EjemplosTestsPermutaciones

Pedro Luque

```
################################################
##ESTADÍSTICA COMPUTACIONAL I                  #
##Ejemplos de tests de permutaciones           #
################################################


################################################
#1. COMPARACIÓN DE DOS MUESTRAS INDEPENDIENTES
################################################

#Se han seleccionado 14 localizaciones en terreno
#despejado (campo) y 11 localizaciones en bosque
#En cada localización se ha contado el número de
#colonias de hormigas
#Se desea investigar si el número medio de colonias
#en el campo es mayor que el número medio de
#colonias en el bosque
#Si X es la v.a. número de colonias de hormigas
#en una localización, se desea realizar el siguiente contraste
#de hipótesis:
#H0: E[X/campo]=E[X/bosque]
#H1: E[X/campo]>E[X/bosque]

#Como se va a realizar mediante un test de permutaciones,
#en realidad el contraste que se va a realizar es:
#H0: F(x)=G(x)                para todo x, siendo F y G
#H1: E[X/campo]>E[X/bosque]   las funciones de distribución de X
                              #en el campo y el bosque

#i) Lectura y análisis de los datos
#----------------------------------
library(ggplot2)
hormigas <- read.table("hormigas.txt",header=TRUE)
hormigas
```

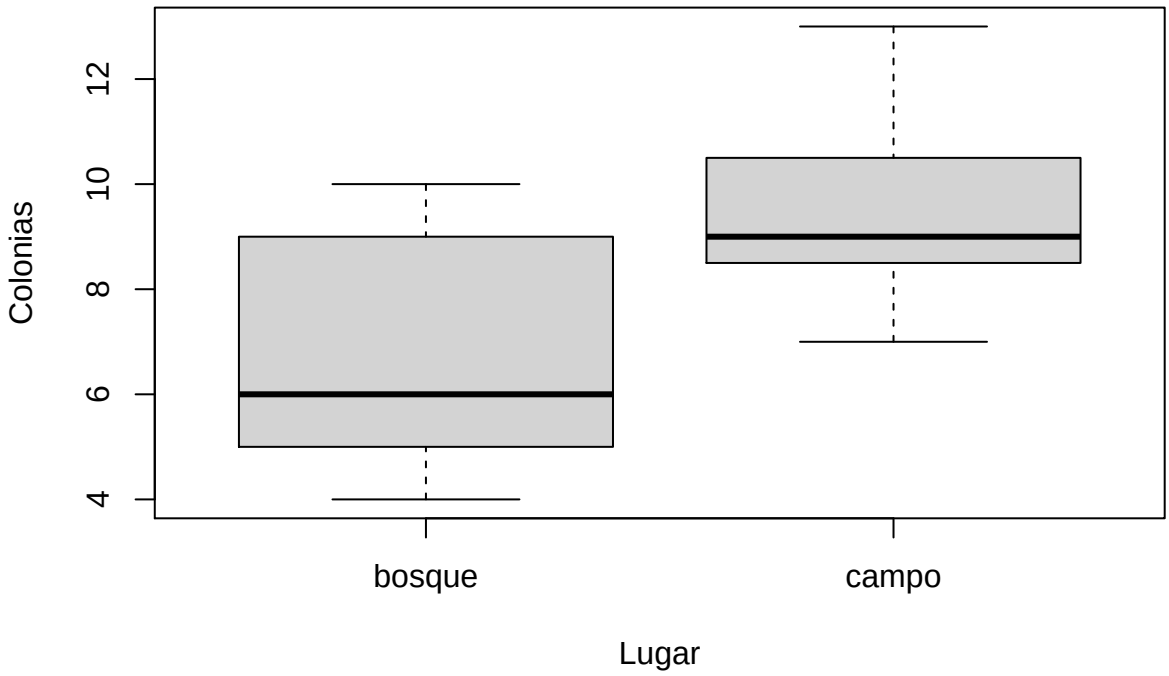
```
##     Lugar Colonias
## 1   campo        9
## 2   campo        9
## 3   campo       10
## 4   campo        9
## 5   campo       11
## 6   campo        7
## 7   campo       11
## 8   campo        8
```

```
