# Clorofilômetro Clorofila B

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 ClorfB 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] 6.68 7.83 7.83 8.71 7.85 8.71 7.93 8.74 7.52 8.39 ...
##  $ 43 E 40 DAS: num [1:80] 7.51 6.24 7.53 6.29 8.79 6.25 8.86 6.65 9.37 6.97 ...
##  $ 52 E 54 DAS: num [1:80] 5.8 5.91 5.66 5.95 5.84 5.53 6.41 6.44 6.1 6.27 ...
##  $ 64 E 61 DAS: num [1:80] 5.83 5.65 6.21 6.19 5.68 5.61 5.57 5.69 5.96 6.1 ...
##  $ 114 DAS    : num [1:80] 12.2 11.7 13.2 12.7 13.5 ...
```

```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$ClorfB = dados_tempo_1$`24 E 29 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43, 40
dados_tempo_2$ClorfB = dados_tempo_2$`43 E 40 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$ClorfB = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfB = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfB = dados_tempo_5$`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 ...
##  $ ClorfB  : num [1:40] 6.68 7.83 7.83 8.71 7.85 8.71 7.93 8.74 7.52 8.39 ...
```

```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 ...
##  $ ClorfB  : num [1:40] 7.51 6.24 7.53 6.29 8.79 6.25 8.86 6.65 9.37 6.97 ...
```

```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 ...
##  $ ClorfB  : num [1:40] 5.8 5.91 5.66 5.95 5.84 5.53 6.41 6.44 6.1 6.27 ...
```

```
"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 ...
##  $ ClorfB  : num [1:40] 5.83 5.65 6.21 6.19 5.68 5.61 5.57 5.69 5.96 6.1 ...
```

```
"dados_tempo_5"
```

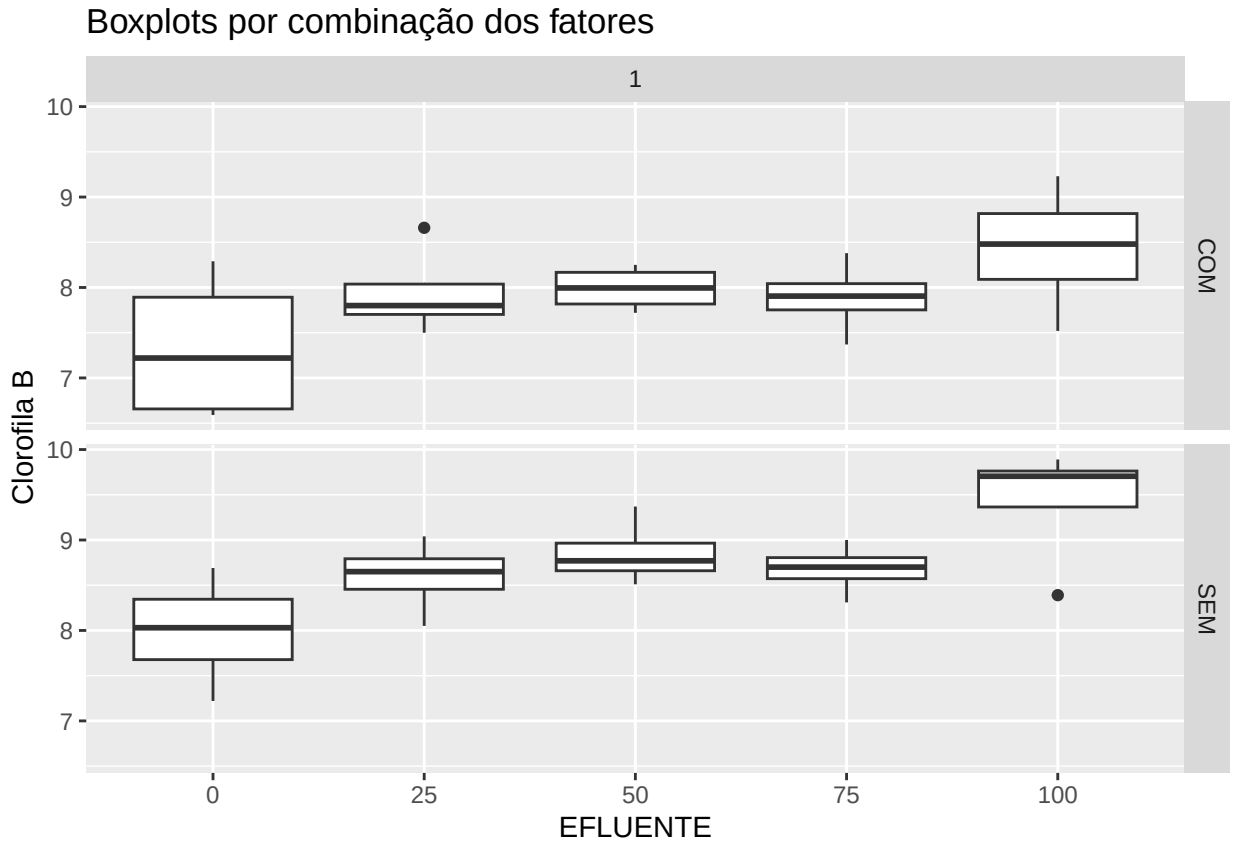
```
## [1] "dados_tempo_5"
```

```
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 ...
##  $ ClorfB  : num [1:40] 12.2 11.7 13.2 12.7 13.5 ...
```

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 6.68 8.29 6.59 7.76 7.83 7.7 7.5 7.77 7.85 8.14 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1:40   Min.   :6.590
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:7.815
##  3:10   50 :8                    Median :8.285
##  4:10   75 :8                    Mean   :8.323
##         100:8                    3rd Qu.:8.717
##                                  Max.   :9.890
```

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

```
##       COM      SEM
## 0   29.32 31.97000
## 25  30.80 34.39000
## 50  31.96 35.42000
## 75  31.56 34.71000
## 100 33.71 39.06667
```

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

```
##        COM      SEM
## 0   7.3300 7.992500
## 25  7.7000 8.597500
## 50  7.9900 8.855000
## 75  7.8900 8.677500
## 100 8.4275 9.766667
```

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

```
##            COM         SEM
## 0   0.69220000 0.388691667
## 25  0.02060000 0.169425000
## 50  0.06086667 0.135300000
## 75  0.17073333 0.081091667
## 100 0.51769167 0.007755556
```

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

```
##           COM        SEM
## 0   0.8319856 0.62345141
## 25  0.1435270 0.41161268
## 50  0.2467117 0.36783148
## 75  0.4131989 0.28476599
## 100 0.7195079 0.08806563
```

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

```
## Tables of means
## Grand mean
##
## 8.322667
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 8.157 8.773 8.103 8.258
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.661 8.149 8.422 8.284 9.097
##
##  INOCULO
## INOCULO
##   COM   SEM
## 7.867 8.778
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   7.330 7.992
##      25  7.700 8.598
##      50  7.990 8.855
##      75  7.890 8.678
##      100 8.428 9.767
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_1 = aggregate(ClorfB ~ 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$ClorfB))$p.value

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

```
## [1] 0.8623361
```

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

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

```
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_1$ClorfB,
```

```
                                    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] 0.9806044
```

