Clorofilômetro Clorofila A

Ana Carolina Murad Lima

2023-06-22

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

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

```
library(dplyr)
```

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

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

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

```
library(ggplot2)
```

CICLO 1

```
# Leitura e tratamento dos dados
dados <- read_excel("Clorofilômetro ClorfA 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] 38.9 40.9 41.5 43.4 40.8 ...
##  $ 43 E 40 DAS: num [1:80] 31.7 36 30.7 35.8 35 ...
##  $ 52 E 54 DAS: num [1:80] 35.9 36.2 34.8 35.2 35.4 ...
##  $ 64 E 61 DAS: num [1:80] 35.8 35 37 36.9 34.3 ...
##  $ 114 DAS    : num [1:80] 42.8 41.6 43.7 43.1 44.4 ...
```

```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$ClorfA = dados_tempo_1$`24 E 29 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43, 40
dados_tempo_2$ClorfA = dados_tempo_2$`43 E 40 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$ClorfA = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfA = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfA = dados_tempo_5$`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 ...
##  $ ClorfA  : num [1:40] 38.9 40.9 41.5 43.4 40.8 ...
```

```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 ...
##  $ ClorfA  : num [1:40] 31.7 36 30.7 35.8 35 ...
```

```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 ...
##  $ ClorfA  : num [1:40] 35.9 36.2 34.8 35.2 35.4 ...
```

```
"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 ...
##  $ ClorfA  : num [1:40] 35.8 35 37 36.9 34.3 ...
```

```
"dados_tempo_5"
```

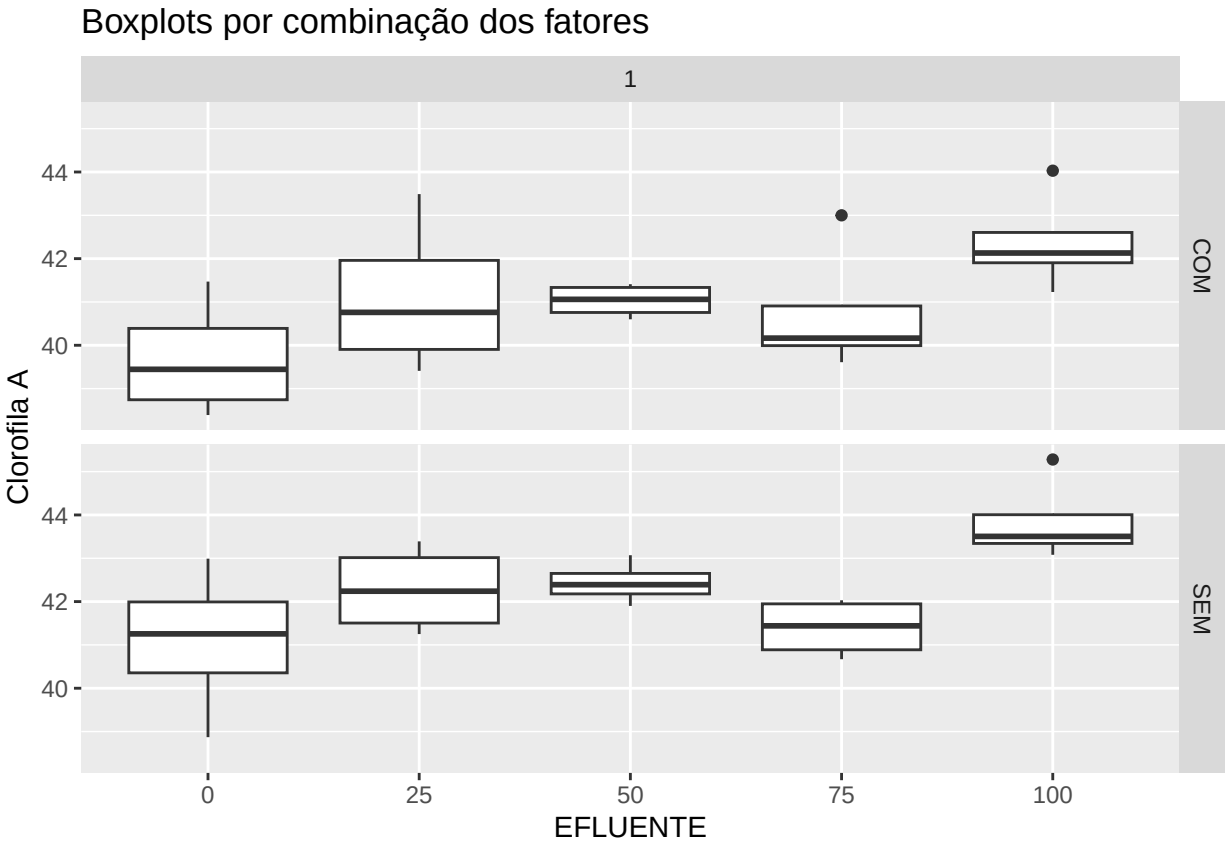
```
## [1] "dados_tempo_5"
```

```
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 ...
##  $ ClorfA  : num [1:40] 42.8 41.6 43.7 43.1 44.4 ...
```

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




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

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

# 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$ClorfA[which(blocos_dados_tempo_1$ClorfA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_1$ClorfA[which(blocos_dados_tempo_1$ClorfA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_1 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_1$ClorfA))] =
  media_blocos_tempo_1$ClorfA[which(is.na(blocos_dados_tempo_1$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 38.9 41.5 38.4 40 41.5 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1:40   Min.   :38.39
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:40.50
##  3:10   50 :8                    Median :41.46
##  4:10   75 :8                    Mean   :41.42
##         100:8                    3rd Qu.:42.33
##                                  Max.   :43.58
```

```r
# Número de observações
with(blocos_dados_tempo_1, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##        COM      SEM
## 0   158.75 164.3700
## 25  164.42 169.1200
## 50  164.13 169.7500
## 75  159.92 165.5800
## 100 167.32 173.4533
```

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

```
##         COM      SEM
## 0   39.6875 41.09250
## 25  41.1050 42.28000
## 50  41.0325 42.43750
## 75  39.9800 41.39500
## 100 41.8300 43.36333
```

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

```
##          COM        SEM
## 0   1.887625 2.97362500
## 25  3.250500 1.04706667
## 50  0.152025 0.24075833
## 75  0.069800 0.46456667
## 100 0.180000 0.04388889
```

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

```
##           COM       SEM
## 0   1.3739087 1.7244202
## 25  1.8029143 1.0232628
## 50  0.3899038 0.4906713
## 75  0.2641969 0.6815913
## 100 0.4242641 0.2094968
```

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

```
## Tables of means
## Grand mean
##
## 41.42033
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 41.32 42.23 40.96 41.17
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 40.39 41.69 41.73 40.69 42.60
##
##  INOCULO
## INOCULO
##   COM   SEM
## 40.73 42.11
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   39.69 41.09
##      25  41.10 42.28
##      50  41.03 42.44
##      75  39.98 41.39
##      100 41.83 43.36
```

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

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

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

```
## [1] 0.8771562
```

```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_1$ClorfA,
```

```
                                    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] 0.9996721
```

