Solo 20-40

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 20-40 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 4.7 4.7 4.7 4.5 5.1 4.5 ...
##  $ P resina: num [1:80] 11 10 11 17 9 6 11 10 14 10 ...
##  $ S       : num [1:80] 2 9 9 11 10 12 10 14 7 26 ...
##  $ K resina: num [1:80] 1.2 1.6 0.5 1.8 0.9 1.2 1.2 1.2 1.4 1 ...
##  $ Na      : num [1:80] 0.08 0 0.09 0 0.08 0 0.08 0 0.08 0 ...
##  $ Ca      : num [1:80] 25 26 24 26 18 19 19 16 26 14 ...
##  $ Mg      : num [1:80] 7 7 10 10 7 9 5 7 8 2 ...
##  $ Al      : num [1:80] 0.2 0.5 0.1 0 1.4 0.5 1.5 1.5 0.4 0.5 ...
##  $ H+Al    : num [1:80] 42.6 45.4 36 39.6 43.5 ...
##  $ MO      : num [1:80] 20 19.4 16.9 17.7 18.5 18.6 18.6 17.8 18.2 18 ...
##  $ CT      : num [1:80] 11.6 11.2 9.8 10.3 10.7 ...
```

```r
# 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)
```

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

```r
# 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 4.7 4.7 4.7 4.5 5.1 4.5 ...
```

```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] 11 10 11 17 9 6 11 10 14 10 ...
```

```r
"dados_3"
```

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

```r
str(dados_3)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S       : num [1:80] 2 9 9 11 10 12 10 14 7 26 ...
```

```
"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] 1.2 1.6 0.5 1.8 0.9 1.2 1.2 1.2 1.4 1 ...
```

```
"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 0.09 0 0.08 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] 25 26 24 26 18 19 19 16 26 14 ...
```

```
"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] 7 7 10 10 7 9 5 7 8 2 ...
```

```
"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 1.4 0.5 1.5 1.5 0.4 0.5 ...
```

```
"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] 42.6 45.4 36 39.6 43.5 ...
```

```
"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] 20 19.4 16.9 17.7 18.5 18.6 18.6 17.8 18.2 18 ...
```

```
"dados_11"
```

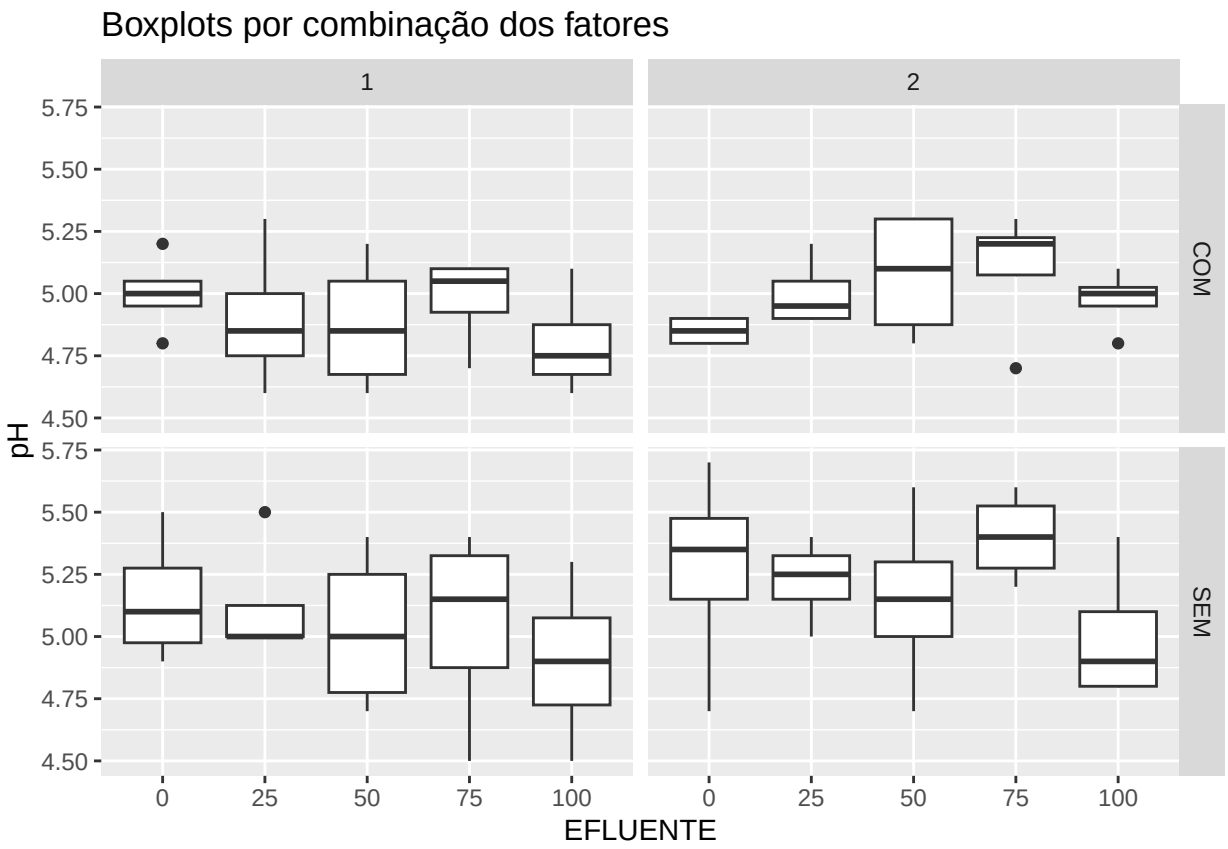
```
## [1] "dados_11"
```

```
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] 11.6 11.2 9.8 10.3 10.7 ...
```

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] 4.93 5 4.8 5 5.3 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        pH
##  1:20   0  :16   COM:40   1:40   Min.   :4.500
##  2:20   25 :16   SEM:40   2:40   1st Qu.:4.800
##  3:20   50 :16                   Median :5.000
##  4:20   75 :16                   Mean   :5.013
##         100:16                   3rd Qu.:5.200
##                                  Max.   :5.700
```

```
# 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   39.13333 41.7
## 25  39.60000 40.9
## 50  39.80000 40.1
## 75  39.70000 41.4
## 100 39.10000 39.6
```

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

```
##          COM    SEM
## 0   4.891667 5.2125
## 25  4.950000 5.1125
## 50  4.975000 5.0125
## 75  4.962500 5.1750
## 100 4.887500 4.9500
```

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

```
##             COM        SEM
## 0   0.007222222 0.10982143
## 25  0.048571429 0.02696429
## 50  0.073571429 0.06696429
## 75  0.033392857 0.08785714
## 100 0.035535714 0.08571429
```

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

```
##            COM       SEM
## 0   0.08498366 0.3313932
## 25  0.22038927 0.1642081
## 50  0.27124054 0.2587746
## 75  0.18273713 0.2964071
## 100 0.18850919 0.2927700
```

```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
##
## 5.012917
##
##  CICLO
## CICLO
##     1     2
## 4.961 5.065
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 4.967 5.003 4.918 5.165
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 5.052 5.031 4.994 5.069 4.919
##
##  INOCULO
## INOCULO
##   COM   SEM
## 4.933 5.093
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.042 4.950 4.950 5.013 4.850
##     2 5.063 5.113 5.038 5.125 4.988
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 4.897 5.025
##     2 4.970 5.160
```

```
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   4.892 5.213
##      25  4.950 5.113
##      50  4.975 5.013
##      75  4.963 5.175
##      100 4.888 4.950
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 4.933 4.900 4.875 4.975 4.800
##     2 4.850 5.000 5.075 4.950 4.975
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 5.150 5.000 5.025 5.050 4.900
##     2 5.275 5.225 5.000 5.300 5.000
```

```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] 0.145961
```

```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] 0.6268683
```

```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 0.6896  0.2299
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1 0.5067  0.5067
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.2773 0.09243
##
## Error: Within
##                         Df Sum Sq Mean Sq F value Pr(>F)
## CICLO                    1 0.2170 0.21701   4.495 0.0386 *
## EFLUENTE                 4 0.2276 0.05689   1.178 0.3307
## CICLO:INOCULO            1 0.0190 0.01901   0.394 0.5330
## CICLO:EFLUENTE           4 0.0472 0.01181   0.245 0.9118
## INOCULO:EFLUENTE         4 0.2126 0.05314   1.101 0.3658
## CICLO:INOCULO:EFLUENTE   4 0.1719 0.04297   0.890 0.4763
## Residuals               54 2.6073 0.04828
## ---
## 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)
## BLOCO     3 0.68962 0.22987  4.5424 0.006354 **
## CICLO     1 0.21701 0.21701  4.2883 0.042915 *
## INOCULO   1 0.50668 0.50668 10.0123 0.002495 **
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

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

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

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

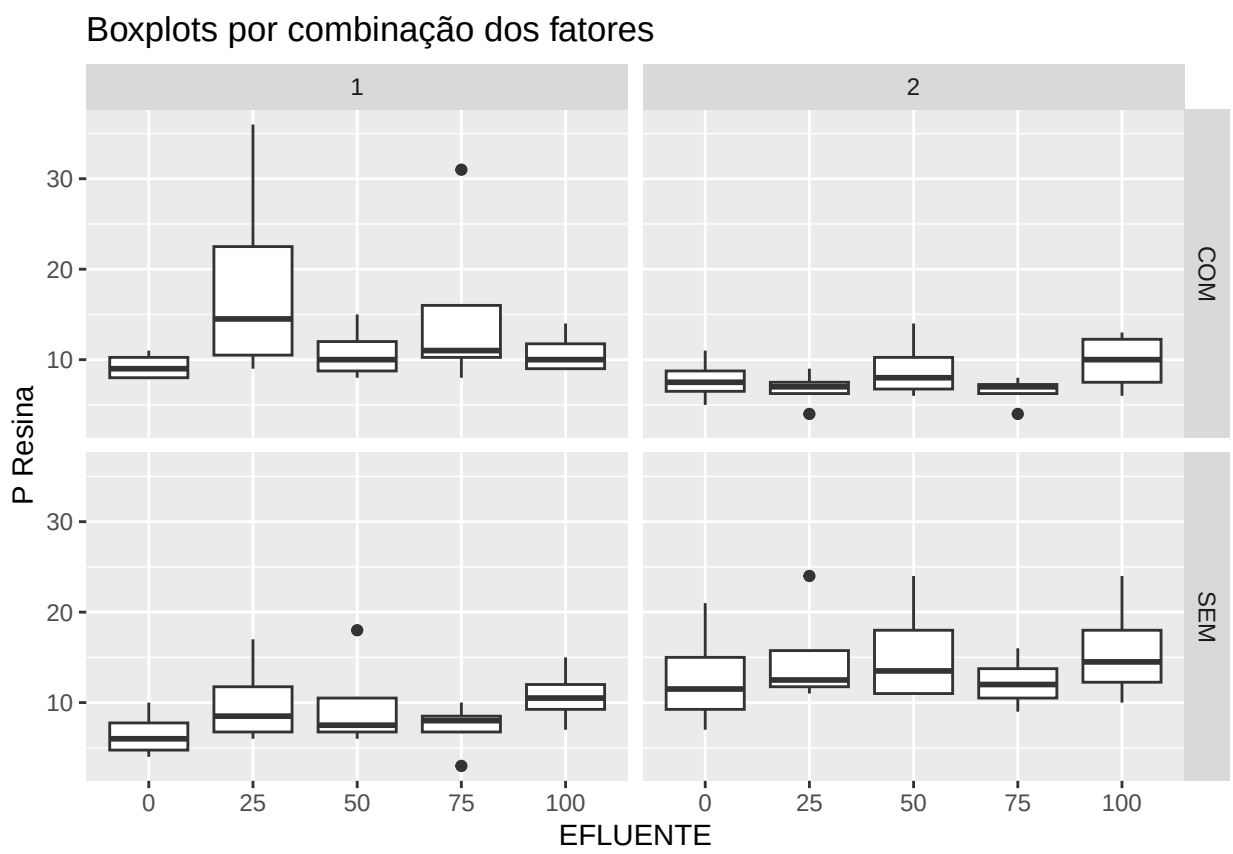
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(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)
  }
}
```

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

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

Análise para 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

# 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
```

```
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
```

```
# 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
rownames(media_blocos_2) <- NULL

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

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

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ P_resina: num [1:80] 11 8 8 10 11 9 18 36 9 11 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     P_resina
##  1:20   0  :16   COM:40   1:40   Min.   : 4.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 7.00
##  3:20   50 :16                   Median : 9.50
##  4:20   75 :16                   Mean   :10.14
##         100:16                   3rd Qu.:11.00
##                                  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    68.00000  77.00000
## 25  101.00000  88.00000
## 50   72.33333  72.00000
## 75   66.00000  78.66667
## 100  82.00000 106.00000
```

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

```
##           COM       SEM
## 0    8.500000  9.625000
## 25  12.625000 11.000000
## 50   9.041667  9.000000
## 75   8.250000  9.833333
## 100 10.250000 13.250000
```

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

```
##            COM       SEM
## 0     4.285714 29.696429
## 25  105.982143 12.000000
## 50    8.109127  4.857143
## 75    5.642857  3.079365
## 100   7.357143 27.357143
```

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

```
##           COM      SEM
## 0    2.070197 5.449443
## 25  10.294763 3.464102
## 50   2.847653 2.203893
## 75   2.375470 1.754812
## 100  2.712405 5.230406
```