```
# 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(ClorfB ~ 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  2.828  0.9426
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  8.287   8.287
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.3836  0.1279
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4  8.631  2.1578  14.706 3.34e-06 ***
## INOCULO:EFLUENTE   4  0.525  0.1313   0.895    0.482
## Residuals         24  3.522  0.1467
## ---
## 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(ClorfB ~ 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: ClorfB
##           Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO      3 2.8279  0.9426  6.5173  0.001848 **
## INOCULO    1 8.2871  8.2871 57.2963 3.839e-08 ***
## EFLUENTE   4 8.6314  2.1578 14.9192 1.494e-06 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

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

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

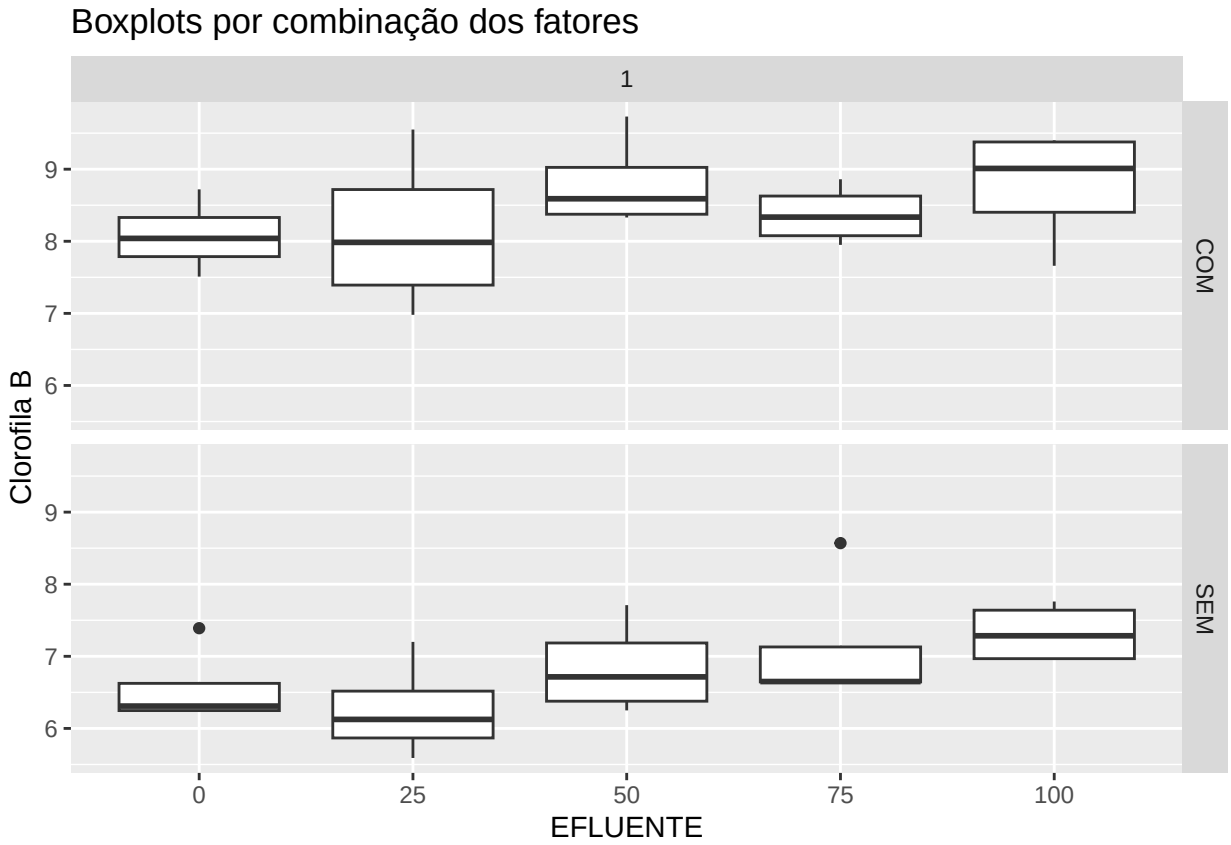
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 7.51 7.88 8.2 8.72 7.53 8.44 9.55 6.98 8.79 8.33 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1:40   Min.   :5.590
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:6.640
##  3:10   50 :8                    Median :7.565
##  4:10   75 :8                    Mean   :7.551
##         100:8                    3rd Qu.:8.402
##                                  Max.   :9.730
```

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

```
##       COM      SEM
## 0   32.31 25.14667
## 25  32.50 25.04000
## 50  35.24 27.39000
## 75  33.48 26.57333
## 100 35.08 29.29000
```

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

```
##        COM      SEM
## 0   8.0775 6.286667
## 25  8.1250 6.260000
## 50  8.8100 6.847500
## 75  8.3700 6.643333
## 100 8.7700 7.322500
```

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

```
##           COM          SEM
## 0   0.2629583 3.488889e-03
## 25  1.2649667 4.744667e-01
## 50  0.4178667 4.366917e-01
## 75  0.1704667 8.888889e-05
## 100 0.6678000 1.746917e-01
```

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

```
##           COM        SEM
## 0   0.5127946 0.05906682
## 25  1.1247074 0.68881541
## 50  0.6464261 0.66082650
## 75  0.4128761 0.00942809
## 100 0.8171903 0.41796132
```

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

```
## Tables of means
## Grand mean
##
## 7.55125
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 7.446 7.421 7.731 7.607
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.182 7.193 7.829 7.507 8.046
##
##  INOCULO
## INOCULO
##   COM   SEM
## 8.431 6.672
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   8.078 6.287
##      25  8.125 6.260
##      50  8.810 6.848
##      75  8.370 6.643
##      100 8.770 7.323
```

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

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

```
## [1] 0.1974483
```

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

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

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

```
                                    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] 0.9993641
```

