Solo 0-10

Ana Carolina Murad Lima

2023-07-18

```
# Bibliotecas
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.1
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
# Leitura e tratamento dos dados
dados <- read_excel("Solo 0-10 ok.xlsx")

# Ordenar o dataframe por quatro colunas diferentes
dados <- with(dados, dados[order(CICLO, BLOCO, EFLUENTE, INOCULO), ])

# Converter as colunas para tipo numérico e arredondar valores em duas casas
for (i in 6:16) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
dados[6:16] =  round(dados[6:16], digits = 2)
dados = dados[-5]
str(dados)
```

```
## tibble [80 x 15] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: num [1:80] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO : chr [1:80] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO   : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ pH      : num [1:80] 5.3 5.4 5.3 5.4 5.3 5.3 5.2 5.1 5.2 4.9 ...
##  $ P resina: num [1:80] 35 34 41 35 16 17 18 18 33 27 ...
##  $ S       : num [1:80] 2 8 0 7 2 8 2 10 11 11 ...
##  $ K resina: num [1:80] 2.5 2.4 1.4 2.1 1.5 1.5 1.9 1.6 1.7 1.6 ...
##  $ Na      : num [1:80] 0.08 0 0.08 0 0.11 0 0.09 0 0.08 0 ...
##  $ Ca      : num [1:80] 35 35 37 35 34 30 31 27 28 26 ...
##  $ Mg      : num [1:80] 10 16 13 13 13 15 14 14 12 7 ...
##  $ Al      : num [1:80] 0.3 0 0.6 0 0.3 0 0.5 0.1 0.4 1.1 ...
##  $ H+Al    : num [1:80] 39.2 35.2 35.2 32.1 40 ...
##  $ MO      : num [1:80] 23 22.9 21.5 22.7 21.7 23.2 24.4 22.9 22.3 22.1 ...
##  $ CT      : num [1:80] 13.3 13.3 12.5 13.2 12.6 ...
```

```
# Transformar as colunas em variáveis categóricas
dados$BLOCO <- factor(dados$BLOCO)
dados$CICLO <- factor(dados$CICLO)
dados$INOCULO <- factor(dados$INOCULO)
dados$EFLUENTE <- factor(dados$EFLUENTE)
```

```
# Mudar nomes de algumas colunas
colnames(dados)[6] = "P_resina"
colnames(dados)[8] = "K_resina"
colnames(dados)[13] = "H_AL"
```

```
# Níveis para cada fator de tratamentos
library(dae)
n.F <- 2
n.D <- 5
n.Bloco <- 4
tr <- data.frame(cbind(INOCULO = paste("I", rep(1:n.F, each = n.D, times =
                                                 n.Bloco),
                                       sep = ""),
                       EFLUENTE = paste("E", rep(1:n.D, times = n.F*n.Bloco),
                                        sep = "")))
units <- list(Bloco = n.Bloco,
              Parcela = (n.F*n.D))
nest <- list(Parcela = "Bloco")
(lay <- designRandomize(allocated = tr,
                        recipient = units,
                        nested.recipients = nest,
                        seed = 9719532))
```

```
##    Bloco Parcela INOCULO EFLUENTE
## 1      1       1      I2       E3
## 2      1       2      I1       E2
## 3      1       3      I1       E1
## 4      1       4      I1       E4
## 5      1       5      I2       E4
## 6      1       6      I1       E5
## 7      1       7      I2       E1
## 8      1       8      I1       E3
## 9      1       9      I2       E5
## 10     1      10      I2       E2
## 11     2       1      I1       E1
```

```
## 12     2       2      I2       E5
## 13     2       3      I2       E4
## 14     2       4      I1       E3
## 15     2       5      I2       E3
## 16     2       6      I2       E1
## 17     2       7      I1       E4
## 18     2       8      I2       E2
## 19     2       9      I1       E2
## 20     2      10      I1       E5
## 21     3       1      I1       E2
## 22     3       2      I2       E4
## 23     3       3      I2       E5
## 24     3       4      I2       E2
## 25     3       5      I1       E5
## 26     3       6      I2       E3
## 27     3       7      I1       E3
## 28     3       8      I1       E4
## 29     3       9      I1       E1
## 30     3      10      I2       E1
## 31     4       1      I2       E5
## 32     4       2      I2       E4
## 33     4       3      I1       E3
## 34     4       4      I1       E2
## 35     4       5      I1       E4
## 36     4       6      I2       E1
## 37     4       7      I2       E3
## 38     4       8      I2       E2
## 39     4       9      I1       E1
## 40     4      10      I1       E5
```

```
table(lay$I)
```

```
##
## I1 I2
## 20 20
```

```
table(lay$E)
```

```
##
## E1 E2 E3 E4 E5
##  8  8  8  8  8
```

```
lay$Tratamento <- factor(paste(lay$I, lay$E, sep = ":"))
print(lay$Tratamento)
```

```
##  [1] I2:E3 I1:E2 I1:E1 I1:E4 I2:E4 I1:E5 I2:E1 I1:E3 I2:E5 I2:E2 I1:E1 I2:E5
## [13] I2:E4 I1:E3 I2:E3 I2:E1 I1:E4 I2:E2 I1:E2 I1:E5 I1:E2 I2:E4 I2:E5 I2:E2
## [25] I1:E5 I2:E3 I1:E3 I1:E4 I1:E1 I2:E1 I2:E5 I2:E4 I1:E3 I1:E2 I1:E4 I2:E1
## [37] I2:E3 I2:E2 I1:E1 I1:E5
## Levels: I1:E1 I1:E2 I1:E3 I1:E4 I1:E5 I2:E1 I2:E2 I2:E3 I2:E4 I2:E5
```

```r
# Separar os dados de acordo com o período de tempo da coleta
dados_1 = dados[c(1:5)]
dados_2 = dados[c(1:4,6)]
dados_3 = dados[c(1:4,7)]
dados_4 = dados[c(1:4,8)]
dados_5 = dados[c(1:4,9)]
dados_6 = dados[c(1:4,10)]
dados_7 = dados[c(1:4,11)]
dados_8 = dados[c(1:4,12)]
dados_9 = dados[c(1:4,13)]
dados_10 = dados[c(1:4,14)]
dados_11 = dados[c(1:4,15)]

# Estrutura dos dados após separados
"dados_1"
```

```
## [1] "dados_1"
```

```r
str(dados_1)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ pH      : num [1:80] 5.3 5.4 5.3 5.4 5.3 5.3 5.2 5.1 5.2 4.9 ...
```

```r
"dados_2"
```

```
## [1] "dados_2"
```

```r
str(dados_2)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ P_resina: num [1:80] 35 34 41 35 16 17 18 18 33 27 ...
```

```r
"dados_3"
```

```
## [1] "dados_3"
```

```r
str(dados_3)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S       : num [1:80] 2 8 0 7 2 8 2 10 11 11 ...
```

```
"dados_4"
```

```
## [1] "dados_4"
```

```
str(dados_4)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ K_resina: num [1:80] 2.5 2.4 1.4 2.1 1.5 1.5 1.9 1.6 1.7 1.6 ...
```

```
"dados_5"
```

```
## [1] "dados_5"
```

```
str(dados_5)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Na      : num [1:80] 0.08 0 0.08 0 0.11 0 0.09 0 0.08 0 ...
```

```
"dados_6"
```

```
## [1] "dados_6"
```

```
str(dados_6)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Ca      : num [1:80] 35 35 37 35 34 30 31 27 28 26 ...
```

```
"dados_7"
```

```
## [1] "dados_7"
```

```
str(dados_7)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Mg      : num [1:80] 10 16 13 13 13 15 14 14 12 7 ...
```

```
"dados_8"
```

```
## [1] "dados_8"
```

```
str(dados_8)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Al      : num [1:80] 0.3 0 0.6 0 0.3 0 0.5 0.1 0.4 1.1 ...
```

```
"dados_9"
```

```
## [1] "dados_9"
```

```
str(dados_9)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ H_AL    : num [1:80] 39.2 35.2 35.2 32.1 40 ...
```

```
"dados_10"
```

```
## [1] "dados_10"
```

```
str(dados_10)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MO      : num [1:80] 23 22.9 21.5 22.7 21.7 23.2 24.4 22.9 22.3 22.1 ...
```

```
"dados_11"
```

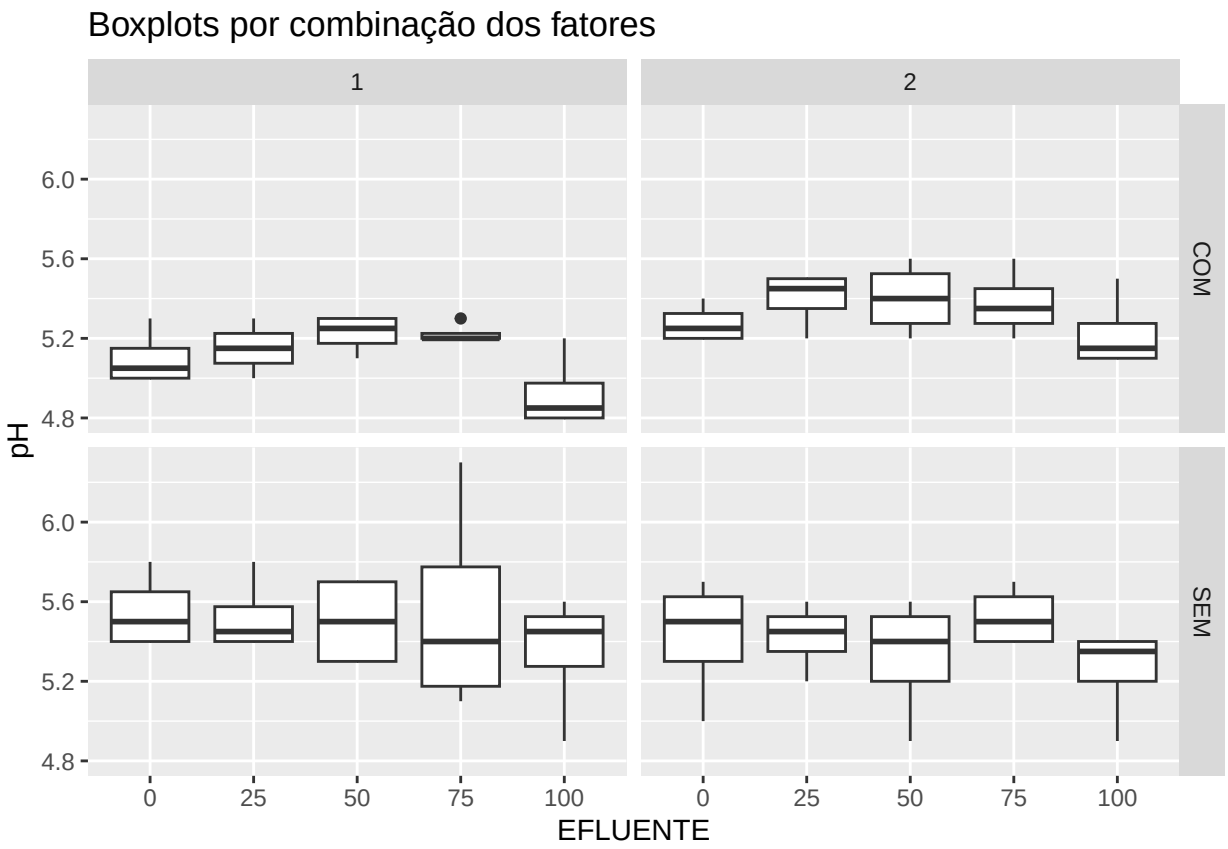
```
## [1] "dados_11"
```

```
str(dados_11)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CT      : num [1:80] 13.3 13.3 12.5 13.2 12.6 ...
```

Análise para pH

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_1, aes(x = factor(EFLUENTE), y = pH)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "pH") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(pH ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_1, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$N

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(pH ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_1, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
```

```
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$pH

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_1 <- with(dados_1,
                       dados_1[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_1$pH[which(blocos_dados_1$pH <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_1$pH[which(blocos_dados_1$pH >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_1$pH > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```
# Calcular a média para cada grupo de 4 linhas
media_blocos_1 = aggregate(pH ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_1, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_1 = media_blocos_1[rep(row.names(media_blocos_1),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_1) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_1$pH[which(is.na(blocos_dados_1$pH))] =
  media_blocos_1$pH[which(is.na(blocos_dados_1$pH))]
```

```
# Análises Descritivas
str(blocos_dados_1)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ pH      : num [1:80] 5.3 5.1 5 5 5.3 5.2 5.1 5 5.3 5.2 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        pH
##  1:20   0  :16   COM:40   1:40   Min.   :4.800
##  2:20   25 :16   SEM:40   2:40   1st Qu.:5.200
##  3:20   50 :16                   Median :5.300
##  4:20   75 :16                   Mean   :5.324
##         100:16                   3rd Qu.:5.500
##                                  Max.   :6.300
```

```
# Número de observações
with(blocos_dados_1, tapply(pH, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_1, tapply(pH, list(EFLUENTE, INOCULO), sum))
```

```
##      COM      SEM
## 0   41.5 43.90000
## 25  42.2 43.80000
## 50  42.5 42.93333
## 75  42.3 43.80000
## 100 40.6 42.40000
```

```
with(blocos_dados_1, tapply(pH, list(EFLUENTE, INOCULO), mean))
```

```
##        COM      SEM
## 0   5.1875 5.487500
## 25  5.2750 5.475000
## 50  5.3125 5.366667
## 75  5.2875 5.475000
## 100 5.0750 5.300000
```

```
with(blocos_dados_1, tapply(pH, list(EFLUENTE, INOCULO), var))
```

```
##            COM        SEM
## 0   0.02125000 0.06125000
## 25  0.03357143 0.03071429
## 50  0.02696429 0.06984127
## 75  0.02125000 0.13357143
## 100 0.05642857 0.06857143
```

```r
with(blocos_dados_1, tapply(pH, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.1457738 0.2474874
## 25  0.1832251 0.1752549
## 50  0.1642081 0.2642750
## 75  0.1457738 0.3654743
## 100 0.2375470 0.2618615
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_1,
                 model.tables(aov(pH ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                       CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                       CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 5.324167
##
##  CICLO
## CICLO
##     1     2
## 5.308 5.341
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 5.310 5.305 5.265 5.417
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 5.338 5.375 5.340 5.381 5.188
##
##  INOCULO
## INOCULO
##   COM   SEM
## 5.228 5.421
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.325 5.338 5.363 5.375 5.138
##     2 5.350 5.413 5.317 5.388 5.238
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 5.120 5.495
##     2 5.335 5.347
```

```
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   5.188 5.488
##      25  5.275 5.475
##      50  5.313 5.367
##      75  5.288 5.475
##      100 5.075 5.300
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.100 5.150 5.225 5.200 4.925
##     2 5.275 5.400 5.400 5.375 5.225
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.550 5.525 5.500 5.550 5.350
##     2 5.425 5.425 5.233 5.400 5.250
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_1 = aggregate(pH ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_1, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_1$pH)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.3618752
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_1$pH,
                                    media_blocos_1$EFLUENTE,
                                    media_blocos_1$INOCULO,
                                    media_blocos_1$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.9714146
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(pH ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_1)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3 0.2525 0.08417
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.7476  0.7476
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.5323  0.1774
##
## Error: Within
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                    1 0.0222  0.0222   0.563 0.456353
## EFLUENTE                 4 0.3990  0.0997   2.527 0.051173 .
## CICLO:INOCULO            1 0.6601  0.6601  16.719 0.000145 ***
## CICLO:EFLUENTE           4 0.0518  0.0130   0.328 0.857921
## INOCULO:EFLUENTE         4 0.1273  0.0318   0.806 0.526666
## CICLO:INOCULO:EFLUENTE   4 0.0131  0.0033   0.083 0.987240
## Residuals               54 2.1318  0.0395
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(pH ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_1)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)
```

```r
# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: pH
##                Df  Sum Sq Mean Sq F value    Pr(>F)
## INOCULO         1 0.74756 0.74756  15.994 0.0001851 ***
## CICLO:INOCULO   1 0.66006 0.66006  14.122 0.0004051 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
```

