Clorofilômetro Clorofila Total

Ana Carolina Murad Lima

2023-06-23

```
# 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("Clorofilômetro ClorfTotal ok.xlsx")

# Excluir colunas irrelevantes para a análise
dados = dados[-c(1:3)]

# 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 5:9) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
```

```
## Warning: NAs introduzidos por coerção
```

```r
dados[5:9] =  round(dados[5:9], digits = 2)
str(dados)
```

```
## tibble [80 x 9] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO      : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE   : num [1:80] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO    : chr [1:80] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO      : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
##  $ 24 E 29 DAS: num [1:80] 45.5 48.7 49.3 52.1 48.6 ...
##  $ 43 E 40 DAS: num [1:80] 39.5 42.2 38.6 42.1 43.9 ...
##  $ 52 E 54 DAS: num [1:80] 41.7 42.1 40.4 41.1 41.2 ...
##  $ 64 E 61 DAS: num [1:80] 41.6 40.7 43.2 43.1 40 ...
##  $ 114 DAS    : num [1:80] 55 53.3 57 55.8 57.9 ...
```

```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
```

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

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

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

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

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

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

```
# 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_tempo_1 = dados[c(1:5)] # 24, 29
dados_tempo_1$ClorfTotal = dados_tempo_1$`24 E 29 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43, 40
dados_tempo_2$ClorfTotal = dados_tempo_2$`43 E 40 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$ClorfTotal = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfTotal = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfTotal = dados_tempo_5$`114 DAS`
dados_tempo_5 = dados_tempo_5[-5]

# Estrutura dos dados após separados
"dados_tempo_1"
```

```
## [1] "dados_tempo_1"
```

```r
str(dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 45.5 48.7 49.3 52.1 48.6 ...
```

```r
"dados_tempo_2"
```

```
## [1] "dados_tempo_2"
```

```r
str(dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 39.5 42.2 38.6 42.1 43.9 ...
```

```r
"dados_tempo_3"
```

```
## [1] "dados_tempo_3"
```

```
str(dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 41.7 42.1 40.4 41.1 41.2 ...
```

```
"dados_tempo_4"
```

```
## [1] "dados_tempo_4"
```

```
str(dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 41.6 40.7 43.2 43.1 40 ...
```

```
"dados_tempo_5"
```

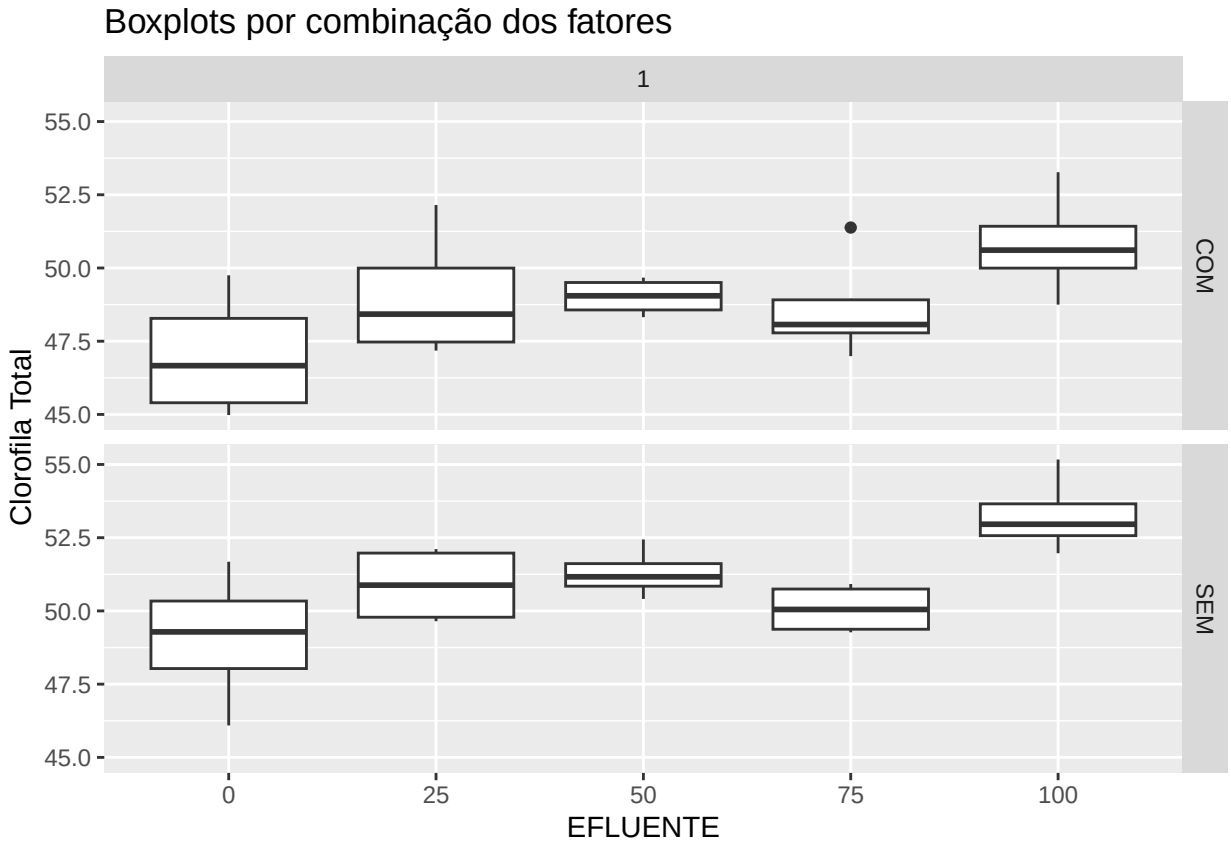
```
## [1] "dados_tempo_5"
```

```
str(dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 55 53.3 57 55.8 57.9 ...
```

Análise para 24 e 29 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_1, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_1 <- with(dados_tempo_1,
                             dados_tempo_1[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_1$ClorfTotal[which(blocos_dados_tempo_1$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_1$ClorfTotal[which(blocos_dados_tempo_1$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_1 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_1, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_1 = media_blocos_tempo_1[rep(row.names(media_blocos_tempo_1),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_1) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_1$ClorfTotal[which(is.na(blocos_dados_tempo_1$ClorfTotal))] =
  media_blocos_tempo_1$ClorfTotal[which(is.na(blocos_dados_tempo_1$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 45.5 49.8 45 47.8 49.3 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1:40   Min.   :44.98
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:48.26
##  3:10   50 :8                    Median :49.71
##  4:10   75 :8                    Mean   :49.82
##         100:8                    3rd Qu.:51.42
##                                  Max.   :55.17
```

```r
# Número de observações
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##        COM    SEM
## 0   188.06 196.34
## 25  196.18 203.52
## 50  196.09 205.18
## 75  190.84 200.29
## 100 203.24 213.06
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##         COM     SEM
## 0   47.0150 49.0850
## 25  49.0450 50.8800
## 50  49.0225 51.2950
## 75  47.7100 50.0725
## 100 50.8100 53.2650
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##           COM       SEM
## 0   4.7992333 5.5053667
## 25  5.1167000 1.7436000
## 50  0.4114250 0.7297667
## 75  0.2594667 0.7274917
## 100 3.4850667 1.8547667
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   2.1907153 2.3463518
## 25  2.2620124 1.3204545
## 50  0.6414242 0.8542638
## 75  0.5093787 0.8529312
## 100 1.8668333 1.3618982
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_1,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 49.82
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 49.34 51.45 49.06 49.43
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 48.05 49.96 50.16 48.89 52.04
##
##  INOCULO
## INOCULO
##   COM   SEM
## 48.72 50.92
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   47.02 49.08
##      25  49.04 50.88
##      50  49.02 51.29
##      75  47.71 50.07
##      100 50.81 53.26
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_1 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_1, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_1$ClorfTotal)$p.value

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

```
## [1] 0.882332
```

```
# 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_tempo_1$ClorfTotal,
```

```
                                    media_blocos_tempo_1$EFLUENTE,
                                    media_blocos_tempo_1$INOCULO,
                                    media_blocos_tempo_1$CICLO)$p.value

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