```
# 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(ClorfB ~ 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 0.6354  0.2118
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  30.92   30.92
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.9113  0.3038
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  4.712  1.1780   2.807 0.0483 *
## INOCULO:EFLUENTE  4  0.303  0.0759   0.181 0.9461
## Residuals        24 10.074  0.4197
## ---
## 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(ClorfB ~ 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: ClorfB
##           Df Sum Sq Mean Sq F value    Pr(>F)
## INOCULO    1 30.923  30.923 76.0059 2.471e-09 ***
## EFLUENTE   4  4.712   1.178  2.8954   0.04081 *
## 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)
```

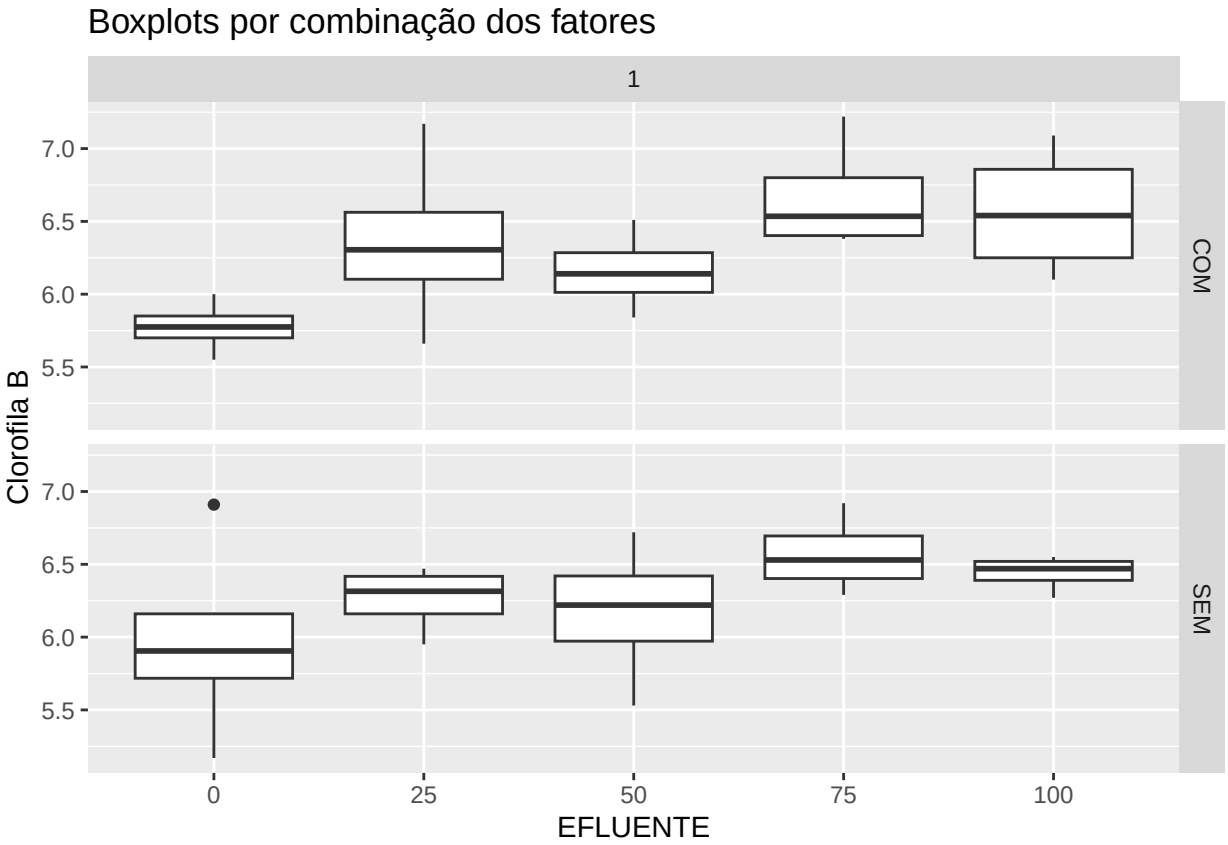
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 5.8 5.75 5.55 6 5.66 7.17 6.25 6.36 5.84 6.07 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1:40   Min.   :5.170
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:5.940
##  3:10   50 :8                    Median :6.295
##  4:10   75 :8                    Mean   :6.263
##         100:8                    3rd Qu.:6.510
##                                  Max.   :7.220
```

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

```
##       COM   SEM
## 0   23.10 22.64
## 25  25.44 25.05
## 50  24.63 24.69
## 75  26.67 26.27
## 100 26.27 25.76
```

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

```
##        COM    SEM
## 0   5.7750 5.6600
## 25  6.3600 6.2625
## 50  6.1575 6.1725
## 75  6.6675 6.5675
## 100 6.5675 6.4400
```

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

```
##            COM        SEM
## 0   0.03416667 0.12006667
## 25  0.38606667 0.05355833
## 50  0.07849167 0.24569167
## 75  0.15142500 0.07342500
## 100 0.20275833 0.01533333
```

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

```
##           COM       SEM
## 0   0.1848423 0.3465064
## 25  0.6213426 0.2314267
## 50  0.2801636 0.4956729
## 75  0.3891337 0.2709705
## 100 0.4502869 0.1238278
```

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

```
## Tables of means
## Grand mean
##
## 6.263
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 5.991 6.368 6.317 6.376
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 5.717 6.311 6.165 6.617 6.504
##
##  INOCULO
## INOCULO
##   COM   SEM
## 6.305 6.221
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   5.775 5.660
##      25  6.360 6.263
##      50  6.157 6.173
##      75  6.667 6.568
##      100 6.568 6.440
```

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

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

```
## [1] 0.3412947
```

```
# 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$ClorfB,
```

```
                                    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] 0.7094925
```

```
# 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(ClorfB ~ 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  1.007  0.3356
##
## Error: INOCULO
##         Df  Sum Sq Mean Sq
## INOCULO  1 0.07225 0.07225
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.1948 0.06495
##
## Error: Within
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## EFLUENTE           4  3.945  0.9863   8.216 0.000254 ***
## INOCULO:EFLUENTE   4  0.026  0.0065   0.055 0.994081
## Residuals         24  2.881  0.1200
## ---
## 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(ClorfB ~ 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: ClorfB
##           Df Sum Sq Mean Sq F value    Pr(>F)
## EFLUENTE   4 3.9451 0.98627  8.6571 0.0001245 ***
## 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)
```

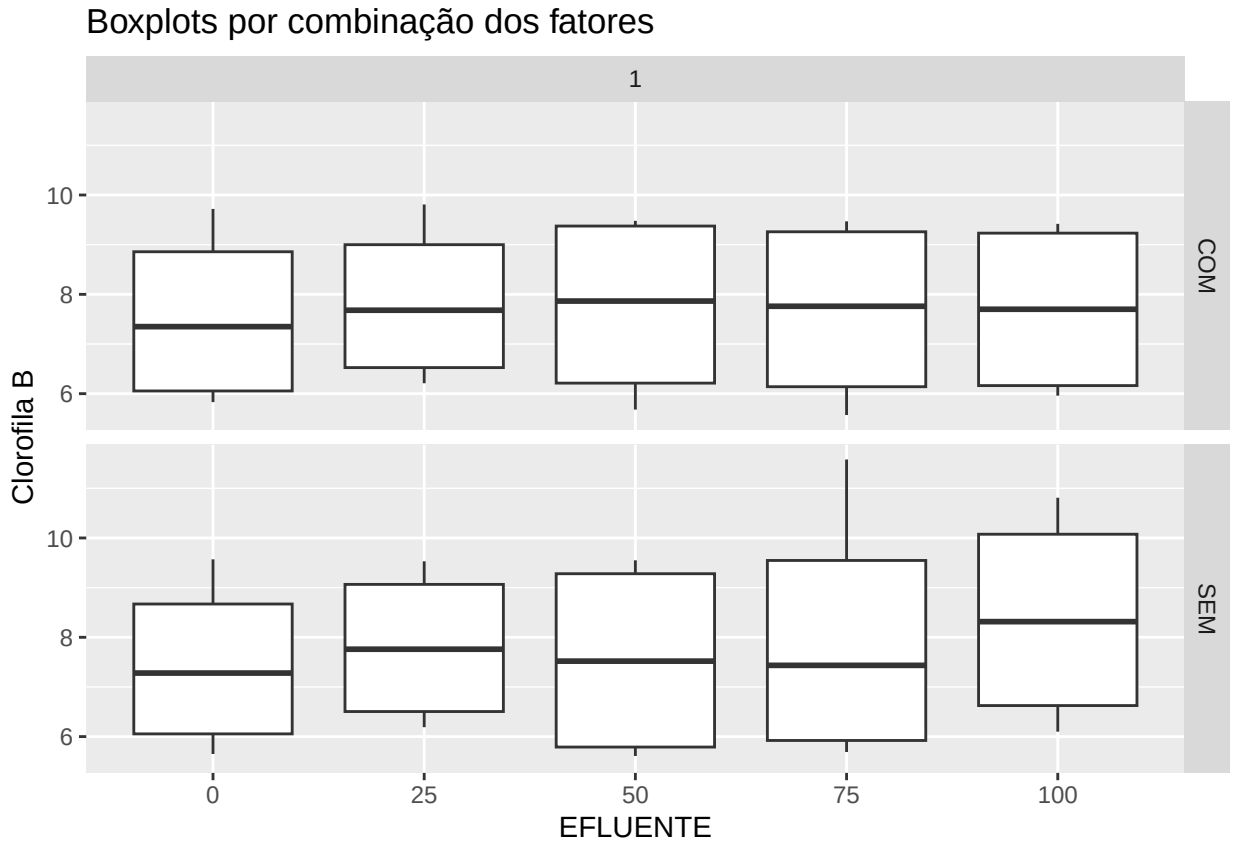
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 5.83 6.13 8.57 9.72 6.21 6.63 9.81 8.73 5.68 6.39 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1:40   Min.   : 5.570
##  2:10   25 :8    SEM:20   2: 0   1st Qu.: 6.122
##  3:10   50 :8                    Median : 7.585
##  4:10   75 :8                    Mean   : 7.769
##         100:8                    3rd Qu.: 9.432
##                                  Max.   :11.580
```

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

