.0089 14.86 <0.0001
##
```

```
anova(mod1)
```

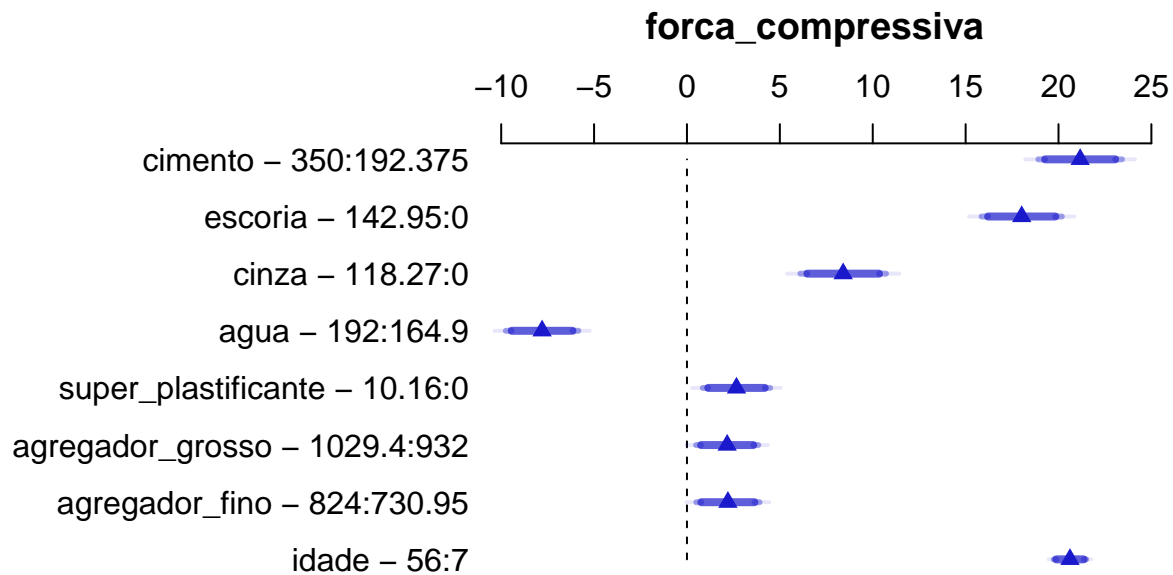
```
## Analysis of Variance Response: forca_compressiva
##
## Factor d.f. Partial SS MS F P
## cimento 4 21678.5689 5419.64223 138.52 <.0001
## Nonlinear 3 1134.3770 378.12566 9.66 <.0001
## escoria 4 11057.1588 2764.28971 70.65 <.0001
## Nonlinear 3 620.2136 206.73786 5.28 0.0013
## cinza 4 2469.8482 617.46206 15.78 <.0001
## Nonlinear 3 481.1788 160.39293 4.10 0.0066
## agua 4 4038.7409 1009.68523 25.81 <.0001
## Nonlinear 3 3373.2984 1124.43280 28.74 <.0001
## super_plastificante 4 1517.3750 379.34375 9.70 <.0001
## Nonlinear 3 1429.0482 476.34940 12.18 <.0001
## agregador_grosso 4 956.0368 239.00921 6.11 0.0001
## Nonlinear 3 133.6797 44.55990 1.14 0.3323
## agregador_fino 4 2129.8638 532.46595 13.61 <.0001
## Nonlinear 3 1095.8837 365.29457 9.34 <.0001
## idade 3 98853.8481 32951.28271 842.21 <.0001
## Nonlinear 2 55926.0048 27963.00241 714.71 <.0001
## TOTAL NONLINEAR 23 71381.7000 3103.55218 79.32 <.0001
## REGRESSION 31 248126.5717 8004.08296 204.58 <.0001
## ERROR 998 39046.4568 39.12471
```

```
summary(mod1)
```

```
## Effects Response : forca_compressiva
##
## Factor Low High Diff. Effect S.E. Lower 0.95
## cimento 192.38 350.00 157.62 21.1670 1.14710 18.91600
## escoria 0.00 142.95 142.95 18.0200 1.10150 15.85800
## cinza 0.00 118.27 118.27 8.4092 1.17510 6.10330
## agua 164.90 192.00 27.10 -7.7991 0.99565 -9.75290
## super_plastificante 0.00 10.16 10.16 2.6655 0.92488 0.85057
## agregador_grosso 932.00 1029.40 97.40 2.1641 0.85377 0.48869
```