```
# 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(ClorfA ~ 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  9.445   3.148
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  19.23   19.23
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.7085  0.2362
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4 25.244   6.311   7.290 0.000543 ***
## INOCULO:EFLUENTE   4  0.136   0.034   0.039 0.996873
## Residuals         24 20.776   0.866
## ---
## 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(ClorfA ~ 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: ClorfA
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3  9.4454  3.1485  3.9568 0.0184592 *
## INOCULO    1 19.2284 19.2284 24.1651 3.819e-05 ***
## EFLUENTE   4 25.2439  6.3110  7.9312 0.0002307 ***
## 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)
```

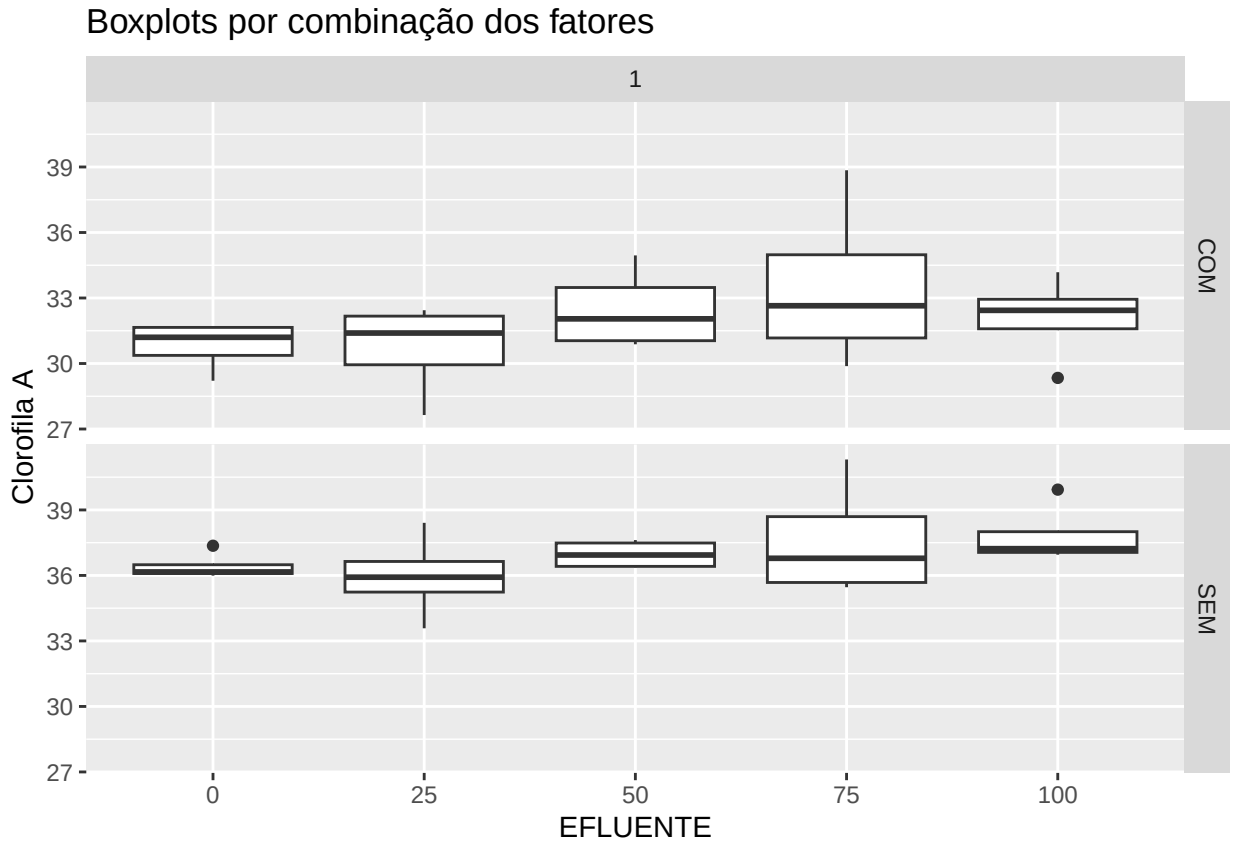
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

```
  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfA ~ 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# 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$ClorfA[which(blocos_dados_tempo_2$ClorfA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_2$ClorfA[which(blocos_dados_tempo_2$ClorfA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_2 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_2$ClorfA))] =
  media_blocos_tempo_2$ClorfA[which(is.na(blocos_dados_tempo_2$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 31.7 31.6 29.2 30.8 30.7 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1:40   Min.   :27.64
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:31.98
##  3:10   50 :8                    Median :35.20
##  4:10   75 :8                    Mean   :34.43
##         100:8                    3rd Qu.:36.56
##                                  Max.   :41.31
```

```r
# Número de observações
with(blocos_dados_tempo_2, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   123.3000 144.4133
## 25  122.8700 143.8300
## 50  129.9200 147.8600
## 75  134.0200 150.3400
## 100 132.0667 148.5333
```

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

```
##          COM      SEM
## 0   30.82500 36.10333
## 25  30.71750 35.95750
## 50  32.48000 36.96500
## 75  33.50500 37.58500
## 100 33.01667 37.13333
```

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

```
##            COM         SEM
## 0    1.3416333 0.007488889
## 25   4.7648250 3.901291667
## 50   3.6084667 0.431633333
## 75  15.1243000 7.271233333
## 100  0.6826889 0.028955556
```

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

```
##           COM        SEM
## 0   1.1582890 0.08653837
## 25  2.1828479 1.97516877
## 50  1.8995964 0.65698808
## 75  3.8889973 2.69652245
## 100 0.8262499 0.17016332
```

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

```
## Tables of means
## Grand mean
##
## 34.42883
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 35.34 34.50 34.76 33.12
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 33.46 33.34 34.72 35.54 35.07
##
##  INOCULO
## INOCULO
##   COM   SEM
## 32.11 36.75
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   30.82 36.10
##      25  30.72 35.96
##      50  32.48 36.96
##      75  33.50 37.58
##      100 33.02 37.13
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_2 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.1807248
```

```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$ClorfA,
```

```
                                    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] 0.9989994
```