```
##       COM   SEM
## 0   30.25 29.78
## 25  31.38 31.24
## 50  30.89 30.20
## 75  30.56 32.14
## 100 30.78 33.54
```

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

```
##        COM   SEM
## 0   7.5625 7.445
## 25  7.8450 7.810
## 50  7.7225 7.550
## 75  7.6400 8.035
## 100 7.6950 8.385
```

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

```
##          COM      SEM
## 0   3.574492 3.389433
## 25  2.931300 2.744267
## 50  3.884158 4.447733
## 75  3.917467 7.634833
## 100 3.435900 5.234033
```

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

```
##          COM      SEM
## 0   1.890633 1.841041
## 25  1.712104 1.656583
## 50  1.970827 2.108965
## 75  1.979259 2.763120
## 100 1.853618 2.287801
```

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

```
## Tables of means
## Grand mean
##
## 7.769
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 5.849 6.316 9.555 9.356
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 7.504 7.827 7.636 7.837 8.040
##
##  INOCULO
## INOCULO
##   COM   SEM
## 7.693 7.845
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   7.562 7.445
##      25  7.845 7.810
##      50  7.722 7.550
##      75  7.640 8.035
##      100 7.695 8.385
```

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

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

```
## [1] 0.2254603
```

```
# 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$ClorfB,
```

```
                                    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] 0.279776
```

```
# 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(ClorfB ~ 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  115.1   38.35
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  0.231   0.231
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.6286  0.2095
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  1.356  0.3391   1.031  0.412
## INOCULO:EFLUENTE  4  1.123  0.2807   0.854  0.506
## Residuals        24  7.893  0.3289
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(ClorfB ~ 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: ClorfB
##       Df Sum Sq Mean Sq F value    Pr(>F)
## BLOCO  3 115.06  38.353  121.53 8.617e-16 ***
## 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(ClorfB ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_4, FUN = mean)
    print(media_interacao)
```

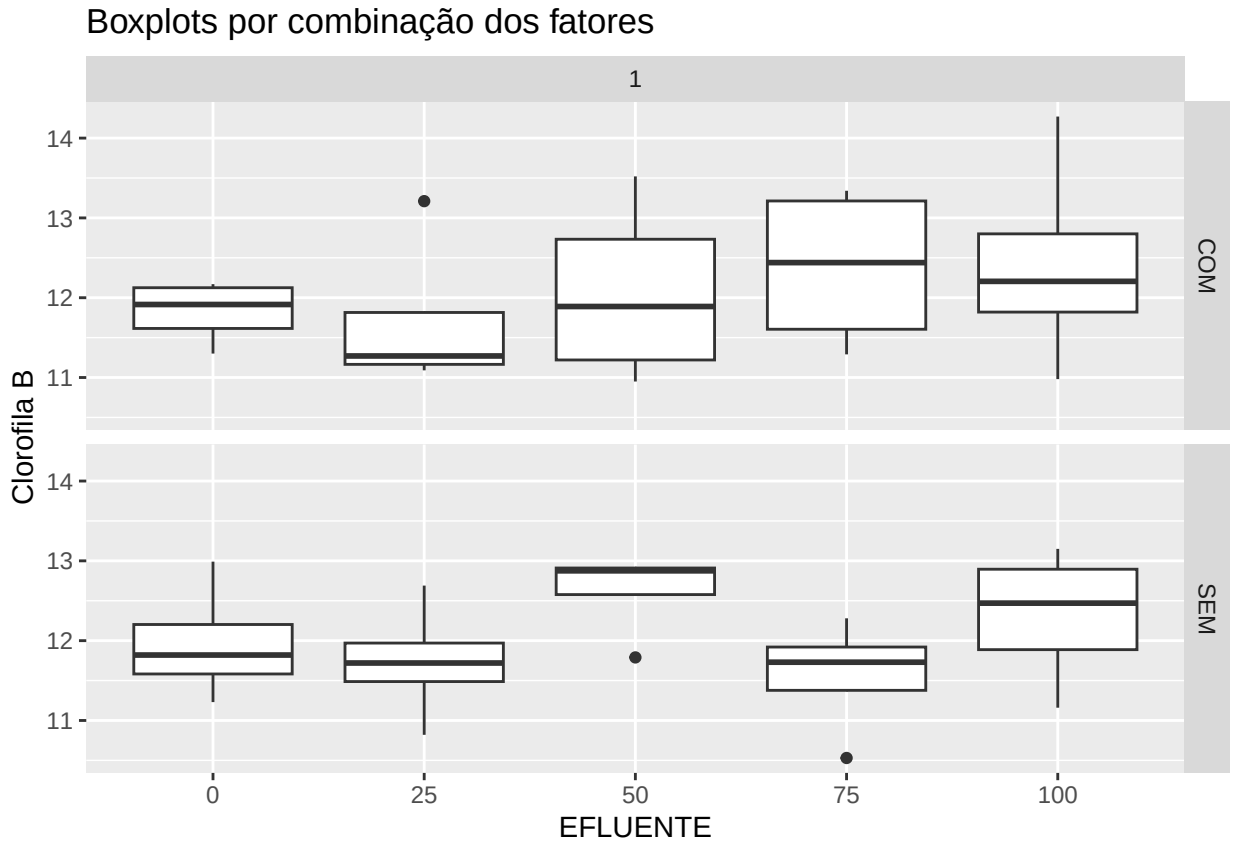
```
  }

}
```

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

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

**Análise para 114 dias**

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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 12.2 11.7 12.1 11.3 11.2 ...
```

```r
summary(blocos_dados_tempo_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1:40   Min.   :10.82
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:11.31
##  3:10   50 :8                    Median :11.93
##  4:10   75 :8                    Mean   :12.07
##         100:8                    3rd Qu.:12.82
##                                  Max.   :14.27
```

