NVDI

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("NDVI ok.xlsx")

# Ordenar o dataframe por quatro colunas diferentes
dados <- with(dados, dados[order(CICLO, BLOCO, EFLUENTE, INOCULO), ])

# Converter as colunas para tipo numérico e arredondar valores em duas casas
for (i in 5:9) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
```

```
## Warning: NAs introduzidos por coerção

## Warning: NAs introduzidos por coerção

## 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 ...
##  $ 37 DAS     : num [1:80] 0.52 0.53 0.58 0.58 0.62 0.54 0.49 0.64 0.54 0.45 ...
##  $ 43 DAS     : num [1:80] 0.66 0.69 0.73 0.71 0.77 0.68 0.69 0.78 0.7 0.59 ...
##  $ 52 E 54 DAS: num [1:80] 0.76 0.81 0.84 0.82 0.87 0.82 0.85 0.88 0.82 0.76 ...
##  $ 63 e 77 DAS: num [1:80] 0.9 0.9 0.91 0.91 0.94 0.93 0.96 0.95 0.9 0.92 ...
##  $ 85 DAS     : num [1:80] NA NA NA NA NA NA NA NA NA NA ...
```

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

```r
# Níveis para cada fator de tratamentos
library(dae)
n.F <- 2
n.D <- 5
n.Bloco <- 4
tr <- data.frame(cbind(INOCULO = paste("I", rep(1:n.F, each = n.D, times =
                                                 n.Bloco),
                                       sep = ""),
                       EFLUENTE = paste("E", rep(1:n.D, times = n.F*n.Bloco),
                                        sep = "")))
units <- list(Bloco = n.Bloco,
              Parcela = (n.F*n.D))
nest <- list(Parcela = "Bloco")
(lay <- designRandomize(allocated = tr,
                        recipient = units,
                        nested.recipients = nest,
                        seed = 9719532))
```

```
##    Bloco Parcela INOCULO EFLUENTE
## 1      1       1      I2       E3
## 2      1       2      I1       E2
## 3      1       3      I1       E1
## 4      1       4      I1       E4
## 5      1       5      I2       E4
## 6      1       6      I1       E5
## 7      1       7      I2       E1
## 8      1       8      I1       E3
## 9      1       9      I2       E5
## 10     1      10      I2       E2
## 11     2       1      I1       E1
## 12     2       2      I2       E5
## 13     2       3      I2       E4
## 14     2       4      I1       E3
```

```
## 15     2       5      I2       E3
## 16     2       6      I2       E1
## 17     2       7      I1       E4
## 18     2       8      I2       E2
## 19     2       9      I1       E2
## 20     2      10      I1       E5
## 21     3       1      I1       E2
## 22     3       2      I2       E4
## 23     3       3      I2       E5
## 24     3       4      I2       E2
## 25     3       5      I1       E5
## 26     3       6      I2       E3
## 27     3       7      I1       E3
## 28     3       8      I1       E4
## 29     3       9      I1       E1
## 30     3      10      I2       E1
## 31     4       1      I2       E5
## 32     4       2      I2       E4
## 33     4       3      I1       E3
## 34     4       4      I1       E2
## 35     4       5      I1       E4
## 36     4       6      I2       E1
## 37     4       7      I2       E3
## 38     4       8      I2       E2
## 39     4       9      I1       E1
## 40     4      10      I1       E5
```

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

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

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

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

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

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

```r
# Separar os dados de acordo com o período de tempo da coleta
dados_tempo_1 = dados[c(1:5)] # 37
dados_tempo_1$NVDI = dados_tempo_1$`37 DAS`
```

```r
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 43
dados_tempo_2$NVDI = dados_tempo_2$`43 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 52, 54
dados_tempo_3$NVDI = dados_tempo_3$`52 E 54 DAS`
dados_tempo_3 = dados_tempo_3[-5]
dados_tempo_4 = dados[c(1:4,8)] # 63, 77
dados_tempo_4$NVDI = dados_tempo_4$`63 e 77 DAS`
dados_tempo_4 = dados_tempo_4[-5]
dados_tempo_5 = dados[c(1:4,9)] # 85
dados_tempo_5$NVDI = dados_tempo_5$`85 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 [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NVDI    : num [1:80] 0.52 0.53 0.58 0.58 0.62 0.54 0.49 0.64 0.54 0.45 ...
```

```r
"dados_tempo_2"
```

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

```r
str(dados_tempo_2)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NVDI    : num [1:80] 0.66 0.69 0.73 0.71 0.77 0.68 0.69 0.78 0.7 0.59 ...
```

```r
"dados_tempo_3"
```

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

```r
str(dados_tempo_3)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NVDI    : num [1:80] 0.76 0.81 0.84 0.82 0.87 0.82 0.85 0.88 0.82 0.76 ...
```

```
"dados_tempo_4"
```

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

```
str(dados_tempo_4)
```

```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NVDI    : num [1:80] 0.9 0.9 0.91 0.91 0.94 0.93 0.96 0.95 0.9 0.92 ...
```