```
## [1] 0.9993249
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_1)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  36.03   12.01
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  48.36   48.36
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  1.546  0.5154
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4  72.38  18.096  11.956 1.74e-05 ***
## INOCULO:EFLUENTE   4   0.49   0.123   0.082    0.987
## Residuals         24  36.32   1.513
## ---
## 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_1)
```

```r
# 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: ClorfTotal
##           Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3 36.029  12.010  8.5625 0.0003705 ***
## INOCULO    1 48.356  48.356 34.4764 2.969e-06 ***
## EFLUENTE   4 72.383  18.096 12.9017 5.354e-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)
```

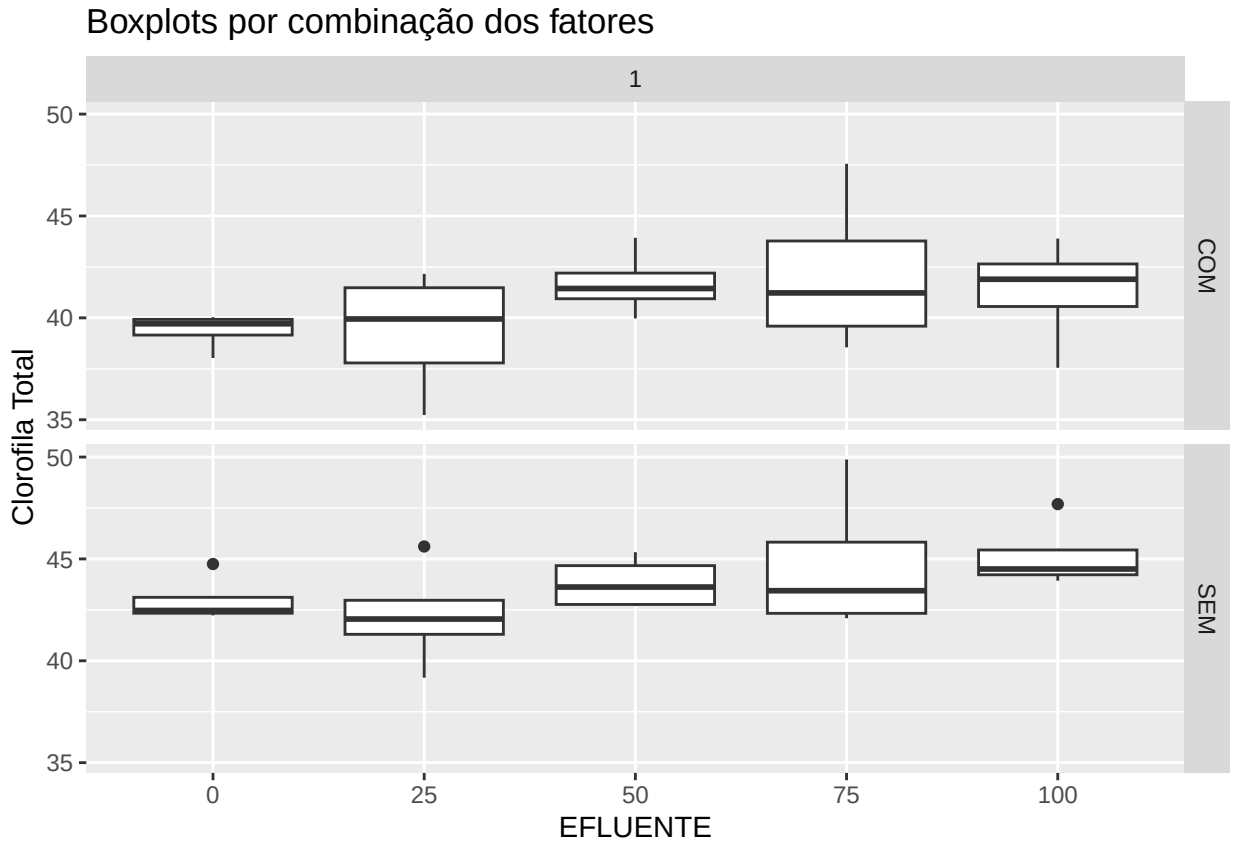
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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 43 e 40 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_2, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_2 <- with(dados_tempo_2,
                             dados_tempo_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_2$ClorfTotal[which(blocos_dados_tempo_2$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_2$ClorfTotal[which(blocos_dados_tempo_2$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_2 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_2, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_2 = media_blocos_tempo_2[rep(row.names(media_blocos_tempo_2),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_2) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_2$ClorfTotal[which(is.na(blocos_dados_tempo_2$ClorfTotal))] =
  media_blocos_tempo_2$ClorfTotal[which(is.na(blocos_dados_tempo_2$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 39.5 39.9 38 40 38.6 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1:40   Min.   :35.23
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:40.02
##  3:10   50 :8                    Median :42.20
##  4:10   75 :8                    Mean   :42.02
##         100:8                    3rd Qu.:43.90
##                                  Max.   :49.88
```

```r
# Número de observações
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##        COM      SEM
## 0   157.48 169.5600
## 25  157.28 164.3600
## 50  166.78 175.2600
## 75  168.56 178.8500
## 100 165.23 177.2533
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##         COM      SEM
## 0   39.3700 42.39000
## 25  39.3200 41.09000
## 50  41.6950 43.81500
## 75  42.1400 44.71250
## 100 41.3075 44.31333
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##           COM         SEM
## 0    0.842400  0.01946667
## 25   9.660333  1.84426667
## 50   2.721900  1.67156667
## 75  15.747133 12.98029167
## 100  7.234292  0.09628889
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0   0.9178235 0.1395230
## 25  3.1081077 1.3580378
## 50  1.6498182 1.2928908
## 75  3.9682658 3.6028172
## 100 2.6896639 0.3103045
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_2,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 42.01533
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 42.90 42.09 42.06 41.01
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 40.88 40.20 42.75 43.43 42.81
##
##  INOCULO
## INOCULO
##   COM   SEM
## 40.77 43.26
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   39.37 42.39
##      25  39.32 41.09
##      50  41.70 43.81
##      75  42.14 44.71
##      100 41.31 44.31
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_2 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_2, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_2$ClorfTotal)$p.value

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

```
## [1] 0.5469774
```

```
# 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_tempo_2$ClorfTotal,
```

```
                                    media_blocos_tempo_2$EFLUENTE,
                                    media_blocos_tempo_2$INOCULO,
                                    media_blocos_tempo_2$CICLO)$p.value

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

```
## [1] 0.9908276
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_2)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  17.97   5.989
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  62.38   62.38
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  31.42   10.47
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4  61.89  15.472   3.405 0.0243 *
## INOCULO:EFLUENTE   4   2.42   0.604   0.133 0.9687
## Residuals         24 109.07   4.544
## ---
## 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_2)
```

```r
# 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: ClorfTotal
##           Df Sum Sq Mean Sq F value   Pr(>F)
## INOCULO    1 62.383  62.383 11.9893 0.001798 **
## EFLUENTE   4 61.890  15.472  2.9736 0.037146 *
## 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)
```