```r
# 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
##
## 10.1375
##
##  CICLO
## CICLO
##      1      2
## 10.217 10.058
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 11.50 10.10  8.95 10.00
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
##  9.062 11.812  9.021  9.042 11.750
##
##  INOCULO
## INOCULO
##    COM    SEM
##  9.733 10.542
##
##  CICLO:EFLUENTE
```

```
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  7.875 14.250  8.875  9.333 10.750
##     2 10.250  9.375  9.167  8.750 12.750
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 11.850  8.583
##     2  7.617 12.500
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    8.500  9.625
##      25  12.625 11.000
##      50   9.042  9.000
##      75   8.250  9.833
##      100 10.250 13.250
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  9.250 18.500 10.750 10.000 10.750
##     2  7.750  6.750  7.333  6.500  9.750
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  6.500 10.000  7.000  8.667 10.750
##     2 12.750 12.000 11.000 11.000 15.750
```

```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] 0.0321228
```

```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_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] 0.5183374
```

```
# 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(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  65.74   21.91
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  13.07   13.07
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  59.14   19.71
##
## Error: Within
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                    1    0.5     0.5   0.033   0.8569
## EFLUENTE                 4  144.1    36.0   2.361   0.0647 .
## CICLO:INOCULO            1  332.1   332.1  21.759 2.07e-05 ***
## CICLO:EFLUENTE           4  134.8    33.7   2.208   0.0802 .
## INOCULO:EFLUENTE         4   48.6    12.1   0.796   0.5331
## CICLO:INOCULO:EFLUENTE   4   42.0    10.5   0.689   0.6029
## Residuals               54  824.2    15.3
## ---
## 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(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)
## CICLO:INOCULO   1 332.11  332.11   21.43 2.166e-05 ***
## 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(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 11.850000
## 2     2     COM  7.616667
## 3     1     SEM  8.583333
## 4     2     SEM 12.500000
```

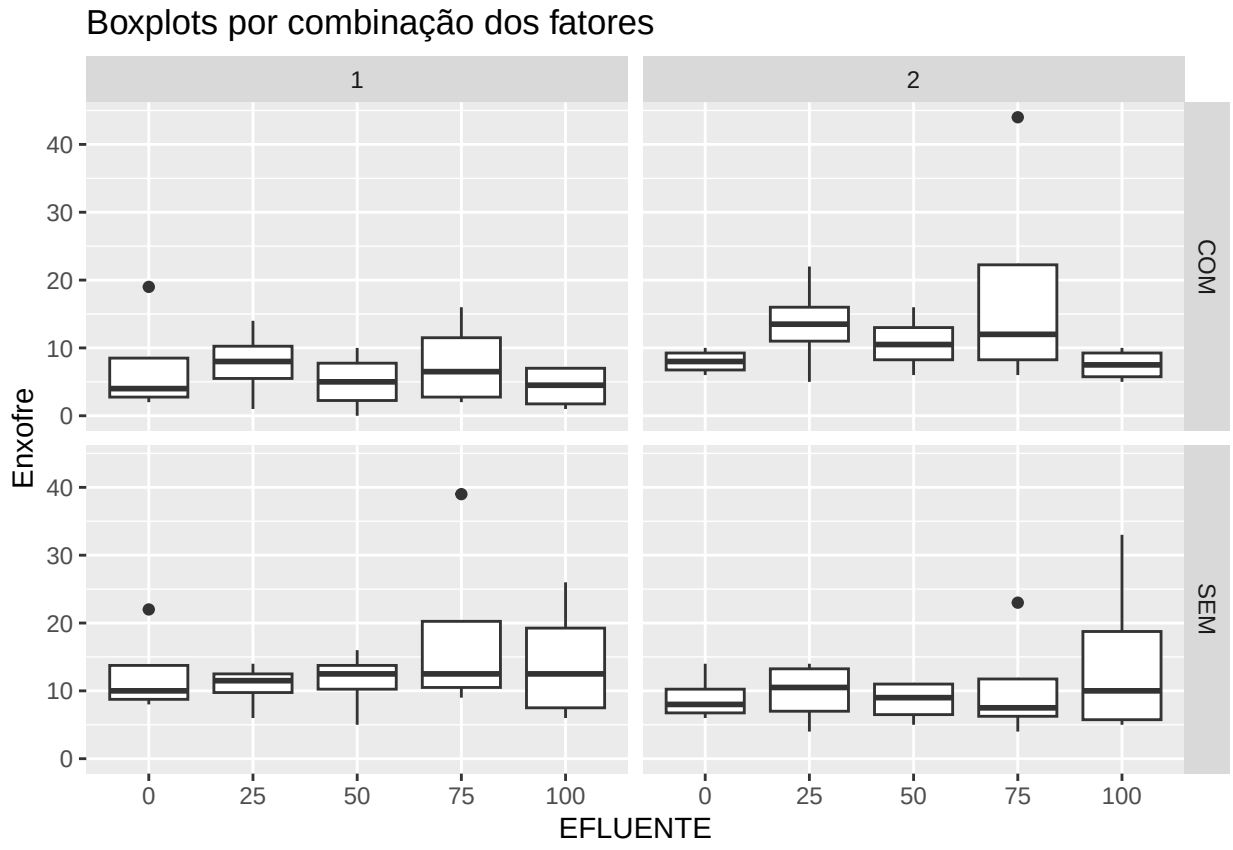
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 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 -4.233333 -7.527879 -0.9387877 0.006589589
## 2:SEM-2:COM  4.883333  1.588788  8.1778790 0.001326252
## 2:SEM-1:SEM  3.916667  0.622121  7.2112123 0.013618726
```

Análise para Enxofre

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ S       : num [1:80] 2 3.33 5 3 9 ...
```

```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.: 6.000
##  3:20   50 :16                   Median : 8.000
##  4:20   75 :16                   Mean   : 8.579
##         100:16                   3rd Qu.:11.000
##                                  Max.   :26.000
```

```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   45.33333 73.33333
```

```
## 25  73.66667 82.00000
## 50  63.00000 80.00000
## 75  61.00000 70.66667
## 100 47.00000 90.33333
```

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

```
##          COM       SEM
## 0   5.666667  9.166667
## 25  9.208333 10.250000
## 50  7.875000 10.000000
## 75  7.625000  8.833333
## 100 5.875000 11.291667
```

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

```
##           COM       SEM
## 0    8.317460  6.126984
## 25  21.775794 14.500000
## 50  25.553571 15.714286
## 75  19.053571 10.190476
## 100  9.839286 53.061508
```

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

```
##          COM      SEM
## 0   2.884001 2.475275
## 25  4.666454 3.807887
## 50  5.055054 3.964125
## 75  4.365040 3.192253
## 100 3.136764 7.284333
```

```
# 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
##
## 8.579167
##
##  CICLO
## CICLO
##     1     2
## 8.525 8.633
##
##  BLOCO
```

```
## BLOCO
##     1     2     3     4
## 9.892 9.917 9.508 5.000
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.417 9.729 8.938 8.229 8.583
##
##  INOCULO
## INOCULO
##   COM   SEM
## 7.250 9.908
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  6.333  9.250  8.250  9.542  9.250
##     2  8.500 10.208  9.625  6.917  7.917
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1  5.617 11.433
##     2  8.883  8.383
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    5.667  9.167
##      25   9.208 10.250
##      50   7.875 10.000
##      75   7.625  8.833
##      100  5.875 11.292
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  3.333  7.750  5.000  7.750  4.250
##     2  8.000 10.667 10.750  7.500  7.500
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  9.333 10.750 11.500 11.333 14.250
##     2  9.000  9.750  8.500  6.333  8.333
```

```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)
```

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

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

```
## [1] 0.883423
```

```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_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] 0.5381499
```

```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  343.7   114.6
##
## Error: INOCULO
```

```
##          Df Sum Sq Mean Sq
## INOCULO   1  141.3   141.3
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3   6.74   2.247
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1    0.2    0.23   0.019 0.889488
## EFLUENTE                  4   46.8   11.70   0.971 0.430778
## CICLO:INOCULO             1  199.5  199.50  16.566 0.000154 ***
## CICLO:EFLUENTE            4   64.5   16.11   1.338 0.267768
## INOCULO:EFLUENTE          4   53.3   13.32   1.106 0.363301
## CICLO:INOCULO:EFLUENTE    4   24.0    6.00   0.498 0.737204
## Residuals                54  650.3   12.04
## ---
## 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)
## BLOCO           3 343.71  114.57  9.9391 2.286e-05 ***
## INOCULO         1 141.33  141.34 12.2611 0.0009062 ***
## CICLO:INOCULO   1 199.50  199.50 17.3072 0.0001084 ***
## 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
```

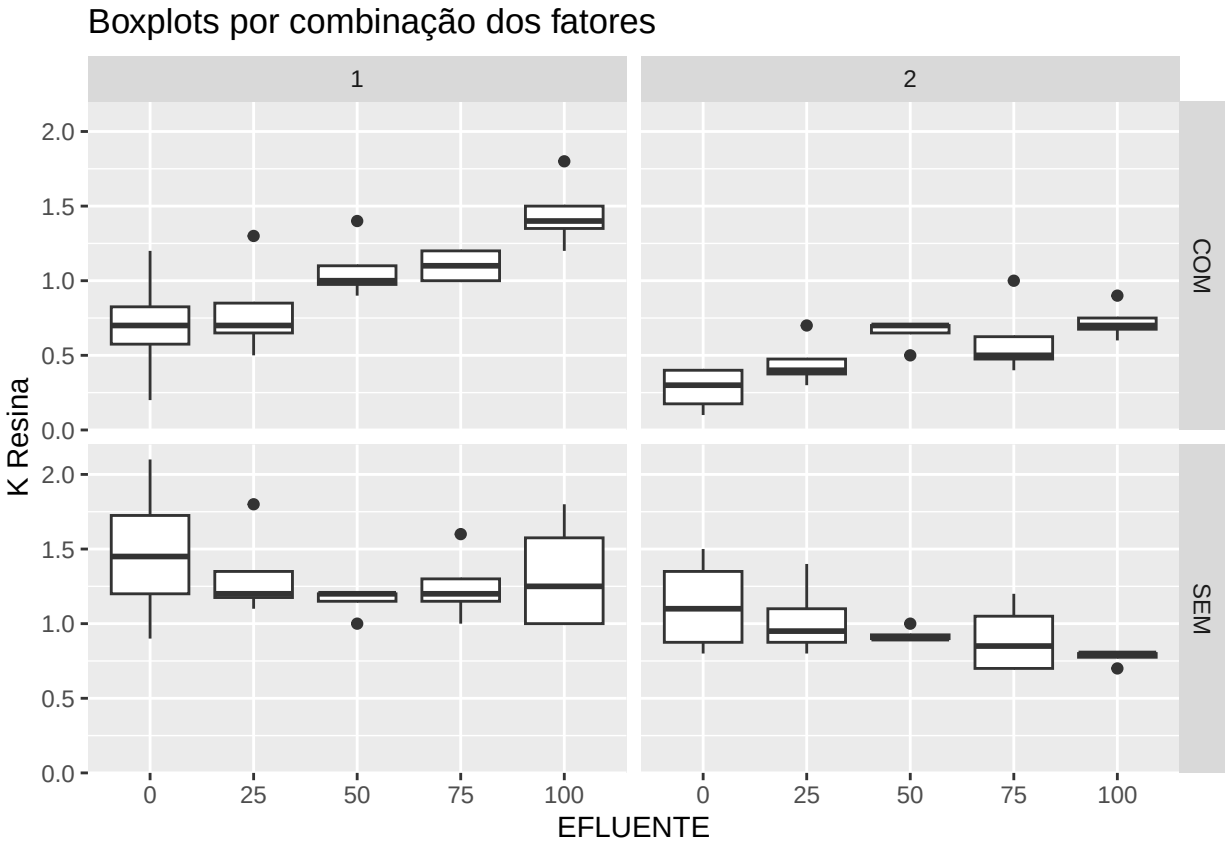
```
## 1     1     COM  5.616667
## 2     2     COM  8.883333
## 3     1     SEM 11.433333
## 4     2     SEM  8.383333
```

```
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  3.266667  0.4253014  6.1080319 1.809149e-02
## 1:SEM-1:COM  5.816667  2.9753014  8.6580319 7.428017e-06
## 2:SEM-1:SEM -3.050000 -5.8913652 -0.2086348 3.080229e-02
```

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))]

# 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] 1.2 0.7 0.7 0.2 0.5 ...
```

```r
summary(blocos_dados_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO     K_resina
##  1:20   0  :16   COM:40   1:40   Min.   :0.1000
##  2:20   25 :16   SEM:40   2:40   1st Qu.:0.7000
##  3:20   50 :16                   Median :0.9000
##  4:20   75 :16                   Mean   :0.9075
##         100:16                   3rd Qu.:1.2000
##                                  Max.   :2.1000
```

```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   3.900000 9.500000
## 25  4.400000 8.766667
## 50  6.466667 8.400000
## 75  6.400000 8.133333
## 100 8.233333 8.400000
```

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

```
##           COM      SEM
## 0   0.4875000 1.187500
## 25  0.5500000 1.095833
## 50  0.8083333 1.050000
## 75  0.8000000 1.016667
## 100 1.0291667 1.050000
```

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

```
##            COM        SEM
## 0   0.13267857 0.20410714
## 25  0.02412698 0.03632937
## 50  0.03388889 0.02571429
## 75  0.10857143 0.04507937
## 100 0.11632937 0.15428571
```