```
"dados_tempo_5"
```

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

```
str(dados_tempo_5)
```

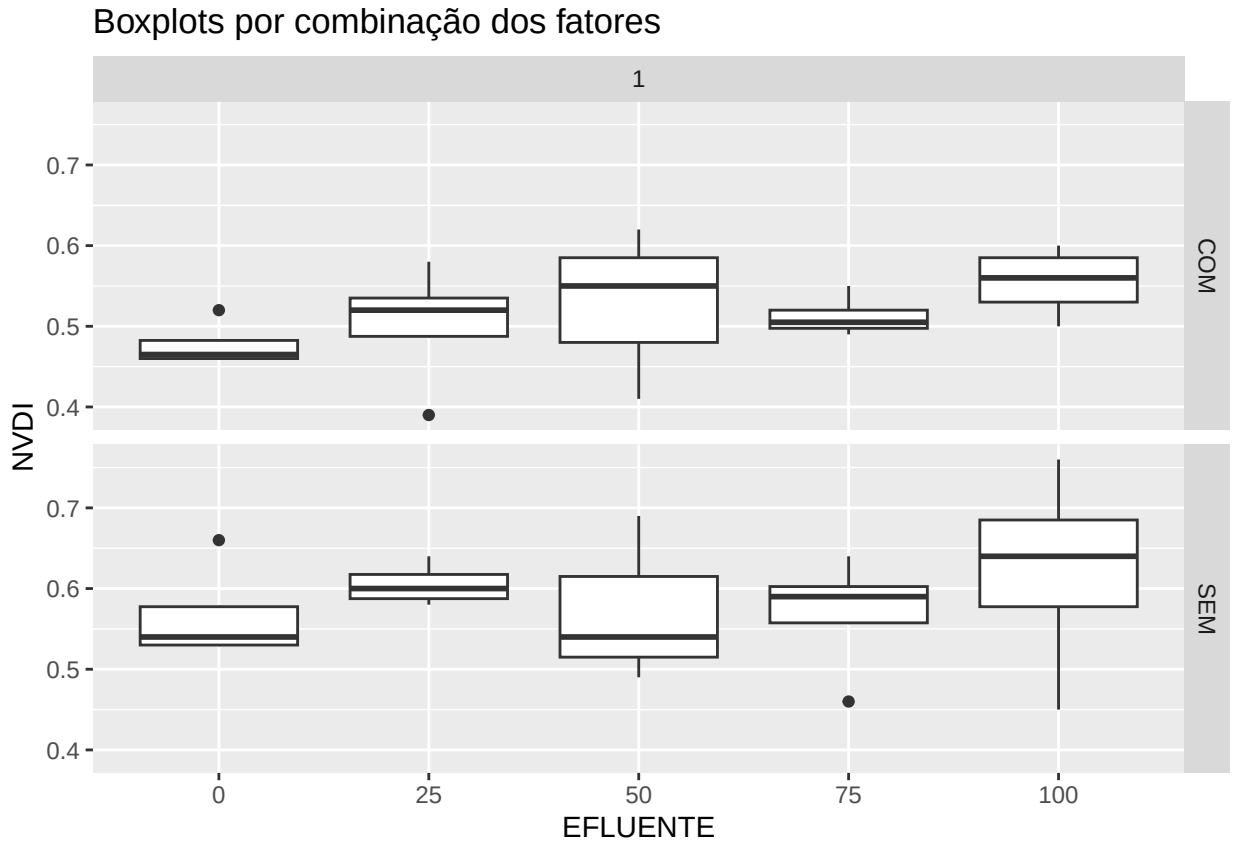
```
## tibble [80 x 5] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: Factor w/ 5 levels "0","25","50",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ INOCULO : Factor w/ 2 levels "COM","SEM": 1 2 1 2 1 2 1 2 1 2 ...
##  $ CICLO   : Factor w/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

```
# Remover os dados dos períodos de tempo onde não houve coleta em um dos
# Ciclos
dados_tempo_1 = subset(dados_tempo_1, CICLO == 1)
dados_tempo_2 = subset(dados_tempo_2, CICLO == 1)
dados_tempo_5 = subset(dados_tempo_5, CICLO == 2)
```

Análise para 37 dias

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

```
## Warning: Removed 2 rows containing non-finite values (`stat_boxplot()`).
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_1, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_inferior <- q[1] - 1.5 * iqr
})

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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_1, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})

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

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

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

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

# Reordenar os dados do tempo 1 seguindo a combinação de fatores por cada bloco
blocos_dados_tempo_1 <- with(dados_tempo_1,
                             dados_tempo_1[order(CICLO, INOCULO, EFLUENTE), ])

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

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

```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 ...
##  $ NVDI    : num [1:40] 0.52 0.47 0.46 0.46 0.58 0.52 0.52 0.39 0.62 0.55 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1:40   Min.   :0.3900
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:0.5000
##  3:10   50 :8                    Median :0.5450
##  4:10   75 :8                    Mean   :0.5513
##         100:8                    3rd Qu.:0.5925
##                                  Max.   :0.7600
```

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

```
##          COM      SEM
## 0   1.910000 2.270000
## 25  2.010000 2.420000
## 50  2.106667 2.293333
## 75  2.050000 2.280000
## 100 2.220000 2.490000
```

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

```
##           COM       SEM
## 0   0.4775000 0.5675000
## 25  0.5025000 0.6050000
## 50  0.5266667 0.5733333
## 75  0.5125000 0.5700000
## 100 0.5550000 0.6225000
```

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

```
##              COM         SEM
## 0   0.0008250000 0.003891667
## 25  0.0064250000 0.000700000
## 50  0.0076222222 0.007222222
## 75  0.0006916667 0.005933333
## 100 0.0019666667 0.016691667
```

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

```
##            COM        SEM
## 0   0.02872281 0.06238322
## 25  0.08015610 0.02645751
## 50  0.08730534 0.08498366
## 75  0.02629956 0.07702813
## 100 0.04434712 0.12919623
```

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

```
## Tables of means
## Grand mean
##
## 0.55125
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 0.549 0.563 0.525 0.568
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.5225 0.5538 0.5500 0.5412 0.5888
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.5148 0.5877
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.4775 0.5675
##      25  0.5025 0.6050
##      50  0.5267 0.5733
##      75  0.5125 0.5700
##      100 0.5550 0.6225
```

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

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

```
## [1] 0.8688227
```

```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$NVDI,
```

```
                                    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] 0.9666807
```

```
# 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(NVDI ~ 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 0.01113 0.003709
##
## Error: INOCULO
##          Df  Sum Sq Mean Sq
## INOCULO   1 0.05305 0.05305
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.03156 0.01052
##
## Error: Within
##                  Df  Sum Sq  Mean Sq F value Pr(>F)
## EFLUENTE          4 0.01872 0.004681   0.992  0.431
## INOCULO:EFLUENTE  4 0.00425 0.001062   0.225  0.922
## Residuals        24 0.11322 0.004718
```

```
modelo = modelo_PARCELASUB
```

```
# Refazer o modelo sem o erro, apenas para facilitar as funções subsequentes
modelo = aov(NVDI ~ 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: NVDI
##          Df   Sum Sq  Mean Sq F value   Pr(>F)
## INOCULO   1 0.053047 0.053047  9.8927 0.004012 **
## 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)
```

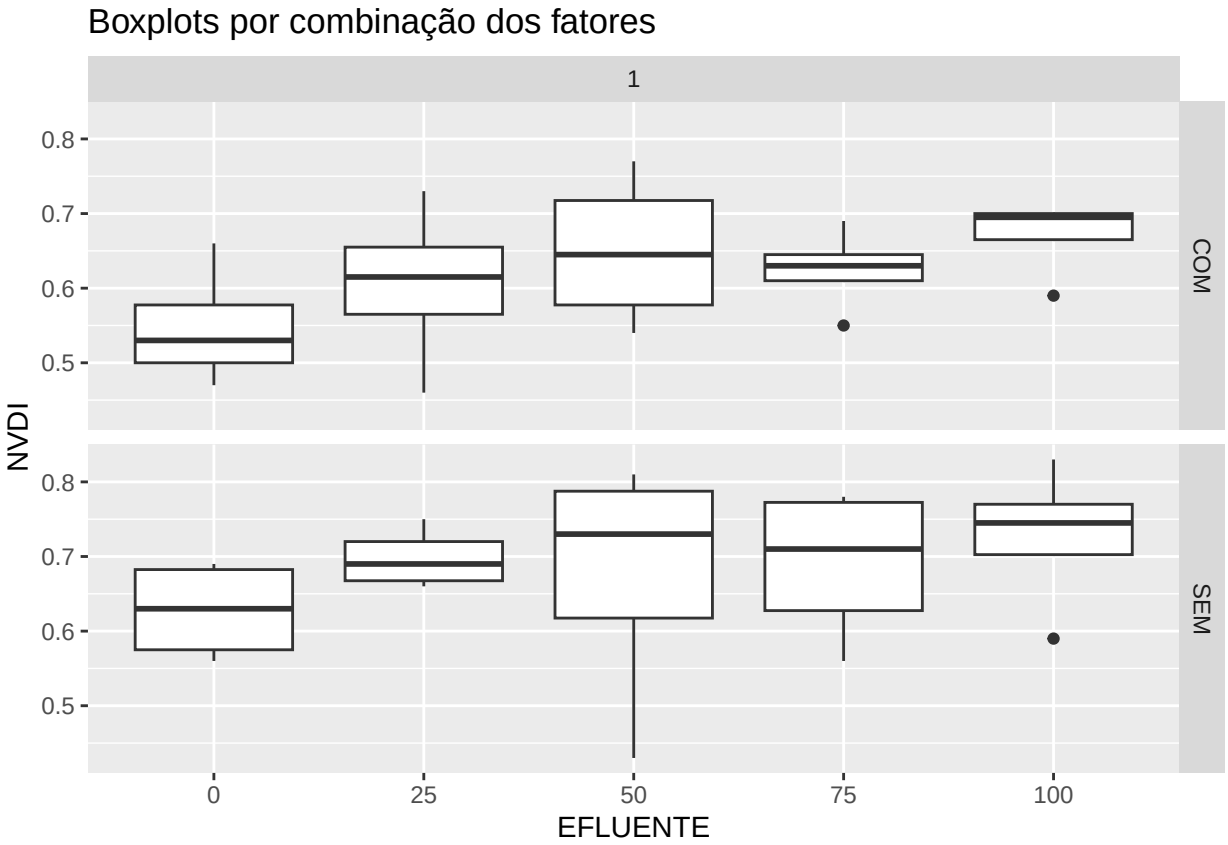
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

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




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

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

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

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

