# Produtividade

Ana Carolina Murad Lima

2023-06-27

```
# 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)
library(stats)
```

```
# Leitura e tratamento dos dados
dados <- read_excel("Produtividade 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
dados[5] =  round(dados[5], digits = 2)
str(dados)
```

```
## tibble [80 x 5] (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 ...
##  $ KG POR HÁ: num [1:80] 1965 3280 2266 624 1415 ...
```

```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
```

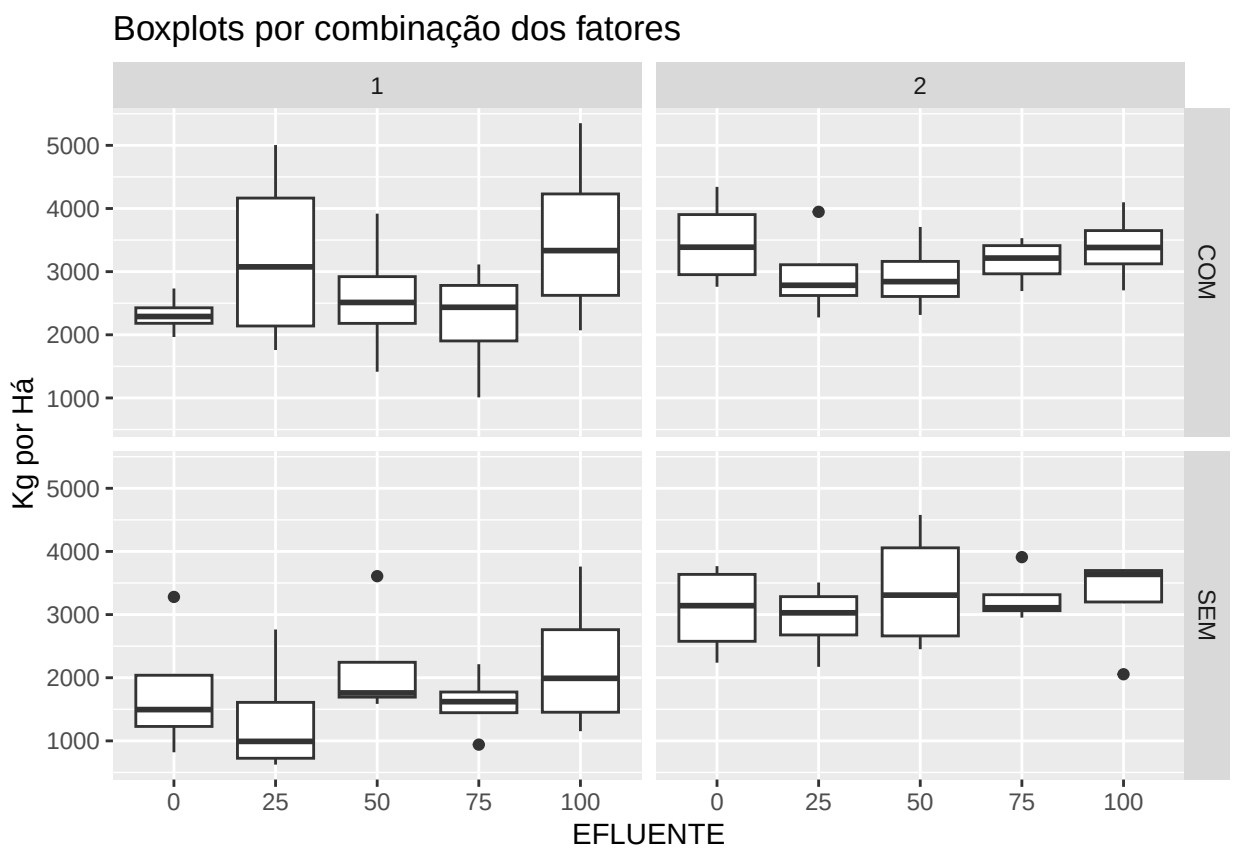
```
# Mudar o nome da variável de interesse
dados_1 = dados[c(1:5)]
colnames(dados_1)[5] = "kg_ha"

# Estrutura dos dados
"dados_1"
```

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

```
str(dados_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 ...
##  $ kg_ha   : num [1:80] 1965 3280 2266 624 1415 ...
```

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




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

# Calcular os limites superiores de outliers para cada combinação de fatores
limites_outliers <- aggregate(kg_ha ~ 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
})
```

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

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

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

# Calcular a média para cada grupo de 4 linhas
media_blocos_1 = aggregate(kg_ha ~ 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$kg_ha[which(is.na(blocos_dados_1$kg_ha))] =
  media_blocos_1$kg_ha[which(is.na(blocos_dados_1$kg_ha))]
```

```r
# Análises Descritivas
str(blocos_dados_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 ...
##  $ kg_ha   : num [1:80] 1965 2254 2733 2325 2266 ...
```

```
summary(blocos_dados_1)
```

```
##  BLOCO  EFLUENTE INOCULO  CICLO      kg_ha
##  1:20   0  :16   COM:40   1:40   Min.   : 624.2
##  2:20   25 :16   SEM:40   2:40   1st Qu.:2044.5
##  3:20   50 :16                   Median :2718.1
##  4:20   75 :16                   Mean   :2698.4
##         100:16                   3rd Qu.:3502.8
##                                  Max.   :5351.0
```

