Foliar

Ana Carolina Murad Lima

2023-06-22

```
# 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)
```

CICLO 1

```
# Leitura e tratamento dos dados
dados <- read_excel("Foliar 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 5:7) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
dados[5:7] =  round(dados[5:7], digits = 2)
str(dados)
```

```
## tibble [70 x 11] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ EFLUENTE: num [1:70] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO : chr [1:70] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
##  $ N       : num [1:70] 49.7 53.7 55.4 47.7 45.7 ...
##  $ P       : num [1:70] 2.3 2.29 2.3 2.27 2.45 2.47 2.13 2.44 2.26 2.18 ...
##  $ K       : num [1:70] 16.4 16.1 14.4 14.5 16.6 ...
##  $ Na      : num [1:70] 0 0.5 0.9 0.1 1 ...
##  $ Ca      : num [1:70] 8.97 11.68 8.98 9.51 8.52 ...
##  $ Mg      : num [1:70] 6.7 6.84 7.26 7.31 6.84 ...
##  $ S       : num [1:70] 1.09 1.06 1.07 1.52 1.03 ...
```

```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
# 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
```

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

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

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

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

```r
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
# Usar apenas dados do Ciclo 1
dados = subset(dados, CICLO == 1)
```

```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)]
```

```r
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)]

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

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

```r
str(dados_1)
```

```
## tibble [40 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 ...
##  $ N       : num [1:40] 49.7 53.7 55.4 47.7 45.7 ...
```

```r
"dados_2"
```

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

```r
str(dados_2)
```

```
## tibble [40 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       : num [1:40] 2.3 2.29 2.3 2.27 2.45 2.47 2.13 2.44 2.26 2.18 ...
```

```r
"dados_3"
```

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

```r
str(dados_3)
```

```
## tibble [40 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       : num [1:40] 16.4 16.1 14.4 14.5 16.6 ...
```

```r
"dados_4"
```

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

```
str(dados_4)
```

```
## tibble [40 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 [1:40] 0 0.5 0.9 0.1 1 ...
```

```
"dados_5"
```

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

```
str(dados_5)
```

```
## tibble [40 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 [1:40] 8.97 11.68 8.98 9.51 8.52 ...
```

```
"dados_6"
```

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

```
str(dados_5)
```

```
## tibble [40 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 [1:40] 8.97 11.68 8.98 9.51 8.52 ...
```

```
"dados_7"
```

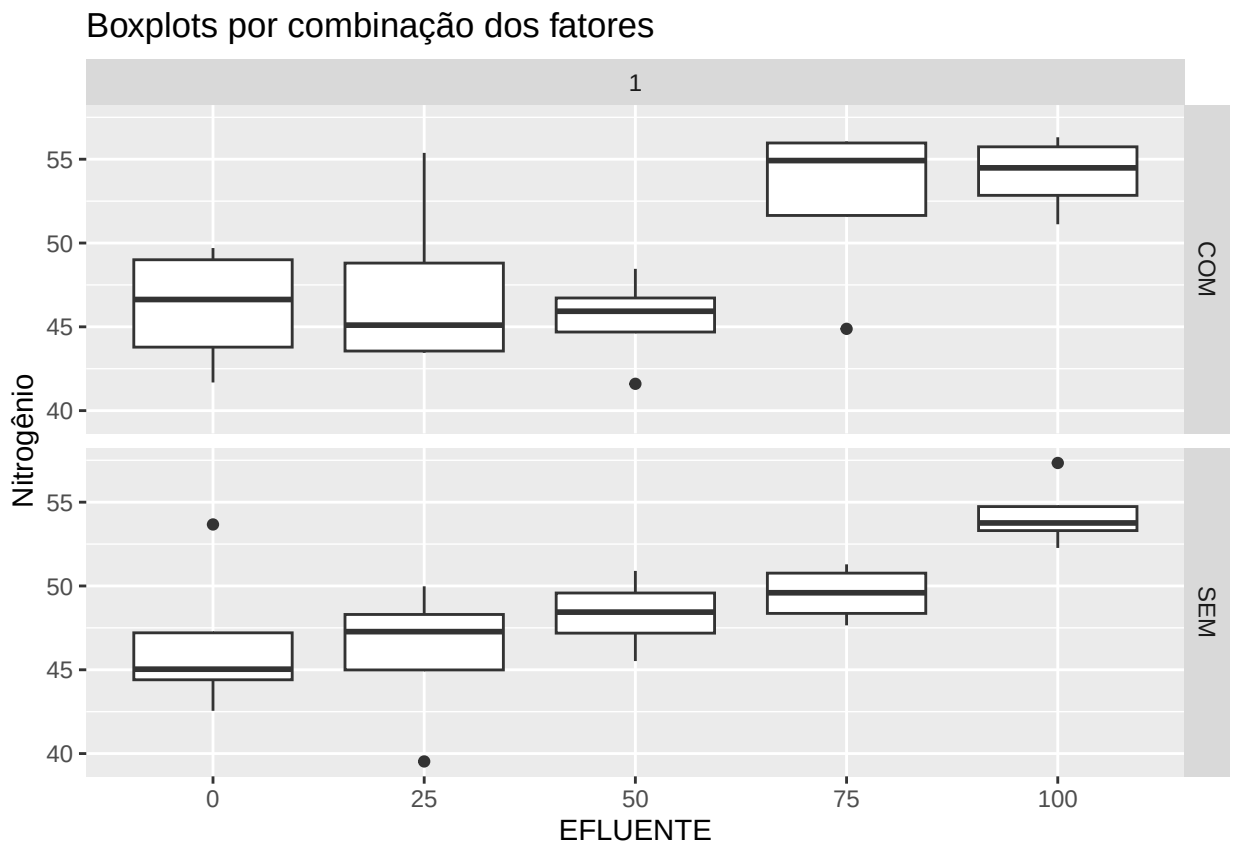
```
## [1] "dados_7"
```

```
str(dados_5)
```

```
## tibble [40 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 [1:40] 8.97 11.68 8.98 9.51 8.52 ...
```

Análise para Nitrogênio

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




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(N ~ 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(N ~ 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
})
```

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

# 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$N[which(blocos_dados_1$N <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_1$N[which(blocos_dados_1$N >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_1 = aggregate(N ~ 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$N[which(is.na(blocos_dados_1$N))] =
  media_blocos_1$N[which(is.na(blocos_dados_1$N))]
```

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

```
## tibble [40 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 ...
##  $ N       : num [1:40] 49.7 44.5 48.8 41.7 55.4 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        N
##  1:10   0  :8    COM:20   1:40   Min.   :41.68
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:46.03
##  3:10   50 :8                    Median :48.69
##  4:10   75 :8                    Mean   :49.31
##         100:8                    3rd Qu.:53.30
##                                  Max.   :56.31
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

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

```
##          COM      SEM
## 0   184.6400 176.8267
## 25  189.0300 192.7067
## 50  187.0933 193.3000
## 75  221.2000 198.1300
## 100 216.4000 213.0533
```

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

```
##          COM      SEM
## 0   46.16000 44.20667
## 25  47.25750 48.17667
## 50  46.77333 48.32500
## 75  55.30000 49.53250
## 100 54.10000 53.26333
```

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

```
##           COM       SEM
## 0   14.067667 1.3724222
## 25  31.447292 1.7701556
## 50   1.451822 5.1683667
## 75   0.982400 2.8734917
## 100  5.443133 0.5014222
```

```
with(blocos_dados_1, tapply(N, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   3.7506888 1.1715043
## 25  5.6077885 1.3304719
## 50  1.2049159 2.2734042
## 75  0.9911609 1.6951377
## 100 2.3330524 0.7081117
```

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

```
## Tables of means
## Grand mean
##
## 49.3095
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 50.81 49.64 48.58 48.21
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 45.18 47.72 47.55 52.42 53.68
##
##  INOCULO
## INOCULO
##   COM   SEM
## 49.92 48.70
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   46.16 44.21
##      25  47.26 48.18
##      50  46.77 48.33
##      75  55.30 49.53
##      100 54.10 53.26
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_1 = aggregate(N ~ 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$N)$p.value

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

```
## [1] 0.3557237
```

```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$N,
                                    media_blocos_1$EFLUENTE,
                                    media_blocos_1$INOCULO,
                                    media_blocos_1$CICLO)$p.value

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

```
## [1] 0.4329772
```

```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(N ~ BLOCO + 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  40.81    13.6
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  14.82   14.82
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3     17   5.667
##
## Error: Within
##                  Df Sum Sq Mean Sq F value   Pr(>F)
```

```
## EFLUENTE          4  411.4  102.86  17.963 6.07e-07 ***
## INOCULO:EFLUENTE  4   67.2   16.81   2.936   0.0415 *
## Residuals        24  137.4    5.73
## ---
## 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(N ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_1)

# 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: N
##                   Df Sum Sq Mean Sq F value    Pr(>F)
## EFLUENTE           4 411.42 102.855 17.9834 2.632e-07 ***
## INOCULO:EFLUENTE   4  67.25  16.811  2.9393   0.03871 *
## 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
print(interacoes_significativas)
```

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

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

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

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

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

}
```

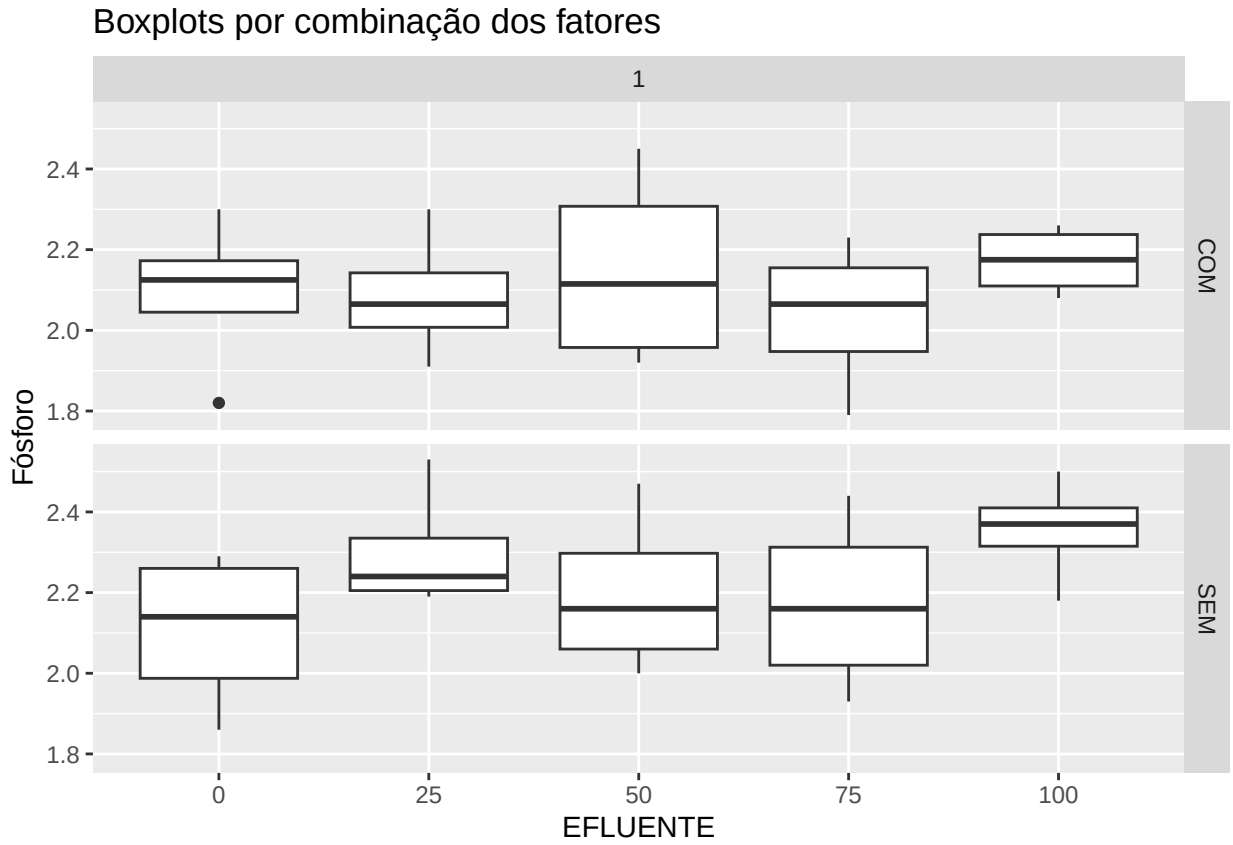
```
##    INOCULO EFLUENTE        N
## 1      COM        0 46.16000
## 2      SEM        0 44.20667
## 3      COM       25 47.25750
## 4      SEM       25 48.17667
## 5      COM       50 46.77333
## 6      SEM       50 48.32500
## 7      COM       75 55.30000
## 8      SEM       75 49.53250
## 9      COM      100 54.10000
## 10     SEM      100 53.26333
```

