Solo 10-20

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 10-20 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] 5.2 5 5.3 5.2 5 4.8 5 4.9 5.1 4.7 ...
##  $ P resina: num [1:80] 28 21 21 31 19 9 13 13 21 25 ...
##  $ S       : num [1:80] 0 8 0 7 2 9 3 8 2 14 ...
##  $ K resina: num [1:80] 1.4 1.8 0.7 1.8 1.4 1.5 1.5 1.3 1.4 1.3 ...
##  $ Na      : num [1:80] 0.08 0 0.11 0 0.09 0 0.08 0 0.08 0 ...
##  $ Ca      : num [1:80] 31 28 34 31 29 20 27 25 28 23 ...
##  $ Mg      : num [1:80] 8 11 12 11 11 13 10 12 11 7 ...
##  $ Al      : num [1:80] 0.2 0.5 0.1 0 0.6 0.4 0.8 0.5 0.2 0.8 ...
##  $ H+Al    : num [1:80] 39.6 40.8 35.6 36 42.1 ...
##  $ MO      : num [1:80] 22 21.7 18.8 21.7 21.5 19.7 22 23.3 20.7 22.3 ...
##  $ CT      : num [1:80] 12.8 12.6 10.9 12.6 12.5 ...
```

```
# 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] 5.2 5 5.3 5.2 5 4.8 5 4.9 5.1 4.7 ...
```

```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] 28 21 21 31 19 9 13 13 21 25 ...
```

```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] 0 8 0 7 2 9 3 8 2 14 ...
```

```
"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] 1.4 1.8 0.7 1.8 1.4 1.5 1.5 1.3 1.4 1.3 ...
```

```
"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 0.11 0 0.09 0 0.08 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] 31 28 34 31 29 20 27 25 28 23 ...
```

```
"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] 8 11 12 11 11 13 10 12 11 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] 0.2 0.5 0.1 0 0.6 0.4 0.8 0.5 0.2 0.8 ...
```

```
"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] 39.6 40.8 35.6 36 42.1 ...
```

```
"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] 22 21.7 18.8 21.7 21.5 19.7 22 23.3 20.7 22.3 ...
```

```
"dados_11"
```

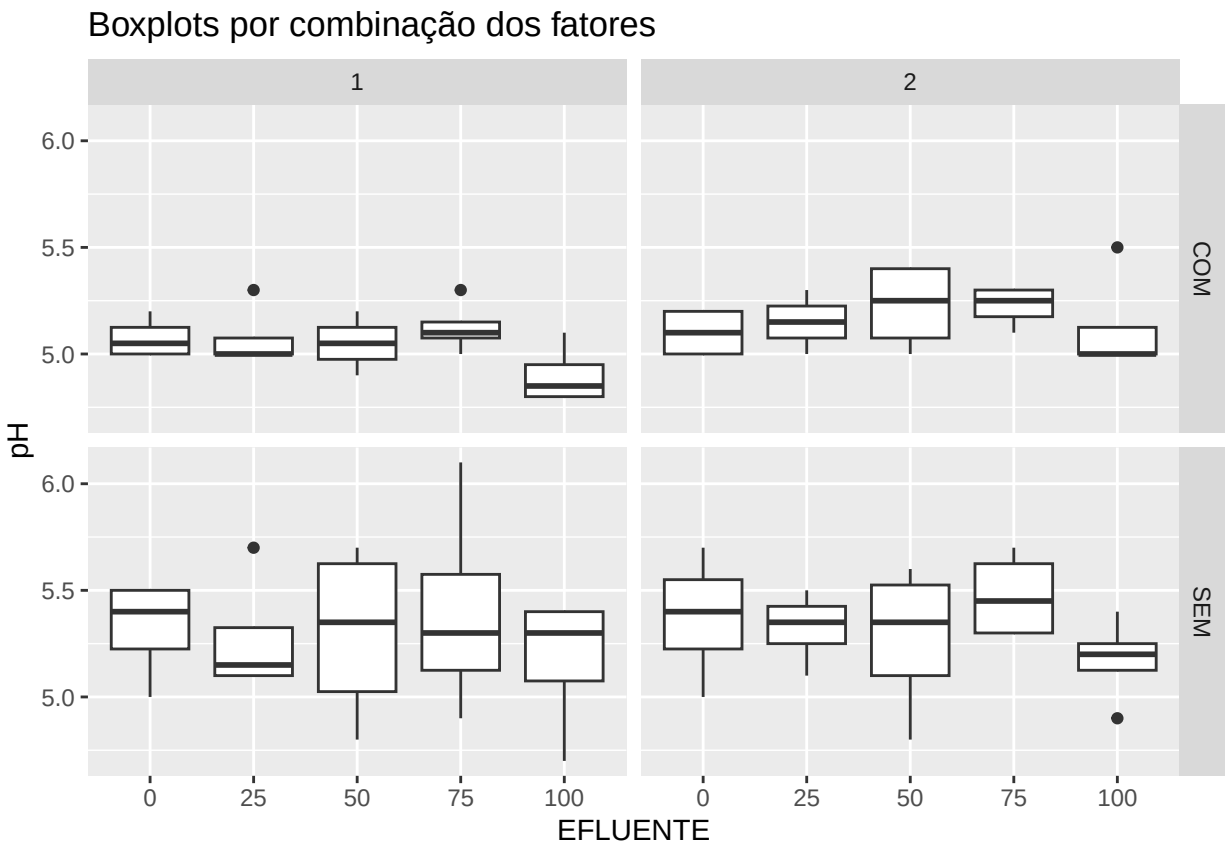
```
## [1] "dados_11"
```

```
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] 12.8 12.6 10.9 12.6 12.5 ...
```

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] 5.2 5.1 5 5 5 5 5 5 5 5.1 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        pH
##  1:20   0  :16   COM:40   1:40   Min.   :4.700
##  2:20   25 :16   SEM:40   2:40   1st Qu.:5.000
##  3:20   50 :16                   Median :5.100
##  4:20   75 :16                   Mean   :5.165
##         100:16                   3rd Qu.:5.300
##                                  Max.   :6.100
```

```
# 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   40.70000 42.80000
## 25  40.60000 41.83333
## 50  41.10000 41.20000
## 75  41.16667 42.80000
## 100 40.10000 40.90000
```

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

```
##          COM      SEM
## 0   5.087500 5.350000
## 25  5.075000 5.229167
## 50  5.137500 5.150000
## 75  5.145833 5.350000
## 100 5.012500 5.112500
```

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

```
##             COM        SEM
## 0   0.009821429 0.06285714
## 25  0.013571429 0.02394841
## 50  0.034107143 0.11428571
## 75  0.012043651 0.11428571
## 100 0.049821429 0.05767857
```

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

```
##            COM       SEM
## 0   0.09910312 0.2507133
## 25  0.11649647 0.1547527
## 50  0.18468119 0.3380617
## 75  0.10974357 0.3380617
## 100 0.22320714 0.2401636
```

```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
##
## 5.165
##
##  CICLO
## CICLO
##     1     2
## 5.142 5.187
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 5.120 5.172 5.115 5.252
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 5.219 5.152 5.144 5.248 5.062
##
##  INOCULO
## INOCULO
##   COM   SEM
## 5.092 5.238
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.200 5.067 5.175 5.233 5.037
##     2 5.237 5.237 5.112 5.262 5.087
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 5.018 5.267
##     2 5.165 5.210
```

```
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   5.087 5.350
##      25  5.075 5.229
##      50  5.137 5.150
##      75  5.146 5.350
##      100 5.012 5.112
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.075 5.000 5.050 5.067 4.900
##     2 5.100 5.150 5.225 5.225 5.125
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.325 5.133 5.300 5.400 5.175
##     2 5.375 5.325 5.000 5.300 5.050
```

```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] 0.6344173
```

```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] 0.9965844
```

```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 0.2447 0.08158
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.4302  0.4302
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.5401    0.18
##
## Error: Within
##                          Df Sum Sq Mean Sq F value Pr(>F)
## CICLO                     1 0.0405 0.04050   1.045 0.3113
## EFLUENTE                  4 0.3342 0.08356   2.155 0.0865 .
## CICLO:INOCULO             1 0.2067 0.20672   5.332 0.0248 *
## CICLO:EFLUENTE            4 0.1109 0.02772   0.715 0.5853
## INOCULO:EFLUENTE          4 0.1478 0.03696   0.953 0.4406
## CICLO:INOCULO:EFLUENTE    4 0.2105 0.05262   1.357 0.2609
## Residuals                54 2.0935 0.03877
## ---
## 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 0.43022 0.43022  9.3115 0.003454 **
## CICLO:INOCULO   1 0.20672 0.20672  4.4742 0.038793 *
## 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)
  }
}
```

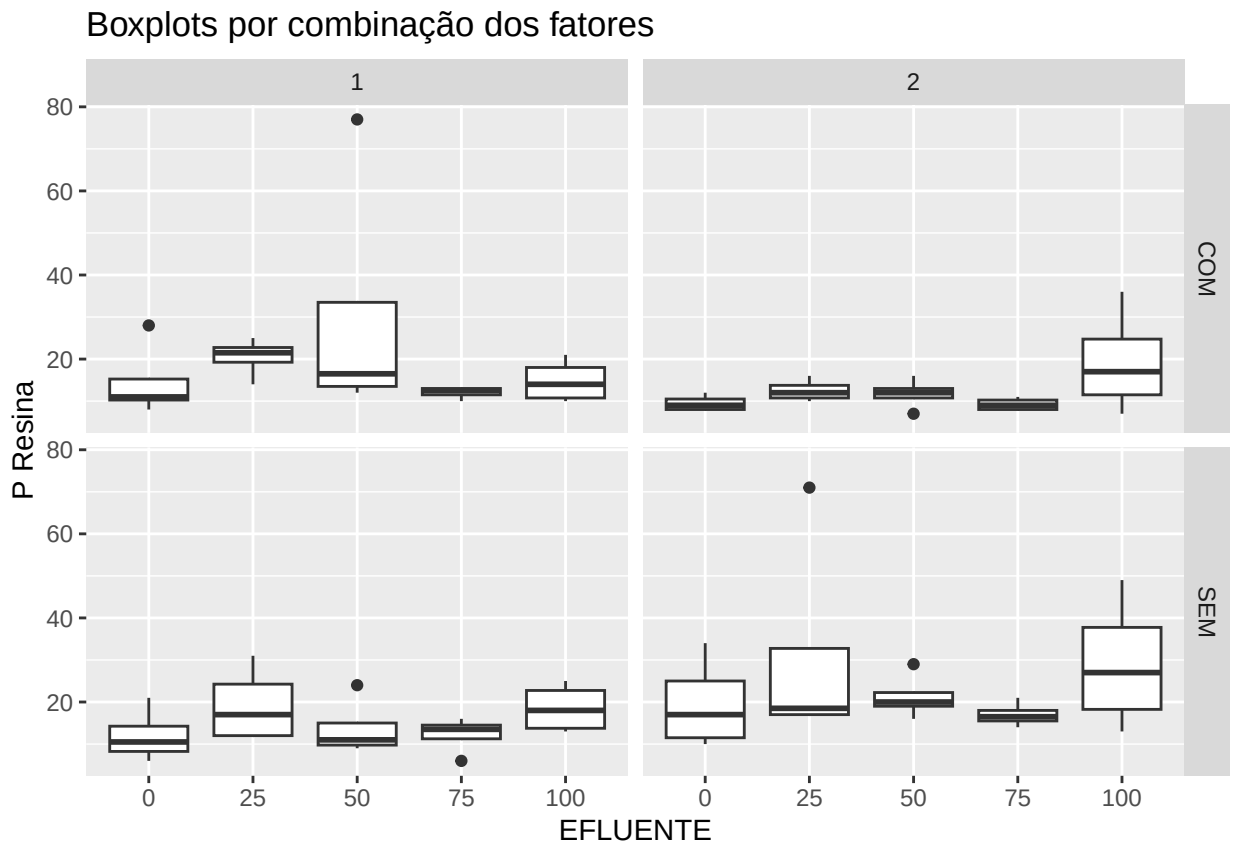
```
##   CICLO INOCULO       pH
## 1     1     COM 5.018333
## 2     2     COM 5.165000
## 3     1     SEM 5.266667
## 4     2     SEM 5.210000
```

```
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 0.2483333 0.06844472 0.4282219 0.003085869
## 2:SEM-1:COM 0.1916667 0.01177805 0.3715553 0.032506040
```

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] 10 11 8 11 21 14 22 25 19 15 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     P_resina
##  1:20   0  :16   COM:40   1:40   Min.   : 6.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:11.00
##  3:20   50 :16                   Median :13.50
##  4:20   75 :16                   Mean   :15.28
##         100:16                   3rd Qu.:19.25
##                                  Max.   :36.00
```

```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    78.0000 160.0000
## 25  132.0000 149.0000
## 50  101.3333 116.0000
## 75   85.0000 125.3333
## 100 136.0000 140.0000
```

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

```
##          COM      SEM
## 0    9.75000 20.00000
```

```
## 25  16.50000 18.62500
## 50  12.66667 14.50000
## 75  10.62500 15.66667
## 100 17.00000 17.50000
```

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

```
##           COM        SEM
## 0    2.500000 101.428571
## 25  30.571429  37.125000
## 50  12.317460  22.031746
## 75   3.982143   6.412698
## 100 84.857143  19.642857
```

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

```
##          COM       SEM
## 0   1.581139 10.071175
## 25  5.529144  6.093029
## 50  3.509624  4.693799
## 75  1.995531  2.532331
## 100 9.211794  4.432026
```

```
# 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
##
## 15.28333
##
##  CICLO
## CICLO
##      1      2
## 14.667 15.900
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 18.975 15.125 13.583 13.450
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 14.875 17.563 13.583 13.146 17.250
##
##  INOCULO
```

