Biométricas Altura

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)
```

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

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

# Converter as colunas para tipo numérico e arredondar valores em duas casas
for (i in 5:7) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
dados[5:7] =  round(dados[5:7], digits = 2)
str(dados)
```

```
## tibble [80 x 7] (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 ...
##  $ 43 E 41 DAS: num [1:80] 36.3 33.2 34.1 37.9 40.1 ...
##  $ 50 E 49 DAS: num [1:80] 51.4 49 49.7 52.9 53.3 ...
##  $ 64 E 76 DAS: num [1:80] 77.9 78.5 75.3 80.5 84.7 ...
```

```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
```

```
# Separar os dados de acordo com o período de tempo da coleta
dados_tempo_1 = dados[c(1:5)] # 43, 41
dados_tempo_1$ALTURA = dados_tempo_1$`43 E 41 DAS`
dados_tempo_1 = dados_tempo_1[-5]
dados_tempo_2 = dados[c(1:4,6)] # 50, 49
dados_tempo_2$ALTURA = dados_tempo_2$`50 E 49 DAS`
dados_tempo_2 = dados_tempo_2[-5]
dados_tempo_3 = dados[c(1:4,7)] # 64, 76
dados_tempo_3$ALTURA = dados_tempo_3$`64 E 76 DAS`
dados_tempo_3 = dados_tempo_3[-5]

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

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

```
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 ...
##  $ ALTURA  : num [1:80] 36.3 33.2 34.1 37.9 40.1 ...
```

```
"dados_tempo_2"
```

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

```
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 ...
##  $ ALTURA  : num [1:80] 51.4 49 49.7 52.9 53.3 ...
```

```
"dados_tempo_3"
```

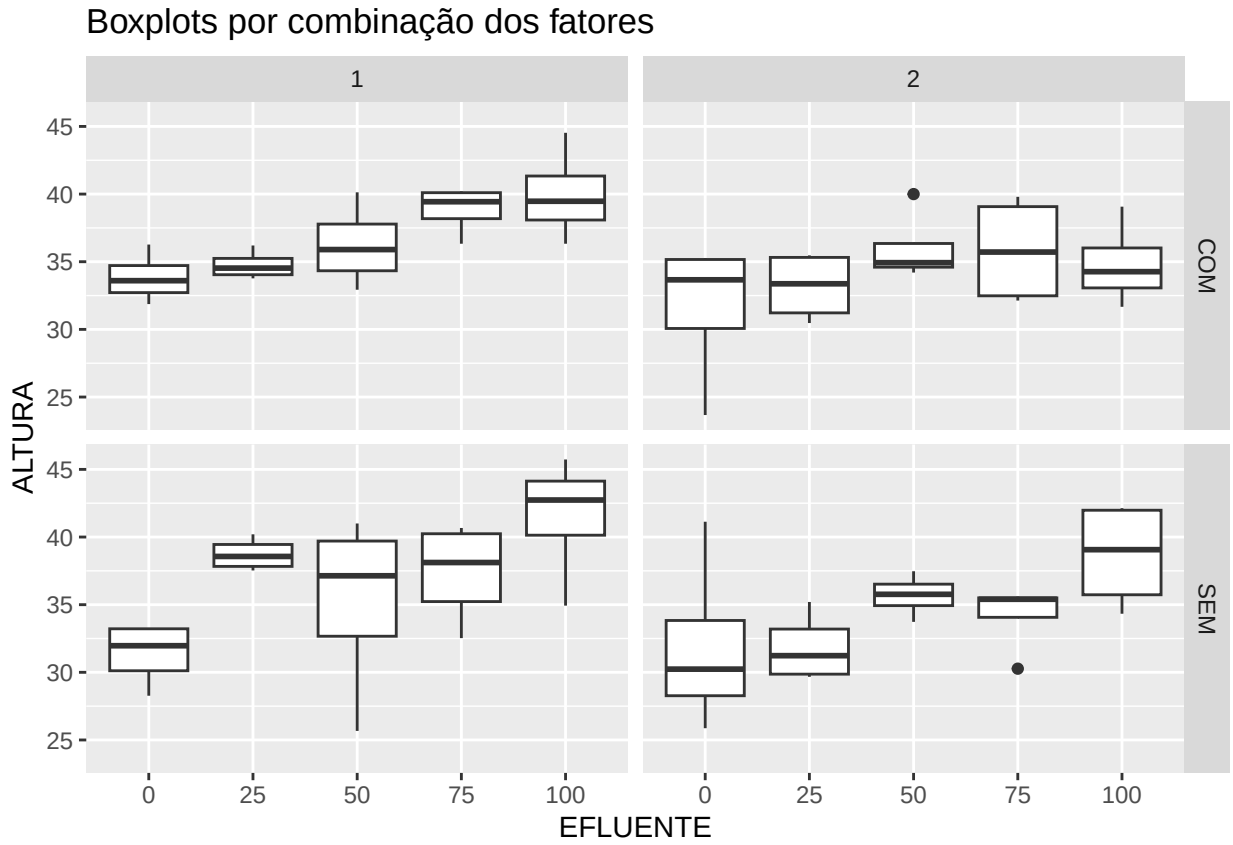
```
## [1] "dados_tempo_3"
```