```
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
## COM:75-COM:0    9.140000 3.3232514 14.95675 3.762426e-04
## COM:100-COM:0   7.940000 2.1232514 13.75675 2.353634e-03
## SEM:100-COM:0   7.103333 1.2865847 12.92008 8.212306e-03
## COM:75-SEM:0   11.093333 5.2765847 16.91008 1.917717e-05
## COM:100-SEM:0   9.893333 4.0765847 15.71008 1.185355e-04
## SEM:100-SEM:0   9.056667 3.2399180 14.87342 4.275756e-04
## COM:75-COM:25   8.042500 2.2257514 13.85925 2.014972e-03
## COM:100-COM:25  6.842500 1.0257514 12.65925 1.201206e-02
## SEM:100-COM:25  6.005833 0.1890847 11.82258 3.883858e-02
## COM:75-SEM:25   7.123333 1.3065847 12.94008 7.974544e-03
## COM:100-SEM:25  5.923333 0.1065847 11.74008 4.339323e-02
## COM:75-COM:50   8.526667 2.7099180 14.34342 9.633853e-04
## COM:100-COM:50  7.326667 1.5099180 13.14342 5.906031e-03
## SEM:100-COM:50  6.490000 0.6732514 12.30675 1.988647e-02
## COM:75-SEM:50   6.975000 1.1582514 12.79175 9.908828e-03
```

Análise para Fósforo

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




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P ~ 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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P ~ 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

# 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_2 <- with(dados_2,
                       dados_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_2$P[which(blocos_dados_2$P <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_2$P[which(blocos_dados_2$P >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_2 = aggregate(P ~ 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$P[which(is.na(blocos_dados_2$P))] =
  media_blocos_2$P[which(is.na(blocos_dados_2$P))]
```

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

```
## tibble [40 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       : num [1:40] 2.3 2.12 2.13 2.18 2.3 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        P
##  1:10   0  :8    COM:20   1:40   Min.   :1.790
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:2.047
##  3:10   50 :8                    Median :2.187
```

```
##  4:10   75 :8                    Mean   :2.176
##         100:8                    3rd Qu.:2.275
##                                  Max.   :2.530
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

```
with(blocos_dados_2, tapply(P, list(EFLUENTE, INOCULO), sum))
```

```
##          COM  SEM
## 0   8.733333 8.43
## 25  8.340000 9.20
## 50  8.600000 8.79
## 75  8.150000 8.69
## 100 8.690000 9.42
```

```
with(blocos_dados_2, tapply(P, list(EFLUENTE, INOCULO), mean))
```

```
##          COM    SEM
## 0   2.183333 2.1075
## 25  2.085000 2.3000
## 50  2.150000 2.1975
## 75  2.037500 2.1725
## 100 2.172500 2.3550
```

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

```
##             COM        SEM
## 0   0.006822222 0.04029167
## 25  0.026300000 0.02466667
## 50  0.062466667 0.04295833
## 75  0.036091667 0.05162500
## 100 0.007425000 0.01743333
```

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

```
##            COM       SEM
## 0   0.08259674 0.2007278
## 25  0.16217275 0.1570563
## 50  0.24993332 0.2072639
## 75  0.18997807 0.2272114
## 100 0.08616844 0.1320353
```

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

```
## Tables of means
## Grand mean
##
## 2.176083
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 2.3090 2.1560 2.1250 2.1143
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 2.1454 2.1925 2.1738 2.1050 2.2638
##
##  INOCULO
## INOCULO
##    COM    SEM
## 2.1257 2.2265
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   2.1833 2.1075
##      25  2.0850 2.3000
##      50  2.1500 2.1975
##      75  2.0375 2.1725
##      100 2.1725 2.3550
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_2 = aggregate(P ~ 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$P)$p.value

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

```
## [1] 0.6067279
```

```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."
```

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

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

```
## [1] 0.7811292
```

```
# 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 ~ BLOCO + 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 0.2449 0.08164
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.1017  0.1017
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.03311 0.01104
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4 0.1116 0.02791   0.999  0.427
## INOCULO:EFLUENTE   4 0.1099 0.02746   0.983  0.435
## Residuals         24 0.6702 0.02792
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(P ~ BLOCO + 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
##       Df  Sum Sq  Mean Sq F value  Pr(>F)
## BLOCO  3 0.24493 0.081643  3.1342 0.04184 *
## 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
print(interacoes_significativas)
```

```
## character(0)
```

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

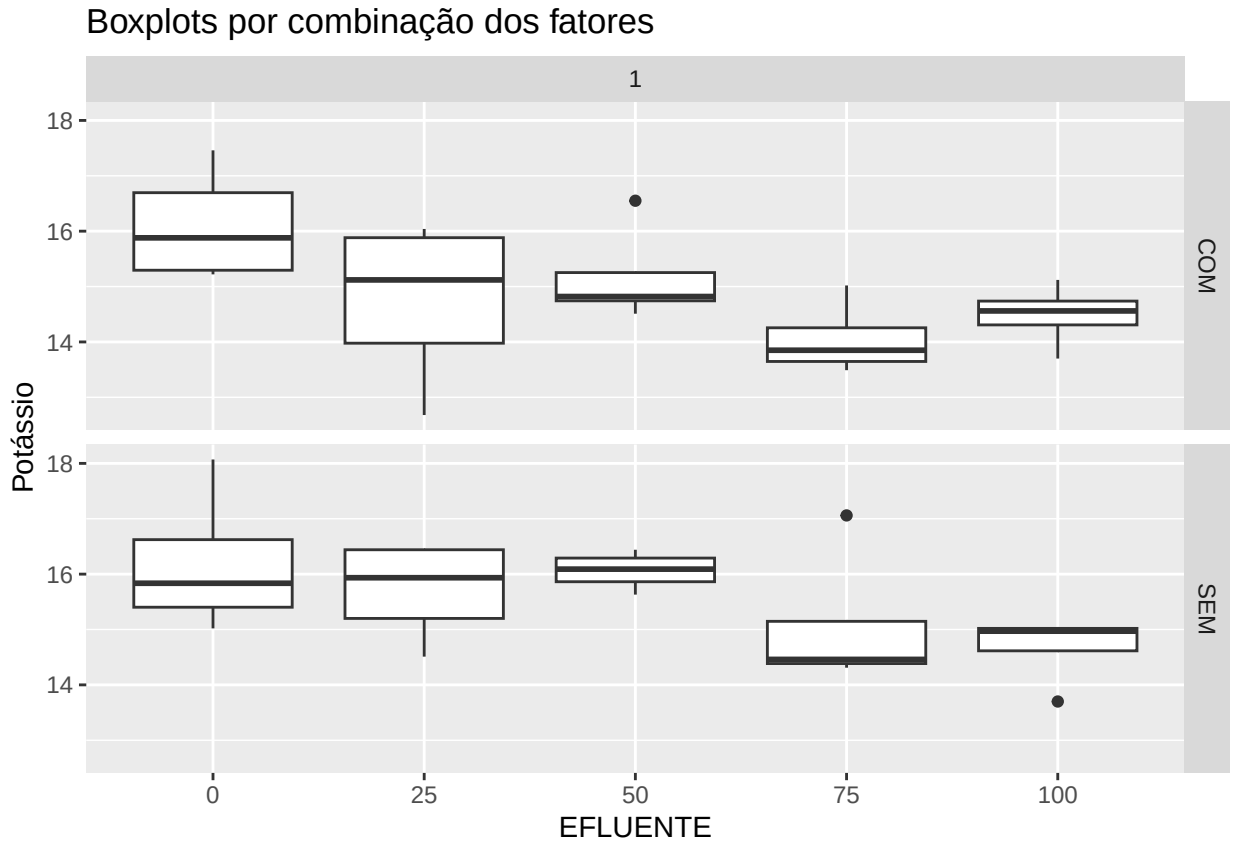
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(P ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_2, 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 Potássio

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K ~ 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$K

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K ~ 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$K

# 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_3 <- with(dados_3,
                           dados_3[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_3$K[which(blocos_dados_3$K <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_3$K[which(blocos_dados_3$K >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_3 = aggregate(K ~ 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$K[which(is.na(blocos_dados_3$K))] =
  media_blocos_3$K[which(is.na(blocos_dados_3$K))]
```

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

```
## tibble [40 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       : num [1:40] 16.4 15.3 17.5 15.2 14.4 ...
```

```r
summary(blocos_dados_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        K
##  1:10   0  :8    COM:20   1:40   Min.   :12.68
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:14.51
##  3:10   50 :8                    Median :15.02
```

```
##  4:10   75 :8                    Mean   :15.15
##         100:8                    3rd Qu.:15.86
##                                  Max.   :18.07
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

```
with(blocos_dados_3, tapply(K, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   64.44000 64.76000
## 25  58.96000 62.82000
## 50  58.86667 64.25000
## 75  56.21000 57.64000
## 100 57.94000 59.94667
```

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

```
##          COM      SEM
## 0   16.11000 16.19000
## 25  14.74000 15.70500
## 50  14.71667 16.06250
## 75  14.05250 14.41000
## 100 14.48500 14.98667
```

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

```
##            COM         SEM
## 0   1.11586667 1.780466667
## 25  2.41020000 0.861366667
## 50  0.02135556 0.125358333
## 75  0.45982500 0.006666667
## 100 0.34523333 0.002222222
```

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

```
##           COM        SEM
## 0   1.0563459 1.33434129
## 25  1.5524819 0.92809841
## 50  0.1461354 0.35405979
## 75  0.6781040 0.08164966
## 100 0.5875656 0.04714045
```

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

```
## Tables of means
## Grand mean
##
## 15.14583
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 14.965 14.876 15.548 15.194
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 16.150 15.222 15.390 14.231 14.736
##
##  INOCULO
## INOCULO
##    COM    SEM
## 14.821 15.471
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   16.110 16.190
##      25  14.740 15.705
##      50  14.717 16.062
##      75  14.052 14.410
##      100 14.485 14.987
```

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

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

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

```
## [1] 0.1646125
```

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

```r
  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$K,
                                    media_blocos_3$EFLUENTE,
                                    media_blocos_3$INOCULO,
                                    media_blocos_3$CICLO)$p.value

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

```
## [1] 0.4039862
```

```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 ~ BLOCO + 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  2.695  0.8982
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  4.225   4.225
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  1.344   0.448
##
## Error: Within
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## EFLUENTE           4 16.626   4.156   5.750 0.00216 **
## INOCULO:EFLUENTE   4  2.032   0.508   0.703 0.59778
## Residuals         24 17.347   0.723
## ---
## 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 ~ BLOCO + 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: K
##           Df Sum Sq Mean Sq F value   Pr(>F)
## INOCULO    1  4.225  4.2250  6.1031 0.020100 *
## EFLUENTE   4 16.626  4.1564  6.0041 0.001368 **
## 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
print(interacoes_significativas)
```