```
## agregador_fino      730.95  824.00  93.05  2.2042 0.86856  0.49980
## idade               7.00   56.00  49.00 20.6180 0.44600 19.74300
## Upper 0.95
## 23.4180
## 20.1810
## 10.7150
## -5.8453
## 4.4804
## 3.8395
## 3.9086
## 21.4940
```

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

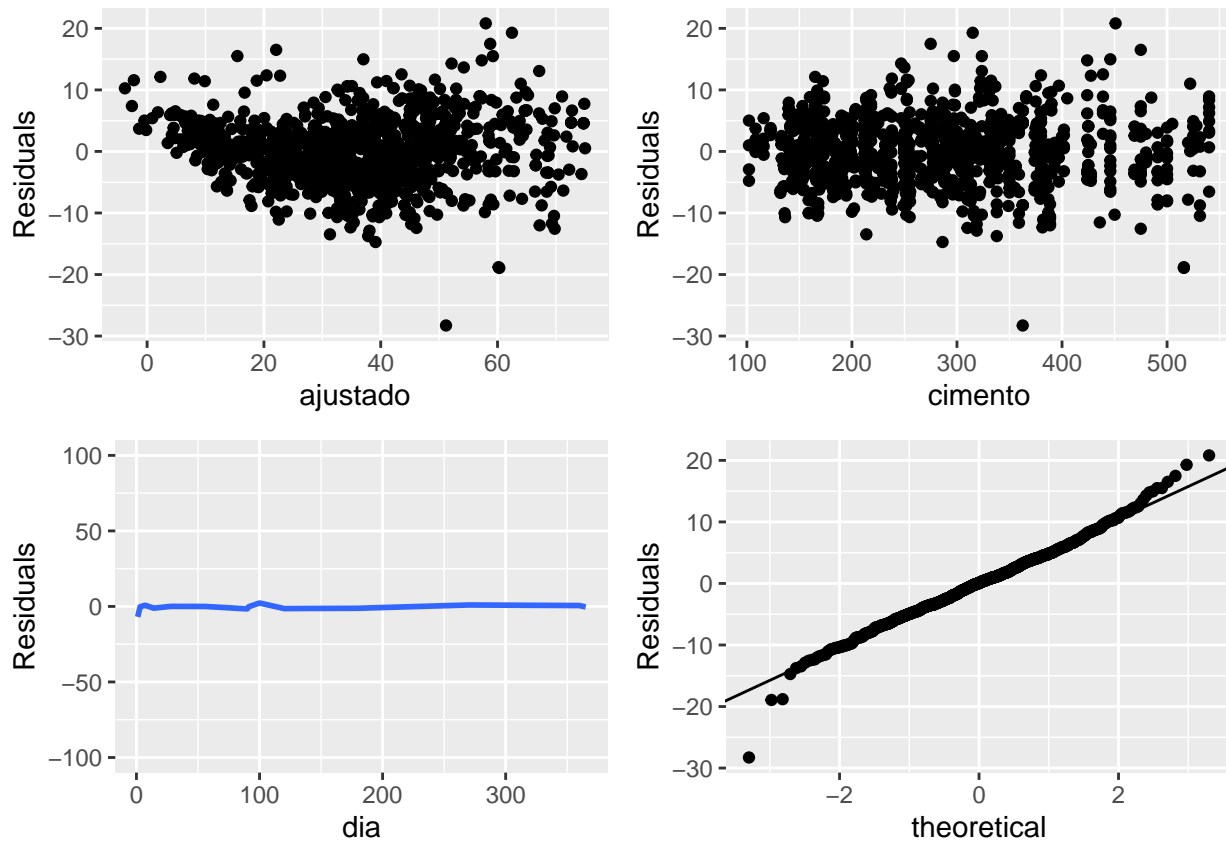


De acordo com o modelo é possível entender o impacto de cada variável na força compressiva. Nesse gráfico está registrada a variação interquartílica e o impacto na variável resposta. É fácil observar que a idade, cimento, cinza e escória, componentes fundamentais na produção do concreto, são as variáveis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

```
## Fazendo os plots dos resíduos
```

```
both <- data.frame(residuos = resid(mod1), ajustado = fitted(mod1))
both$cimento <- dados$cimento
both$dia <- dados$idade

y1 <- ylab('Residuals')
p1 <- ggplot(both, aes(x = ajustado, y = residuos)) + geom_point() + y1
p2 <- ggplot(both, aes(x = cimento, y = residuos)) + geom_point() + y1
p3 <- ggplot(both, aes(x = dia, y = residuos)) + y1 + 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))) + y1
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 seja 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)
```


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.