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

```
##           COM       SEM
## 0   0.3642507 0.4517822
## 25  0.1553286 0.1906026
## 50  0.1840894 0.1603567
## 75  0.3295018 0.2123190
## 100 0.3410709 0.3927922
```

```
# 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
##
## 0.9075
##
##  CICLO
## CICLO
##      1      2
## 1.1033 0.7117
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.8933 0.8750 0.9983 0.8633
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.8375 0.8229 0.9292 0.9083 1.0396
##
##  INOCULO
## INOCULO
##   COM   SEM
## 0.735 1.080
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.0875 0.9000 1.0833 1.1167 1.3292
##     2 0.5875 0.7458 0.7750 0.7000 0.7500
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 0.9467 1.2600
##     2 0.5233 0.9000
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   0.4875 1.1875
##       25  0.5500 1.0958
##       50  0.8083 1.0500
##       75  0.8000 1.0167
##       100 1.0292 1.0500
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 0.7000 0.6333 0.9667 1.1000 1.3333
##     2 0.2750 0.4667 0.6500 0.5000 0.7250
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 1.4750 1.1667 1.2000 1.1333 1.3250
##     2 0.9000 1.0250 0.9000 0.9000 0.7750
```

```
# 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] 0.982674
```

```
# 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] 0.7705681
```

```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 0.2292 0.07639
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  2.381   2.381
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.1416  0.0472
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1 3.0681  3.0681  77.993 4.60e-12 ***
## EFLUENTE                  4 0.4795  0.1199   3.048   0.0245 *
## CICLO:INOCULO             1 0.0201  0.0201   0.510   0.4783
## CICLO:EFLUENTE            4 0.4435  0.1109   2.818   0.0339 *
## INOCULO:EFLUENTE          4 1.1944  0.2986   7.590 6.29e-05 ***
## CICLO:INOCULO:EFLUENTE    4 0.1412  0.0353   0.897   0.4721
## Residuals                54 2.1242  0.0393
## ---
## 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 3.06806 3.06806 77.1809 3.469e-12 ***
## INOCULO            1 2.38050 2.38050 59.8846 1.870e-10 ***
## EFLUENTE           4 0.47953 0.11988  3.0158   0.02515 *
## CICLO:EFLUENTE     4 0.44347 0.11087  2.7890   0.03473 *
## INOCULO:EFLUENTE   4 1.19436 0.29859  7.5114 6.231e-05 ***
## 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: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(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)
  }
}
```

```
##    CICLO EFLUENTE  K_resina
## 1      1        0 1.0875000
## 2      2        0 0.5875000
## 3      1       25 0.9000000
## 4      2       25 0.7458333
## 5      1       50 1.0833333
## 6      2       50 0.7750000
## 7      1       75 1.1166667
## 8      2       75 0.7000000
## 9      1      100 1.3291667
## 10     2      100 0.7500000
##    INOCULO EFLUENTE  K_resina
## 1      COM        0 0.4875000
## 2      SEM        0 1.1875000
## 3      COM       25 0.5500000
## 4      SEM       25 1.0958333
## 5      COM       50 0.8083333
## 6      SEM       50 1.0500000
## 7      COM       75 0.8000000
## 8      SEM       75 1.0166667
## 9      COM      100 1.0291667
## 10     SEM      100 1.0500000
```

```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
```

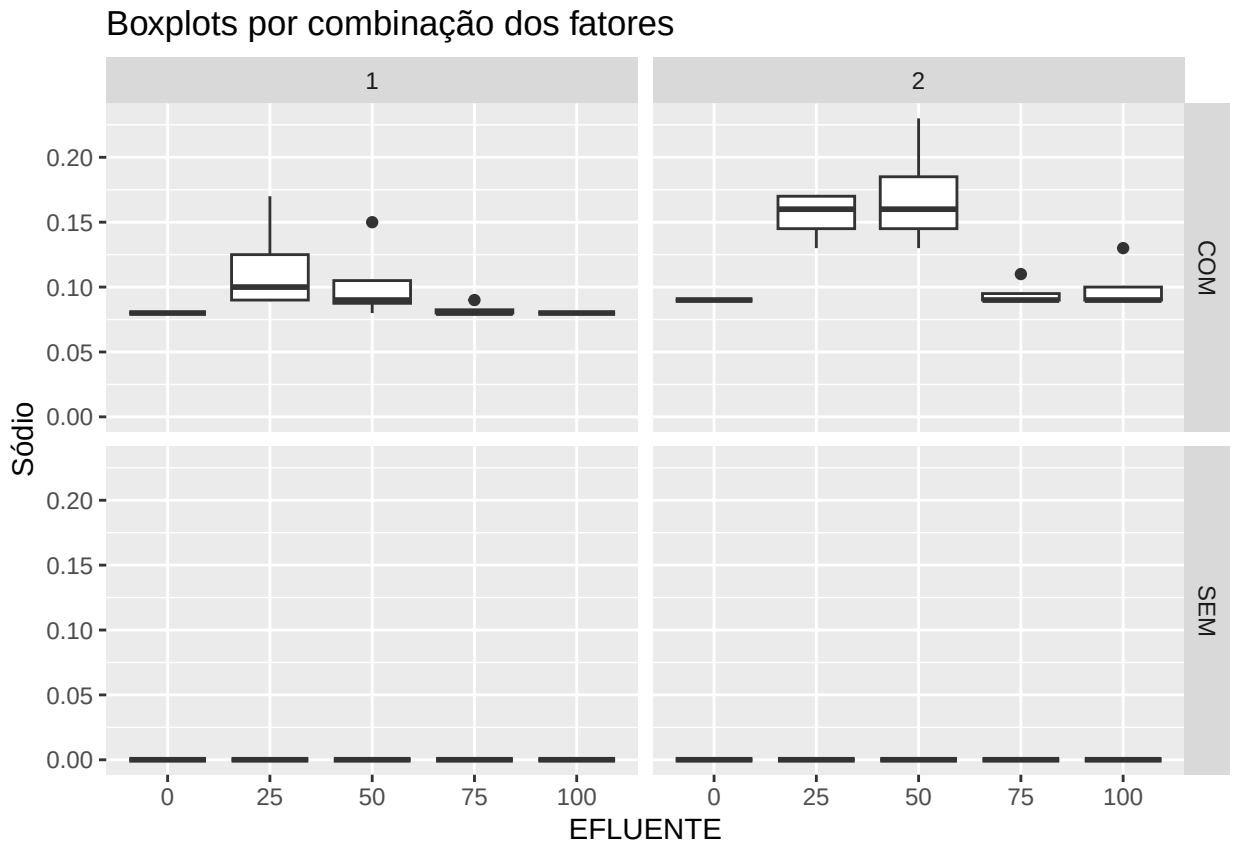
```
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- 0
  }
  else{
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  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
## 2:0-1:0      -0.5000000 -0.828172099 -0.171827901 2.226467e-04
## 2:25-1:0     -0.3416667 -0.669838766 -0.013494568 3.488003e-02
## 2:75-1:0     -0.3875000 -0.715672099 -0.059327901 9.283830e-03
## 2:100-1:0    -0.3375000 -0.665672099 -0.009327901 3.904351e-02
## 1:50-2:0      0.4958333  0.167661234  0.824005432 2.579217e-04
## 1:75-2:0      0.5291667  0.200994568  0.857338766 7.842420e-05
## 1:100-2:0     0.7416667  0.413494568  1.069838766 2.596165e-08
## 1:100-1:25    0.4291667  0.100994568  0.757338766 2.492295e-03
## 1:50-2:25     0.3375000  0.009327901  0.665672099 3.904351e-02
## 1:75-2:25     0.3708333  0.042661234  0.699005432 1.527450e-02
## 1:100-2:25    0.5833333  0.255161234  0.911505432 1.071215e-05
## 2:75-1:50    -0.3833333 -0.711505432 -0.055161234 1.053173e-02
## 2:100-1:50   -0.3333333 -0.661505432 -0.005161234 4.364317e-02
## 1:75-2:50     0.3416667  0.013494568  0.669838766 3.488003e-02
## 1:100-2:50    0.5541667  0.225994568  0.882338766 3.152408e-05
## 2:75-1:75    -0.4166667 -0.744838766 -0.088494568 3.733563e-03
## 2:100-1:75   -0.3666667 -0.694838766 -0.038494568 1.725047e-02
## 1:100-2:75    0.6291667  0.300994568  0.957338766 1.912989e-06
## 2:100-1:100 -0.5791667 -0.907338766 -0.250994568 1.250966e-05
##                      diif        lwr         upr        p_adj
## SEM:0-COM:0     0.7000000  0.3718279  1.02817210 1.283864e-07
## SEM:25-COM:0    0.6083333  0.2801612  0.93650543 4.200079e-06
## SEM:50-COM:0    0.5625000  0.2343279  0.89067210 2.319589e-05
## SEM:75-COM:0    0.5291667  0.2009946  0.85733877 7.842420e-05
## COM:100-COM:0   0.5416667  0.2134946  0.86983877 4.980971e-05
## SEM:100-COM:0   0.5625000  0.2343279  0.89067210 2.319589e-05
## COM:25-SEM:0   -0.6375000 -0.9656721 -0.30932790 1.394846e-06
## COM:50-SEM:0   -0.3791667 -0.7073388 -0.05099457 1.193452e-02
## COM:75-SEM:0   -0.3875000 -0.7156721 -0.05932790 9.283830e-03
## SEM:25-COM:25   0.5458333  0.2176612  0.87400543 4.278127e-05
## SEM:50-COM:25   0.5000000  0.1718279  0.82817210 2.226467e-04
## SEM:75-COM:25   0.4666667  0.1384946  0.79483877 7.105846e-04
## COM:100-COM:25  0.4791667  0.1509946  0.80733877 4.619045e-04
```

```
## SEM:100-COM:25  0.5000000  0.1718279  0.82817210 2.226467e-04
```

Análise para Sódio

```r
# 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

# 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
```

```
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
```

```
# 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
```

```r
# 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 0.09 0.09 0.11 0.17 0.08 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   :0.05752
##         100:16                   3rd Qu.:0.09000
##                                  Max.   :0.23000
##                                  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  1.080000   0
## 50  1.266667  NA
## 75  0.680000  NA
## 100 0.320000  NA
```

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

```
##           COM SEM
## 0   0.0850000   0
## 25  0.1350000   0
## 50  0.1583333  NA
## 75  0.0850000  NA
## 100 0.0400000  NA
```

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

```
##              COM SEM
## 0   2.857143e-05   0
## 25  1.228571e-03   0
## 50  5.879365e-03  NA
## 75  2.857143e-05  NA
## 100 1.828571e-03  NA
```

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

```
##             COM SEM
## 0   0.005345225   0
## 25  0.035050983   0
## 50  0.076677018  NA
## 75  0.005345225  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
##
## 0.05752381
##
##  CICLO
##            1        2
##      0.04417  0.07533
## rep 40.00000 30.00000
##
##  BLOCO
##            1        2        3        4
##      0.05451  0.05284  0.05935  0.06386
## rep 18.00000 18.00000 17.00000 17.00000
##
##  EFLUENTE
##            0       25       50       75      100
##      0.04014  0.06514  0.09093  0.05955  0.02955
## rep 16.00000 16.00000 14.00000 12.00000 12.00000
##
##  INOCULO
##          COM       SEM
##      0.09883  0.002452
## rep 40.00000 30.000000
##
##  CICLO:EFLUENTE
```

```
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.030  0.048  0.042  0.050  0.050
##   rep  8.000  8.000  8.000  8.000  8.000
##   2    0.055  0.087  0.156  0.069 -0.021
##   rep  8.000  8.000  6.000  4.000  4.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##   1    0.085  0.003
##   rep 20.000 20.000
##   2    0.117 -0.008
##   rep 20.000 10.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    0.075  0.006
##      rep  8.000  8.000
##      25   0.125  0.006
##      rep  8.000  8.000
##      50   0.146  0.018
##      rep  8.000  6.000
##      75   0.088  0.002
##      rep  8.000  4.000
##      100  0.058 -0.028
##      rep  8.000  4.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.080 0.115 0.087 0.080 0.080
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.090 0.155 0.230 0.090 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 0.004
##   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
```

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

```
## [1] 0.00129535
```

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

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

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

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

```
## [1] 0.6085548
```

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

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

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

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

```
##
## Error: BLOCO
##       Df   Sum Sq   Mean Sq
## BLOCO  3 0.001033 0.0003444
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1 0.1731  0.1731
##
```

```
## Error: BLOCO:INOCULO
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## CICLO      1 0.0001715 1.715e-04   2.432  0.259
## Residuals  2 0.0001410 7.051e-05
##
## Error: Within
##                         Df  Sum Sq  Mean Sq F value   Pr(>F)
## CICLO                    1 0.00366 0.003664   35.66 3.20e-07 ***
## EFLUENTE                 4 0.04032 0.010081   98.11  < 2e-16 ***
## CICLO:INOCULO            1 0.00538 0.005377   52.33 4.06e-09 ***
## CICLO:EFLUENTE           4 0.06030 0.015075  146.71  < 2e-16 ***
## INOCULO:EFLUENTE         4 0.00894 0.002234   21.74 4.01e-10 ***
## CICLO:INOCULO:EFLUENTE   2 0.00829 0.004146   40.35 7.59e-11 ***
## Residuals               46 0.00473 0.000103
## ---
## 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 0.001033 0.000344    3.3531   0.02626 *
## CICLO                    1 0.016917 0.016917  164.7089 < 2.2e-16 ***
## INOCULO                  1 0.159927 0.159927 1557.0573 < 2.2e-16 ***
## EFLUENTE                 4 0.040425 0.010106   98.3956 < 2.2e-16 ***
## CICLO:INOCULO            1 0.005335 0.005335   51.9407 3.172e-09 ***
## CICLO:EFLUENTE           4 0.060344 0.015086  146.8787 < 2.2e-16 ***
## INOCULO:EFLUENTE         4 0.008930 0.002232   21.7355 2.315e-10 ***
## CICLO:INOCULO:EFLUENTE   2 0.008137 0.004069   39.6114 5.816e-11 ***
## 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]
```