```
## character(0)
```

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

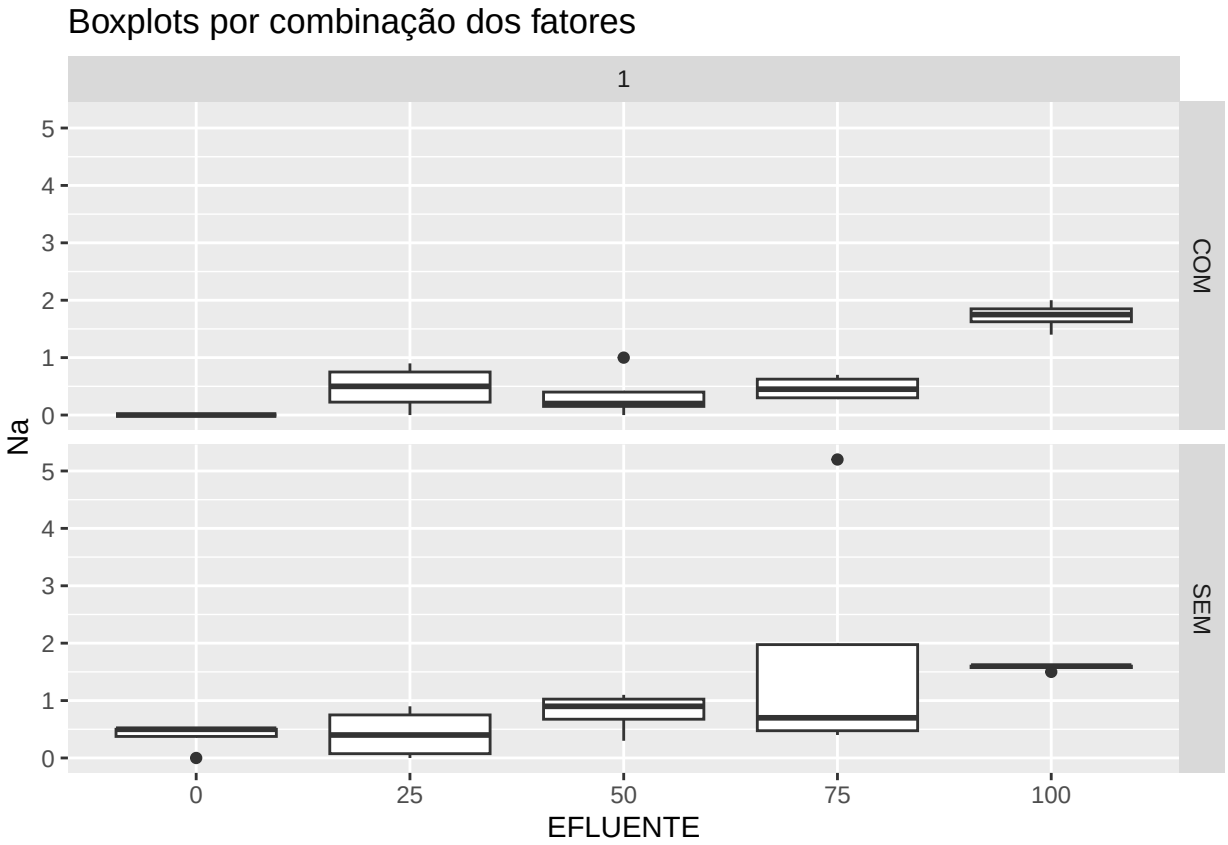
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

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

}
```

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

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

Análise para Sódio

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_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$Na

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ 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$Na

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

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

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

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

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

# Definir como NA (valor ausente) os outliers
blocos_dados_4$Na[which(blocos_dados_4$Na <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_4$Na[which(blocos_dados_4$Na >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_4 = aggregate(Na ~ 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$Na[which(is.na(blocos_dados_4$Na))] =
  media_blocos_4$Na[which(is.na(blocos_dados_4$Na))]
```

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

```
## tibble [40 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 [1:40] 0 0 0 0 0.9 ...
```

```r
summary(blocos_dados_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Na
##  1:10   0  :8    COM:20   1:40   Min.   :0.0000
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:0.2000
##  3:10   50 :8                    Median :0.5000
```

```
##  4:10   75 :8                    Mean   :0.6733
##         100:8                    3rd Qu.:0.9250
##                                  Max.   :2.0000
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), sum))
```

```
##           COM SEM
## 0   0.0000000 2.0
## 25  1.9000000 1.7
## 50  0.5333333 3.2
## 75  1.9000000 2.4
## 100 6.9000000 6.4
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), mean))
```

```
##           COM   SEM
## 0   0.0000000 0.500
## 25  0.4750000 0.425
## 50  0.1333333 0.800
## 75  0.4750000 0.600
## 100 1.7250000 1.600
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), var))
```

```
##             COM        SEM
## 0   0.000000000 0.00000000
## 25  0.162499990 0.19583332
## 50  0.008888889 0.12666667
## 75  0.042499997 0.04666666
## 100 0.062500002 0.00000000
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), sd))
```

```
##            COM       SEM
## 0   0.00000000 0.0000000
## 25  0.40311288 0.4425306
## 50  0.09428091 0.3559026
## 75  0.20615528 0.2160247
## 100 0.25000000 0.0000000
```

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

```
## Tables of means
## Grand mean
##
## 0.6733333
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.7833 0.5800 0.7400 0.5900
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.2500 0.4500 0.4667 0.5375 1.6625
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.5617 0.7850
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.0000 0.5000
##      25  0.4750 0.4250
##      50  0.1333 0.8000
##      75  0.4750 0.6000
##      100 1.7250 1.6000
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_4 = aggregate(Na ~ 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$Na)$p.value

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

```
## [1] 0.05546459
```

```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."
```

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

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

```
## [1] 0.2967077
```

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

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

```
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Na ~ BLOCO + 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  0.322  0.1073
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1 0.4988  0.4988
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.05433 0.01811
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4 10.150  2.5374  39.029 3.57e-10 ***
## INOCULO:EFLUENTE   4  0.958  0.2394   3.682   0.0178 *
## Residuals         24  1.560  0.0650
## ---
## 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(Na ~ BLOCO + 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, ]

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

```
## Analysis of Variance Table
##
## Response: Na
##                   Df  Sum Sq Mean Sq F value    Pr(>F)
## INOCULO            1  0.4988 0.49878  8.3404  0.007547 **
## EFLUENTE           4 10.1496 2.53740 42.4297 2.874e-11 ***
## INOCULO:EFLUENTE   4  0.9576 0.23940  4.0032  0.011216 *
## 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
print(interacoes_significativas)
```

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

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

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

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

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

}
```

```
##    INOCULO EFLUENTE        Na
## 1      COM        0 0.0000000
## 2      SEM        0 0.5000000
## 3      COM       25 0.4750000
## 4      SEM       25 0.4250000
## 5      COM       50 0.1333333
## 6      SEM       50 0.8000000
## 7      COM       75 0.4750000
## 8      SEM       75 0.6000000
## 9      COM      100 1.7250000
## 10     SEM      100 1.6000000
```

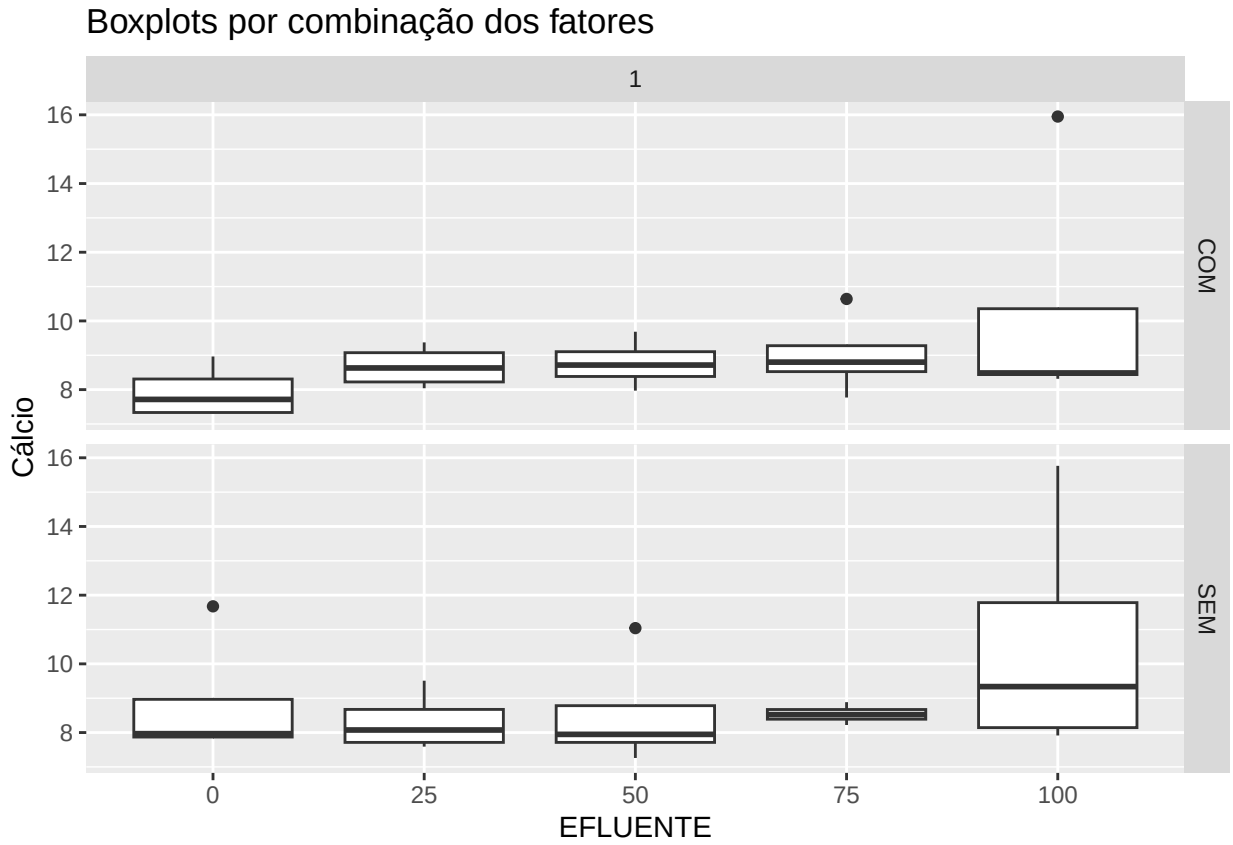
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

```
##                     diif         lwr      upr        p_adj
## SEM:50-COM:0   0.8000000 0.205210684 1.394789 2.806388e-03
## SEM:75-COM:0   0.6000000 0.005210666 1.194789 4.673484e-02
## COM:100-COM:0  1.7250000 1.130210666 2.319789 6.254187e-09
```

```
## SEM:100-COM:0  1.6000000 1.005210696 2.194789 3.034774e-08
## COM:100-SEM:0  1.2250000 0.630210666 1.819789 5.123093e-06
## SEM:100-SEM:0  1.1000000 0.505210696 1.694789 3.182094e-05
## COM:100-COM:25 1.2500000 0.655210672 1.844789 3.577137e-06
## SEM:100-COM:25 1.1250000 0.530210702 1.719789 2.200326e-05
## COM:100-SEM:25 1.3000000 0.705210675 1.894789 1.755941e-06
## SEM:100-SEM:25 1.1750000 0.580210705 1.769789 1.057568e-05
## SEM:50-COM:50  0.6666667 0.071877349 1.261456 1.906953e-02
## COM:100-COM:50 1.5916667 0.996877331 2.186456 3.379627e-08
## SEM:100-COM:50 1.4666667 0.871877361 2.061456 1.759187e-07
## COM:100-SEM:50 0.9250000 0.330210654 1.519789 4.346009e-04
## SEM:100-SEM:50 0.8000000 0.205210684 1.394789 2.806388e-03
## COM:100-COM:75 1.2500000 0.655210657 1.844789 3.577138e-06
## SEM:100-COM:75 1.1250000 0.530210687 1.719789 2.200326e-05
## COM:100-SEM:75 1.1250000 0.530210672 1.719789 2.200327e-05
## SEM:100-SEM:75 1.0000000 0.405210702 1.594789 1.410924e-04
```

Análise para Cálcio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_5, aes(x = factor(EFLUENTE), y = Ca)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Cálcio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ 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$Ca

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ 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$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_5 <- with(dados_5,
                       dados_5[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_5$Ca[which(blocos_dados_5$Ca <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_5$Ca[which(blocos_dados_5$Ca >
                                  limites_outliers$LIM_SUP)] = NA

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

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

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

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

```r
# Substituir o outlier na linha 37 pela média das amostras de EFLUENTE = 100 e
# INÓCULO = "SEM"
blocos_dados_5$Ca[37] = media_blocos_5$Ca[37]
```

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

```
## tibble [40 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 [1:40] 8.97 7.32 7.34 8.09 8.98 ...
```

```
summary(blocos_dados_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Ca
##  1:10   0  :8    COM:20   1:40   Min.   : 7.260
##  2:10   25 :8    SEM:20   2: 0   1st Qu.: 7.904
##  3:10   50 :8                    Median : 8.300
##  4:10   75 :8                    Mean   : 8.405
##         100:8                    3rd Qu.: 8.787
##                                  Max.   :10.589
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

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

```
##          COM      SEM
## 0   31.71000 31.74000
## 25  34.67500 33.25000
## 50  35.08500 30.87333
## 75  33.82667 34.15000
## 100 33.71333 37.17875
```

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

```
##          COM      SEM
## 0   7.927500 7.935000
## 25  8.668750 8.312500
## 50  8.771250 7.718333
## 75  8.456666 8.537500
## 100 8.428333 9.294688
```

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