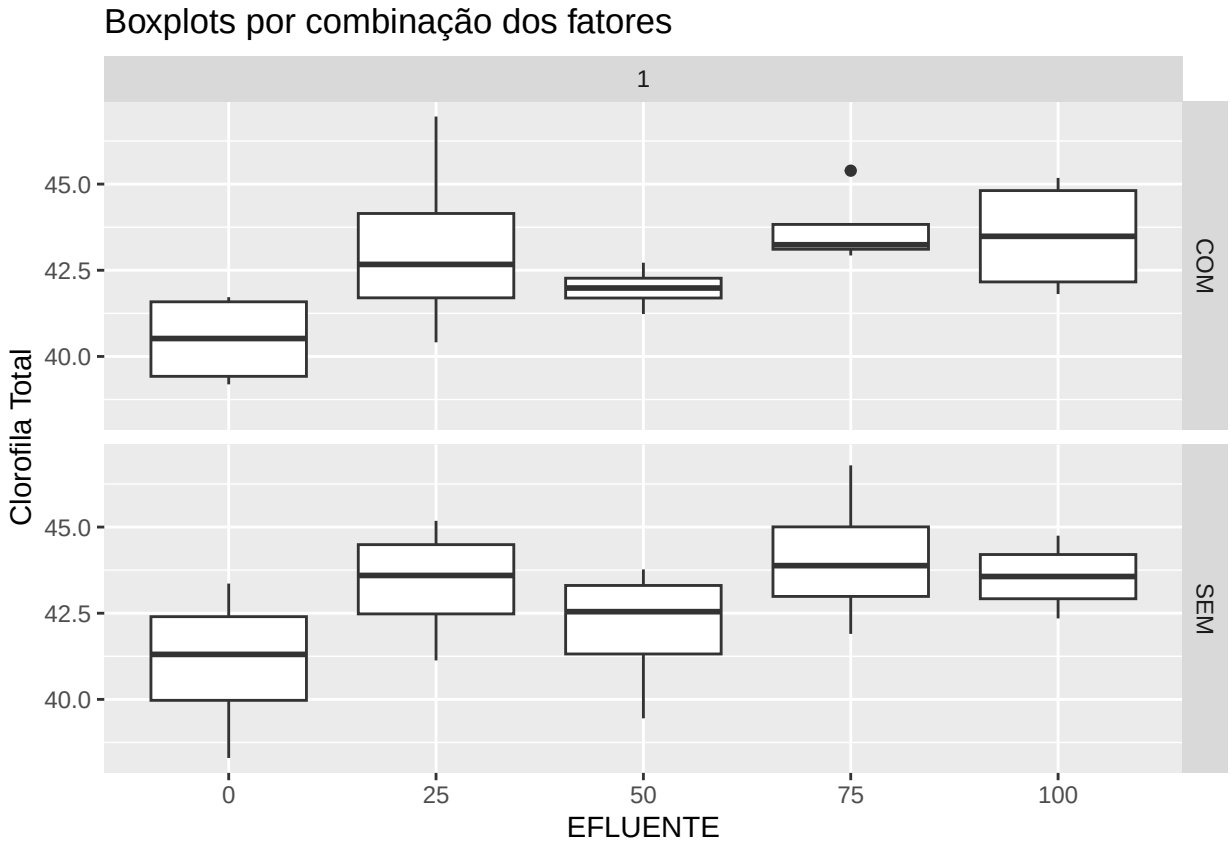
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

```
    media_interacao <- aggregate(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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 52 e 54 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_3, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_3 <- with(dados_tempo_3,
                             dados_tempo_3[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_3$ClorfTotal[which(blocos_dados_tempo_3$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_3$ClorfTotal[which(blocos_dados_tempo_3$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_3 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_3, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_3 = media_blocos_tempo_3[rep(row.names(media_blocos_tempo_3),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_3) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_3$ClorfTotal[which(is.na(blocos_dados_tempo_3$ClorfTotal))] =
  media_blocos_tempo_3$ClorfTotal[which(is.na(blocos_dados_tempo_3$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 41.7 39.5 39.2 41.5 40.4 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1:40   Min.   :38.30
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:41.79
##  3:10   50 :8                    Median :42.83
##  4:10   75 :8                    Mean   :42.65
##         100:8                    3rd Qu.:43.46
##                                  Max.   :46.96
```

```r
# Número de observações
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   161.9500 164.27
## 25  172.7100 173.50
## 50  167.9200 168.31
## 75  172.5467 176.45
## 100 173.9600 174.23
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##          COM     SEM
## 0   40.48750 41.0675
## 25  43.17750 43.3750
## 50  41.98000 42.0775
## 75  43.13667 44.1125
## 100 43.49000 43.5575
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##            COM      SEM
## 0   1.76182500 4.742892
## 25  7.68822500 3.093100
## 50  0.38220000 3.645825
## 75  0.02462222 4.244692
## 100 2.86086667 1.098092
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##           COM      SEM
## 0   1.3273376 2.177818
## 25  2.7727649 1.758721
## 50  0.6182233 1.909404
## 75  0.1569147 2.060265
## 100 1.6914097 1.047899
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_3,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 42.64617
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 41.54 43.32 42.62 43.11
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 40.78 43.28 42.03 43.62 43.52
##
##  INOCULO
## INOCULO
##   COM   SEM
## 42.45 42.84
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   40.49 41.07
##      25  43.18 43.38
##      50  41.98 42.08
##      75  43.14 44.11
##      100 43.49 43.56
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_3 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_3, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_3$ClorfTotal)$p.value

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

```
## [1] 0.2412332
```

```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_tempo_3$ClorfTotal,
```

```
                                    media_blocos_tempo_3$EFLUENTE,
                                    media_blocos_tempo_3$INOCULO,
                                    media_blocos_tempo_3$CICLO)$p.value

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

```
## [1] 0.2303467
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_3)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  18.93    6.31
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  1.472   1.472
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  3.788   1.263
##
## Error: Within
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## EFLUENTE           4  47.98  11.995   4.368 0.00854 **
## INOCULO:EFLUENTE   4   1.21   0.303   0.110 0.97774
## Residuals         24  65.91   2.746
## ---
## 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_3)
```

```r
# 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: ClorfTotal
##           Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE   4 47.981  11.995  4.6468 0.005524 **
## 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)
```

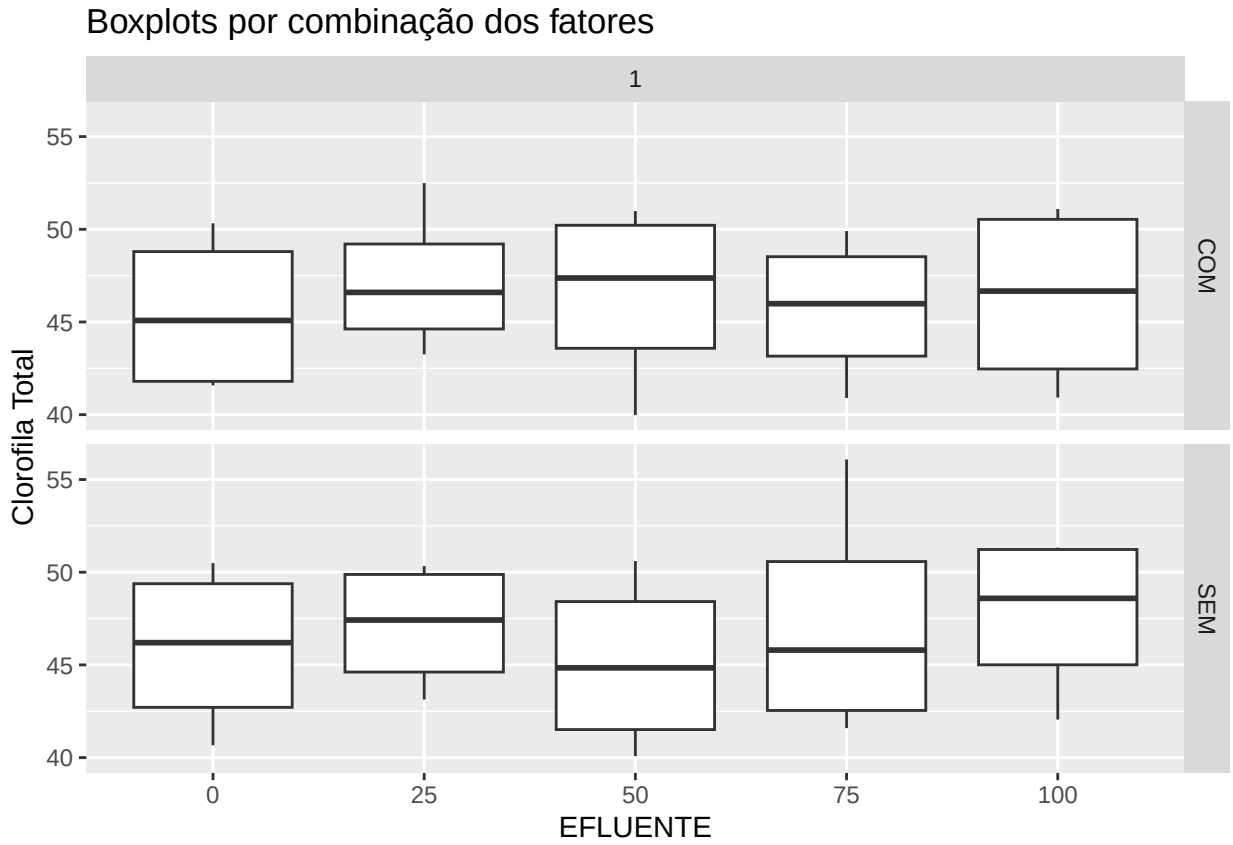
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

```
                              data = blocos_dados_tempo_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 64 e 61 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_4, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_4 <- with(dados_tempo_4,
                             dados_tempo_4[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_4$ClorfTotal[which(blocos_dados_tempo_4$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_4$ClorfTotal[which(blocos_dados_tempo_4$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_4 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_4, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_4 = media_blocos_tempo_4[rep(row.names(media_blocos_tempo_4),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_4) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_4$ClorfTotal[which(is.na(blocos_dados_tempo_4$ClorfTotal))] =
  media_blocos_tempo_4$ClorfTotal[which(is.na(blocos_dados_tempo_4$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 41.6 41.9 48.3 50.3 43.2 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1:40   Min.   :39.98
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:42.66
##  3:10   50 :8                    Median :46.84
##  4:10   75 :8                    Mean   :46.42
##         100:8                    3rd Qu.:50.32
##                                  Max.   :56.08
```

```r
# Número de observações
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##        COM    SEM
## 0   182.06 183.56
## 25  188.93 188.30
## 50  185.70 180.36
## 75  182.77 189.27
## 100 185.34 190.56
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##         COM     SEM
## 0   45.5150 45.8900
## 25  47.2325 47.0750
## 50  46.4250 45.0900
## 75  45.6925 47.3175
## 100 46.3350 47.6400
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##          COM      SEM
## 0   19.85297 21.46427
## 25  16.30163 12.35610
## 50  25.82943 23.94340
## 75  16.48449 43.82616
## 100 26.43617 20.06307
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   4.455667 4.632954
## 25  4.037527 3.515124
## 50  5.082267 4.893199
## 75  4.060110 6.620133
## 100 5.141611 4.479181
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_4,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 46.42125
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 41.42 43.80 50.42 50.06
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 45.70 47.15 45.76 46.51 46.99
##
##  INOCULO
## INOCULO
##   COM   SEM
## 46.24 46.60
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   45.52 45.89
##      25  47.23 47.08
##      50  46.43 45.09
##      75  45.69 47.32
##      100 46.34 47.64
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_4 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_4, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_4$ClorfTotal)$p.value

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

