Relatório 1 - Regressão

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

14/05/2022

Conjunto de dados

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
## Carregando os pacotes
require(readxl)
require(corrplot)
require(psych)
require(kableExtra)
require(caret)
require(car)
require(GGally)
require(ggplot2)
```

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"
  )


## Sumario dos dados
describe(dados)
```

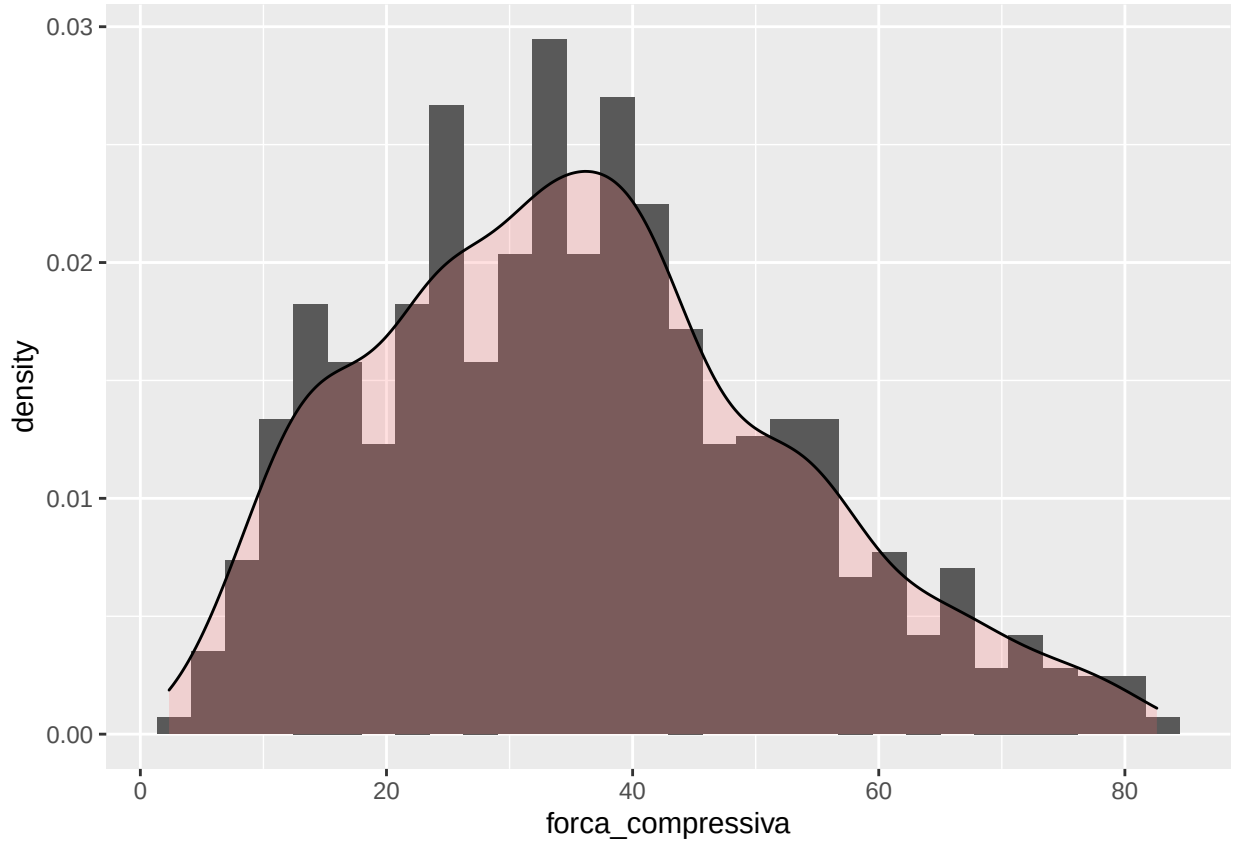
```
##                      vars    n   mean     sd median trimmed    mad    min    max
## cimento                 1 1030 281.17 104.51 272.90  273.47 117.72 102.00  540.0
## escoria                 2 1030  73.90  86.28  22.00   62.43  32.62   0.00  359.4
## cinza                   3 1030  54.19  64.00   0.00   46.85   0.00   0.00  200.1
## agua                    4 1030 181.57  21.36 185.00  181.19  19.27 121.75  247.0
## super_plastificante     5 1030   6.20   5.97   6.35    5.56   7.87   0.00   32.2
## agregador_grosso        6 1030 972.92  77.75 968.00  973.49  68.64 801.00 1145.0
## agregador_fino          7 1030 773.58  80.18 779.51  776.41  67.44 594.00  992.6