```
## INOCULO
##    COM    SEM
## 13.308 17.258
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.000 19.875 12.667 13.167 16.625
##     2 18.750 15.250 14.500 13.125 17.875
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 14.450 14.883
##     2 12.167 19.633
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    9.750 20.000
##      25  16.500 18.625
##      50  12.667 14.500
##      75  10.625 15.667
##      100 17.000 17.500
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 10.000 20.500 15.000 12.000 14.750
##     2  9.500 12.500 10.333  9.250 19.250
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 12.000 19.250 10.333 14.333 18.500
##     2 28.000 18.000 18.667 17.000 16.500
```

```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] 0.1053229
```

```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] 0.1607771
```

```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  398.1   132.7
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  312.1   312.1
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  4.175   1.392
##
## Error: Within
##                       Df Sum Sq Mean Sq F value   Pr(>F)
```

```
## CICLO                   1   30.4   30.42   1.748 0.191721
## EFLUENTE                4  267.0   66.75   3.835 0.008155 **
## CICLO:INOCULO           1  247.3  247.34  14.210 0.000407 ***
## CICLO:EFLUENTE          4  315.1   78.77   4.526 0.003168 **
## INOCULO:EFLUENTE        4  242.4   60.60   3.481 0.013336 *
## CICLO:INOCULO:EFLUENTE  4  311.1   77.77   4.468 0.003427 **
## Residuals              54  939.9   17.41
## ---
## 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                    3 398.09 132.697  8.0118 0.0001525 ***
## INOCULO                  1 312.05 312.050 18.8405 5.885e-05 ***
## EFLUENTE                 4 267.01  66.752  4.0303 0.0060195 **
## CICLO:INOCULO            1 247.34 247.339 14.9335 0.0002876 ***
## CICLO:EFLUENTE           4 315.09  78.773  4.7560 0.0022156 **
## INOCULO:EFLUENTE         4 242.38  60.595  3.6585 0.0101262 *
## CICLO:INOCULO:EFLUENTE   4 311.06  77.766  4.6952 0.0024070 **
## 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(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)
  }
}
```

```
##   CICLO INOCULO P_resina
## 1     1     COM 14.45000
## 2     2     COM 12.16667
## 3     1     SEM 14.88333
## 4     2     SEM 19.63333
```

```
##    CICLO EFLUENTE P_resina
## 1      1        0 11.00000
## 2      2        0 18.75000
## 3      1       25 19.87500
## 4      2       25 15.25000
## 5      1       50 12.66667
## 6      2       50 14.50000
## 7      1       75 13.16667
## 8      2       75 13.12500
## 9      1      100 16.62500
## 10     2      100 17.87500
##    INOCULO EFLUENTE P_resina
## 1      COM        0  9.75000
## 2      SEM        0 20.00000
## 3      COM       25 16.50000
## 4      SEM       25 18.62500
## 5      COM       50 12.66667
## 6      SEM       50 14.50000
## 7      COM       75 10.62500
## 8      SEM       75 15.66667
## 9      COM      100 17.00000
## 10     SEM      100 17.50000
##    CICLO INOCULO EFLUENTE P_resina
## 1      1     COM        0 10.00000
## 2      2     COM        0  9.50000
## 3      1     SEM        0 12.00000
## 4      2     SEM        0 28.00000
## 5      1     COM       25 20.50000
## 6      2     COM       25 12.50000
## 7      1     SEM       25 19.25000
## 8      2     SEM       25 18.00000
## 9      1     COM       50 15.00000
## 10     2     COM       50 10.33333
## 11     1     SEM       50 10.33333
## 12     2     SEM       50 18.66667
## 13     1     COM       75 12.00000
## 14     2     COM       75  9.25000
## 15     1     SEM       75 14.33333
## 16     2     SEM       75 17.00000
## 17     1     COM      100 14.75000
## 18     2     COM      100 19.25000
## 19     1     SEM      100 18.50000
## 20     2     SEM      100 16.50000
```

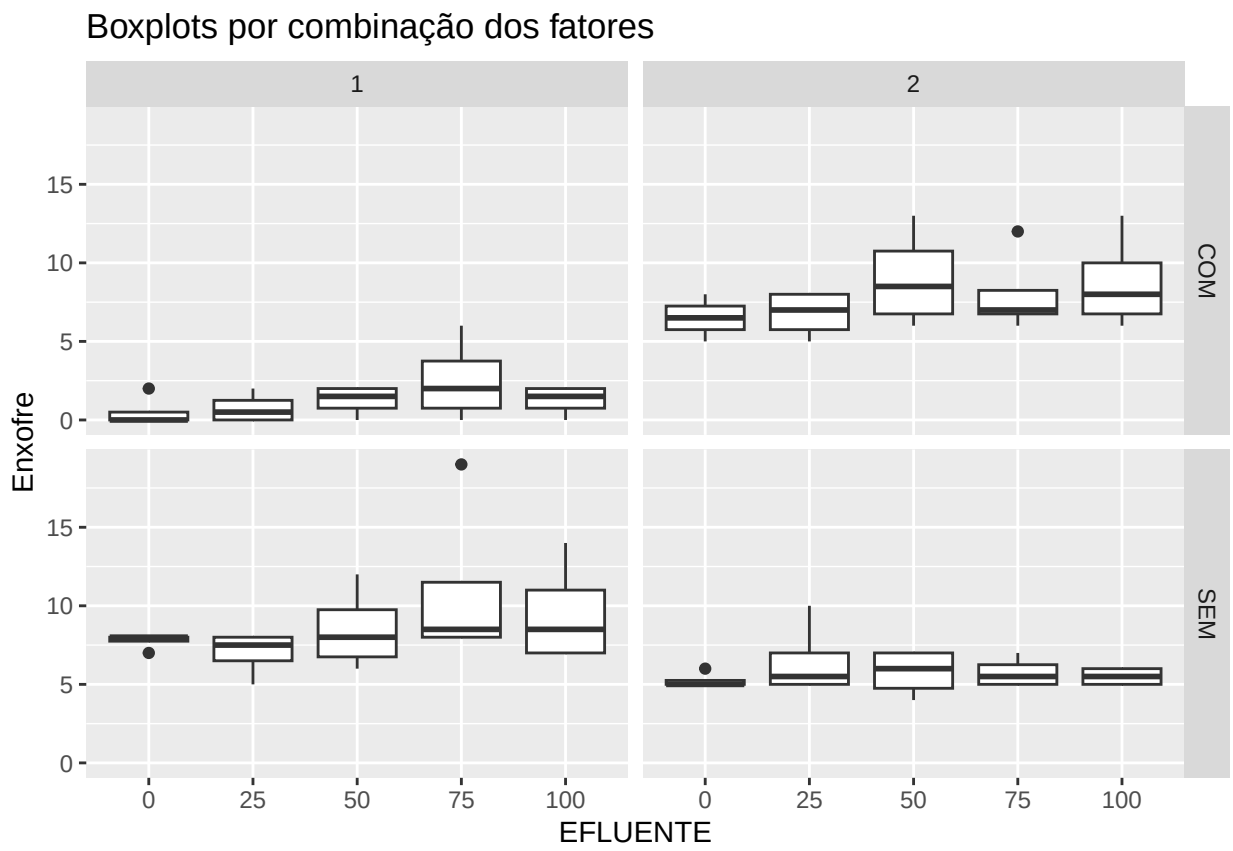
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 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:SEM-1:COM 5.183333 1.777422  8.589245 9.459282e-04
## 2:SEM-2:COM 7.466667 4.060755 10.872578 1.790071e-06
## 2:SEM-1:SEM 4.750000 1.344088  8.155912 2.749877e-03
##                diif         lwr          upr      p_adj
## 2:0-1:0    7.750000   1.0512939 14.448706079 0.01175439
## 1:25-1:0   8.875000   2.1762939 15.573706079 0.00207290
## 2:100-1:0  6.875000   0.1762939 13.573706079 0.03977540
## 1:50-1:25 -7.208333 -13.9070394 -0.509627255 0.02539384
## 1:75-1:25 -6.708333 -13.4070394 -0.009627255 0.04938712
## 2:75-1:25 -6.750000 -13.4487061 -0.051293921 0.04680994
##                      diif          lwr        upr        p_adj
## SEM:0-COM:0    10.250000   3.55129392 16.9487061 0.0002063234
## COM:25-COM:0    6.750000   0.05129392 13.4487061 0.0468099385
## SEM:25-COM:0    8.875000   2.17629392 15.5737061 0.0020729003
## COM:100-COM:0   7.250000   0.55129392 13.9487061 0.0239746776
## SEM:100-COM:0   7.750000   1.05129392 14.4487061 0.0117543935
## COM:50-SEM:0   -7.333333 -14.03203941 -0.6346273 0.0213505117
## COM:75-SEM:0   -9.375000 -16.07370608 -2.6762939 0.0009138974
## COM:75-SEM:25  -8.000000 -14.69870608 -1.3012939 0.0081112158
## SEM:100-COM:75  6.875000   0.17629392 13.5737061 0.0397753989
##                         diif         lwr        upr        p_adj
## 2:SEM:0-1:COM:0    18.00000   7.3100504 28.6899496 9.579059e-06
## 2:SEM:0-2:COM:0    18.50000   7.8100504 29.1899496 4.990747e-06
## 1:COM:25-2:COM:0   11.00000   0.3100504 21.6899496 3.719467e-02
## 2:SEM:0-1:SEM:0    16.00000   5.3100504 26.6899496 1.240670e-04
## 2:COM:25-2:SEM:0  -15.50000 -26.1899496 -4.8100504 2.316484e-04
## 1:COM:50-2:SEM:0  -13.00000 -23.6899496 -2.3100504 4.525077e-03
## 2:COM:50-2:SEM:0  -17.66667 -28.3566162 -6.9767171 1.476294e-05
## 1:SEM:50-2:SEM:0  -17.66667 -28.3566162 -6.9767171 1.476294e-05
## 1:COM:75-2:SEM:0  -16.00000 -26.6899496 -5.3100504 1.240670e-04
## 2:COM:75-2:SEM:0  -18.75000 -29.4399496 -8.0600504 3.597867e-06
## 1:SEM:75-2:SEM:0  -13.66667 -24.3566162 -2.9767171 2.109605e-03
## 2:SEM:75-2:SEM:0  -11.00000 -21.6899496 -0.3100504 3.719467e-02
## 1:COM:100-2:SEM:0 -13.25000 -23.9399496 -2.5600504 3.409071e-03
## 2:SEM:100-2:SEM:0 -11.50000 -22.1899496 -0.8100504 2.264041e-02
## 2:COM:75-1:COM:25 -11.25000 -21.9399496 -0.5600504 2.910207e-02
```

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")
```




```
# 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
```

```
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
```

```
# 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] 0 0 0 0 0 1 2 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.: 2.750
##  3:20   50 :16                   Median : 6.000
##  4:20   75 :16                   Mean   : 5.514
##         100:16                   3rd Qu.: 8.000
##                                  Max.   :14.000
##                                  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   26.00000 53.00000
## 25  30.00000 49.33333
## 50  35.66667 56.00000
## 75  44.66667       NA
## 100 31.00000       NA
```

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

```
##          COM      SEM
## 0   3.250000 6.625000
## 25  3.750000 6.166667
## 50  4.458333 7.000000
## 75  5.583333       NA
## 100 3.875000       NA
```

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

```
##           COM      SEM
## 0   12.785714 2.267857
## 25  11.642857 1.746032
## 50  13.394841 6.214286
## 75  16.246032       NA
## 100  8.339286       NA
```

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

```
##          COM      SEM
## 0   3.575712 1.505941
## 25  3.412163 1.321375
## 50  3.659896 2.492847
## 75  4.030637       NA
## 100 2.887782       NA
```

```r
# 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
##
## 5.513889
##
##  CICLO
##          1      2
##      4.708  6.521
## rep 40.000 32.000
##
##  BLOCO
##          1      2     3      4
##      6.083  5.194  5.88  4.898
## rep 18.000 18.000 18.00 18.000
##
##  EFLUENTE
##          0     25     50     75    100
##      4.837  4.858  5.628  6.701  5.951
## rep 16.000 16.000 16.000 12.000 12.000
##
##  INOCULO
##        COM   SEM
##      3.989  7.42
## rep 40.000 32.00
##
##  CICLO:EFLUENTE
```

```
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   4.277 4.152 5.152 5.001 4.959
##   rep 8.000 8.000 8.000 8.000 8.000
##   2   5.598 5.764 6.306 9.498 7.332
##   rep 8.000 8.000 8.000 4.000 4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##   1    1.270  8.147
##   rep 20.000 20.000
##   2    6.910  5.873
##   rep 20.000 12.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   2.723 6.950
##      rep 8.000 8.000
##      25  3.223 6.492
##      rep 8.000 8.000
##      50  3.932 7.325
##      rep 8.000 8.000
##      75  5.944 8.215
##      rep 8.000 4.000
##      100 4.590 8.674
##      rep 8.000 4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.000 0.750 1.250 2.500 1.250
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   6.500 6.750 7.667 8.667 6.500
##   rep 4.000 4.000 4.000 4.000 4.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   8.000 7.000 8.500 8.333 9.500
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   5.250 5.333 5.500
##   rep 4.000 4.000 4.000 0.000 0.000
```

```r
# 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
```

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

```
## [1] 0.04181569
```

```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_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] 0.9897846
```

```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(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  16.91   5.635
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1  159.3   159.3
##
```