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

```
##       COM      SEM
## 0   47.30 47.86000
## 25  44.84 46.95000
## 50  48.25 51.54667
## 75  49.51 47.65333
## 100 49.66 49.25000
```

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

```
##         COM      SEM
## 0   11.8250 11.96500
## 25  11.2100 11.73750
## 50  12.0625 12.88667
## 75  12.3775 11.91333
## 100 12.4150 12.31250
```

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

```
##            COM         SEM
## 0   0.16230000 0.553900000
## 25  0.01146667 0.583291667
## 50  1.36475833 0.001088889
## 75  1.06089167 0.070488889
## 100 1.87016667 0.770158333
```

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

```
##           COM        SEM
## 0   0.4028647 0.74424458
## 25  0.1070825 0.76373534
## 50  1.1682287 0.03299832
## 75  1.0299960 0.26549744
## 100 1.3675404 0.87758665
```

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

```
## Tables of means
## Grand mean
##
## 12.0705
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 12.676 12.046 12.022 11.538
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 11.895 11.474 12.475 12.145 12.364
##
##  INOCULO
## INOCULO
##    COM    SEM
## 11.978 12.163
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   11.825 11.965
##      25  11.210 11.738
##      50  12.063 12.887
##      75  12.378 11.913
##      100 12.415 12.313
```

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

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

```
## [1] 0.9374441
```

```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_5$ClorfB,
```

```
                                    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] 0.5150462
```

```
# 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(ClorfB ~ 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  6.531   2.177
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.3422  0.3422
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.128  0.7095
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4  5.134  1.2836   2.883 0.0441 *
## INOCULO:EFLUENTE   4  2.064  0.5160   1.159 0.3535
## Residuals         24 10.686  0.4452
## ---
## 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(ClorfB ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_5)
```

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

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

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

```
## Analysis of Variance Table
##
## Response: ClorfB
##       Df Sum Sq Mean Sq F value  Pr(>F)
## BLOCO  3 6.5314  2.1771  4.5873 0.01013 *
## 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(ClorfB ~ 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$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfB
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfB
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfB
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfB
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfB

# 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 ClorfB atualizado - Ciclo 1.xlsx",
      sheetName = "R - ClorfB_1", append = FALSE)
```

### CICLO 2

```
# Leitura e tratamento dos dados
dados <- read_excel("Clorofilômetro ClorfB 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] 6.68 7.83 7.83 8.71 7.85 8.71 7.93 8.74 7.52 8.39 ...
##  $ 43 E 40 DAS: num [1:80] 7.51 6.24 7.53 6.29 8.79 6.25 8.86 6.65 9.37 6.97 ...
##  $ 52 E 54 DAS: num [1:80] 5.8 5.91 5.66 5.95 5.84 5.53 6.41 6.44 6.1 6.27 ...
##  $ 64 E 61 DAS: num [1:80] 5.83 5.65 6.21 6.19 5.68 5.61 5.57 5.69 5.96 6.1 ...
##  $ 114 DAS    : num [1:80] 12.2 11.7 13.2 12.7 13.5 ...
```

```
# Transformar as colunas em variáveis categóricas
dados$BLOCO <- factor(dados$BLOCO)
dados$CICLO <- factor(dados$CICLO)
dados$INOCULO <- factor(dados$INOCULO)
dados$EFLUENTE <- factor(dados$EFLUENTE)
```

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

```
dados_tempo_4 = dados[c(1:4,8)] # 64, 61
dados_tempo_4$ClorfB = dados_tempo_4$`64 E 61 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 114
dados_tempo_5$ClorfB = dados_tempo_5$`114 DAS`
dados_tempo_5 = dados_tempo_5[-5]

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

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

```
str(dados_tempo_1)
```

```
## tibble [40 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfB  : num [1:40] 7.03 7.69 8.17 8.13 8.01 9.34 8.61 8.36 8.2 7.63 ...
```

```
"dados_tempo_2"
```

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

```
str(dados_tempo_2)
```

```
## tibble [40 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfB  : num [1:40] 23.02 8.89 8.06 8.39 20.68 ...
```

```
"dados_tempo_3"
```

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

```
str(dados_tempo_3)
```

```
## tibble [40 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ClorfB  : num [1:40] 7.09 8.59 9.03 14.31 11.07 ...
```

```
"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 ...
##  $ ClorfB  : num [1:40] 10.74 10.92 10.71 9.39 10.61 ...
```

```
"dados_tempo_5"
```

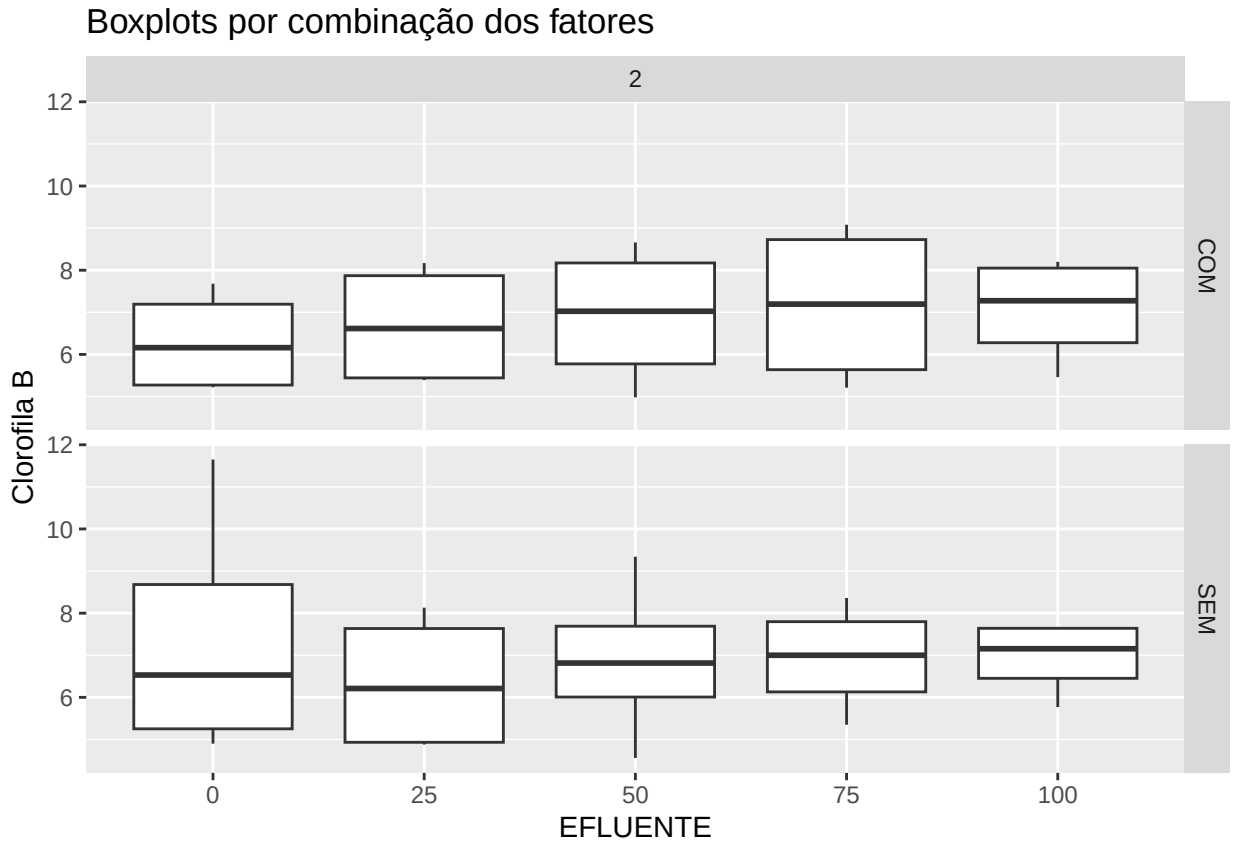
```
## [1] "dados_tempo_5"
```

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 7.03 7.68 5.22 5.29 8.17 7.77 5.39 5.46 8.01 8.66 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1: 0   Min.   : 4.560
##  2:10   25 :8    SEM:20   2:40   1st Qu.: 5.385
##  3:10   50 :8                   Median : 6.855
##  4:10   75 :8                   Mean   : 6.865
##         100:8                   3rd Qu.: 8.002
##                                 Max.   :11.650
```