```
## [1] 0.6303785
```

```
# 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_tempo_4$ClorfTotal,
```

```
                                    media_blocos_tempo_4$EFLUENTE,
                                    media_blocos_tempo_4$INOCULO,
                                    media_blocos_tempo_4$CICLO)$p.value

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

```
## [1] 0.4970608
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_4)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  611.4   203.8
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  1.314   1.314
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.9131  0.3044
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  14.57   3.643   1.298  0.299
## INOCULO:EFLUENTE  4  11.27   2.817   1.004  0.425
## Residuals        24  67.35   2.806
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_4)

# Realizar a análise de variância (ANOVA)
```

```r
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: ClorfTotal
##       Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO  3 611.41   203.8  80.605 1.361e-13 ***
## 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(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_4, FUN = mean)
    print(media_interacao)
```

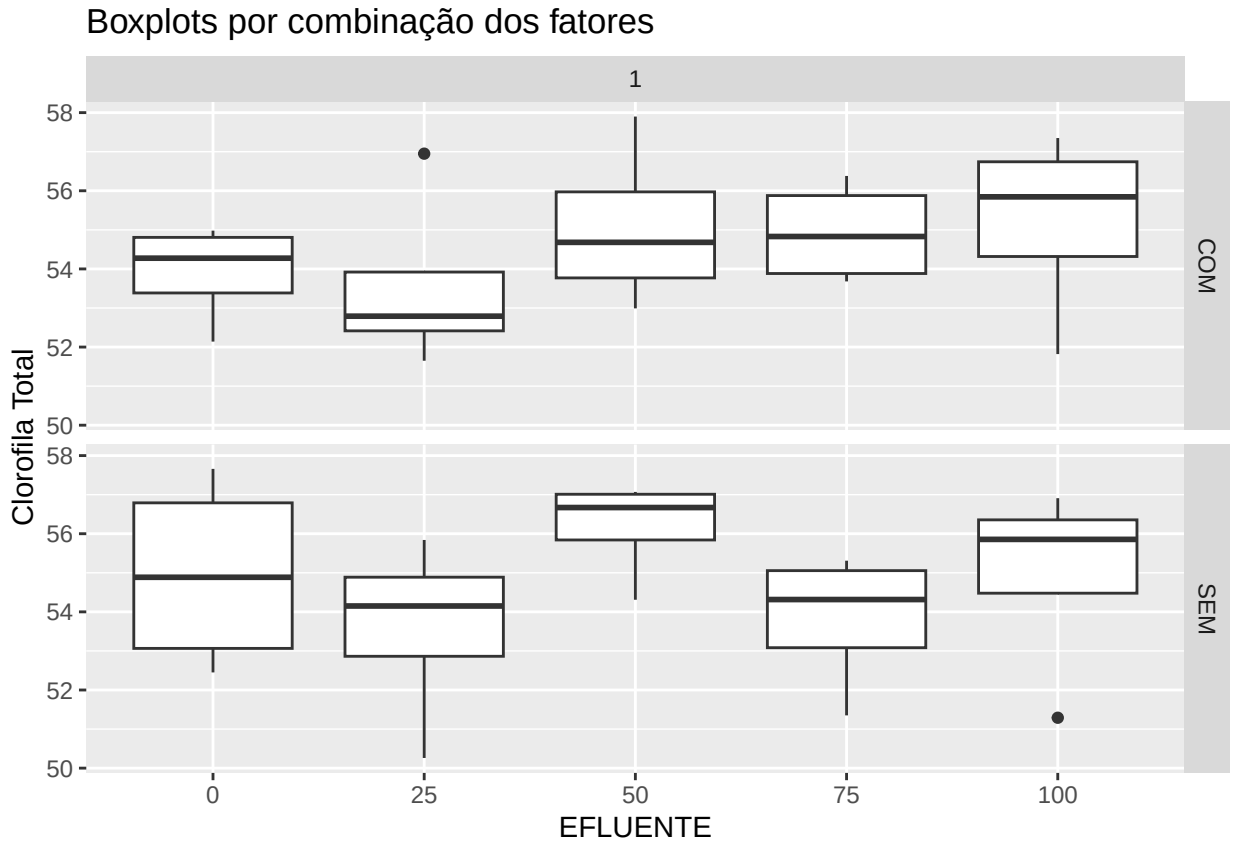
```
  }

}
```

```
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 114 dias