```
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3   1.32  0.4399
##
## Error: Within
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                    1  86.36   86.36  35.985 2.52e-07 ***
## EFLUENTE                 4  66.85   16.71   6.964 0.000167 ***
## CICLO:INOCULO            1 301.52  301.52 125.644 5.30e-15 ***
## CICLO:EFLUENTE           4   7.37    1.84   0.767 0.551746
## INOCULO:EFLUENTE         4   8.05    2.01   0.839 0.507583
## CICLO:INOCULO:EFLUENTE   2   1.87    0.93   0.389 0.680034
## Residuals               48 115.19    2.40
## ---
## 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(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  58.403  58.403  25.5645 5.881e-06 ***
## INOCULO         1 187.289 187.289  81.9817 3.408e-12 ***
## EFLUENTE        4  66.852  16.713   7.3157 9.786e-05 ***
## CICLO:INOCULO   1 301.522 301.522 131.9846 9.233e-16 ***
## 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"
```

```
# 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(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 1.150000
```

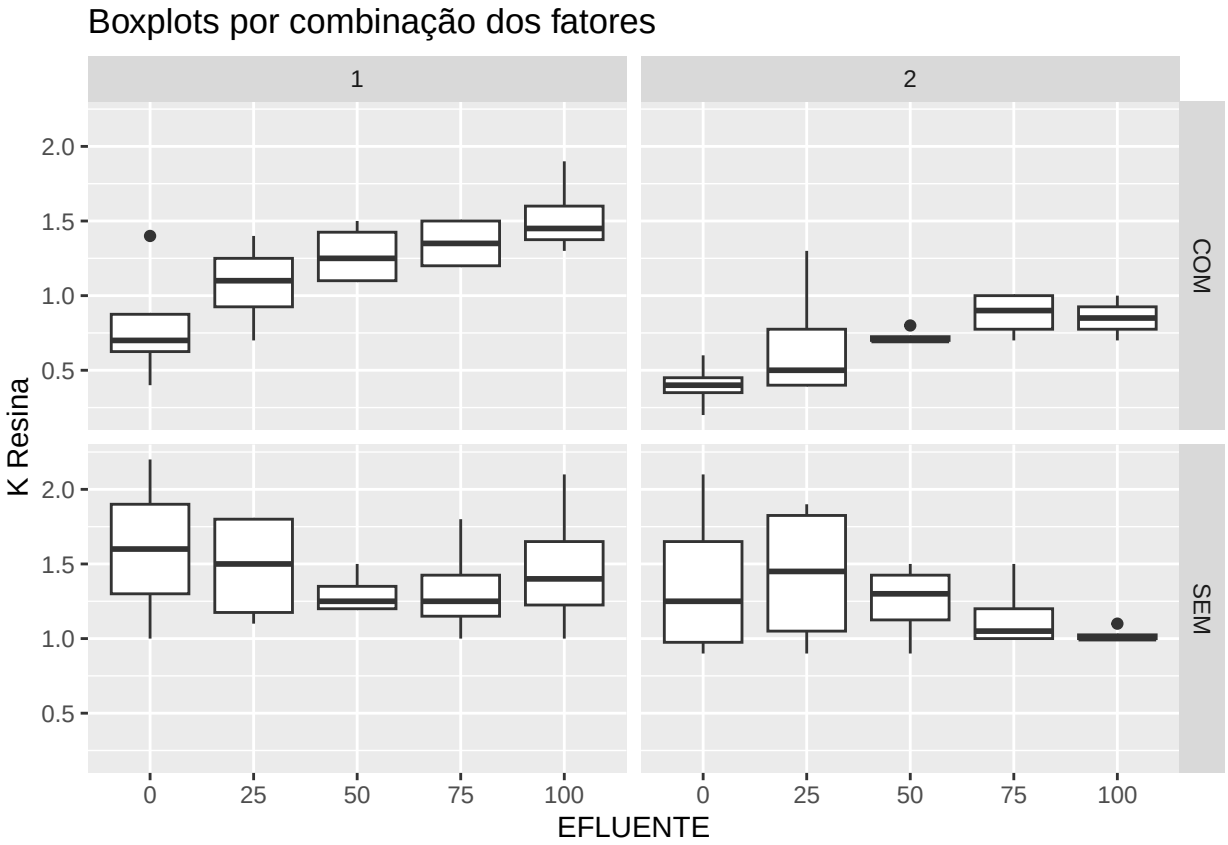
```
## 2     2     COM 7.216667
## 3     1     SEM 8.266667
## 4     2     SEM 5.361111
```

```
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  5.73355  4.464161  7.0029385 0.000000e+00
## 1:SEM-1:COM  6.78355  5.514161  8.0529385 0.000000e+00
## 2:SEM-1:COM  4.32215  2.856386  5.7879139 1.587792e-09
## 2:SEM-1:SEM -2.46140 -3.927164 -0.9956359 2.584471e-04
```

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")
```




```r
# 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] 0.6 0.7 0.7 0.4 0.7 1.2 1.4 1 1.4 1.1 ...
```

```r
summary(blocos_dados_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     K_resina
##  1:20   0  :16   COM:40   1:40   Min.   :0.200
##  2:20   25 :16   SEM:40   2:40   1st Qu.:0.800
##  3:20   50 :16                   Median :1.017
##  4:20   75 :16                   Mean   :1.089
##         100:16                   3rd Qu.:1.300
##                                  Max.   :2.200
```

```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   4.0 11.900000
## 25  6.3 10.300000
## 50  8.0  8.800000
## 75  8.9  9.433333
## 100 9.5 10.000000
```

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

```
##        COM      SEM
## 0   0.5000 1.487500
## 25  0.7875 1.287500
## 50  1.0000 1.100000
## 75  1.1125 1.179167
## 100 1.1875 1.250000
```

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

```
##            COM        SEM
## 0   0.03142857 0.25839286
## 25  0.13553571 0.10125000
## 50  0.10571429 0.05428571
## 75  0.08696429 0.07490079
## 100 0.16696429 0.15142857
```

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

```
##           COM       SEM
## 0   0.1772811 0.5083236
## 25  0.3681518 0.3181981
## 50  0.3251373 0.2329929
## 75  0.2948971 0.2736801
## 100 0.4086126 0.3891382
```

```
# 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
##
## 1.089167
##
##  CICLO
## CICLO
##      1      2
## 1.3000 0.8783
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 1.1150 0.9967 1.2050 1.0400
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.9938 1.0375 1.0500 1.1458 1.2188
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.9175 1.2608
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.1000 1.2750 1.2875 1.3375 1.5000
##     2 0.8875 0.8000 0.8125 0.9542 0.9375
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 1.1650 1.4350
##     2 0.6700 1.0867
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   0.5000 1.4875
##       25  0.7875 1.2875
##       50  1.0000 1.1000
##       75  1.1125 1.1792
##       100 1.1875 1.2500
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 0.6000 1.0750 1.2750 1.3500 1.5250
##     2 0.4000 0.5000 0.7250 0.8750 0.8500
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.6000 1.4750 1.3000 1.3250 1.4750
##     2 1.3750 1.1000 0.9000 1.0333 1.0250
```

```
# 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] 0.3967721
```

```
# 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] 0.5986078
```

```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 0.5012  0.1671
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  2.358   2.358
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  0.775  0.2583
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  3.556   3.556  66.091 6.10e-11 ***
## EFLUENTE                  4  0.533   0.133   2.476   0.0549 .
## CICLO:INOCULO             1  0.108   0.108   1.999   0.1631
## CICLO:EFLUENTE            4  0.283   0.071   1.315   0.2762
## INOCULO:EFLUENTE          4  2.616   0.654  12.157 4.07e-07 ***
## CICLO:INOCULO:EFLUENTE    4  0.040   0.010   0.185   0.9452
## Residuals                54  2.905   0.054
## ---
## 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 3.5561  3.5561  55.073 6.320e-10 ***
## INOCULO            1 2.3576  2.3576  36.511 1.225e-07 ***
## INOCULO:EFLUENTE   4 2.6165  0.6541  10.130 2.896e-06 ***
## 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] "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(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)
  }
}
```

```
##    INOCULO EFLUENTE K_resina
## 1      COM        0 0.500000
## 2      SEM        0 1.487500
## 3      COM       25 0.787500
## 4      SEM       25 1.287500
## 5      COM       50 1.000000
## 6      SEM       50 1.100000
## 7      COM       75 1.112500
## 8      SEM       75 1.179167
## 9      COM      100 1.187500
## 10     SEM      100 1.250000
```

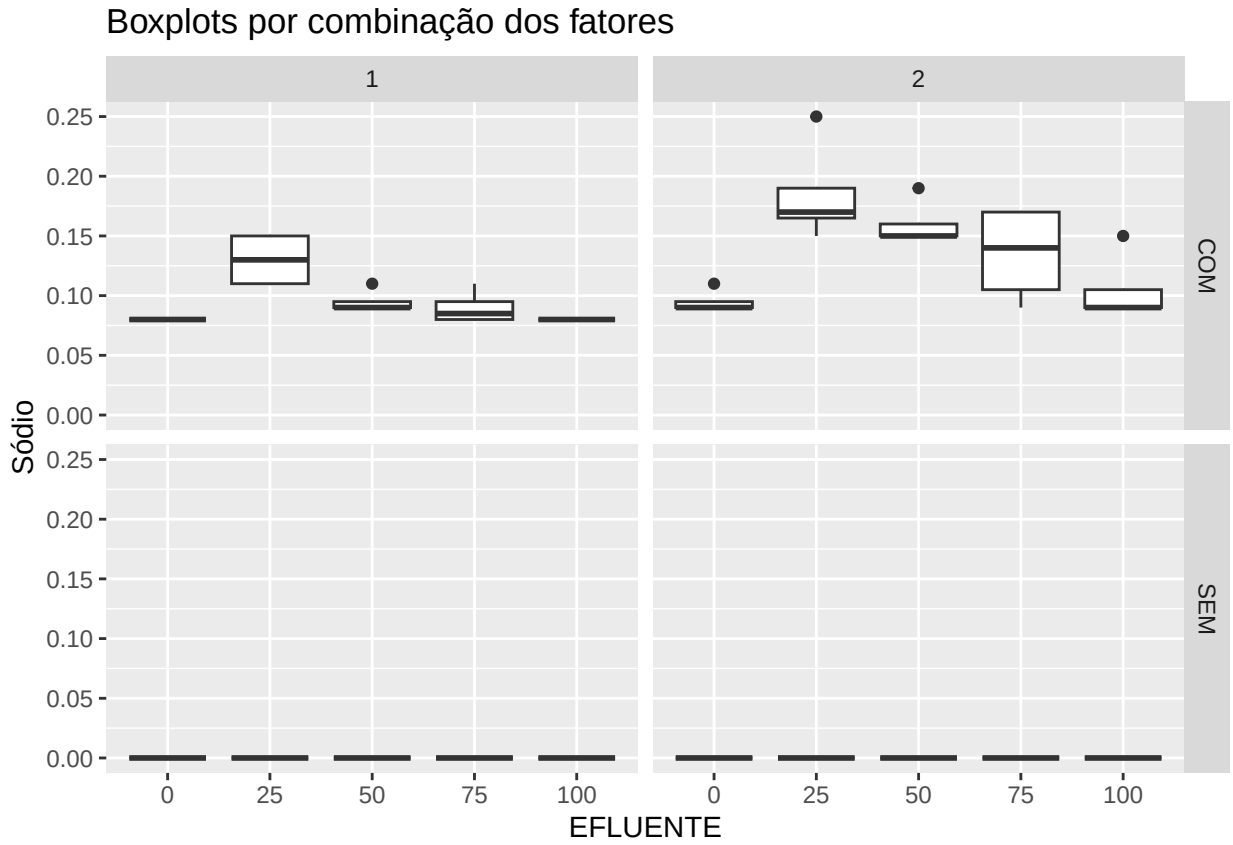
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 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
## SEM:0-COM:0      0.9875000  0.56924505  1.40575495 7.283851e-09
## SEM:25-COM:0     0.7875000  0.36924505  1.20575495 2.929914e-06
## COM:50-COM:0     0.5000000  0.08174505  0.91825495 8.015233e-03
## SEM:50-COM:0     0.6000000  0.18174505  1.01825495 6.172091e-04
## COM:75-COM:0     0.6125000  0.19424505  1.03075495 4.397775e-04
## SEM:75-COM:0     0.6791667  0.26091172  1.09742161 6.853483e-05
## COM:100-COM:0    0.6875000  0.26924505  1.10575495 5.404320e-05
## SEM:100-COM:0    0.7500000  0.33174505  1.16825495 8.845290e-06
## COM:25-SEM:0    -0.7000000 -1.11825495 -0.28174505 3.777384e-05
## COM:50-SEM:0    -0.4875000 -0.90575495 -0.06924505 1.079830e-02
## SEM:25-COM:25    0.5000000  0.08174505  0.91825495 8.015233e-03
## SEM:100-COM:25   0.4625000  0.04424505  0.88075495 1.924306e-02
```

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")
```




```r
# 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

# 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_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 0.11 0.11 0.15 0.15 0.09 0.09 ...
```

```r
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   :0.05562
##         100:16                   3rd Qu.:0.09000
##                                  Max.   :0.17000
##                                  NA's   :10
```

```r
# 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
```

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

```
##          COM SEM
## 0   0.680000   0
## 25  1.173333   0
## 50  0.960000  NA
## 75  0.760000  NA
## 100 0.320000  NA
```

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

```
##           COM SEM
## 0   0.0850000   0
## 25  0.1466667   0
## 50  0.1200000  NA
## 75  0.0950000  NA
## 100 0.0400000  NA
```

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

```
##              COM SEM
## 0   2.857143e-05   0
## 25  5.841270e-04   0
## 50  1.028571e-03  NA
## 75  1.428571e-04  NA
## 100 1.828571e-03  NA
```