## idade                   8 1030  45.66  63.17  28.00   32.53  31.13   1.00  365.0
## forca_compressiva       9 1030  35.82  16.71  34.44   34.96  16.20   2.33   82.6
##                       range  skew kurtosis   se
## cimento              438.00  0.51    -0.53 3.26
## escoria              359.40  0.80    -0.52 2.69
## cinza                200.10  0.54    -1.33 1.99
## agua                 125.25  0.07     0.11 0.67
## super_plastificante  32.20  0.91     1.39 0.19
## agregador_grosso     344.00 -0.04    -0.61 2.42
## agregador_fino       398.60 -0.25    -0.11 2.50
## idade                364.00  3.26    12.07 1.97
## forca_compressiva     80.27  0.42    -0.32 0.52
```

```
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 / m3)
- Escoria (kg / m3)
- Cinza (kg / m3)
- Agua (kg / m3)
- Super plastificante (kg / m3)
- Agregadro grosso (kg / m3)
- Agregador fino (kg / m3)
- Idade (Dias 1~365)
- 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

Preparação dos dados

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

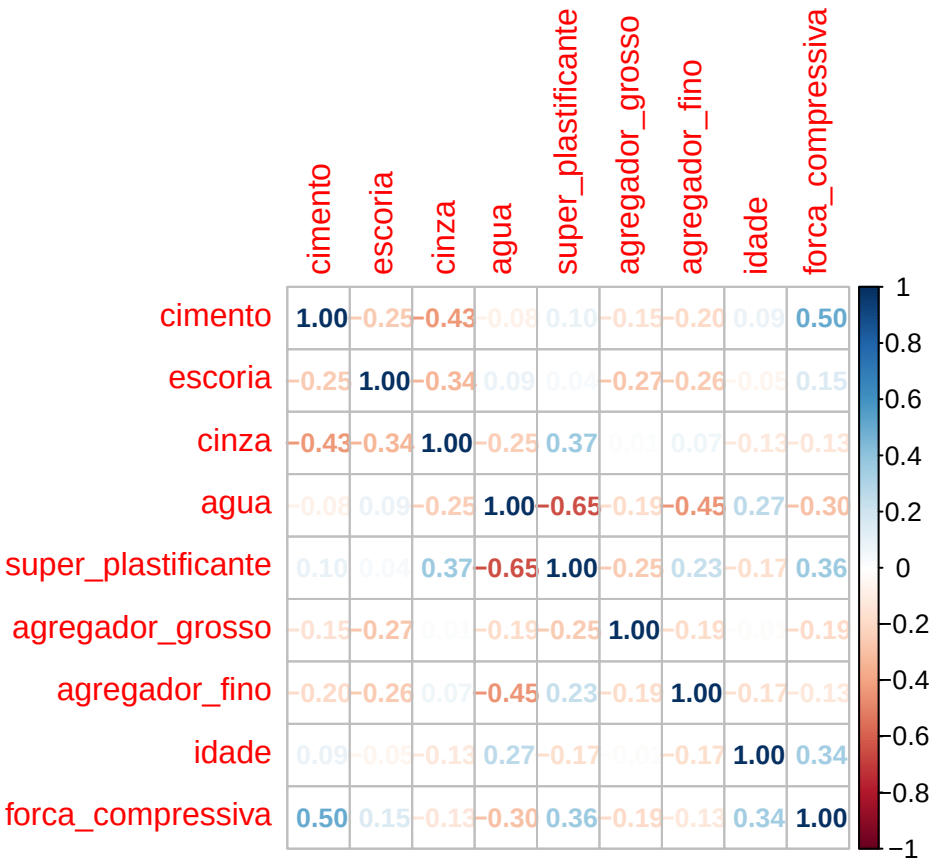
```
treino <- dados[inTrain,]
teste <- dados[-inTrain,]