```
# Utilizar apenas o Ciclo 1, pois não foram coletadas amostras do Ciclo 2 para
# 114 dias
dados_tempo_5 = subset(dados_tempo_5, CICLO == 1)
```

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_5, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_5 <- with(dados_tempo_5,
                             dados_tempo_5[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_5$ClorfTotal[which(blocos_dados_tempo_5$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_5$ClorfTotal[which(blocos_dados_tempo_5$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_5 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_5, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_5 = media_blocos_tempo_5[rep(row.names(media_blocos_tempo_5),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_5) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_5$ClorfTotal[which(is.na(blocos_dados_tempo_5$ClorfTotal))] =
  media_blocos_tempo_5$ClorfTotal[which(is.na(blocos_dados_tempo_5$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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 ...
##  $ ClorfTotal: num [1:40] 55 54.8 53.8 52.1 52.4 ...
```

```r
summary(blocos_dados_tempo_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1:40   Min.   :50.26
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:53.20
##  3:10   50 :8                    Median :54.86
##  4:10   75 :8                    Mean   :54.63
##         100:8                    3rd Qu.:56.24
##                                  Max.   :57.90
```

```r
# Número de observações
with(blocos_dados_tempo_5, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_5, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##        COM      SEM
## 0   215.67 219.8800
## 25  209.64 214.4000
## 50  220.25 224.7200
## 75  219.72 215.2900
## 100 220.86 224.8267
```

```r
with(blocos_dados_tempo_5, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##         COM      SEM
## 0   53.9175 54.97000
## 25  52.4100 53.60000
## 50  55.0625 56.18000
## 75  54.9300 53.82250
## 100 55.2150 56.20667
```

```r
with(blocos_dados_tempo_5, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##          COM       SEM
## 0   1.665092 6.2724667
## 25  0.298400 5.7103333
## 50  4.494758 1.6580000
## 75  1.744600 3.2230250
## 100 5.948033 0.3134889
```

```r
with(blocos_dados_tempo_5, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##          COM       SEM
## 0   1.290384 2.5044893
## 25  0.546260 2.3896304
## 50  2.120085 1.2876335
## 75  1.320833 1.7952785
## 100 2.438859 0.5599008
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_5,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                      EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 54.63142
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 55.53 55.33 54.44 53.23
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 54.44 53.01 55.62 54.38 55.71
##
##  INOCULO
## INOCULO
##   COM   SEM
## 54.31 54.96
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   53.92 54.97
##      25  52.41 53.60
##      50  55.06 56.18
##      75  54.93 53.82
##      100 55.22 56.21
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_5 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_5, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_5$ClorfTotal)$p.value

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

```
## [1] 0.569253
```

```
# 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_tempo_5$ClorfTotal,
```

```
                                    media_blocos_tempo_5$EFLUENTE,
                                    media_blocos_tempo_5$INOCULO,
                                    media_blocos_tempo_5$CICLO)$p.value

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

```
## [1] 0.9999125
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_5)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3   32.8   10.93
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1   4.21    4.21
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  3.825   1.275
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4  39.12   9.781   4.092 0.0114 *
## INOCULO:EFLUENTE   4   7.76   1.939   0.811 0.5304
## Residuals         24  57.36   2.390
## ---
## 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_5)
```

```r
# 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: ClorfTotal
##           Df Sum Sq Mean Sq F value   Pr(>F)
## BLOCO      3 32.796 10.9320  4.8239 0.008133 **
## EFLUENTE   4 39.124  9.7809  4.3159 0.007920 **
## 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(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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)
}
```

```
# Mudar nome das colunas dos dados atualizados
blocos_dados_tempo_1$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfTotal
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfTotal
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfTotal
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfTotal
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfTotal

# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_tempo_1, blocos_dados_tempo_2["43 E 41 DAS"],
                    blocos_dados_tempo_3["52 E 54 DAS"],
                    blocos_dados_tempo_4["64 E 61 DAS"],
                    blocos_dados_tempo_5["114 DAS"])


# 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 = "Clorofilômetro ClorfTotal atualizado - Ciclo 1.xlsx",
      sheetName = "R - ClorfTotal_1", append = FALSE)
```

CICLO 2

```
# Leitura e tratamento dos dados
dados <- read_excel("Clorofilômetro ClorfTotal ok.xlsx")

# Excluir colunas irrelevantes para a análise
dados = dados[-c(1:3)]

# 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 5:9) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
```

```
## Warning: NAs introduzidos por coerção
```

```
dados[5:9] =  round(dados[5:9], digits = 2)
str(dados)
```

```
## tibble [80 x 9] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO      : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE   : num [1:80] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO    : chr [1:80] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO      : num [1:80] 1 1 1 1 1 1 1 1 1 1 ...
##  $ 24 E 29 DAS: num [1:80] 45.5 48.7 49.3 52.1 48.6 ...
##  $ 43 E 40 DAS: num [1:80] 39.5 42.2 38.6 42.1 43.9 ...
##  $ 52 E 54 DAS: num [1:80] 41.7 42.1 40.4 41.1 41.2 ...
##  $ 64 E 61 DAS: num [1:80] 41.6 40.7 43.2 43.1 40 ...
##  $ 114 DAS    : num [1:80] 55 53.3 57 55.8 57.9 ...
```

```
# 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_tempo_1 = dados[c(1:5)] # 24, 29
dados_tempo_1$ClorfTotal = dados_tempo_1$`24 E 29 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43, 40
dados_tempo_2$ClorfTotal = dados_tempo_2$`43 E 40 DAS`
```

```
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$ClorfTotal = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfTotal = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfTotal = dados_tempo_5$`114 DAS`
dados_tempo_5 = dados_tempo_5[-5]

# Estrutura dos dados após separados
"dados_tempo_1"
```

```
## [1] "dados_tempo_1"
```

```
str(dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 42.5 42.9 47.4 46.5 43 ...
```

```
"dados_tempo_2"
```

```
## [1] "dados_tempo_2"
```

```
str(dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 71.2 48.5 47.9 49.2 68.3 ...
```

```
"dados_tempo_3"
```

```
## [1] "dados_tempo_3"
```

```
str(dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 43.2 47.6 51.7 59.9 52.8 ...
```

```
"dados_tempo_4"
```

```
## [1] "dados_tempo_4"
```

```
str(dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 47.6 48.3 49.6 48.1 45.9 ...
```

```
"dados_tempo_5"
```

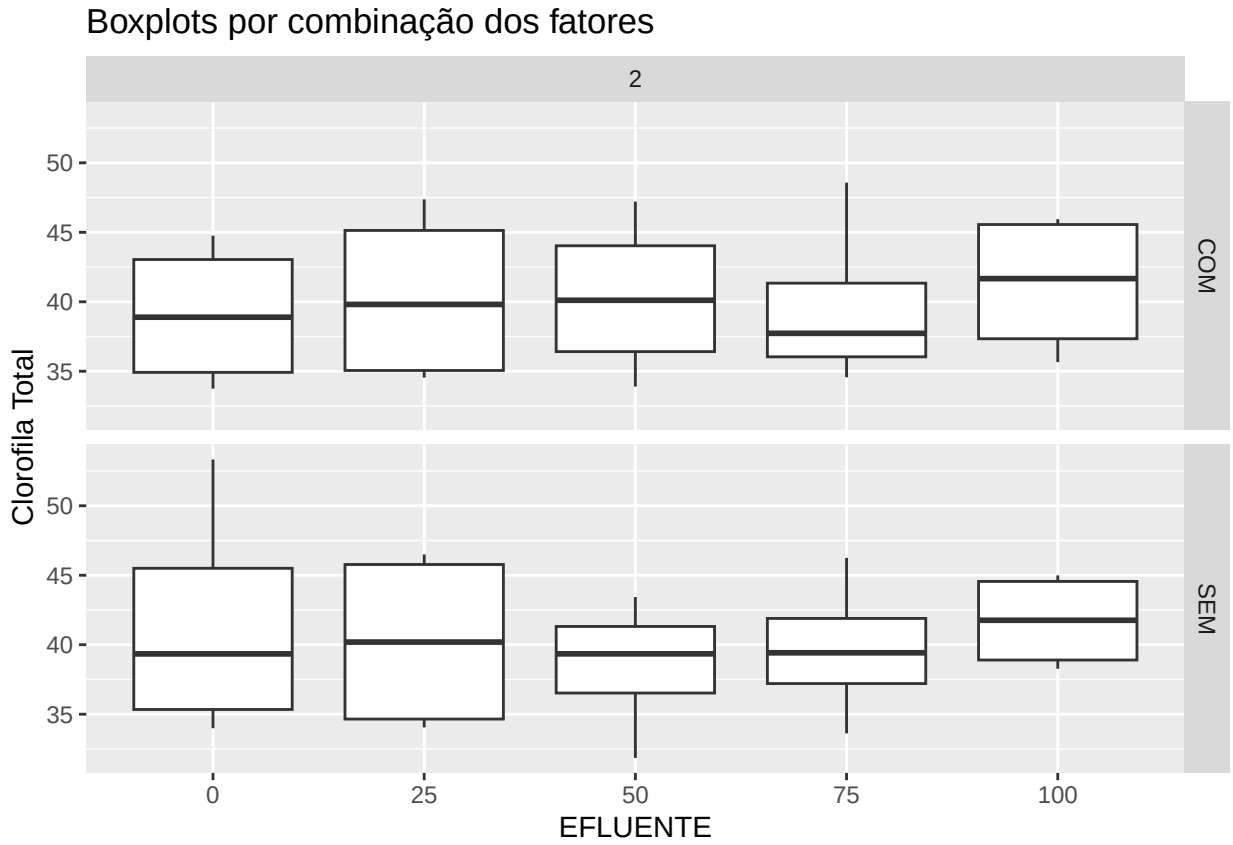
```
## [1] "dados_tempo_5"
```

```
str(dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] NA NA NA NA NA NA NA NA NA NA ...
```