```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 ...
##  $ NVDI    : num [1:40] 0.66 0.55 0.51 0.47 0.73 0.63 0.6 0.46 0.77 0.7 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1:40   Min.   :0.4300
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:0.5975
##  3:10   50 :8                    Median :0.6800
##  4:10   75 :8                    Mean   :0.6613
##         100:8                    3rd Qu.:0.7325
##                                  Max.   :0.8300
```

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

```
##          COM      SEM
## 0   2.190000 2.510000
## 25  2.420000 2.790000
## 50  2.600000 2.700000
## 75  2.600000 2.760000
## 100 2.786667 3.093333
```

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

```
##           COM       SEM
## 0   0.5475000 0.6275000
## 25  0.6050000 0.6975000
## 50  0.6500000 0.6750000
## 75  0.6500000 0.6900000
## 100 0.6966667 0.7733333
```

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

```
##              COM         SEM
## 0   6.691667e-03 0.004491667
## 25  1.243333e-02 0.001691667
## 50  1.086667e-02 0.029766667
## 75  8.000000e-04 0.011000000
## 100 2.222222e-05 0.001622222
```

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

```
##             COM        SEM
## 0   0.081802608 0.06701990
## 25  0.111504858 0.04112988
## 50  0.104243305 0.17253019
## 75  0.028284271 0.10488088
## 100 0.004714045 0.04027682
```

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

```
## Tables of means
## Grand mean
##
## 0.66125
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.7183 0.6967 0.5990 0.6310
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.5875 0.6513 0.6625 0.6700 0.7350
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.6298 0.6927
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.5475 0.6275
##      25  0.6050 0.6975
##      50  0.6500 0.6750
##      75  0.6500 0.6900
##      100 0.6967 0.7733
```

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

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

```
## [1] 0.8994593
```

```
# 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$NVDI,
```

```
                                    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] 0.855415
```

```
# 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(NVDI ~ 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 0.09303 0.03101
##
## Error: INOCULO
##         Df  Sum Sq Mean Sq
## INOCULO  1 0.03948 0.03948
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 0.03697 0.01232
##
## Error: Within
##                  Df  Sum Sq  Mean Sq F value  Pr(>F)
## EFLUENTE          4 0.08845 0.022112   4.907 0.00492 **
## INOCULO:EFLUENTE  4 0.00664 0.001659   0.368 0.82884
## Residuals        24 0.10816 0.004507
## ---
## 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(NVDI ~ 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: NVDI
##           Df  Sum Sq  Mean Sq F value   Pr(>F)
## BLOCO      3 0.09303 0.031010  5.7691 0.003492 **
## INOCULO    1 0.03948 0.039480  7.3450 0.011542 *
## EFLUENTE   4 0.08845 0.022112  4.1139 0.009909 **
## 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(NVDI ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_2, FUN = mean)
    print(media_interacao)
  }

}
```

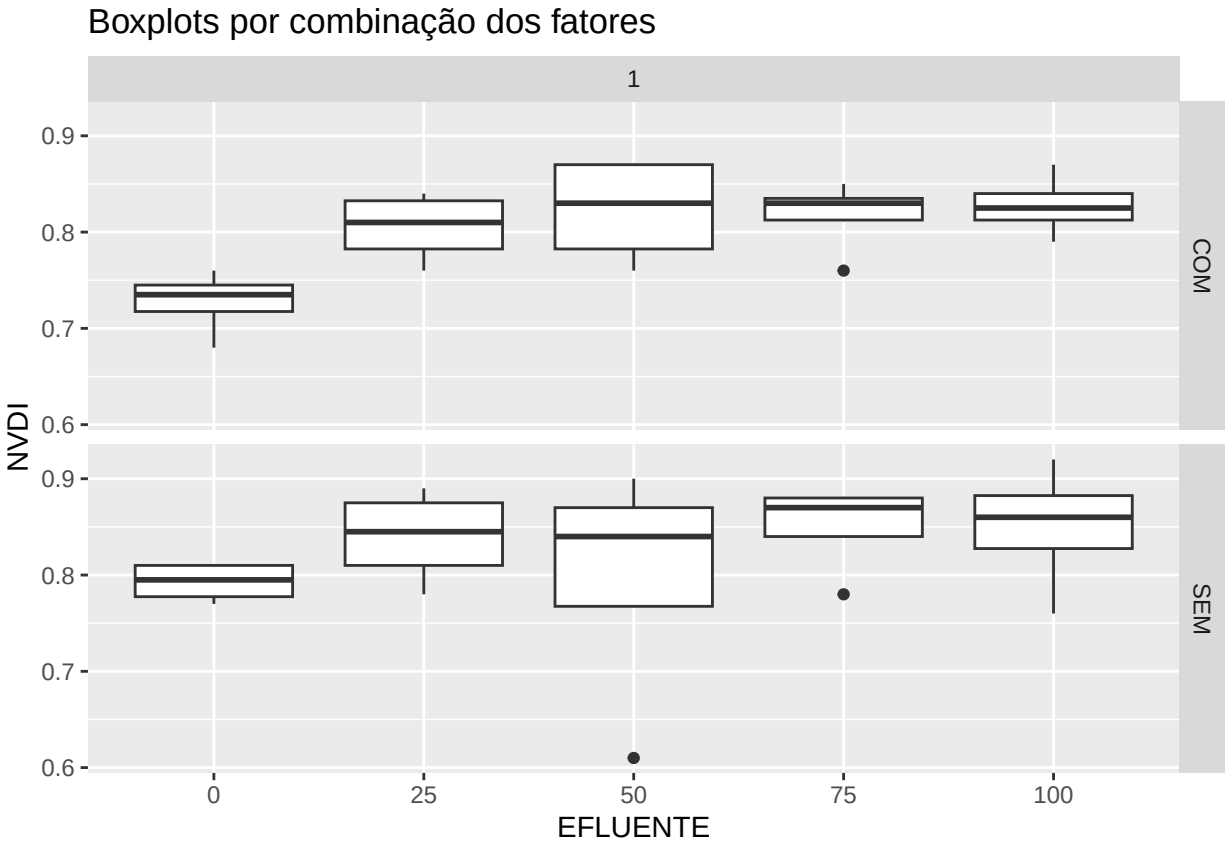
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

CICLO 1

