Raízes

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("Raízes 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 5:6) {
  dados[, i] <- as.numeric(unlist(dados[, i]))
}
dados[5:6] =  round(dados[5:6], digits = 2)
str(dados)
```

```
## tibble [70 x 6] (S3: tbl_df/tbl/data.frame)
##  $ BLOCO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EFLUENTE: num [1:70] 0 0 25 25 50 50 75 75 100 100 ...
##  $ INOCULO : chr [1:70] "COM" "SEM" "COM" "SEM" ...
##  $ CICLO   : num [1:70] 1 1 1 1 1 1 1 1 1 1 ...
##  $ VOLUME  : num [1:70] 0.09 0.18 0.3 0.05 0.11 0.22 0.18 0.29 0.26 0.47 ...
##  $ AREA    : num [1:70] 73.7 162.8 272.5 25.4 93.6 ...
```

```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 a coluna
dados_1 = dados[c(1:5)]
dados_2 = dados[c(1:4,6)]

# Estrutura dos dados após separados
"dados_1"
```

```
## [1] "dados_1"
```

```
str(dados_1)
```

```
## tibble [70 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 ...
##  $ VOLUME  : num [1:70] 0.09 0.18 0.3 0.05 0.11 0.22 0.18 0.29 0.26 0.47 ...
```

```
"dados_2"
```

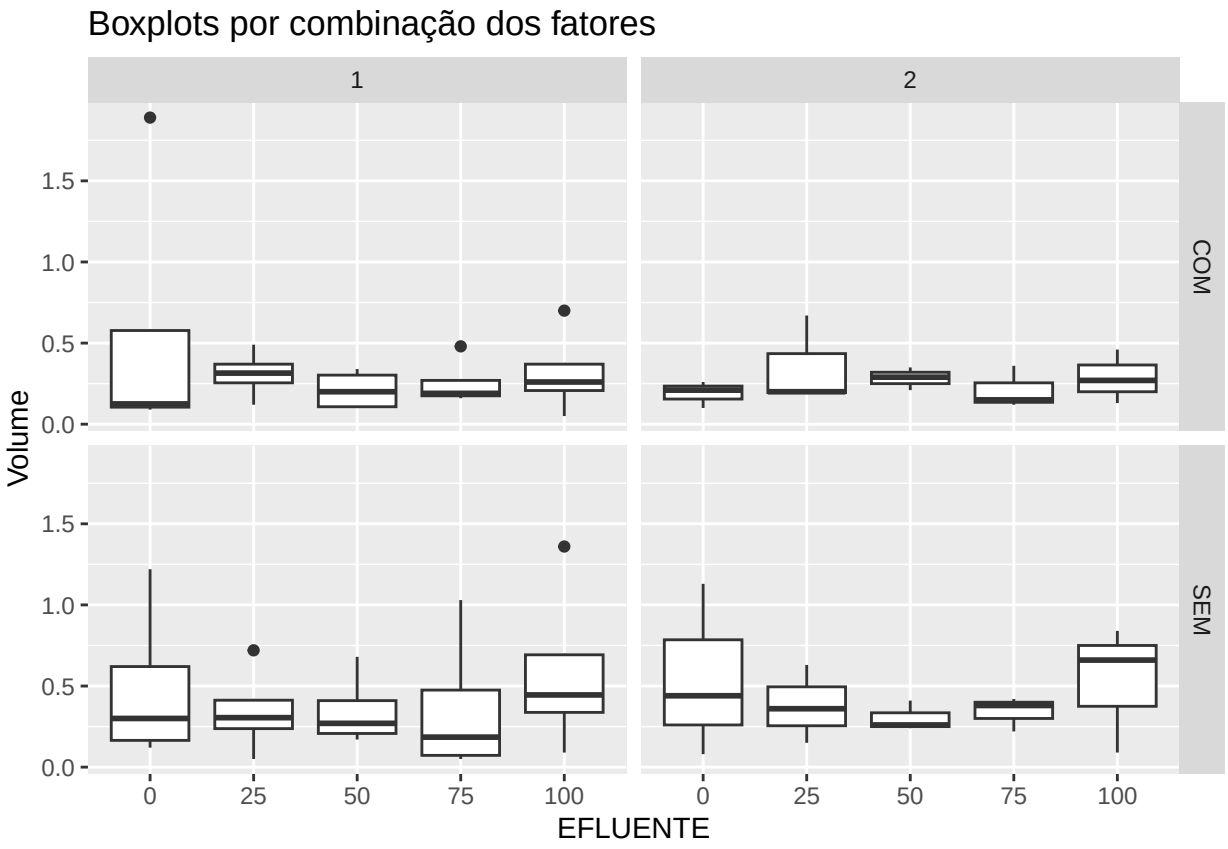
```
## [1] "dados_2"
```