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

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

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

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

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(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 0.08833333
## 2     2     COM 0.11300000
## 3     1     SEM 0.00000000
## 4     2     SEM 0.00000000
##    CICLO EFLUENTE         Na
## 1      1        0 0.04000000
## 2      2        0 0.04500000
## 3      1       25 0.05750000
## 4      2       25 0.07750000
## 5      1       50 0.04333333
## 6      2       50 0.15333333
## 7      1       75 0.04000000
## 8      2       75 0.09000000
## 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 0.1350000
## 4      SEM       25 0.0000000
## 5      COM       50 0.1583333
## 6      SEM       50 0.0000000
## 7      COM       75 0.0850000
## 8      SEM       75 0.0000000
## 9      COM      100 0.0400000
## 10     SEM      100 0.0000000
##    CICLO INOCULO EFLUENTE         Na
## 1      1     COM        0 0.08000000
## 2      2     COM        0 0.09000000
## 3      1     SEM        0 0.00000000
## 4      2     SEM        0 0.00000000
## 5      1     COM       25 0.11500000
## 6      2     COM       25 0.15500000
## 7      1     SEM       25 0.00000000
## 8      2     SEM       25 0.00000000
## 9      1     COM       50 0.08666667
## 10     2     COM       50 0.23000000
## 11     1     SEM       50 0.00000000
## 12     2     SEM       50 0.00000000
## 13     1     COM       75 0.08000000
## 14     2     COM       75 0.09000000
## 15     1     SEM       75 0.00000000
## 16     1     COM      100 0.08000000
## 17     2     COM      100 0.00000000
## 18     1     SEM      100 0.00000000
```

```
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  0.02919209  0.02066897  0.03771522 2.040756e-11
## 1:SEM-1:COM -0.08380791 -0.09233103 -0.07528478 0.000000e+00
## 2:SEM-1:COM -0.08990487 -0.10034352 -0.07946622 0.000000e+00
## 1:SEM-2:COM -0.11300000 -0.12152312 -0.10447688 0.000000e+00
## 2:SEM-2:COM -0.11909696 -0.12953562 -0.10865831 0.000000e+00
##                     diif           lwr          upr        p_adj
## 2:0-1:0      0.02599205  0.0092023835  0.042781724 1.995443e-04
## 1:25-1:0     0.01750000  0.0007103299  0.034289670 3.473877e-02
## 2:25-1:0     0.05849205  0.0417023835  0.075281724 0.000000e+00
## 2:50-1:0     0.11653440  0.0983994830  0.134669316 0.000000e+00
## 2:100-1:0   -0.07502973 -0.0955927878 -0.054466663 0.000000e+00
## 2:25-2:0     0.03250000  0.0157103299  0.049289670 2.301630e-06
## 1:50-2:0    -0.02060715 -0.0373968227 -0.003817482 6.083203e-03
## 2:50-2:0     0.09054235  0.0724074294  0.108677262 0.000000e+00
## 1:75-2:0    -0.02135703 -0.0381467004 -0.004567360 3.876119e-03
## 1:100-2:0   -0.02135703 -0.0381467004 -0.004567360 3.876119e-03
## 2:100-2:0   -0.10102178 -0.1215848414 -0.080458717 0.000000e+00
## 2:25-1:25    0.04099205  0.0242023835  0.057781724 6.139678e-09
## 2:50-1:25    0.09903440  0.0808994830  0.117169316 0.000000e+00
## 2:100-1:25  -0.09252973 -0.1130927878 -0.071966663 0.000000e+00
## 1:50-2:25   -0.05310715 -0.0698968227 -0.036317482 0.000000e+00
## 2:50-2:25    0.05804235  0.0399074294  0.076177262 0.000000e+00
## 1:75-2:25   -0.05385703 -0.0706467004 -0.037067360 0.000000e+00
## 2:75-2:25   -0.04352178 -0.0640848414 -0.022958717 2.753608e-07
## 1:100-2:25  -0.05385703 -0.0706467004 -0.037067360 0.000000e+00
## 2:100-2:25  -0.13352178 -0.1540848414 -0.112958717 0.000000e+00
## 2:50-1:50    0.11114950  0.0930145819  0.129284415 0.000000e+00
## 2:100-1:50  -0.08041463 -0.1009776888 -0.059851564 0.000000e+00
## 1:75-2:50   -0.11189938 -0.1300342925 -0.093764460 0.000000e+00
## 2:75-2:50   -0.10156412 -0.1232394958 -0.079888754 0.000000e+00
## 1:100-2:50  -0.11189938 -0.1300342925 -0.093764460 0.000000e+00
```

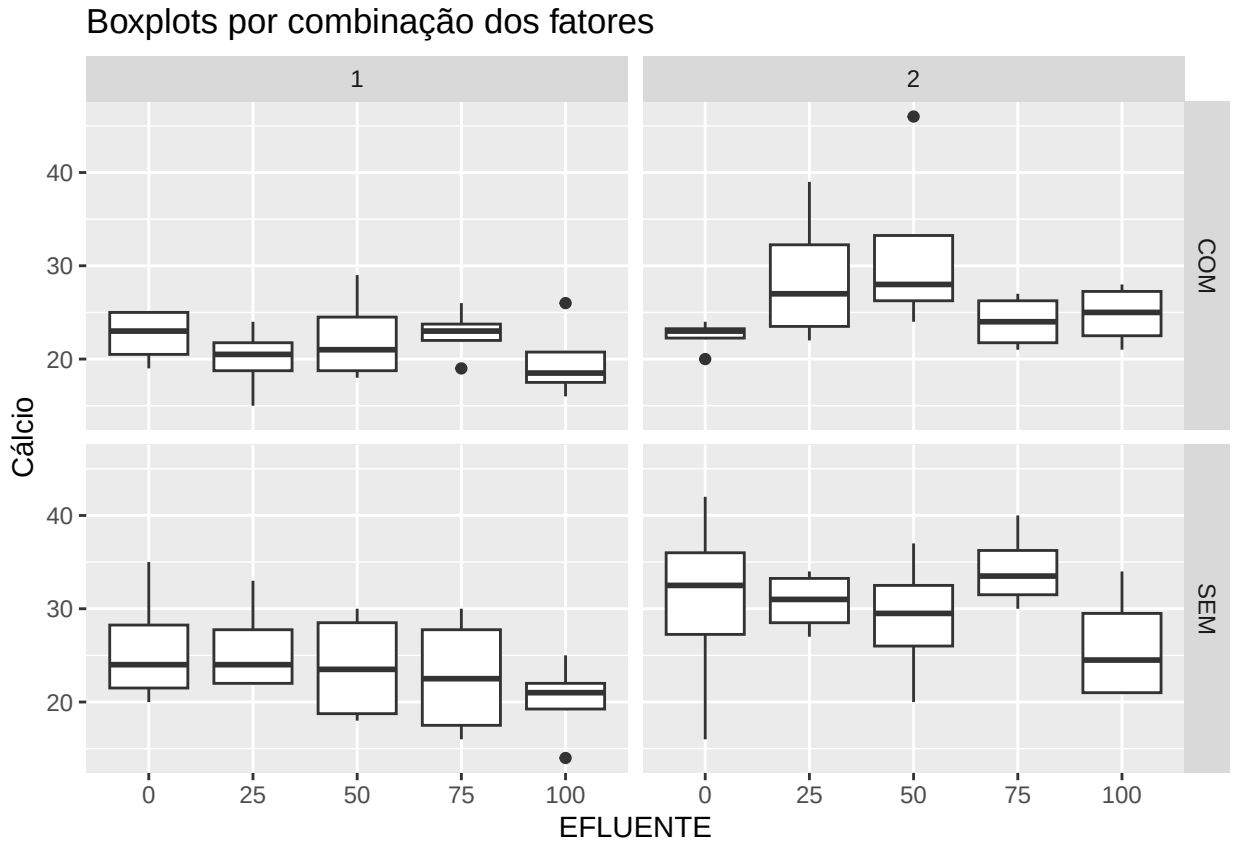
```
## 2:100-2:50  -0.19156412 -0.2132394958 -0.169888754 0.000000e+00
## 2:100-1:75  -0.07966475 -0.1002278112 -0.059101686 0.000000e+00
## 2:100-2:75  -0.09000000 -0.1137441792 -0.066255821 0.000000e+00
## 2:100-1:100 -0.07966475 -0.1002278112 -0.059101686 0.000000e+00
##                        diif          lwr          upr        p_adj
## SEM:0-COM:0     -0.06912758 -0.085917255 -0.052337915 0.000000e+00
## COM:25-COM:0     0.05000000  0.033210330  0.066789670 1.011780e-11
## SEM:25-COM:0    -0.06912758 -0.085917255 -0.052337915 0.000000e+00
## COM:50-COM:0     0.07112278  0.054333107  0.087912447 0.000000e+00
## SEM:50-COM:0    -0.05697307 -0.075107986 -0.038838153 0.000000e+00
## SEM:75-COM:0    -0.07236732 -0.092930385 -0.051804260 0.000000e+00
## SEM:100-COM:0   -0.10236732 -0.122930385 -0.081804260 0.000000e+00
## COM:25-SEM:0     0.11912758  0.102337915  0.135917255 0.000000e+00
## COM:50-SEM:0     0.14025036  0.123460692  0.157040032 0.000000e+00
## COM:75-SEM:0     0.08257338  0.065783712  0.099363052 0.000000e+00
## COM:100-SEM:0    0.05257338  0.035783712  0.069363052 0.000000e+00
## SEM:100-SEM:0   -0.03323974 -0.053802800 -0.012676675 9.239074e-05
## SEM:25-COM:25   -0.11912758 -0.135917255 -0.102337915 0.000000e+00
## COM:50-COM:25    0.02112278  0.004333107  0.037912447 4.466939e-03
## SEM:50-COM:25   -0.10697307 -0.125107986 -0.088838153 0.000000e+00
## COM:75-COM:25   -0.03655420 -0.053343873 -0.019764533 1.349678e-07
## SEM:75-COM:25   -0.12236732 -0.142930385 -0.101804260 0.000000e+00
## COM:100-COM:25  -0.06655420 -0.083343873 -0.049764533 0.000000e+00
## SEM:100-COM:25  -0.15236732 -0.172930385 -0.131804260 0.000000e+00
## COM:50-SEM:25    0.14025036  0.123460692  0.157040032 0.000000e+00
## COM:75-SEM:25    0.08257338  0.065783712  0.099363052 0.000000e+00
## COM:100-SEM:25   0.05257338  0.035783712  0.069363052 0.000000e+00
## SEM:100-SEM:25  -0.03323974 -0.053802800 -0.012676675 9.239074e-05
## SEM:50-COM:50   -0.12809585 -0.146230763 -0.109960930 0.000000e+00
## COM:75-COM:50   -0.05767698 -0.074466650 -0.040887310 0.000000e+00
## SEM:75-COM:50   -0.14349010 -0.164053162 -0.122927037 0.000000e+00
## COM:100-COM:50  -0.08767698 -0.104466650 -0.070887310 0.000000e+00
## SEM:100-COM:50  -0.17349010 -0.194053162 -0.152927037 0.000000e+00
## COM:75-SEM:50    0.07041887  0.052283950  0.088553783 0.000000e+00
## COM:100-SEM:50   0.04041887  0.022283950  0.058553783 7.364759e-08
## SEM:100-SEM:50  -0.04539425 -0.067069624 -0.023718882 3.575705e-07
## SEM:75-COM:75   -0.08581312 -0.106376182 -0.065250057 0.000000e+00
## COM:100-COM:75  -0.03000000 -0.046789670 -0.013210330 1.307446e-05
## SEM:100-COM:75  -0.11581312 -0.136376182 -0.095250057 0.000000e+00
## COM:100-SEM:75   0.05581312  0.035250057  0.076376182 2.637536e-10
## SEM:100-SEM:75  -0.03000000 -0.053744179 -0.006255821 4.229366e-03
## SEM:100-COM:100 -0.08581312 -0.106376182 -0.065250057 0.000000e+00
##                           diif          lwr          upr        p_adj
## 1:SEM:0-1:COM:0     -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:SEM:0-1:COM:0     -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 1:COM:25-1:COM:0     0.03500000  0.008177202  0.061822798 1.690955e-03
## 2:COM:25-1:COM:0     0.07500000  0.048177202  0.101822798 4.333534e-12
## 1:SEM:25-1:COM:0    -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:SEM:25-1:COM:0    -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:COM:50-1:COM:0     0.15000000  0.123177202  0.176822798 0.000000e+00
## 1:SEM:50-1:COM:0    -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:SEM:50-1:COM:0    -0.07654412 -0.109395202 -0.043693034 2.826161e-09
## 1:SEM:75-1:COM:0    -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:COM:100-1:COM:0   -0.08000000 -0.106822798 -0.053177202 0.000000e+00
```