```
    media_interacao <- aggregate(pH ~ CICLO + INOCULO,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(pH ~ CICLO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(pH ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(pH ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }
}
```

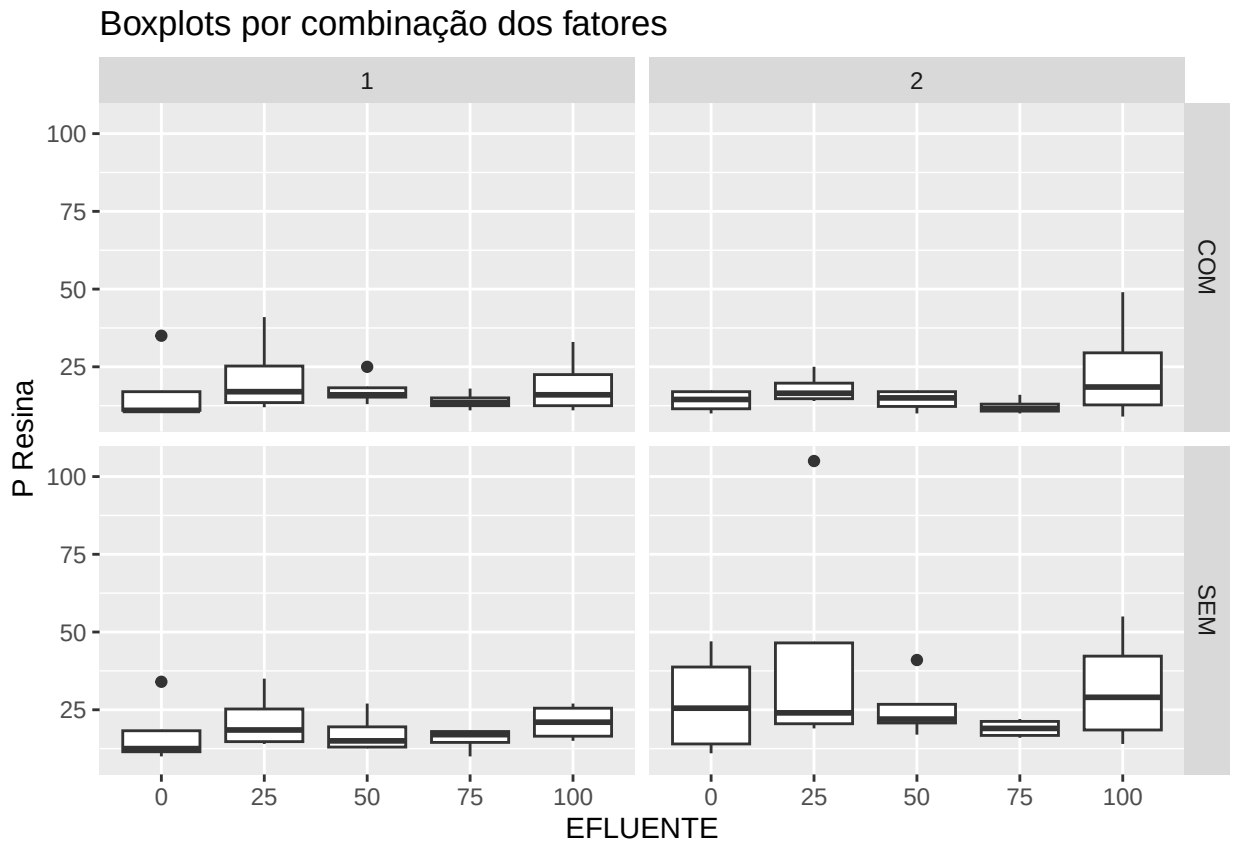
```
##   CICLO INOCULO       pH
## 1     1     COM 5.120000
## 2     2     COM 5.335000
## 3     1     SEM 5.495000
## 4     2     SEM 5.346667
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                   diif        lwr       upr        p_adj
## 2:COM-1:COM 0.2150000 0.03406989 0.3959301 1.367177e-02
## 1:SEM-1:COM 0.3750000 0.19406989 0.5559301 5.796303e-06
## 2:SEM-1:COM 0.2266667 0.04573656 0.4075968 8.438543e-03
```

Análise para P Resina

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_2, aes(x = factor(EFLUENTE), y = P_resina)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "P Resina") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P_resina ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_2, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$P_resina
```

```r
# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P_resina ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_2, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$P_resina

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_2 <- with(dados_2,
                           dados_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_2$P_resina[which(blocos_dados_2$P_resina <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_2$P_resina < limites_outliers$LIM_INF: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_2$P_resina[which(blocos_dados_2$P_resina >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_2$P_resina > limites_outliers$LIM_SUP: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_2 = aggregate(P_resina ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_2, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_2 = media_blocos_2[rep(row.names(media_blocos_2),
                                                each = 4), ]

# Redefinir os índices das linhas
```

```r
rownames(media_blocos_2) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_2$P_resina[which(is.na(blocos_dados_2$P_resina))] =
  media_blocos_2$P_resina[which(is.na(blocos_dados_2$P_resina))]
```

```r
# Análises Descritivas
str(blocos_dados_2)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ P_resina: num [1:80] 11 11 11 11 41 14 12 20 16 15 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     P_resina
##  1:20   0  :16   COM:40   1:40   Min.   : 9.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:13.00
##  3:20   50 :16                   Median :16.00
##  4:20   75 :16                   Mean   :18.02
##         100:16                   3rd Qu.:21.25
##                                  Max.   :49.00
```

```r
# Número de observações
with(blocos_dados_2, tapply(P_resina, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```r
with(blocos_dados_2, tapply(P_resina, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   100.0000 148.6667
## 25  159.0000 175.3333
## 50  113.3333 151.3333
## 75  105.0000 138.0000
## 100 171.0000 180.0000
```

```r
with(blocos_dados_2, tapply(P_resina, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   12.50000 18.58333
```

```
## 25  19.87500 21.91667
## 50  14.16667 18.91667
## 75  13.12500 17.25000
## 100 21.37500 22.50000
```

```
with(blocos_dados_2, tapply(P_resina, list(EFLUENTE, INOCULO), var))
```

```
##            COM      SEM
## 0     8.000000 86.84127
## 25   90.125000 45.29365
## 50    5.174603 23.38889
## 75    7.553571 13.35714
## 100 184.553571 62.00000
```

```
with(blocos_dados_2, tapply(P_resina, list(EFLUENTE, INOCULO), sd))
```

```
##           COM      SEM
## 0    2.828427 9.318866
## 25   9.493419 6.730056
## 50   2.274775 4.836206
## 75   2.748376 3.654743
## 100 13.585050 7.874008
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_2,
                 model.tables(aov(P_resina ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 18.02083
##
##  CICLO
## CICLO
##      1      2
## 16.792 19.250
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 23.900 17.775 16.208 14.200
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 15.542 20.896 16.542 15.188 21.938
##
##  INOCULO
```

```
## INOCULO
##    COM    SEM
## 16.208 19.833
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.333 21.625 16.250 14.750 20.000
##     2 19.750 20.167 16.833 15.625 23.875
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 16.150 17.433
##     2 16.267 22.233
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   12.500 18.583
##      25  19.875 21.917
##      50  14.167 18.917
##      75  13.125 17.250
##      100 21.375 22.500
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.000 21.750 15.000 14.000 19.000
##     2 14.000 18.000 13.333 12.250 23.750
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.667 21.500 17.500 15.500 21.000
##     2 25.500 22.333 20.333 19.000 24.000
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_2 = aggregate(P_resina ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_2, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_2$P_resina)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.414054
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_2$P_resina,
                                    media_blocos_2$EFLUENTE,
                                    media_blocos_2$INOCULO,
                                    media_blocos_2$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.224632
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(P_resina ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_2)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3   1050   350.1
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  262.8   262.8
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  162.5   54.17
##
## Error: Within
##                       Df Sum Sq Mean Sq F value  Pr(>F)
```

```
## CICLO                   1  120.9  120.87   3.389 0.07112 .
## EFLUENTE                4  639.5  159.87   4.483 0.00336 **
## CICLO:INOCULO           1  109.7  109.67   3.075 0.08517 .
## CICLO:EFLUENTE          4  235.5   58.87   1.651 0.17496
## INOCULO:EFLUENTE        4   65.3   16.32   0.458 0.76650
## CICLO:INOCULO:EFLUENTE  4   79.6   19.89   0.558 0.69418
## Residuals              54 1925.7   35.66
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(P_resina ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_2)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: P_resina
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3 1050.18  350.06  9.5551 3.306e-05 ***
## INOCULO    1  262.81  262.81  7.1737  0.009650 **
## EFLUENTE   4  639.49  159.87  4.3638  0.003792 **
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "Não houve interações significativas no modelo"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

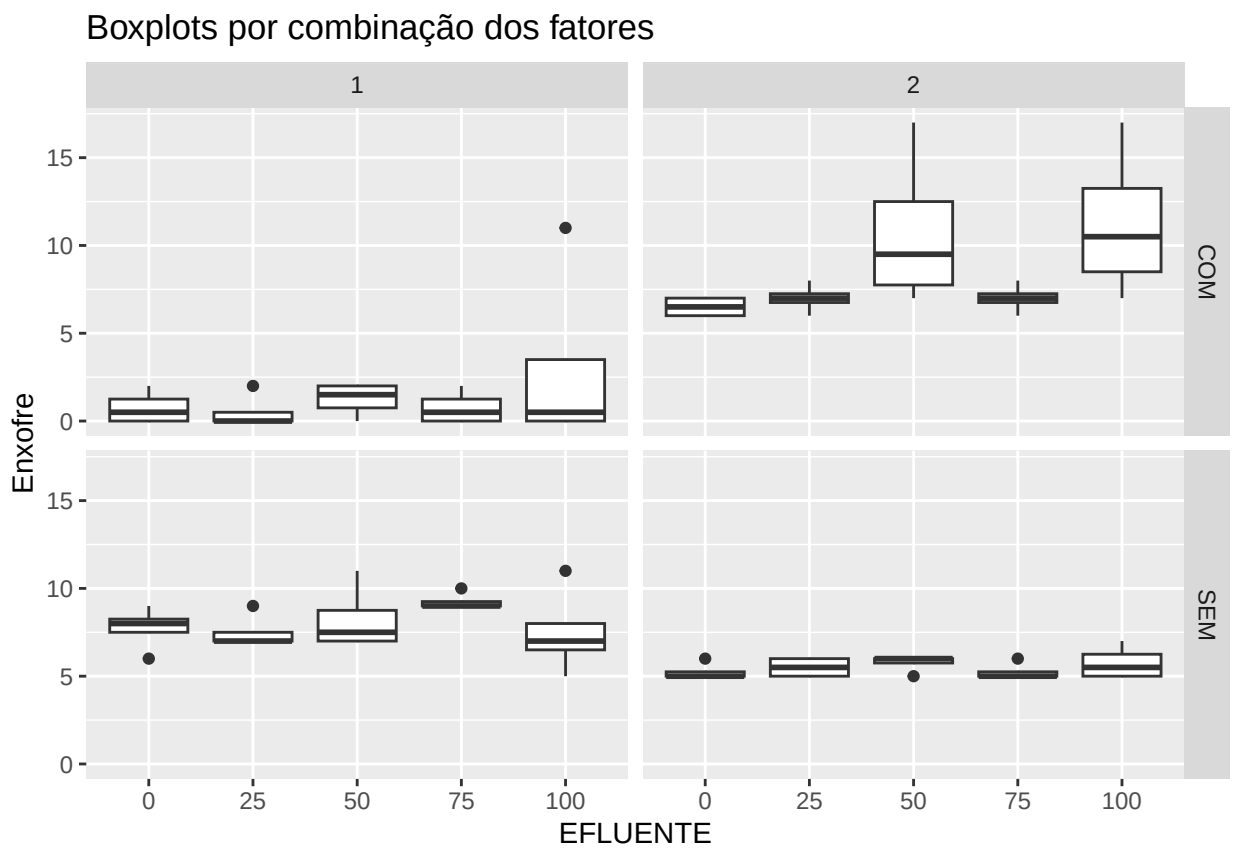
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(P_resina ~ CICLO + INOCULO,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(P_resina ~ CICLO + EFLUENTE,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(P_resina ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(P_resina ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }
}
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
```

```
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

Análise para Enxofre

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_3, aes(x = factor(EFLUENTE), y = S)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Enxofre") +
  ggtitle("Boxplots por combinação dos fatores")
```



```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_3, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$S

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_3, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$S

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_3 <- with(dados_3,
                       dados_3[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_3$S[which(blocos_dados_3$S <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_3$S < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_3$S[which(blocos_dados_3$S >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_3$S > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_3 = aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_3, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_3 = media_blocos_3[rep(row.names(media_blocos_3),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_3) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_3$S[which(is.na(blocos_dados_3$S))] =
  media_blocos_3$S[which(is.na(blocos_dados_3$S))]
```

```r
# Análises Descritivas
str(blocos_dados_3)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S       : num [1:80] 2 1 0 0 0 0 0 0 2 2 ...
```

```r
summary(blocos_dados_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        S
##  1:20   0  :16   COM:40   1:40   Min.   : 0.000
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 2.000
##  3:20   50 :16                   Median : 6.167
##  4:20   75 :16                   Mean   : 5.417
##         100:16                   3rd Qu.: 7.000
##                                  Max.   :17.000
##                                  NA's   :8
```

```r
# Número de observações
with(blocos_dados_3, tapply(S, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```r
with(blocos_dados_3, tapply(S, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
```

```
## 0   29.00000 57.33333
## 25  28.00000 48.00000
## 50  47.66667 57.00000
## 75  32.33333       NA
## 100 29.33333       NA
```

```
with(blocos_dados_3, tapply(S, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   3.625000 7.166667
## 25  3.500000 6.000000
## 50  5.958333 7.125000
## 75  4.041667       NA
## 100 3.666667       NA
```

```
with(blocos_dados_3, tapply(S, list(EFLUENTE, INOCULO), var))
```

```
##           COM      SEM
## 0    9.982143 1.650794
## 25  14.285714 1.142857
## 50  34.394841 2.982143
## 75  12.871032       NA
## 100 12.793651       NA
```

```
with(blocos_dados_3, tapply(S, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   3.159453 1.284832
## 25  3.779645 1.069045
## 50  5.864712 1.726888
## 75  3.587622       NA
## 100 3.576821       NA
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_3,
                 model.tables(aov(S ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 5.416667
##
##  CICLO
##        1      2
##      4.2  6.937
## rep 40.0 32.000
##
```

```
##  BLOCO
##          1      2      3      4
##      5.926  5.278  5.556  4.907
## rep 18.000 18.000 18.000 18.000
##
##  EFLUENTE
##          0     25    50     75   100
##      5.244  4.598  6.39  5.999  4.86
## rep 16.000 16.000 16.00 12.000 12.00
##
##  INOCULO
##        COM    SEM
##      4.005  7.181
## rep 40.000 32.000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   4.798 3.757 5.007 4.490 2.948
##   rep 8.000 8.000 8.000 8.000 8.000
##   2   5.993 5.743 8.077 8.103 7.770
##   rep 8.000 8.000 8.000 4.000 4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##   1    0.740  7.660
##   rep 20.000 20.000
##   2    7.574  5.877
##   rep 20.000 12.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   2.999 7.489
##      rep 8.000 8.000
##      25  2.874 6.322
##      rep 8.000 8.000
##      50  5.332 7.447
##      rep 8.000 8.000
##      75  4.701 8.593
##      rep 8.000 4.000
##      100 4.125 6.329
##      rep 8.000 4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.750  0.000  1.250  0.750  0.333
##   rep  4.000  4.000  4.000  4.000  4.000
##   2    6.500  7.000 10.667  7.333  7.000
##   rep  4.000  4.000  4.000  4.000  4.000
```

```
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    8.333  7.000  8.250  9.000  6.333
##   rep  4.000  4.000  4.000  4.000  4.000
##   2    6.000  5.000  6.000
##   rep  4.000  4.000  4.000  0.000  0.000
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_3 = aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_3, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_3$S)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.02541681
```

```
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados não seguem uma distribuição normal."
```

```
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_3$S,
                                    media_blocos_3$EFLUENTE,
                                    media_blocos_3$INOCULO,
                                    media_blocos_3$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.9910984
```

```
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(S ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_3)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  10.03   3.344
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1  142.5   142.5
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  6.699   2.233
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  172.6   172.6 117.340 1.73e-14 ***
## EFLUENTE                  4   40.7    10.2   6.921 0.000175 ***
## CICLO:INOCULO             1  358.5   358.5 243.719  < 2e-16 ***
## CICLO:EFLUENTE            4   10.7     2.7   1.825 0.139471
## INOCULO:EFLUENTE          4   16.4     4.1   2.789 0.036641 *
## CICLO:INOCULO:EFLUENTE    2    6.9     3.5   2.356 0.105684
## Residuals                48   70.6     1.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(S ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_3)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: S
##                  Df Sum Sq Mean Sq  F value    Pr(>F)
## CICLO             1 133.23  133.23  87.8947 1.110e-12 ***
```

```
## INOCULO           1 181.88  181.88 119.9924 5.268e-15 ***
## EFLUENTE          4  40.72   10.18   6.7163 0.0002018 ***
## CICLO:INOCULO     1 358.49  358.49 236.5111 < 2.2e-16 ***
## INOCULO:EFLUENTE  4  16.41    4.10   2.7069 0.0403188 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"    "INOCULO:EFLUENTE"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(S ~ CICLO + INOCULO,
                                 data = blocos_dados_3, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(S ~ CICLO + EFLUENTE,
                                 data = blocos_dados_3, FUN = mean)
    print(media_interacao)
```

```
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(S ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_3, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(S ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_3, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO         S
## 1     1     COM 0.6166667
## 2     2     COM 7.7000000
## 3     1     SEM 7.7833333
## 4     2     SEM 5.6666667
##    INOCULO EFLUENTE        S
## 1      COM        0 3.625000
## 2      SEM        0 7.166667
## 3      COM       25 3.500000
## 4      SEM       25 6.000000
## 5      COM       50 5.958333
## 6      SEM       50 7.125000
## 7      COM       75 4.041667
## 8      SEM       75 9.000000
## 9      COM      100 3.666667
## 10     SEM      100 6.333333
```