```
str(dados_2)
```

```
## tibble [70 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 ...
##  $ AREA    : num [1:70] 73.7 162.8 272.5 25.4 93.6 ...
```

Análise para Volume

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_1, aes(x = factor(EFLUENTE), y = VOLUME)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Volume") +
  ggtitle("Boxplots por combinação dos fatores")
```



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

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

# 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

# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# 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_1 <- with(dados_1,
                       dados_1[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_1$VOLUME[which(blocos_dados_1$VOLUME <
                              limites_outliers$LIM_INF)] = NA
blocos_dados_1$VOLUME[which(blocos_dados_1$VOLUME >
                              limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_1 = aggregate(VOLUME ~ EFLUENTE + INOCULO + CICLO,
                           data = blocos_dados_1, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_1 = media_blocos_1[rep(row.names(media_blocos_1),
```

```
                                                each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_1) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_1$VOLUME[which(is.na(blocos_dados_1$VOLUME))] =
  media_blocos_1$VOLUME[which(is.na(blocos_dados_1$VOLUME))]
```

```
# Análises Descritivas
str(blocos_dados_1)
```

```
## tibble [70 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 ...
##  $ VOLUME  : num [1:70] 0.09 0.11 0.14 0.113 0.3 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      VOLUME
##  1:20   0  :14   COM:35   1:40   Min.   :0.0500
##  2:20   25 :14   SEM:35   2:30   1st Qu.:0.1425
##  3:20   50 :14                   Median :0.2500
##  4:10   75 :14                   Mean   :0.3037
##         100:14                   3rd Qu.:0.3600
##                                  Max.   :1.2200
```

```
# Número de observações
with(blocos_dados_1, tapply(VOLUME, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     7   7
## 25    7   7
## 50    7   7
## 75    7   7
## 100   7   7
```

```
with(blocos_dados_1, tapply(VOLUME, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   1.023333 3.590000
## 25  2.300000 2.020000
## 50  1.690000 2.300000
## 75  1.350000 2.470000
## 100 1.620000 2.896667
```

```r
with(blocos_dados_1, tapply(VOLUME, list(EFLUENTE, INOCULO), mean))
```

```
##           COM       SEM
## 0   0.1461905 0.5128571
## 25  0.3285714 0.2885714
## 50  0.2414286 0.3285714
## 75  0.1928571 0.3528571
## 100 0.2314286 0.4138095
```

```r
with(blocos_dados_1, tapply(VOLUME, list(EFLUENTE, INOCULO), var))
```

```
##             COM        SEM
## 0   0.004123810 0.22455714
## 25  0.037114286 0.03384762
## 50  0.010747619 0.02994762
## 75  0.006090476 0.10859048
## 100 0.016714286 0.07712381
```

```r
with(blocos_dados_1, tapply(VOLUME, list(EFLUENTE, INOCULO), sd))
```

```
##            COM       SEM
## 0   0.06421689 0.4738746
## 25  0.19265068 0.1839772
## 50  0.10367072 0.1730538
## 75  0.07804150 0.3295307
## 100 0.12928374 0.2777117
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_1,
                 model.tables(aov(VOLUME ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                    CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                    CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 0.3037143
##
##  CICLO
##           1       2
##      0.2745  0.3427
## rep 40.0000 30.0000
##
##  BLOCO
##           1       2       3       4
##      0.1981  0.2586  0.3626  0.4872
## rep 20.0000 20.0000 20.0000 10.0000
##
##  EFLUENTE
```