```
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 ...
##  $ ALTURA  : num [1:80] 77.9 78.5 75.3 80.5 84.7 ...
```

Análise para 43 e 41 dias

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




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

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

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

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

```r
# Análises Descritivas
str(blocos_dados_tempo_1)
```

```
## tibble [80 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 ...
##  $ ALTURA  : num [1:80] 36.3 31.9 34.2 33 34.1 ...
```

```r
summary(blocos_dados_tempo_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ALTURA
##  1:20   0  :16   COM:40   1:40   Min.   :23.67
##  2:20   25 :16   SEM:40   2:40   1st Qu.:33.15
##  3:20   50 :16                   Median :35.27
##  4:20   75 :16                   Mean   :35.57
##         100:16                   3rd Qu.:38.81
##                                  Max.   :45.73
```

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

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

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

```
##          COM      SEM
## 0   261.6100 252.9400
## 25  271.7100 282.1900
## 50  283.6067 283.6700
## 75  298.7600 291.2967
## 100 299.0700 320.7200
```

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

```
##          COM      SEM
## 0   32.70125 31.61750
## 25  33.96375 35.27375
## 50  35.45083 35.45875
## 75  37.34500 36.41208
## 100 37.38375 40.09000
```

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

```
##           COM       SEM
## 0   15.724927 21.040993
## 25   4.057113 17.045855
## 50   4.834825 21.265841
## 75  10.952486  7.213206
## 100 16.884027 18.540086
```

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

```
##          COM      SEM
## 0   3.965467 4.587046
## 25  2.014228 4.128663
## 50  2.198823 4.611490
## 75  3.309454 2.685741
## 100 4.109018 4.305820
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_1,
                 model.tables(aov(ALTURA ~ CICLO + BLOCO + EFLUENTE + INOCULO +
```

```
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                                "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 35.56967
##
##  CICLO
## CICLO
##     1     2
## 36.78 34.36
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 36.67 36.08 34.12 35.41
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 32.16 34.62 35.45 36.88 38.74
##
##  INOCULO
## INOCULO
##   COM   SEM
## 35.37 35.77
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 32.60 36.74 35.73 38.10 40.74
##     2 31.72 32.50 35.18 35.65 36.73
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 36.72 36.84
##     2 34.02 34.70
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   32.70 31.62
##      25  33.96 35.27
##      50  35.45 35.46
##      75  37.35 36.41
##      100 37.38 40.09
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
```

```
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 33.84 34.76 36.22 38.85 39.95
##     2 31.57 33.17 34.69 35.84 34.82
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 31.37 38.72 35.24 37.36 41.53
##     2 31.87 31.83 35.68 35.47 38.65
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_1_backup = media_blocos_tempo_1
media_blocos_tempo_1 = aggregate(ALTURA ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_tempo_1, FUN = mean)

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

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

```
## [1] 0.4579127
```

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

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

```
## [1] 0.1326547
```

```
# 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(ALTURA ~ BLOCO + CICLO * 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  71.83   23.94
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  3.224   3.224
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  19.38    6.46
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1  117.5  117.48   9.591   0.0031 **
## EFLUENTE                  4  388.7   97.17   7.932 4.18e-05 ***
## CICLO:INOCULO             1    1.6    1.58   0.129   0.7205
## CICLO:EFLUENTE            4   46.8   11.71   0.956   0.4391
## INOCULO:EFLUENTE          4   41.1   10.28   0.839   0.5064
## CICLO:INOCULO:EFLUENTE    4   44.3   11.08   0.905   0.4679
## Residuals                54  661.5   12.25
## ---
## 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(ALTURA ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_tempo_1)

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

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

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