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


## 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, "/")


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

# ## Removendo a variável resposta
# treino <- treino[,-ncol(treino)]
# teste <- teste[,-ncol(teste)]

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

Redução de dimensionalidade

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



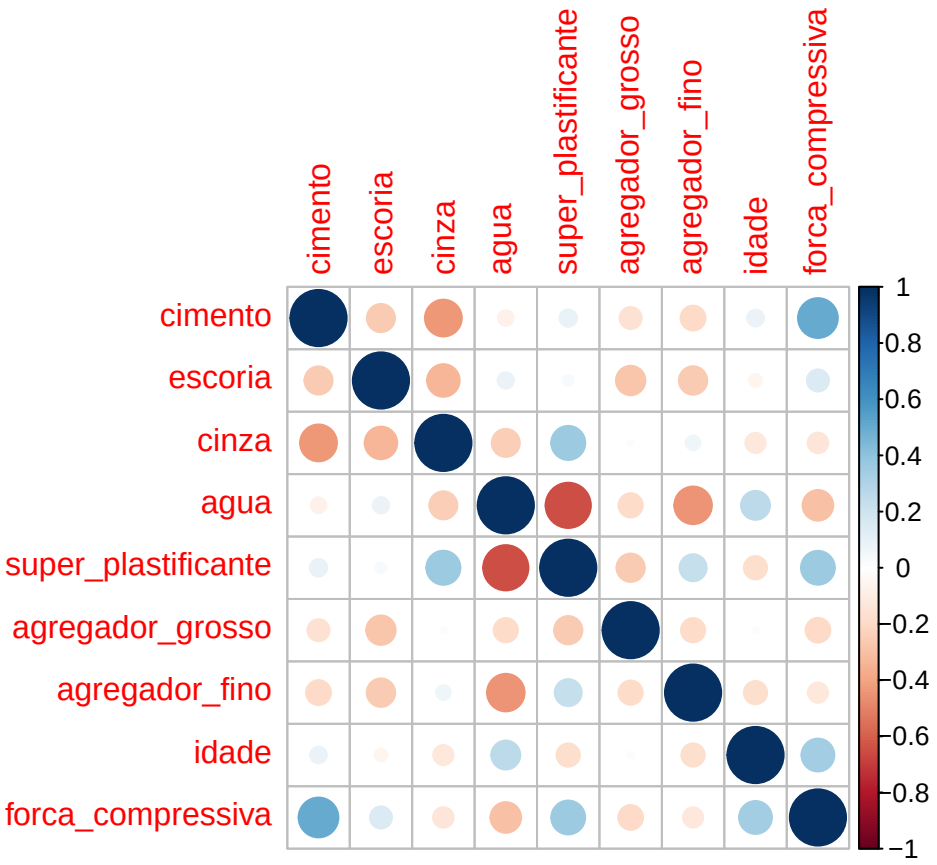

```
summary(M[upper.tri(M)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.64810 -0.24911 -0.13175 -0.07276  0.08627  0.50191
```