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

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

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

```
##          COM      SEM
## 0   23154.80 17368.60
## 25  23372.97 17108.12
## 50  22065.38 20451.93
## 75  21647.68 19498.24
## 100 27655.21 23546.22
```

```
with(blocos_dados_1, tapply(kg_ha, list(EFLUENTE, INOCULO), mean))
```

```
##          COM      SEM
## 0   2894.350 2171.075
## 25  2921.621 2138.515
## 50  2758.173 2556.492
## 75  2705.960 2437.280
## 100 3456.901 2943.277
```

```
with(blocos_dados_1, tapply(kg_ha, list(EFLUENTE, INOCULO), var))
```

```
##           COM     SEM
## 0    642545.1 1203814
## 25  1086118.8 1277945
## 50   633282.1 1262423
## 75   649824.1  473243
## 100 1015104.0 1163867
```

```
with(blocos_dados_1, tapply(kg_ha, list(EFLUENTE, INOCULO), sd))
```

```
##           COM       SEM
## 0    801.5891 1097.1847
## 25  1042.1702 1130.4624
## 50   795.7902 1123.5762
## 75   806.1167  687.9266
## 100 1007.5237 1078.8267
```

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

```
## Tables of means
## Grand mean
##
## 2698.364
##
##  CICLO
## CICLO
##    1    2
## 2227 3170
##
##  BLOCO
## BLOCO
##      1      2      3      4
## 2803.1 2788.4 2812.2 2389.7
##
##  EFLUENTE
## EFLUENTE
##    0   25   50   75  100
## 2533 2530 2657 2572 3200
##
##  INOCULO
## INOCULO
##    COM    SEM
## 2947.4 2449.3
##
##  CICLO:EFLUENTE
##      EFLUENTE
## CICLO 0    25   50   75   100
##     1 1795 2286 2146 2034 2873
##     2 3271 2774 3169 3109 3527
##
##  CICLO:INOCULO
##      INOCULO
## CICLO COM  SEM
##     1 2782 1671
##     2 3113 3227
##
##  EFLUENTE:INOCULO
##         INOCULO
```

```
## EFLUENTE COM  SEM
##       0   2894 2171
##       25  2922 2139
##       50  2758 2556
##       75  2706 2437
##       100 3457 2943
##
##  CICLO:EFLUENTE:INOCULO
## , , INOCULO = COM
##
##       EFLUENTE
## CICLO 0    25   50   75   100
##     1 2319 3229 2590 2249 3522
##     2 3469 2614 2927 3163 3392
##
## , , INOCULO = SEM
##
##       EFLUENTE
## CICLO 0    25   50   75   100
##     1 1270 1343 1702 1819 2224
##     2 3072 2934 3411 3056 3663
```

```
# Media dos blocos para fazer os testes de Normalidade e Homocedasticidade
media_blocos_1 = aggregate(kg_ha ~ 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$kg_ha)$p.value

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

```
## [1] 0.1158752
```

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

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

```
## [1] 0.9593743
```

```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(kg_ha ~ 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 2545870  848623
##
## Error: INOCULO
##          Df  Sum Sq Mean Sq
## INOCULO   1 4961538 4961538
##
## Error: BLOCO:INOCULO
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  3 755741  251914
##
## Error: Within
##                          Df   Sum Sq  Mean Sq F value   Pr(>F)
## CICLO                     1 17806260 17806260  29.479 1.38e-06 ***
## EFLUENTE                  4  5203837  1300959   2.154 0.086646 .
## CICLO:INOCULO             1  7496197  7496197  12.410 0.000877 ***
## CICLO:EFLUENTE            4  2392243   598061   0.990 0.420862
## INOCULO:EFLUENTE          4  1090684   272671   0.451 0.770899
## CICLO:INOCULO:EFLUENTE    4  2243315   560829   0.928 0.454364
## Residuals                54 32617543   604029
## ---
## 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(kg_ha ~ 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, ]
```

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

```
## Analysis of Variance Table
##
## Response: kg_ha
##               Df   Sum Sq  Mean Sq F value    Pr(>F)
## CICLO          1 17806260 17806260 30.4123 8.833e-07 ***
## INOCULO        1  4961538  4961538  8.4741 0.0051324 **
## CICLO:INOCULO  1  7496197  7496197 12.8032 0.0007147 ***
## 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)
```

```
## [1] "CICLO:INOCULO"
```

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

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

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

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

```
##   CICLO INOCULO    kg_ha
## 1     1     COM 2781.727
## 2     2     COM 3113.074
## 3     1     SEM 1671.437
## 4     2     SEM 3227.218
```

```
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)
}
```

```
##                   diif        lwr       upr        p_adj
## 1:SEM-1:COM -1110.290 -1750.6583 -469.9223 1.438269e-04
## 1:SEM-2:COM -1441.637 -2082.0052 -801.2692 9.973265e-07
## 2:SEM-1:SEM  1555.781   915.4132 2196.1492 1.675114e-07
```

```r
# Juntar dados em um mesmo dataframe
dados_final = blocos_dados_1


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

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

```r
write.xlsx(dados_final, file = "Produtividade atualizado.xlsx",
      sheetName = "R - Produtividade", append = FALSE)
```