```
## 1:SEM:100-1:COM:0   -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 1:SEM:0-2:COM:0     -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:SEM:0-2:COM:0     -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:COM:25-2:COM:0     0.06500000  0.038177202  0.091822798 8.505502e-10
## 1:SEM:25-2:COM:0    -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:SEM:25-2:COM:0    -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:COM:50-2:COM:0     0.14000000  0.113177202  0.166822798 0.000000e+00
## 1:SEM:50-2:COM:0    -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:SEM:50-2:COM:0    -0.08654412 -0.119395202 -0.053693034 5.572198e-11
## 1:SEM:75-2:COM:0    -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:COM:100-2:COM:0   -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 1:SEM:100-2:COM:0   -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 1:COM:25-1:SEM:0     0.11500000  0.088177202  0.141822798 0.000000e+00
## 2:COM:25-1:SEM:0     0.15500000  0.128177202  0.181822798 0.000000e+00
## 1:COM:50-1:SEM:0     0.08666667  0.059843869  0.113489465 0.000000e+00
## 2:COM:50-1:SEM:0     0.23000000  0.203177202  0.256822798 0.000000e+00
## 1:COM:75-1:SEM:0     0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:75-1:SEM:0     0.09000000  0.063177202  0.116822798 0.000000e+00
## 1:COM:100-1:SEM:0    0.08000000  0.053177202  0.106822798 0.000000e+00
## 1:COM:25-2:SEM:0     0.11500000  0.088177202  0.141822798 0.000000e+00
## 2:COM:25-2:SEM:0     0.15500000  0.128177202  0.181822798 0.000000e+00
## 1:COM:50-2:SEM:0     0.08666667  0.059843869  0.113489465 0.000000e+00
## 2:COM:50-2:SEM:0     0.23000000  0.203177202  0.256822798 0.000000e+00
## 1:COM:75-2:SEM:0     0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:75-2:SEM:0     0.09000000  0.063177202  0.116822798 0.000000e+00
## 1:COM:100-2:SEM:0    0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:25-1:COM:25    0.04000000  0.013177202  0.066822798 1.656897e-04
## 1:SEM:25-1:COM:25   -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 2:SEM:25-1:COM:25   -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 1:COM:50-1:COM:25   -0.02833333 -0.055156131 -0.001510535 2.829755e-02
## 2:COM:50-1:COM:25    0.11500000  0.088177202  0.141822798 0.000000e+00
## 1:SEM:50-1:COM:25   -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 2:SEM:50-1:COM:25   -0.11154412 -0.144395202 -0.078693034 0.000000e+00
## 1:COM:75-1:COM:25   -0.03500000 -0.061822798 -0.008177202 1.690955e-03
## 1:SEM:75-1:COM:25   -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 1:COM:100-1:COM:25  -0.03500000 -0.061822798 -0.008177202 1.690955e-03
## 2:COM:100-1:COM:25  -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 1:SEM:100-1:COM:25  -0.11500000 -0.141822798 -0.088177202 0.000000e+00
## 1:SEM:25-2:COM:25   -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 2:SEM:25-2:COM:25   -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 1:COM:50-2:COM:25   -0.06833333 -0.095156131 -0.041510535 1.727827e-10
## 2:COM:50-2:COM:25    0.07500000  0.048177202  0.101822798 4.333534e-12
## 1:SEM:50-2:COM:25   -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 2:SEM:50-2:COM:25   -0.15154412 -0.184395202 -0.118693034 0.000000e+00
## 1:COM:75-2:COM:25   -0.07500000 -0.101822798 -0.048177202 4.333534e-12
## 2:COM:75-2:COM:25   -0.06500000 -0.091822798 -0.038177202 8.505502e-10
## 1:SEM:75-2:COM:25   -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 1:COM:100-2:COM:25  -0.07500000 -0.101822798 -0.048177202 4.333534e-12
## 2:COM:100-2:COM:25  -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 1:SEM:100-2:COM:25  -0.15500000 -0.181822798 -0.128177202 0.000000e+00
## 1:COM:50-1:SEM:25    0.08666667  0.059843869  0.113489465 0.000000e+00
## 2:COM:50-1:SEM:25    0.23000000  0.203177202  0.256822798 0.000000e+00
## 1:COM:75-1:SEM:25    0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:75-1:SEM:25    0.09000000  0.063177202  0.116822798 0.000000e+00
```

```
## 1:COM:100-1:SEM:25   0.08000000  0.053177202  0.106822798 0.000000e+00
## 1:COM:50-2:SEM:25    0.08666667  0.059843869  0.113489465 0.000000e+00
## 2:COM:50-2:SEM:25    0.23000000  0.203177202  0.256822798 0.000000e+00
## 1:COM:75-2:SEM:25    0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:75-2:SEM:25    0.09000000  0.063177202  0.116822798 0.000000e+00
## 1:COM:100-2:SEM:25   0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:50-1:COM:50    0.14333333  0.116510535  0.170156131 0.000000e+00
## 1:SEM:50-1:COM:50   -0.08666667 -0.113489465 -0.059843869 0.000000e+00
## 2:SEM:50-1:COM:50   -0.08321078 -0.116061868 -0.050359700 2.085070e-10
## 1:SEM:75-1:COM:50   -0.08666667 -0.113489465 -0.059843869 0.000000e+00
## 2:COM:100-1:COM:50  -0.08666667 -0.113489465 -0.059843869 0.000000e+00
## 1:SEM:100-1:COM:50  -0.08666667 -0.113489465 -0.059843869 0.000000e+00
## 1:SEM:50-2:COM:50   -0.23000000 -0.256822798 -0.203177202 0.000000e+00
## 2:SEM:50-2:COM:50   -0.22654412 -0.259395202 -0.193693034 0.000000e+00
## 1:COM:75-2:COM:50   -0.15000000 -0.176822798 -0.123177202 0.000000e+00
## 2:COM:75-2:COM:50   -0.14000000 -0.166822798 -0.113177202 0.000000e+00
## 1:SEM:75-2:COM:50   -0.23000000 -0.256822798 -0.203177202 0.000000e+00
## 1:COM:100-2:COM:50  -0.15000000 -0.176822798 -0.123177202 0.000000e+00
## 2:COM:100-2:COM:50  -0.23000000 -0.256822798 -0.203177202 0.000000e+00
## 1:SEM:100-2:COM:50  -0.23000000 -0.256822798 -0.203177202 0.000000e+00
## 1:COM:75-1:SEM:50    0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:75-1:SEM:50    0.09000000  0.063177202  0.116822798 0.000000e+00
## 1:COM:100-1:SEM:50   0.08000000  0.053177202  0.106822798 0.000000e+00
## 1:COM:75-2:SEM:50    0.07654412  0.043693034  0.109395202 2.826161e-09
## 2:COM:75-2:SEM:50    0.08654412  0.053693034  0.119395202 5.572198e-11
## 1:COM:100-2:SEM:50   0.07654412  0.043693034  0.109395202 2.826161e-09
## 1:SEM:75-1:COM:75   -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 2:COM:100-1:COM:75  -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 1:SEM:100-1:COM:75  -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 1:SEM:75-2:COM:75   -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 2:COM:100-2:COM:75  -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 1:SEM:100-2:COM:75  -0.09000000 -0.116822798 -0.063177202 0.000000e+00
## 1:COM:100-1:SEM:75   0.08000000  0.053177202  0.106822798 0.000000e+00
## 2:COM:100-1:COM:100 -0.08000000 -0.106822798 -0.053177202 0.000000e+00
## 1:SEM:100-1:COM:100 -0.08000000 -0.106822798 -0.053177202 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))]
```

```r
# 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] 25 25 19 21 24 15 20 21 18 23 ...
```

```
summary(blocos_dados_6)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Ca
##  1:20   0  :16   COM:40   1:40   Min.   :15.00
##  2:20   25 :16   SEM:40   2:40   1st Qu.:21.00
##  3:20   50 :16                   Median :24.00
##  4:20   75 :16                   Mean   :25.08
##         100:16                   3rd Qu.:28.25
##                                  Max.   :42.00
```

```
# 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   183.3333 245.6667
## 25  195.0000 226.0000
## 50  195.6667 200.3333
## 75  192.0000 211.0000
## 100 174.6667 182.6667
```

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

```
##          COM      SEM
## 0   22.91667 30.70833
## 25  24.37500 28.25000
## 50  24.45833 25.04167
## 75  24.00000 26.37500
## 100 21.83333 22.83333
```

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

```
##           COM      SEM
## 0    4.150794 56.29960
## 25  52.839286 23.35714
## 50  18.061508 27.25198
## 75   4.571429 34.83929
## 100 22.507937  6.47619
```

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

```
##          COM      SEM
## 0   2.037350 7.503306
## 25  7.269064 4.832923
## 50  4.249883 5.220343
## 75  2.138090 5.902481
## 100 4.744253 2.544836
```

```
# 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
##
## 25.07917
##
##  CICLO
## CICLO
##      1      2
## 22.675 27.483
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 24.850 24.750 23.450 27.267
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 26.813 26.313 24.750 25.188 22.333
##
##  INOCULO
## INOCULO
##    COM    SEM
## 23.517 26.642
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 24.125 22.875 23.000 23.375 20.000
##     2 29.500 29.750 26.500 27.000 24.667
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 21.283 24.067
##     2 25.750 29.217
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   22.917 30.708
##       25  24.375 28.250
##       50  24.458 25.042
##       75  24.000 26.375
##       100 21.833 22.833
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 22.50 20.00 22.25 24.00 17.67
##     2 23.33 28.75 26.67 24.00 26.00
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0     25    50    75    100
##     1 25.75 25.75 23.75 22.75 22.33
##     2 35.67 30.75 26.33 30.00 23.33
```

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

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

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

```
## [1] 0.2860949
```

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

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

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

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

```
## [1] 0.5416224
```

```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    152   50.67
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  195.3   195.3
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  43.82   14.61
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  462.4   462.4  29.133 1.54e-06 ***
## EFLUENTE                  4  195.0    48.7   3.071   0.0237 *
## CICLO:INOCULO             1    2.3     2.3   0.147   0.7028
## CICLO:EFLUENTE            4   30.9     7.7   0.487   0.7454
## INOCULO:EFLUENTE          4  135.5    33.9   2.134   0.0890 .
## CICLO:INOCULO:EFLUENTE    4  203.9    51.0   3.212   0.0194 *
## Residuals                54  857.1    15.9
## ---
## 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                     3 152.00   50.67  3.2057 0.0297826 *
## CICLO                     1 462.40  462.40 29.2558 1.306e-06 ***
## INOCULO                   1 195.31  195.31 12.3573 0.0008687 ***
## EFLUENTE                  4 194.96   48.74  3.0838 0.0228277 *
## CICLO:INOCULO:EFLUENTE    4 203.94   50.98  3.2257 0.0186613 *
## 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: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(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 EFLUENTE       Ca
## 1      1     COM        0 22.50000
## 2      2     COM        0 23.33333
## 3      1     SEM        0 25.75000
## 4      2     SEM        0 35.66667
## 5      1     COM       25 20.00000
## 6      2     COM       25 28.75000
## 7      1     SEM       25 25.75000
## 8      2     SEM       25 30.75000
## 9      1     COM       50 22.25000
## 10     2     COM       50 26.66667
## 11     1     SEM       50 23.75000
## 12     2     SEM       50 26.33333
## 13     1     COM       75 24.00000
## 14     2     COM       75 24.00000
## 15     1     SEM       75 22.75000
## 16     2     SEM       75 30.00000
## 17     1     COM      100 17.66667
## 18     2     COM      100 26.00000
## 19     1     SEM      100 22.33333
## 20     2     SEM      100 23.33333
```

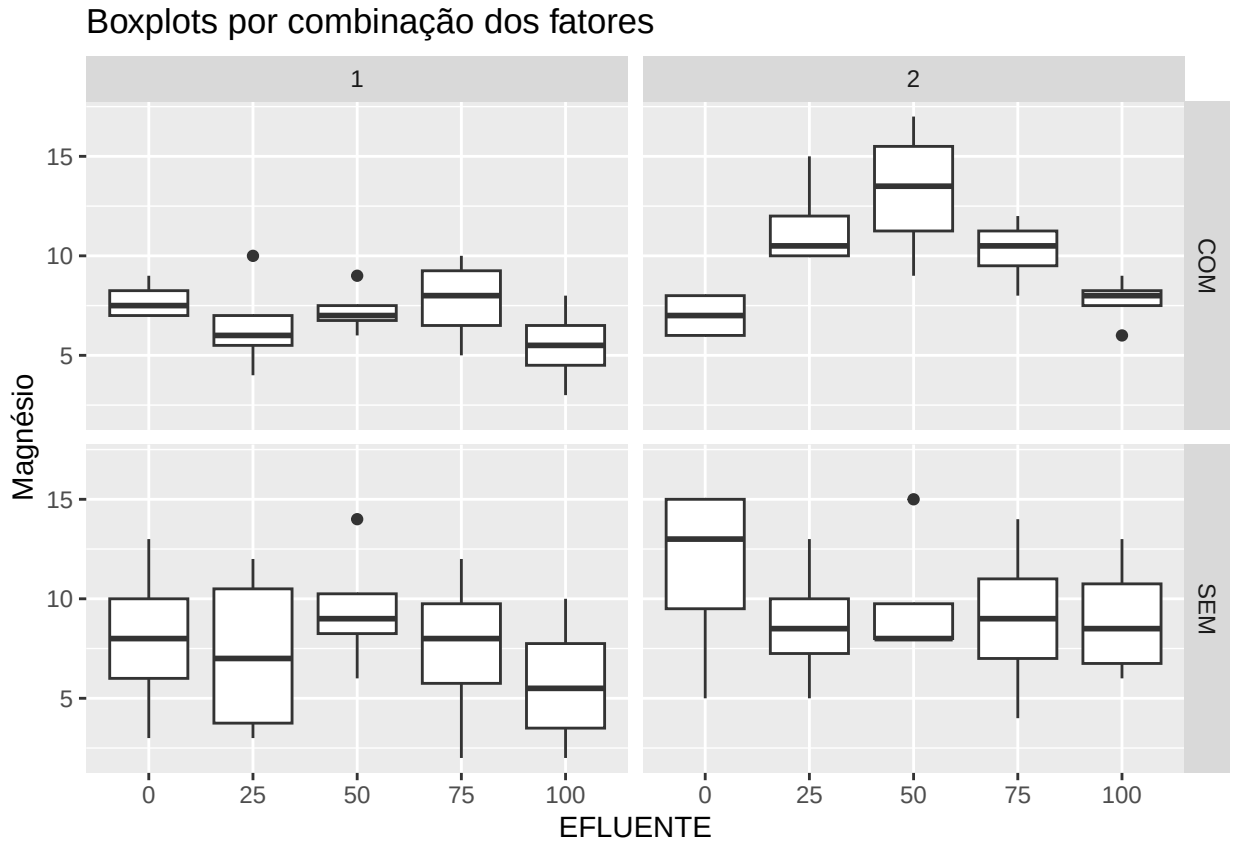
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
```

```
  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:0-1:COM:0     13.16667   2.7239447 23.6093886 2.623636e-03
## 2:SEM:0-2:COM:0     12.33333   1.8906114 22.7760553 6.867522e-03
## 1:COM:25-2:SEM:0   -15.66667 -26.1093886 -5.2239447 1.183443e-04
## 1:COM:50-2:SEM:0   -13.41667 -23.8593886 -2.9739447 1.949769e-03
## 1:SEM:50-2:SEM:0   -11.91667 -22.3593886 -1.4739447 1.091576e-02
## 1:COM:75-2:SEM:0   -11.66667 -22.1093886 -1.2239447 1.432324e-02
## 2:COM:75-2:SEM:0   -11.66667 -22.1093886 -1.2239447 1.432324e-02
## 1:SEM:75-2:SEM:0   -12.91667 -23.3593886 -2.4739447 3.517721e-03
## 1:COM:100-2:SEM:0  -18.00000 -28.4427219 -7.5572781 5.496503e-06
## 1:SEM:100-2:SEM:0  -13.33333 -23.7760553 -2.8906114 2.153399e-03
## 2:SEM:100-2:SEM:0  -12.33333 -22.7760553 -1.8906114 6.867522e-03
## 2:SEM:25-1:COM:25   10.75000   0.3072781 21.1927219 3.703263e-02
## 1:COM:100-2:COM:25 -11.08333 -21.5260553 -0.6406114 2.645194e-02
## 1:COM:100-2:SEM:25 -13.08333 -23.5260553 -2.6406114 2.894240e-03
## 1:COM:100-2:SEM:75 -12.33333 -22.7760553 -1.8906114 6.867522e-03
```