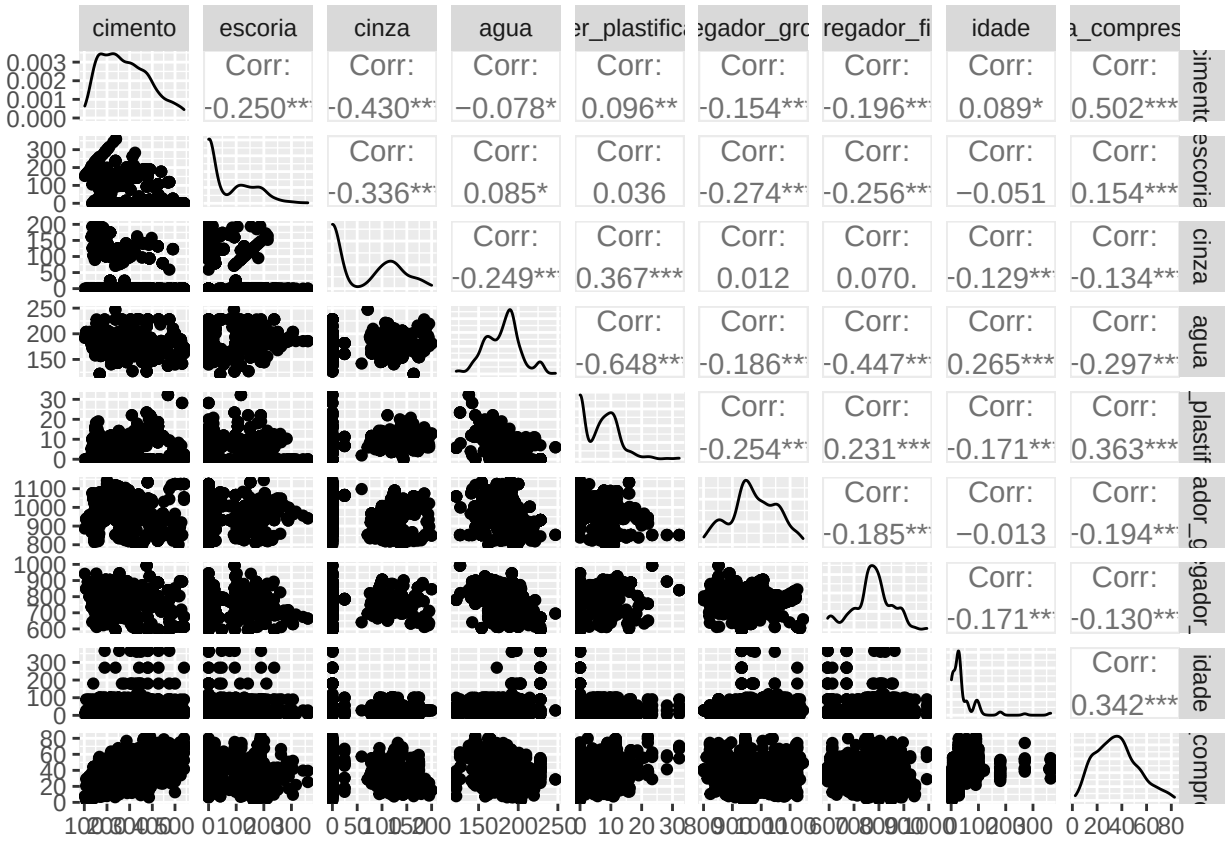
```
## Imprimindo as correlações na forma de circulos
M <- cor(numericas, use = 'complete.obs')
summary(M[upper.tri(M)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.64810 -0.24911 -0.13175 -0.07276  0.08627  0.50191
```

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




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




```
## Análise de Multicolinearidade


modelo1 <- lm(forca_compressiva ~., data = treino)

vif(modelo1)
```

```
##             cimento             escoria              cinza               agua
##            7.601991            6.945083            6.565158            6.651876
## super_plastificante    agregador_grosso      agregador_fino              idade
##            2.759587            5.052023            6.382397            1.109771
```

Como podemos notar, não temos uma correlação muita alta (pensando em módulo) entre as covariáveis. Entretanto, mesmo que tivéssemos, provavelmente não poderíamos remover alguma delas, pois, todas podem ser importantes, seja em termos químicos, seja em termos legas/legislativos. 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.