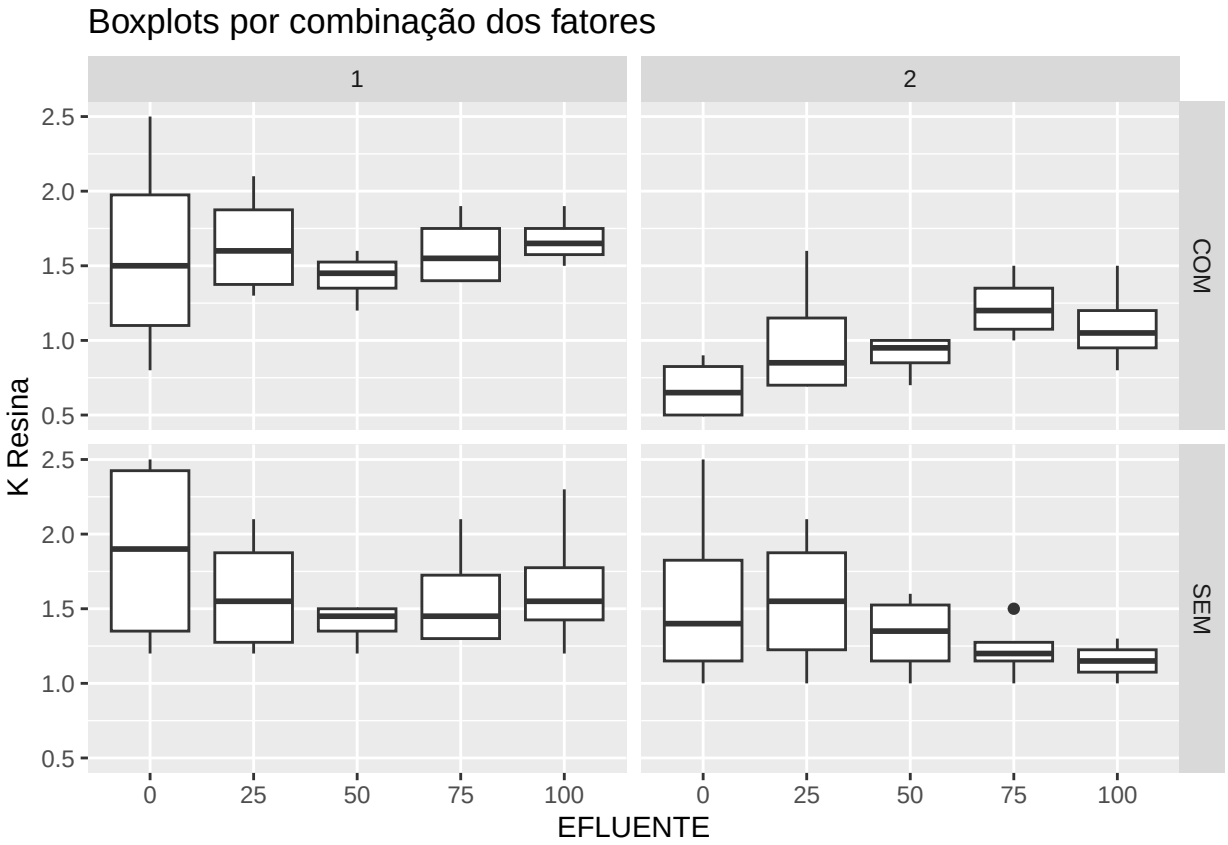
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
```

```
  print(inter_tukey)
}
```

```
##                   diif       lwr        upr        p_adj
## 2:COM-1:COM  6.970022  5.936051  8.0039923 0.0000000000
## 1:SEM-1:COM  7.053355  6.019384  8.0873256 0.0000000000
## 2:SEM-1:COM  5.087771  3.893844  6.2816970 0.0000000000
## 2:SEM-2:COM -1.882251 -3.076178 -0.6883246 0.0006303234
## 2:SEM-1:SEM -1.965584 -3.159511 -0.7716580 0.0003448866
##                     diif         lwr        upr        p_adj
## SEM:0-COM:0     4.489899  2.45397919  6.5258188 8.182507e-08
## SEM:25-COM:0    3.323232  1.28731252  5.3591521 7.270201e-05
## COM:50-COM:0    2.333333  0.29741353  4.3692531 1.333395e-02
## SEM:50-COM:0    4.448232  2.41231252  6.4841521 1.045346e-07
## SEM:75-COM:0    5.594129  3.10064645  8.0876111 5.183770e-08
## SEM:100-COM:0   3.330240  0.83675756  5.8237222 1.962447e-03
## COM:25-SEM:0   -4.614899 -6.65081879 -2.5789792 3.926419e-08
## COM:50-SEM:0   -2.156566 -4.19248546 -0.1206459 2.982505e-02
## COM:75-SEM:0   -2.787247 -4.82316728 -0.7513277 1.373860e-03
## COM:100-SEM:0  -3.363636 -5.39955616 -1.3277166 5.782545e-05
## SEM:25-COM:25   3.448232  1.41231252  5.4841521 3.571680e-05
## COM:50-COM:25   2.458333  0.42241353  4.4942531 7.322867e-03
## SEM:50-COM:25   4.573232  2.53731252  6.6091521 5.014715e-08
## SEM:75-COM:25   5.719129  3.22564645  8.2126111 2.847560e-08
## SEM:100-COM:25  3.455240  0.96175756  5.9487222 1.148242e-03
## COM:100-SEM:25 -2.196970 -4.23288950 -0.1610499 2.493105e-02
## SEM:50-COM:50   2.114899  0.07897919  4.1508188 3.576634e-02
## SEM:75-COM:50   3.260795  0.76731312  5.7542778 2.630823e-03
## COM:75-SEM:50  -2.745581 -4.78150061 -0.7096610 1.709653e-03
## COM:100-SEM:50 -3.321970 -5.35788950 -1.2860499 7.322306e-05
## SEM:75-COM:75   3.891477  1.39799494  6.3849596 1.649463e-04
## COM:100-SEM:75 -4.467866 -6.96134850 -1.9743838 1.132064e-05
```

Análise para K Resina

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_4, aes(x = factor(EFLUENTE), y = K_resina)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "K Resina") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K_resina ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_4, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$K_resina

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K_resina ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_4, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$K_resina

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup
```

```r
# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_4 <- with(dados_4,
                       dados_4[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_4$K_resina[which(blocos_dados_4$K_resina <
                                  limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_4$K_resina < limites_outliers$LIM_INF: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_4$K_resina[which(blocos_dados_4$K_resina >
                                  limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_4$K_resina > limites_outliers$LIM_SUP: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_4 = aggregate(K_resina ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_4, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_4 = media_blocos_4[rep(row.names(media_blocos_4),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_4) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_4$K_resina[which(is.na(blocos_dados_4$K_resina))] =
  media_blocos_4$K_resina[which(is.na(blocos_dados_4$K_resina))]
```

```r
# Análises Descritivas
str(blocos_dados_4)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ K_resina: num [1:80] 2.5 0.8 1.8 1.2 1.4 2.1 1.8 1.3 1.5 1.4 ...
```

```r
summary(blocos_dados_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     K_resina
##  1:20   0  :16   COM:40   1:40   Min.   :0.500
##  2:20   25 :16   SEM:40   2:40   1st Qu.:1.000
##  3:20   50 :16                   Median :1.300
##  4:20   75 :16                   Mean   :1.355
##         100:16                   3rd Qu.:1.600
##                                  Max.   :2.500
```

```r
# Número de observações
with(blocos_dados_4, tapply(K_resina, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```r
with(blocos_dados_4, tapply(K_resina, list(EFLUENTE, INOCULO), sum))
```

```
##      COM      SEM
## 0    9.0 13.80000
## 25   9.8 11.00000
## 50   9.3 10.66667
## 75  11.3 11.20000
## 100 11.1 11.20000
```

```r
with(blocos_dados_4, tapply(K_resina, list(EFLUENTE, INOCULO), mean))
```

```
##        COM      SEM
## 0   1.1250 1.725000
## 25  1.2250 1.375000
## 50  1.1625 1.333333
## 75  1.4125 1.400000
## 100 1.3875 1.400000
```

```r
with(blocos_dados_4, tapply(K_resina, list(EFLUENTE, INOCULO), var))
```

```
##            COM        SEM
## 0   0.48500000 0.40785714
## 25  0.27357143 0.14142857
## 50  0.09982143 0.04031746
## 75  0.08696429 0.11428571
## 100 0.14410714 0.17142857
```

```r
with(blocos_dados_4, tapply(K_resina, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.6964194 0.6386369
## 25  0.5230406 0.3760699
## 50  0.3159453 0.2007921
## 75  0.2948971 0.3380617
## 100 0.3796145 0.4140393
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_4,
                 model.tables(aov(K_resina ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 1.354583
##
##  CICLO
## CICLO
##      1      2
## 1.6025 1.1067
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 1.4758 1.2550 1.4075 1.2800
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 1.4250 1.3000 1.2479 1.4063 1.3938
##
##  INOCULO
## INOCULO
##    COM    SEM
## 1.2625 1.4467
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.7250 1.6250 1.4125 1.5875 1.6625
##     2 1.1250 0.9750 1.0833 1.2250 1.1250
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 1.5850 1.6200
##     2 0.9400 1.2733
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   1.1250 1.7250
##       25  1.2250 1.3750
##       50  1.1625 1.3333
##       75  1.4125 1.4000
##       100 1.3875 1.4000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.5750 1.6500 1.4250 1.6000 1.6750
##     2 0.6750 0.8000 0.9000 1.2250 1.1000
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.8750 1.6000 1.4000 1.5750 1.6500
##     2 1.5750 1.1500 1.2667 1.2250 1.1500
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_4 = aggregate(K_resina ~ EFLUENTE + INOCULO + CICLO,
                                  data = blocos_dados_4, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_4$K_resina)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.2262135
```

```
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_4$K_resina,
                                    media_blocos_4$EFLUENTE,
                                    media_blocos_4$INOCULO,
                                    media_blocos_4$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.581988
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(K_resina ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_4)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3 0.6596  0.2199
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.6783  0.6783
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  0.746  0.2487
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  4.917   4.917  41.335 3.47e-08 ***
## EFLUENTE                  4  0.376   0.094   0.791   0.5363
## CICLO:INOCULO             1  0.445   0.445   3.741   0.0583 .
## CICLO:EFLUENTE            4  0.328   0.082   0.689   0.6030
## INOCULO:EFLUENTE          4  0.970   0.242   2.038   0.1020
## CICLO:INOCULO:EFLUENTE    4  0.235   0.059   0.493   0.7408
## Residuals                54  6.424   0.119
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(K_resina ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_4)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]
```

```r
# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: K_resina
##         Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO    1 4.9170  4.9170 39.0917 5.535e-08 ***
## INOCULO  1 0.6783  0.6783  5.3931   0.02381 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "Não houve interações significativas no modelo"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(K_resina ~ CICLO + INOCULO,
                                 data = blocos_dados_4, FUN = mean)
```

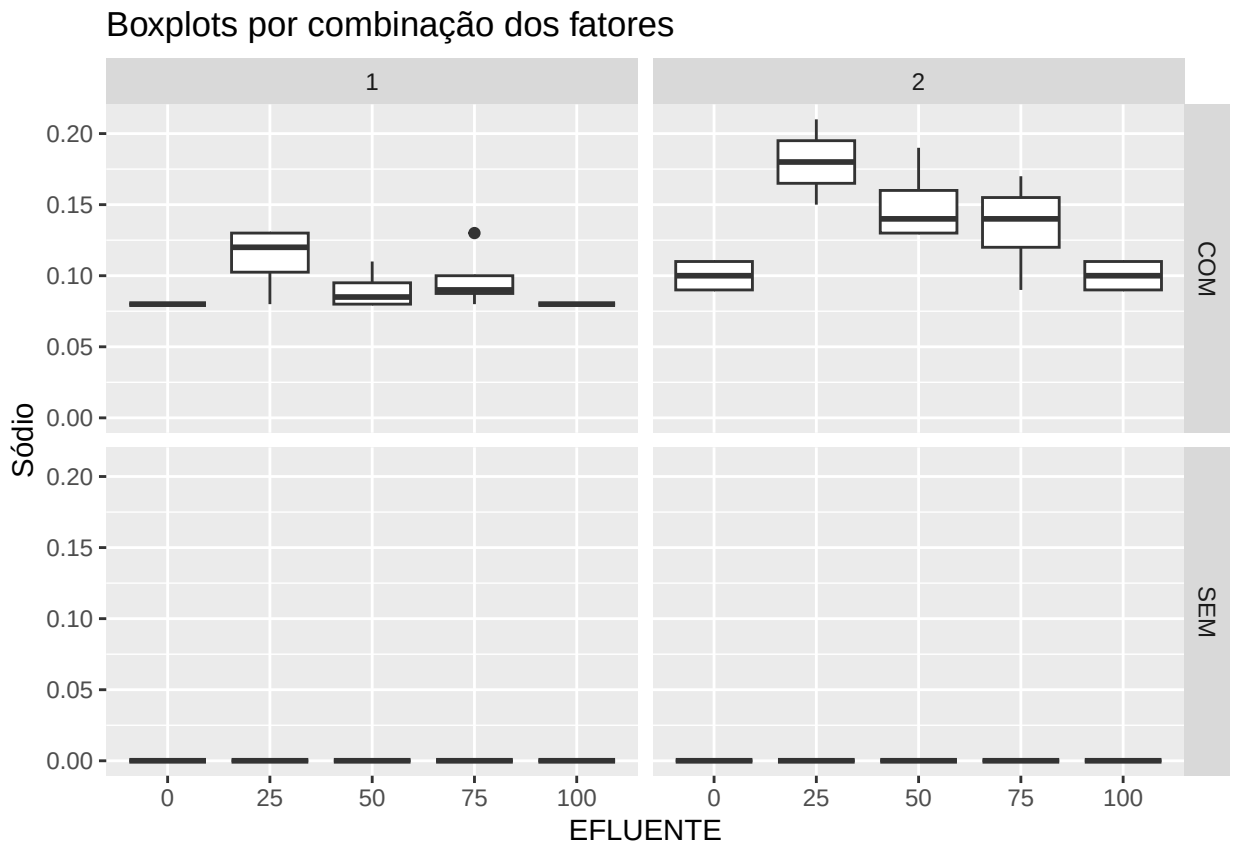
```
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(K_resina ~ CICLO + EFLUENTE,
                                 data = blocos_dados_4, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(K_resina ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_4, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(K_resina ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_4, FUN = mean)
    print(media_interacao)
  }
}
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

Análise para Sódio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_5, aes(x = factor(EFLUENTE), y = Na)) +
  geom_boxplot() +
```

```
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Sódio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_5, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$Na

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_5, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$Na
```

```r
# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_5 <- with(dados_5,
                           dados_5[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_5$Na[which(blocos_dados_5$Na <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_5$Na < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_5$Na[which(blocos_dados_5$Na >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_5$Na > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_5 = aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_5, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_5 = media_blocos_5[rep(row.names(media_blocos_5),
                                              each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_5) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_5$Na[which(is.na(blocos_dados_5$Na))] =
  media_blocos_5$Na[which(is.na(blocos_dados_5$Na))]
```

```r
# Análises Descritivas
str(blocos_dados_5)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Na      : num [1:80] 0.08 0.08 0.08 0.08 0.08 0.11 0.13 0.13 0.11 0.08 ...
```

```
summary(blocos_dados_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Na
##  1:20   0  :16   COM:40   1:40   Min.   :0.00000
##  2:20   25 :16   SEM:40   2:40   1st Qu.:0.00000
##  3:20   50 :16                   Median :0.08000
##  4:20   75 :16                   Mean   :0.05671
##         100:16                   3rd Qu.:0.10000
##                                  Max.   :0.21000
##                                  NA's   :10
```

```
# Número de observações
with(blocos_dados_5, tapply(Na, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_5, tapply(Na, list(EFLUENTE, INOCULO), sum))
```

```
##           COM SEM
## 0   0.7333333   0
## 25  1.1700000   0
## 50  0.9600000  NA
## 75  0.7866667  NA
## 100 0.3200000  NA
```

```
with(blocos_dados_5, tapply(Na, list(EFLUENTE, INOCULO), mean))
```

```
##            COM SEM
## 0   0.09166667   0
## 25  0.14625000   0
## 50  0.12000000  NA
## 75  0.09833333  NA
## 100 0.04000000  NA
```

```
with(blocos_dados_5, tapply(Na, list(EFLUENTE, INOCULO), var))
```

```
##              COM SEM
## 0   0.0001936508   0
```

```
## 25  0.0018267857   0
## 50  0.0014571429  NA
## 75  0.0002793651  NA
## 100 0.0018285714  NA
```

```
with(blocos_dados_5, tapply(Na, list(EFLUENTE, INOCULO), sd))
```

```
##            COM SEM
## 0   0.01391585   0
## 25  0.04274091   0
## 50  0.03817254  NA
## 75  0.01671422  NA
## 100 0.04276180  NA
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_5,
                 model.tables(aov(Na ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 0.05671429
##
##  CICLO
##            1        2
##      0.04492  0.07244
## rep 40.00000 30.00000
##
##  BLOCO
##            1        2        3        4
##      0.05623  0.05623  0.05791  0.05654
## rep 18.00000 18.00000 17.00000 17.00000
##
##  EFLUENTE
##            0       25       50       75      100
##      0.04385  0.07114  0.06863  0.06817  0.02928
## rep 16.00000 16.00000 14.00000 12.00000 12.00000
##
##  INOCULO
##          COM       SEM
##      0.09785  0.001861
## rep 40.00000 30.000000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.030  0.046  0.044  0.054  0.050
##   rep  8.000  8.000  8.000  8.000  8.000
##   2    0.062  0.100  0.101  0.089 -0.021
```

```
##   rep  8.000  8.000  6.000  4.000  4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##   1    0.087  0.003
##   rep 20.000 20.000
##   2    0.113 -0.009
##   rep 20.000 10.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    0.082  0.005
##      rep  8.000  8.000
##      25   0.137  0.005
##      rep  8.000  8.000
##      50   0.112  0.011
##      rep  8.000  6.000
##      75   0.098  0.009
##      rep  8.000  4.000
##      100  0.057 -0.027
##      rep  8.000  4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.080 0.112 0.090 0.087 0.080
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.103 0.180 0.150 0.110 0.000
##   rep 4.000 4.000 4.000 4.000 4.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.000 0.000 0.000 0.000 0.000
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.000 0.000 0.000
##   rep 4.000 4.000 2.000 0.000 0.000
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_5 = aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                                  data = blocos_dados_5, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_5$Na)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.001996149
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados não seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_5$Na,
                                    media_blocos_5$EFLUENTE,
                                    media_blocos_5$INOCULO,
                                    media_blocos_5$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.8770927
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Na ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_5)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df    Sum Sq   Mean Sq
## BLOCO  3 1.618e-05 5.394e-06
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1  0.169   0.169
##
## Error: BLOCO:INOCULO
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## CICLO      1 1.293e-04 0.0001293   23.07 0.0407 *
## Residuals  2 1.121e-05 0.0000056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Error: Within
##                          Df  Sum Sq  Mean Sq F value   Pr(>F)
## CICLO                     1 0.00214 0.002136  13.137 0.000721 ***
## EFLUENTE                  4 0.02864 0.007159  44.037 3.59e-15 ***
## CICLO:INOCULO             1 0.00390 0.003898  23.978 1.24e-05 ***
## CICLO:EFLUENTE            4 0.03891 0.009727  59.828  < 2e-16 ***
## INOCULO:EFLUENTE          4 0.00628 0.001571   9.664 9.29e-06 ***
## CICLO:INOCULO:EFLUENTE    2 0.00109 0.000547   3.366 0.043228 *
## Residuals                46 0.00748 0.000163
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Na ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_5)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: Na
##                          Df   Sum Sq  Mean Sq   F value    Pr(>F)
## CICLO                     1 0.013008 0.013008   83.7073 3.538e-12 ***
## INOCULO                   1 0.158116 0.158116 1017.5085 < 2.2e-16 ***
## EFLUENTE                  4 0.028586 0.007147   45.9894 5.248e-16 ***
## CICLO:INOCULO             1 0.004000 0.004000   25.7407 6.009e-06 ***
## CICLO:EFLUENTE            4 0.038883 0.009721   62.5544 < 2.2e-16 ***
## INOCULO:EFLUENTE          4 0.006286 0.001571   10.1125 4.766e-06 ***
## CICLO:INOCULO:EFLUENTE    2 0.001124 0.000562    3.6168   0.03427 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
```

```
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"          "CICLO:EFLUENTE"         "INOCULO:EFLUENTE"
## [4] "CICLO:INOCULO:EFLUENTE"
```

```
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(Na ~ CICLO + INOCULO,
                                 data = blocos_dados_5, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Na ~ CICLO + EFLUENTE,
                                 data = blocos_dados_5, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Na ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_5, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(Na ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_5, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO         Na
```

```
## 1     1     COM 0.08983333
## 2     2     COM 0.10866667
## 3     1     SEM 0.00000000
## 4     2     SEM 0.00000000
##    CICLO EFLUENTE         Na
## 1      1        0 0.04000000
## 2      2        0 0.05166667
## 3      1       25 0.05625000
## 4      2       25 0.09000000
## 5      1       50 0.04500000
## 6      2       50 0.10000000
## 7      1       75 0.04333333
## 8      2       75 0.11000000
## 9      1      100 0.04000000
## 10     2      100 0.00000000
##    INOCULO EFLUENTE         Na
## 1      COM        0 0.09166667
## 2      SEM        0 0.00000000
## 3      COM       25 0.14625000
## 4      SEM       25 0.00000000
## 5      COM       50 0.12000000
## 6      SEM       50 0.00000000
## 7      COM       75 0.09833333
## 8      SEM       75 0.00000000
## 9      COM      100 0.04000000
## 10     SEM      100 0.00000000
##    CICLO INOCULO EFLUENTE         Na
## 1      1     COM        0 0.08000000
## 2      2     COM        0 0.10333333
## 3      1     SEM        0 0.00000000
## 4      2     SEM        0 0.00000000
## 5      1     COM       25 0.11250000
## 6      2     COM       25 0.18000000
## 7      1     SEM       25 0.00000000
## 8      2     SEM       25 0.00000000
## 9      1     COM       50 0.09000000
## 10     2     COM       50 0.15000000
## 11     1     SEM       50 0.00000000
## 12     2     SEM       50 0.00000000
## 13     1     COM       75 0.08666667
## 14     2     COM       75 0.11000000
## 15     1     SEM       75 0.00000000
## 16     1     COM      100 0.08000000
## 17     2     COM      100 0.00000000
## 18     1     SEM      100 0.00000000
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
```

```
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                    diif         lwr         upr        p_adj
## 2:COM-1:COM  0.02350785  0.01302428  0.03399143 1.570341e-06
## 1:SEM-1:COM -0.08515881 -0.09564239 -0.07467524 0.000000e+00
## 2:SEM-1:COM -0.09216079 -0.10500049 -0.07932109 0.000000e+00
## 1:SEM-2:COM -0.10866667 -0.11915024 -0.09818309 0.000000e+00
## 2:SEM-2:COM -0.11566864 -0.12850834 -0.10282894 0.000000e+00
##                    diif          lwr          upr        p_adj
## 2:0-1:0      0.03195795  0.011306400  0.052609506 2.009182e-04
## 2:25-1:0     0.07029129  0.049639733  0.090942839 0.000000e+00
## 2:50-1:0     0.06172659  0.039420362  0.084032815 1.430043e-10
## 2:75-1:0     0.03548727  0.010194387  0.060780154 9.807957e-04
## 2:100-1:0   -0.07451273 -0.099805613 -0.049219846 1.604483e-11
## 2:25-2:0     0.03833333  0.017681780  0.058984886 5.832623e-06
## 1:50-2:0    -0.02518150 -0.045833056 -0.004529950 6.588747e-03
## 2:50-2:0     0.02976864  0.007462409  0.052074862 2.030165e-03
## 1:75-2:0    -0.02461116 -0.045262711 -0.003959605 8.650343e-03
## 1:100-2:0   -0.02794449 -0.048596045 -0.007292939 1.669350e-03
## 2:100-2:0   -0.10647068 -0.131763566 -0.081177799 0.000000e+00
## 2:25-1:25    0.05404129  0.033389733  0.074692839 8.091267e-10
## 2:50-1:25    0.04547659  0.023170362  0.067782815 6.868573e-07
## 2:100-1:25  -0.09076273 -0.116055613 -0.065469846 0.000000e+00
## 1:50-2:25   -0.06351484 -0.084166389 -0.042863283 9.513501e-13
## 1:75-2:25   -0.06294449 -0.083596045 -0.042292939 2.640221e-12
## 2:75-2:25   -0.03480402 -0.060096899 -0.009511132 1.309471e-03
## 1:100-2:25  -0.06627782 -0.086929378 -0.045626272 0.000000e+00
## 2:100-2:25  -0.14480402 -0.170096899 -0.119511132 0.000000e+00
## 2:50-1:50    0.05495014  0.032643912  0.077256365 4.759095e-09
## 2:75-1:50    0.02871082  0.003417937  0.054003704 1.490337e-02
## 2:100-1:50  -0.08128918 -0.106582063 -0.055996296 0.000000e+00
## 1:75-2:50   -0.05437979 -0.076686020 -0.032073567 6.404110e-09
## 1:100-2:50  -0.05771313 -0.080019354 -0.035406900 1.136545e-09
## 2:100-2:50  -0.13623932 -0.162900358 -0.109578278 0.000000e+00
## 2:75-1:75    0.02814048  0.002847592  0.053433359 1.840633e-02
## 2:100-1:75  -0.08185952 -0.107152408 -0.056566641 0.000000e+00
## 1:100-2:75  -0.03147381 -0.056766693 -0.006180926 5.130200e-03
## 2:100-2:75  -0.11000000 -0.139205706 -0.080794294 0.000000e+00
## 2:100-1:100 -0.07852619 -0.103819074 -0.053233307 0.000000e+00
```