```
# 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(ClorfA ~ 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  26.58    8.86
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  215.3   215.3
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  26.76   8.921
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  30.97   7.742   3.196 0.0308 *
## INOCULO:EFLUENTE  4   2.76   0.689   0.285 0.8851
## Residuals        24  58.14   2.423
## ---
## 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(ClorfA ~ 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: ClorfA
##          Df Sum Sq Mean Sq F value    Pr(>F)
## INOCULO   1  215.3   215.3  68.463 6.988e-09 ***
## 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)
```

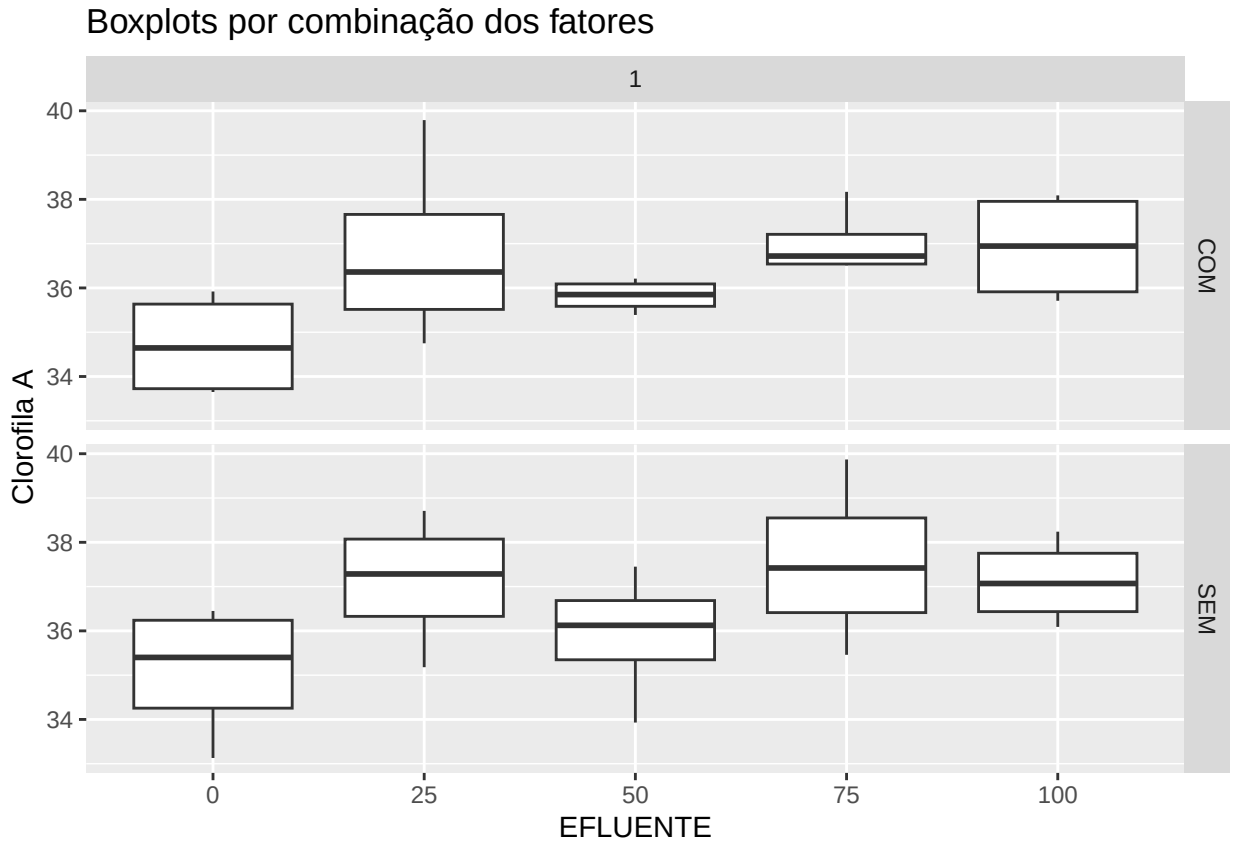
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfA ~ 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

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

# 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$ClorfA[which(blocos_dados_tempo_3$ClorfA <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_3$ClorfA[which(blocos_dados_tempo_3$ClorfA >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_3 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_3$ClorfA))] =
  media_blocos_tempo_3$ClorfA[which(is.na(blocos_dados_tempo_3$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 35.9 33.8 33.6 35.5 34.8 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1:40   Min.   :33.13
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:35.62
##  3:10   50 :8                    Median :36.32
##  4:10   75 :8                    Mean   :36.41
##         100:8                    3rd Qu.:37.48
##                                  Max.   :39.87
```

```r
# Número de observações
with(blocos_dados_tempo_3, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##        COM    SEM
## 0   138.86 140.38
## 25  147.26 148.46
## 50  143.30 143.63
## 75  148.12 150.17
## 100 147.69 148.47
```

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

```
##         COM     SEM
## 0   34.7150 35.0950
## 25  36.8150 37.1150
## 50  35.8250 35.9075
## 75  37.0300 37.5425
## 100 36.9225 37.1175
```

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

```
##           COM       SEM
## 0   1.3993667 2.3563667
## 25  4.7417000 2.3357667
## 50  0.1395667 2.1901583
## 75  0.6066667 3.5787583
## 100 1.5655583 0.9536917
```

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

```
##           COM       SEM
## 0   1.1829483 1.5350461
## 25  2.1775445 1.5283215
## 50  0.3735862 1.4799184
## 75  0.7788881 1.8917606
## 100 1.2512227 0.9765714
```

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

```
## Tables of means
## Grand mean
##
## 36.4085
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 35.55 36.95 36.53 36.61
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 34.91 36.97 35.87 37.29 37.02
##
##  INOCULO
## INOCULO
##   COM   SEM
## 36.26 36.56
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   34.72 35.10
##      25  36.82 37.12
##      50  35.83 35.91
##      75  37.03 37.54
##      100 36.92 37.12
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_3 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.1434538
```

```
# 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$ClorfA,
```

```
                                    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] 0.7319784
```

```
# 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(ClorfA ~ 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  10.86   3.619
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1 0.8644  0.8644
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.009  0.6695
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  32.07   8.017   4.117 0.0111 *
## INOCULO:EFLUENTE  4   0.22   0.055   0.028 0.9984
## Residuals        24  46.74   1.947
## ---
## 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(ClorfA ~ 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: ClorfA
##           Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE   4 32.069  8.0172  4.4406 0.006907 **
## 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)
```

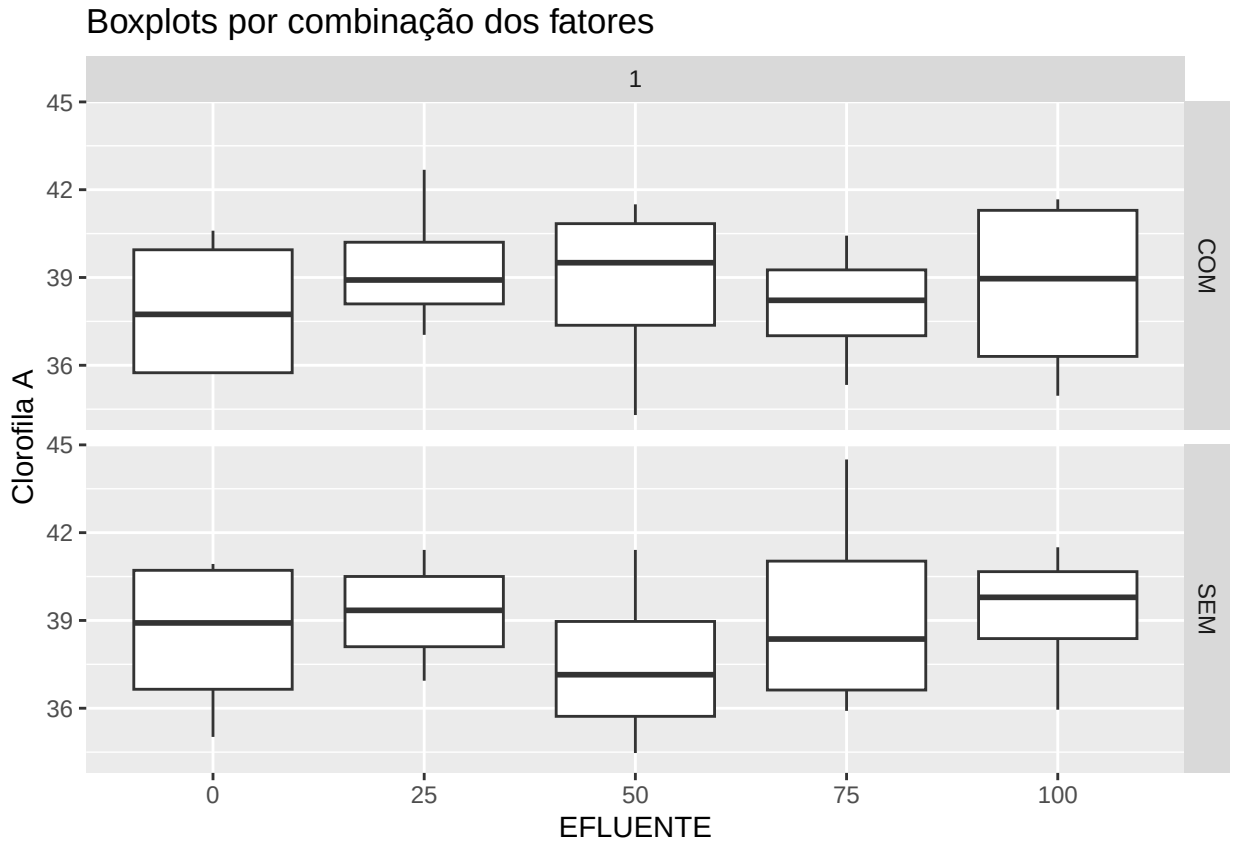
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfA ~ 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