Análise para 24 e 29 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_1, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_1 <- with(dados_tempo_1,
                             dados_tempo_1[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_1$ClorfTotal[which(blocos_dados_tempo_1$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_1$ClorfTotal[which(blocos_dados_tempo_1$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_1 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_1, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_1 = media_blocos_tempo_1[rep(row.names(media_blocos_tempo_1),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_1) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_1$ClorfTotal[which(is.na(blocos_dados_tempo_1$ClorfTotal))] =
  media_blocos_tempo_1$ClorfTotal[which(is.na(blocos_dados_tempo_1$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 42.5 44.8 35.3 33.8 47.4 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1: 0   Min.   :31.85
##  2:10   25 :8    SEM:20   2:40   1st Qu.:35.29
##  3:10   50 :8                    Median :39.02
##  4:10   75 :8                    Mean   :40.23
##         100:8                    3rd Qu.:44.82
##                                  Max.   :53.33
```

```r
# Número de observações
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##        COM    SEM
## 0   156.29 166.00
## 25  161.53 160.93
## 50  161.33 153.97
## 75  158.60 158.71
## 100 164.94 166.78
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##         COM     SEM
## 0   39.0725 41.5000
## 25  40.3825 40.2325
## 50  40.3325 38.4925
## 75  39.6500 39.6775
## 100 41.2350 41.6950
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##          COM      SEM
## 0   28.79202 76.96173
## 25  41.85609 44.84656
## 50  35.04509 24.38522
## 75  38.54187 27.36816
## 100 27.34403 12.21370
```

```r
with(blocos_dados_tempo_1, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   5.365820 8.772784
## 25  6.469628 6.696757
## 50  5.919889 4.938140
## 75  6.208210 5.231459
## 100 5.229152 3.494810
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_1,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 40.227
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 43.25 46.39 36.67 34.60
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 40.29 40.31 39.41 39.66 41.47
##
##  INOCULO
## INOCULO
##   COM   SEM
## 40.13 40.32
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   39.07 41.50
##      25  40.38 40.23
##      50  40.33 38.49
##      75  39.65 39.68
##      100 41.24 41.70
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_1 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_1, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_1$ClorfTotal)$p.value

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

```
## [1] 0.7748954
```

```
# 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_tempo_1$ClorfTotal,
```

```
                                    media_blocos_tempo_1$EFLUENTE,
                                    media_blocos_tempo_1$INOCULO,
                                    media_blocos_tempo_1$CICLO)$p.value

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

```
## [1] 0.05834993
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_1)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  914.8   304.9
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.3422  0.3422
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.092  0.6974
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  20.19   5.047   0.781  0.549
## INOCULO:EFLUENTE  4  18.68   4.671   0.723  0.585
## Residuals        24 155.13   6.464
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_1)

# Realizar a análise de variância (ANOVA)
```

```r
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: ClorfTotal
##       Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO  3 914.84  304.95  52.368 2.201e-11 ***
## 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)
```

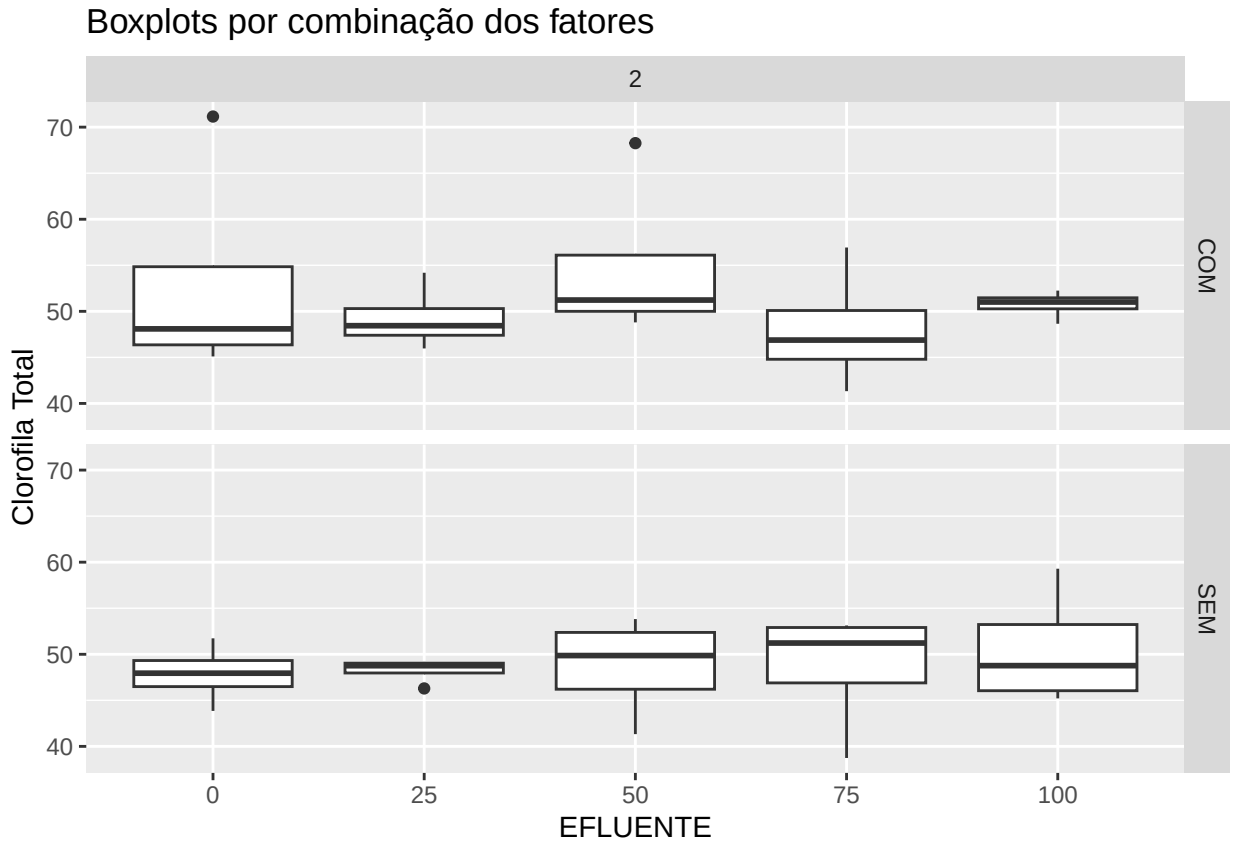
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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 43 e 40 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_2, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_2 <- with(dados_tempo_2,
                             dados_tempo_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_2$ClorfTotal[which(blocos_dados_tempo_2$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_2$ClorfTotal[which(blocos_dados_tempo_2$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_2 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_2, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_2 = media_blocos_tempo_2[rep(row.names(media_blocos_tempo_2),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_2) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_2$ClorfTotal[which(is.na(blocos_dados_tempo_2$ClorfTotal))] =
  media_blocos_tempo_2$ClorfTotal[which(is.na(blocos_dados_tempo_2$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 47.1 49.4 45.1 46.8 47.9 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1: 0   Min.   :38.74
##  2:10   25 :8    SEM:20   2:40   1st Qu.:46.66
##  3:10   50 :8                    Median :48.90
##  4:10   75 :8                    Mean   :48.94
##         100:8                    3rd Qu.:51.34
##                                  Max.   :59.29
```

```r
# Número de observações
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   188.3867 191.47
## 25  197.0400 192.98
## 50  201.6667 194.87
## 75  192.0200 194.31
## 100 202.9200 202.03
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##          COM     SEM
## 0   47.09667 47.8675
## 25  49.26000 48.2450
## 50  50.41667 48.7175
## 75  48.00500 48.5775
## 100 50.73000 50.5075
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##           COM      SEM
## 0    3.146156 10.57749
## 25  12.367000  1.77130
## 50   1.760556 30.48769
## 75  42.824100 45.55049
## 100  2.291200 41.07482
```

```r
with(blocos_dados_tempo_2, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   1.773741 3.252306
## 25  3.516675 1.330902
## 50  1.326859 5.521566
## 75  6.544013 6.749110
## 100 1.513671 6.408964
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_2,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 48.94233
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 48.81 46.99 48.71 51.27
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 47.48 48.75 49.57 48.29 50.62
##
##  INOCULO
## INOCULO
##   COM   SEM
## 49.10 48.78
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   47.10 47.87
##      25  49.26 48.25
##      50  50.42 48.72
##      75  48.01 48.58
##      100 50.73 50.51
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_2 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_2, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_2$ClorfTotal)$p.value

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