```
## Analysis of Variance Table
##
## Response: ALTURA
```

```
##           Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO      1 117.48 117.483  9.8355 0.002707 **
## EFLUENTE   4 388.67  97.168  8.1347 2.92e-05 ***
## 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)
```

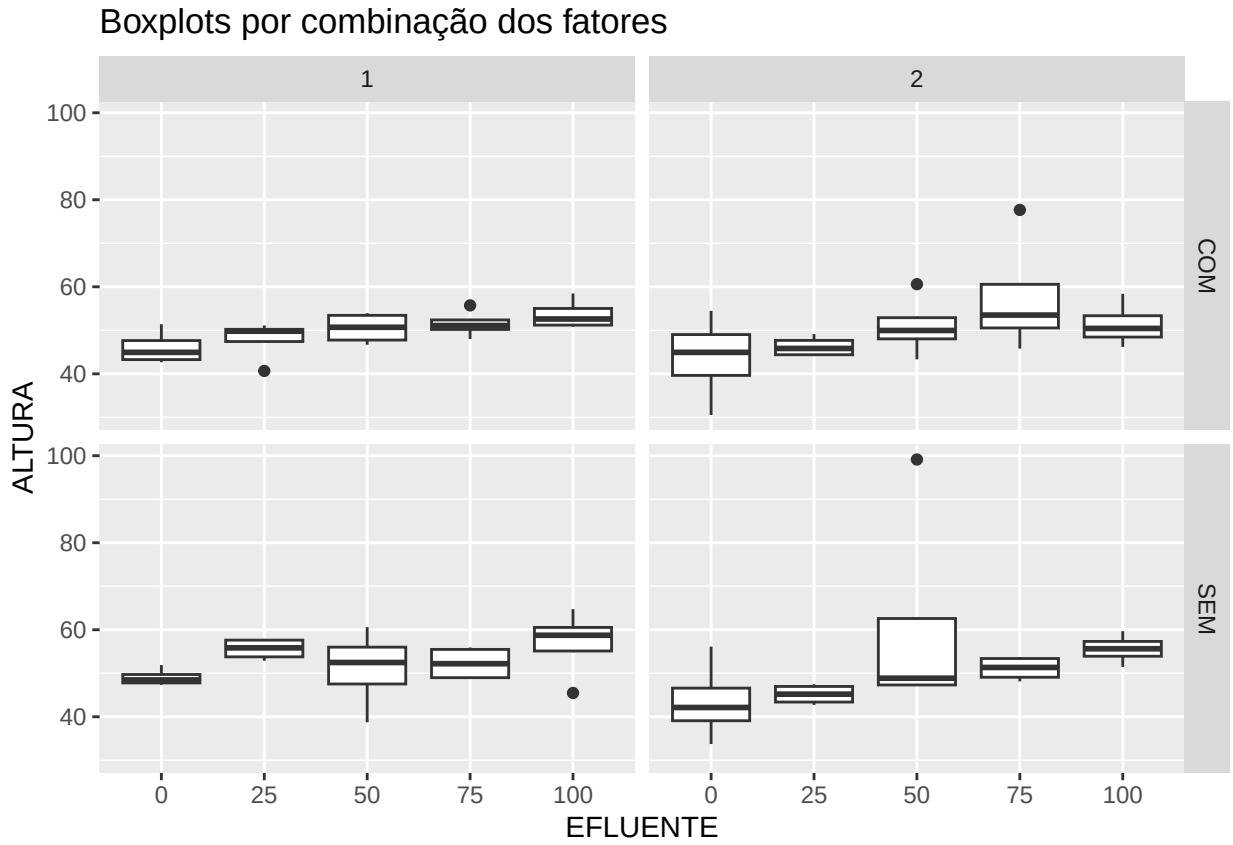
```
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ CICLO + INOCULO,
                                 data = blocos_dados_tempo_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ CICLO + EFLUENTE,
                                 data = blocos_dados_tempo_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_tempo_1, FUN = mean)
```

```
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(ALTURA ~ CICLO + 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 50 e 49 dias

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




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

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

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

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

```r
# Análises Descritivas
str(blocos_dados_tempo_2)
```

```
## tibble [80 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 ...
##  $ ALTURA  : num [1:80] 51.4 46.4 43.5 42.7 49.7 ...
```

```r
summary(blocos_dados_tempo_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ALTURA
##  1:20   0  :16   COM:40   1:40   Min.   :30.53
##  2:20   25 :16   SEM:40   2:40   1st Qu.:47.25
##  3:20   50 :16                   Median :50.16
##  4:20   75 :16                   Mean   :50.14
##         100:16                   3rd Qu.:53.89
##                                  Max.   :64.73
```