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

```
##             COM SEM
## 0   0.005345225   0
## 25  0.024168719   0
## 50  0.032071349  NA
## 75  0.011952286  NA
## 100 0.042761799  NA
```

```r
# 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
##
## 0.05561905
##
##  CICLO
##          1        2
##      0.047  0.06711
## rep 40.000 30.00000
##
##  BLOCO
##            1       2        3        4
##      0.05209  0.0519  0.05975  0.05916
## rep 18.00000 18.0000 17.00000 17.00000
##
##  EFLUENTE
##            0       25     50       75     100
##      0.04094  0.07177  0.069  0.06517  0.0285
## rep 16.00000 16.00000 14.000 12.00000 12.0000
##
##  INOCULO
##          COM       SEM
##      0.09641  0.001232
## rep 40.00000 30.000000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.030  0.055  0.044  0.055  0.050
##   rep  8.000  8.000  8.000  8.000  8.000
##   2    0.055  0.091  0.102  0.079 -0.021
##   rep  8.000  8.000  6.000  4.000  4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
```

```
##   1    0.091  0.003
##   rep 20.000 20.000
##   2    0.105 -0.008
##   rep 20.000 10.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    0.078  0.004
##      rep  8.000  8.000
##      25   0.139  0.004
##      rep  8.000  8.000
##      50   0.112  0.011
##      rep  8.000  6.000
##      75   0.094  0.007
##      rep  8.000  4.000
##      100  0.055 -0.025
##      rep  8.000  4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.080 0.130 0.090 0.090 0.080
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.090 0.163 0.150 0.100 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 0.003
##   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] 0.00146516
```

```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."
```

```
# 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] 0.8598981
```

```
# 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."
```

```
# 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 0.0008125 0.0002708
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1 0.1618  0.1618
##
## Error: BLOCO:INOCULO
##           Df    Sum Sq   Mean Sq F value  Pr(>F)
## CICLO      1 1.715e-04 1.715e-04   148.7 0.00666 **
## Residuals  2 2.310e-06 1.150e-06
## ---
## 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 0.00027 0.000268   5.447 0.024024 *
## EFLUENTE                4 0.03027 0.007568 154.047  < 2e-16 ***
## CICLO:INOCULO           1 0.00145 0.001450  29.513 2.03e-06 ***
## CICLO:EFLUENTE          4 0.03243 0.008106 164.996  < 2e-16 ***
## INOCULO:EFLUENTE        4 0.00825 0.002062  41.972 8.50e-15 ***
## CICLO:INOCULO:EFLUENTE  2 0.00110 0.000548  11.145 0.000113 ***
## Residuals              46 0.00226 0.000049
## ---
## 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)
## BLOCO                    3 0.000812 0.000271    5.4667 0.0025405 **
## CICLO                    1 0.007099 0.007099  143.2880 3.703e-16 ***
## INOCULO                  1 0.155016 0.155016 3129.1026 < 2.2e-16 ***
## EFLUENTE                 4 0.030322 0.007580  153.0148 < 2.2e-16 ***
## CICLO:INOCULO            1 0.001398 0.001398   28.2195 2.630e-06 ***
## CICLO:EFLUENTE           4 0.032488 0.008122  163.9472 < 2.2e-16 ***
## INOCULO:EFLUENTE         4 0.008246 0.002062   41.6134 3.484e-15 ***
## CICLO:INOCULO:EFLUENTE   2 0.001026 0.000513   10.3547 0.0001775 ***
## 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 0.0940000
## 2     2     COM 0.1006667
## 3     1     SEM 0.0000000
## 4     2     SEM 0.0000000
```

```
##    CICLO EFLUENTE         Na
## 1      1        0 0.04000000
## 2      2        0 0.04500000
## 3      1       25 0.06500000
## 4      2       25 0.08166667
## 5      1       50 0.04500000
## 6      2       50 0.10000000
## 7      1       75 0.04500000
## 8      2       75 0.10000000
## 9      1      100 0.04000000
## 10     2      100 0.00000000
##    INOCULO EFLUENTE        Na
## 1      COM        0 0.0850000
## 2      SEM        0 0.0000000
## 3      COM       25 0.1466667
## 4      SEM       25 0.0000000
## 5      COM       50 0.1200000
## 6      SEM       50 0.0000000
## 7      COM       75 0.0950000
## 8      SEM       75 0.0000000
## 9      COM      100 0.0400000
## 10     SEM      100 0.0000000
##    CICLO INOCULO EFLUENTE        Na
## 1      1     COM        0 0.0800000
## 2      2     COM        0 0.0900000
## 3      1     SEM        0 0.0000000
## 4      2     SEM        0 0.0000000
## 5      1     COM       25 0.1300000
## 6      2     COM       25 0.1633333
## 7      1     SEM       25 0.0000000
## 8      2     SEM       25 0.0000000
## 9      1     COM       50 0.0900000
## 10     2     COM       50 0.1500000
## 11     1     SEM       50 0.0000000
## 12     2     SEM       50 0.0000000
## 13     1     COM       75 0.0900000
## 14     2     COM       75 0.1000000
## 15     1     SEM       75 0.0000000
## 16     1     COM      100 0.0800000
## 17     2     COM      100 0.0000000
## 18     1     SEM      100 0.0000000
```

```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  0.01143285  0.005513564  0.01735213 2.810621e-05
## 1:SEM-1:COM -0.08923382 -0.095153102 -0.08331454 0.000000e+00
## 2:SEM-1:COM -0.09570281 -0.102952416 -0.08845320 0.000000e+00
## 1:SEM-2:COM -0.10066667 -0.106585948 -0.09474739 0.000000e+00
## 2:SEM-2:COM -0.10713565 -0.114385261 -0.09988604 0.000000e+00
##                     diif          lwr          upr        p_adj
## 2:0-1:0      0.02352692  0.011866551  0.035187290 8.796603e-07
## 1:25-1:0     0.02500000  0.013339630  0.036660370 1.992852e-07
## 2:25-1:0     0.06019359  0.048533217  0.071853957 0.000000e+00
## 2:50-1:0     0.06194697  0.049352332  0.074541610 0.000000e+00
## 2:75-1:0     0.02668300  0.012402018  0.040963974 5.056647e-06
## 2:100-1:0   -0.07331700 -0.087597982 -0.059036026 0.000000e+00
## 2:25-2:0     0.03666667  0.025006297  0.048327036 0.000000e+00
## 1:50-2:0    -0.01747671 -0.029137078 -0.005816339 3.451602e-04
## 2:50-2:0     0.03842005  0.025825411  0.051014689 2.459033e-12
## 1:75-2:0    -0.01615422 -0.027814589 -0.004493849 1.185114e-03
## 1:100-2:0   -0.02115422 -0.032814589 -0.009493849 9.512690e-06
## 2:100-2:0   -0.09684392 -0.111124903 -0.082562946 0.000000e+00
## 2:25-1:25    0.03519359  0.023533217  0.046853957 5.125345e-12
## 1:50-1:25   -0.01894979 -0.030610158 -0.007289418 8.373999e-05
## 2:50-1:25    0.03694697  0.024352332  0.049541610 1.899036e-11
## 1:75-1:25   -0.01762730 -0.029287669 -0.005966929 2.991791e-04
## 1:100-1:25  -0.02262730 -0.034287669 -0.010966929 2.175326e-06
## 2:100-1:25  -0.09831700 -0.112597982 -0.084036026 0.000000e+00
## 1:50-2:25   -0.05414338 -0.065803745 -0.042483005 0.000000e+00
## 1:75-2:25   -0.05282089 -0.064481256 -0.041160516 0.000000e+00
## 2:75-2:25   -0.03351059 -0.047791569 -0.019229613 1.856241e-08
## 1:100-2:25  -0.05782089 -0.069481256 -0.046160516 0.000000e+00
## 2:100-2:25  -0.13351059 -0.147791569 -0.119229613 0.000000e+00
## 2:50-1:50    0.05589676  0.043302120  0.068491398 0.000000e+00
## 2:75-1:50    0.02063278  0.006351806  0.034913762 6.245019e-04
## 2:100-1:50  -0.07936722 -0.093648194 -0.065086238 0.000000e+00
## 1:75-2:50   -0.05457427 -0.067168909 -0.041979631 0.000000e+00
## 2:75-2:50   -0.03526397 -0.050317448 -0.020210502 1.943611e-08
## 1:100-2:50  -0.05957427 -0.072168909 -0.046979631 0.000000e+00
## 2:100-2:50  -0.13526397 -0.150317448 -0.120210502 0.000000e+00
## 2:75-1:75    0.01931029  0.005029317  0.033591273 1.686595e-03
## 2:100-1:75  -0.08068971 -0.094970683 -0.066408727 0.000000e+00
## 1:100-2:75  -0.02431029 -0.038591273 -0.010029317 3.469513e-05
## 2:100-2:75  -0.10000000 -0.116490253 -0.083509747 0.000000e+00
```