```
##             COM         SEM
## 0   0.607708372 0.008449974
## 25  0.378389716 0.757875092
## 50  0.519773194 0.109572074
## 75  0.236172103 0.078008365
## 100 0.006438895 2.026471116
```

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

```
##            COM        SEM
## 0   0.77955652 0.09192374
## 25  0.61513390 0.87056022
## 50  0.72095298 0.33101673
## 75  0.48597541 0.27929978
## 100 0.08024273 1.42354175
```

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

```
## Tables of means
## Grand mean
##
## 8.405052
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 8.775 8.160 8.162 8.523
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.931 8.491 8.245 8.497 8.862
##
##  INOCULO
## INOCULO
##  COM  SEM
## 8.45 8.36
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   7.928 7.935
##      25  8.669 8.313
##      50  8.771 7.718
##      75  8.457 8.538
##      100 8.428 9.295
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_5 = aggregate(Ca ~ 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$Ca)$p.value

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

```
## [1] 0.8101944
```

```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_5$Ca,
                                    media_blocos_5$EFLUENTE,
                                    media_blocos_5$INOCULO,
                                    media_blocos_5$CICLO)$p.value

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

```
## [1] 0.06355835
```

```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 + 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  2.699  0.8996
##
## Error: INOCULO
##          Df  Sum Sq Mean Sq
## INOCULO   1 0.08262 0.08262
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.4514  0.1504
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
```

```
## EFLUENTE          4  3.795  0.9486   2.063  0.117
## INOCULO:EFLUENTE  4  3.903  0.9757   2.122  0.109
## Residuals        24 11.036  0.4598
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Ca ~ BLOCO + 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: Ca
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```
# 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
print(interacoes_significativas)
```

```
## character(0)
```

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

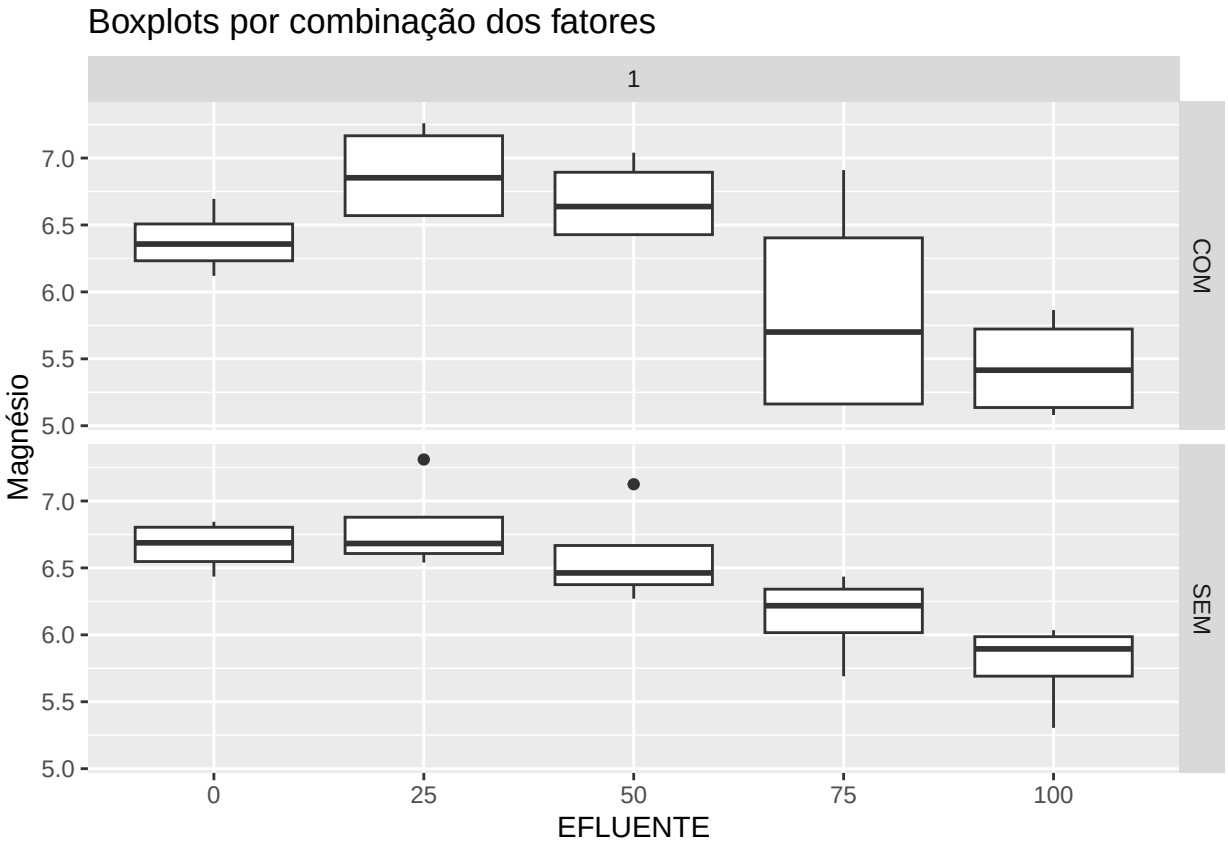
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Ca ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_5, 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 Magnésio

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




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ 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$Mg

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ 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$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_6 <- with(dados_6,
                       dados_6[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_6$Mg[which(blocos_dados_6$Mg <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_6$Mg[which(blocos_dados_6$Mg >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_6 = aggregate(Mg ~ 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$Mg[which(is.na(blocos_dados_6$Mg))] =
  media_blocos_6$Mg[which(is.na(blocos_dados_6$Mg))]
```

```r
# Análises Descritivas
str(blocos_dados_6)
```

```
## tibble [40 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 [1:40] 6.7 6.45 6.27 6.12 7.26 ...
```

```r
summary(blocos_dados_6)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Mg
##  1:10   0  :8    COM:20   1:40   Min.   :5.080
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:6.019
##  3:10   50 :8                    Median :6.425
```

```
##  4:10   75 :8                    Mean   :6.288
##         100:8                    3rd Qu.:6.631
##                                  Max.   :7.260
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

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

```
##        COM      SEM
## 0   25.530 26.65500
## 25  27.535 26.54000
## 50  26.735 25.59333
## 75  23.465 24.56000
## 100 21.775 23.13000
```

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

```
##         COM      SEM
## 0   6.38250 6.663750
## 25  6.88375 6.635000
## 50  6.68375 6.398333
## 75  5.86625 6.140000
## 100 5.44375 5.782500
```

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

```
##            COM         SEM
## 0   0.06104173 0.035772895
## 25  0.13385627 0.006350011
## 50  0.09562290 0.010072212
## 75  0.74100609 0.106216632
## 100 0.14887286 0.109441674
```

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

```
##           COM        SEM
## 0   0.2470662 0.18913724
## 25  0.3658637 0.07968696
## 50  0.3092295 0.10036041
## 75  0.8608171 0.32590893
## 100 0.3858405 0.33081970
```

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

```
## Tables of means
## Grand mean
##
## 6.287958
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 6.555 6.462 6.093 6.043
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 6.523 6.759 6.541 6.003 5.613
##
##  INOCULO
## INOCULO
##   COM   SEM
## 6.252 6.324
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   6.383 6.664
##      25  6.884 6.635
##      50  6.684 6.398
##      75  5.866 6.140
##      100 5.444 5.782
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_6 = aggregate(Mg ~ 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$Mg)$p.value

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

```
## [1] 0.5055782
```

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

```r
  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$Mg,
                                    media_blocos_6$EFLUENTE,
                                    media_blocos_6$INOCULO,
                                    media_blocos_6$CICLO)$p.value

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

```
## [1] 0.9993263
```

```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 + 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  1.996  0.6653
##
## Error: INOCULO
##          Df  Sum Sq Mean Sq
## INOCULO   1 0.05172 0.05172
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.8382  0.2794
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4  7.025  1.7562  27.902 1.03e-08 ***
## INOCULO:EFLUENTE   4  0.773  0.1931   3.068   0.0356 *
## Residuals         24  1.511  0.0629
## ---
## 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 + 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, ]

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

```
## Analysis of Variance Table
##
## Response: Mg
##           Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3 1.9960 0.66532  7.6481 0.0007431 ***
## EFLUENTE   4 7.0249 1.75624 20.1885 8.542e-08 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

```
## character(0)
```

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

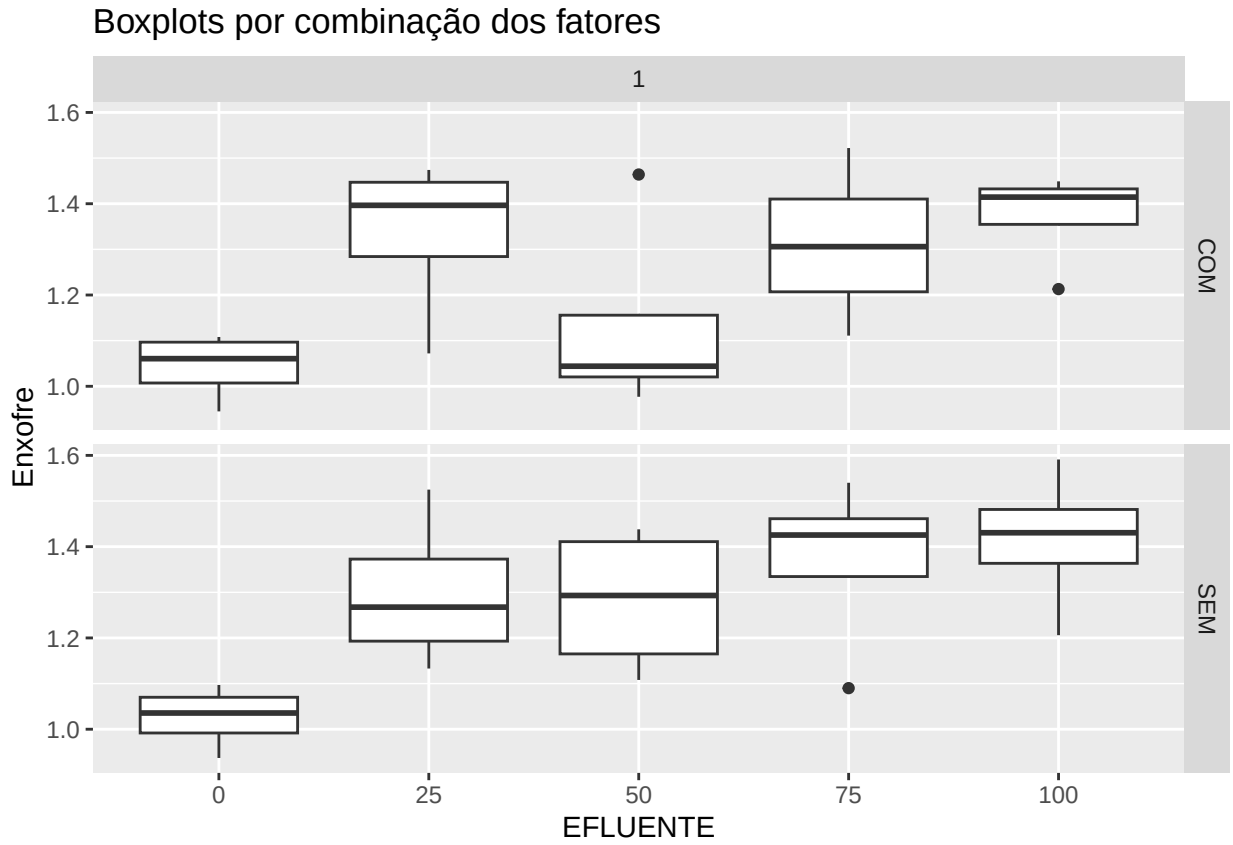
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
    fator2 <- partes[[1]][2]
    fator3 <- partes[[1]][3]
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Mg ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_6, 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 Enxofre

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_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$S

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ 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$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
```

```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$S[which(blocos_dados_7$S <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_7$S[which(blocos_dados_7$S >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_7 = aggregate(S ~ 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$S[which(is.na(blocos_dados_7$S))] =
  media_blocos_7$S[which(is.na(blocos_dados_7$S))]
```

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

```
## tibble [40 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 [1:40] 1.093 1.108 0.945 1.028 1.072 ...
```

```r
summary(blocos_dados_7)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        S
##  1:10   0  :8    COM:20   1:40   Min.   :0.937
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:1.088
##  3:10   50 :8                    Median :1.280
```

```
##  4:10   75 :8                    Mean   :1.262
##         100:8                    3rd Qu.:1.436
##                                  Max.   :1.591
```