```
##           0      25     50      75     100
##      0.3295  0.3086  0.285  0.2729  0.3226
## rep 14.0000 14.0000 14.000 14.0000 14.0000
##
##  INOCULO
##         COM     SEM
##      0.2281  0.3793
## rep 35.0000 35.0000
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.299 0.265 0.279 0.271 0.258
##   rep 8.000 8.000 8.000 8.000 8.000
##   2   0.370 0.367 0.293 0.275 0.408
##   rep 6.000 6.000 6.000 6.000 6.000
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM    SEM
##   1    0.201  0.348
##   rep 20.000 20.000
##   2    0.265  0.421
##   rep 15.000 15.000
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   0.146 0.513
##      rep 7.000 0.231
##      25  0.329 0.289
##      rep 0.146 0.513
##      50  0.241 0.329
##      rep 0.329 0.289
##      75  0.193 0.353
##      rep 0.241 0.329
##      100 0.231 0.414
##      rep 0.193 0.353
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.113 0.310 0.210 0.180 0.190
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.190 0.353 0.283 0.210 0.287
##   rep 3.000 3.000 3.000 3.000 3.000
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   0.485 0.220 0.348 0.363 0.327
```

```
##   rep 4.000 4.000 4.000 4.000 4.000
##   2   0.550 0.380 0.303 0.340 0.530
##   rep 3.000 3.000 3.000 3.000 3.000
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_1 = aggregate(VOLUME ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_1, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_1$VOLUME)$p.value

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

```
## [1] 0.2927035
```

```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_1$VOLUME,
                                    media_blocos_1$EFLUENTE,
                                    media_blocos_1$INOCULO,
                                    media_blocos_1$CICLO)$p.value

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

```
## [1] 0.1914839
```

```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(VOLUME ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_1)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3 0.5546  0.1849
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO  1 0.4003  0.4003
##
## Error: BLOCO:INOCULO
##                Df  Sum Sq Mean Sq F value Pr(>F)
## CICLO:INOCULO   1 0.30510 0.30510    22.8 0.0412 *
## Residuals       2 0.02676 0.01338
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Within
##                          Df Sum Sq Mean Sq F value Pr(>F)
## CICLO                     1 0.2509 0.25091   5.471 0.0239 *
## EFLUENTE                  4 0.0329 0.00822   0.179 0.9479
## CICLO:INOCULO             1 0.0516 0.05163   1.126 0.2945
## CICLO:EFLUENTE            4 0.0509 0.01273   0.277 0.8910
## INOCULO:EFLUENTE          4 0.3085 0.07712   1.682 0.1713
## CICLO:INOCULO:EFLUENTE    4 0.0354 0.00886   0.193 0.9407
## Residuals                44 2.0178 0.04586
## ---
## 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(VOLUME ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_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: VOLUME
##          Df  Sum Sq Mean Sq F value   Pr(>F)
## BLOCO     3 0.55462 0.18487  3.6190 0.019713 *
## CICLO     1 0.25091 0.25091  4.9116 0.031555 *
## INOCULO   1 0.40028 0.40028  7.8356 0.007408 **
## 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
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "Não houve interações significativas no modelo"
```