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

# 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$ClorfA[which(blocos_dados_tempo_4$ClorfA <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_4$ClorfA[which(blocos_dados_tempo_4$ClorfA >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_4 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_4$ClorfA))] =
  media_blocos_tempo_4$ClorfA[which(is.na(blocos_dados_tempo_4$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 35.8 35.7 39.7 40.6 37 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1:40   Min.   :34.30
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:36.60
##  3:10   50 :8                    Median :38.68
##  4:10   75 :8                    Mean   :38.65
##         100:8                    3rd Qu.:40.62
##                                  Max.   :44.50
```

```r
# Número de observações
with(blocos_dados_tempo_4, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##        COM    SEM
## 0   151.81 153.78
## 25  157.55 157.04
## 50  154.81 150.17
## 75  152.20 157.14
## 100 154.55 157.03
```

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

```
##         COM     SEM
## 0   37.9525 38.4450
## 25  39.3875 39.2600
## 50  38.7025 37.5425
## 75  38.0500 39.2850
## 100 38.6375 39.2575
```

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

```
##           COM       SEM
## 0    6.653092  8.099633
## 25   5.743425  3.827133
## 50  10.327492  8.911292
## 75   4.655200 14.936567
## 100 10.898758  5.751825
```

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

```
##          COM      SEM
## 0   2.579359 2.845985
## 25  2.396544 1.956306
## 50  3.213641 2.985179
## 75  2.157591 3.864785
## 100 3.301327 2.398296
```

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

```
## Tables of means
## Grand mean
##
## 38.652
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 35.57 37.48 40.87 40.70
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 38.20 39.32 38.12 38.67 38.95
##
##  INOCULO
## INOCULO
##   COM   SEM
## 38.55 38.76
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   37.95 38.44
##      25  39.39 39.26
##      50  38.70 37.54
##      75  38.05 39.28
##      100 38.64 39.26
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_4 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.2645879
```

```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$ClorfA,
```

```
                                    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] 0.5808453
```

```
# 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(ClorfA ~ 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  199.9   66.64
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1 0.4494  0.4494
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.02934 0.00978
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4   8.20   2.049   1.246  0.318
## INOCULO:EFLUENTE  4   6.58   1.645   1.000  0.427
## Residuals        24  39.46   1.644
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfA ~ 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: ClorfA
##       Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO  3 199.92  66.641  45.562 1.065e-10 ***
## 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)
```

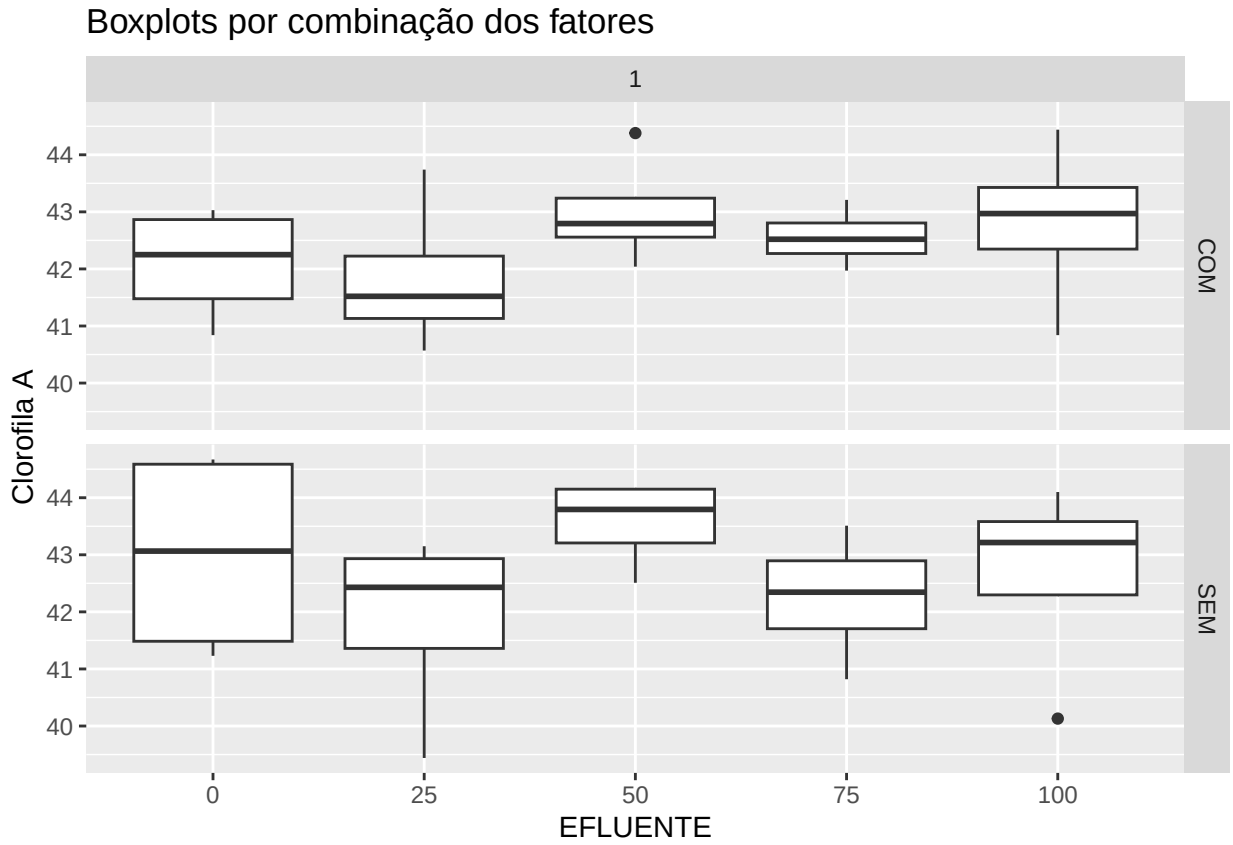
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

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




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

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

# 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$ClorfA[which(blocos_dados_tempo_5$ClorfA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_5$ClorfA[which(blocos_dados_tempo_5$ClorfA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_5 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_5$ClorfA))] =
  media_blocos_tempo_5$ClorfA[which(is.na(blocos_dados_tempo_5$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 42.8 43 41.7 40.8 43.7 ...
```

```r
summary(blocos_dados_tempo_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1:40   Min.   :39.44
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:41.91
##  3:10   50 :8                    Median :42.77
##  4:10   75 :8                    Mean   :42.60
##         100:8                    3rd Qu.:43.42
##                                  Max.   :44.67
```

```r
# Número de observações
with(blocos_dados_tempo_5, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   168.3700 172.03
## 25  167.3500 167.45
## 50  170.1733 174.25
## 75  170.2200 169.02
## 100 171.2200 174.04
```