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

```
##     COM SEM
## 0     4   4
## 25    4   4
## 50    4   4
## 75    4   4
## 100   4   4
```

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

```
##          COM      SEM
## 0   4.174000 4.105000
## 25  5.339000 5.193000
## 50  4.086667 5.132000
## 75  5.245000 5.854667
## 100 5.704000 5.658000
```

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

```
##          COM      SEM
## 0   1.043500 1.026250
## 25  1.334750 1.298250
## 50  1.021667 1.283000
## 75  1.311250 1.463667
## 100 1.426000 1.414500
```

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

```
##              COM         SEM
## 0   0.0055176707 0.004814251
## 25  0.0331662410 0.028851575
## 50  0.0010515548 0.026203987
## 75  0.0311829210 0.002973554
## 100 0.0003686675 0.025185667
```

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

```
##            COM        SEM
## 0   0.07428103 0.06938481
## 25  0.18211601 0.16985751
## 50  0.03242769 0.16187645
## 75  0.17658687 0.05453031
## 100 0.01920072 0.15869993
```

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

```
## Tables of means
## Grand mean
##
## 1.262283
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 1.3049 1.2728 1.2791 1.1924
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 1.0349 1.3165 1.1523 1.3875 1.4202
##
##  INOCULO
## INOCULO
##    COM    SEM
## 1.2274 1.2971
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   1.0435 1.0262
##      25  1.3347 1.2983
##      50  1.0217 1.2830
##      75  1.3112 1.4637
##      100 1.4260 1.4145
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_7 = aggregate(S ~ 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$S)$p.value

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

```
## [1] 0.05889467
```

```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."
```

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

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

```
## [1] 0.1211888
```

```
# 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(S ~ BLOCO + 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 0.07097 0.02366
##
## Error: INOCULO
##         Df  Sum Sq Mean Sq
## INOCULO  1 0.04858 0.04858
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.03966 0.01322
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4 0.8589  0.2147  14.030 4.91e-06 ***
## INOCULO:EFLUENTE   4 0.1380  0.0345   2.254   0.0931 .
## Residuals         24 0.3673  0.0153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

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

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

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

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

```
## Analysis of Variance Table
##
## Response: S
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## EFLUENTE   4 0.85892 0.21473  14.246 2.258e-06 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

```
## character(0)
```

```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 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(S ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_7, 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)
}
```

```
# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_1, blocos_dados_2["P"],
                    blocos_dados_3["K"], blocos_dados_4["Na"],
                    blocos_dados_5["Ca"], blocos_dados_6["Mg"],
                    blocos_dados_7["S"])


# 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 = "Foliar atualizado - Ciclo 1.xlsx",
      sheetName = "R - Foliar_1", append = FALSE)
```

CICLO 2

```
# Leitura e tratamento dos dados
dados <- read_excel("Foliar 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 5:7) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
dados[5:7] =  round(dados[5:7], digits = 2)
str(dados)
```

```
## tibble [70 x 11] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: num [1:70] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO : chr [1:70] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
##  $ N       : num [1:70] 49.7 53.7 55.4 47.7 45.7 ...
##  $ P       : num [1:70] 2.3 2.29 2.3 2.27 2.45 2.47 2.13 2.44 2.26 2.18 ...
##  $ K       : num [1:70] 16.4 16.1 14.4 14.5 16.6 ...
##  $ Na      : num [1:70] 0 0.5 0.9 0.1 1 ...
##  $ Ca      : num [1:70] 8.97 11.68 8.98 9.51 8.52 ...
##  $ Mg      : num [1:70] 6.7 6.84 7.26 7.31 6.84 ...
##  $ S       : num [1:70] 1.09 1.06 1.07 1.52 1.03 ...
```

```
# 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)
```

```
# Usar apenas dados do Ciclo 2
dados = subset(dados, CICLO == 2)
```

```
# 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)]

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

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

```
str(dados_1)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ N       : num [1:30] 36.5 43.2 38.5 37.3 38.3 ...
```

```
"dados_2"
```

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

```
str(dados_2)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ P       : num [1:30] 1.74 1.68 1.84 1.92 1.86 1.62 2.05 2.25 2.3 2.12 ...
```

```
"dados_3"
```

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

```
str(dados_3)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ K       : num [1:30] 16.1 17.2 13.1 15.1 18.2 ...
```

```
"dados_4"
```

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

```
str(dados_4)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Na      : num [1:30] 1.2 0.6 0 0 0 ...
```

```
"dados_5"
```

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

```
str(dados_5)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Ca      : num [1:30] 17.8 16.5 20.9 19.1 22.2 ...
```

```
"dados_6"
```

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

```
str(dados_5)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Ca      : num [1:30] 17.8 16.5 20.9 19.1 22.2 ...
```

```
"dados_7"
```

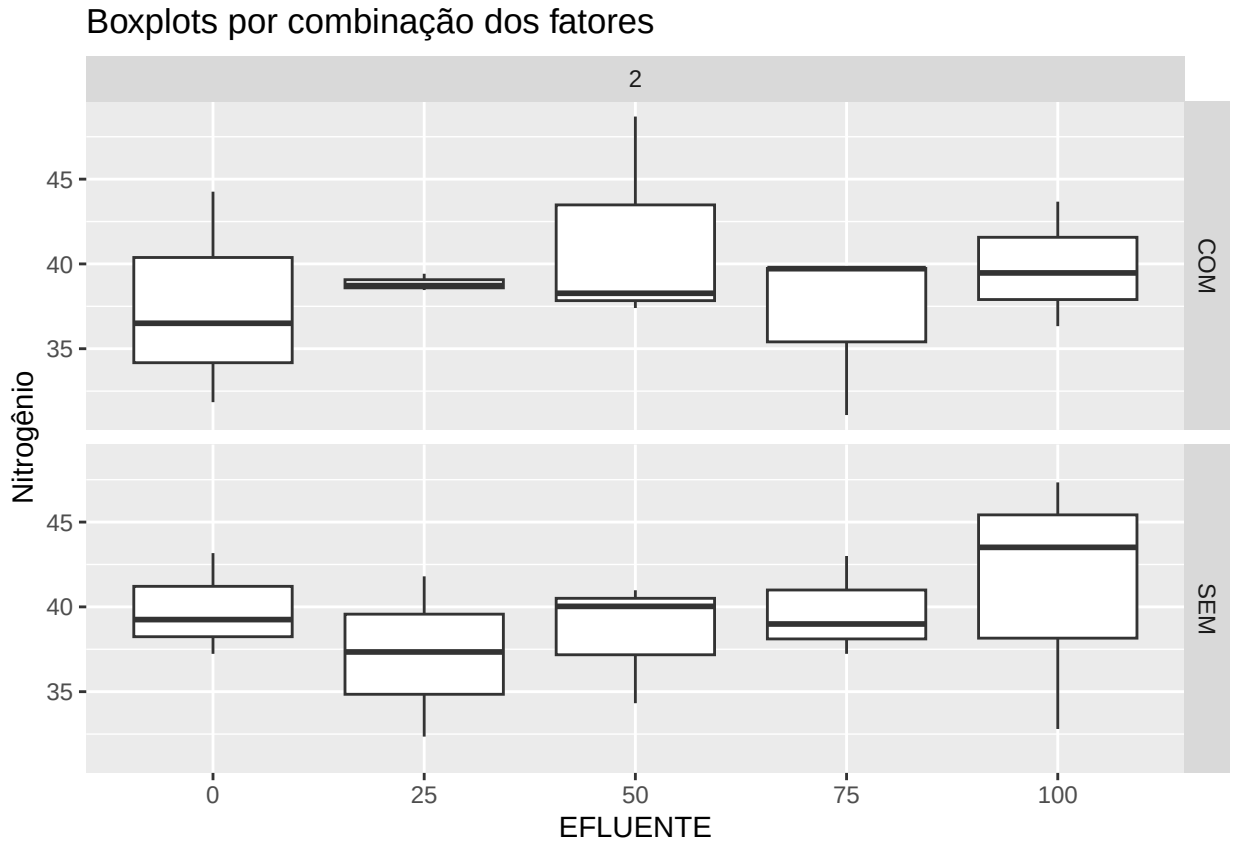
```
## [1] "dados_7"
```

```
str(dados_5)
```

```
## tibble [30 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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Ca      : num [1:30] 17.8 16.5 20.9 19.1 22.2 ...
```

Análise para Nitrogênio

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




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(N ~ 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(N ~ 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$N

# 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_1 <- with(dados_1,
                       dados_1[order(CICLO, INOCULO, EFLUENTE), ])

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

# Calcular a média para cada grupo de 4 linhas
media_blocos_1 = aggregate(N ~ 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$N[which(is.na(blocos_dados_1$N))] =
  media_blocos_1$N[which(is.na(blocos_dados_1$N))]
```

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ N       : num [1:30] 36.5 31.9 44.3 38.5 38.7 ...
```

```r
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        N
##  1:10   0  :6    COM:15   1: 0   Min.   :31.09
##  2:10   25 :6    SEM:15   2:30   1st Qu.:37.23
##  3:10   50 :6                   Median :39.12
```

```
##  4: 0   75 :6                    Mean   :39.10
##         100:6                    3rd Qu.:41.59
##                                  Max.   :48.69
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

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

```
##        COM    SEM
## 0   112.61 119.65
## 25  116.60 111.49
## 50  124.36 115.33
## 75  110.56 119.22
## 100 119.47 123.65
```

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

```
##          COM      SEM
## 0   37.53667 39.88333
## 25  38.86667 37.16333
## 50  41.45333 38.44333
## 75  36.85333 39.74000
## 100 39.82333 41.21667
```

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

```
##            COM       SEM
## 0   39.3080333  9.121733
## 25   0.2465333 22.349033
## 50  39.4662333 12.977033
## 75  24.9122333  8.745100
## 100 13.5625333 56.797433
```

```
with(blocos_dados_1, tapply(N, list(EFLUENTE, INOCULO), sd))
```

```
##           COM      SEM
## 0   6.2696119 3.020221
## 25  0.4965212 4.727476
## 50  6.2822156 3.602365
## 75  4.9912156 2.957212
## 100 3.6827345 7.536407
```

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

```
## Tables of means
## Grand mean
##
## 39.098
##
##  BLOCO
## BLOCO
##     1     2     3
## 39.76 38.61 38.93
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 38.71 38.02 39.95 38.30 40.52
##
##  INOCULO
## INOCULO
##   COM   SEM
## 38.91 39.29
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   37.54 39.88
##      25  38.87 37.16
##      50  41.45 38.44
##      75  36.85 39.74
##      100 39.82 41.22
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_1 = aggregate(N ~ 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$N)$p.value

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

```
## [1] 0.5564032
```

```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."
```

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

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

```
## [1] 0.9654929
```

```
# 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(N ~ BLOCO + 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  2  7.063   3.531
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  1.098   1.098
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  2  18.09   9.046
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4   28.3   7.066   0.263  0.897
## INOCULO:EFLUENTE  4   40.5  10.129   0.377  0.822
## Residuals        16  429.8  26.864
```

```r
modelo = modelo_PARCELASUB
```