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

```
##       COM   SEM
## 0   25.22 29.61
## 25  26.79 25.43
## 50  27.69 27.53
## 75  28.68 27.71
## 100 28.21 27.75
```

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

```
##        COM    SEM
## 0   6.3050 7.4025
## 25  6.6975 6.3575
## 50  6.9225 6.8825
## 75  7.1700 6.9275
## 100 7.0525 6.9375
```

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

```
##          COM       SEM
## 0   1.541233 9.5058250
## 25  2.186492 2.8478250
## 50  2.917892 3.8845583
## 75  3.831800 1.7650917
## 100 1.667692 0.8151583
```

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

```
##          COM       SEM
## 0   1.241464 3.0831518
## 25  1.478679 1.6875500
## 50  1.708184 1.9709283
## 75  1.957498 1.3285675
## 100 1.291391 0.9028612
```

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

```
## Tables of means
## Grand mean
##
## 6.8655
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 8.117 8.273 5.839 5.233
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 6.854 6.528 6.903 7.049 6.995
##
##  INOCULO
## INOCULO
##   COM   SEM
## 6.830 6.902
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   6.305 7.403
##      25  6.698 6.358
##      50  6.923 6.883
##      75  7.170 6.928
##      100 7.053 6.938
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_1 = aggregate(ClorfB ~ 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$ClorfB))$p.value

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

```
## [1] 0.4087098
```

```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$ClorfB,
```

```
                                    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] 0.161172
```

```
# 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(ClorfB ~ 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  72.66   24.22
##
## Error: INOCULO
##         Df  Sum Sq Mean Sq
## INOCULO  1 0.05184 0.05184
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.1287 0.04289
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  1.329  0.3322   0.397  0.809
## INOCULO:EFLUENTE  4  2.736  0.6839   0.817  0.527
## Residuals        24 20.101  0.8376
```

```
modelo = modelo_PARCELASUB
```

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

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

```r
anova_result <- anova(modelo)

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

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

```
## Analysis of Variance Table
##
## Response: ClorfB
##       Df Sum Sq Mean Sq F value  Pr(>F)
## BLOCO  3 72.661   24.22  32.325 4.4e-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)
```

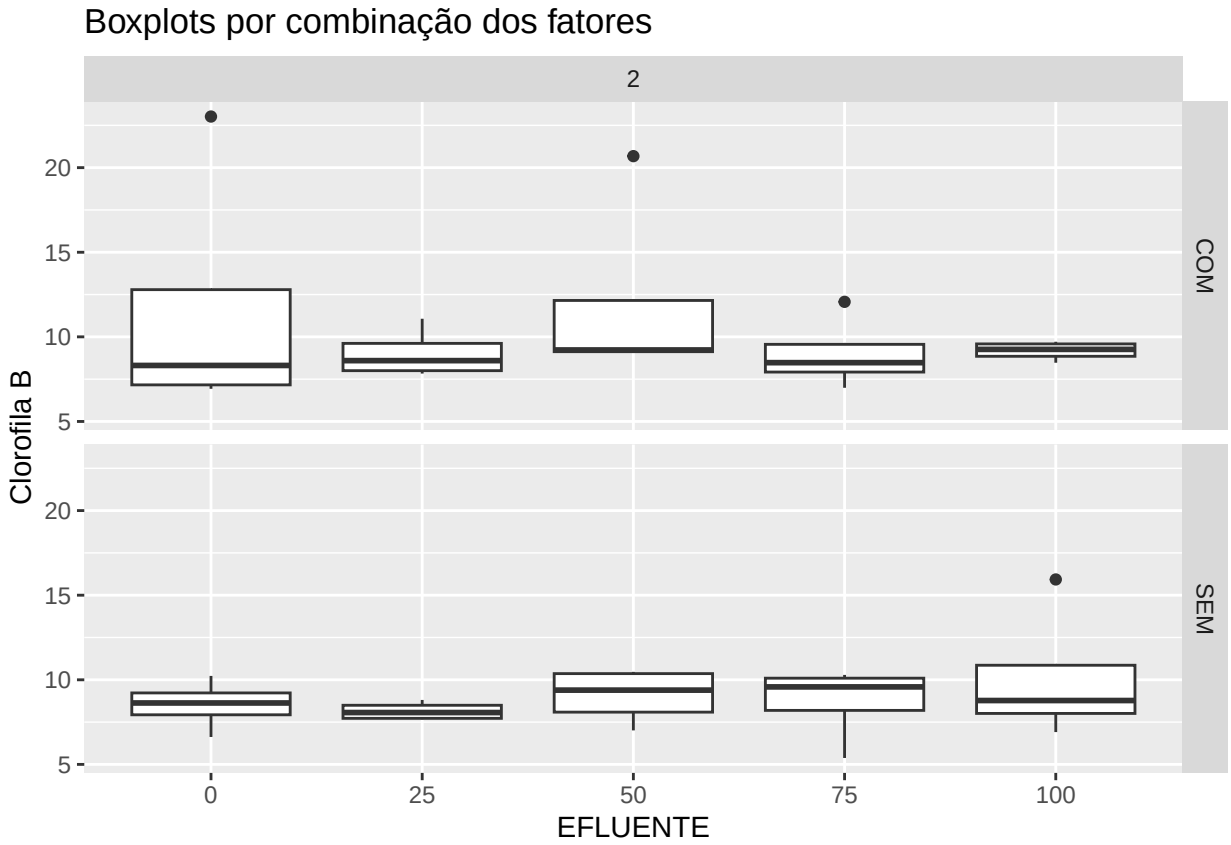
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 7.85 9.38 7.24 6.93 8.06 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1: 0   Min.   : 5.380
##  2:10   25 :8    SEM:20   2:40   1st Qu.: 7.845
##  3:10   50 :8                    Median : 8.595
##  4:10   75 :8                    Mean   : 8.581
##         100:8                    3rd Qu.: 9.210
##                                  Max.   :11.070
```

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

```
##          COM      SEM
## 0   31.40000 34.11000
## 25  36.09000 32.61000
## 50  36.70667 36.26000
## 75  31.92000 34.83000
## 100 36.70000 32.61333
```

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