```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(VOLUME ~ CICLO + INOCULO,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(VOLUME ~ CICLO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(VOLUME ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }
```

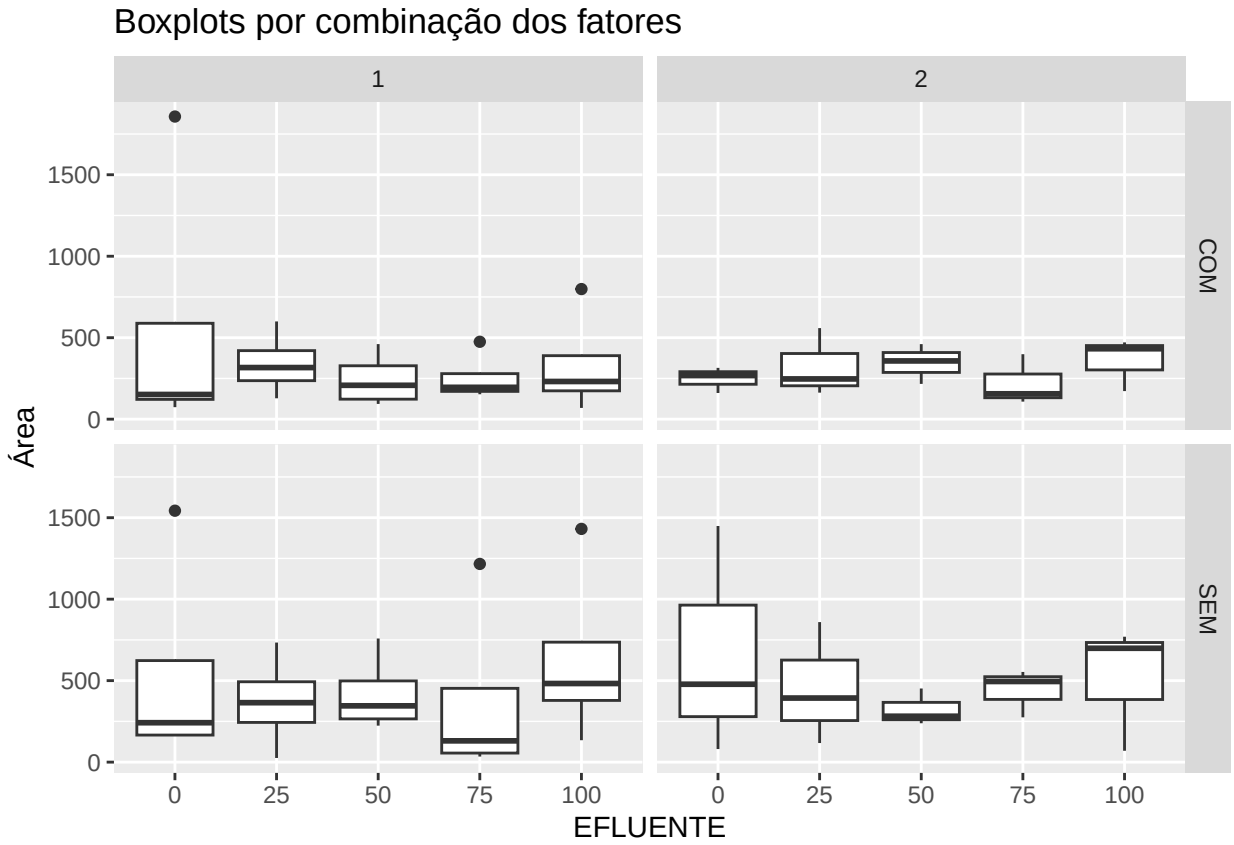
```
  if(fator1 == "CICLO" && fator2 == "INOCULO" && fator3 == "EFLUENTE"){
    media_interacao <- aggregate(VOLUME ~ CICLO + INOCULO + EFLUENTE,
                                 data = blocos_dados_1, FUN = mean)
    print(media_interacao)
  }
}
```

```
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 Área

```
# Gráficos com os boxplots para cada combinação de todos os fatores
ggplot(dados_2, aes(x = factor(EFLUENTE), y = AREA)) +
  geom_boxplot() +
  facet_grid(INOCULO ~ CICLO) +
  labs(x = "EFLUENTE", y = "Área") +
  ggtitle("Boxplots por combinação dos fatores")
```




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

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

# 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
# Definir o número de replicação de acordo com o valor de CICLO
num_rep <- ifelse(limites_outliers$CICLO == 1, 4, 3)

# Replicar as linhas do dataframe
limites_outliers <- limites_outliers[rep(row.names(limites_outliers),
                                         num_rep), ]

# 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_2 <- with(dados_2,
                           dados_2[order(CICLO, INOCULO, EFLUENTE), ])

# Definir como NA (valor ausente) os outliers
blocos_dados_2$AREA[which(blocos_dados_2$AREA <
                                    limites_outliers$LIM_INF)] = NA
blocos_dados_2$AREA[which(blocos_dados_2$AREA >
                                    limites_outliers$LIM_SUP)] = NA