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

```
##          COM     SEM
## 0   42.09250 43.0075
## 25  41.83750 41.8625
## 50  42.54333 43.5625
## 75  42.55500 42.2550
## 100 42.80500 43.5100
```

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

```
##           COM       SEM
## 0   1.0414917 3.4666917
## 25  1.8358917 2.8466917
## 50  0.1294889 0.6043583
## 75  0.2729000 1.2961667
## 100 2.2059000 0.1994000
```

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

```
##           COM       SEM
## 0   1.0205350 1.8619054
## 25  1.3549508 1.6872142
## 50  0.3598456 0.7774049
## 75  0.5223983 1.1384932
## 100 1.4852273 0.4465423
```

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

```
## Tables of means
## Grand mean
##
## 42.60308
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 42.92 43.28 42.42 41.79
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 42.55 41.85 43.05 42.40 43.16
##
##  INOCULO
## INOCULO
##   COM   SEM
## 42.37 42.84
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   42.09 43.01
##      25  41.84 41.86
##      50  42.54 43.56
##      75  42.56 42.25
##      100 42.80 43.51
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_5 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.442341
```

```
# 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$ClorfA,
```

```
                                    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] 0.2810962
```

```
# 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(ClorfA ~ 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  12.55   4.185
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  2.236   2.236
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.7817  0.2606
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  8.951  2.2378   1.894  0.144
## INOCULO:EFLUENTE  4  2.691  0.6729   0.569  0.687
## Residuals        24 28.360  1.1817
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfA ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_5)

# 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: ClorfA
##       Df Sum Sq Mean Sq F value  Pr(>F)
## BLOCO  3 12.555  4.1851  3.8775 0.01994 *
## 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(ClorfA ~ 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$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfA
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfA
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfA
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfA
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfA

# 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 ClorfA atualizado - Ciclo 1.xlsx",
      sheetName = "R - ClorfA_1", append = FALSE)
```

CICLO 2

```
# Leitura e tratamento dos dados
dados <- read_excel("Clorofilômetro ClorfA 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] 38.9 40.9 41.5 43.4 40.8 ...
##  $ 43 E 40 DAS: num [1:80] 31.7 36 30.7 35.8 35 ...
##  $ 52 E 54 DAS: num [1:80] 35.9 36.2 34.8 35.2 35.4 ...
##  $ 64 E 61 DAS: num [1:80] 35.8 35 37 36.9 34.3 ...
##  $ 114 DAS    : num [1:80] 42.8 41.6 43.7 43.1 44.4 ...
```

```
# 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$ClorfA = dados_tempo_1$`24 E 29 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43, 40
dados_tempo_2$ClorfA = dados_tempo_2$`43 E 40 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$ClorfA = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
```

```r
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfA = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfA = dados_tempo_5$`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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfA  : num [1:40] 35.4 35.2 39.2 38.4 35 ...
```

```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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfA  : num [1:40] 48.1 39.6 39.8 40.8 47.6 ...
```

```r
"dados_tempo_3"
```

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

```r
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 ...
##  $ ClorfA  : num [1:40] 36.1 39.1 42.6 45.6 41.7 ...
```

```
"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 ...
##  $ ClorfA  : num [1:40] 36.8 37.4 38.9 38.7 35.3 ...
```

```
"dados_tempo_5"
```

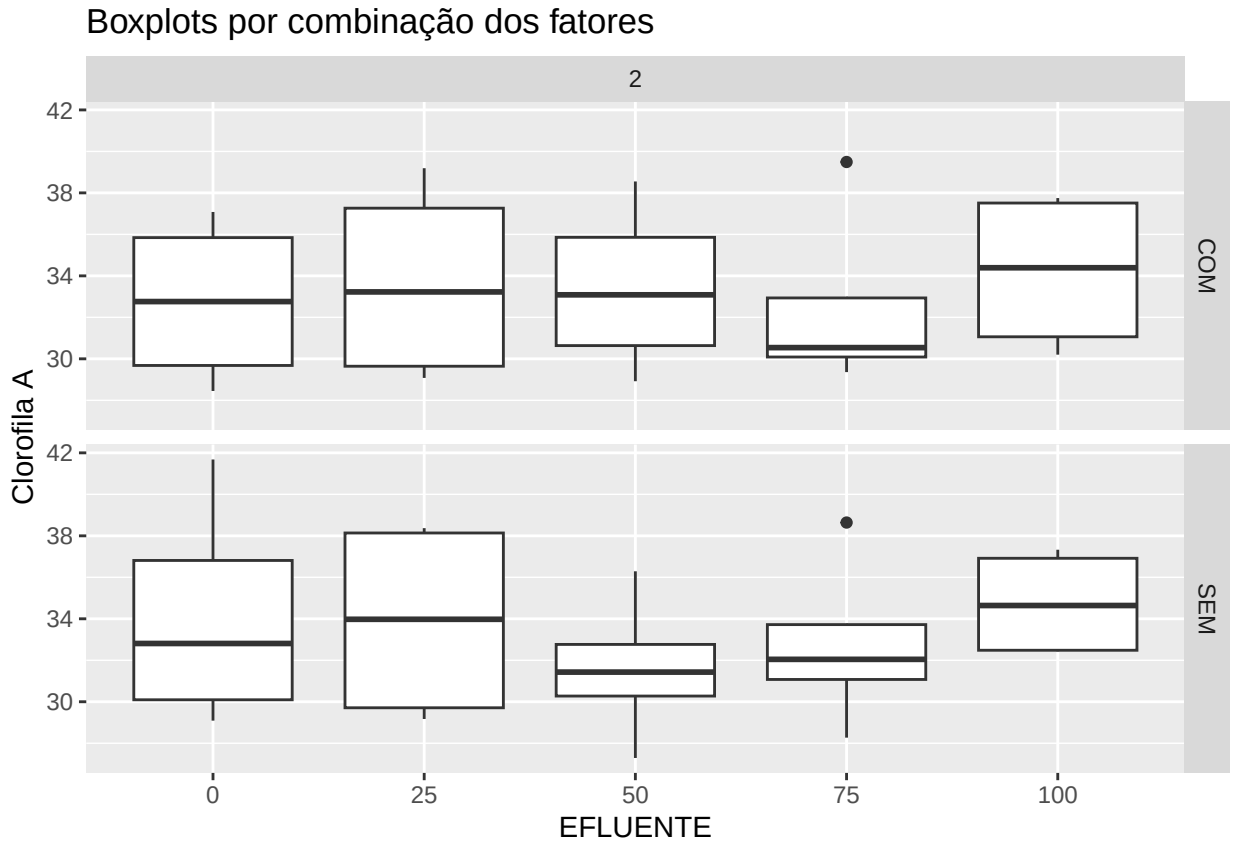
```
## [1] "dados_tempo_5"
```

```
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 ...
##  $ ClorfA  : 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

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

# 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$ClorfA[which(blocos_dados_tempo_1$ClorfA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_1$ClorfA[which(blocos_dados_tempo_1$ClorfA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_1 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_1$ClorfA))] =
  media_blocos_tempo_1$ClorfA[which(is.na(blocos_dados_tempo_1$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 35.4 37.1 30.1 28.4 39.2 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1: 0   Min.   :27.29
##  2:10   25 :8    SEM:20   2:40   1st Qu.:30.04
##  3:10   50 :8                    Median :31.47
##  4:10   75 :8                    Mean   :32.93
##         100:8                    3rd Qu.:36.66
##                                  Max.   :41.68
```

```r
# Número de observações
with(blocos_dados_tempo_1, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   131.0500 136.3900
## 25  134.7200 135.4900
## 50  133.6400 126.4400
## 75  120.5867 123.1467
## 100 136.7300 139.0400
```