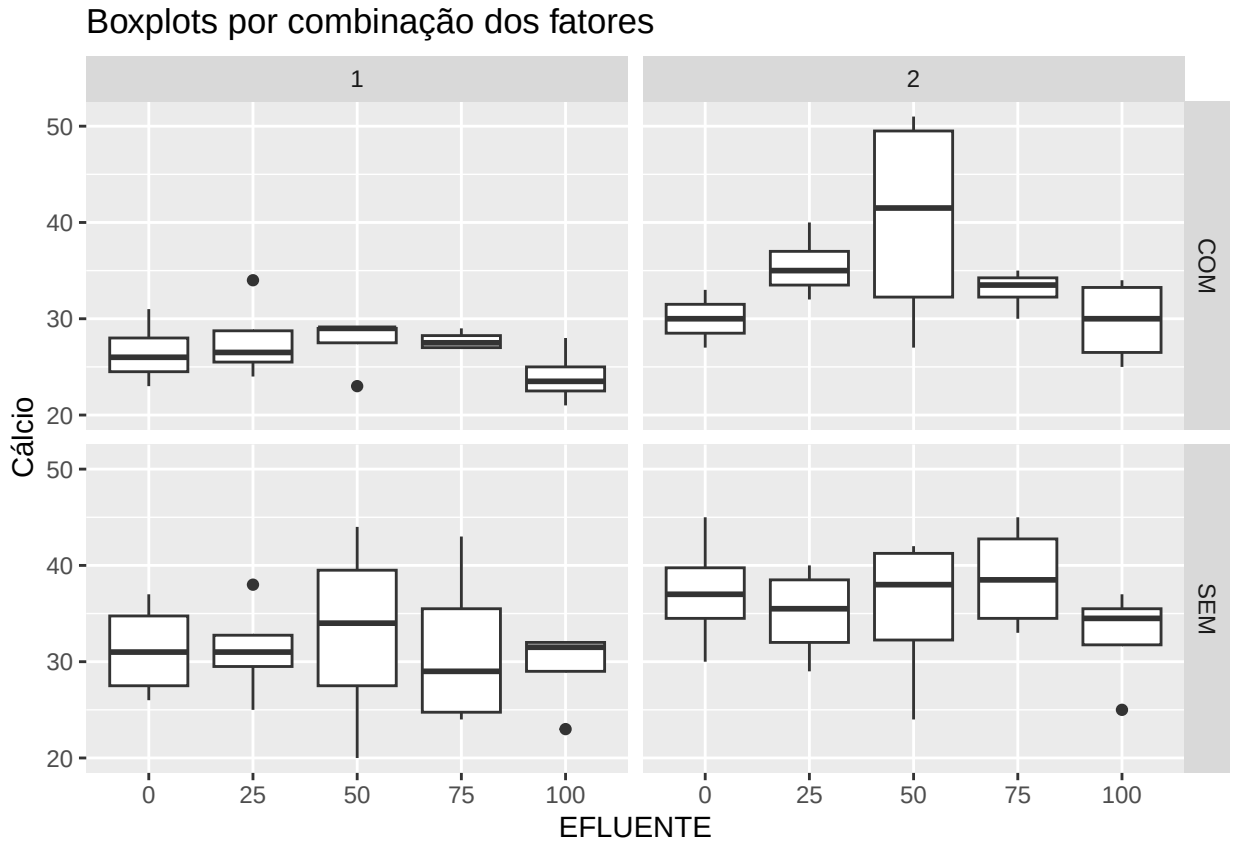
```
## 2:100-1:100 -0.07568971 -0.089970683 -0.061408727 0.000000e+00
##                         diif          lwr         upr        p_adj
## SEM:0-COM:0     -0.07330321 -0.084963578 -0.06164284 0.000000e+00
## COM:25-COM:0     0.06166667  0.050006297  0.07332704 0.000000e+00
## SEM:25-COM:0    -0.07330321 -0.084963578 -0.06164284 0.000000e+00
## COM:50-COM:0     0.03472914  0.023068772  0.04638951 1.006095e-11
## SEM:50-COM:0    -0.06635456 -0.078949197 -0.05375992 0.000000e+00
## COM:75-COM:0     0.01683975  0.005179382  0.02850012 6.285069e-04
## SEM:75-COM:0    -0.07095429 -0.085235269 -0.05667331 0.000000e+00
## COM:100-COM:0   -0.02232691 -0.033987285 -0.01066655 2.941520e-06
## SEM:100-COM:0   -0.10262096 -0.116901936 -0.08833998 0.000000e+00
## COM:25-SEM:0     0.13496988  0.123309505  0.14663025 0.000000e+00
## COM:50-SEM:0     0.10803235  0.096371980  0.11969272 0.000000e+00
## COM:75-SEM:0     0.09014296  0.078482590  0.10180333 0.000000e+00
## COM:100-SEM:0    0.05097629  0.039315924  0.06263666 0.000000e+00
## SEM:100-SEM:0   -0.02931775 -0.043598727 -0.01503677 5.814014e-07
## SEM:25-COM:25   -0.13496988 -0.146630245 -0.12330951 0.000000e+00
## COM:50-COM:25   -0.02693752 -0.038597895 -0.01527716 2.838457e-08
## SEM:50-COM:25   -0.12802122 -0.140615864 -0.11542659 0.000000e+00
## COM:75-COM:25   -0.04482691 -0.056487285 -0.03316655 0.000000e+00
## SEM:75-COM:25   -0.13262096 -0.146901936 -0.11833998 0.000000e+00
## COM:100-COM:25  -0.08399358 -0.095653951 -0.07233321 0.000000e+00
## SEM:100-COM:25  -0.16428762 -0.178568602 -0.15000665 0.000000e+00
## COM:50-SEM:25    0.10803235  0.096371980  0.11969272 0.000000e+00
## COM:75-SEM:25    0.09014296  0.078482590  0.10180333 0.000000e+00
## COM:100-SEM:25   0.05097629  0.039315924  0.06263666 0.000000e+00
## SEM:100-SEM:25  -0.02931775 -0.043598727 -0.01503677 5.814014e-07
## SEM:50-COM:50   -0.10108370 -0.113678339 -0.08848906 0.000000e+00
## COM:75-COM:50   -0.01788939 -0.029549760 -0.00622902 2.330262e-04
## SEM:75-COM:50   -0.10568343 -0.119964411 -0.09140245 0.000000e+00
## COM:100-COM:50  -0.05705606 -0.068716426 -0.04539569 0.000000e+00
## SEM:100-COM:50  -0.13735010 -0.151631077 -0.12306912 0.000000e+00
## COM:75-SEM:50    0.08319431  0.070599671  0.09578895 0.000000e+00
## COM:100-SEM:50   0.04402764  0.031433004  0.05662228 0.000000e+00
## SEM:100-SEM:50  -0.03626640 -0.051319872 -0.02121293 8.941814e-09
## SEM:75-COM:75   -0.08779404 -0.102075021 -0.07351306 0.000000e+00
## COM:100-COM:75  -0.03916667 -0.050827036 -0.02750630 0.000000e+00
## SEM:100-COM:75  -0.11946071 -0.133741687 -0.10517973 0.000000e+00
## COM:100-SEM:75   0.04862738  0.034346398  0.06290835 0.000000e+00
## SEM:100-SEM:75  -0.03166667 -0.048156920 -0.01517641 2.756227e-06
## SEM:100-COM:100 -0.08029404 -0.094575021 -0.06601306 0.000000e+00
##                           diif          lwr          upr        p_adj
## 1:SEM:0-1:COM:0     -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 2:SEM:0-1:COM:0     -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 1:COM:25-1:COM:0     0.05000000  0.031371657  0.068628343 2.834988e-11
## 2:COM:25-1:COM:0     0.08333333  0.064704990  0.101961677 0.000000e+00
## 1:SEM:25-1:COM:0    -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 2:SEM:25-1:COM:0    -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 2:COM:50-1:COM:0     0.07000000  0.051371657  0.088628343 0.000000e+00
## 1:SEM:50-1:COM:0    -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 2:SEM:50-1:COM:0    -0.07659858 -0.099413552 -0.053783616 0.000000e+00
## 2:COM:75-1:COM:0     0.02000000  0.001371657  0.038628343 2.360630e-02
## 1:SEM:75-1:COM:0    -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 2:COM:100-1:COM:0   -0.08000000 -0.098628343 -0.061371657 0.000000e+00
```

```
## 1:SEM:100-1:COM:0   -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 1:SEM:0-2:COM:0     -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:SEM:0-2:COM:0     -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:COM:25-2:COM:0     0.04000000  0.021371657  0.058628343 3.046577e-08
## 2:COM:25-2:COM:0     0.07333333  0.054704990  0.091961677 0.000000e+00
## 1:SEM:25-2:COM:0    -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:SEM:25-2:COM:0    -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:COM:50-2:COM:0     0.06000000  0.041371657  0.078628343 0.000000e+00
## 1:SEM:50-2:COM:0    -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:SEM:50-2:COM:0    -0.08659858 -0.109413552 -0.063783616 0.000000e+00
## 1:SEM:75-2:COM:0    -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:COM:100-2:COM:0   -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:SEM:100-2:COM:0   -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:COM:25-1:SEM:0     0.13000000  0.111371657  0.148628343 0.000000e+00
## 2:COM:25-1:SEM:0     0.16333333  0.144704990  0.181961677 0.000000e+00
## 1:COM:50-1:SEM:0     0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:50-1:SEM:0     0.15000000  0.131371657  0.168628343 0.000000e+00
## 1:COM:75-1:SEM:0     0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:75-1:SEM:0     0.10000000  0.081371657  0.118628343 0.000000e+00
## 1:COM:100-1:SEM:0    0.08000000  0.061371657  0.098628343 0.000000e+00
## 1:COM:25-2:SEM:0     0.13000000  0.111371657  0.148628343 0.000000e+00
## 2:COM:25-2:SEM:0     0.16333333  0.144704990  0.181961677 0.000000e+00
## 1:COM:50-2:SEM:0     0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:50-2:SEM:0     0.15000000  0.131371657  0.168628343 0.000000e+00
## 1:COM:75-2:SEM:0     0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:75-2:SEM:0     0.10000000  0.081371657  0.118628343 0.000000e+00
## 1:COM:100-2:SEM:0    0.08000000  0.061371657  0.098628343 0.000000e+00
## 2:COM:25-1:COM:25    0.03333333  0.014704990  0.051961677 3.407696e-06
## 1:SEM:25-1:COM:25   -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 2:SEM:25-1:COM:25   -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 1:COM:50-1:COM:25   -0.04000000 -0.058628343 -0.021371657 3.046577e-08
## 2:COM:50-1:COM:25    0.02000000  0.001371657  0.038628343 2.360630e-02
## 1:SEM:50-1:COM:25   -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 2:SEM:50-1:COM:25   -0.12659858 -0.149413552 -0.103783616 0.000000e+00
## 1:COM:75-1:COM:25   -0.04000000 -0.058628343 -0.021371657 3.046577e-08
## 2:COM:75-1:COM:25   -0.03000000 -0.048628343 -0.011371657 3.562585e-05
## 1:SEM:75-1:COM:25   -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 1:COM:100-1:COM:25  -0.05000000 -0.068628343 -0.031371657 2.834988e-11
## 2:COM:100-1:COM:25  -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 1:SEM:100-1:COM:25  -0.13000000 -0.148628343 -0.111371657 0.000000e+00
## 1:SEM:25-2:COM:25   -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 2:SEM:25-2:COM:25   -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 1:COM:50-2:COM:25   -0.07333333 -0.091961677 -0.054704990 0.000000e+00
## 1:SEM:50-2:COM:25   -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 2:SEM:50-2:COM:25   -0.15993192 -0.182746885 -0.137116949 0.000000e+00
## 1:COM:75-2:COM:25   -0.07333333 -0.091961677 -0.054704990 0.000000e+00
## 2:COM:75-2:COM:25   -0.06333333 -0.081961677 -0.044704990 0.000000e+00
## 1:SEM:75-2:COM:25   -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 1:COM:100-2:COM:25  -0.08333333 -0.101961677 -0.064704990 0.000000e+00
## 2:COM:100-2:COM:25  -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 1:SEM:100-2:COM:25  -0.16333333 -0.181961677 -0.144704990 0.000000e+00
## 1:COM:50-1:SEM:25    0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:50-1:SEM:25    0.15000000  0.131371657  0.168628343 0.000000e+00
## 1:COM:75-1:SEM:25    0.09000000  0.071371657  0.108628343 0.000000e+00
```

```
## 2:COM:75-1:SEM:25    0.10000000  0.081371657  0.118628343 0.000000e+00
## 1:COM:100-1:SEM:25   0.08000000  0.061371657  0.098628343 0.000000e+00
## 1:COM:50-2:SEM:25    0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:50-2:SEM:25    0.15000000  0.131371657  0.168628343 0.000000e+00
## 1:COM:75-2:SEM:25    0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:75-2:SEM:25    0.10000000  0.081371657  0.118628343 0.000000e+00
## 1:COM:100-2:SEM:25   0.08000000  0.061371657  0.098628343 0.000000e+00
## 2:COM:50-1:COM:50    0.06000000  0.041371657  0.078628343 0.000000e+00
## 1:SEM:50-1:COM:50   -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:SEM:50-1:COM:50   -0.08659858 -0.109413552 -0.063783616 0.000000e+00
## 1:SEM:75-1:COM:50   -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:COM:100-1:COM:50  -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:SEM:100-1:COM:50  -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:SEM:50-2:COM:50   -0.15000000 -0.168628343 -0.131371657 0.000000e+00
## 2:SEM:50-2:COM:50   -0.14659858 -0.169413552 -0.123783616 0.000000e+00
## 1:COM:75-2:COM:50   -0.06000000 -0.078628343 -0.041371657 0.000000e+00
## 2:COM:75-2:COM:50   -0.05000000 -0.068628343 -0.031371657 2.834988e-11
## 1:SEM:75-2:COM:50   -0.15000000 -0.168628343 -0.131371657 0.000000e+00
## 1:COM:100-2:COM:50  -0.07000000 -0.088628343 -0.051371657 0.000000e+00
## 2:COM:100-2:COM:50  -0.15000000 -0.168628343 -0.131371657 0.000000e+00
## 1:SEM:100-2:COM:50  -0.15000000 -0.168628343 -0.131371657 0.000000e+00
## 1:COM:75-1:SEM:50    0.09000000  0.071371657  0.108628343 0.000000e+00
## 2:COM:75-1:SEM:50    0.10000000  0.081371657  0.118628343 0.000000e+00
## 1:COM:100-1:SEM:50   0.08000000  0.061371657  0.098628343 0.000000e+00
## 1:COM:75-2:SEM:50    0.08659858  0.063783616  0.109413552 0.000000e+00
## 2:COM:75-2:SEM:50    0.09659858  0.073783616  0.119413552 0.000000e+00
## 1:COM:100-2:SEM:50   0.07659858  0.053783616  0.099413552 0.000000e+00
## 1:SEM:75-1:COM:75   -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 2:COM:100-1:COM:75  -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:SEM:100-1:COM:75  -0.09000000 -0.108628343 -0.071371657 0.000000e+00
## 1:SEM:75-2:COM:75   -0.10000000 -0.118628343 -0.081371657 0.000000e+00
## 1:COM:100-2:COM:75  -0.02000000 -0.038628343 -0.001371657 2.360630e-02
## 2:COM:100-2:COM:75  -0.10000000 -0.118628343 -0.081371657 0.000000e+00
## 1:SEM:100-2:COM:75  -0.10000000 -0.118628343 -0.081371657 0.000000e+00
## 1:COM:100-1:SEM:75   0.08000000  0.061371657  0.098628343 0.000000e+00
## 2:COM:100-1:COM:100 -0.08000000 -0.098628343 -0.061371657 0.000000e+00
## 1:SEM:100-1:COM:100 -0.08000000 -0.098628343 -0.061371657 0.000000e+00
```

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] 31 27 23 25 25.7 ...
```

```
summary(blocos_dados_6)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Ca
##  1:20   0  :16   COM:40   1:40   Min.   :20.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:27.00
##  3:20   50 :16                   Median :31.00
##  4:20   75 :16                   Mean   :31.17
##         100:16                   3rd Qu.:34.00
##                                  Max.   :49.00
```

```
# 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   230.0000 274.0000
## 25  244.6667 256.0000
## 50  282.0000 265.3333
## 75  243.0000 257.0000
## 100 215.0000 226.6667
```

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

```
##          COM      SEM
## 0   28.75000 34.25000
## 25  30.58333 32.00000
## 50  35.25000 33.16667
## 75  30.37500 32.12500
## 100 26.87500 28.33333
```

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

```
##          COM      SEM
## 0   11.92857 37.92857
## 25  33.29365 24.28571
## 50  60.71429 67.55556
## 75  10.26786 34.12500
## 100 21.55357 12.79365
```

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

```
##          COM      SEM
## 0   3.453776 6.158618
## 25  5.770065 4.928054
## 50  7.791937 8.219219
## 75  3.204350 5.841661
## 100 4.642582 3.576821
```

```
# 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
##
## 31.17083
##
##  CICLO
## CICLO
##     1     2
## 28.91 33.43
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 31.77 31.78 28.40 32.74
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 31.50 31.29 34.21 31.25 27.60
##
##  INOCULO
## INOCULO
##   COM   SEM
## 30.37 31.98
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 28.88 27.33 31.00 29.50 27.83
##     2 34.13 35.25 37.42 33.00 27.38
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 26.58 31.23
##     2 34.15 32.72
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM   SEM
##       0   28.75 34.25
##       25  30.58 32.00
##       50  35.25 33.17
##       75  30.38 32.13
##       100 26.88 28.33
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 26.50 25.67 29.00 27.75 24.00
##     2 31.00 35.50 41.50 33.00 29.75
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 31.25 29.00 33.00 31.25 31.67
##     2 37.25 35.00 33.33 33.00 25.00
```

```
# 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] 0.8542728
```

```
# 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_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] 0.8152287
```