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

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

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

```
##          COM      SEM
## 0   358.8100 370.2600
## 25  385.9033 402.8100
## 50  393.0233 397.6033
## 75  404.0000 413.5900
## 100 419.9700 465.3200
```

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

```
##          COM      SEM
## 0   44.85125 46.28250
## 25  48.23792 50.35125
## 50  49.12792 49.70042
## 75  50.50000 51.69875
## 100 52.49625 58.16500
```

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

```
##           COM      SEM
## 0   51.406212 47.79414
## 25   7.177873 35.83678
## 50  12.074768 39.48233
## 75   7.321022 10.20404
## 100 18.250627 16.02820
```

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

```
##          COM      SEM
## 0   7.169813 6.913330
## 25  2.679155 5.986383
## 50  3.474877 6.283496
## 75  2.705739 3.194376
## 100 4.272075 4.003523
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_2,
                 model.tables(aov(ALTURA ~ CICLO + BLOCO + EFLUENTE + INOCULO +
```

```
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                                "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 50.14113
##
##  CICLO
## CICLO
##     1     2
## 51.91 48.38
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 51.69 50.68 48.22 49.97
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 45.57 49.29 49.41 51.10 55.33
##
##  INOCULO
## INOCULO
##   COM   SEM
## 49.04 51.24
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 47.51 52.90 50.79 51.17 57.18
##     2 43.63 45.69 48.04 51.03 53.48
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##     1 50.08 53.73
##     2 48.00 48.75
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   44.85 46.28
##      25  48.24 50.35
##      50  49.13 49.70
##      75  50.50 51.70
##      100 52.50 58.17
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
```

```
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 45.99 50.24 50.50 50.07 53.63
##     2 43.72 46.23 47.75 50.93 51.37
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##     1 49.03 55.55 51.07 52.26 60.73
##     2 43.53 45.15 48.33 51.13 55.60
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_tempo_2_backup = media_blocos_tempo_2
media_blocos_tempo_2 = aggregate(ALTURA ~ 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$ALTURA)$p.value

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

```
## [1] 0.5370337
```

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

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

```r
# Realizar o teste de Bartlett no conjunto de dados completo
resultado_bartlett <- bartlett.test(media_blocos_tempo_2$ALTURA,
                                    media_blocos_tempo_2$EFLUENTE,
                                    media_blocos_tempo_2$INOCULO,
                                    media_blocos_tempo_2$CICLO)$p.value

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

```
## [1] 0.1142217
```

```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."
```

```
# Ajustar o modelo linear para parcelas subdivididas
modelo_PARCELASUB = aov(ALTURA ~ BLOCO + CICLO * 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  128.7   42.91
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  96.53   96.53
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  21.88   7.294
##
## Error: Within
##                         Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                    1  249.5  249.47  11.693   0.0012 **
## EFLUENTE                 4  800.3  200.07   9.378 7.88e-06 ***
## CICLO:INOCULO            1   41.9   41.94   1.966   0.1666
## CICLO:EFLUENTE           4  103.3   25.83   1.211   0.3170
## INOCULO:EFLUENTE         4   65.1   16.28   0.763   0.5539
## CICLO:INOCULO:EFLUENTE   4   21.6    5.40   0.253   0.9064
## Residuals               54 1152.1   21.33
## ---
## 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(ALTURA ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_tempo_2)

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

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

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