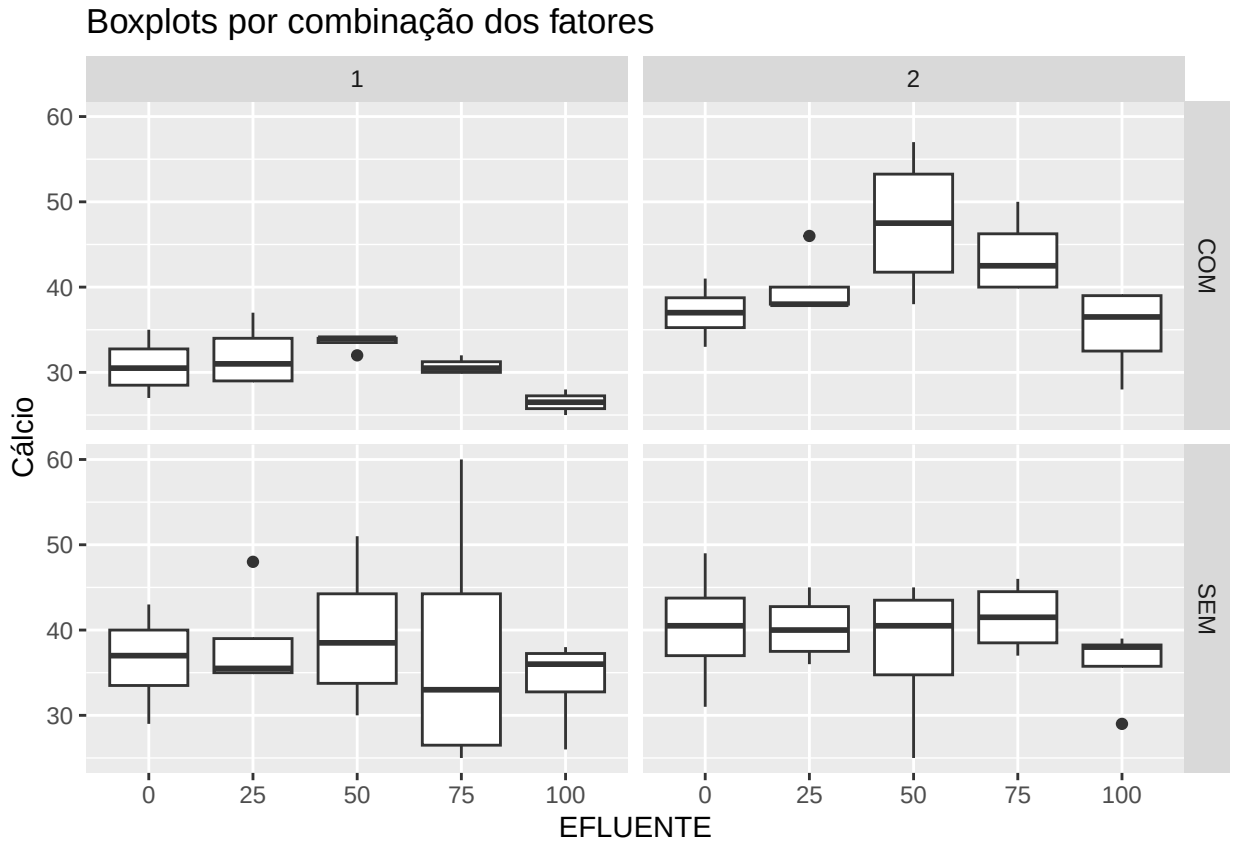
```
##                         diif          lwr          upr        p_adj
## SEM:0-COM:0     -0.07684088 -0.097492428 -0.056189322 0.000000e+00
## COM:25-COM:0     0.05458333  0.033931780  0.075234886 5.982240e-10
## SEM:25-COM:0    -0.07684088 -0.097492428 -0.056189322 0.000000e+00
## COM:50-COM:0     0.02966089  0.009009338  0.050312444 6.851273e-04
## SEM:50-COM:0    -0.07138002 -0.093686248 -0.049073795 0.000000e+00
## SEM:75-COM:0    -0.07360659 -0.098899471 -0.048313704 2.574385e-11
## COM:100-COM:0   -0.02490932 -0.045560870 -0.004257765 7.506813e-03
## SEM:100-COM:0   -0.10916214 -0.134455027 -0.083869260 0.000000e+00
## COM:25-SEM:0     0.13142421  0.110772656  0.152075762 0.000000e+00
## COM:50-SEM:0     0.10650177  0.085850214  0.127153319 0.000000e+00
## COM:75-SEM:0     0.09248711  0.071835560  0.113138666 0.000000e+00
## COM:100-SEM:0    0.05193156  0.031280005  0.072583111 2.629409e-09
## SEM:100-SEM:0   -0.03232127 -0.057614152 -0.007028385 3.651401e-03
## SEM:25-COM:25   -0.13142421 -0.152075762 -0.110772656 0.000000e+00
## COM:50-COM:25   -0.02492244 -0.045573995 -0.004270889 7.459906e-03
## SEM:50-COM:25   -0.12596335 -0.148269581 -0.103657128 0.000000e+00
## COM:75-COM:25   -0.03893710 -0.059588648 -0.018285542 4.145768e-06
## SEM:75-COM:25   -0.12818992 -0.153482805 -0.102897038 0.000000e+00
## COM:100-COM:25  -0.07949265 -0.100144204 -0.058841098 0.000000e+00
## SEM:100-COM:25  -0.16374548 -0.189038360 -0.138452593 0.000000e+00
## COM:50-SEM:25    0.10650177  0.085850214  0.127153319 0.000000e+00
## COM:75-SEM:25    0.09248711  0.071835560  0.113138666 0.000000e+00
## COM:100-SEM:25   0.05193156  0.031280005  0.072583111 2.629409e-09
## SEM:100-SEM:25  -0.03232127 -0.057614152 -0.007028385 3.651401e-03
## SEM:50-COM:50   -0.10104091 -0.123347139 -0.078734686 0.000000e+00
## SEM:75-COM:50   -0.10326748 -0.128560363 -0.077974596 0.000000e+00
## COM:100-COM:50  -0.05457021 -0.075221762 -0.033918656 6.026132e-10
## SEM:100-COM:50  -0.13882303 -0.164115918 -0.113530151 0.000000e+00
## COM:75-SEM:50    0.08702626  0.064720033  0.109332486 0.000000e+00
## COM:100-SEM:50   0.04647070  0.024164477  0.068776930 4.067938e-07
## SEM:100-SEM:50  -0.03778212 -0.064443162 -0.011121082 8.428060e-04
## SEM:75-COM:75   -0.08925283 -0.114545709 -0.063959943 0.000000e+00
## COM:100-COM:75  -0.04055556 -0.061207108 -0.019904003 1.655845e-06
## SEM:100-COM:75  -0.12480838 -0.150101265 -0.099515498 0.000000e+00
## COM:100-SEM:75   0.04869727  0.023404387  0.073990154 2.599212e-06
## SEM:100-SEM:75  -0.03555556 -0.064761262 -0.006349849 6.716443e-03
## SEM:100-COM:100 -0.08425283 -0.109545709 -0.058959943 0.000000e+00
##                            diif          lwr          upr        p_adj
## 1:SEM:0-1:COM:0     -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:SEM:0-1:COM:0     -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:COM:25-1:COM:0     0.10000000  0.067007546  0.132992454 0.000000e+00
## 1:SEM:25-1:COM:0    -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:SEM:25-1:COM:0    -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:COM:50-1:COM:0     0.07000000  0.037007546  0.102992454 4.259290e-08
## 1:SEM:50-1:COM:0    -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:SEM:50-1:COM:0    -0.07995098 -0.120358320 -0.039543641 2.799798e-07
## 1:SEM:75-1:COM:0    -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 2:COM:100-1:COM:0   -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 1:SEM:100-1:COM:0   -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 1:SEM:0-2:COM:0     -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 2:SEM:0-2:COM:0     -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 2:COM:25-2:COM:0     0.07666667  0.043674212  0.109659121 3.064612e-09
## 1:SEM:25-2:COM:0    -0.10333333 -0.136325788 -0.070340879 0.000000e+00
```

```
## 2:SEM:25-2:COM:0    -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 2:COM:50-2:COM:0     0.04666667  0.013674212  0.079659121 4.379113e-04
## 1:SEM:50-2:COM:0    -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 2:SEM:50-2:COM:0    -0.10328431 -0.143691653 -0.062876974 1.548851e-10
## 1:SEM:75-2:COM:0    -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 2:COM:100-2:COM:0   -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 1:SEM:100-2:COM:0   -0.10333333 -0.136325788 -0.070340879 0.000000e+00
## 1:COM:25-1:SEM:0     0.11250000  0.079507546  0.145492454 0.000000e+00
## 2:COM:25-1:SEM:0     0.18000000  0.147007546  0.212992454 0.000000e+00
## 1:COM:50-1:SEM:0     0.09000000  0.057007546  0.122992454 1.493949e-11
## 2:COM:50-1:SEM:0     0.15000000  0.117007546  0.182992454 0.000000e+00
## 1:COM:75-1:SEM:0     0.08666667  0.053674212  0.119659121 6.161749e-11
## 2:COM:75-1:SEM:0     0.11000000  0.077007546  0.142992454 0.000000e+00
## 1:COM:100-1:SEM:0    0.08000000  0.047007546  0.112992454 8.344928e-10
## 1:COM:25-2:SEM:0     0.11250000  0.079507546  0.145492454 0.000000e+00
## 2:COM:25-2:SEM:0     0.18000000  0.147007546  0.212992454 0.000000e+00
## 1:COM:50-2:SEM:0     0.09000000  0.057007546  0.122992454 1.493949e-11
## 2:COM:50-2:SEM:0     0.15000000  0.117007546  0.182992454 0.000000e+00
## 1:COM:75-2:SEM:0     0.08666667  0.053674212  0.119659121 6.161749e-11
## 2:COM:75-2:SEM:0     0.11000000  0.077007546  0.142992454 0.000000e+00
## 1:COM:100-2:SEM:0    0.08000000  0.047007546  0.112992454 8.344928e-10
## 2:COM:25-1:COM:25    0.06750000  0.034507546  0.100492454 1.153007e-07
## 1:SEM:25-1:COM:25   -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 2:SEM:25-1:COM:25   -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 2:COM:50-1:COM:25    0.03750000  0.004507546  0.070492454 1.194486e-02
## 1:SEM:50-1:COM:25   -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 2:SEM:50-1:COM:25   -0.11245098 -0.152858320 -0.072043641 5.758172e-12
## 1:SEM:75-1:COM:25   -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 2:COM:100-1:COM:25  -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 1:SEM:100-1:COM:25  -0.11250000 -0.145492454 -0.079507546 0.000000e+00
## 1:SEM:25-2:COM:25   -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 2:SEM:25-2:COM:25   -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 1:COM:50-2:COM:25   -0.09000000 -0.122992454 -0.057007546 1.493949e-11
## 1:SEM:50-2:COM:25   -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 2:SEM:50-2:COM:25   -0.17995098 -0.220358320 -0.139543641 0.000000e+00
## 1:COM:75-2:COM:25   -0.09333333 -0.126325788 -0.060340879 1.667222e-12
## 2:COM:75-2:COM:25   -0.07000000 -0.102992454 -0.037007546 4.259290e-08
## 1:SEM:75-2:COM:25   -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 1:COM:100-2:COM:25  -0.10000000 -0.132992454 -0.067007546 0.000000e+00
## 2:COM:100-2:COM:25  -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 1:SEM:100-2:COM:25  -0.18000000 -0.212992454 -0.147007546 0.000000e+00
## 1:COM:50-1:SEM:25    0.09000000  0.057007546  0.122992454 1.493949e-11
## 2:COM:50-1:SEM:25    0.15000000  0.117007546  0.182992454 0.000000e+00
## 1:COM:75-1:SEM:25    0.08666667  0.053674212  0.119659121 6.161749e-11
## 2:COM:75-1:SEM:25    0.11000000  0.077007546  0.142992454 0.000000e+00
## 1:COM:100-1:SEM:25   0.08000000  0.047007546  0.112992454 8.344928e-10
## 1:COM:50-2:SEM:25    0.09000000  0.057007546  0.122992454 1.493949e-11
## 2:COM:50-2:SEM:25    0.15000000  0.117007546  0.182992454 0.000000e+00
## 1:COM:75-2:SEM:25    0.08666667  0.053674212  0.119659121 6.161749e-11
## 2:COM:75-2:SEM:25    0.11000000  0.077007546  0.142992454 0.000000e+00
## 1:COM:100-2:SEM:25   0.08000000  0.047007546  0.112992454 8.344928e-10
## 2:COM:50-1:COM:50    0.06000000  0.027007546  0.092992454 2.317754e-06
## 1:SEM:50-1:COM:50   -0.09000000 -0.122992454 -0.057007546 1.493949e-11
## 2:SEM:50-1:COM:50   -0.08995098 -0.130358320 -0.049543641 1.088023e-08
```

```
## 1:SEM:75-1:COM:50   -0.09000000 -0.122992454 -0.057007546 1.493949e-11
## 2:COM:100-1:COM:50  -0.09000000 -0.122992454 -0.057007546 1.493949e-11
## 1:SEM:100-1:COM:50  -0.09000000 -0.122992454 -0.057007546 1.493949e-11
## 1:SEM:50-2:COM:50   -0.15000000 -0.182992454 -0.117007546 0.000000e+00
## 2:SEM:50-2:COM:50   -0.14995098 -0.190358320 -0.109543641 0.000000e+00
## 1:COM:75-2:COM:50   -0.06333333 -0.096325788 -0.030340879 6.101698e-07
## 2:COM:75-2:COM:50   -0.04000000 -0.072992454 -0.007007546 5.059017e-03
## 1:SEM:75-2:COM:50   -0.15000000 -0.182992454 -0.117007546 0.000000e+00
## 1:COM:100-2:COM:50  -0.07000000 -0.102992454 -0.037007546 4.259290e-08
## 2:COM:100-2:COM:50  -0.15000000 -0.182992454 -0.117007546 0.000000e+00
## 1:SEM:100-2:COM:50  -0.15000000 -0.182992454 -0.117007546 0.000000e+00
## 1:COM:75-1:SEM:50    0.08666667  0.053674212  0.119659121 6.161749e-11
## 2:COM:75-1:SEM:50    0.11000000  0.077007546  0.142992454 0.000000e+00
## 1:COM:100-1:SEM:50   0.08000000  0.047007546  0.112992454 8.344928e-10
## 1:COM:75-2:SEM:50    0.08661765  0.046210308  0.127024986 3.195907e-08
## 2:COM:75-2:SEM:50    0.10995098  0.069543641  0.150358320 1.658351e-11
## 1:COM:100-2:SEM:50   0.07995098  0.039543641  0.120358320 2.799798e-07
## 1:SEM:75-1:COM:75   -0.08666667 -0.119659121 -0.053674212 6.161749e-11
## 2:COM:100-1:COM:75  -0.08666667 -0.119659121 -0.053674212 6.161749e-11
## 1:SEM:100-1:COM:75  -0.08666667 -0.119659121 -0.053674212 6.161749e-11
## 1:SEM:75-2:COM:75   -0.11000000 -0.142992454 -0.077007546 0.000000e+00
## 2:COM:100-2:COM:75  -0.11000000 -0.142992454 -0.077007546 0.000000e+00
## 1:SEM:100-2:COM:75  -0.11000000 -0.142992454 -0.077007546 0.000000e+00
## 1:COM:100-1:SEM:75   0.08000000  0.047007546  0.112992454 8.344928e-10
## 2:COM:100-1:COM:100 -0.08000000 -0.112992454 -0.047007546 8.344928e-10
## 1:SEM:100-1:COM:100 -0.08000000 -0.112992454 -0.047007546 8.344928e-10
```