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

# 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
```

```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_7 <- with(dados_7,
                       dados_7[order(CICLO, INOCULO, EFLUENTE), ])

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

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

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

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

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

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

# Redefinir os índices das linhas
rownames(media_blocos_7) <- NULL

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

```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] 7 8 7 9 5.33 ...
```

```
summary(blocos_dados_7)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Mg
##  1:20   0  :16   COM:40   1:40   Min.   : 2.000
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 6.000
##  3:20   50 :16                   Median : 8.000
##  4:20   75 :16                   Mean   : 7.942
##         100:16                   3rd Qu.: 9.625
##                                  Max.   :15.000
```

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

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

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

```
##          COM      SEM
## 0   59.00000 76.00000
## 25  67.33333 64.00000
## 50  74.66667 64.00000
## 75  69.00000 59.33333
## 100 53.00000 49.00000
```

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

```
##          COM      SEM
## 0   7.375000 9.500000
## 25  8.416667 8.000000
## 50  9.333333 8.000000
## 75  8.625000 7.416667
## 100 6.625000 6.125000
```

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

```
##           COM        SEM
## 0    1.125000 10.0000000
## 25  13.674603 13.7142857
## 50  10.793651  0.8571429
## 75   3.625000 10.2460317
## 100  3.982143  5.4821429
```

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

```
##          COM       SEM
## 0   1.060660 3.1622777
## 25  3.697919 3.7032804
## 50  3.285369 0.9258201
## 75  1.903943 3.2009423
## 100 1.995531 2.3413976
```

```
# 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
##
## 7.941667
##
##  CICLO
## CICLO
##     1     2
## 6.950 8.933
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 7.992 7.750 6.600 9.425
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 8.438 8.208 8.667 8.021 6.375
##
##  INOCULO
## INOCULO
##   COM   SEM
## 8.075 7.808
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  7.875  6.292  7.333  7.625  5.625
##     2  9.000 10.125 10.000  8.417  7.125
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 6.600 7.300
##     2 9.550 8.317
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM   SEM
##       0   7.375 9.500
##       25  8.417 8.000
##       50  9.333 8.000
##       75  8.625 7.417
##       100 6.625 6.125
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1  7.750  5.333  6.667  7.750  5.500
##     2  7.000 11.500 12.000  9.500  7.750
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1  8.000  7.250  8.000  7.500  5.750
##     2 11.000  8.750  8.000  7.333  6.500
```

```
# 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] 0.05633719
```

```
# 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_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] 0.4302818
```

```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  80.79   26.93
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  1.422   1.422
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  31.45   10.48
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  78.67   78.67  18.612 6.86e-05 ***
## EFLUENTE                  4  52.85   13.21   3.126   0.0220 *
## CICLO:INOCULO             1  18.69   18.69   4.421   0.0402 *
## CICLO:EFLUENTE            4  25.12    6.28   1.486   0.2194
## INOCULO:EFLUENTE          4  31.29    7.82   1.850   0.1326
## CICLO:INOCULO:EFLUENTE    4  51.52   12.88   3.047   0.0245 *
## Residuals                54 228.26    4.23
## ---
## 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(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, ]
```

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

```
## Analysis of Variance Table
##
## Response: Mg
##                          Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO                     3 80.792  26.931  5.9106 0.0013889 **
## CICLO                     1 78.672  78.672 17.2667 0.0001102 ***
## EFLUENTE                  4 52.853  13.213  2.9000 0.0296529 *
## CICLO:INOCULO             1 18.689  18.689  4.1018 0.0475301 *
## CICLO:INOCULO:EFLUENTE    4 51.519  12.880  2.8268 0.0329111 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

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

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

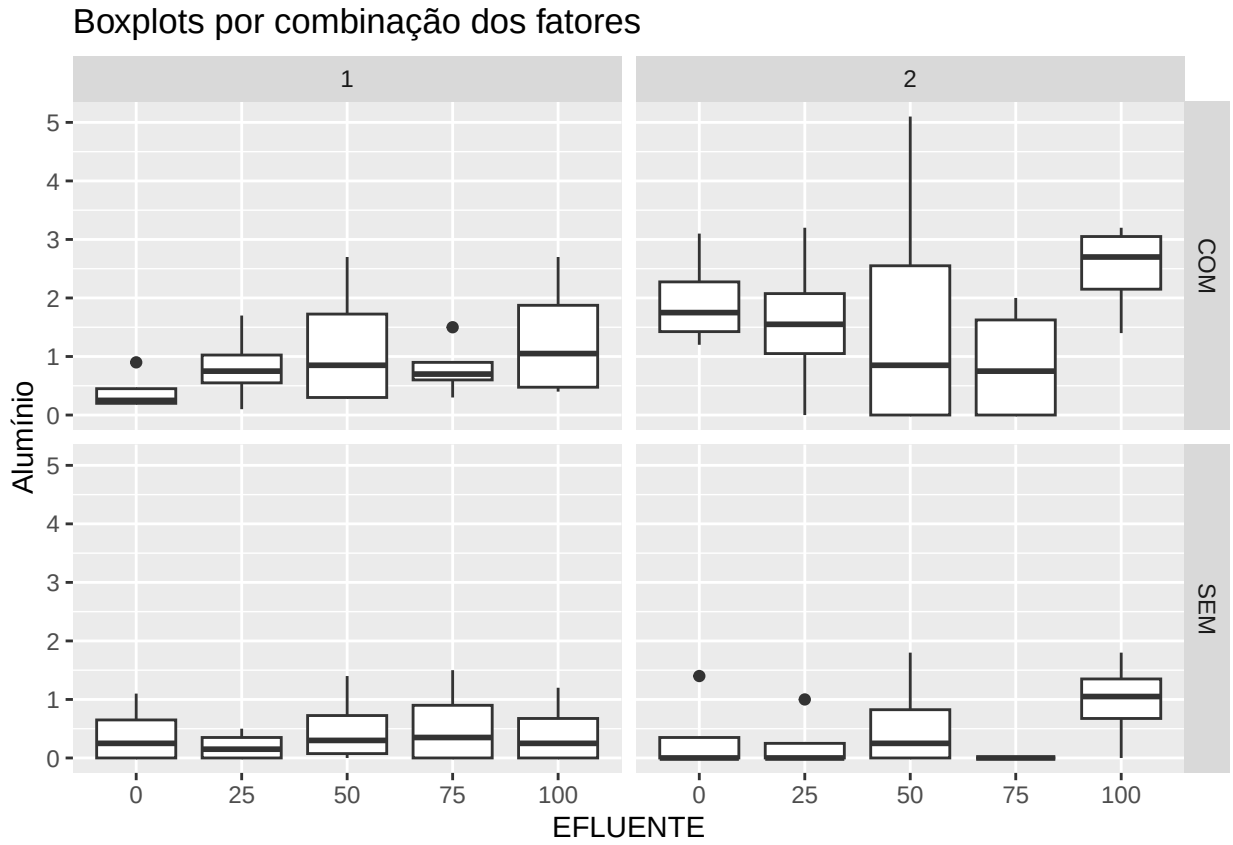
```
##   CICLO INOCULO       Mg
## 1     1     COM 6.600000
## 2     2     COM 9.550000
## 3     1     SEM 7.300000
## 4     2     SEM 8.316667
##    CICLO INOCULO EFLUENTE        Mg
## 1      1     COM        0  7.750000
## 2      2     COM        0  7.000000
## 3      1     SEM        0  8.000000
## 4      2     SEM        0 11.000000
## 5      1     COM       25  5.333333
## 6      2     COM       25 11.500000
## 7      1     SEM       25  7.250000
## 8      2     SEM       25  8.750000
## 9      1     COM       50  6.666667
## 10     2     COM       50 12.000000
## 11     1     SEM       50  8.000000
## 12     2     SEM       50  8.000000
## 13     1     COM       75  7.750000
## 14     2     COM       75  9.500000
## 15     1     SEM       75  7.500000
## 16     2     SEM       75  7.333333
## 17     1     COM      100  5.500000
## 18     2     COM      100  7.750000
## 19     1     SEM      100  5.750000
## 20     2     SEM      100  6.500000
```

```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  2.95  1.163623  4.7363769 0.0003034412
## 1:SEM-2:COM -2.25 -4.036377 -0.4636231 0.0080148010
##                          diif        lwr         upr       p_adj
## 1:COM:25-2:SEM:0   -5.666667 -11.273470 -0.05986298 0.044890892
## 2:COM:25-1:COM:25   6.166667   0.559863 11.77347036 0.017384897
## 2:COM:50-1:COM:25   6.666667   1.059863 12.27347036 0.006249827
## 1:COM:100-2:COM:25 -6.000000 -11.606804 -0.39319631 0.024068005
## 1:SEM:100-2:COM:25 -5.750000 -11.356804 -0.14319631 0.038551837
## 1:COM:100-2:COM:50 -6.500000 -12.106804 -0.89319631 0.008854123
## 1:SEM:100-2:COM:50 -6.250000 -11.856804 -0.64319631 0.014729223
```

Análise para Alumínio

```r
# 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 0.3 0.233 0.2 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 :0.3500
##  4:20   75 :16                   Mean   :0.6494
##         100:16                   3rd Qu.:1.1833
##                                  Max.   :3.2000
##                                  NA's   :1
```

```
# 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   8.733333 3.0000000
## 25  9.600000 0.9666667
## 50  6.966667 2.5000000
## 75  6.933333 2.2000000
## 100 6.600000        NA
```

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

```
##           COM       SEM
## 0   1.0916667 0.3750000
## 25  1.2000000 0.1208333
## 50  0.8708333 0.3125000
## 75  0.8666667 0.2750000
## 100 0.8250000        NA
```

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

```
##           COM        SEM
## 0   1.1415079 0.32785714
## 25  1.0857143 0.03585317
## 50  0.9391865 0.24125000
## 75  0.4276190 0.30500000
## 100 0.7578571         NA
```

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

```
##           COM       SEM
## 0   1.0684137 0.5725881
## 25  1.0419761 0.1893493
## 50  0.9691164 0.4911721
## 75  0.6539259 0.5522681
## 100 0.8705499        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
##
## 0.6493671
##
##  CICLO
##           1       2
##      0.6175  0.6821
## rep 40.0000 39.0000
##
##  BLOCO
##           1       2       3       4
##      0.5904  0.8478  0.8971  0.2721
## rep 20.0000 19.0000 20.0000 20.0000
##
##  EFLUENTE
##           0      25      50      75     100
##      0.7304  0.6575  0.5887  0.5679  0.7058
## rep 16.0000 16.0000 16.0000 16.0000 15.0000
##
##  INOCULO
##         COM     SEM
##      0.9672  0.3234
## rep 40.0000 39.0000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.312 0.508 0.833 0.554 0.881
##   rep 8.000 8.000 8.000 8.000 8.000
##   2   1.150 0.808 0.345 0.583 0.502
##   rep 8.000 8.000 8.000 8.000 7.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
```