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. /linebreak Para esse teste de multicolinearidade vamos utilizar a função vif (variance inflation factor) do pacote car. /linebreak Como não temos valores

muito altos (a cima de 10) de vif para nenhuma das covariáveis, vamos assumir que não sofremos do problema de multicolinearidade

Detecção de Outliers

```
## Outliers em X

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(modelo1)
```

```
## 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

Modelagem

```
f1 <-
  formula(
    forca_compressiva ~ cimento + escoria + cinza + agua +
      super_plastificante + agregador_grosso +
      agregador_fino + idade
  )


f1_log <-
  formula(
    log(forca_compressiva) ~ cimento + escoria + cinza + agua +
      super_plastificante + agregador_grosso +
      agregador_fino + idade
  )

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)
  )

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)
  )

formulas <- c(f1, f2)


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('')

}
```

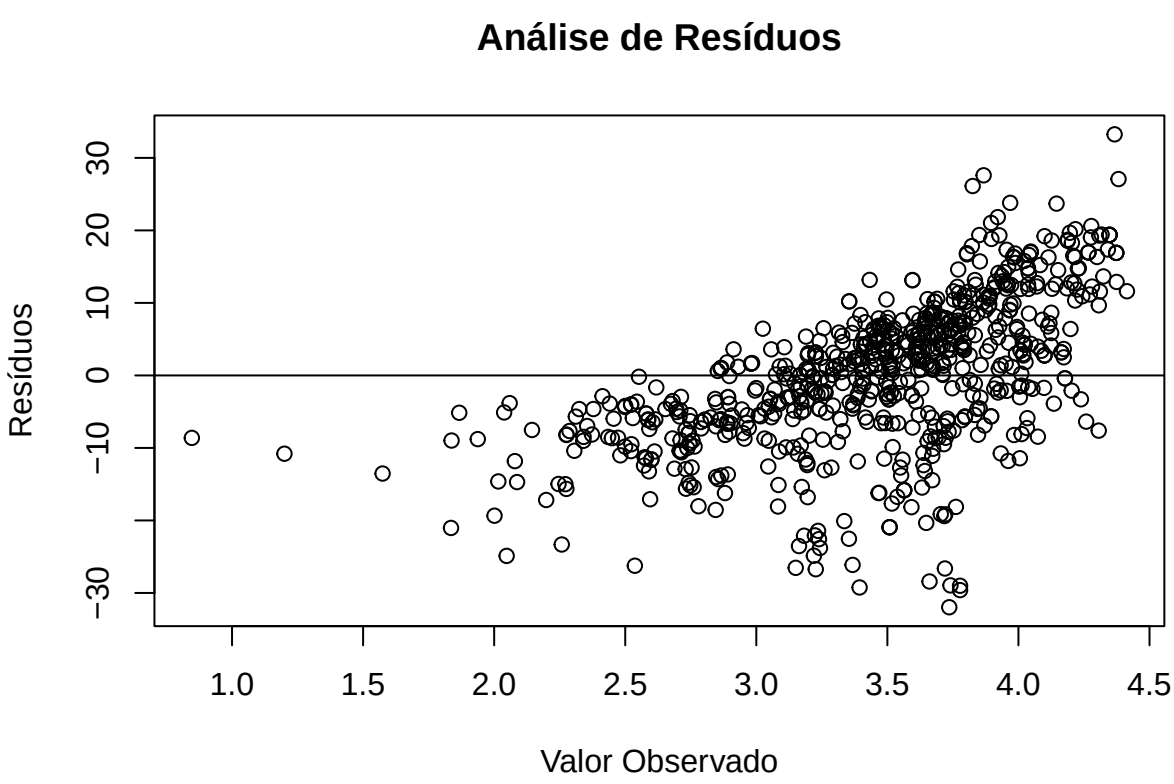
```
## log(forca_compressiva) ~ cimento + escoria + cinza + agua + super_plastificante +
##     agregador_grosso + agregador_fino + idade
## [1] "DATASET NÃO NORMALIZADO (TREINO -- TESTE): \n"
## [1] "MAE :0.282 --  0.307"
```

```
## [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] ""
```

Temos aqui que aplicando o log na variável resposta, analisando MAE e RMSE, temos um melhor desempenho. Notamos também que normalizar ou não os dados fez pouca diferença, entretanto os modelos com termos quadráticos tiverem um melhor resultado

Análise de Resíduos

```
### A partir do tópico anterior, selecionamos o modelo a baixo. Vamos utilizá-lo para fazer a análise d


final_model <- lm(formula = f1, data = treino)

e <- resid(final_model)


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


Análise de Resíduos
Resíduos
30
20
10
0
−10
−30
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Valor Observado