Análise para Cálcio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_6, aes(x = factor(EFLUENTE), y = Ca)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Cálcio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_6, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$Ca

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_6, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$Ca

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup
```

```r
# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_6 <- with(dados_6,
                       dados_6[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_6$Ca[which(blocos_dados_6$Ca <
                                  limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_6$Ca < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_6$Ca[which(blocos_dados_6$Ca >
                                  limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_6$Ca > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_6 = aggregate(Ca ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_6, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_6 = media_blocos_6[rep(row.names(media_blocos_6),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_6) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_6$Ca[which(is.na(blocos_dados_6$Ca))] =
  media_blocos_6$Ca[which(is.na(blocos_dados_6$Ca))]

# Análises Descritivas
str(blocos_dados_6)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Ca      : num [1:80] 35 29 27 32 37 29 33 29 34 34 ...
```

```
summary(blocos_dados_6)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Ca
##  1:20   0  :16   COM:40   1:40   Min.   :25.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:31.50
##  3:20   50 :16                   Median :36.00
##  4:20   75 :16                   Mean   :36.25
##         100:16                   3rd Qu.:39.00
##                                  Max.   :60.00
```

```
# Número de observações
with(blocos_dados_6, tapply(Ca, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_6, tapply(Ca, list(EFLUENTE, INOCULO), sum))
```

```
##          COM SEM
## 0   271.0000 307
## 25  288.0000 296
## 50  326.0000 284
## 75  298.0000 303
## 100 255.3333 272
```

```
with(blocos_dados_6, tapply(Ca, list(EFLUENTE, INOCULO), mean))
```

```
##          COM    SEM
## 0   33.87500 38.375
## 25  36.00000 37.000
## 50  40.75000 35.500
## 75  37.25000 37.875
## 100 31.91667 34.000
```

```
with(blocos_dados_6, tapply(Ca, list(EFLUENTE, INOCULO), var))
```

```
##          COM        SEM
## 0   21.26786  43.125000
## 25  31.42857   5.936508
## 50  83.64286  65.928571
## 75  58.50000 110.982143
## 100 36.62698  20.000000
```

```
with(blocos_dados_6, tapply(Ca, list(EFLUENTE, INOCULO), sd))
```

```
##          COM       SEM
## 0   4.611709  6.566963
## 25  5.606119  2.436495
## 50  9.145647  8.119641
## 75  7.648529 10.534806
## 100 6.052023  4.472136
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_6,
                 model.tables(aov(Ca ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 36.25417
##
##  CICLO
## CICLO
##     1     2
## 33.71 38.80
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 35.77 36.15 34.20 38.89
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 36.12 36.50 38.12 37.56 32.96
##
##  INOCULO
## INOCULO
##   COM   SEM
## 35.96 36.55
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 33.63 33.67 36.75 34.25 30.25
##     2 38.62 39.33 39.50 40.87 35.67
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 30.80 36.62
##     2 41.12 36.48
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM   SEM
##       0   33.88 38.38
##       25  36.00 37.00
##       50  40.75 35.50
##       75  37.25 37.87
##       100 31.92 34.00
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 30.75 32.00 34.00 30.75 26.50
##     2 37.00 40.00 47.50 43.75 37.33
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 36.50 35.33 39.50 37.75 34.00
##     2 40.25 38.67 31.50 38.00 34.00
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_6 = aggregate(Ca ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_6, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_6$Ca)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.9438712
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_6$Ca,
                                    media_blocos_6$EFLUENTE,
                                    media_blocos_6$INOCULO,
                                    media_blocos_6$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.8074116
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Ca ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_6)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  228.3   76.11
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  7.001   7.001
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  282.4   94.14
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  518.5   518.5  18.549 7.03e-05 ***
## EFLUENTE                  4  258.4    64.6   2.311   0.0694 .
## CICLO:INOCULO             1  546.0   546.0  19.533 4.81e-05 ***
## CICLO:EFLUENTE            4   33.1     8.3   0.296   0.8792
## INOCULO:EFLUENTE          4  207.2    51.8   1.853   0.1322
## CICLO:INOCULO:EFLUENTE    4  224.2    56.0   2.005   0.1068
## Residuals                54 1509.5    28.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Ca ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_6)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]
```

```r
# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: Ca
##               Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO          1 518.50  518.50  16.493 0.0001508 ***
## CICLO:INOCULO  1 546.01  546.01  17.368 0.0001057 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(Ca ~ CICLO + INOCULO,
                                 data = blocos_dados_6, FUN = mean)
```

```
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Ca ~ CICLO + EFLUENTE,
                                 data = blocos_dados_6, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Ca ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_6, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(Ca ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_6, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO       Ca
## 1     1     COM 30.80000
## 2     2     COM 41.11667
## 3     1     SEM 36.61667
## 4     2     SEM 36.48333
```

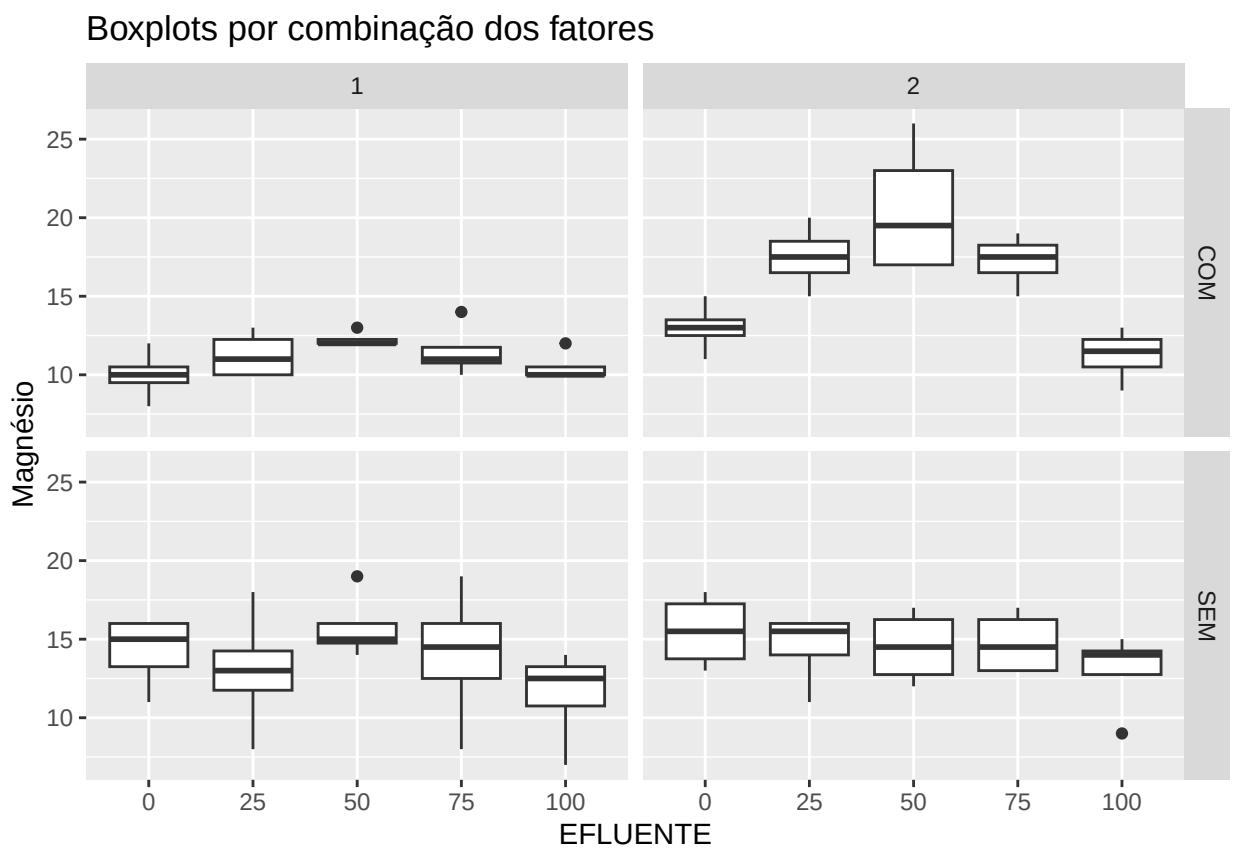
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                  diif       lwr      upr        p_adj
```

```
## 2:COM-1:COM 10.316667 5.6243258 15.00901 1.681086e-06
## 1:SEM-1:COM  5.816667 1.1243258 10.50901 9.326668e-03
## 2:SEM-1:COM  5.683333 0.9909924 10.37567 1.154359e-02
```

Análise para Magnésio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_7, aes(x = factor(EFLUENTE), y = Mg)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Magnésio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_7, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$Mg
```

```r
# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_7, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$Mg

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_7 <- with(dados_7,
                           dados_7[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_7$Mg[which(blocos_dados_7$Mg <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_7$Mg < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_7$Mg[which(blocos_dados_7$Mg >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_7$Mg > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_7 = aggregate(Mg ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_7, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_7 = media_blocos_7[rep(row.names(media_blocos_7),
                                                each = 4), ]

# Redefinir os índices das linhas
```

```
rownames(media_blocos_7) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_7$Mg[which(is.na(blocos_dados_7$Mg))] =
  media_blocos_7$Mg[which(is.na(blocos_dados_7$Mg))]
```

```
# Análises Descritivas
str(blocos_dados_7)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Mg      : num [1:80] 10 10 8 12 13 10 12 10 12 12 ...
```

```
summary(blocos_dados_7)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Mg
##  1:20   0  :16   COM:40   1:40   Min.   : 7.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:11.00
##  3:20   50 :16                   Median :13.00
##  4:20   75 :16                   Mean   :13.59
##         100:16                   3rd Qu.:15.25
##                                  Max.   :26.00
```

```
# Número de observações
with(blocos_dados_7, tapply(Mg, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_7, tapply(Mg, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0    92.0000 119.0000
## 25  115.0000 110.0000
## 50  128.0000 114.6667
## 75  118.6667 108.0000
## 100  88.0000  94.0000
```

```
with(blocos_dados_7, tapply(Mg, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   11.50000 14.87500
```

```
## 25  14.37500 13.75000
## 50  16.00000 14.33333
## 75  14.83333 13.50000
## 100 11.00000 11.75000
```

```
with(blocos_dados_7, tapply(Mg, list(EFLUENTE, INOCULO), var))
```

```
##           COM       SEM
## 0    4.857143  5.267857
## 25  13.982143 10.214286
## 50  26.000000  1.365079
## 75  19.936508  9.142857
## 100  1.428571  6.785714
```

```
with(blocos_dados_7, tapply(Mg, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   2.203893 2.295181
## 25  3.739270 3.195980
## 50  5.099020 1.168366
## 75  4.465032 3.023716
## 100 1.195229 2.604940
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_7,
                 model.tables(aov(Mg ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 13.59167
##
##  CICLO
## CICLO
##      1      2
## 12.133 15.050
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 13.583 13.800 12.750 14.233
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 13.187 14.062 15.167 14.167 11.375
##
##  INOCULO
```

```
## INOCULO
##    COM    SEM
## 13.542 13.642
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 12.125 12.125 13.333 12.333 10.750
##     2 14.250 16.000 17.000 16.000 12.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 10.783 13.483
##     2 16.300 13.800
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   11.500 14.875
##      25  14.375 13.750
##      50  16.000 14.333
##      75  14.833 13.500
##      100 11.000 11.750
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 10.000 11.250 12.000 10.667 10.000
##     2 13.000 17.500 20.000 19.000 12.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 14.250 13.000 14.667 14.000 11.500
##     2 15.500 14.500 14.000 13.000 12.000
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_7 = aggregate(Mg ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_7, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_7$Mg)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.0984742
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_7$Mg,
                                    media_blocos_7$EFLUENTE,
                                    media_blocos_7$INOCULO,
                                    media_blocos_7$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.3929326
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Mg ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_7)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  23.27   7.757
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1    0.2     0.2
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  22.88   7.626
##
## Error: Within
##                       Df Sum Sq Mean Sq F value   Pr(>F)
```