```
## Analysis of Variance Table
##
## Response: ALTURA
```

```
##           Df Sum Sq Mean Sq F value    Pr(>F)
## CICLO      1 249.47 249.465 12.1124 0.0009676 ***
## INOCULO    1  96.53  96.529  4.6868 0.0345957 *
## EFLUENTE   4 800.29 200.072  9.7142 4.623e-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)
```

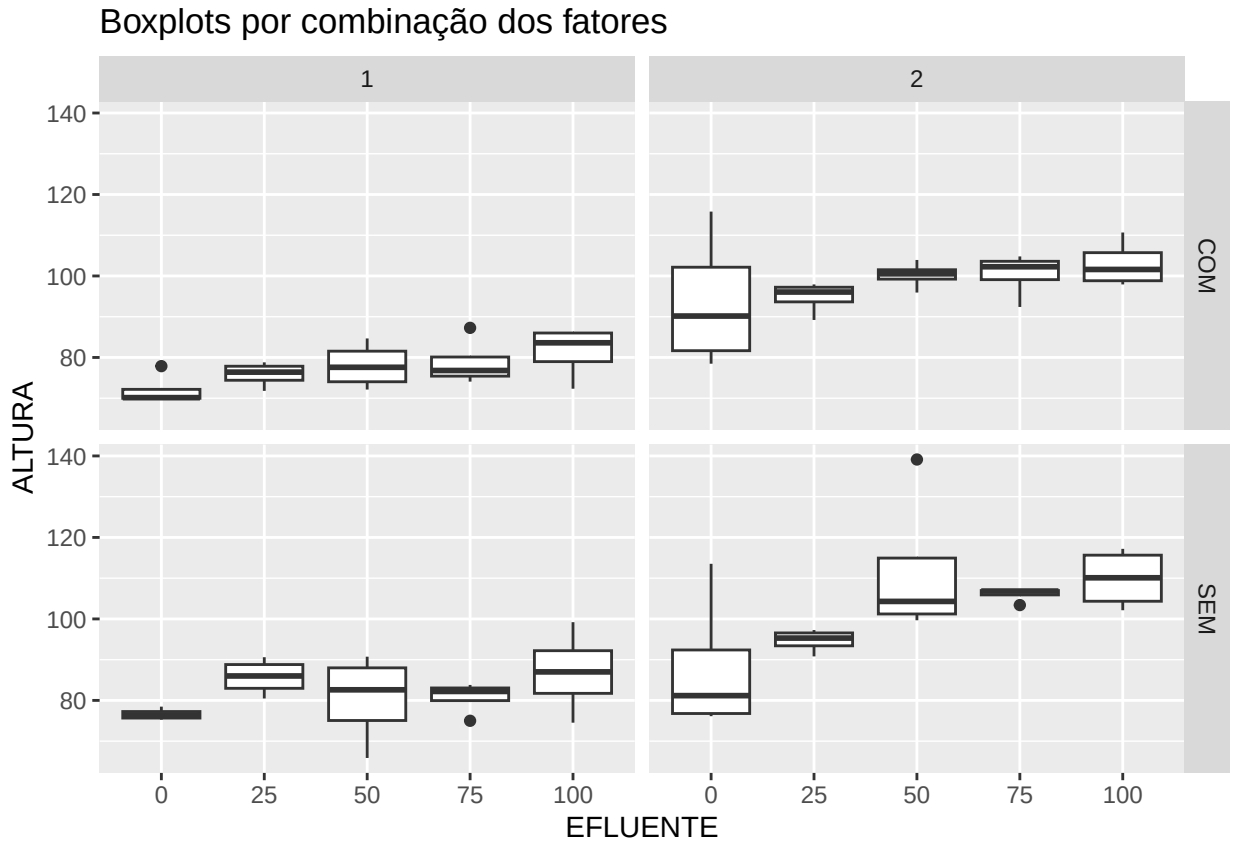
```r
for (i in 1:length(interacoes_significativas)) {
  if (length(interacoes_significativas) == 0){
    break
  }
  interacao = interacoes_significativas[i]
  partes <- strsplit(interacao, split = ":")
  if (nchar(interacao) < 20){
    fator1 <- partes[[1]][1]
    fator2 <- partes[[1]][2]
    fator3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

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ CICLO + INOCULO,
                                 data = blocos_dados_tempo_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ CICLO + EFLUENTE,
                                 data = blocos_dados_tempo_2, FUN = mean)
    print(media_interacao)
  }

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

```
                              data = blocos_dados_tempo_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(ALTURA ~ CICLO + 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 64 e 76 dias

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




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

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

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

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

```r
# Análises Descritivas
str(blocos_dados_tempo_3)
```

```
## tibble [80 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 ...
##  $ ALTURA  : num [1:80] 70.1 70.3 69.9 70 75.3 ...
```

```r
summary(blocos_dados_tempo_3)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      ALTURA
##  1:20   0  :16   COM:40   1:40   Min.   : 65.87
##  2:20   25 :16   SEM:40   2:40   1st Qu.: 77.68
##  3:20   50 :16                   Median : 87.64
##  4:20   75 :16                   Mean   : 89.40
##         100:16                   3rd Qu.:100.43
##                                  Max.   :117.20
```

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