```
##   1    0.817  0.418
##   rep 20.000 20.000
##   2    1.118  0.223
##   rep 20.000 19.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   1.084 0.377
##      rep 8.000 8.000
##      25  1.192 0.123
##      rep 8.000 8.000
##      50  0.863 0.315
##      rep 8.000 8.000
##      75  0.859 0.277
##      rep 8.000 8.000
##      100 0.842 0.550
##      rep 8.000 7.000
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.233  0.825  1.175  0.567  1.300
##   rep  4.000  4.000  4.000  4.000  4.000
##   2    1.945  1.570  0.562  1.162  0.345
##   rep  4.000  4.000  4.000  4.000  4.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##   1    0.400  0.200  0.500  0.550  0.425
##   rep  4.000  4.000  4.000  4.000  4.000
##   2    0.345  0.037  0.120 -0.005  0.760
##   rep  4.000  4.000  4.000  4.000  3.000
```

```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] 0.04951219
```

```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] 0.7546731
```

```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  4.879   1.626
##
## Error: INOCULO
##       Df Sum Sq Mean Sq
## CICLO  1   8.25    8.25
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## CICLO      1 1.2126  1.2126   10.74 0.0818 .
## Residuals  2 0.2258  0.1129
## ---
## 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  0.060  0.0599   0.150 0.7002
## EFLUENTE                4  0.272  0.0680   0.170 0.9527
## CICLO:INOCULO           1  1.227  1.2270   3.070 0.0855 .
## CICLO:EFLUENTE          4  4.756  1.1889   2.975 0.0273 *
## INOCULO:EFLUENTE        4  1.158  0.2894   0.724 0.5793
## CICLO:INOCULO:EFLUENTE  4  5.440  1.3601   3.403 0.0150 *
## Residuals              53 21.183  0.3997
## ---
## 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)
## BLOCO                    3 4.8788  1.6263  4.0577   0.01114 *
## INOCULO                  1 8.2273  8.2273 20.5282 3.129e-05 ***
## CICLO:EFLUENTE           4 4.6327  1.1582  2.8898   0.03026 *
## CICLO:INOCULO:EFLUENTE   4 5.6811  1.4203  3.5437   0.01201 *
## 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:EFLUENTE"         "CICLO:INOCULO:EFLUENTE"
```

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

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

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(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 EFLUENTE        Al
## 1      1        0 0.3166667
## 2      2        0 1.1500000
## 3      1       25 0.5125000
## 4      2       25 0.8083333
## 5      1       50 0.8375000
## 6      2       50 0.3458333
## 7      1       75 0.5583333
```

```
## 8      2       75 0.5833333
## 9      1      100 0.8625000
## 10     2      100 0.5000000
##    CICLO INOCULO EFLUENTE         Al
## 1      1     COM        0 0.23333333
## 2      2     COM        0 1.95000000
## 3      1     SEM        0 0.40000000
## 4      2     SEM        0 0.35000000
## 5      1     COM       25 0.82500000
## 6      2     COM       25 1.57500000
## 7      1     SEM       25 0.20000000
## 8      2     SEM       25 0.04166667
## 9      1     COM       50 1.17500000
## 10     2     COM       50 0.56666667
## 11     1     SEM       50 0.50000000
## 12     2     SEM       50 0.12500000
## 13     1     COM       75 0.56666667
## 14     2     COM       75 1.16666667
## 15     1     SEM       75 0.55000000
## 16     2     SEM       75 0.00000000
## 17     1     COM      100 1.30000000
## 18     2     COM      100 0.35000000
## 19     1     SEM      100 0.42500000
## 20     2     SEM      100 0.70000000
```

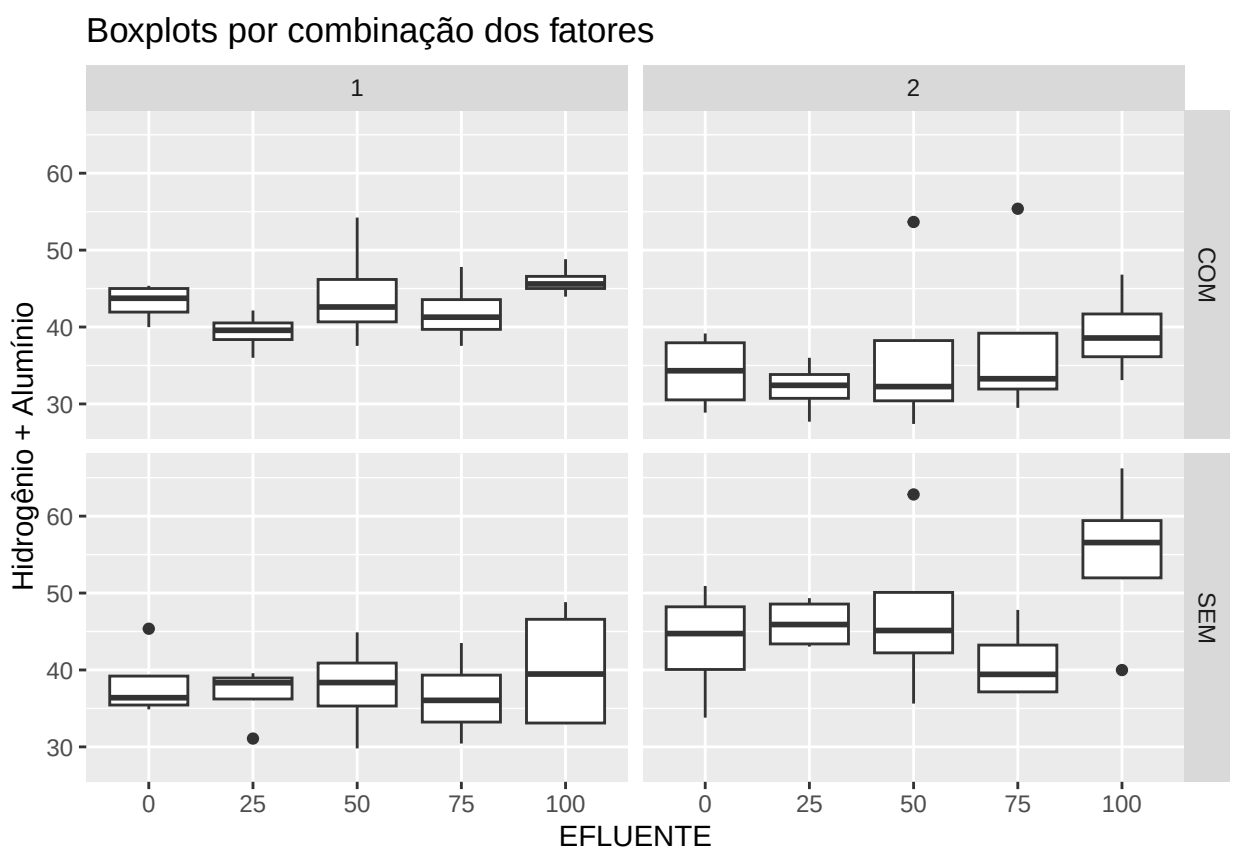
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

```
## [1] diif  lwr   upr   p_adj
## <0 linhas> (ou row.names de comprimento 0)
##                         diif         lwr         upr      p_adj
## 2:COM:0-1:COM:0   1.716667  0.05239433  3.38093901 0.03627985
## 1:SEM:25-2:COM:0 -1.750000 -3.41427234 -0.08572766 0.02941575
```

```
## 2:SEM:25-2:COM:0 -1.908333 -3.57260567 -0.24406099 0.01029609
## 2:SEM:50-2:COM:0 -1.825000 -3.48927234 -0.16072766 0.01807854
## 2:SEM:75-2:COM:0 -1.950000 -3.61427234 -0.28572766 0.00770901
```

Análise para Hidrogênio + Alumínio

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




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

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

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

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

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

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

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

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

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

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

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

```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
```

```r
rownames(media_blocos_9) <- NULL

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

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

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ H_AL    : num [1:80] 42.6 45.4 40 44.9 36 ...
```

```r
summary(blocos_dados_9)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       H_AL
##  1:20   0  :16   COM:40   1:40   Min.   :27.40
##  2:20   25 :16   SEM:40   2:40   1st Qu.:34.70
##  3:20   50 :16                   Median :39.78
##  4:20   75 :16                   Mean   :39.46
##         100:16                   3rd Qu.:44.07
##                                  Max.   :54.23
```

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

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

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

```
##          COM      SEM
## 0   309.4900 317.7267
## 25  285.8300 339.2033
## 50  299.5367 331.9300
## 75  295.9733 314.9233
## 100 341.0200 320.8200
```

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

```
##          COM      SEM
## 0   38.68625 39.71583
```

```
## 25  35.72875 42.40042
## 50  37.44208 41.49125
## 75  36.99667 39.36542
## 100 42.62750 40.10250
```

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

```
##          COM      SEM
## 0   36.52768 40.77890
## 25  22.67904 19.89035
## 50  77.02230 32.10638
## 75  37.83979 30.84978
## 100 29.01554 29.57634
```

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

```
##          COM      SEM
## 0   6.043814 6.385836
## 25  4.762252 4.459860
## 50  8.776235 5.666250
## 75  6.151406 5.554258
## 100 5.386607 5.438413
```

```
# 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
##
## 39.45567
##
##  CICLO
## CICLO
##     1     2
## 40.40 38.51
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 39.79 40.27 40.40 37.36
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 39.20 39.06 39.47 38.18 41.37
##
##  INOCULO
```

```
## INOCULO
##   COM   SEM
## 38.30 40.62
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 39.55 39.04 41.05 39.24 43.10
##     2 38.86 39.09 37.88 37.12 39.63
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 42.95 37.84
##     2 33.64 43.39
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   38.69 39.72
##      25  35.73 42.40
##      50  37.44 41.49
##      75  37.00 39.37
##      100 42.63 40.10
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 43.21 39.33 44.25 41.98 45.99
##     2 34.17 32.13 30.64 32.01 39.26
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 35.89 38.75 37.85 36.51 40.22
##     2 43.55 46.05 45.13 42.22 39.99
```

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

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

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

```
## [1] 0.3667587
```

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

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

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

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

```
## [1] 0.822809
```

```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  120.8   40.28
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  107.5   107.5
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  145.3   48.44
##
## Error: Within
##                       Df Sum Sq Mean Sq F value  Pr(>F)
```

```
## CICLO                   1   70.9    70.9   4.223  0.0447 *
## EFLUENTE                4   87.8    22.0   1.308  0.2786
## CICLO:INOCULO           1 1103.0  1103.0  65.742 6.6e-11 ***
## CICLO:EFLUENTE          4   37.6     9.4   0.561  0.6922
## INOCULO:EFLUENTE        4  188.3    47.1   2.805  0.0345 *
## CICLO:INOCULO:EFLUENTE  4  110.4    27.6   1.644  0.1765
## Residuals              54  906.0    16.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modelo = modelo_PARCELASUB
```

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

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

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

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

```
## Analysis of Variance Table
##
## Response: H_AL
##                   Df  Sum Sq Mean Sq F value    Pr(>F)
## INOCULO            1  107.54  107.54  5.8305   0.01898 *
## CICLO:INOCULO      1 1103.01 1103.01 59.8022 1.909e-10 ***
## INOCULO:EFLUENTE   4  188.27   47.07  2.5519   0.04871 *
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

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

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

```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 42.95050
## 2     2     COM 33.64200
## 3     1     SEM 37.84300
## 4     2     SEM 43.38717
##    INOCULO EFLUENTE     H_AL
## 1      COM        0 38.68625
## 2      SEM        0 39.71583
```

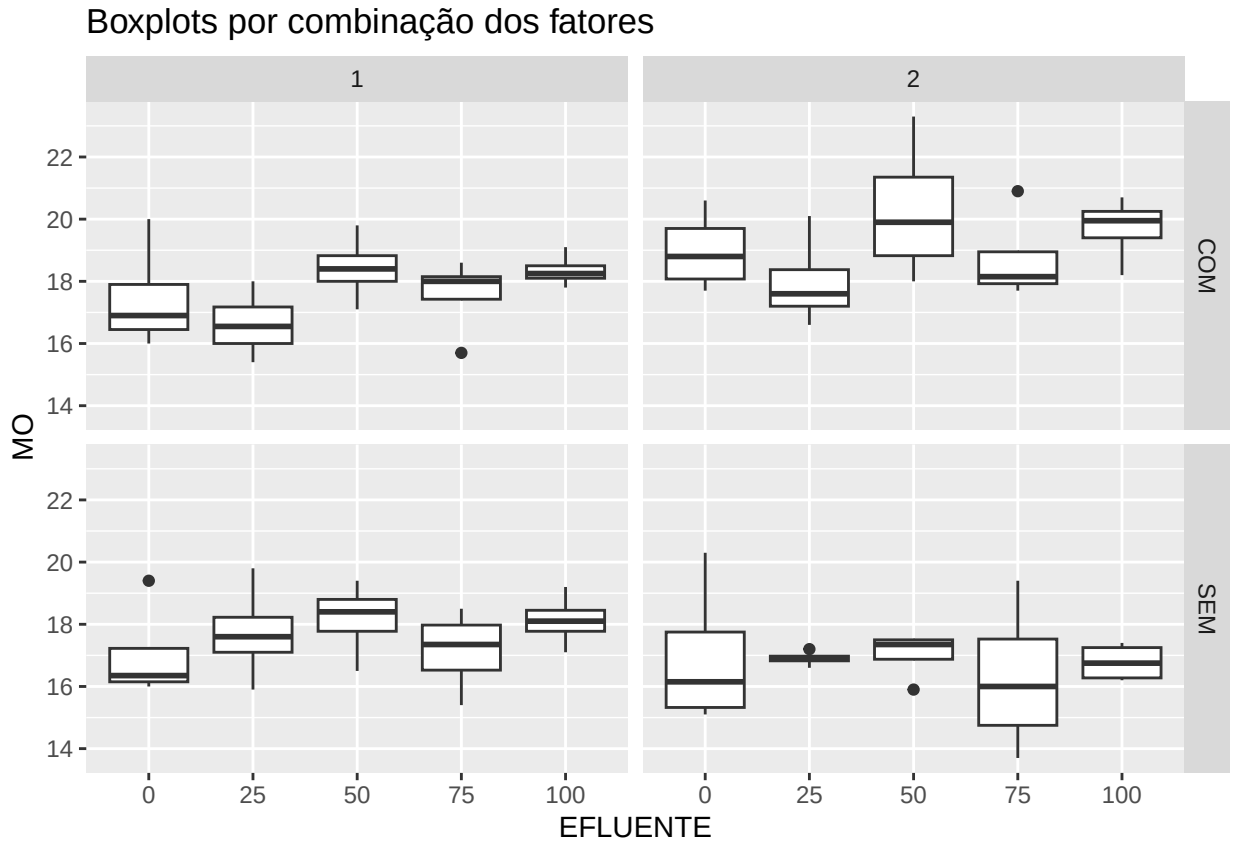
```
## 3      COM       25 35.72875
## 4      SEM       25 42.40042
## 5      COM       50 37.44208
## 6      SEM       50 41.49125
## 7      COM       75 36.99667
## 8      SEM       75 39.36542
## 9      COM      100 42.62750
## 10     SEM      100 40.10250
```