```
## [1] 0.3896954
```

```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_tempo_2$ClorfTotal,
```

```
                                    media_blocos_tempo_2$EFLUENTE,
                                    media_blocos_tempo_2$INOCULO,
                                    media_blocos_tempo_2$CICLO)$p.value

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

```
## [1] 0.6524238
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_2)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  92.97   30.99
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  1.016   1.016
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  7.814   2.605
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4   46.3  11.586   0.586  0.676
## INOCULO:EFLUENTE  4    8.8   2.191   0.111  0.978
## Residuals        24  474.8  19.782
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_2)

# Realizar a análise de variância (ANOVA)
```

```r
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: ClorfTotal
##    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)
```

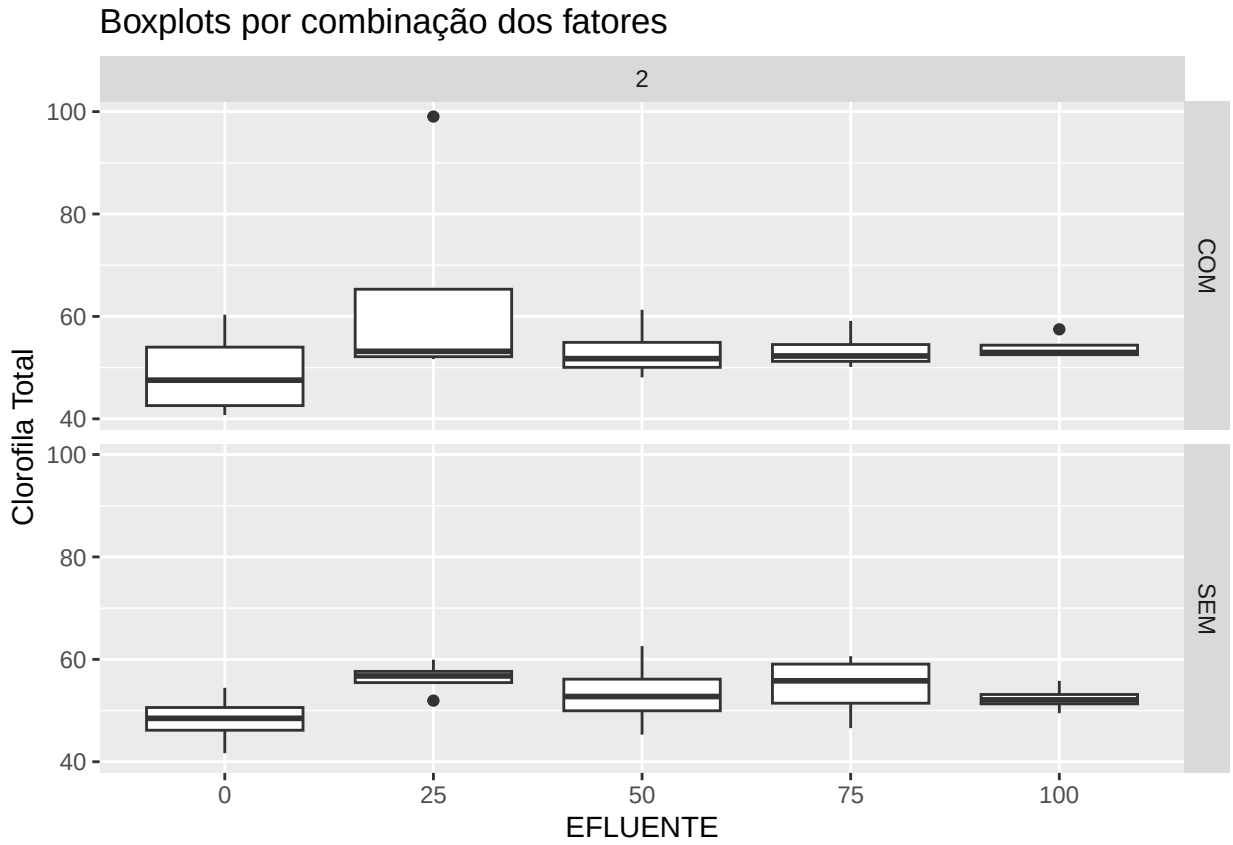
```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```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(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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 52 e 54 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_3, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_3 <- with(dados_tempo_3,
                             dados_tempo_3[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_3$ClorfTotal[which(blocos_dados_tempo_3$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_3$ClorfTotal[which(blocos_dados_tempo_3$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_3 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_3, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_3 = media_blocos_tempo_3[rep(row.names(media_blocos_tempo_3),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_3) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_3$ClorfTotal[which(is.na(blocos_dados_tempo_3$ClorfTotal))] =
  media_blocos_tempo_3$ClorfTotal[which(is.na(blocos_dados_tempo_3$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 43.2 60.3 51.9 40.7 51.7 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1: 0   Min.   :40.74
##  2:10   25 :8    SEM:20   2:40   1st Qu.:50.56
##  3:10   50 :8                    Median :52.61
##  4:10   75 :8                    Mean   :52.77
##         100:8                    3rd Qu.:56.00
##                                  Max.   :62.59
```

```r
# Número de observações
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   196.1600 193.0900
## 25  210.7067 231.2667
## 50  212.9000 213.3800
## 75  213.8100 218.8000
## 100 211.1600 209.4500
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   49.04000 48.27250
## 25  52.67667 57.81667
## 50  53.22500 53.34500
## 75  53.45250 54.70000
## 100 52.79000 52.36250
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##            COM       SEM
## 0   79.5274000 27.695492
## 25   1.0347556  2.244356
## 50  32.6192333 51.279500
## 75  15.6614917 39.551800
## 100  0.1578667  6.735825
```

```r
with(blocos_dados_tempo_3, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##          COM      SEM
## 0   8.9178136 5.262651
## 25  1.0172294 1.498117
## 50  5.7113250 7.160971
## 75  3.9574603 6.289022
## 100 0.3973244 2.595347
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_3,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 52.76808
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 50.55 55.20 52.56 52.77
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 48.66 55.25 53.29 54.08 52.58
##
##  INOCULO
## INOCULO
##   COM   SEM
## 52.24 53.30
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   49.04 48.27
##      25  52.68 57.82
##      50  53.23 53.35
##      75  53.45 54.70
##      100 52.79 52.36
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_3 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_3, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_3$ClorfTotal)$p.value

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

```
## [1] 0.2955335
```

```
# 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_tempo_3$ClorfTotal,
```

```
                                    media_blocos_tempo_3$EFLUENTE,
                                    media_blocos_tempo_3$INOCULO,
                                    media_blocos_tempo_3$CICLO)$p.value