```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  217.3   72.43
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  51.73   51.73
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  109.6   36.54
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  409.5   409.5  22.860 1.38e-05 ***
## EFLUENTE                  4  353.2    88.3   4.930  0.00184 **
## CICLO:INOCULO             1  185.0   185.0  10.329  0.00221 **
## CICLO:EFLUENTE            4  166.0    41.5   2.316  0.06890 .
## INOCULO:EFLUENTE          4  115.4    28.9   1.611  0.18491
## CICLO:INOCULO:EFLUENTE    4  146.4    36.6   2.043  0.10133
## Residuals                54  967.3    17.9
## ---
## 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)
## BLOCO          3 217.30   72.43  3.8337  0.014317 *
## CICLO          1 409.51  409.51 21.6744 1.975e-05 ***
## EFLUENTE       4 353.23   88.31  4.6738  0.002478 **
## CICLO:INOCULO  1 185.03  185.03  9.7934  0.002760 **
## 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)
  }
}
```

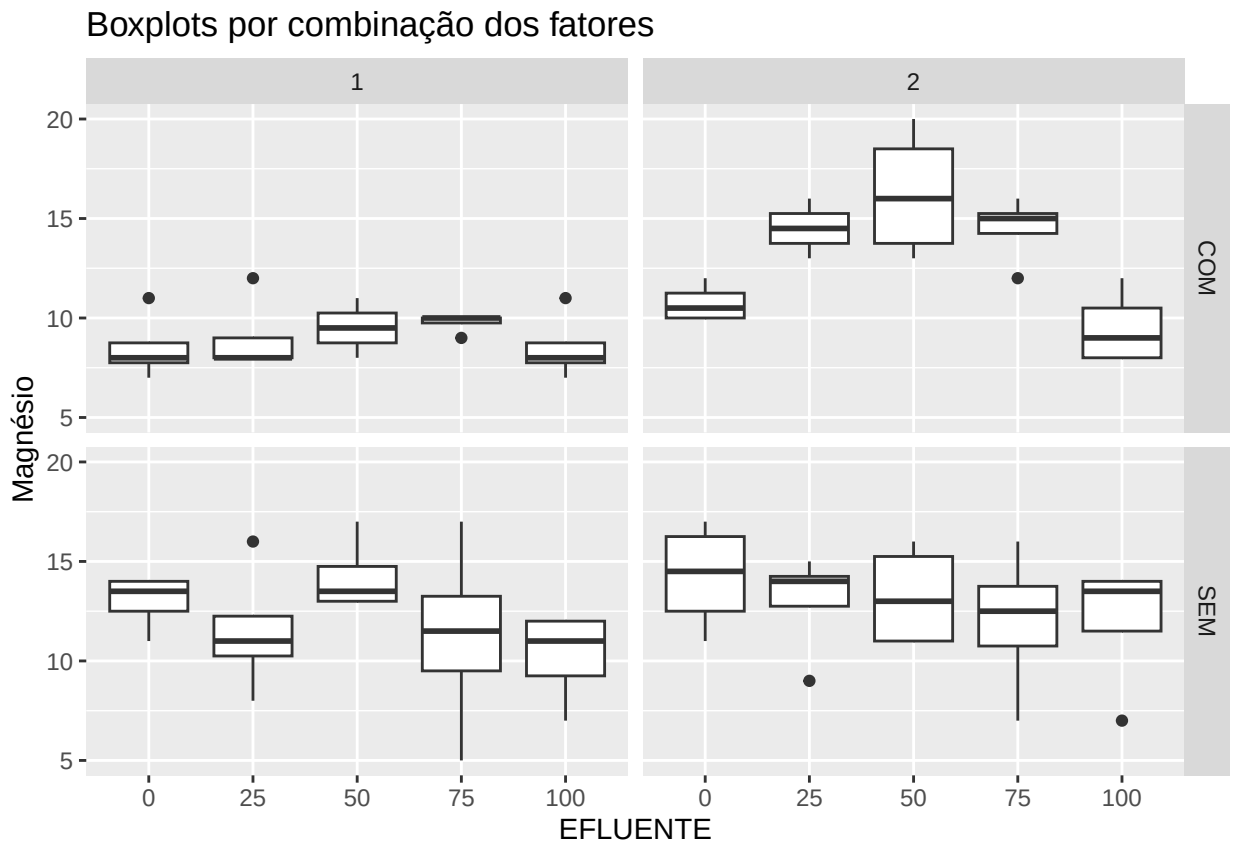
```
##   CICLO INOCULO       Ca
## 1     1     COM 26.58333
## 2     2     COM 34.15000
## 3     1     SEM 31.23333
## 4     2     SEM 32.71667
```

```
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 7.566667 3.928960 11.204373 5.390474e-06
## 1:SEM-1:COM 4.650000 1.012294  8.287706 6.938991e-03
## 2:SEM-1:COM 6.133333 2.495627  9.771040 2.221511e-04
```

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
```

```r
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))]
```

```r
# 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] 8 8 7 7.67 8 ...
```

```r
summary(blocos_dados_7)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Mg
##  1:20   0  :16   COM:40   1:40   Min.   : 5.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 8.00
##  3:20   50 :16                   Median :11.00
##  4:20   75 :16                   Mean   :11.07
##         100:16                   3rd Qu.:13.12
##                                  Max.   :20.00
```

```r
# 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
```

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

```
##           COM SEM
## 0    74.66667 109
## 25   90.00000  92
## 50  100.66667 109
## 75   94.00000  73
## 100  74.66667  69
```

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

```
##           COM    SEM
## 0    9.333333 13.625
```

```
## 25  11.250000 11.500
## 50  12.583333 13.625
## 75  11.750000  9.125
## 100  9.333333  8.625
```

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

```
##           COM       SEM
## 0    3.555556  4.553571
## 25  12.785714  6.571429
## 50  15.674603  3.125000
## 75   4.142857 15.553571
## 100  3.555556  5.410714
```

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

```
##          COM      SEM
## 0   1.885618 2.133910
## 25  3.575712 2.563480
## 50  3.959116 1.767767
## 75  2.035401 3.943802
## 100 1.885618 2.326094
```

```
# 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
##
## 11.075
##
##  CICLO
## CICLO
##      1      2
## 10.158 11.992
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 11.208 11.275  9.950 11.867
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 11.479 11.375 13.104 10.437  8.979
##
##  INOCULO
```

```
## INOCULO
##   COM   SEM
## 10.85 11.30
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 10.333  9.000 11.875 10.625  8.958
##     2 12.625 13.750 14.333 10.250  9.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1  8.567 11.750
##     2 13.133 10.850
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    9.333 13.625
##      25  11.250 11.500
##      50  12.583 13.625
##      75  11.750  9.125
##      100  9.333  8.625
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  7.667  8.000  9.500 10.000  7.667
##     2 11.000 14.500 15.667 13.500 11.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 13.000 10.000 14.250 11.250 10.250
##     2 14.250 13.000 13.000  7.000  7.000
```

```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] 0.2351851
```

```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] 0.9723063
```

```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     39      13
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1   4.05    4.05
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  17.47   5.823
##
## Error: Within
##                       Df Sum Sq Mean Sq F value   Pr(>F)
```

```
## CICLO                   1  67.22   67.22  23.117 1.26e-05 ***
## EFLUENTE                4 146.72   36.68  12.614 2.58e-07 ***
## CICLO:INOCULO           1 149.42  149.42  51.385 2.20e-09 ***
## CICLO:EFLUENTE          4  68.78   17.19   5.913 0.000508 ***
## INOCULO:EFLUENTE        4 103.78   25.95   8.922 1.32e-05 ***
## CICLO:INOCULO:EFLUENTE  4  25.58    6.39   2.199 0.081300 .
## Residuals              54 157.03    2.91
## ---
## 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)
## BLOCO              3  39.003  13.001  4.2468 0.0088995 **
## CICLO              1  67.222  67.222 21.9583 1.775e-05 ***
## EFLUENTE           4 146.717  36.679 11.9814 3.927e-07 ***
## CICLO:INOCULO      1 149.422 149.422 48.8092 3.353e-09 ***
## CICLO:EFLUENTE     4  68.778  17.194  5.6166 0.0006986 ***
## INOCULO:EFLUENTE   4 103.783  25.946  8.4753 1.944e-05 ***
## 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"
```

```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  8.566667
## 2     2     COM 13.133333
## 3     1     SEM 11.750000
## 4     2     SEM 10.850000
##    CICLO EFLUENTE        Mg
## 1      1        0 10.333333
## 2      2        0 12.625000
```

```
## 3      1       25  9.000000
## 4      2       25 13.750000
## 5      1       50 11.875000
## 6      2       50 14.333333
## 7      1       75 10.625000
## 8      2       75 10.250000
## 9      1      100  8.958333
## 10     2      100  9.000000
##    INOCULO EFLUENTE        Mg
## 1      COM        0  9.333333
## 2      SEM        0 13.625000
## 3      COM       25 11.250000
## 4      SEM       25 11.500000
## 5      COM       50 12.583333
## 6      SEM       50 13.625000
## 7      COM       75 11.750000
## 8      SEM       75  9.125000
## 9      COM      100  9.333333
## 10     SEM      100  8.625000
```

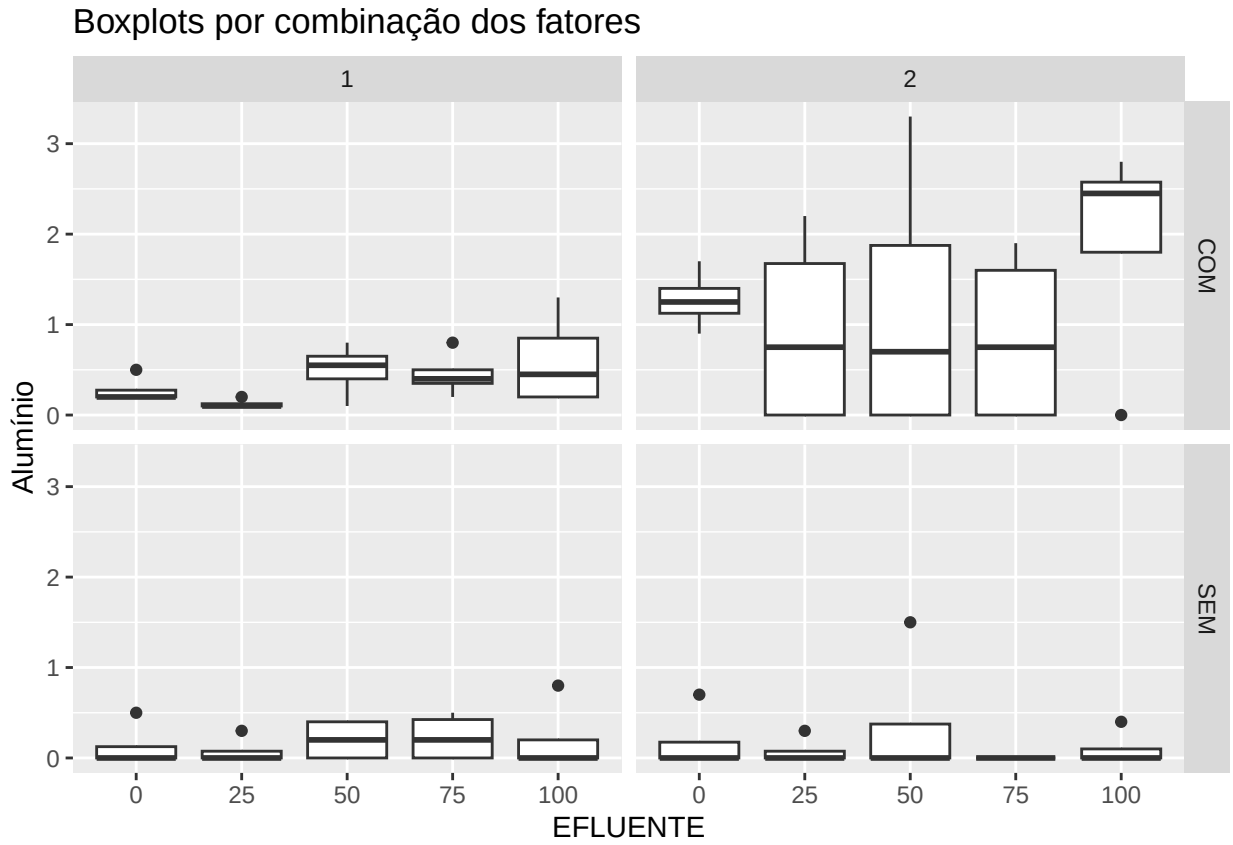
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 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  4.566667  3.1023857  6.0309476 1.673046e-10
## 1:SEM-1:COM  3.183333  1.7190524  4.6476143 2.144194e-06
## 2:SEM-1:COM  2.283333  0.8190524  3.7476143 6.839913e-04
## 2:SEM-2:COM -2.283333 -3.7476143 -0.8190524 6.839913e-04
##                  diif        lwr         upr        p_adj
## 2:25-1:0    3.416667  0.5367367  6.29659667 8.779567e-03
## 2:50-1:0    4.000000  1.1200700  6.87993001 1.028205e-03
## 1:25-2:0   -3.625000 -6.5049300 -0.74506999 4.187381e-03
## 1:100-2:0  -3.666667 -6.5465967 -0.78673666 3.597859e-03
```