# Calcular a média para cada grupo de 4 linhas
media_blocos_2 = aggregate(AREA ~ EFLUENTE + INOCULO + CICLO,
                                 data = blocos_dados_2, FUN = mean)

# Replicar cada linha por 4 vezes
media_blocos_2 = media_blocos_2[rep(row.names(media_blocos_2),
                                              each = 4), ]

# Redefinir os índices das linhas
rownames(media_blocos_2) <- NULL

# Preencher os NA's com as médias dos 4 blocos para a
# combinação de fatores específica
blocos_dados_2$AREA[which(is.na(blocos_dados_2$AREA))] =
  media_blocos_2$AREA[which(is.na(blocos_dados_2$AREA))]
```

```r
# Análises Descritivas
str(blocos_dados_2)
```

```
## tibble [70 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 ...
##  $ AREA    : num [1:70] 73.7 137.4 165.6 125.6 272.5 ...
```

```r
summary(blocos_dados_2)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO       AREA
##  1:20   0  :14   COM:35   1:40   Min.   :  25.41
##  2:20   25 :14   SEM:35   2:30   1st Qu.: 156.91
##  3:20   50 :14                   Median : 250.16
```

```
##  4:10   75 :14                  Mean   : 312.04
##         100:14                  3rd Qu.: 427.44
##                                 Max.   :1449.07
```

```
# Número de observações
with(blocos_dados_2, tapply(AREA, list(EFLUENTE, INOCULO), length))
```

```
##     COM SEM
## 0     7   7
## 25    7   7
## 50    7   7
## 75    7   7
## 100   7   7
```

```
with(blocos_dados_2, tapply(AREA, list(EFLUENTE, INOCULO), sum))
```

```
##          COM      SEM
## 0   1245.610 2869.133
## 25  2329.890 2857.950
## 50  2004.600 2645.050
## 75  1386.747 1716.373
## 100 1784.457 3003.333
```

```
with(blocos_dados_2, tapply(AREA, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   177.9443 409.8762
## 25  332.8414 408.2786
## 50  286.3714 377.8643
## 75  198.1067 245.1962
## 100 254.9224 429.0476
```

```
with(blocos_dados_2, tapply(AREA, list(EFLUENTE, INOCULO), var))
```

```
##           COM       SEM
## 0    7080.457 226620.23
## 25  34166.697  91562.32
## 50  21921.961  35621.50
## 75   8873.218  43291.17
## 100 21290.580  69203.05
```

```
with(blocos_dados_2, tapply(AREA, list(EFLUENTE, INOCULO), sd))
```

```
##           COM      SEM
## 0    84.14545 476.0465
## 25  184.84236 302.5927
## 50  148.06067 188.7366
## 75   94.19776 208.0653
## 100 145.91292 263.0647
```

```r
# Tamanho das Médias
T.medias <- with(blocos_dados_2,
                 model.tables(aov(AREA ~ CICLO + BLOCO + EFLUENTE + INOCULO +
                                      CICLO:EFLUENTE:INOCULO + CICLO:EFLUENTE +
                                      CICLO:INOCULO + EFLUENTE:INOCULO),
                              "means"))
T.medias
```

```
## Tables of means
## Grand mean
##
## 312.0449
##
##  CICLO
##         1     2
##     253.7 389.8
## rep  40.0  30.0
##
##  BLOCO
##         1     2     3     4
##     192.4 306.3 399.3 388.4
## rep  20.0  20.0  20.0  10.0
##
##  EFLUENTE
##         0    25    50    75 100
##     293.9 370.6 332.1 221.7 342
## rep  14.0  14.0  14.0  14.0  14
##
##  INOCULO
##     COM   SEM
##     250 374.1
## rep  35  35.0
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   170.5 356.2 330.4 139.8 271.8
##   rep   8.0   8.0   8.0   8.0   8.0
##   2   458.4 389.7 334.4 330.8 435.6
##   rep   6.0   6.0   6.0   6.0   6.0
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM   SEM
##   1   213.4 294.1
##   rep  20.0  20.0
##   2   298.9 480.6
##   rep  15.0  15.0
##
##  EFLUENTE:INOCULO
##         INOCULO
## EFLUENTE COM   SEM
##      0   177.9 409.9
##      rep   7.0 254.9
```