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

# 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: N
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

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

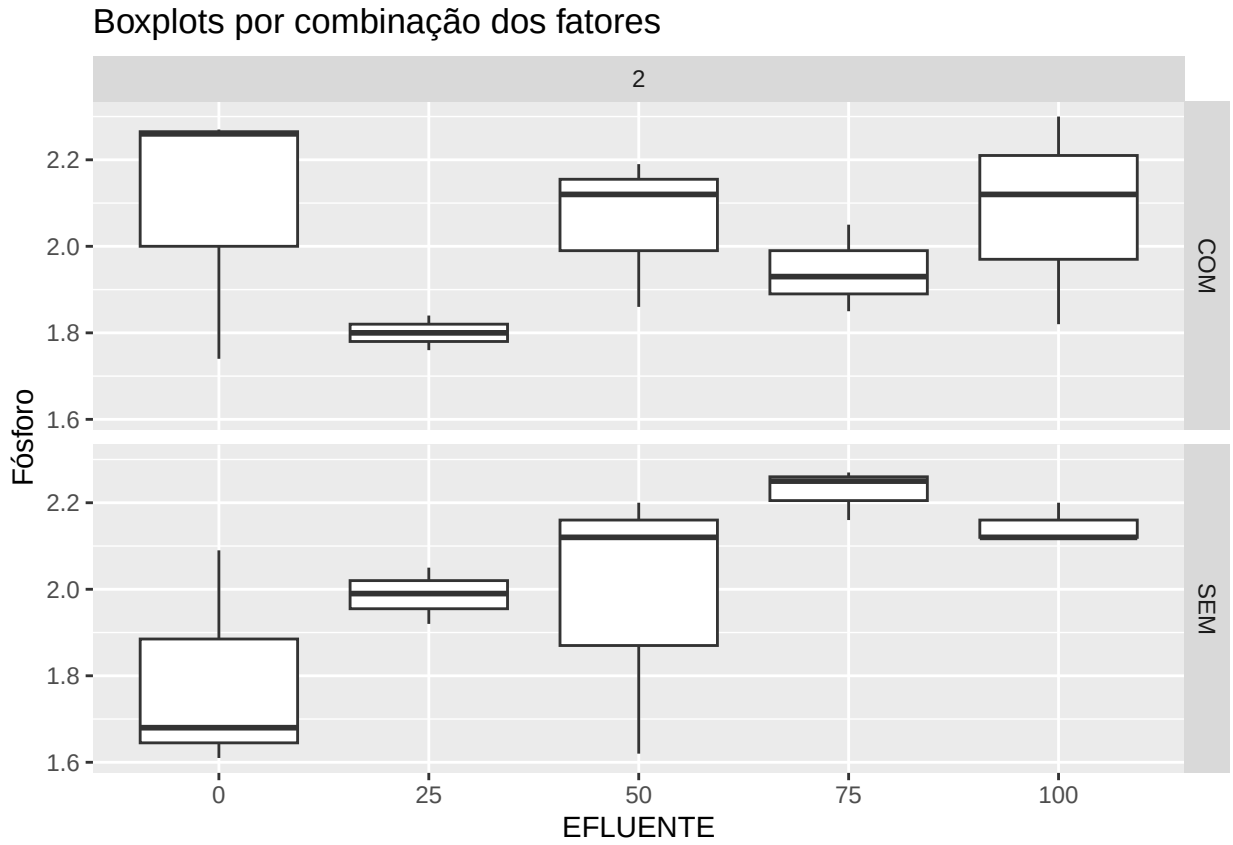
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(N ~ 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 Fósforo

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P ~ 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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(P ~ 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

# 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_2 <- with(dados_2,
                           dados_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_2$P[which(blocos_dados_2$P <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_2$P[which(blocos_dados_2$P >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_2 = aggregate(P ~ 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$P[which(is.na(blocos_dados_2$P))] =
  media_blocos_2$P[which(is.na(blocos_dados_2$P))]
```

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ P       : num [1:30] 1.74 2.27 2.26 1.84 1.8 1.76 1.86 2.12 2.19 2.05 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        P
##  1:10   0  :6    COM:15   1: 0   Min.   :1.610
##  2:10   25 :6    SEM:15   2:30   1st Qu.:1.843
##  3:10   50 :6                   Median :2.070
```

```
##  4: 0   75 :6                    Mean   :2.010
##         100:6                    3rd Qu.:2.183
##                                  Max.   :2.300
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

```
with(blocos_dados_2, tapply(P, list(EFLUENTE, INOCULO), sum))
```

```
##      COM  SEM
## 0   6.27 5.38
## 25  5.40 5.96
## 50  6.17 5.94
## 75  5.83 6.68
## 100 6.24 6.44
```

```
with(blocos_dados_2, tapply(P, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   2.090000 1.793333
## 25  1.800000 1.986667
## 50  2.056667 1.980000
## 75  1.943333 2.226667
## 100 2.080000 2.146667
```

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

```
##            COM         SEM
## 0   0.09190000 0.067233333
## 25  0.00160000 0.004233333
## 50  0.03023333 0.098800000
## 75  0.01013333 0.003433333
## 100 0.05880000 0.002133333
```

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

```
##           COM        SEM
## 0   0.3031501 0.25929391
## 25  0.0400000 0.06506407
## 50  0.1738774 0.31432467
## 75  0.1006645 0.05859465
## 100 0.2424871 0.04618802
```

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

```
## Tables of means
## Grand mean
##
## 2.010333
##
##  BLOCO
## BLOCO
##     1     2     3
## 1.938 2.089 2.004
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 1.9417 1.8933 2.0183 2.0850 2.1133
##
##  INOCULO
## INOCULO
##    COM    SEM
## 1.9940 2.0267
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   2.0900 1.7933
##      25  1.8000 1.9867
##      50  2.0567 1.9800
##      75  1.9433 2.2267
##      100 2.0800 2.1467
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_2 = aggregate(P ~ 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$P)$p.value

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

```
## [1] 0.6815655
```

```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."
```

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

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

```
## [1] 0.6972943
```

```
# 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 ~ BLOCO + 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  2 0.1146  0.0573
##
## Error: INOCULO
##          Df   Sum Sq  Mean Sq
## INOCULO  1 0.008003 0.008003
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  2 0.02993 0.01496
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4 0.2079 0.05198   1.404  0.277
## INOCULO:EFLUENTE  4 0.3122 0.07804   2.108  0.127
## Residuals        16 0.5925 0.03703
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(P ~ BLOCO + 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
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

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

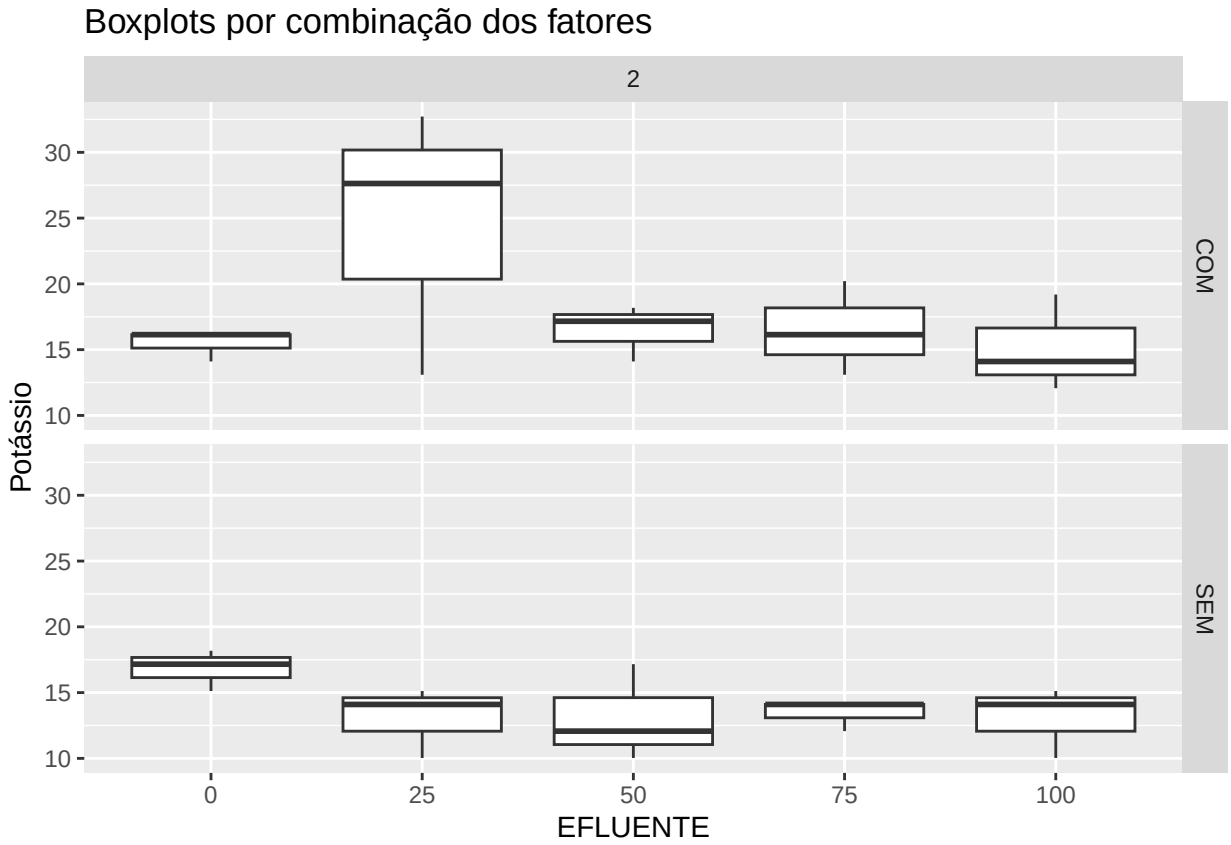
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(P ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_2, 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 Potássio

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K ~ 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$K

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(K ~ 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$K

# 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_3 <- with(dados_3,
                           dados_3[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_3$K[which(blocos_dados_3$K <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_3$K[which(blocos_dados_3$K >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_3 = aggregate(K ~ 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$K[which(is.na(blocos_dados_3$K))] =
  media_blocos_3$K[which(is.na(blocos_dados_3$K))]
```

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ K       : num [1:30] 16.1 16.1 14.1 13.1 32.7 ...
```

```r
summary(blocos_dados_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        K
##  1:10   0  :6    COM:15   1: 0   Min.   :10.03
##  2:10   25 :6    SEM:15   2:30   1st Qu.:13.34
##  3:10   50 :6                    Median :14.61
```

```
##  4: 0   75 :6                    Mean   :15.75
##         100:6                    3rd Qu.:17.16
##                                  Max.   :32.72
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

```
with(blocos_dados_3, tapply(K, list(EFLUENTE, INOCULO), sum))
```

```
##       COM   SEM
## 0   46.38 50.46
## 25  73.44 39.25
## 50  49.44 39.26
## 75  49.44 40.27
## 100 45.36 39.25
```

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

```
##       COM      SEM
## 0   15.46 16.82000
## 25  24.48 13.08333
## 50  16.48 13.08667
## 75  16.48 13.42333
## 100 15.12 13.08333
```

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

```
##          COM       SEM
## 0     1.3872  2.427600
## 25  103.7761  7.252233
## 50    4.5084 13.484433
## 75   12.7603  1.373633
## 100  13.4539  7.252233
```

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

```
##           COM      SEM
## 0    1.177795 1.558076
## 25  10.187056 2.692997
## 50   2.123299 3.672116
## 75   3.572156 1.172021
## 100  3.667956 2.692997
```

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

```
## Tables of means
## Grand mean
##
## 15.75167
##
##  BLOCO
## BLOCO
##      1      2      3
## 15.529 16.372 15.354
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 16.140 18.782 14.783 14.952 14.102
##
##  INOCULO
## INOCULO
##    COM    SEM
## 17.604 13.899
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   15.460 16.820
##      25  24.480 13.083
##      50  16.480 13.087
##      75  16.480 13.423
##      100 15.120 13.083
```

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

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

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

```
## [1] 0.00329588
```

```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_3$K,
                                    media_blocos_3$EFLUENTE,
                                    media_blocos_3$INOCULO,
                                    media_blocos_3$CICLO)$p.value

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

```
## [1] 0.4018438
```

```
# 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(K ~ BLOCO + 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  2  5.925   2.963
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  102.9   102.9
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  2  51.62   25.81
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  81.79   20.45   1.178  0.358
## INOCULO:EFLUENTE  4 132.18   33.04   1.903  0.159
## Residuals        16 277.81   17.36
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(K ~ BLOCO + 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: K
##          Df Sum Sq Mean Sq F value  Pr(>F)
## INOCULO   1 102.93  102.93  5.6244 0.02907 *
## 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
print(interacoes_significativas)
```