```
dados_tempo_3_ciclo_1 = subset(dados_tempo_3, CICLO == 1)
```

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_3_ciclo_1, aes(x = factor(EFLUENTE), y = NVDI)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "NVDI") +
  ggtitle("Boxplots por combinação dos fatores")
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_3_ciclo_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$NVDI

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_3_ciclo_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$NVDI

# 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_ciclo_1,
                             dados_tempo_3_ciclo_1[order(CICLO, INOCULO, EFLUENTE), ])

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

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

# Criar dataframe com os dados atualizados para serem mesclados no final
# Mudar nome das colunas dos dados atualizados
dados_final_ciclo_1 = cbind(blocos_dados_tempo_1, blocos_dados_tempo_2["NVDI"],
                    blocos_dados_tempo_3["NVDI"])

# Mudar o nome das colunas
colnames(dados_final_ciclo_1)[5:7] = c("37 DAS", "43 DAS", "52 E 54 DAS")
```

```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 ...
##  $ NVDI    : num [1:40] 0.76 0.73 0.74 0.68 0.84 0.83 0.79 0.76 0.87 0.87 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1:40   Min.   :0.6800
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:0.7875
##  3:10   50 :8                   Median :0.8300
##  4:10   75 :8                   Mean   :0.8235
##         100:8                   3rd Qu.:0.8700
##                                 Max.   :0.9200
```

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

```
##          COM      SEM
## 0   2.910000 3.170000
## 25  3.220000 3.360000
## 50  3.290000 3.440000
## 75  3.346667 3.493333
## 100 3.310000 3.400000
```

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

```
##           COM       SEM
## 0   0.7275000 0.7925000
## 25  0.8050000 0.8400000
## 50  0.8225000 0.8600000
## 75  0.8366667 0.8733333
## 100 0.8275000 0.8500000
```

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

```
##              COM          SEM
## 0   1.158333e-03 4.250000e-04
## 25  1.366667e-03 2.466667e-03
## 50  3.158333e-03 1.066667e-03
## 75  8.888889e-05 8.888889e-05
## 100 1.091667e-03 4.466667e-03
```

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

```
##            COM        SEM
## 0   0.03403430 0.02061553
## 25  0.03696846 0.04966555
## 50  0.05619905 0.03265986
## 75  0.00942809 0.00942809
## 100 0.03304038 0.06683313
```

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

```
## Tables of means
## Grand mean
##
## 0.8235
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.8230 0.8367 0.8083 0.8260
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.7600 0.8225 0.8413 0.8550 0.8388
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.8038 0.8432
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.7275 0.7925
##      25  0.8050 0.8400
##      50  0.8225 0.8600
##      75  0.8367 0.8733
##      100 0.8275 0.8500
```

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

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

```
## [1] 0.2128613
```

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

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

```
## [1] 0.9332752
```

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

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

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(NVDI ~ 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 0.004099 0.001366
##
## Error: INOCULO
##         Df  Sum Sq Mean Sq
## INOCULO  1 0.01547 0.01547
##
## Error: BLOCO:INOCULO
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## Residuals  3 0.01056 0.003519
##
## Error: Within
##                  Df  Sum Sq  Mean Sq F value   Pr(>F)
```

```
## EFLUENTE          4 0.04458 0.011146   8.499 0.000203 ***
## INOCULO:EFLUENTE  4 0.00194 0.000486   0.370 0.827387
## Residuals        24 0.03148 0.001312
## ---
## 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(NVDI ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_3)

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

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

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

```
## Analysis of Variance Table
##
## Response: NVDI
##           Df   Sum Sq  Mean Sq F value    Pr(>F)
## INOCULO    1 0.015471 0.015471  9.9376 0.0039418 **
## EFLUENTE   4 0.044585 0.011146  7.1596 0.0004583 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

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

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

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

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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(NVDI ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_3, FUN = mean)
    print(media_interacao)
  }

}
```

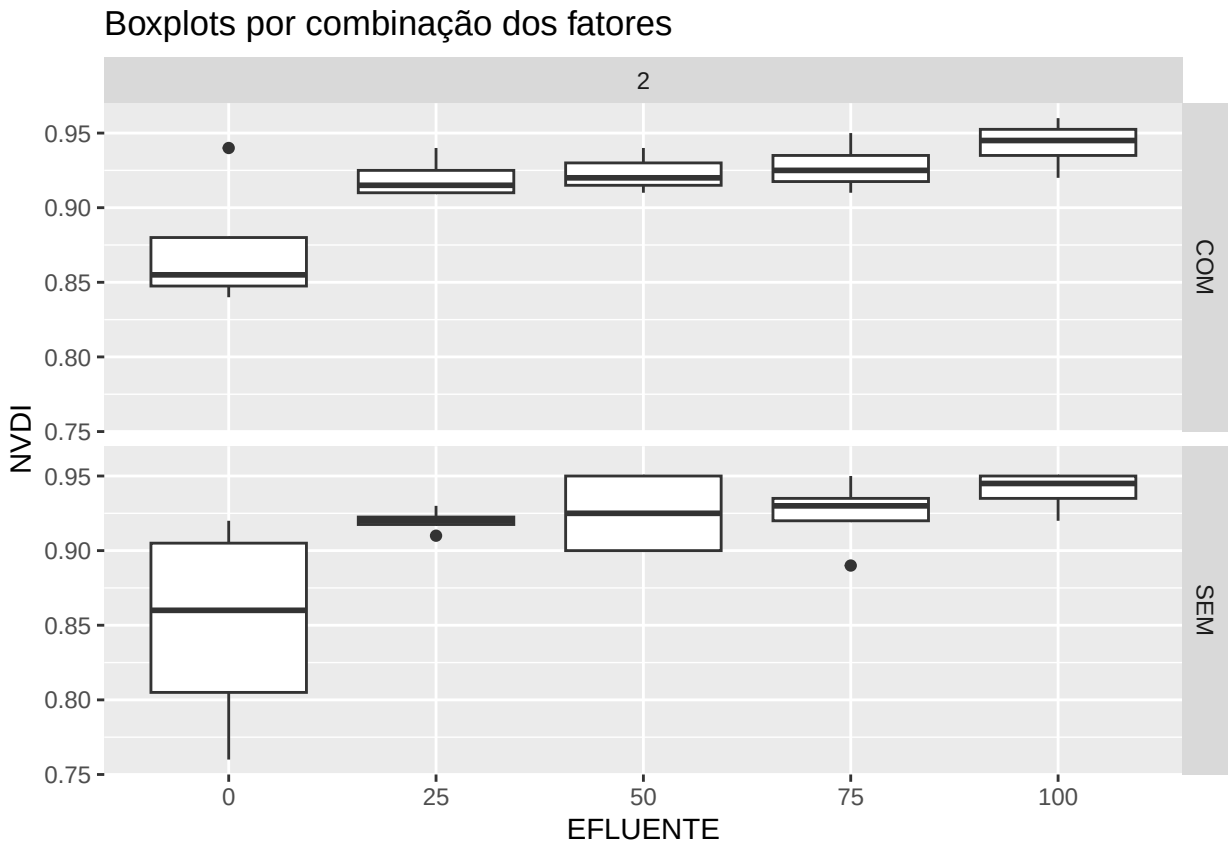
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