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

```
##          COM      SEM
## 0   32.76250 34.09750
## 25  33.68000 33.87250
## 50  33.41000 31.61000
## 75  30.14667 30.78667
## 100 34.18250 34.76000
```

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

```
##            COM       SEM
## 0   17.1654250 32.404492
## 25  24.9954000 25.245492
## 50  17.9407333 13.560267
## 75   0.3388222  3.167622
## 100 15.7188917  7.073933
```

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

```
##           COM      SEM
## 0   4.1431178 5.692494
## 25  4.9995400 5.024489
## 50  4.2356503 3.682427
## 75  0.5820844 1.779782
## 100 3.9647057 2.659687
```

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

```
## Tables of means
## Grand mean
##
## 32.93083
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 35.14 36.40 30.82 29.37
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 33.43 33.78 32.51 30.47 34.47
##
##  INOCULO
## INOCULO
##   COM   SEM
## 32.84 33.03
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   32.76 34.10
##      25  33.68 33.87
##      50  33.41 31.61
##      75  30.15 30.79
##      100 34.18 34.76
```

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

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

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

```
## [1] 0.16115
```

```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_1$ClorfA,
```

```
                                    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] 0.5486984
```

```
# 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(ClorfA ~ 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  340.1   113.4
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1 0.3572  0.3572
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  3.674   1.225
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4  76.69  19.172   3.567 0.0203 *
## INOCULO:EFLUENTE   4  11.25   2.812   0.523 0.7197
## Residuals         24 129.01   5.376
## ---
## 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(ClorfA ~ 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: ClorfA
##           Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3 340.15 113.382 23.0717 1.298e-07 ***
## EFLUENTE   4  76.69  19.172  3.9012   0.01258 *
## 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)
```

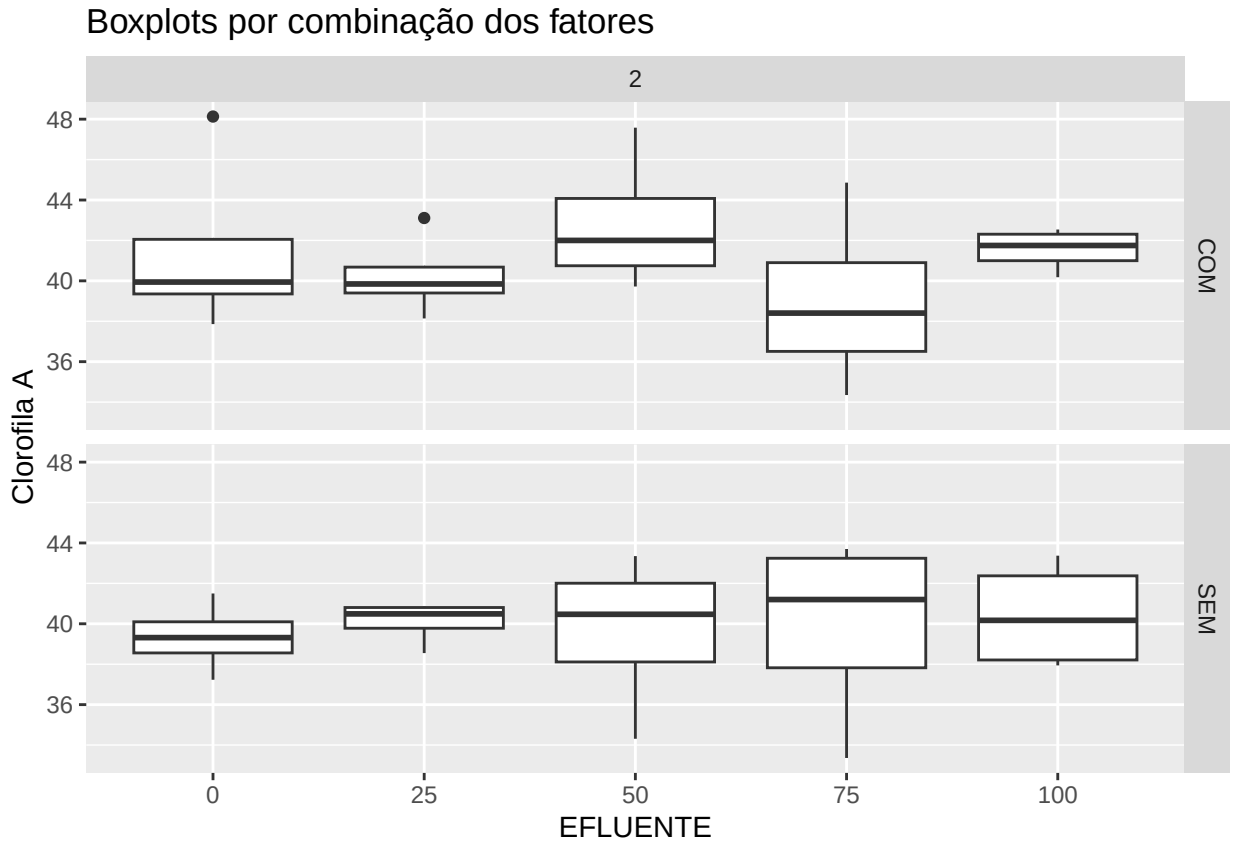
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

```
    media_interacao <- aggregate(ClorfA ~ 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

# 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$ClorfA[which(blocos_dados_tempo_2$ClorfA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_2$ClorfA[which(blocos_dados_tempo_2$ClorfA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_2 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_2$ClorfA))] =
  media_blocos_tempo_2$ClorfA[which(is.na(blocos_dados_tempo_2$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 39.2 40 37.9 39.9 39.8 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1: 0   Min.   :33.36
##  2:10   25 :8    SEM:20   2:40   1st Qu.:38.89
##  3:10   50 :8                   Median :39.86
##  4:10   75 :8                   Mean   :40.13
##         100:8                   3rd Qu.:41.68
##                                 Max.   :47.58
```

```r
# Número de observações
with(blocos_dados_tempo_2, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   156.9867 157.36
## 25  157.1067 160.38
## 50  171.3000 158.60
## 75  156.0200 159.46
## 100 166.2200 161.65
```