```
## character(0)
```

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

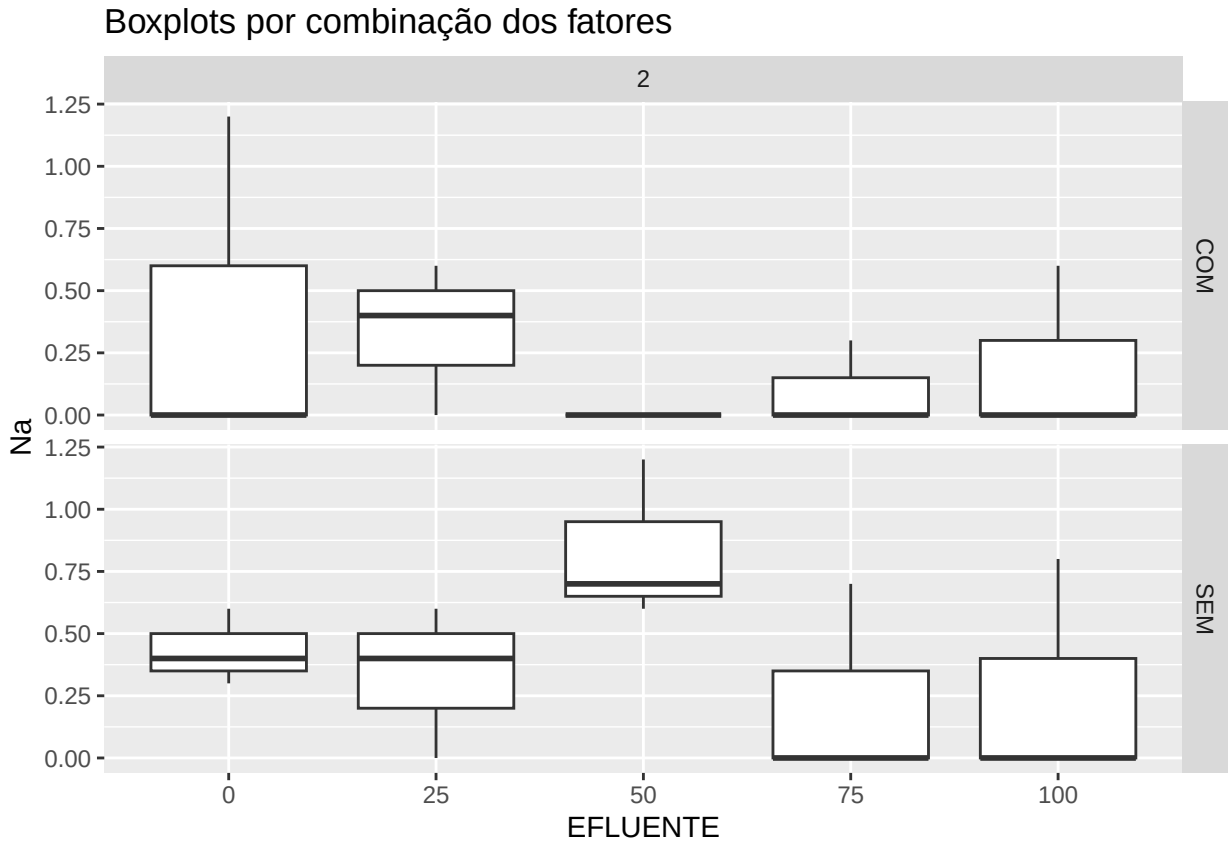
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

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

}
```

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

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

Análise para Sódio

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_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$Na

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Na ~ 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$Na

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

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

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

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

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

# Definir como NA (valor ausente) os outliers
blocos_dados_4$Na[which(blocos_dados_4$Na <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_4$Na[which(blocos_dados_4$Na >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_4 = aggregate(Na ~ 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$Na[which(is.na(blocos_dados_4$Na))] =
  media_blocos_4$Na[which(is.na(blocos_dados_4$Na))]
```

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Na      : num [1:30] 1.2 0 0 0 0.4 ...
```

```r
summary(blocos_dados_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Na
##  1:10   0  :6    COM:15   1: 0   Min.   :0.0000
##  2:10   25 :6    SEM:15   2:30   1st Qu.:0.0000
##  3:10   50 :6                    Median :0.1500
```

```
##  4: 0   75 :6                   Mean   :0.3133
##         100:6                   3rd Qu.:0.6000
##                                 Max.   :1.2000
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), sum))
```

```
##     COM SEM
## 0   1.2 1.3
## 25  1.0 1.0
## 50  0.0 2.5
## 75  0.3 0.7
## 100 0.6 0.8
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), mean))
```

```
##           COM       SEM
## 0   0.4000000 0.4333333
## 25  0.3333333 0.3333333
## 50  0.0000000 0.8333334
## 75  0.1000000 0.2333333
## 100 0.2000000 0.2666667
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), var))
```

```
##            COM        SEM
## 0   0.48000004 0.02333334
## 25  0.09333334 0.09333334
## 50  0.00000000 0.10333335
## 75  0.03000000 0.16333333
## 100 0.12000001 0.21333334
```

```
with(blocos_dados_4, tapply(Na, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.6928204 0.1527525
## 25  0.3055051 0.3055051
## 50  0.0000000 0.3214550
## 75  0.1732051 0.4041452
## 100 0.3464102 0.4618802
```

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

```
## Tables of means
## Grand mean
##
## 0.3133333
##
##  BLOCO
## BLOCO
##    1    2    3
## 0.24 0.24 0.46
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.4167 0.3333 0.4167 0.1667 0.2333
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.2067 0.4200
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.4000 0.4333
##      25  0.3333 0.3333
##      50  0.0000 0.8333
##      75  0.1000 0.2333
##      100 0.2000 0.2667
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_4 = aggregate(Na ~ 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$Na)$p.value

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

```
## [1] 0.277757
```

```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."
```

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

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

```
## [1] 0
```

```
# 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 não é homogênea entre os grupos."
```

```
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(Na ~ BLOCO + 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  2 0.3227  0.1613
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1 0.3413  0.3413
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  2 0.2427  0.1213
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4 0.2980  0.0745   0.575  0.685
## INOCULO:EFLUENTE  4 0.7353  0.1838   1.418  0.273
## Residuals        16 2.0747  0.1297
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Na ~ BLOCO + 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, ]

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

```
## Analysis of Variance Table
##
## Response: Na
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

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

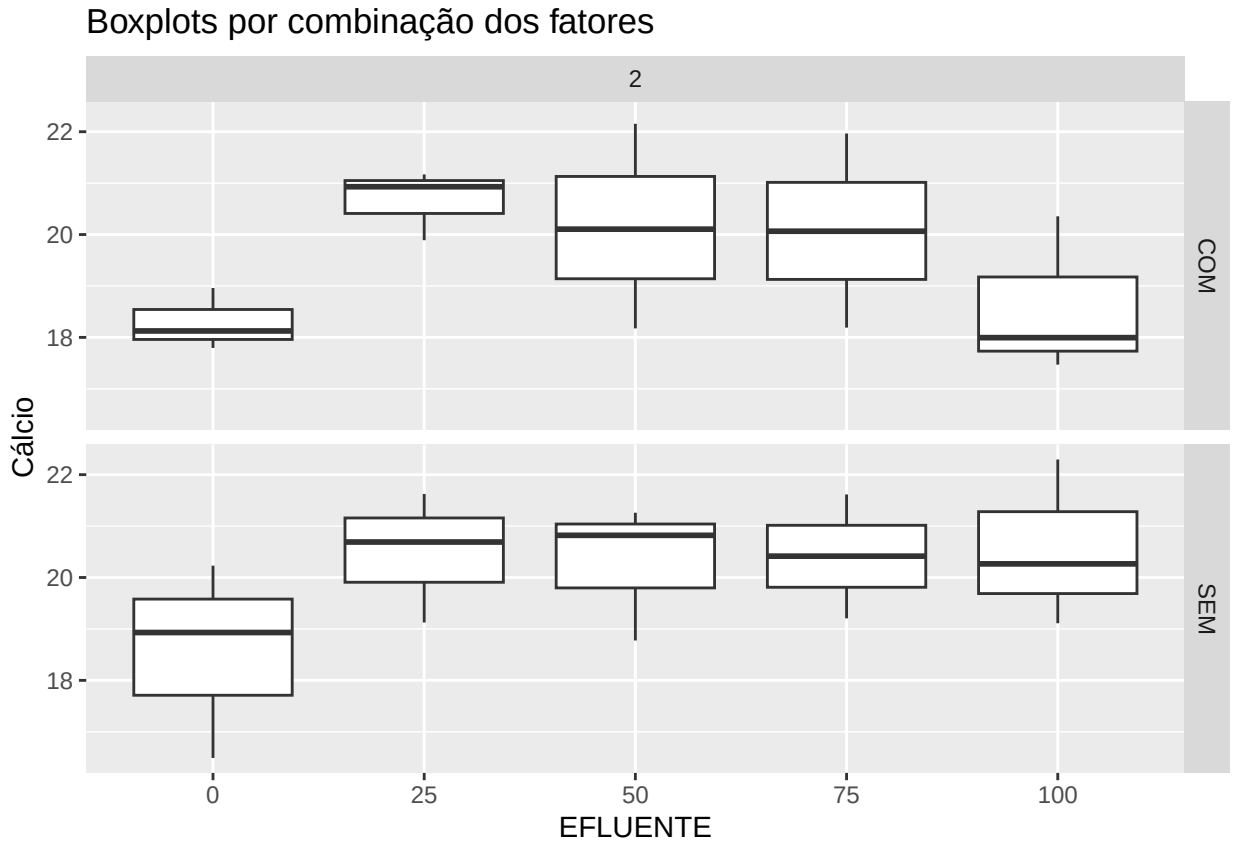
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

}
```

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

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

Análise para Cálcio

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_5, aes(x = factor(EFLUENTE), y = Ca)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Cálcio") +
  ggtitle("Boxplots por combinação dos fatores")
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ 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$Ca

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Ca ~ 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$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_5 <- with(dados_5,
                       dados_5[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_5$Ca[which(blocos_dados_5$Ca <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_5$Ca[which(blocos_dados_5$Ca >
                                  limites_outliers$LIM_SUP)] = NA

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

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

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

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

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Ca      : num [1:30] 17.8 19 18.1 20.9 21.2 ...
```

```r
summary(blocos_dados_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Ca
##  1:10   0  :6    COM:15   1: 0   Min.   :16.49
##  2:10   25 :6    SEM:15   2:30   1st Qu.:18.81
##  3:10   50 :6                   Median :20.09
```

```
##  4: 0   75 :6                    Mean   :19.81
##         100:6                    3rd Qu.:20.90
##                                  Max.   :22.30
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

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

```
##        COM    SEM
## 0   54.880 55.650
## 25  61.990 61.440
## 50  60.435 60.855
## 75  60.220 61.235
## 100 55.820 61.670
```

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

```
##          COM      SEM
## 0   18.29333 18.55000
## 25  20.66333 20.48000
## 50  20.14500 20.28500
## 75  20.07333 20.41167
## 100 18.60667 20.55667
```

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

```
##           COM      SEM
## 0   0.3605577 3.605200
## 25  0.4629339 1.595575
## 50  3.9613029 1.758476
## 75  3.5627076 1.452033
## 100 2.3614078 2.599858
```

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

```
##           COM      SEM
## 0   0.6004646 1.898736
## 25  0.6803925 1.263161
## 50  1.9903022 1.326075
## 75  1.8875136 1.205003
## 100 1.5366873 1.612407
```

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

```
## Tables of means
## Grand mean
##
## 19.8065
##
##  BLOCO
## BLOCO
##      1      2      3
## 19.417 19.952 20.050
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 18.422 20.572 20.215 20.243 19.582
##
##  INOCULO
## INOCULO
##    COM    SEM
## 19.556 20.057
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   18.293 18.550
##      25  20.663 20.480
##      50  20.145 20.285
##      75  20.073 20.412
##      100 18.607 20.557
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_5 = aggregate(Ca ~ 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$Ca)$p.value

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

```
## [1] 0.007675188
```

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

```
  print("Os dados não seguem uma distribuição normal.")
}
```

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

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

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

```
## [1] 0.1356611
```

```
# 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(Ca ~ BLOCO + 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  2  2.319   1.159
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  1.877   1.877
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  2  12.04    6.02
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4 17.465   4.366   2.402 0.0929 .
## INOCULO:EFLUENTE   4  4.177   1.044   0.574 0.6851
## Residuals         16 29.080   1.818
## ---
## 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 + 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: Ca
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