```
##          COM      SEM
## 0   7.850000 8.527500
## 25  9.022500 8.152500
## 50  9.176667 9.065000
## 75  7.980000 8.707500
## 100 9.175000 8.153333
```

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

```
##             COM       SEM
## 0   1.186466667 2.2310917
## 25  2.184091667 0.2972250
## 50  0.009488889 2.7251667
## 75  0.530066667 5.1680250
## 100 0.318166667 0.8769556
```

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

```
##            COM       SEM
## 0   1.08925051 1.4936839
## 25  1.47786727 0.5451835
## 50  0.09741093 1.6508079
## 75  0.72805677 2.2733291
## 100 0.56406264 0.9364591
```

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

```
## Tables of means
## Grand mean
##
## 8.581
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 8.907 7.947 8.529 8.941
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 8.189 8.588 9.121 8.344 8.664
##
##  INOCULO
## INOCULO
##   COM   SEM
## 8.641 8.521
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM   SEM
##      0   7.850 8.528
##      25  9.023 8.152
##      50  9.177 9.065
##      75  7.980 8.708
##      100 9.175 8.153
```

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

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

```
## [1] 0.1536514
```

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

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

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

```
                                    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] 0.6913777
```

```
# 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(ClorfB ~ 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  6.399   2.133
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 0.1432  0.1432
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  2.895  0.9649
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4   4.07   1.017   0.655  0.629
## INOCULO:EFLUENTE  4   5.46   1.365   0.879  0.491
## Residuals        24  37.29   1.554
```

```
modelo = modelo_PARCELASUB
```

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

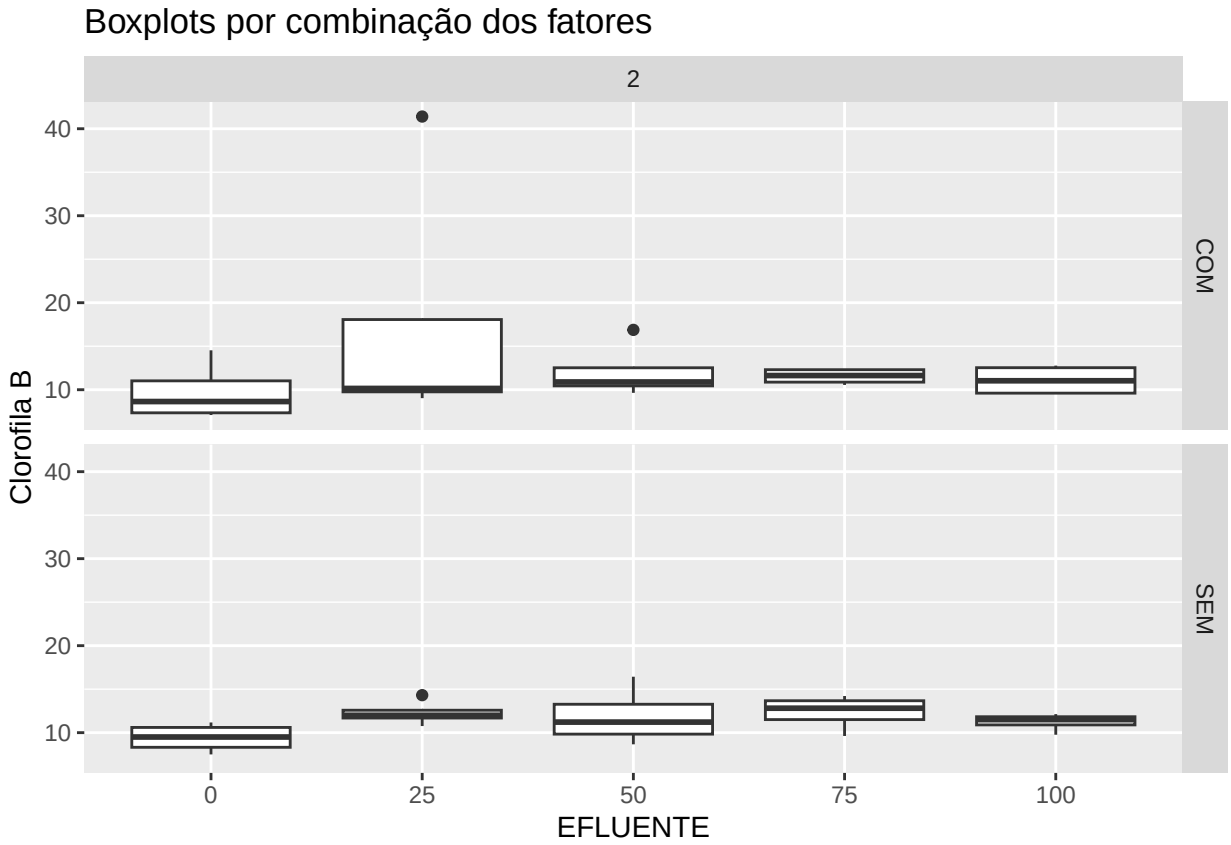
```r
# Realizar o teste de Tukey para todas as interações significativas
tukey_result <- TukeyHSD(modelo)
```

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 7.09 14.53 9.85 7.43 9.03 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1: 0   Min.   : 7.09
##  2:10   25 :8    SEM:20   2:40   1st Qu.: 9.73
##  3:10   50 :8                    Median :10.73
##  4:10   75 :8                    Mean   :10.91
##         100:8                    3rd Qu.:12.13
##                                  Max.   :16.43
```

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

```
##          COM      SEM
## 0   38.90000 37.67000
## 25  39.09333 46.33333
## 50  41.89333 47.52000
## 75  46.17000 49.45000
## 100 44.42000 44.89000
```

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

```
##           COM      SEM
## 0    9.725000  9.41750
## 25   9.773333 11.58333
## 50  10.473333 11.88000
## 75  11.542500 12.36250
## 100 11.105000 11.22250
```

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

```
##            COM        SEM
## 0   11.7713000  2.8059583
## 25   0.2902889  0.3389556
## 50   0.3688222 11.3118000
## 75   0.8378917  4.0866250
## 100  3.0371667  1.0740917
```

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

```
##           COM       SEM
## 0   3.4309328 1.6750995
## 25  0.5387846 0.5821989
## 50  0.6073074 3.3633019
## 75  0.9153642 2.0215403
## 100 1.7427469 1.0363839
```

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

```
## Tables of means
## Grand mean
##
## 10.9085
##
##  BLOCO
## BLOCO
##      1      2      3      4
##  9.710 12.070 10.913 10.940
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
##  9.571 10.678 11.177 11.953 11.164
##
##  INOCULO
## INOCULO
##    COM    SEM
## 10.524 11.293
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0    9.725  9.418
##      25   9.773 11.583
##      50  10.473 11.880
##      75  11.543 12.363
##      100 11.105 11.223
```

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

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

```
## [1] 0.4780832
```

```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$ClorfB,
```

```
                                    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] 0.3589301
```

```
# 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(ClorfB ~ 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  27.86   9.288
##
## Error: INOCULO
##         Df Sum Sq Mean Sq
## INOCULO  1  5.919   5.919
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3   4.31   1.437
##
## Error: Within
##                  Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE          4  24.55   6.136   1.948  0.135
## INOCULO:EFLUENTE  4   6.15   1.538   0.488  0.744
## Residuals        24  75.59   3.150
```

```
modelo = modelo_PARCELASUB
```