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

```
##          COM     SEM
## 0   39.24667 39.3400
## 25  39.27667 40.0950
## 50  42.82500 39.6500
## 75  39.00500 39.8650
## 100 41.55500 40.4125
```

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

```
##            COM       SEM
## 0    0.9668222  3.105800
## 25   0.6464222  1.149700
## 50  11.7561667 15.308867
## 75  19.8104333 22.576967
## 100  1.1325667  7.323825
```

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

```
##           COM      SEM
## 0   0.9832712 1.762328
## 25  0.8040039 1.072241
## 50  3.4287267 3.912655
## 75  4.4508913 4.751523
## 100 1.0642212 2.706257
```

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

```
## Tables of means
## Grand mean
##
## 40.12708
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 40.53 38.27 40.18 41.53
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 39.29 39.69 41.24 39.44 40.98
##
##  INOCULO
## INOCULO
##   COM   SEM
## 40.38 39.87
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   39.25 39.34
##      25  39.28 40.10
##      50  42.83 39.65
##      75  39.01 39.87
##      100 41.56 40.41
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_2 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.03425552
```

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_2$ClorfA,
```

```
                                    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] 0.2479947
```

```
# 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(ClorfA ~ 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  56.08   18.69
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  2.592   2.592
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.761  0.9202
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  26.69   6.671   0.832  0.518
## INOCULO:EFLUENTE  4  23.02   5.754   0.717  0.588
## Residuals        24 192.49   8.020
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfA ~ 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: ClorfA
##    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)
```

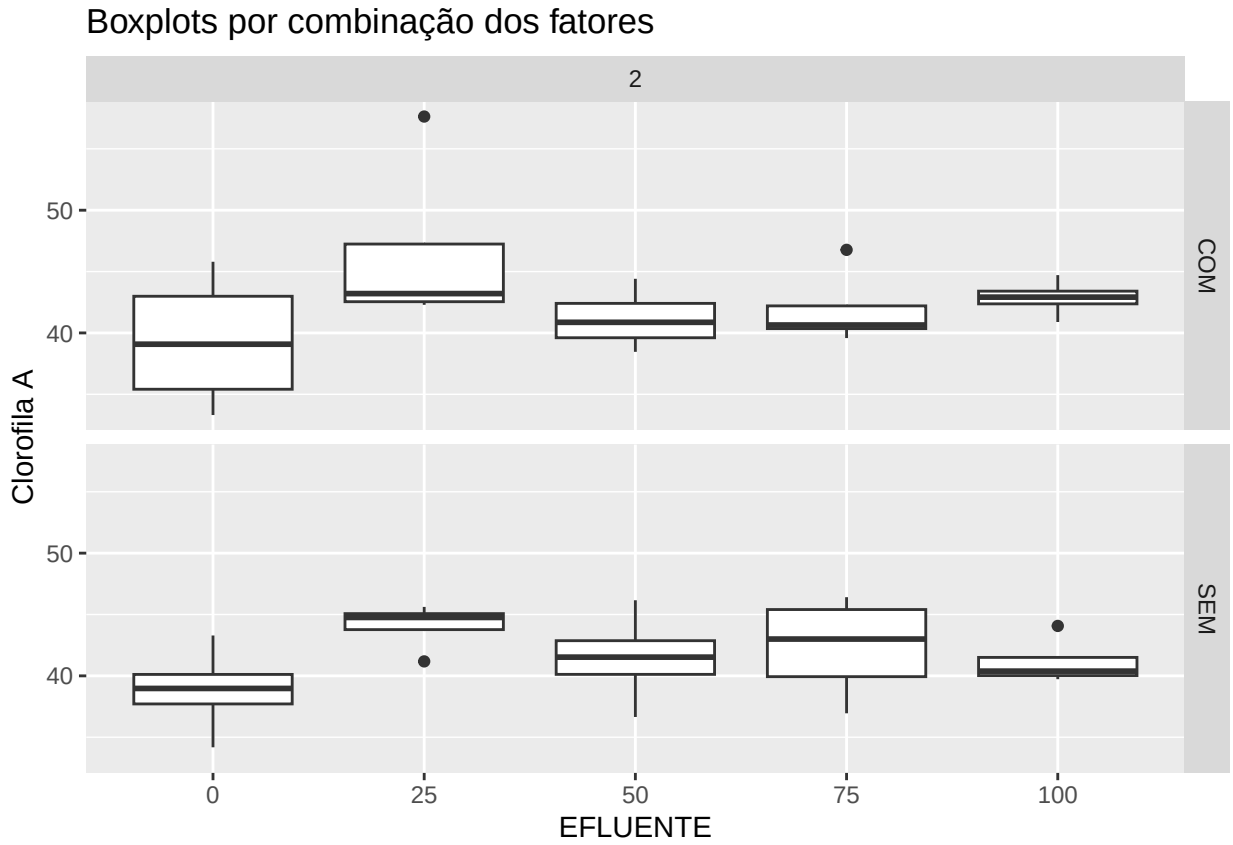
```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

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




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

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

# 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$ClorfA[which(blocos_dados_tempo_3$ClorfA <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_3$ClorfA[which(blocos_dados_tempo_3$ClorfA >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_3 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_3$ClorfA))] =
  media_blocos_tempo_3$ClorfA[which(is.na(blocos_dados_tempo_3$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 36.1 45.8 42 33.3 42.6 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1: 0   Min.   :33.31
##  2:10   25 :8    SEM:20   2:40   1st Qu.:39.92
##  3:10   50 :8                   Median :41.51
##  4:10   75 :8                   Mean   :41.44
##         100:8                   3rd Qu.:43.94
##                                 Max.   :46.41
```

```r
# Número de observações
with(blocos_dados_tempo_3, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##        COM      SEM
## 0   157.27 155.4200
## 25  171.60 180.1867
## 50  164.60 165.8600
## 75  161.16 169.3600
## 100 171.43 160.6667
```

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

```
##         COM      SEM
## 0   39.3175 38.85500
## 25  42.9000 45.04667
## 50  41.1500 41.46500
## 75  40.2900 42.34000
## 100 42.8575 40.16667
```

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

```
##            COM        SEM
## 0   31.9558250 13.8560333
## 25   0.4064667  0.1756222
## 50   6.5191333 15.1181667
## 75   0.2528667 18.3526667
## 100  2.4384917  0.1421556
```

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

```
##           COM       SEM
## 0   5.6529483 3.7223693
## 25  0.6375474 0.4190731
## 50  2.5532594 3.8882087
## 75  0.5028585 4.2840012
## 100 1.5615671 0.3770352
```

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

```
## Tables of means
## Grand mean
##
## 41.43883
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 40.57 42.56 41.45 41.18
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 39.09 43.97 41.31 41.31 41.51
##
##  INOCULO
## INOCULO
##   COM   SEM
## 41.30 41.57
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   39.32 38.85
##      25  42.90 45.05
##      50  41.15 41.46
##      75  40.29 42.34
##      100 42.86 40.17
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_3 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.84021
```