CICLO 2

```
dados_tempo_3_ciclo_2 = subset(dados_tempo_3, CICLO == 2)
```

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_3_ciclo_2, aes(x = factor(EFLUENTE), y = NVDI)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "NVDI") +
  ggtitle("Boxplots por combinação dos fatores")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_3_ciclo_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$NVDI

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_3_ciclo_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$NVDI
```

```r
# 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

# 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_ciclo_2,
                             dados_tempo_3_ciclo_2[order(CICLO, INOCULO, EFLUENTE), ])

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

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

# Criar dataframe final do ciclo 2
dados_final_ciclo_2 = dados_final_ciclo_1[1:4]
dados_final_ciclo_2$CICLO = 2
dados_final_ciclo_2$`52 E 54 DAS` = blocos_dados_tempo_3$NVDI
```

```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 ...
##  $ NVDI    : num [1:40] 0.85 0.85 0.84 0.86 0.92 0.91 0.91 0.94 0.91 0.94 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1: 0   Min.   :0.7600
##  2:10   25 :8    SEM:20   2:40   1st Qu.:0.9100
##  3:10   50 :8                   Median :0.9200
##  4:10   75 :8                   Mean   :0.9138
##         100:8                   3rd Qu.:0.9400
##                                 Max.   :0.9600
```

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

```
##          COM      SEM
## 0   3.400000 3.400000
## 25  3.680000 3.693333
## 50  3.693333 3.700000
## 75  3.710000 3.746667
## 100 3.770000 3.760000
```

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

```
##           COM       SEM
## 0   0.8500000 0.8500000
## 25  0.9200000 0.9233333
## 50  0.9233333 0.9250000
## 75  0.9275000 0.9366667
## 100 0.9425000 0.9400000
```

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

```
##              COM          SEM
## 0   6.666667e-05 5.466667e-03
## 25  2.000000e-04 2.222222e-05
## 50  1.555556e-04 8.333333e-04
## 75  2.916667e-04 8.888889e-05
## 100 2.916667e-04 2.000000e-04
```

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

```
##             COM         SEM
## 0   0.008164966 0.073936910
## 25  0.014142136 0.004714045
## 50  0.012472191 0.028867513
## 75  0.017078251 0.009428090
## 100 0.017078251 0.014142136
```

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

```
## Tables of means
## Grand mean
##
## 0.9138333
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.9153 0.9220 0.9007 0.9173
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.8500 0.9217 0.9242 0.9321 0.9412
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.9127 0.9150
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.8500 0.8500
##      25  0.9200 0.9233
##      50  0.9233 0.9250
##      75  0.9275 0.9367
##      100 0.9425 0.9400
```

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

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

```
## [1] 0.0009875599
```

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_3$NVDI,
                                    media_blocos_tempo_3$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
```

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

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

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(NVDI ~ 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 0.002546 0.0008485
##
## Error: INOCULO
##         Df    Sum Sq   Mean Sq
## INOCULO  1 5.444e-05 5.444e-05
##
## Error: BLOCO:INOCULO
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## Residuals  3 0.002386 0.0007952
##
## Error: Within
##                  Df  Sum Sq  Mean Sq F value   Pr(>F)
```

```
## EFLUENTE          4 0.04262 0.010655  14.271 4.27e-06 ***
## INOCULO:EFLUENTE  4 0.00015 0.000038   0.052    0.995
## Residuals        24 0.01792 0.000747
## ---
## 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(NVDI ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_3)

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

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

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

```
## Analysis of Variance Table
##
## Response: NVDI
##           Df   Sum Sq  Mean Sq F value    Pr(>F)
## EFLUENTE  4 0.042621 0.010655  14.169 2.369e-06 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

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

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

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

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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(NVDI ~ 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 63 e 77 dias

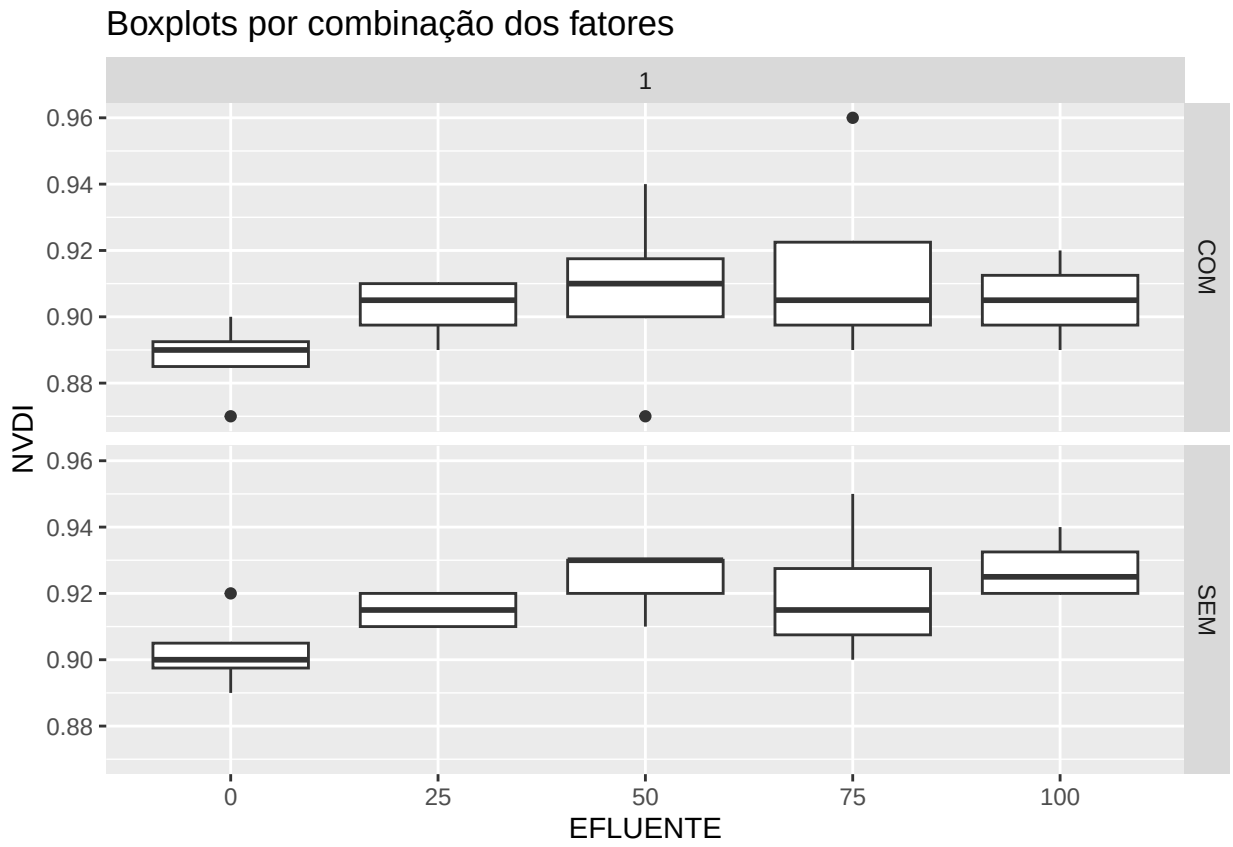
CICLO 1