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

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

```r
anova_result <- anova(modelo)

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

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

```
## Analysis of Variance Table
##
## Response: ClorfB
##       Df Sum Sq Mean Sq F value  Pr(>F)
## BLOCO  3 27.865  9.2883  3.1386 0.04166 *
## 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)
```

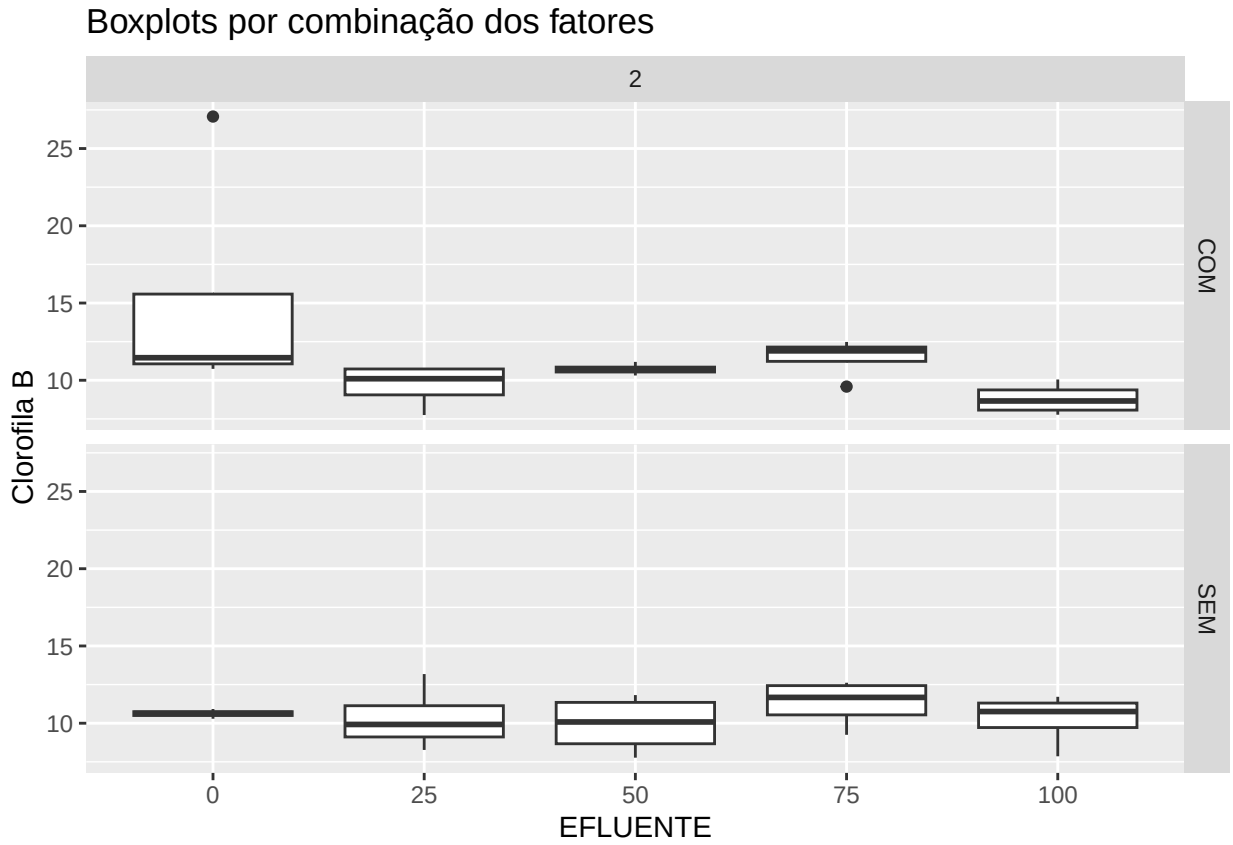
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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




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

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

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

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

```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 ...
##  $ ClorfB  : num [1:40] 10.7 11.8 11.2 11.2 10.7 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ClorfB
##  1:10   0  :8    COM:20   1: 0   Min.   : 7.750
##  2:10   25 :8    SEM:20   2:40   1st Qu.: 9.465
##  3:10   50 :8                    Median :10.725
##  4:10   75 :8                    Mean   :10.495
##         100:8                    3rd Qu.:11.342
##                                  Max.   :13.180
```

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

```
##          COM   SEM
## 0   44.88000 42.47
## 25  38.74000 41.29
## 50  42.85000 39.75
## 75  48.38667 45.21
## 100 35.14000 41.08
```

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

```
##          COM     SEM
## 0   11.22000 10.6175
## 25   9.68500 10.3225
## 50  10.71250  9.9375
## 75  12.09667 11.3025
## 100  8.78500 10.2700
```

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

```
##            COM        SEM
## 0   0.17126667 0.06875833
## 25  2.01796667 4.42129167
## 50  0.13375833 3.58222500
## 75  0.08562222 2.39955833
## 100 1.04730000 2.89886667
```

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

```
##           COM       SEM
## 0   0.4138438 0.2622181
## 25  1.4205515 2.1026868
## 50  0.3657299 1.8926767
## 75  0.2926128 1.5490508
## 100 1.0233768 1.7026058
```

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

```
## Tables of means
## Grand mean
##
## 10.49492
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 10.667 10.889 10.157 10.267
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 10.919 10.004 10.325 11.700  9.527
##
##  INOCULO
## INOCULO
##   COM   SEM
## 10.50 10.49
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   11.220 10.617
##      25   9.685 10.322
##      50  10.712  9.937
##      75  12.097 11.302
##      100  8.785 10.270
```

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

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

```
## [1] 0.9952551
```

```
# 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$ClorfB,
```

```
                                    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] 0.9277216
```

```
# 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(ClorfB ~ 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  3.509    1.17
##
## Error: INOCULO
##          Df    Sum Sq   Mean Sq
## INOCULO  1 0.0009669 0.0009669
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.9801  0.3267
##
## Error: Within
##                   Df Sum Sq Mean Sq F value Pr(>F)
## EFLUENTE           4  22.69   5.674   2.961 0.0403 *
## INOCULO:EFLUENTE   4   8.41   2.103   1.097 0.3805
## Residuals         24  45.99   1.916
## ---
## 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(ClorfB ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_4)
```

```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: ClorfB
##           Df Sum Sq Mean Sq F value  Pr(>F)
## EFLUENTE   4 22.695  5.6737  3.2614 0.02637 *
## 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(ClorfB ~ 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$`24 E 29 DAS` = blocos_dados_tempo_1$ClorfB
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`43 E 41 DAS` = blocos_dados_tempo_2$ClorfB
blocos_dados_tempo_3$`52 E 54 DAS` = blocos_dados_tempo_3$ClorfB
blocos_dados_tempo_4$`64 E 61 DAS` = blocos_dados_tempo_4$ClorfB
blocos_dados_tempo_5$`114 DAS` = blocos_dados_tempo_5$ClorfB

# 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 ClorfB atualizado - Ciclo 2.xlsx",
      sheetName = "R - ClorfB_2", append = FALSE)
```