```
## CICLO                   1 170.14  170.14  34.711 2.54e-07 ***
## EFLUENTE                4 129.76   32.44   6.618 0.000207 ***
## CICLO:INOCULO           1 135.20  135.20  27.583 2.61e-06 ***
## CICLO:EFLUENTE          4  21.79    5.45   1.111 0.360681
## INOCULO:EFLUENTE        4  67.40   16.85   3.438 0.014178 *
## CICLO:INOCULO:EFLUENTE  4  54.90   13.72   2.800 0.034760 *
## Residuals              54 264.68    4.90
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Mg ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_7)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: Mg
##                         Df  Sum Sq Mean Sq F value    Pr(>F)
## CICLO                    1 170.139 170.139 33.7247 2.969e-07 ***
## EFLUENTE                 4 129.758  32.440  6.4301 0.0002423 ***
## CICLO:INOCULO            1 135.200 135.200 26.7992 3.057e-06 ***
## INOCULO:EFLUENTE         4  67.397  16.849  3.3398 0.0158747 *
## CICLO:INOCULO:EFLUENTE   4  54.897  13.724  2.7204 0.0383035 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"          "INOCULO:EFLUENTE"       "CICLO:INOCULO:EFLUENTE"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(Mg ~ CICLO + INOCULO,
                                 data = blocos_dados_7, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Mg ~ CICLO + EFLUENTE,
                                 data = blocos_dados_7, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Mg ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_7, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(Mg ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_7, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO       Mg
## 1     1     COM 10.78333
## 2     2     COM 16.30000
## 3     1     SEM 13.48333
## 4     2     SEM 13.80000
##    INOCULO EFLUENTE       Mg
## 1      COM        0 11.50000
## 2      SEM        0 14.87500
```

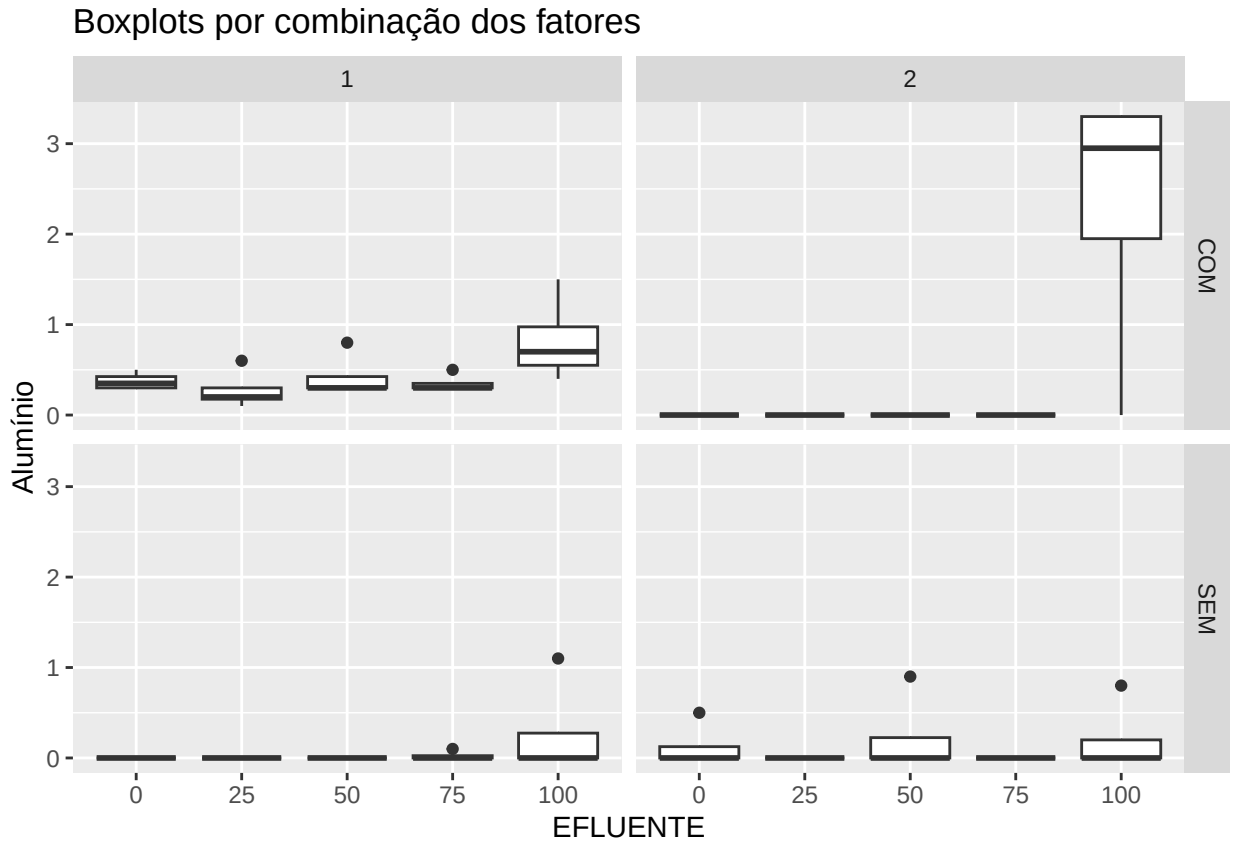
```
## 3      COM       25 14.37500
## 4      SEM       25 13.75000
## 5      COM       50 16.00000
## 6      SEM       50 14.33333
## 7      COM       75 14.83333
## 8      SEM       75 13.50000
## 9      COM      100 11.00000
## 10     SEM      100 11.75000
##    CICLO INOCULO EFLUENTE       Mg
## 1      1     COM        0 10.00000
## 2      2     COM        0 13.00000
## 3      1     SEM        0 14.25000
## 4      2     SEM        0 15.50000
## 5      1     COM       25 11.25000
## 6      2     COM       25 17.50000
## 7      1     SEM       25 13.00000
## 8      2     SEM       25 14.50000
## 9      1     COM       50 12.00000
## 10     2     COM       50 20.00000
## 11     1     SEM       50 14.66667
## 12     2     SEM       50 14.00000
## 13     1     COM       75 10.66667
## 14     2     COM       75 19.00000
## 15     1     SEM       75 14.00000
## 16     2     SEM       75 13.00000
## 17     1     COM      100 10.00000
## 18     2     COM      100 12.00000
## 19     1     SEM      100 11.50000
## 20     2     SEM      100 12.00000
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                   diif        lwr        upr        p_adj
## 2:COM-1:COM  5.516667  3.6369378  7.3963956 1.013226e-09
## 1:SEM-1:COM  2.700000  0.8202711  4.5797289 1.949103e-03
## 2:SEM-1:COM  3.016667  1.1369378  4.8963956 4.590364e-04
## 1:SEM-2:COM -2.816667 -4.6963956 -0.9369378 1.155803e-03
## 2:SEM-2:COM -2.500000 -4.3797289 -0.6202711 4.628720e-03
##                     diif        lwr        upr       p_adj
## COM:50-COM:0    4.500000  0.8029719  8.1970281 0.006429190
## COM:100-SEM:0  -3.875000 -7.5720281 -0.1779719 0.032752948
## COM:100-COM:50 -5.000000 -8.6970281 -1.3029719 0.001536651
## SEM:100-COM:50 -4.250000 -7.9470281 -0.5529719 0.012633648
## COM:100-COM:75 -3.833333 -7.5303614 -0.1363052 0.036227795
##                          diif         lwr        upr        p_adj
## 2:COM:25-1:COM:0     7.500000   1.6001974 13.3998026 2.307691e-03
## 2:COM:50-1:COM:0    10.000000   4.1001974 15.8998026 8.203145e-06
## 2:COM:75-1:COM:0     9.000000   3.1001974 14.8998026 8.422179e-05
## 2:COM:50-2:COM:0     7.000000   1.1001974 12.8998026 6.441210e-03
## 2:COM:75-2:COM:0     6.000000   0.1001974 11.8998026 4.209586e-02
## 2:COM:25-1:COM:25    6.250000   0.3501974 12.1498026 2.702048e-02
## 2:COM:50-1:COM:25    8.750000   2.8501974 14.6498026 1.489495e-04
## 2:COM:75-1:COM:25    7.750000   1.8501974 13.6498026 1.357868e-03
## 1:COM:75-2:COM:25   -6.833333 -12.7331360 -0.9335307 8.965189e-03
## 1:COM:100-2:COM:25  -7.500000 -13.3998026 -1.6001974 2.307691e-03
## 1:SEM:100-2:COM:25  -6.000000 -11.8998026 -0.1001974 4.209586e-02
## 2:COM:50-1:SEM:25    7.000000   1.1001974 12.8998026 6.441210e-03
## 2:COM:75-1:SEM:25    6.000000   0.1001974 11.8998026 4.209586e-02
## 2:COM:50-1:COM:50    8.000000   2.1001974 13.8998026 7.911421e-04
## 2:COM:75-1:COM:50    7.000000   1.1001974 12.8998026 6.441210e-03
## 2:SEM:50-2:COM:50   -6.000000 -11.8998026 -0.1001974 4.209586e-02
## 1:COM:75-2:COM:50   -9.333333 -15.2331360 -3.4335307 3.904968e-05
## 1:SEM:75-2:COM:50   -6.000000 -11.8998026 -0.1001974 4.209586e-02
## 2:SEM:75-2:COM:50   -7.000000 -12.8998026 -1.1001974 6.441210e-03
## 1:COM:100-2:COM:50 -10.000000 -15.8998026 -4.1001974 8.203145e-06
## 2:COM:100-2:COM:50  -8.000000 -13.8998026 -2.1001974 7.911421e-04
## 1:SEM:100-2:COM:50  -8.500000 -14.3998026 -2.6001974 2.617892e-04
## 2:SEM:100-2:COM:50  -8.000000 -13.8998026 -2.1001974 7.911421e-04
## 2:COM:75-1:COM:75    8.333333   2.4335307 14.2331360 3.798007e-04
## 2:SEM:75-2:COM:75   -6.000000 -11.8998026 -0.1001974 4.209586e-02
## 1:COM:100-2:COM:75  -9.000000 -14.8998026 -3.1001974 8.422179e-05
## 2:COM:100-2:COM:75  -7.000000 -12.8998026 -1.1001974 6.441210e-03
## 1:SEM:100-2:COM:75  -7.500000 -13.3998026 -1.6001974 2.307691e-03
## 2:SEM:100-2:COM:75  -7.000000 -12.8998026 -1.1001974 6.441210e-03
```

Análise para Alumínio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_8, aes(x = factor(EFLUENTE), y = Al)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Alumínio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Al ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_8, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$Al

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Al ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_8, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$Al

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup
```

```r
# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_8 <- with(dados_8,
                       dados_8[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_8$Al[which(blocos_dados_8$Al <
                                  limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_8$Al < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_8$Al[which(blocos_dados_8$Al >
                                  limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_8$Al > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_8 = aggregate(Al ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_8, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_8 = media_blocos_8[rep(row.names(media_blocos_8),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_8) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_8$Al[which(is.na(blocos_dados_8$Al))] =
  media_blocos_8$Al[which(is.na(blocos_dados_8$Al))]
```

```r
# Análises Descritivas
str(blocos_dados_8)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Al      : num [1:80] 0.3 0.4 0.5 0.3 0.167 ...
```

```
summary(blocos_dados_8)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Al
##  1:20   0  :16   COM:40   1:40   Min.   :0.0000
##  2:20   25 :16   SEM:40   2:40   1st Qu.:0.0000
##  3:20   50 :16                   Median :0.0000
##  4:20   75 :16                   Mean   :0.1046
##         100:16                   3rd Qu.:0.1167
##                                  Max.   :1.5000
```

```
# Número de observações
with(blocos_dados_8, tapply(Al, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_8, tapply(Al, list(EFLUENTE, INOCULO), sum))
```

```
##           COM SEM
## 0   1.5000000 0.5
## 25  0.6666667 0.0
## 50  1.2000000 0.0
## 75  1.2000000 0.0
## 100 3.3000000 0.0
```

```
with(blocos_dados_8, tapply(Al, list(EFLUENTE, INOCULO), mean))
```

```
##            COM    SEM
## 0   0.18750000 0.0625
## 25  0.08333333 0.0000
## 50  0.15000000 0.0000
## 75  0.15000000 0.0000
## 100 0.41250000 0.0000
```

```
with(blocos_dados_8, tapply(Al, list(EFLUENTE, INOCULO), var))
```

```
##             COM     SEM
## 0   0.044107143 0.03125
## 25  0.008888889 0.00000
## 50  0.025714286 0.00000
## 75  0.025714286 0.00000
## 100 0.292678571 0.00000
```

```
with(blocos_dados_8, tapply(Al, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.2100170 0.1767767
## 25  0.0942809 0.0000000
## 50  0.1603567 0.0000000
## 75  0.1603567 0.0000000
## 100 0.5409978 0.0000000
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_8,
                 model.tables(aov(Al ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 0.1045833
##
##  CICLO
## CICLO
##       1       2
## 0.19667 0.01250
##
##  BLOCO
## BLOCO
##       1       2       3       4
## 0.07333 0.13500 0.12000 0.09000
##
##  EFLUENTE
## EFLUENTE
##       0      25      50      75     100
## 0.12500 0.04167 0.07500 0.07500 0.20625
##
##  INOCULO
## INOCULO
##     COM     SEM
## 0.19667 0.01250
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 0.1875 0.0833 0.1500 0.1500 0.4125
##     2 0.0625 0.0000 0.0000 0.0000 0.0000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 0.3933 0.0000
##     2 0.0000 0.0250
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   0.1875 0.0625
##       25  0.0833 0.0000
##       50  0.1500 0.0000
##       75  0.1500 0.0000
##       100 0.4125 0.0000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 0.3750 0.1667 0.3000 0.3000 0.8250
##     2 0.0000 0.0000 0.0000 0.0000 0.0000
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 0.0000 0.0000 0.0000 0.0000 0.0000
##     2 0.1250 0.0000 0.0000 0.0000 0.0000
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_8 = aggregate(Al ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_8, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_8$Al)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 2.074116e-06
```

```
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados não seguem uma distribuição normal."
```

```
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_8$Al,
                                    media_blocos_8$EFLUENTE,
                                    media_blocos_8$INOCULO,
                                    media_blocos_8$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.1060526
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Al ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_8)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df  Sum Sq Mean Sq
## BLOCO  3 0.04704 0.01568
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.6783  0.6783
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.05371  0.0179
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1 0.6783  0.6783  45.312 1.12e-08 ***
## EFLUENTE                  4 0.2634  0.0658   4.398  0.00377 **
## CICLO:INOCULO             1 0.8750  0.8750  58.449 3.70e-10 ***
## CICLO:EFLUENTE            4 0.2726  0.0681   4.551  0.00306 **
## INOCULO:EFLUENTE          4 0.2726  0.0681   4.551  0.00306 **
## CICLO:INOCULO:EFLUENTE    4 0.2634  0.0658   4.398  0.00377 **
## Residuals                54 0.8084  0.0150
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Al ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_8)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]
```

```r
# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: Al
##                         Df  Sum Sq Mean Sq F value    Pr(>F)
## CICLO                    1 0.67835 0.67835 44.8494 1.016e-08 ***
## INOCULO                  1 0.67835 0.67835 44.8494 1.016e-08 ***
## EFLUENTE                 4 0.26339 0.06585  4.3535  0.003846 **
## CICLO:INOCULO            1 0.87501 0.87501 57.8522 3.108e-10 ***
## CICLO:EFLUENTE           4 0.27256 0.06814  4.5051  0.003122 **
## INOCULO:EFLUENTE         4 0.27256 0.06814  4.5051  0.003122 **
## CICLO:INOCULO:EFLUENTE   4 0.26339 0.06585  4.3535  0.003846 **
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"          "CICLO:EFLUENTE"         "INOCULO:EFLUENTE"
## [4] "CICLO:INOCULO:EFLUENTE"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
```

```
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(Al ~ CICLO + INOCULO,
                                 data = blocos_dados_8, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Al ~ CICLO + EFLUENTE,
                                 data = blocos_dados_8, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Al ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_8, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(Al ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_8, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO        Al
## 1     1     COM 0.3933333
## 2     2     COM 0.0000000
## 3     1     SEM 0.0000000
## 4     2     SEM 0.0250000
##    CICLO EFLUENTE         Al
## 1      1        0 0.18750000
## 2      2        0 0.06250000
## 3      1       25 0.08333333
## 4      2       25 0.00000000
## 5      1       50 0.15000000
## 6      2       50 0.00000000
## 7      1       75 0.15000000
## 8      2       75 0.00000000
## 9      1      100 0.41250000
## 10     2      100 0.00000000
##    INOCULO EFLUENTE         Al
## 1      COM        0 0.18750000
## 2      SEM        0 0.06250000
## 3      COM       25 0.08333333
## 4      SEM       25 0.00000000
## 5      COM       50 0.15000000
## 6      SEM       50 0.00000000
## 7      COM       75 0.15000000
## 8      SEM       75 0.00000000
```

```
## 9      COM      100 0.41250000
## 10     SEM      100 0.00000000
##    CICLO INOCULO EFLUENTE        Al
## 1      1     COM        0 0.3750000
## 2      2     COM        0 0.0000000
## 3      1     SEM        0 0.0000000
## 4      2     SEM        0 0.1250000
## 5      1     COM       25 0.1666667
## 6      2     COM       25 0.0000000
## 7      1     SEM       25 0.0000000
## 8      2     SEM       25 0.0000000
## 9      1     COM       50 0.3000000
## 10     2     COM       50 0.0000000
## 11     1     SEM       50 0.0000000
## 12     2     SEM       50 0.0000000
## 13     1     COM       75 0.3000000
## 14     2     COM       75 0.0000000
## 15     1     SEM       75 0.0000000
## 16     2     SEM       75 0.0000000
## 17     1     COM      100 0.8250000
## 18     2     COM      100 0.0000000
## 19     1     SEM      100 0.0000000
## 20     2     SEM      100 0.0000000
```