```
## 2:100-2:0  -3.625000 -6.5049300 -0.74506999 4.187381e-03
## 2:25-1:25   4.750000  1.8700700  7.62993001 5.053666e-05
## 2:50-1:25   5.333333  2.4534033  8.21326334 4.294744e-06
## 1:75-2:25  -3.125000 -6.0049300 -0.24506999 2.336028e-02
## 2:75-2:25  -3.500000 -6.3799300 -0.62006999 6.554090e-03
## 1:100-2:25 -4.791667 -7.6715967 -1.91173666 4.249470e-05
## 2:100-2:25 -4.750000 -7.6299300 -1.87006999 5.053666e-05
## 1:100-1:50 -2.916667 -5.7965967 -0.03673666 4.478638e-02
## 1:75-2:50  -3.708333 -6.5882633 -0.82840333 3.087796e-03
## 2:75-2:50  -4.083333 -6.9632633 -1.20340333 7.445476e-04
## 1:100-2:50 -5.375000 -8.2549300 -2.49506999 3.591768e-06
## 2:100-2:50 -5.333333 -8.2132633 -2.45340333 4.294744e-06
##                      diif       lwr        upr        p_adj
## SEM:0-COM:0     4.291667  1.411737  7.1715967 3.274111e-04
## COM:50-COM:0    3.250000  0.370070  6.1299300 1.549610e-02
## SEM:50-COM:0    4.291667  1.411737  7.1715967 3.274111e-04
## SEM:75-SEM:0   -4.500000 -7.379930 -1.6200700 1.413471e-04
## COM:100-SEM:0  -4.291667 -7.171597 -1.4117367 3.274111e-04
## SEM:100-SEM:0  -5.000000 -7.879930 -2.1200700 1.773832e-05
## SEM:75-COM:50  -3.458333 -6.338263 -0.5784033 7.590636e-03
## COM:100-COM:50 -3.250000 -6.129930 -0.3700700 1.549610e-02
## SEM:100-COM:50 -3.958333 -6.838263 -1.0784033 1.206664e-03
## SEM:75-SEM:50  -4.500000 -7.379930 -1.6200700 1.413471e-04
## COM:100-SEM:50 -4.291667 -7.171597 -1.4117367 3.274111e-04
## SEM:100-SEM:50 -5.000000 -7.879930 -2.1200700 1.773832e-05
## SEM:100-COM:75 -3.125000 -6.004930 -0.2450700 2.336028e-02
```

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")
```




```
# 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] 0.2 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.6 0.1 ...
```

```
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 :0.0500
##  4:20   75 :16                   Mean   :0.3801
##         100:16                   3rd Qu.:0.4250
##                                  Max.   :3.3000
##                                  NA's   :8
```

```
# 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   5.900000 0.7
## 25  4.100000 0.0
## 50  6.700000 0.8
## 75  5.866667  NA
## 100 2.400000  NA
```

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

```
##           COM    SEM
## 0   0.7375000 0.0875
## 25  0.5125000 0.0000
## 50  0.8375000 0.1000
## 75  0.7333333     NA
## 100 0.3000000     NA
```

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

```
##           COM        SEM
## 0   0.3769643 0.06125000
## 25  0.7183929 0.00000000
## 50  1.2141071 0.03428571
## 75  0.4733333         NA
## 100 0.2200000         NA
```

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

```
##           COM       SEM
## 0   0.6139742 0.2474874
## 25  0.8475806 0.0000000
## 50  1.1018653 0.1851640
## 75  0.6879922        NA
## 100 0.4690416        NA
```

```r
# 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
##
## 0.3800926
##
##  CICLO
##           1       2
##      0.2158  0.5854
## rep 40.0000 32.0000
##
##  BLOCO
##           1       2       3       4
##      0.3019  0.4741  0.5667  0.1778
## rep 18.0000 18.0000 18.0000 18.0000
##
##  EFLUENTE
##          0      25      50     75     100
##      0.392  0.2357  0.4482  0.605  0.2411
## rep 16.000 16.0000 16.0000 12.000 12.0000
##
##  INOCULO
##         COM     SEM
##      0.5987  0.1069
## rep 40.0000 32.0000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.060  0.010  0.310  0.339  0.360
##   rep  8.000  8.000  8.000  8.000  8.000
##   2    0.765  0.502  0.627  1.014 -0.119
##   rep  8.000  8.000  8.000  4.000  4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
```

```
##   1    0.323  0.108
##   rep 20.000 20.000
##   2    0.915  0.036
##   rep 20.000 12.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    0.643  0.141
##      rep  8.000  8.000
##      25   0.418  0.053
##      rep  8.000  8.000
##      50   0.743  0.153
##      rep  8.000  8.000
##      75   0.727  0.360
##      rep  8.000  4.000
##      100  0.486 -0.249
##      rep  8.000  4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.200 0.100 0.500 0.333 0.600
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   1.275 0.925 1.175 1.133 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 0.200 0.225 0.000
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.175 0.000 0.000
##   rep 4.000 4.000 4.000 0.000 0.000
```

```r
# 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] 0.001535278
```

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

```r
} 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_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] 0.9678497
```

```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  1.633  0.5442
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1  5.361   5.361
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.9165  0.3055
##
## Error: Within
##                        Df Sum Sq Mean Sq F value Pr(>F)
## CICLO                   1  1.636  1.6356   6.220 0.0161 *
## EFLUENTE                4  1.199  0.2998   1.140 0.3490
## CICLO:INOCULO           1  1.589  1.5895   6.045 0.0176 *
```

```
## CICLO:EFLUENTE          4  3.563  0.8906   3.387 0.0161 *
## INOCULO:EFLUENTE        4  0.402  0.1004   0.382 0.8205
## CICLO:INOCULO:EFLUENTE  2  0.003  0.0015   0.006 0.9945
## Residuals              48 12.622  0.2630
## ---
## 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(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, ]

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

```
## Analysis of Variance Table
##
## Response: Al
##                Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO           1 2.4283  2.4283  9.1476 0.0038935 **
## INOCULO         1 4.5688  4.5688 17.2109 0.0001268 ***
## CICLO:INOCULO   1 1.5895  1.5895  5.9878 0.0178903 *
## CICLO:EFLUENTE  4 3.5625  0.8906  3.3551 0.0163361 *
## 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)
```

```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 0.34666667
## 2     2     COM 0.90166667
## 3     1     SEM 0.08500000
## 4     2     SEM 0.05833333
##    CICLO EFLUENTE        Al
## 1      1        0 0.1000000
## 2      2        0 0.7250000
## 3      1       25 0.0500000
## 4      2       25 0.4625000
## 5      1       50 0.3500000
```

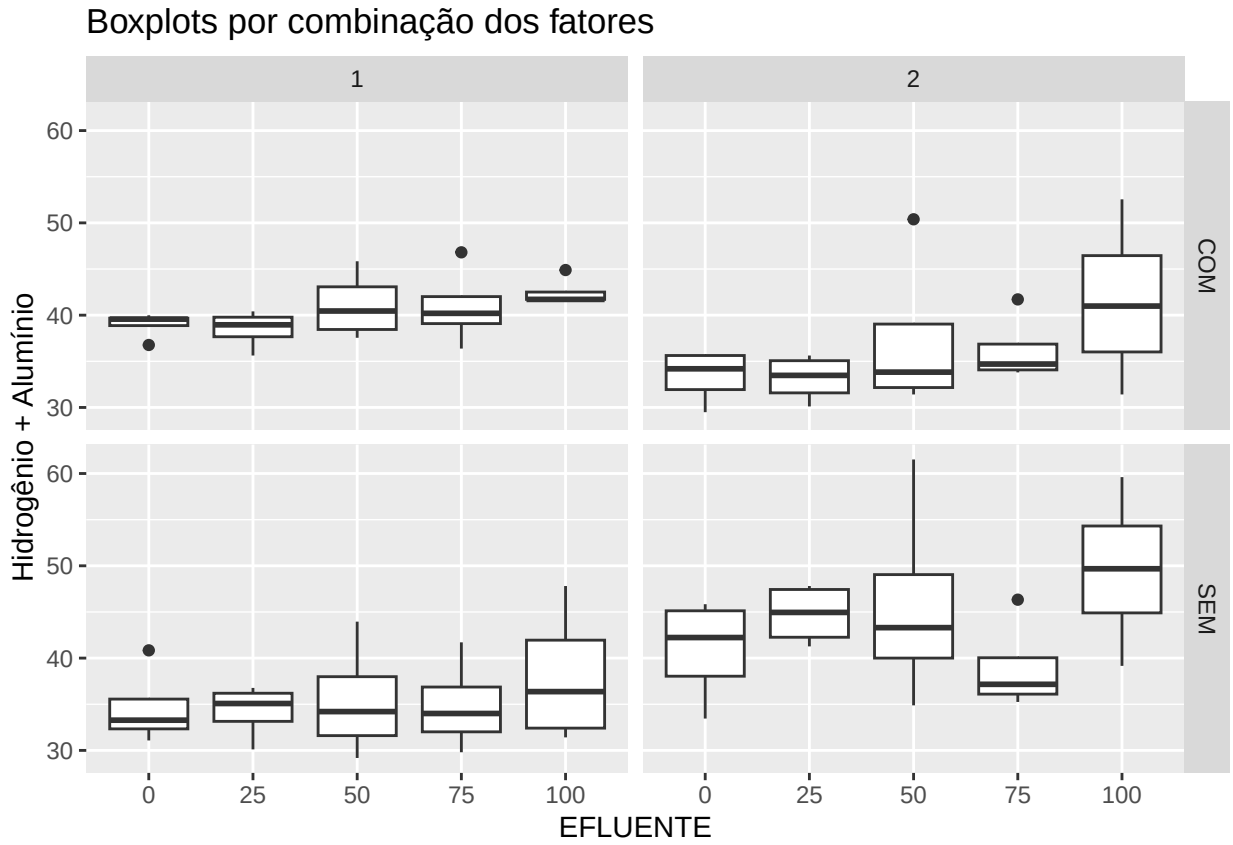
```
## 6      2       50 0.5875000
## 7      1       75 0.2791667
## 8      2       75 1.1333333
## 9      1      100 0.3000000
## 10     2      100 0.0000000
```

```
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  0.5607143  0.1280073  0.9934212 6.201697e-03
## 1:SEM-2:COM -0.8166667 -1.2493736 -0.3839597 3.976055e-05
## 2:SEM-2:COM -0.8509524 -1.3505993 -0.3513054 2.094231e-04
##               diif       lwr         upr      p_adj
## 2:100-2:0 -1.09381 -2.137308 -0.05031069 0.03291067
```

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")
```




```r
# 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
```

```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_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
```

```r
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
```

```r
# 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))]

# 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] 39.6 39.6 40 39.7 35.6 ...
```

```
summary(blocos_dados_9)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       H_AL
##  1:20   0  :16   COM:40   1:40   Min.   :29.18
##  2:20   25 :16   SEM:40   2:40   1st Qu.:33.80
##  3:20   50 :16                   Median :37.74
##  4:20   75 :16                   Mean   :37.90
##         100:16                   3rd Qu.:41.71
##                                  Max.   :52.55
```

```
# 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
```

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

```
##          COM      SEM
## 0   292.3400 293.9233
## 25  286.6400 304.7600
## 50  296.3833 314.7400
## 75  300.6267 285.8900
## 100 332.7700 323.9000
```

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

```
##          COM      SEM
## 0   36.54250 36.74042
## 25  35.83000 38.09500
## 50  37.04792 39.34250
## 75  37.57833 35.73625
## 100 41.59625 40.48750
```

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

```
##           COM      SEM
## 0   15.147993 34.61097
## 25  12.728943 20.72703
## 50  25.601044 35.92011
## 75   9.363013 12.37811
## 100 35.448827 35.76396
```

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

```
##          COM      SEM
## 0   3.892042 5.883109
## 25  3.567764 4.552695
## 50  5.059747 5.993339
## 75  3.059904 3.518253
## 100 5.953892 5.980297
```

```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
##
## 37.89967
##
##  CICLO
## CICLO
##    1    2
## 37.5 38.3
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 38.44 38.47 38.70 35.99
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 36.64 36.96 38.20 36.66 41.04
##
##  INOCULO
## INOCULO
##   COM   SEM
## 37.72 38.08
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 36.13 36.37 38.23 36.90 39.85
##     2 37.16 37.55 38.16 36.41 42.23
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 39.98 35.01
##     2 35.46 41.15
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM   SEM
##       0   36.54 36.74
##       25  35.83 38.09
##       50  37.05 39.34
##       75  37.58 35.74
##       100 41.60 40.49
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 39.71 38.49 41.07 38.93 41.71
##     2 33.37 33.17 33.02 36.23 41.48
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 32.54 34.26 35.38 34.88 37.99
##     2 40.94 41.93 43.30 36.59 42.98
```

```
# 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

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

```
## [1] 0.1203661
```

```
# 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_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] 0.4354531
```

```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  97.59   32.53
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  2.611   2.611
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  124.1   41.37
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1   13.0    13.0   0.974  0.32797
## EFLUENTE                  4  223.5    55.9   4.181  0.00506 **
## CICLO:INOCULO             1  568.5   568.5  42.550 2.45e-08 ***
## CICLO:EFLUENTE            4   20.5     5.1   0.383  0.81978
## INOCULO:EFLUENTE          4   57.6    14.4   1.078  0.37647
## CICLO:INOCULO:EFLUENTE    4  118.7    29.7   2.221  0.07886 .
## Residuals                54  721.5    13.4
## ---
## 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(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, ]
```

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