```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$ClorfA,
```

```
                                    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] 0.4846748
```

```
# 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(ClorfA ~ 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  20.72   6.906
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  0.738   0.738
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  3.376   1.125
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  95.97  23.993   2.364 0.0815 .
## INOCULO:EFLUENTE  4  31.99   7.998   0.788 0.5443
## Residuals        24 243.56  10.148
## ---
## 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(ClorfA ~ 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: ClorfA
##    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)
```

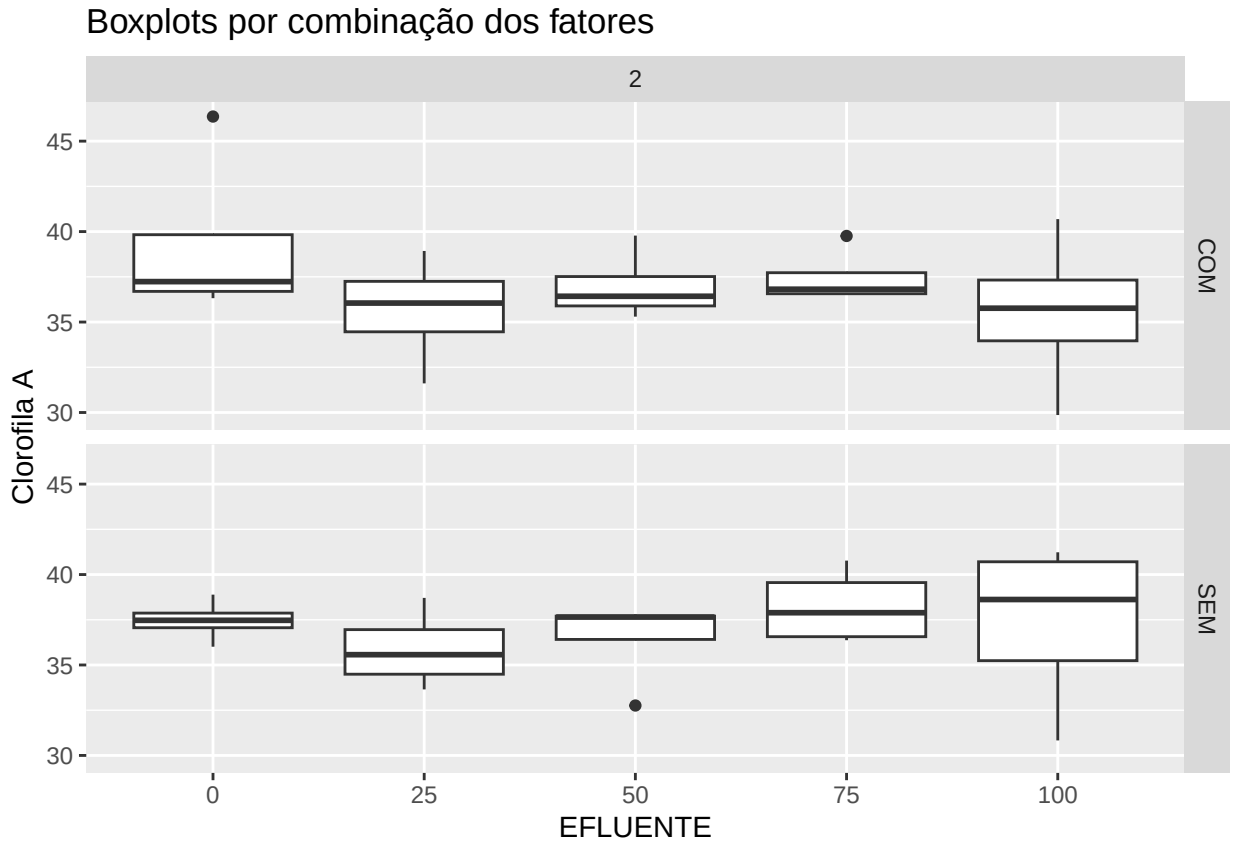
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ClorfA ~ 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 = ClorfA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Clorofila A") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

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

# 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$ClorfA[which(blocos_dados_tempo_4$ClorfA <
                                  limites_outliers$LIM_INF)] = NA
blocos_dados_tempo_4$ClorfA[which(blocos_dados_tempo_4$ClorfA >
                                  limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_tempo_4 = aggregate(ClorfA ~ 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$ClorfA[which(is.na(blocos_dados_tempo_4$ClorfA))] =
  media_blocos_tempo_4$ClorfA[which(is.na(blocos_dados_tempo_4$ClorfA))]
```

```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 ...
##  $ ClorfA  : num [1:40] 36.8 36.3 37.6 36.9 38.9 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfA
##  1:10   0  :8    COM:20   1: 0   Min.   :29.86
##  2:10   25 :8    SEM:20   2:40   1st Qu.:36.17
##  3:10   50 :8                    Median :36.74
##  4:10   75 :8                    Mean   :36.84
##         100:8                    3rd Qu.:37.73
##                                  Max.   :41.23
```

```r
# Número de observações
with(blocos_dados_tempo_4, tapply(ClorfA, 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(ClorfA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM    SEM
## 0   147.7200 149.84
## 25  142.6400 143.50
## 50  147.9300 150.80
## 75  146.8667 152.92
## 100 142.0800 149.30
```

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

```
##          COM    SEM
## 0   36.93000 37.460
## 25  35.66000 35.875
## 50  36.98250 37.700
## 75  36.71667 38.230
## 100 35.52000 37.325
```

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

```
##             COM          SEM
## 0    0.30086667  1.384933333
## 25   9.40626667  4.817966667
## 50   3.83429167  0.005266667
## 75   0.05615556  4.439200000
## 100 19.75433333 22.694766667
```

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

```
##           COM       SEM
## 0   0.5485131 1.1768319
## 25  3.0669638 2.1949867
## 50  1.9581347 0.0725718
## 75  0.2369716 2.1069409
## 100 4.4445847 4.7639025
```

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

```
## Tables of means
## Grand mean
##
## 36.83992
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 37.06 36.72 35.15 38.43
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 37.20 35.77 37.34 37.47 36.42
##
##  INOCULO
## INOCULO
##   COM   SEM
## 36.36 37.32
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   36.93 37.46
##      25  35.66 35.87
##      50  36.98 37.70
##      75  36.72 38.23
##      100 35.52 37.32
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_4 = aggregate(ClorfA ~ 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$ClorfA)$p.value

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

```
## [1] 0.5918375
```

```
# 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$ClorfA,
```

```
                                    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] 0.5548663
```

```
# 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(ClorfA ~ 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  54.32   18.11
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  9.143   9.143
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  1.462  0.4873
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  16.82   4.206   0.700  0.600
## INOCULO:EFLUENTE  4   3.64   0.909   0.151  0.961
## Residuals        24 144.30   6.012
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfA ~ 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: ClorfA
##       Df Sum Sq Mean Sq F value  Pr(>F)
## BLOCO  3 54.322  18.107  3.3541 0.03349 *
## 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(ClorfA ~ 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$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfA
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfA
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfA
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfA
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfA

# 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 ClorfA atualizado - Ciclo 2.xlsx",
      sheetName = "R - ClorfA_2", append = FALSE)
```