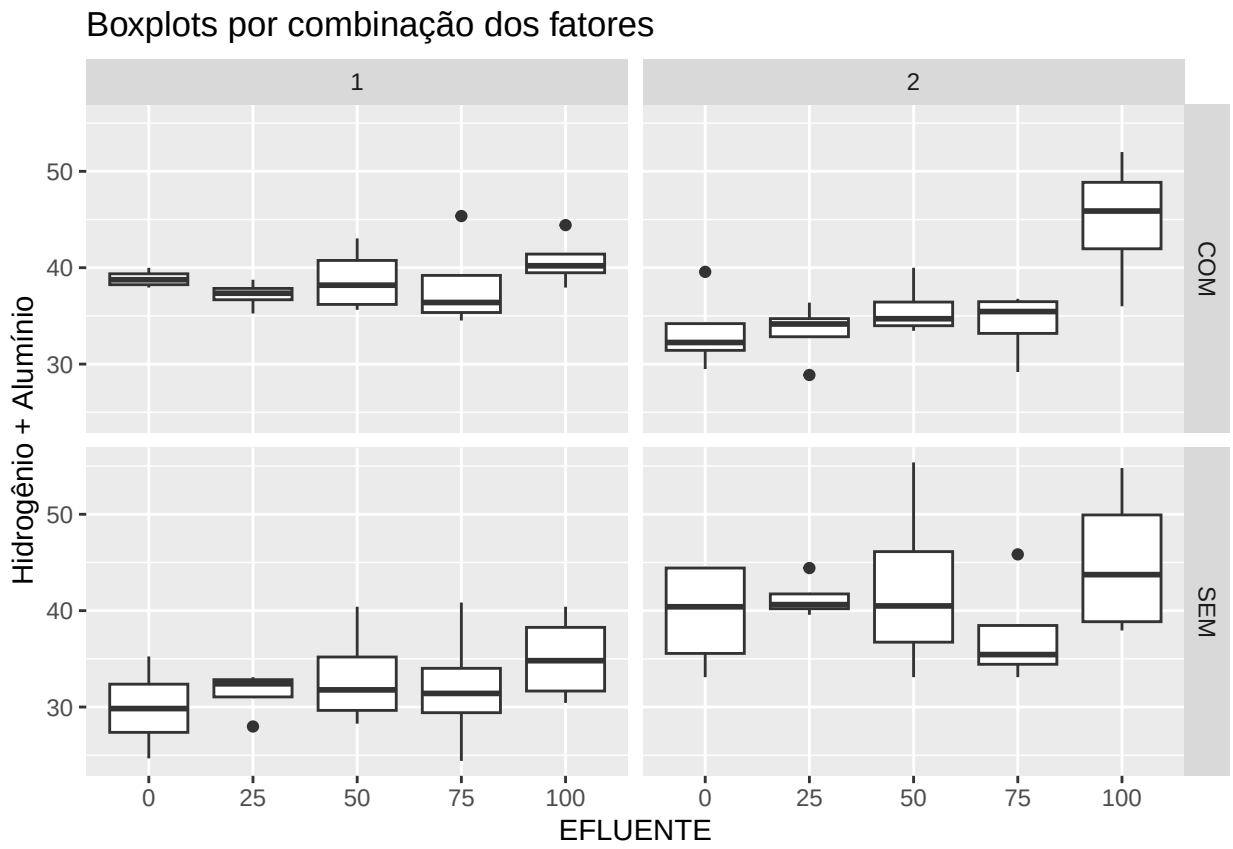
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                   diif       lwr        upr        p_adj
## 2:COM-1:COM -0.3933333 -0.496257 -0.2904097 1.138944e-11
## 1:SEM-1:COM -0.3933333 -0.496257 -0.2904097 1.138944e-11
## 2:SEM-1:COM -0.3683333 -0.471257 -0.2654097 1.283229e-11
##                   diif         lwr        upr        p_adj
## 1:100-1:0    0.2250000  0.02257097  0.4274290 1.822901e-02
```

```
## 1:100-2:0    0.3500000  0.14757097  0.5524290 1.934811e-05
## 1:100-1:25   0.3291667  0.12673764  0.5315957 6.670497e-05
## 1:100-2:25   0.4125000  0.21007097  0.6149290 4.246997e-07
## 1:100-1:50   0.2625000  0.06007097  0.4649290 2.802991e-03
## 1:100-2:50   0.4125000  0.21007097  0.6149290 4.246997e-07
## 1:100-1:75   0.2625000  0.06007097  0.4649290 2.802991e-03
## 1:100-2:75   0.4125000  0.21007097  0.6149290 4.246997e-07
## 2:100-1:100 -0.4125000 -0.61492903 -0.2100710 4.246997e-07
##                       diif         lwr        upr        p_adj
## COM:100-COM:0    0.2250000  0.02257097  0.4274290 1.822901e-02
## COM:100-SEM:0    0.3500000  0.14757097  0.5524290 1.934811e-05
## COM:100-COM:25   0.3291667  0.12673764  0.5315957 6.670497e-05
## COM:100-SEM:25   0.4125000  0.21007097  0.6149290 4.246997e-07
## COM:100-COM:50   0.2625000  0.06007097  0.4649290 2.802991e-03
## COM:100-SEM:50   0.4125000  0.21007097  0.6149290 4.246997e-07
## COM:100-COM:75   0.2625000  0.06007097  0.4649290 2.802991e-03
## COM:100-SEM:75   0.4125000  0.21007097  0.6149290 4.246997e-07
## SEM:100-COM:100 -0.4125000 -0.61492903 -0.2100710 4.246997e-07
##                           diif        lwr         upr        p_adj
## 2:COM:0-1:COM:0     -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:SEM:0-1:COM:0     -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:COM:25-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:SEM:25-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:SEM:25-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:COM:50-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:SEM:50-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:SEM:50-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:COM:75-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:SEM:75-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:SEM:75-1:COM:0    -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:COM:100-1:COM:0    0.4500000  0.1269591  0.77304091 4.898436e-04
## 2:COM:100-1:COM:0   -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:SEM:100-1:COM:0   -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 2:SEM:100-1:COM:0   -0.3750000 -0.6980409 -0.05195909 8.697475e-03
## 1:COM:100-2:COM:0    0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:SEM:0    0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-2:SEM:0    0.7000000  0.3769591  1.02304091 1.050576e-08
## 1:COM:100-1:COM:25   0.6583333  0.3352924  0.98137425 6.522201e-08
## 1:COM:100-2:COM:25   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:SEM:25   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-2:SEM:25   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:COM:50   0.5250000  0.2019591  0.84804091 2.157309e-05
## 1:COM:100-2:COM:50   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:SEM:50   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-2:SEM:50   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:COM:75   0.5250000  0.2019591  0.84804091 2.157309e-05
## 1:COM:100-2:COM:75   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-1:SEM:75   0.8250000  0.5019591  1.14804091 5.816425e-11
## 1:COM:100-2:SEM:75   0.8250000  0.5019591  1.14804091 5.816425e-11
## 2:COM:100-1:COM:100 -0.8250000 -1.1480409 -0.50195909 5.816425e-11
## 1:SEM:100-1:COM:100 -0.8250000 -1.1480409 -0.50195909 5.816425e-11
## 2:SEM:100-1:COM:100 -0.8250000 -1.1480409 -0.50195909 5.816425e-11
```

Análise para Hidrogênio + Alumínio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_9, aes(x = factor(EFLUENTE), y = H_AL)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Hidrogênio + Alumínio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(H_AL ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_9, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$H_AL

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(H_AL ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_9, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
```

```
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$H_AL

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_9 <- with(dados_9,
                       dados_9[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_9$H_AL[which(blocos_dados_9$H_AL <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_9$H_AL < limites_outliers$LIM_INF: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```
blocos_dados_9$H_AL[which(blocos_dados_9$H_AL >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_9$H_AL > limites_outliers$LIM_SUP: comprimento do
## objeto maior não é múltiplo do comprimento do objeto menor
```

```
# Calcular a média para cada grupo de 4 linhas
media_blocos_9 = aggregate(H_AL ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_9, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_9 = media_blocos_9[rep(row.names(media_blocos_9),
                                    each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_9) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_9$H_AL[which(is.na(blocos_dados_9$H_AL))] =
  media_blocos_9$H_AL[which(is.na(blocos_dados_9$H_AL))]
```

```r
# Análises Descritivas
str(blocos_dados_9)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ H_AL    : num [1:80] 39.2 37.9 40 38.3 35.2 ...
```

```r
summary(blocos_dados_9)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       H_AL
##  1:20   0  :16   COM:40   1:40   Min.   :24.41
##  2:20   25 :16   SEM:40   2:40   1st Qu.:33.19
##  3:20   50 :16                   Median :36.58
##  4:20   75 :16                   Mean   :36.43
##         100:16                   3rd Qu.:39.54
##                                  Max.   :52.00
##                                  NA's   :2
```

```r
# Número de observações
with(blocos_dados_9, tapply(H_AL, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```r
with(blocos_dados_9, tapply(H_AL, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   280.7233 271.4667
## 25  288.3000 295.8000
## 50  297.8900 294.2000
## 75  279.9167 282.2700
## 100 333.7200       NA
```

```r
with(blocos_dados_9, tapply(H_AL, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   35.09042 33.93333
## 25  36.03750 36.97500
## 50  37.23625 36.77500
## 75  34.98958 35.28375
## 100 41.71500       NA
```

```r
with(blocos_dados_9, tapply(H_AL, list(EFLUENTE, INOCULO), var))
```

```
##           COM      SEM
## 0   17.301497 36.98601
## 25   2.852050 23.51671
## 50  11.415313 29.89086
## 75   6.425259 31.71623
## 100 24.665375       NA
```

```r
with(blocos_dados_9, tapply(H_AL, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   4.159507 6.081613
## 25  1.688801 4.849403
## 50  3.378655 5.467253
## 75  2.534809 5.631716
## 100 4.966425       NA
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_9,
                 model.tables(aov(H_AL ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 36.43393
##
##  CICLO
##         1     2
##     35.27 37.66
## rep 40.00 38.00
##
##  BLOCO
##         1     2     3     4
##     36.77 36.71 37.17 35.11
## rep 19.00 19.00 20.00 20.00
##
##  EFLUENTE
##         0    25    50   75   100
##     34.47 36.47 36.97 35.1 39.55
## rep 16.00 16.00 16.00 16.0 14.00
##
##  INOCULO
##       COM   SEM
##     36.89 35.95
## rep 40.00 38.00
##
##  CICLO:EFLUENTE
```

```
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   34.36 34.89 35.90 33.88 37.34
##   rep  8.00  8.00  8.00  8.00  8.00
##   2   34.64 38.10 38.10 36.38 42.18
##   rep  8.00  8.00  8.00  8.00  6.00
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##   1   38.04 32.51
##   rep 20.00 20.00
##   2   35.81 39.71
##   rep 20.00 18.00
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   35.17 33.77
##      rep  8.00  8.00
##      25  36.12 36.82
##      rep  8.00  8.00
##      50  37.32 36.62
##      rep  8.00  8.00
##      75  35.07 35.13
##      rep  8.00  8.00
##      100 41.20 37.36
##      rep  8.00  6.00
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   38.86 37.17 38.76 35.77 39.45
##   rep  4.00  4.00  4.00  4.00  4.00
##   2   31.31 34.88 35.70 34.20 43.97
##   rep  4.00  4.00  4.00  4.00  4.00
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   29.90 32.64 33.06 32.02 35.11
##   rep  4.00  4.00  4.00  4.00  4.00
##   2   37.95 41.29 40.47 38.53 38.83
##   rep  4.00  4.00  4.00  4.00  2.00
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_9 = aggregate(H_AL ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_9, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_9$H_AL)$p.value
```

```r
# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.9604473
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_9$H_AL,
                                    media_blocos_9$EFLUENTE,
                                    media_blocos_9$INOCULO,
                                    media_blocos_9$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.949529
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(H_AL ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_9)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3   48.1   16.04
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1  26.97   26.97
##
```

```
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## CICLO      1  16.51  16.511    1.67  0.325
## Residuals  2  19.78   9.889
##
## Error: Within
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                    1  111.5   111.5   9.650  0.00307 **
## EFLUENTE                 4  232.3    58.1   5.028  0.00168 **
## CICLO:INOCULO            1  426.9   426.9  36.955 1.44e-07 ***
## CICLO:EFLUENTE           4   62.2    15.6   1.346  0.26531
## INOCULO:EFLUENTE         4   39.2     9.8   0.848  0.50154
## CICLO:INOCULO:EFLUENTE   4  115.7    28.9   2.503  0.05338 .
## Residuals               52  600.7    11.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(H_AL ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_9)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: H_AL
##                         Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO                    1 111.88  111.88  9.5577  0.003124 **
## EFLUENTE                 4 225.06   56.26  4.8064  0.002135 **
## CICLO:INOCULO            1 419.64  419.64 35.8483 1.686e-07 ***
## CICLO:INOCULO:EFLUENTE   4 125.58   31.40  2.6820  0.040860 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
```

```
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"          "CICLO:INOCULO:EFLUENTE"
```

```
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(H_AL ~ CICLO + INOCULO,
                                 data = blocos_dados_9, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(H_AL ~ CICLO + EFLUENTE,
                                 data = blocos_dados_9, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(H_AL ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_9, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(H_AL ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_9, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO     H_AL
```

```
## 1     1     COM 38.00117
## 2     2     COM 36.02633
## 3     1     SEM 32.54650
## 4     2     SEM 39.46481
##    CICLO INOCULO EFLUENTE     H_AL
## 1      1     COM        0 38.85750
## 2      2     COM        0 31.32333
## 3      1     SEM        0 29.90000
## 4      2     SEM        0 37.96667
## 5      1     COM       25 37.17500
## 6      2     COM       25 34.90000
## 7      1     SEM       25 32.64000
## 8      2     SEM       25 41.31000
## 9      1     COM       50 38.76000
## 10     2     COM       50 35.71250
## 11     1     SEM       50 33.06000
## 12     2     SEM       50 40.49000
## 13     1     COM       75 35.76667
## 14     2     COM       75 34.21250
## 15     1     SEM       75 32.01750
## 16     2     SEM       75 38.55000
## 17     1     COM      100 39.44667
## 18     2     COM      100 43.98333
## 19     1     SEM      100 35.11500
## 20     2     SEM      100 38.55000
```

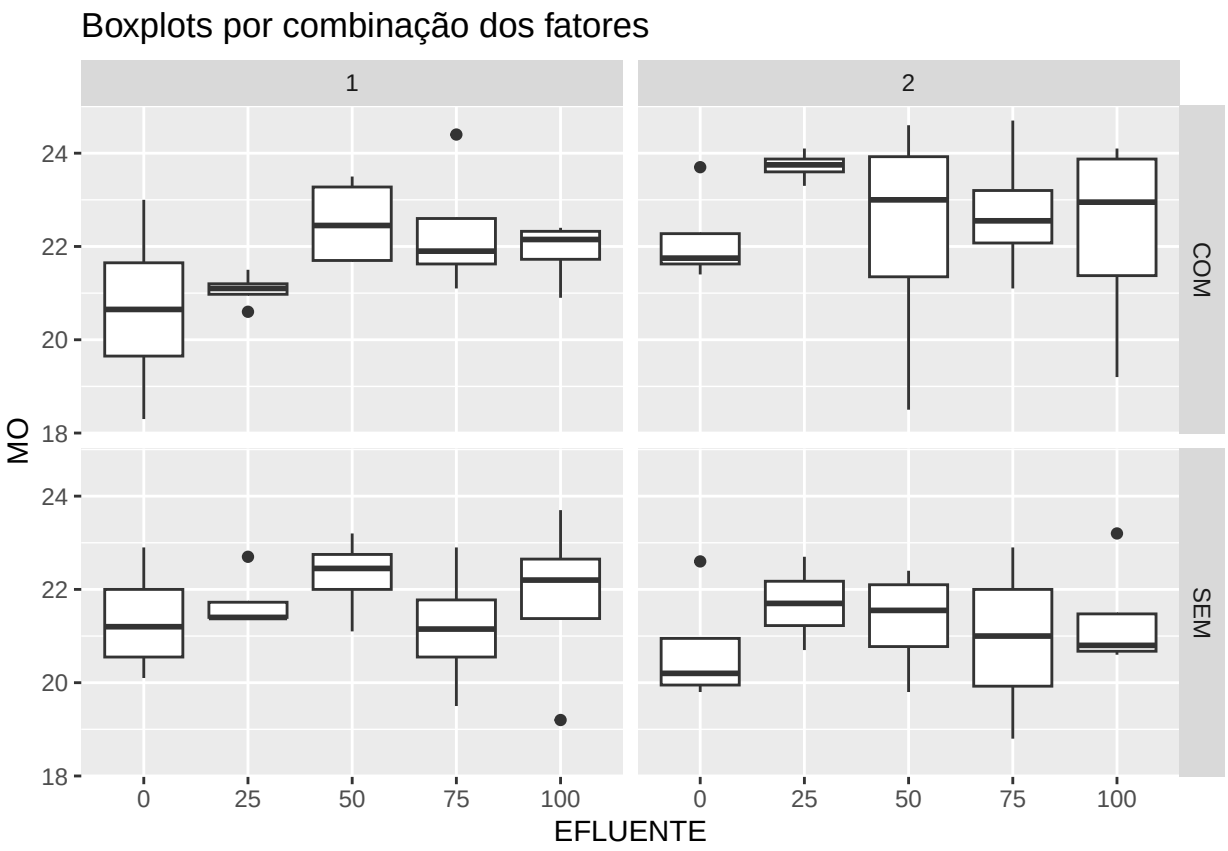
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
##                   diif        lwr        upr        p_adj
## 1:SEM-1:COM -5.621458 -8.4879214 -2.7549941 1.795799e-05
## 1:SEM-2:COM -3.479833 -6.3462970 -0.6133697 1.135974e-02
## 2:SEM-2:COM  3.644171  0.6991591  6.5891820 9.535124e-03
```