```
dados_tempo_4_ciclo_1 = subset(dados_tempo_4, CICLO == 1)
```

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_4_ciclo_1, aes(x = factor(EFLUENTE), y = NVDI)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
```

```
  labs(x = "EFLUENTE", y = "NVDI") +
  ggtitle("Boxplots por combinação dos fatores")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```




```
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_4_ciclo_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$NVDI

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_4_ciclo_1, FUN = function(x) {
  q <- quantile(x, c(0.25, 0.75), na.rm = TRUE)
  iqr <- q[2] - q[1]
  limite_superior <- q[2] + 1.5 * iqr
})
```

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

# 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

# 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_ciclo_1,
                             dados_tempo_4_ciclo_1[order(CICLO, INOCULO, EFLUENTE), ])

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

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

# Incluir coluna 4 no ciclo 1
dados_final_ciclo_1$`63 E 77 DAS` = blocos_dados_tempo_4$NVDI
```

```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 ...
##  $ NVDI    : num [1:40] 0.9 0.89 0.89 0.893 0.91 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1:40   Min.   :0.8900
##  2:10   25 :8    SEM:20   2: 0   1st Qu.:0.9000
##  3:10   50 :8                    Median :0.9100
##  4:10   75 :8                    Mean   :0.9103
##         100:8                    3rd Qu.:0.9200
##                                  Max.   :0.9500
```

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

```
##          COM      SEM
## 0   3.573333 3.586667
## 25  3.610000 3.660000
## 50  3.680000 3.693333
## 75  3.600000 3.680000
## 100 3.620000 3.710000
```

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

```
##           COM       SEM
## 0   0.8933333 0.8966667
## 25  0.9025000 0.9150000
## 50  0.9200000 0.9233333
## 75  0.9000000 0.9200000
## 100 0.9050000 0.9275000
```

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

```
##              COM          SEM
## 0   2.222222e-05 2.222222e-05
## 25  9.166667e-05 3.333333e-05
## 50  2.000000e-04 8.888889e-05
## 75  6.666667e-05 4.666667e-04
## 100 1.666667e-04 9.166667e-05
```

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

```
##             COM         SEM
## 0   0.004714045 0.004714045
## 25  0.009574271 0.005773503
## 50  0.014142136 0.009428090
## 75  0.008164966 0.021602469
## 100 0.012909944 0.009574271
```

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

```
## Tables of means
## Grand mean
##
## 0.9103333
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.9160 0.9057 0.9103 0.9093
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.8950 0.9087 0.9217 0.9100 0.9162
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.9042 0.9165
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.8933 0.8967
##      25  0.9025 0.9150
##      50  0.9200 0.9233
##      75  0.9000 0.9200
##      100 0.9050 0.9275
```

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

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

```
## [1] 0.3738848
```

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_4$NVDI,
                                    media_blocos_tempo_4$EFLUENTE,
                                    media_blocos_tempo_4$INOCULO,
                                    media_blocos_tempo_4$CICLO)$p.value

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

```
## [1] 0.4874601
```

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

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

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(NVDI ~ 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 0.0005489 0.000183
##
## Error: INOCULO
##         Df   Sum Sq  Mean Sq
## INOCULO  1 0.001521 0.001521
##
## Error: BLOCO:INOCULO
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## Residuals  3 5.889e-05 1.963e-05
##
## Error: Within
##                  Df   Sum Sq   Mean Sq F value  Pr(>F)
```

```
## EFLUENTE          4 0.003209 0.0008024   6.128 0.00152 **
## INOCULO:EFLUENTE  4 0.000648 0.0001621   1.238 0.32141
## Residuals        24 0.003142 0.0001309
## ---
## 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(NVDI ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_4)

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

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

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

```
## Analysis of Variance Table
##
## Response: NVDI
##           Df    Sum Sq    Mean Sq F value    Pr(>F)
## INOCULO    1 0.0015211 0.00152111 12.8299 0.0013225 **
## EFLUENTE   4 0.0032094 0.00080236  6.7676 0.0006581 ***
## NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

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

# Exibir o resultado
print(interacoes_significativas)
```

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

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

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

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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(NVDI ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_4, FUN = mean)
    print(media_interacao)
  }