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

```
## [1] 0.08401076
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_3)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  108.6   36.19
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  11.29   11.29
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  5.618   1.873
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  200.5   50.13   1.836  0.155
## INOCULO:EFLUENTE  4   46.2   11.56   0.423  0.790
## Residuals        24  655.3   27.31
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_3)

# Realizar a análise de variância (ANOVA)
```

```r
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: ClorfTotal
##    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)
```

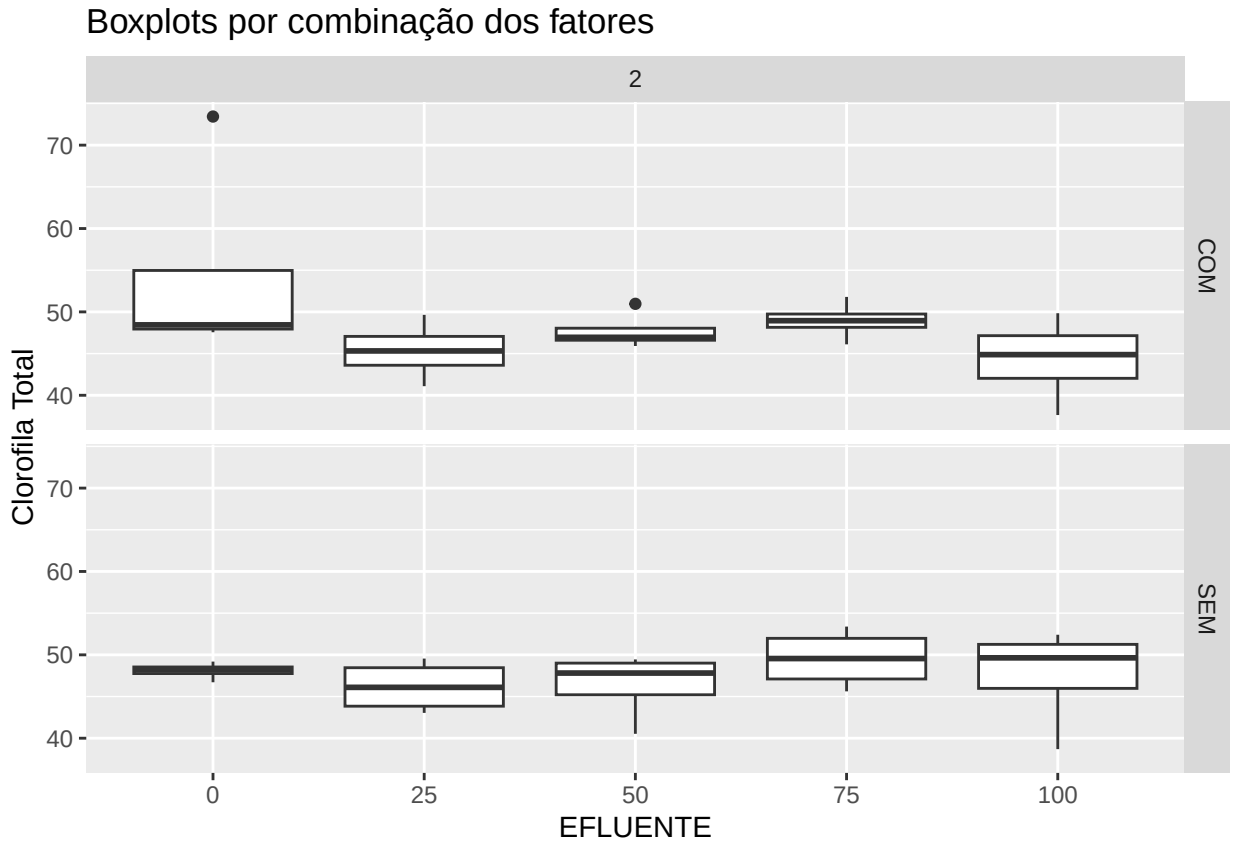
```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

```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(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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 64 e 61 dias

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_4, aes(x = factor(EFLUENTE), y = ClorfTotal)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila Total") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_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$ClorfTotal

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

# 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_tempo_4 <- with(dados_tempo_4,
                             dados_tempo_4[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_tempo_4$ClorfTotal[which(blocos_dados_tempo_4$ClorfTotal <
                                      limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_4$ClorfTotal[which(blocos_dados_tempo_4$ClorfTotal >
                                      limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_4 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_4, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_tempo_4 = media_blocos_tempo_4[rep(row.names(media_blocos_tempo_4),
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_tempo_4) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_tempo_4$ClorfTotal[which(is.na(blocos_dados_tempo_4$ClorfTotal))] =
  media_blocos_tempo_4$ClorfTotal[which(is.na(blocos_dados_tempo_4$ClorfTotal))]
```

```r
# Análises Descritivas
str(blocos_dados_tempo_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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfTotal: num [1:40] 47.6 48.1 48.8 48.1 49.6 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO    ClorfTotal
##  1:10   0  :8    COM:20   1: 0   Min.   :37.63
##  2:10   25 :8    SEM:20   2:40   1st Qu.:46.06
##  3:10   50 :8                    Median :47.84
##  4:10   75 :8                    Mean   :47.12
##         100:8                    3rd Qu.:49.09
##                                  Max.   :53.39
```

```r
# Número de observações
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), length))
```

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

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   192.6000 192.33
## 25  181.3800 184.79
## 50  186.4133 185.61
## 75  195.8000 198.12
## 100 177.2200 190.38
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), mean))
```

```
##          COM     SEM
## 0   48.15000 48.0825
## 25  45.34500 46.1975
## 50  46.60333 46.4025
## 75  48.95000 49.5300
## 100 44.30500 47.5950
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), var))
```

```
##            COM       SEM
## 0    0.2678000  1.057292
## 25  12.7293667  9.716092
## 50   0.2499556 16.649292
## 75   5.4048667 12.611000
## 100 26.5409667 37.957700
```

```r
with(blocos_dados_tempo_4, tapply(ClorfTotal, list(EFLUENTE, INOCULO), sd))
```

```
##           COM      SEM
## 0   0.5174940 1.028247
## 25  3.5678238 3.117065
## 50  0.4999556 4.080354
## 75  2.3248369 3.551197
## 100 5.1517926 6.160982
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_4,
                 model.tables(aov(ClorfTotal ~ BLOCO + EFLUENTE + INOCULO +
                                    EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 47.11608
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 47.48 47.61 44.81 48.56
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 48.12 45.77 46.50 49.24 45.95
##
##  INOCULO
## INOCULO
##   COM   SEM
## 46.67 47.56
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   48.15 48.08
##      25  45.35 46.20
##      50  46.60 46.40
##      75  48.95 49.53
##      100 44.31 47.60
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_4 = aggregate(ClorfTotal ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_4, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_tempo_4$ClorfTotal)$p.value

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

```
## [1] 0.9417242
```

```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_tempo_4$ClorfTotal,
```

```
                                    media_blocos_tempo_4$EFLUENTE,
                                    media_blocos_tempo_4$INOCULO,
                                    media_blocos_tempo_4$CICLO)$p.value

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

```
## [1] 0.07322374
```

```
# 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(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_tempo_4)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3  77.68   25.89
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  7.936   7.936
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  3.669   1.223
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  72.45  18.111   1.508  0.231
## INOCULO:EFLUENTE  4  15.93   3.982   0.332  0.854
## Residuals        24 288.21  12.009
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfTotal ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_4)

# Realizar a análise de variância (ANOVA)
```

```r
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: ClorfTotal
##    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(ClorfTotal ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_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)
}
```

```
# Mudar nome das colunas dos dados atualizados
blocos_dados_tempo_1$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfTotal
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfTotal
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfTotal
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfTotal
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfTotal

# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_tempo_1, blocos_dados_tempo_2["43 E 41 DAS"],
                    blocos_dados_tempo_3["52 E 54 DAS"],
                    blocos_dados_tempo_4["64 E 61 DAS"],
                    blocos_dados_tempo_5["114 DAS"])


# Criar planilha com todos os dados atualizados
library("xlsx")
write.xlsx(dados_final, file = "Clorofilômetro ClorfTotal atualizado - Ciclo 2.xlsx",
      sheetName = "R - ClorfTotal_2", append = FALSE)
```