```
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 -9.308500 -12.9026674 -5.714333 3.316224e-08
## 1:SEM-1:COM -5.107500  -8.7016674 -1.513333 2.213180e-03
## 1:SEM-2:COM  4.201000   0.6068326  7.795167 1.575836e-02
## 2:SEM-2:COM  9.745167   6.1509992 13.339334 9.682292e-09
## 2:SEM-1:SEM  5.544167   1.9499992  9.138334 7.914183e-04
## [1] diif  lwr   upr   p_adj
## <0 linhas> (ou row.names de comprimento 0)
```

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

# 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
```

```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_10 <- with(dados_10,
                        dados_10[order(CICLO, INOCULO, EFLUENTE), ])

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

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

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

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

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

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

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

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

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

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ MO      : num [1:80] 20 16.6 17.2 16 16.9 15.4 16.2 18 18.5 17.1 ...
```

```
summary(blocos_dados_10)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        MO
##  1:20   0  :16   COM:40   1:40   Min.   :15.10
##  2:20   25 :16   SEM:40   2:40   1st Qu.:16.90
##  3:20   50 :16                   Median :17.75
##  4:20   75 :16                   Mean   :17.92
##         100:16                   3rd Qu.:18.73
##                                  Max.   :23.30
```

```
# 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
```

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

```
##          COM      SEM
## 0   145.7000 132.5333
## 25  138.4000 138.5000
## 50  154.2333 140.8000
## 75  151.2000 137.1333
## 100 155.0000 140.2333
```

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

```
##          COM      SEM
## 0   18.21250 16.56667
## 25  17.30000 17.31250
## 50  19.27917 17.60000
## 75  18.90000 17.14167
## 100 19.37500 17.52917
```

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

```
##          COM       SEM
## 0   2.726964 0.1450794
## 25  2.008571 1.3183929
## 50  3.592996 1.2685714
## 75  1.077143 2.0996032
## 100 1.353571 0.8420437
```

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

```
##          COM       SEM
## 0   1.651352 0.3808929
## 25  1.417241 1.1482129
## 50  1.895520 1.1263088
## 75  1.037855 1.4490008
## 100 1.163431 0.9176294
```

```r
# 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
##
## 17.92167
##
##  CICLO
## CICLO
##      1      2
## 17.646 18.197
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 18.377 17.890 17.660 17.760
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 17.390 17.306 18.440 18.021 18.452
##
##  INOCULO
## INOCULO
##    COM    SEM
## 18.613 17.230
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 16.842 17.175 18.300 17.675 18.237
##     2 17.938 17.438 18.579 18.367 18.667
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 17.810 17.482
##     2 19.417 16.978
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM    SEM
##       0   18.212 16.567
##       25  17.300 17.312
##       50  19.279 17.600
##       75  18.900 17.142
##       100 19.375 17.529
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 17.450 16.625 18.425 18.200 18.350
##     2 18.975 17.975 20.133 19.600 20.400
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0      25     50     75     100
##     1 16.233 17.725 18.175 17.150 18.125
##     2 16.900 16.900 17.025 17.133 16.933
```

```
# 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] 0.1531962
```

```
# 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_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] 0.8059487
```

```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  6.053   2.018
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  38.27   38.27
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  15.35   5.116
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1   6.09   6.087   5.530  0.02237 *
## EFLUENTE                  4  19.54   4.885   4.438  0.00357 **
## CICLO:INOCULO             1  22.26  22.261  20.224 3.69e-05 ***
## CICLO:EFLUENTE            4   1.95   0.489   0.444  0.77632
## INOCULO:EFLUENTE          4   9.84   2.459   2.234  0.07734 .
## CICLO:INOCULO:EFLUENTE    4   3.89   0.973   0.884  0.47974
## Residuals                54  59.44   1.101
## ---
## 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(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, ]
```

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

```
## Analysis of Variance Table
##
## Response: MO
##               Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO          1  6.087   6.087  4.6393 0.0354902 *
## INOCULO        1 38.272  38.272 29.1709 1.345e-06 ***
## EFLUENTE       4 19.540   4.885  3.7234 0.0092446 **
## CICLO:INOCULO  1 22.261  22.261 16.9668 0.0001244 ***
## 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(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)
  }
}
```

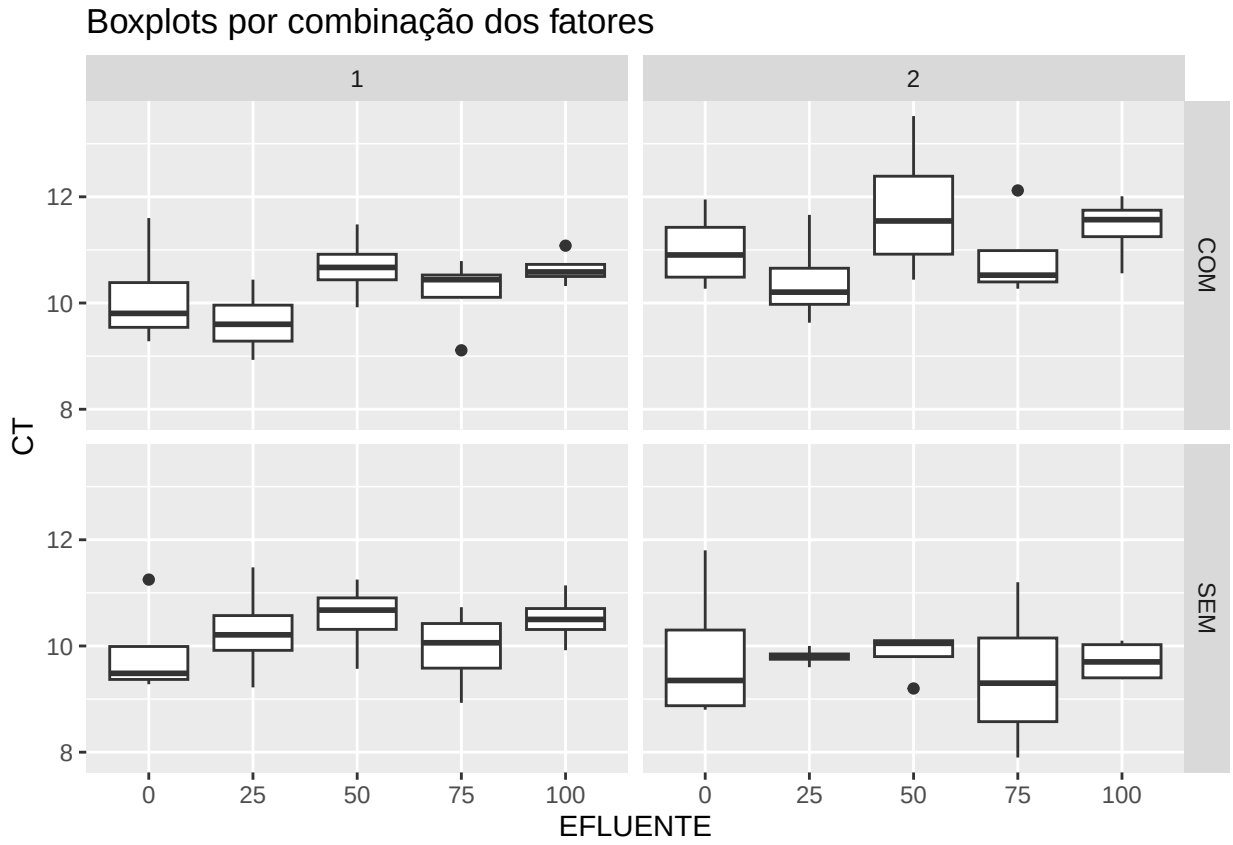
```
##   CICLO INOCULO       MO
## 1     1     COM 17.81000
## 2     2     COM 19.41667
## 3     1     SEM 17.48167
## 4     2     SEM 16.97833
```

```
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  1.606667  0.6480731  2.5652602 2.430914e-04
## 1:SEM-2:COM -1.935000 -2.8935935 -0.9764065 9.797518e-06
## 2:SEM-2:COM -2.438333 -3.3969269 -1.4797398 5.294404e-08
```

Análise para CT

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




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

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

```r
# 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

# 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))]
```

```
# 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] 11.6 9.63 9.98 9.28 9.8 ...
```

```
summary(blocos_dados_11)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        CT
##  1:20   0  :16   COM:40   1:40   Min.   : 8.80
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 9.80
##  3:20   50 :16                   Median :10.29
##  4:20   75 :16                   Mean   :10.39
##         100:16                   3rd Qu.:10.79
##                                  Max.   :13.52
```

```
# 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   84.52000 76.86667
## 25  80.27000 80.32000
## 50  89.46000 81.57000
## 75  87.68667 79.51333
## 100 89.32667 81.39333
```

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

```
##          COM       SEM
## 0   10.56500  9.608333
```

```
## 25  10.03375 10.040000
## 50  11.18250 10.196250
## 75  10.96083  9.939167
## 100 11.16583 10.174167
```

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

```
##           COM        SEM
## 0   0.9162000 0.04805079
## 25  0.6760554 0.44591429
## 50  1.2125929 0.43385536
## 75  0.3612175 0.67357698
## 100 0.5273698 0.28118651
```

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

```
##           COM       SEM
## 0   0.9571834 0.2192049
## 25  0.8222259 0.6677681
## 50  1.1011779 0.6586770
## 75  0.6010137 0.8207174
## 100 0.7262023 0.5302702
```

```
# 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
##
## 10.38658
##
##  CICLO
## CICLO
##      1      2
## 10.220 10.553
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 10.655 10.371 10.246 10.274
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 10.087 10.037 10.689 10.450 10.670
##
##  INOCULO
```

```
## INOCULO
##    COM    SEM
## 10.782  9.992
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  9.770  9.961 10.614 10.251 10.506
##     2 10.404 10.113 10.765 10.649 10.834
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1 10.301 10.140
##     2 11.262  9.843
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   10.565  9.608
##      25  10.034 10.040
##      50  11.182 10.196
##      75  10.961  9.939
##      100 11.166 10.174
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1 10.122  9.642 10.685 10.557 10.497
##     2 11.008 10.425 11.680 11.365 11.835
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  9.417 10.280 10.542  9.945 10.515
##     2  9.800  9.800  9.850  9.933  9.833
```

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

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

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

```
## [1] 0.1201501
```

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

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

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

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

```
## [1] 0.8012291
```

```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  2.099  0.6997
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  12.48   12.48
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  4.985   1.662
##
## Error: Within
##                       Df Sum Sq Mean Sq F value   Pr(>F)
```

```
## CICLO                   1  2.213   2.213   6.084  0.01684 *
## EFLUENTE                4  6.212   1.553   4.269  0.00449 **
## CICLO:INOCULO           1  7.917   7.917  21.763 2.07e-05 ***
## CICLO:EFLUENTE          4  0.644   0.161   0.443  0.77717
## INOCULO:EFLUENTE        4  3.179   0.795   2.184  0.08299 .
## CICLO:INOCULO:EFLUENTE  4  1.529   0.382   1.051  0.38984
## Residuals              54 19.645   0.364
## ---
## 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(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, ]

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

```
## Analysis of Variance Table
##
## Response: CT
##                Df  Sum Sq Mean Sq F value    Pr(>F)
## CICLO           1  2.2133  2.2133  5.1223   0.02745 *
## INOCULO         1 12.4820 12.4820 28.8870 1.482e-06 ***
## EFLUENTE        4  6.2124  1.5531  3.5943   0.01108 *
## CICLO:INOCULO   1  7.9170  7.9170 18.3223 7.222e-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"
```

```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 10.300667
## 2     2     COM 11.262500
## 3     1     SEM 10.139833
## 4     2     SEM  9.843333
```

```
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  0.9618333  0.4117118  1.5119549 1.258402e-04
## 1:SEM-2:COM -1.1226667 -1.6727882 -0.5725451 7.902559e-06
## 2:SEM-2:COM -1.4191667 -1.9692882 -0.8690451 3.675382e-08
```

```
# 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 20-40 atualizado.xlsx",
      sheetName = "R - Solo", append = FALSE)
```