```
## 2:SEM-1:SEM  7.124004  4.1789925 10.0690153 2.072508e-07
##                          diif          lwr        upr        p_adj
## 2:SEM:25-2:COM:0    9.986667   0.98445156 18.9888818 1.574477e-02
## 2:SEM:50-2:COM:0    9.166667   0.16445156 18.1688818 4.158460e-02
## 2:COM:100-2:COM:0  12.660000   3.65778490 21.6622151 4.273874e-04
## 2:SEM:25-1:SEM:0   11.410000   2.40778490 20.4122151 2.468471e-03
## 2:SEM:50-1:SEM:0   10.590000   1.58778490 19.5922151 7.342477e-03
## 1:COM:100-1:SEM:0   9.546667   0.54445156 18.5488818 2.678807e-02
## 2:COM:100-1:SEM:0  14.083333   5.08111823 23.0855484 5.273816e-05
## 2:COM:100-2:COM:25  9.083333   0.08111823 18.0855484 4.567685e-02
## 2:COM:100-1:SEM:25 11.343333   2.34111823 20.3455484 2.702665e-03
## 1:SEM:75-2:SEM:25  -9.292500 -18.29471510 -0.2902849 3.602453e-02
## 2:COM:100-1:SEM:50 10.923333   1.92111823 19.9255484 4.746118e-03
## 2:COM:100-2:COM:75  9.770833   0.76861823 18.7730484 2.049172e-02
## 2:COM:100-1:SEM:75 11.965833   2.96361823 20.9680484 1.145378e-03
```

Análise para MO

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_10, aes(x = factor(EFLUENTE), y = MO)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "MO") +
  ggtitle("Boxplots por combinação dos fatores")
```



```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(MO ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_10, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$MO

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(MO ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_10, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$MO

# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_10 <- with(dados_10,
                            dados_10[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_10$MO[which(blocos_dados_10$MO <
                                    limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_10$MO < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_10$MO[which(blocos_dados_10$MO >
                                    limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_10$MO > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_10 = aggregate(MO ~ EFLUENTE + INOCULO + CICLO,
                            data = blocos_dados_10, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_10 = media_blocos_10[rep(row.names(media_blocos_10),
                                      each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_10) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_10$MO[which(is.na(blocos_dados_10$MO))] =
  media_blocos_10$MO[which(is.na(blocos_dados_10$MO))]
```

```r
# Análises Descritivas
str(blocos_dados_10)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MO      : num [1:80] 23 21.2 18.3 20.1 21.5 ...
```

```r
summary(blocos_dados_10)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        MO
##  1:20   0  :16   COM:40   1:40   Min.   :18.30
##  2:20   25 :16   SEM:40   2:40   1st Qu.:21.05
##  3:20   50 :16                   Median :21.67
##  4:20   75 :16                   Mean   :21.72
##         100:16                   3rd Qu.:22.40
##                                  Max.   :24.70
```

```r
# Número de observações
with(blocos_dados_10, tapply(MO, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```r
with(blocos_dados_10, tapply(MO, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   169.0000 168.2000
```

```
## 25  179.8333 171.0667
## 50  179.2000 174.5000
## 75  177.4333 173.9000
## 100 170.2000 174.0000
```

```
with(blocos_dados_10, tapply(MO, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   21.12500 21.02500
## 25  22.47917 21.38333
## 50  22.40000 21.81250
## 75  22.17917 21.73750
## 100 21.27500 21.75000
```

```
with(blocos_dados_10, tapply(MO, list(EFLUENTE, INOCULO), var))
```

```
##          COM       SEM
## 0   1.933571 1.4792857
## 25  1.835853 0.1212698
## 50  3.511429 1.1755357
## 75  1.353948 1.3083929
## 100 1.250000 1.2514286
```

```
with(blocos_dados_10, tapply(MO, list(EFLUENTE, INOCULO), sd))
```

```
##          COM       SEM
## 0   1.390529 1.2162589
## 25  1.354937 0.3482382
## 50  1.873881 1.0842212
## 75  1.163593 1.1438500
## 100 1.118034 1.1186727
```

```
# Tamanho das Médias
T.medias <- with(blocos_dados_10,
                 model.tables(aov(MO ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 21.71667
##
##  CICLO
## CICLO
##      1      2
## 21.687 21.747
##
##  BLOCO
```

```
## BLOCO
##      1      2      3      4
## 22.032 21.689 21.503 21.643
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 21.075 21.931 22.106 21.958 21.513
##
##  INOCULO
## INOCULO
##    COM    SEM
## 21.892 21.542
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 21.000 21.317 22.413 21.404 22.300
##     2 21.150 22.546 21.800 22.513 20.725
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 21.588 21.785
##     2 22.195 21.298
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   21.125 21.025
##      25  22.479 21.383
##      50  22.400 21.813
##      75  22.179 21.738
##      100 21.275 21.750
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 20.650 21.233 22.525 21.633 21.900
##     2 21.600 23.725 22.275 22.725 20.650
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 21.350 21.400 22.300 21.175 22.700
##     2 20.700 21.367 21.325 22.300 20.800
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_10 = aggregate(MO ~ EFLUENTE + INOCULO + CICLO,
                            data = blocos_dados_10, FUN = mean)
```

```r
# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_10$MO)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.2032523
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_10$MO,
                                    media_blocos_10$EFLUENTE,
                                    media_blocos_10$INOCULO,
                                    media_blocos_10$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.5250788
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(MO ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_10)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  3.025   1.008
##
## Error: INOCULO
```

```
##         Df Sum Sq Mean Sq
## INOCULO  1   2.45    2.45
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  7.294   2.431
##
## Error: Within
##                          Df Sum Sq Mean Sq F value  Pr(>F)
## CICLO                     1   0.07   0.072   0.061 0.80606
## EFLUENTE                  4  11.35   2.839   2.400 0.06121 .
## CICLO:INOCULO             1   5.98   5.977   5.053 0.02869 *
## CICLO:EFLUENTE            4  22.40   5.600   4.734 0.00239 **
## INOCULO:EFLUENTE          4   5.46   1.364   1.153 0.34165
## CICLO:INOCULO:EFLUENTE    4   3.91   0.977   0.826 0.51443
## Residuals                54  63.87   1.183
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(MO ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_10)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)

# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: MO
##                 Df  Sum Sq Mean Sq F value  Pr(>F)
## CICLO:INOCULO    1  5.9769  5.9769  4.7872 0.03279 *
## CICLO:EFLUENTE   4 22.3981  5.5995  4.4849 0.00321 **
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
```

```
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"  "CICLO:EFLUENTE"
```

```
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(MO ~ CICLO + INOCULO,
                                 data = blocos_dados_10, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(MO ~ CICLO + EFLUENTE,
                                 data = blocos_dados_10, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(MO ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_10, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(MO ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_10, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO       MO
```

```
## 1     1     COM 21.58833
## 2     2     COM 22.19500
## 3     1     SEM 21.78500
## 4     2     SEM 21.29833
##    CICLO EFLUENTE       MO
## 1      1        0 21.00000
## 2      2        0 21.15000
## 3      1       25 21.31667
## 4      2       25 22.54583
## 5      1       50 22.41250
## 6      2       50 21.80000
## 7      1       75 21.40417
## 8      2       75 22.51250
## 9      1      100 22.30000
## 10     2      100 20.72500
```

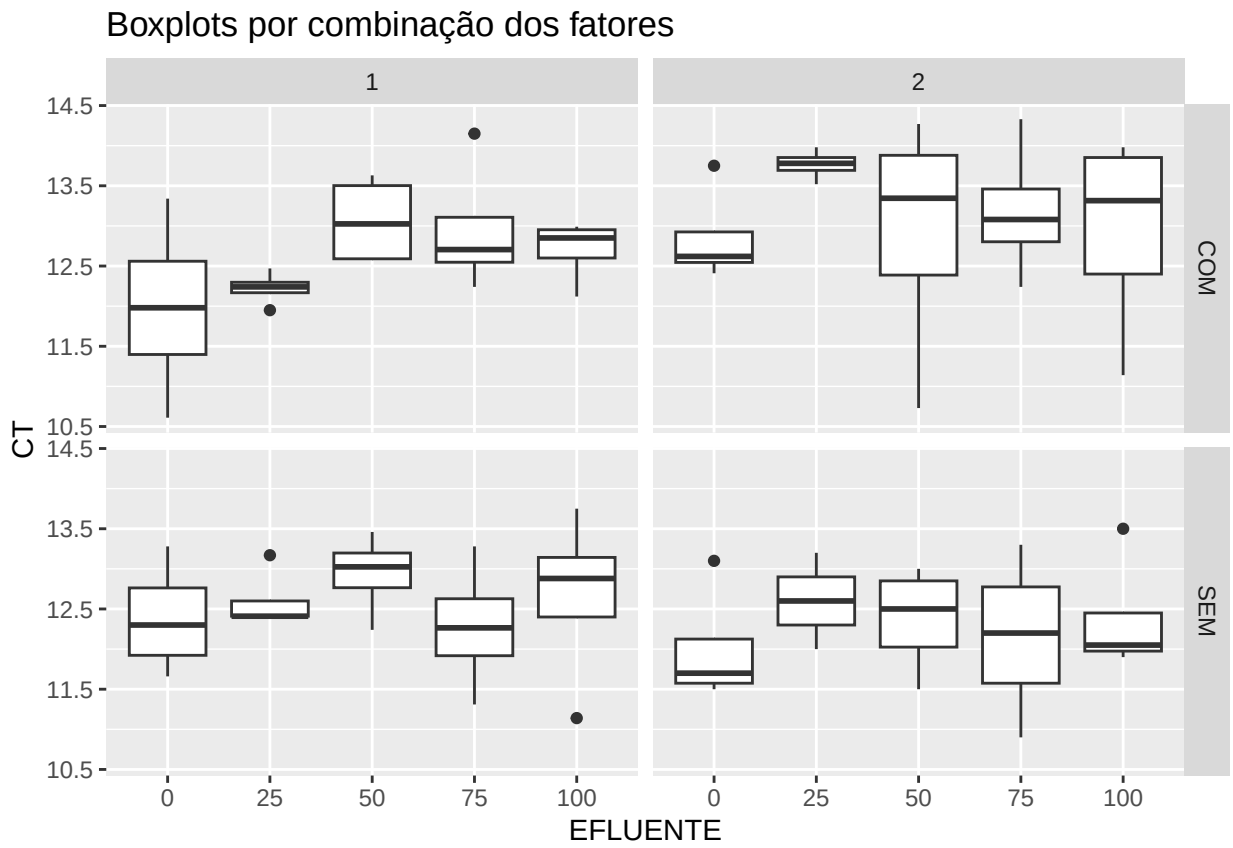
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
## [1] diif  lwr   upr   p_adj
## <0 linhas> (ou row.names de comprimento 0)
## [1] diif  lwr   upr   p_adj
## <0 linhas> (ou row.names de comprimento 0)
```

Análise para CT

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_11, aes(x = factor(EFLUENTE), y = CT)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
```

```
  labs(x = "EFLUENTE", y = "CT") +
  ggtitle("Boxplots por combinação dos fatores")
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(CT ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_11, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

# Armazenar vetor com os limites inferiores
lim_inf = limites_outliers$CT

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(CT ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_11, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

# Armazenar vetor com os limites superiores
lim_sup = limites_outliers$CT
```

```r
# Montar Dataframe com os limites inferior e superior
limites_outliers = limites_outliers[c(1:3)]
limites_outliers$LIM_INF = lim_inf
limites_outliers$LIM_SUP = lim_sup

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# Redefinir os índices das linhas
rownames(limites_outliers) <- NULL

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_11 <- with(dados_11,
                        dados_11[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_11$CT[which(blocos_dados_11$CT <
                                   limites_outliers$LIM_INF)] = NA
```

```
## Warning in blocos_dados_11$CT < limites_outliers$LIM_INF: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
blocos_dados_11$CT[which(blocos_dados_11$CT >
                                   limites_outliers$LIM_SUP)] = NA
```

```
## Warning in blocos_dados_11$CT > limites_outliers$LIM_SUP: comprimento do objeto
## maior não é múltiplo do comprimento do objeto menor
```

```r
# Calcular a média para cada grupo de 4 linhas
media_blocos_11 = aggregate(CT ~ EFLUENTE + INOCULO + CICLO,
                            data = blocos_dados_11, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_11 = media_blocos_11[rep(row.names(media_blocos_11),
                                      each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_11) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_11$CT[which(is.na(blocos_dados_11$CT))] =
  media_blocos_11$CT[which(is.na(blocos_dados_11$CT))]
```

```r
# Análises Descritivas
str(blocos_dados_11)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
```

```
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CT      : num [1:80] 13.3 12.3 10.6 11.7 12.5 ...
```

```
summary(blocos_dados_11)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        CT
##  1:20   0  :16   COM:40   1:40   Min.   :10.61
##  2:20   25 :16   SEM:40   2:40   1st Qu.:12.18
##  3:20   50 :16                   Median :12.57
##  4:20   75 :16                   Mean   :12.60
##         100:16                   3rd Qu.:13.03
##                                  Max.   :14.33
```

```
# Número de observações
with(blocos_dados_11, tapply(CT, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

```
with(blocos_dados_11, tapply(CT, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0    98.0300  97.54
## 25  104.3267  99.24
## 50  103.9600 101.25
## 75  103.1867 100.92
## 100  98.7300 100.88
```

```
with(blocos_dados_11, tapply(CT, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   12.25375 12.19250
## 25  13.04083 12.40500
## 50  12.99500 12.65625
## 75  12.89833 12.61500
## 100 12.34125 12.61000
```

```
with(blocos_dados_11, tapply(CT, list(EFLUENTE, INOCULO), var))
```

```
##           COM        SEM
## 0   0.6529411 0.49510714
## 25  0.6198722 0.04574286
## 50  1.1823429 0.39851250
## 75  0.4733937 0.44660000
## 100 0.4195554 0.43222857
```

```r
with(blocos_dados_11, tapply(CT, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.8080477 0.7036385
## 25  0.7873196 0.2138758
## 50  1.0873559 0.6312785
## 75  0.6880361 0.6682814
## 100 0.6477309 0.6574409
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_11,
                 model.tables(aov(CT ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 12.60079
##
##  CICLO
## CICLO
##      1      2
## 12.580 12.622
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 12.782 12.577 12.473 12.571
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 12.223 12.723 12.826 12.757 12.476
##
##  INOCULO
## INOCULO
##    COM    SEM
## 12.706 12.496
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 12.181 12.363 13.003 12.415 12.936
##     2 12.265 13.083 12.649 13.098 12.015
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 12.523 12.637
##     2 12.889 12.355
```

```
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   12.254 12.193
##      25  13.041 12.405
##      50  12.995 12.656
##      75  12.898 12.615
##      100 12.341 12.610
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.978 12.317 13.068 12.550 12.703
##     2 12.530 13.765 12.923 13.247 11.980
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 12.385 12.410 12.938 12.280 13.170
##     2 12.000 12.400 12.375 12.950 12.050
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_11 = aggregate(CT ~ EFLUENTE + INOCULO + CICLO,
                            data = blocos_dados_11, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_11$CT)$p.value

# Exibir o resultado do teste Shapiro-Wilk
print(resultado_shapiro)
```

```
## [1] 0.2188331
```

```r
# Interpretação do p-valor
if (resultado_shapiro > 0.05) {
  print("Os dados seguem uma distribuição normal.")
} else {
  print("Os dados não seguem uma distribuição normal.")
}
```

```
## [1] "Os dados seguem uma distribuição normal."
```

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_11$CT,
                                    media_blocos_11$EFLUENTE,
                                    media_blocos_11$INOCULO,
                                    media_blocos_11$CICLO)$p.value

# Exibir o resultado do teste de Bartlett
print(resultado_bartlett)
```

```
## [1] 0.5381843
```

```r
# Interpretação do p-valor
if (resultado_bartlett > 0.05) {
  print("A variância é homogênea entre os grupos.")
} else {
  print("A variância não é homogênea entre os grupos.")
}
```

```
## [1] "A variância é homogênea entre os grupos."
```

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(CT ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_11)

# Visualizar os resultados do modelo
summary(modelo_PARCELASUB)
```

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  1.012  0.3374
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.8827  0.8827
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.497  0.8323
##
## Error: Within
##                         Df Sum Sq Mean Sq F value  Pr(>F)
## CICLO                    1  0.036  0.0357   0.090 0.76552
## EFLUENTE                 4  3.969  0.9922   2.497 0.05334 .
## CICLO:INOCULO            1  2.096  2.0963   5.276 0.02553 *
## CICLO:EFLUENTE           4  7.824  1.9561   4.923 0.00186 **
## INOCULO:EFLUENTE         4  1.818  0.4546   1.144 0.34576
## CICLO:INOCULO:EFLUENTE   4  1.242  0.3106   0.782 0.54204
## Residuals               54 21.456  0.3973
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(CT ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_11)

# Realizar a análise de variância (ANOVA)
anova_result <- anova(modelo)
```

```r
# Filtrar apenas os fatores significativos ao nível 0.05
fatores_significativos <- anova_result[anova_result$"Pr(>F)" < 0.05, ]

# Exibir as interações significativas
print(fatores_significativos)
```

```
## Analysis of Variance Table
##
## Response: CT
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO:INOCULO    1 2.0963  2.0963  4.9884 0.029459 *
## CICLO:EFLUENTE   4 7.8243  1.9561  4.6548 0.002544 **
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Vetor com os fatores significativos
nomes_linhas = row.names(fatores_significativos)

# Filtrar apenas as interações
interacoes_significativas <- nomes_linhas[nchar(nomes_linhas) > 8]

# Exibir o resultado
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "CICLO:INOCULO"  "CICLO:EFLUENTE"
```

```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
```

```
    media_interacao <- aggregate(CT ~ CICLO + INOCULO,
                                 data = blocos_dados_11, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(CT ~ CICLO + EFLUENTE,
                                 data = blocos_dados_11, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(CT ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_11, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(CT ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_11, FUN = mean)
    print(media_interacao)
  }
}
```

```
##   CICLO INOCULO       CT
## 1     1     COM 12.52283
## 2     2     COM 12.88883
## 3     1     SEM 12.63650
## 4     2     SEM 12.35500
##    CICLO EFLUENTE       CT
## 1      1        0 12.18125
## 2      2        0 12.26500
## 3      1       25 12.36333
## 4      2       25 13.08250
## 5      1       50 13.00250
## 6      2       50 12.64875
## 7      1       75 12.41500
## 8      2       75 13.09833
## 9      1      100 12.93625
## 10     2      100 12.01500
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
```

```
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  inter_tukey = tukey_result[interacao]
  inter_tukey = data.frame(inter_tukey)
  colnames(inter_tukey) = c('diif','lwr','upr','p_adj')
  inter_tukey = subset(inter_tukey, p_adj < 0.05)
  print(inter_tukey)
}
```

```
## [1] diif  lwr   upr   p_adj
## <0 linhas> (ou row.names de comprimento 0)
##                 diif       lwr           upr      p_adj
## 2:100-2:25 -1.067500 -2.134510 -0.0004896809 0.04980357
## 2:100-2:75 -1.083333 -2.150344 -0.0163230142 0.04380675
```

```
# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_1, blocos_dados_2["P_resina"],
                    blocos_dados_3["S"], blocos_dados_4["K_resina"],
                    blocos_dados_5["Na"], blocos_dados_6["Ca"],
                    blocos_dados_7["Mg"], blocos_dados_8["Al"],
                    blocos_dados_9["H_AL"], blocos_dados_10["MO"],
                    blocos_dados_11["CT"])


# Criar planilha com todos os dados atualizados
library("xlsx")
```

```
## Warning: package 'xlsx' was built under R version 4.3.1
```

```
write.xlsx(dados_final, file = "Solo 0-10 atualizado.xlsx",
      sheetName = "R - Solo", append = FALSE)
```