}
```

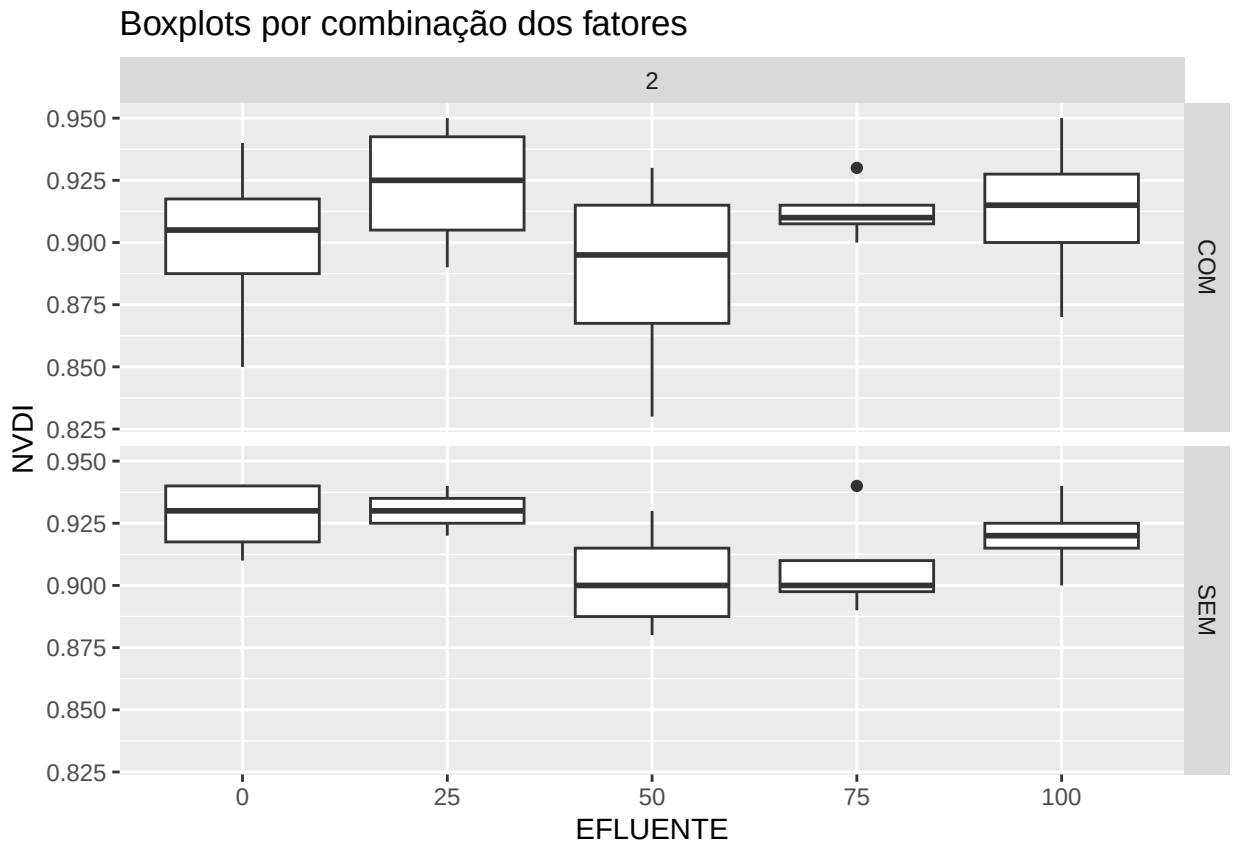
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

CICLO 2

```
dados_tempo_4_ciclo_2 = subset(dados_tempo_4, CICLO == 2)
```

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_tempo_4_ciclo_2, aes(x = factor(EFLUENTE), y = NVDI)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "NVDI") +
  ggtitle("Boxplots por combinação dos fatores")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```




```r
# Calcular os limites inferiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_4_ciclo_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$NVDI

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(NVDI ~ EFLUENTE + INOCULO + CICLO,
                              data = dados_tempo_4_ciclo_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$NVDI
```

```r
# 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

# 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_ciclo_2,
                             dados_tempo_4_ciclo_2[order(CICLO, INOCULO, EFLUENTE), ])

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

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

# Incluir coluna 4 no ciclo 2
dados_final_ciclo_2$`63 E 77 DAS` = blocos_dados_tempo_4$NVDI
```

```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 ...
##  $ NVDI    : num [1:40] 0.94 0.85 0.9 0.91 0.91 0.89 0.94 0.95 0.93 0.83 ...
```

```r
summary(blocos_dados_tempo_4)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1: 0   Min.   :0.8300
##  2:10   25 :8    SEM:20   2:40   1st Qu.:0.9000
##  3:10   50 :8                   Median :0.9100
##  4:10   75 :8                   Mean   :0.9106
##         100:8                   3rd Qu.:0.9300
##                                 Max.   :0.9500
```

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

```
##          COM      SEM
## 0   3.600000 3.710000
## 25  3.690000 3.720000
## 50  3.550000 3.610000
## 75  3.626667 3.586667
## 100 3.650000 3.680000
```

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

```
##           COM       SEM
## 0   0.9000000 0.9275000
## 25  0.9225000 0.9300000
## 50  0.8875000 0.9025000
## 75  0.9066667 0.8966667
## 100 0.9125000 0.9200000
```

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

```
##              COM          SEM
## 0   1.400000e-03 2.250000e-04
## 25  7.583333e-04 6.666667e-05
## 50  1.891667e-03 4.916667e-04
## 75  2.222222e-05 2.222222e-05
## 100 1.091667e-03 2.666667e-04
```

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

```
##             COM         SEM
## 0   0.037416574 0.015000000
## 25  0.027537853 0.008164966
## 50  0.043493295 0.022173558
## 75  0.004714045 0.004714045
## 100 0.033040379 0.016329932
```

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

```
## Tables of means
## Grand mean
##
## 0.9105833
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.9227 0.8947 0.9130 0.9120
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.9137 0.9262 0.8950 0.9017 0.9162
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.9058 0.9153
##
##  EFLUENTE:INOCULO
##          INOCULO
## EFLUENTE COM    SEM
##      0   0.9000 0.9275
##      25  0.9225 0.9300
##      50  0.8875 0.9025
##      75  0.9067 0.8967
##      100 0.9125 0.9200
```

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

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

```
## [1] 0.7698492
```

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_4$NVDI,
                                    media_blocos_tempo_4$EFLUENTE,
                                    media_blocos_tempo_4$INOCULO,
                                    media_blocos_tempo_4$CICLO)$p.value

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

```
## [1] 0.7531175
```

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

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

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(NVDI ~ 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 0.004072 0.001357
##
## Error: INOCULO
##         Df    Sum Sq   Mean Sq
## INOCULO  1 0.0009025 0.0009025
##
## Error: BLOCO:INOCULO
##           Df   Sum Sq  Mean Sq F value Pr(>F)
## Residuals  3 0.006516 0.002172
##
## Error: Within
##                  Df   Sum Sq   Mean Sq F value Pr(>F)
```

```
## EFLUENTE          4 0.004879 0.0012199   3.606 0.0194 *
## INOCULO:EFLUENTE  4 0.001485 0.0003713   1.097 0.3805
## Residuals        24 0.008120 0.0003383
## ---
## 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(NVDI ~ BLOCO + INOCULO * EFLUENTE,
             data = blocos_dados_tempo_4)

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

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

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

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

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

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

# Exibir o resultado
print(interacoes_significativas)
```