```
## Analysis of Variance Table
##
## Response: H_AL
##               Df Sum Sq Mean Sq F value    Pr(>F)
## EFLUENTE       4 223.45   55.86  3.7657  0.008712 **
## CICLO:INOCULO  1 568.50  568.50 38.3223 6.999e-08 ***
## 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(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 39.98133
## 2     2     COM 35.45667
## 3     1     SEM 35.01117
## 4     2     SEM 41.14950
```

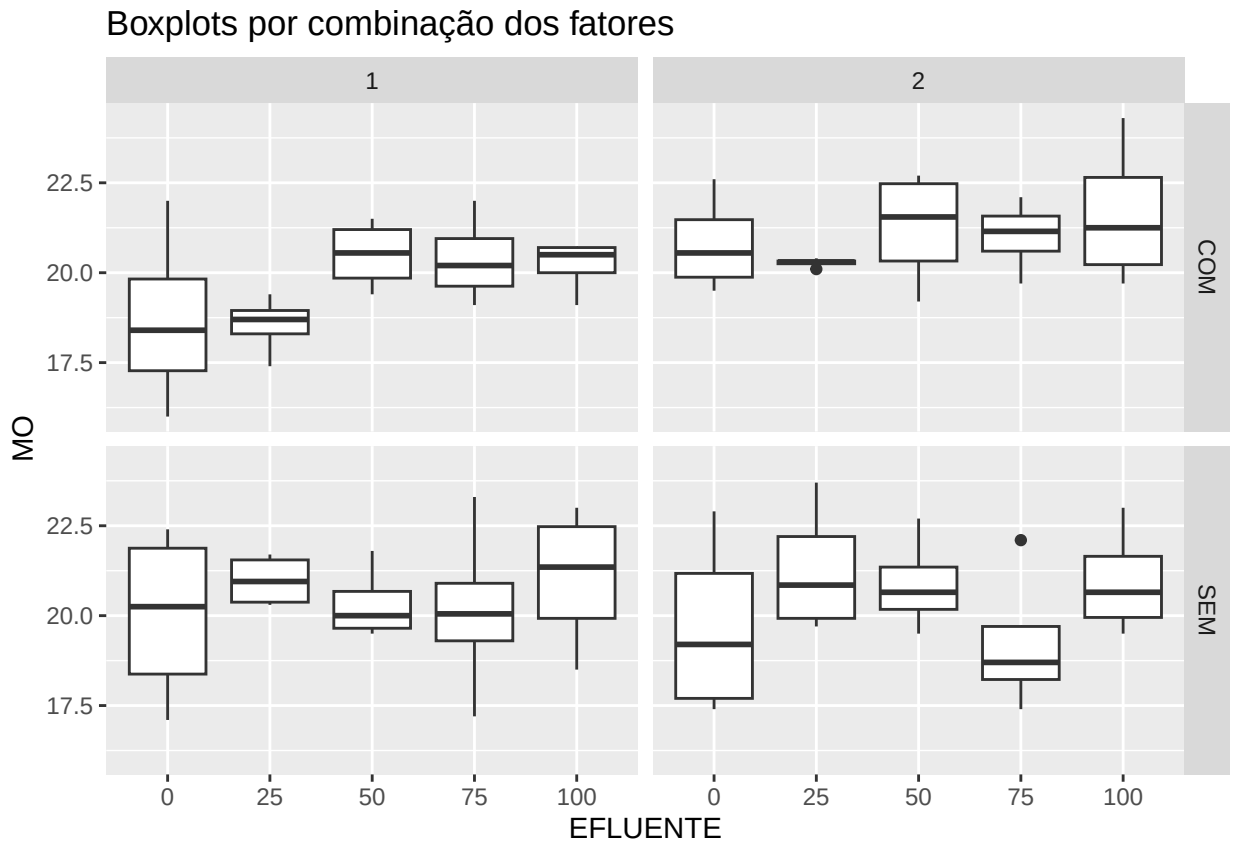
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 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 -4.524667 -7.748007 -1.301326 2.552719e-03
## 1:SEM-1:COM -4.970167 -8.193507 -1.746826 7.956419e-04
## 2:SEM-2:COM  5.692833  2.469493  8.916174 1.068929e-04
## 2:SEM-1:SEM  6.138333  2.914993  9.361674 2.929818e-05
```

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")
```




```
# 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
```

```r
# 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
```

```r
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] 22 19.1 16 17.7 18.8 18.6 17.4 19.4 21.5 19.4 ...
```

```r
summary(blocos_dados_10)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        MO
##  1:20   0  :16   COM:40   1:40   Min.   :16.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:19.50
##  3:20   50 :16                   Median :20.32
##  4:20   75 :16                   Mean   :20.39
##         100:16                   3rd Qu.:21.70
##                                  Max.   :24.30
```

```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   157.6000 167.0000
## 25  155.5333 165.7667
## 50  167.0000 164.8000
## 75  165.6000 157.5000
## 100 167.3000 163.4000
```

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

```
##          COM      SEM
## 0   19.70000 20.87500
```

```
## 25  19.44167 20.72083
## 50  20.87500 20.60000
## 75  20.70000 19.68750
## 100 20.91250 20.42500
```

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

```
##          COM      SEM
## 0   4.697143 3.895714
## 25  1.211032 0.632996
## 50  1.696429 1.328571
## 75  1.222857 4.655536
## 100 2.624107 2.227857
```

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

```
##          COM       SEM
## 0   2.167289 1.9737564
## 25  1.100469 0.7956105
## 50  1.302470 1.1526367
## 75  1.105829 2.1576690
## 100 1.619910 1.4926008
```

```
# 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
##
## 20.39375
##
##  CICLO
## CICLO
##      1      2
## 20.082 20.705
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 21.475 20.510 19.908 19.683
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 20.288 20.081 20.738 20.194 20.669
##
##  INOCULO
```

```
## INOCULO
##    COM    SEM
## 20.326 20.462
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 19.350 19.762 20.413 20.263 20.625
##     2 21.225 20.400 21.062 20.125 20.713
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 19.665 20.500
##     2 20.987 20.423
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   19.700 20.875
##      25  19.442 20.721
##      50  20.875 20.600
##      75  20.700 19.688
##      100 20.913 20.425
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 18.700 18.550 20.500 20.375 20.200
##     2 20.700 20.333 21.250 21.025 21.625
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 20.000 20.975 20.325 20.150 21.050
##     2 21.750 20.467 20.875 19.225 19.800
```

```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)

# 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] 0.321415
```

```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] 0.5175055
```

```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   38.5   12.83
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  0.369   0.369
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  4.154   1.385
##
## Error: Within
##                       Df Sum Sq Mean Sq F value Pr(>F)
```

```
## CICLO                   1   7.75   7.750   4.442 0.0397 *
## EFLUENTE                4   5.48   1.371   0.786 0.5395
## CICLO:INOCULO           1   9.78   9.777   5.603 0.0215 *
## CICLO:EFLUENTE          4   9.73   2.434   1.395 0.2481
## INOCULO:EFLUENTE        4  17.05   4.263   2.443 0.0576 .
## CICLO:INOCULO:EFLUENTE  4   5.21   1.303   0.747 0.5643
## Residuals              54  94.22   1.745
## ---
## 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)
## BLOCO          3 38.499 12.8329  7.4358 0.0002753 ***
## CICLO          1  7.750  7.7501  4.4907 0.0384492 *
## CICLO:INOCULO  1  9.777  9.7767  5.6649 0.0206749 *
## 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"
```

```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(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)
  }
}
```

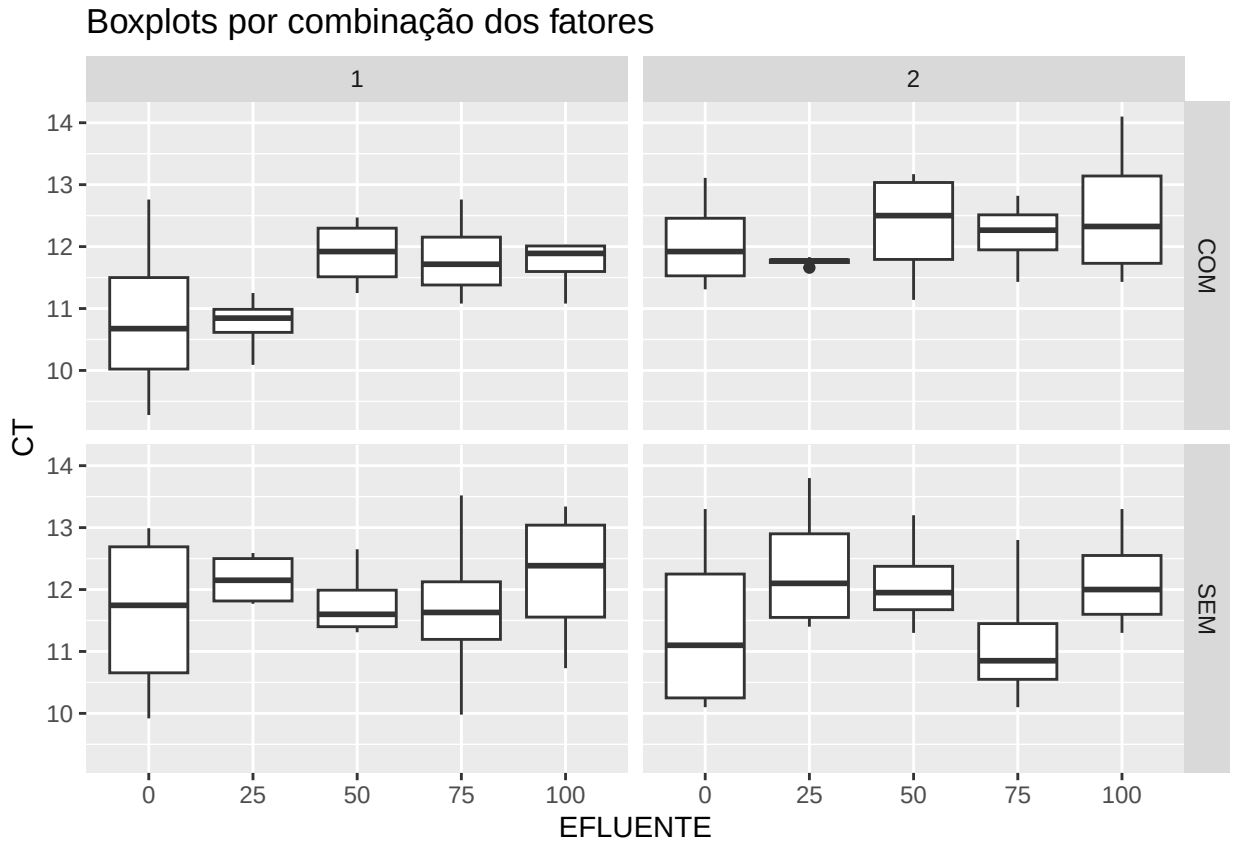
```
##   CICLO INOCULO       MO
## 1     1     COM 19.66500
## 2     2     COM 20.98667
## 3     1     SEM 20.50000
## 4     2     SEM 20.42333
```

```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 1.321667 0.2222398 2.421094 0.0123459
```

Análise para CT

```r
# 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

# 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_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] 12.76 11.08 9.28 10.27 10.9 ...
```

```
summary(blocos_dados_11)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        CT
##  1:20   0  :16   COM:40   1:40   Min.   : 9.28
##  2:20   25 :16   SEM:40   2:40   1st Qu.:11.31
##  3:20   50 :16                   Median :11.78
##  4:20   75 :16                   Mean   :11.83
##         100:16                   3rd Qu.:12.59
##                                  Max.   :14.10
```

```
# 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   91.41667 96.80000
## 25  90.19000 96.12667
## 50  96.87000 95.56000
## 75  96.05000 91.36000
## 100 97.05000 94.84000
```

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

```
##          COM      SEM
## 0   11.42708 12.10000
## 25  11.27375 12.01583
## 50  12.10875 11.94500
## 75  12.00625 11.42000
## 100 12.13125 11.85500
```

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

```
##           COM       SEM
## 0   1.5796681 1.3149429
## 25  0.4062839 0.2209389
## 50  0.5707268 0.4617429
## 75  0.4109125 1.5593143
## 100 0.8854125 0.7475429
```

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

```
##           COM       SEM
## 0   1.2568485 1.1467096
## 25  0.6374041 0.4700414
## 50  0.7554646 0.6795166
## 75  0.6410246 1.2487251
## 100 0.9409636 0.8646056
```

```
# 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
##
## 11.82829
##
##  CICLO
## CICLO
##      1      2
## 11.649 12.008
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 12.459 11.896 11.549 11.410
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 11.764 11.645 12.027 11.713 11.993
##
##  INOCULO
## INOCULO
##    COM    SEM
## 11.789 11.867
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.224 11.461 11.840 11.754 11.964
##     2 12.303 11.828 12.214 11.673 12.023
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 11.406 11.891
##     2 12.173 11.843
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   11.427 12.100
##       25  11.274 12.016
##       50  12.109 11.945
##       75  12.006 11.420
##       100 12.131 11.855
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 10.848 10.758 11.890 11.818 11.718
##     2 12.007 11.790 12.328 12.195 12.545
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 11.600 12.165 11.790 11.690 12.210
##     2 12.600 11.867 12.100 11.150 11.500
```

```
# 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] 0.2912728
```

```
# 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_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] 0.5223927
```

```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   13.1   4.367
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.1209  0.1209
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  1.426  0.4755
##
## Error: Within
##                          Df Sum Sq Mean Sq F value Pr(>F)
## CICLO                     1   2.59   2.586   4.397 0.0407 *
## EFLUENTE                  4   1.88   0.471   0.801 0.5300
## CICLO:INOCULO             1   3.32   3.317   5.641 0.0211 *
## CICLO:EFLUENTE            4   3.21   0.803   1.366 0.2578
## INOCULO:EFLUENTE          4   5.68   1.420   2.415 0.0599 .
## CICLO:INOCULO:EFLUENTE    4   1.70   0.425   0.723 0.5799
## Residuals                54  31.76   0.588
## ---
## 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)

# 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: CT
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## BLOCO          3 13.1014  4.3671  7.5018 0.0002572 ***
## CICLO          1  2.5860  2.5860  4.4422 0.0394719 *
## CICLO:INOCULO  1  3.3171  3.3171  5.6979 0.0203252 *
## 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(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 11.40600
## 2     2     COM 12.17283
## 3     1     SEM 11.89100
## 4     2     SEM 11.84333
```

```
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 0.7668333 0.1282977 1.405369 0.01245755
```

```
# 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 10-20 atualizado.xlsx",
      sheetName = "R - Solo", append = FALSE)
```