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

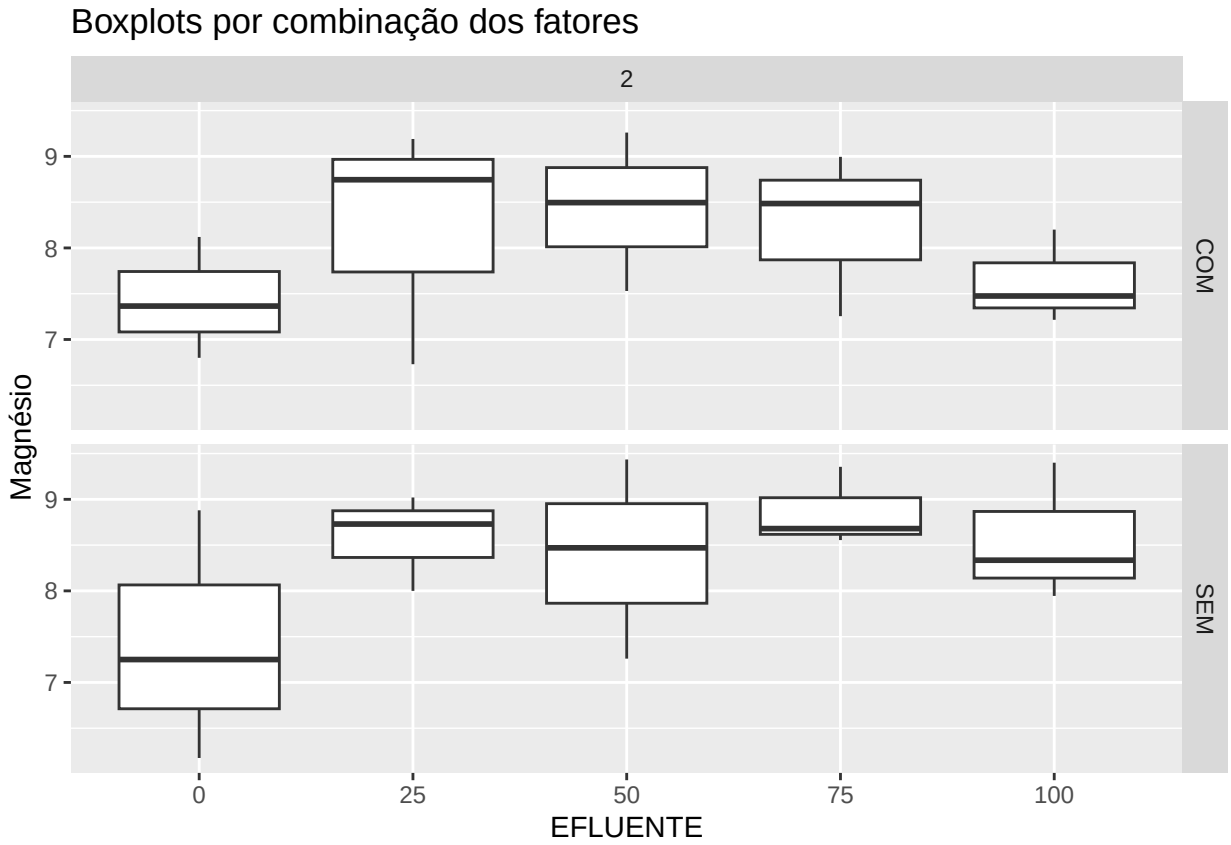
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Ca ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_5, 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 Magnésio

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




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ 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$Mg

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(Mg ~ 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$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_6 <- with(dados_6,
                       dados_6[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_6$Mg[which(blocos_dados_6$Mg <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_6$Mg[which(blocos_dados_6$Mg >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_6 = aggregate(Mg ~ 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$Mg[which(is.na(blocos_dados_6$Mg))] =
  media_blocos_6$Mg[which(is.na(blocos_dados_6$Mg))]
```

```r
# Análises Descritivas
str(blocos_dados_6)
```

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Mg      : num [1:30] 6.8 8.12 7.36 9.19 8.74 ...
```

```r
summary(blocos_dados_6)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        Mg
##  1:10   0  :6    COM:15   1: 0   Min.   :6.175
##  2:10   25 :6    SEM:15   2:30   1st Qu.:7.393
##  3:10   50 :6                   Median :8.402
```

```
##  4: 0   75 :6                    Mean   :8.178
##         100:6                    3rd Qu.:8.846
##                                  Max.   :9.435
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

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

```
##        COM    SEM
## 0   22.285 22.305
## 25  24.665 25.750
## 50  25.285 25.165
## 75  24.735 26.590
## 100 22.890 25.680
```

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

```
##          COM      SEM
## 0   7.428333 7.435000
## 25  8.221666 8.583333
## 50  8.428333 8.388334
## 75  8.245000 8.863333
## 100 7.630000 8.560000
```

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

```
##           COM       SEM
## 0   0.4386081 1.8549249
## 25  1.7183078 0.2762335
## 50  0.7515583 1.1876585
## 75  0.8000997 0.1852080
## 100 0.2605748 0.5672246
```

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

```
##           COM       SEM
## 0   0.6622750 1.3619563
## 25  1.3108424 0.5255792
## 50  0.8669246 1.0897975
## 75  0.8944829 0.4303579
## 100 0.5104653 0.7531431
```

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

```
## Tables of means
## Grand mean
##
## 8.178333
##
##  BLOCO
## BLOCO
##     1     2     3
## 7.991 8.276 8.268
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.432 8.402 8.408 8.554 8.095
##
##  INOCULO
## INOCULO
##   COM   SEM
## 7.991 8.366
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   7.428 7.435
##      25  8.222 8.583
##      50  8.428 8.388
##      75  8.245 8.863
##      100 7.630 8.560
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_6 = aggregate(Mg ~ 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$Mg)$p.value

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

```
## [1] 0.1612175
```

```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."
```

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

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

```
## [1] 0.05181671
```

```
# 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(Mg ~ BLOCO + 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  2 0.5267  0.2634
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  1.057   1.057
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  2  4.613   2.306
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  4.853  1.2133   1.774  0.183
## INOCULO:EFLUENTE  4  1.013  0.2532   0.370  0.826
## Residuals        16 10.942  0.6838
```

```r
modelo = modelo_PARCELASUB
```

```r
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(Mg ~ BLOCO + 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, ]

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

```
## Analysis of Variance Table
##
## Response: Mg
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

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

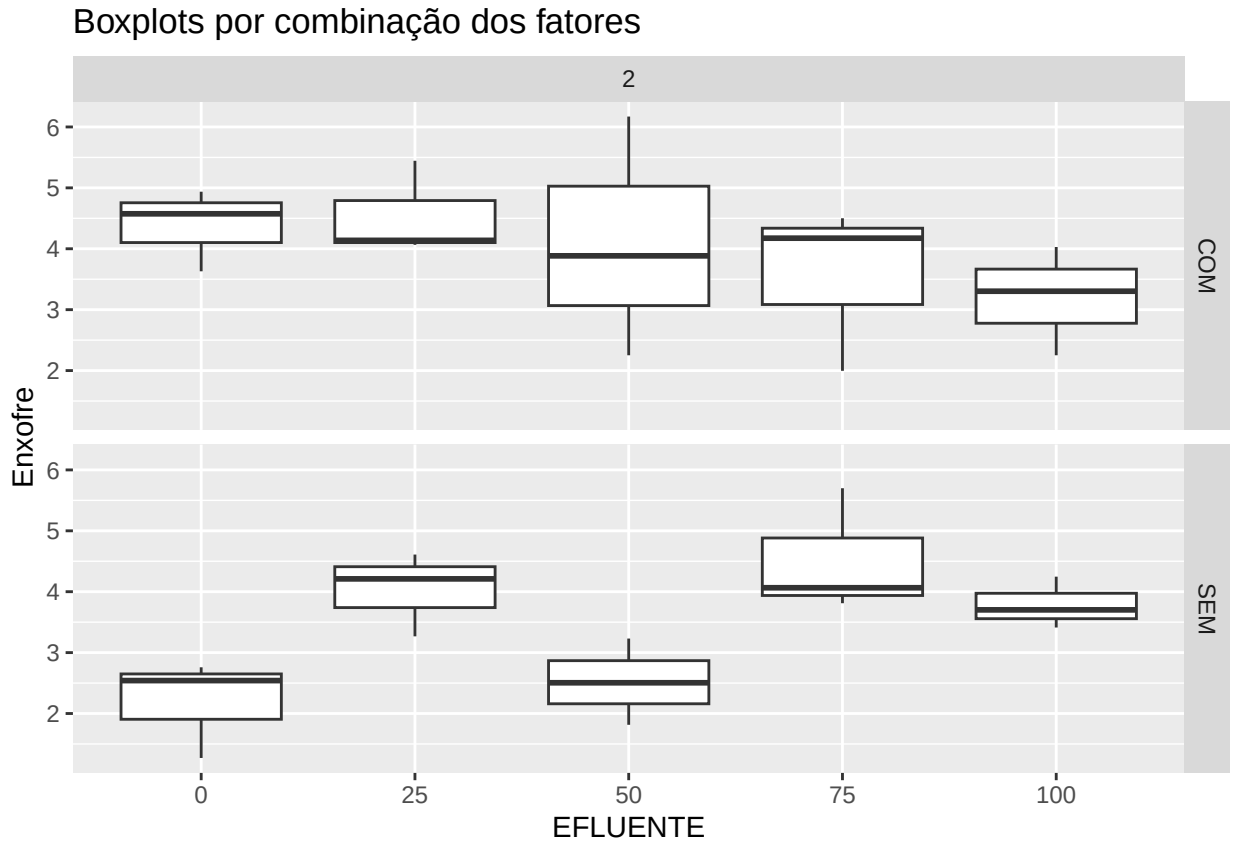
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(Mg ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_6, 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 Enxofre

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




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_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$S

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(S ~ 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$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_7 <- with(dados_7,
                           dados_7[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_7$S[which(blocos_dados_7$S <
                                   limites_outliers$LIM_INF)] = NA
blocos_dados_7$S[which(blocos_dados_7$S >
                                   limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_7 = aggregate(S ~ 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$S[which(is.na(blocos_dados_7$S))] =
  media_blocos_7$S[which(is.na(blocos_dados_7$S))]
```

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

```
## tibble [30 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 2 3 1 2 3 1 2 3 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ S       : num [1:30] 4.94 4.57 3.63 4.14 4.07 ...
```

```
summary(blocos_dados_7)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO        S
##  1:10   0  :6    COM:15   1: 0   Min.   :1.270
##  2:10   25 :6    SEM:15   2:30   1st Qu.:2.877
##  3:10   50 :6                    Median :3.848
```

```
##  4: 0   75 :6                    Mean   :3.683
##         100:6                    3rd Qu.:4.238
##                                  Max.   :6.171
```

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

```
##     COM SEM
## 0     3   3
## 25    3   3
## 50    3   3
## 75    3   3
## 100   3   3
```

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

```
##        COM    SEM
## 0   13.141  6.570
## 25  13.648 12.088
## 50  12.306  7.551
## 75  10.671 13.575
## 100  9.583 11.361
```

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

```
##          COM      SEM
## 0   4.380333 2.190000
## 25  4.549333 4.029333
## 50  4.102000 2.517000
## 75  3.557000 4.525000
## 100 3.194333 3.787000
```

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

```
##           COM       SEM
## 0   0.4551921 0.6466810
## 25  0.6029965 0.4756644
## 50  3.8772431 0.5013719
## 75  1.8542727 1.0498358
## 100 0.7991772 0.1797251
```

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

```
##           COM       SEM
## 0   0.6746793 0.8041648
## 25  0.7765285 0.6896843
## 50  1.9690716 0.7080762
## 75  1.3617168 1.0246149
## 100 0.8939671 0.4239400
```

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

```
## Tables of means
## Grand mean
##
## 3.683133
##
##  BLOCO
## BLOCO
##     1     2     3
## 3.757 3.488 3.804
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 3.285 4.289 3.310 4.041 3.491
##
##  INOCULO
## INOCULO
##   COM   SEM
## 3.957 3.410
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   4.380 2.190
##      25  4.549 4.029
##      50  4.102 2.517
##      75  3.557 4.525
##      100 3.194 3.787
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_7 = aggregate(S ~ 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$S)$p.value

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

```
## [1] 0.2314208
```

```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."
```

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

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

```
## [1] 0.7363913
```

```
# 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(S ~ BLOCO + 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  2 0.5805  0.2903
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  2.244   2.244
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  2 0.05275 0.02638
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  4.983   1.246   0.984  0.444
## INOCULO:EFLUENTE  4 11.059   2.765   2.184  0.117
## Residuals        16 20.251   1.266
```

```r
modelo = modelo_PARCELASUB
```

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

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

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

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

```
## Analysis of Variance Table
##
## Response: S
##    Df Sum Sq Mean Sq F value Pr(>F)
## NA
```

```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
print(interacoes_significativas)
```

```
## character(0)
```

```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 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(S ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_7, 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)
}
```

```
# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_1, blocos_dados_2["P"],
                    blocos_dados_3["K"], blocos_dados_4["Na"],
                    blocos_dados_5["Ca"], blocos_dados_6["Mg"],
                    blocos_dados_7["S"])


# Criar planilha com todos os dados atualizados
library("xlsx")
write.xlsx(dados_final, file = "Foliar atualizado - Ciclo 2.xlsx",
      sheetName = "R - Foliar_2", append = FALSE)
```