```
##      25  332.8 408.3
##      rep 177.9 409.9
##      50  286.4 377.9
##      rep 332.8 408.3
##      75  198.1 245.2
##      rep 286.4 377.9
##      100 254.9 429.0
##      rep 198.1 245.2
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   125.6 340.3 242.5 181.2 177.2
##   rep   4.0   4.0   4.0   4.0   4.0
##   2   247.8 322.9 344.8 220.7 358.6
##   rep   3.0   3.0   3.0   3.0   3.0
##
## , , INOCULO = SEM
##
##      EFLUENTE
## CICLO 0     25    50    75    100
##   1   215.4 372.1 418.3  98.4 366.4
##   rep   4.0   4.0   4.0   4.0   4.0
##   2   669.1 456.5 323.9 441.0 512.6
##   rep   3.0   3.0   3.0   3.0   3.0
```

```r
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_2 = aggregate(AREA ~ EFLUENTE + INOCULO + CICLO,
                                data = blocos_dados_2, FUN = mean)

# Realizar o teste Shapiro-Wilk no conjunto de dados completo
resultado_shapiro <- shapiro.test(media_blocos_2$AREA)$p.value

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

```
## [1] 0.6813962
```

```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_2$AREA,
                                    media_blocos_2$EFLUENTE,
```

```
                                    media_blocos_2$INOCULO,
                                    media_blocos_2$CICLO)$p.value

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

```
## [1] 0.1812024
```

```
# 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(AREA ~ BLOCO + CICLO * INOCULO * EFLUENTE +
                          Error(BLOCO*INOCULO), data = blocos_dados_2)

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

```
##
## Error: BLOCO
##       Df Sum Sq Mean Sq
## BLOCO  3 433298  144433
##
## Error: INOCULO
##          Df Sum Sq Mean Sq
## INOCULO   1 269147  269147
##
## Error: BLOCO:INOCULO
##                Df Sum Sq Mean Sq F value Pr(>F)
## CICLO:INOCULO   1  21009   21009   0.908  0.441
## Residuals       2  46271   23136
##
## Error: Within
##                          Df  Sum Sq Mean Sq F value Pr(>F)
## CICLO                     1  391200  391200   8.487 0.0056 **
## EFLUENTE                  4  185125   46281   1.004 0.4157
## CICLO:INOCULO             1   77371   77371   1.678 0.2019
## CICLO:EFLUENTE            4  187995   46999   1.020 0.4077
## INOCULO:EFLUENTE          4   82222   20556   0.446 0.7747
## CICLO:INOCULO:EFLUENTE    4  172408   43102   0.935 0.4525
## Residuals                44 2028237   46096
## ---
## 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(AREA ~ BLOCO + CICLO * INOCULO * EFLUENTE,
             data = blocos_dados_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: AREA
##          Df Sum Sq Mean Sq F value   Pr(>F)
## BLOCO     3 433298  144433  3.1882 0.032125 *
## CICLO     1 391200  391200  8.6352 0.005097 **
## INOCULO   1 269147  269147  5.9411 0.018630 *
## 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
if (length(interacoes_significativas) == 0){
  print("Não houve interações significativas no modelo")
} else {
  print(interacoes_significativas)
}
```

```
## [1] "Não houve interações significativas no modelo"
```

```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(AREA ~ CICLO + INOCULO,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "CICLO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(AREA ~ CICLO + EFLUENTE,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

  if(fator1 == "INOCULO" && fator2 == "EFLUENTE" && fator3 == 0){
    media_interacao <- aggregate(AREA ~ INOCULO + EFLUENTE,
                                 data = blocos_dados_2, FUN = mean)
    print(media_interacao)
  }

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

```
# Juntar dados em um mesmo dataframe
dados_final = cbind(blocos_dados_1, blocos_dados_2["AREA"])


# 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 = "Raízes atualizado.xlsx",
      sheetName = "R - Raízes", append = FALSE)
```