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

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

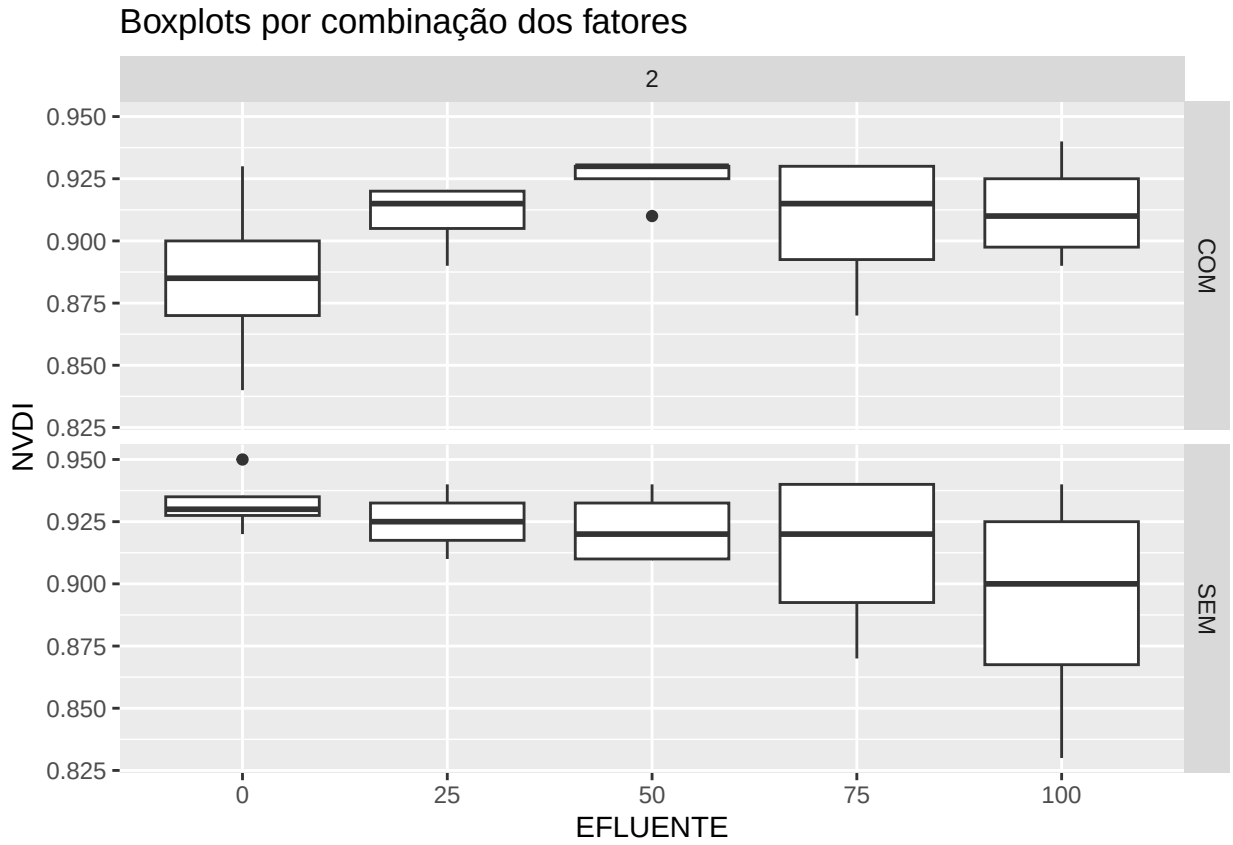
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

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

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(NVDI ~ 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 85 dias

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




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

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

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

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

# Incluir coluna 5 no ciclo 2
dados_final_ciclo_2$`85 DAS` = blocos_dados_tempo_5$NVDI
```

```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": 2 2 2 2 2 2 2 2 2 2 ...
##  $ NVDI    : num [1:40] 0.93 0.89 0.88 0.84 0.92 0.89 0.91 0.92 0.93 0.93 ...
```

```r
summary(blocos_dados_tempo_5)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       NVDI
##  1:10   0  :8    COM:20   1: 0   Min.   :0.8300
##  2:10   25 :8    SEM:20   2:40   1st Qu.:0.9000
##  3:10   50 :8                   Median :0.9200
```

```
##  4:10   75 :8                    Mean   :0.9124
##         100:8                    3rd Qu.:0.9300
##                                  Max.   :0.9400
```

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

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

```
with(blocos_dados_tempo_5, tapply(NVDI, list(EFLUENTE, INOCULO), sum))
```

```
##      COM      SEM
## 0   3.54 3.706667
## 25  3.64 3.700000
## 50  3.72 3.690000
## 75  3.63 3.650000
## 100 3.65 3.570000
```

```
with(blocos_dados_tempo_5, tapply(NVDI, list(EFLUENTE, INOCULO), mean))
```

```
##        COM       SEM
## 0   0.8850 0.9266667
## 25  0.9100 0.9250000
## 50  0.9300 0.9225000
## 75  0.9075 0.9125000
## 100 0.9125 0.8925000
```

```
with(blocos_dados_tempo_5, tapply(NVDI, list(EFLUENTE, INOCULO), var))
```

```
##              COM          SEM
## 0   0.0013666667 2.222222e-05
## 25  0.0002000000 1.666667e-04
## 50  0.0000000000 2.250000e-04
## 75  0.0008250000 1.158333e-03
## 100 0.0004916667 2.358333e-03
```

```
with(blocos_dados_tempo_5, tapply(NVDI, list(EFLUENTE, INOCULO), sd))
```

```
##            COM         SEM
## 0   0.03696846 0.004714045
## 25  0.01414214 0.012909944
## 50  0.00000000 0.015000000
## 75  0.02872281 0.034034296
## 100 0.02217356 0.048562674
```

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

```
## Tables of means
## Grand mean
##
## 0.9124167
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 0.9297 0.9220 0.9010 0.8970
##
##  EFLUENTE
## EFLUENTE
##      0     25     50     75    100
## 0.9058 0.9175 0.9262 0.9100 0.9025
##
##  INOCULO
## INOCULO
##    COM    SEM
## 0.9090 0.9158
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM    SEM
##      0   0.8850 0.9267
##      25  0.9100 0.9250
##      50  0.9300 0.9225
##      75  0.9075 0.9125
##      100 0.9125 0.8925
```

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

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

```
## [1] 0.3499601
```

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

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_5$NVDI,
                                    media_blocos_tempo_5$EFLUENTE,
                                    media_blocos_tempo_5$INOCULO,
                                    media_blocos_tempo_5$CICLO)$p.value

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

```
## [1] 0.4843411
```

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

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

```r
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(NVDI ~ 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 0.007574 0.002525
##
## Error: INOCULO
##         Df    Sum Sq   Mean Sq
## INOCULO  1 0.0004669 0.0004669
##
## Error: BLOCO:INOCULO
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## Residuals  3 0.001194 0.0003981
##
## Error: Within
##                   Df   Sum Sq   Mean Sq F value Pr(>F)
## EFLUENTE           4 0.002918 0.0007294   1.500 0.2337
## INOCULO:EFLUENTE   4 0.004418 0.0011044   2.271 0.0912 .
## Residuals         24 0.011673 0.0004864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
modelo = modelo_PARCELASUB
```

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

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

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

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

```
## Analysis of Variance Table
##
## Response: NVDI
##       Df    Sum Sq   Mean Sq F value   Pr(>F)
## BLOCO  3 0.0075742 0.0025247  5.2976 0.005291 **
## 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(NVDI ~ 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)
}
```

```
# Criar planilha com todos os dados atualizados
library("xlsx")
```

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

```
write.xlsx(dados_final_ciclo_1, file = "NVDI atualizado - Ciclo 1.xlsx",
      sheetName = "R - NVDI_1", append = FALSE)
write.xlsx(dados_final_ciclo_2, file = "NVDI atualizado - Ciclo 2.xlsx",
      sheetName = "R - NVDI_2", append = FALSE)
```