```
##     COM SEM
## 0     8   8
## 25    8   8
## 50    8   8
## 75    8   8
## 100   8   8
```

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

```
##          COM      SEM
## 0   654.8667 658.4000
## 25  682.6600 721.7400
## 50  712.8900 732.8267
## 75  705.2900 758.5733
## 100 737.3300 787.2600
```

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

```
##          COM      SEM
## 0   81.85833 82.30000
## 25  85.33250 90.21750
## 50  89.11125 91.60333
## 75  88.16125 94.82167
## 100 92.16625 98.40750
```

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

```
##          COM       SEM
## 0   281.1879 169.49677
## 25  113.3873  34.89674
## 50  159.6768 198.59106
## 75  186.2040 167.35821
## 100 165.0833 219.79214
```

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

```
##          COM       SEM
## 0   16.76866 13.019093
## 25  10.64835  5.907346
## 50  12.63633 14.092234
## 75  13.64566 12.936700
## 100 12.84848 14.825388
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_tempo_3,
                 model.tables(aov(ALTURA ~ CICLO + BLOCO + EFLUENTE + INOCULO +
```

```
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                                "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 89.39796
##
##  CICLO
## CICLO
##     1     2
## 79.37 99.43
##
##  BLOCO
## BLOCO
##     1     2     3     4
## 91.47 89.95 87.55 88.62
##
##  EFLUENTE
## EFLUENTE
##     0    25    50    75   100
## 82.08 87.77 90.36 91.49 95.29
##
##  INOCULO
## INOCULO
##   COM   SEM
## 87.33 91.47
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  73.33  80.81  79.22  79.31  84.16
##     2  90.82  94.74 101.49 103.67 106.42
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##     1  76.24  82.50
##     2  98.42 100.44
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   81.86 82.30
##      25  85.33 90.22
##      50  89.11 91.60
##      75  88.16 94.82
##      100 92.17 98.41
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
```

```
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  70.07  75.85  77.99  75.89  81.38
##     2  93.65  94.81 100.23 100.43 102.95
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0      25     50     75     100
##     1  76.60  85.77  80.45  82.73  86.93
##     2  88.00  94.67 102.76 106.91 109.88
```

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

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

```
## [1] 0.4015048
```

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

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

```
## [1] 0.9462296
```

```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(ALTURA ~ BLOCO + CICLO * 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  171.8   57.28
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1  343.5   343.5
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3  115.5   38.52
##
## Error: Within
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## CICLO                     1   8050    8050 145.339  < 2e-16 ***
## EFLUENTE                  4   1539     385   6.946 0.000138 ***
## CICLO:INOCULO             1     90      90   1.618 0.208833
## CICLO:EFLUENTE            4    289      72   1.307 0.279233
## INOCULO:EFLUENTE          4    111      28   0.500 0.735510
## CICLO:INOCULO:EFLUENTE    4    162      41   0.733 0.573746
## Residuals                54   2991      55
## ---
## 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(ALTURA ~ BLOCO + CICLO * 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: ALTURA
##          Df Sum Sq Mean Sq  F value    Pr(>F)
## CICLO     1 8050.0  8050.0 147.7072 < 2.2e-16 ***
```

```
## INOCULO   1  343.5   343.5   6.3022 0.0149210 *
## EFLUENTE  4 1538.9   384.7   7.0592 0.0001092 ***
## 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 == "CICLO" && fator2 == "INOCULO" && fator3 == 0){
    media_interacao <- aggregate(ALTURA ~ CICLO + INOCULO,
                                 data = blocos_dados_tempo_3, FUN = mean)
    print(media_interacao)
  }

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

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

```
  }

  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(ALTURA ~ CICLO + 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)
}
```

```
# Mudar nome das colunas dos dados atualizados
blocos_dados_tempo_1$`43 E 41 DAS` = blocos_dados_tempo_1$ALTURA
blocos_dados_tempo_1 = blocos_dados_tempo_1[-5]
blocos_dados_tempo_2$`50 E 49 DAS` = blocos_dados_tempo_2$ALTURA
blocos_dados_tempo_3$`64 E 76 DAS` = blocos_dados_tempo_3$ALTURA

# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_tempo_1, blocos_dados_tempo_2["50 E 49 DAS"],
                    blocos_dados_tempo_3["64 E 76 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 = "Biometricas Altura atualizado.xlsx",
      sheetName = "R - Biometricas Altura", append = FALSE)
```