## 9   campo        7
## 10  campo       13
## 11  campo       10
## 12 bosque        9
## 13 bosque        5
## 14 bosque        4
## 15 bosque        6
## 16 bosque        7
## 17 bosque       10
## 18 bosque       10
## 19 bosque        6
## 20 bosque        4
## 21 bosque        5
## 22 bosque        5
## 23 bosque        8
## 24 bosque        4
## 25 bosque        9
```

```
table(hormigas$Lugar)
```

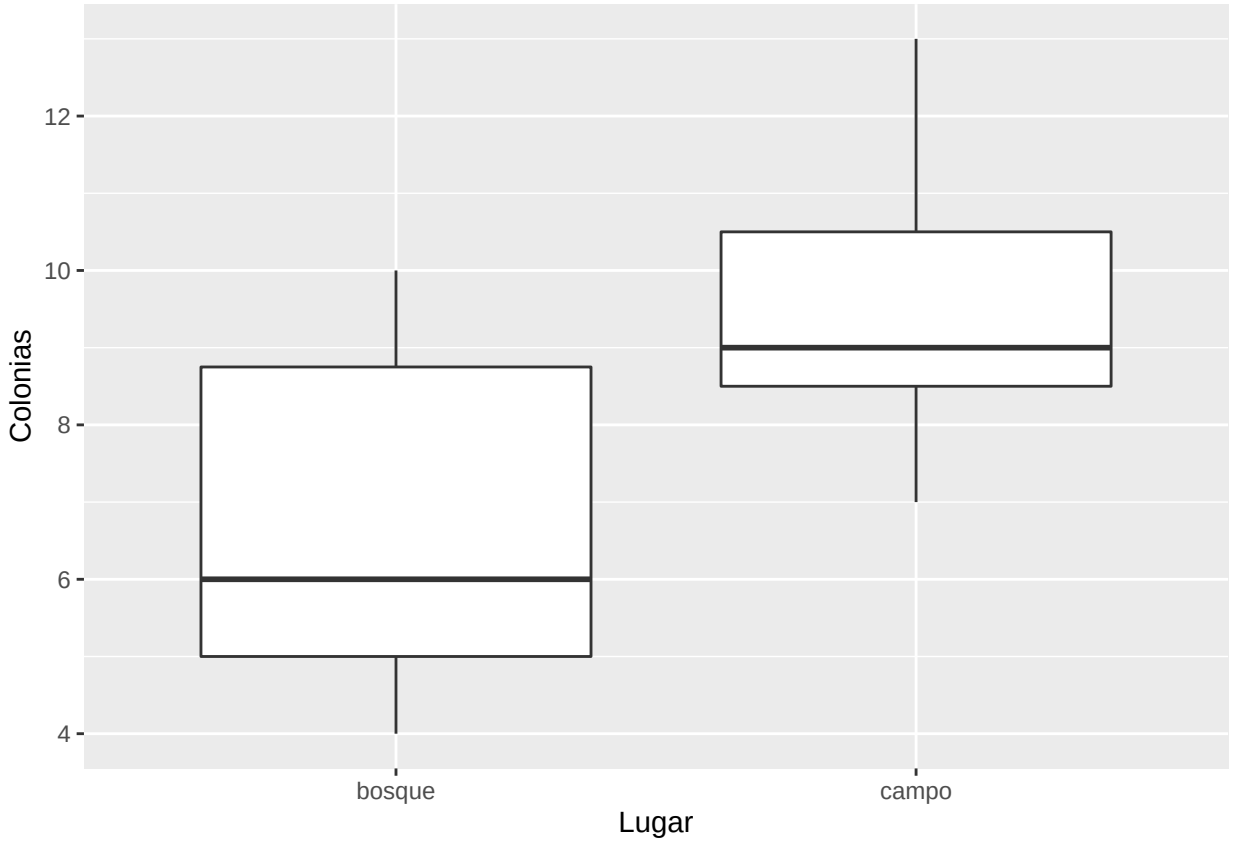
```
##
## bosque  campo
##     14     11
```

```
nC=sum(hormigas$Lugar=="campo")
nB=sum(hormigas$Lugar=="bosque")


boxplot(Colonias~Lugar,data=hormigas)
```

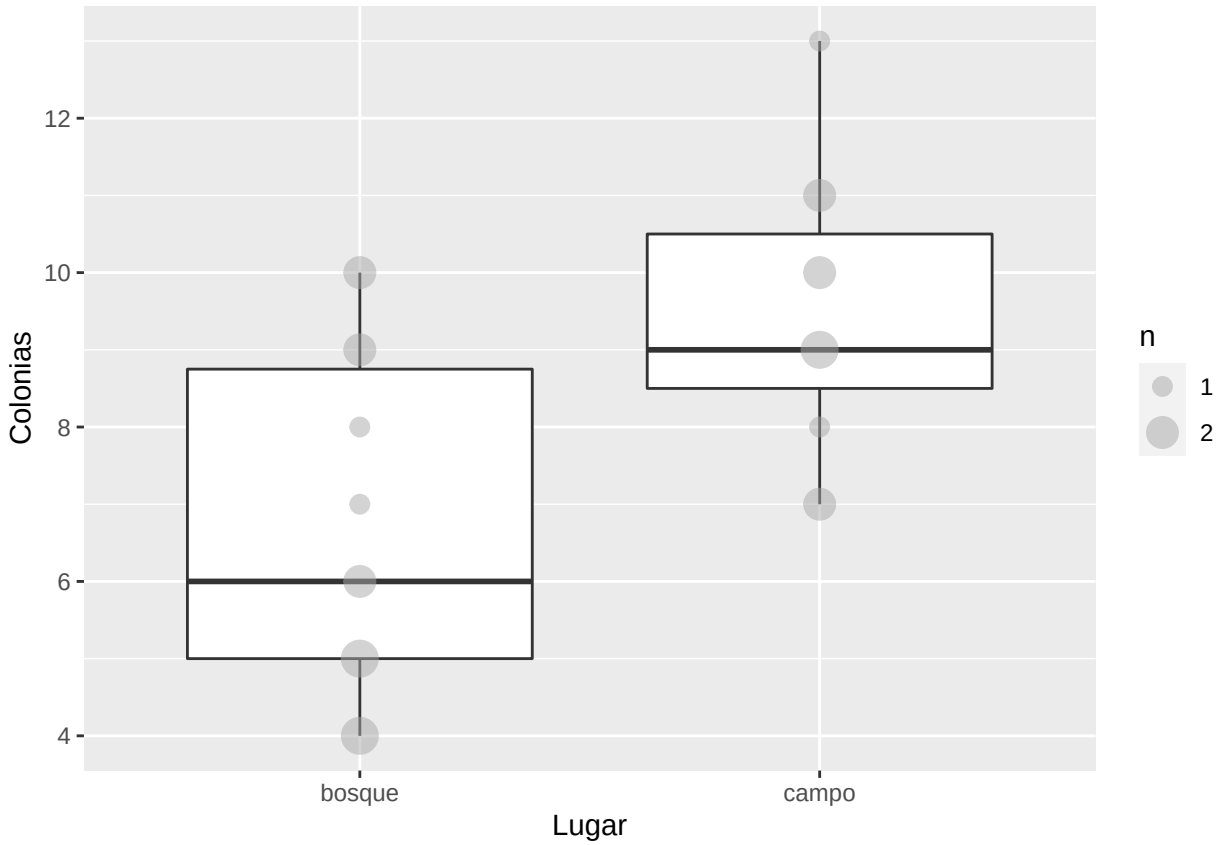



```
#o bien mediante ggplot
library(ggplot2)
(ggplot(hormigas,aes(Lugar,Colonias))
  + geom_boxplot()
```

```
)
```




```
#para que se vea la frecuencia de cada valor
#cada círculo tiene un tamaño proporcionak
#al número de casos que presenta ese valor:

(ggplot(hormigas,aes(Lugar,Colonias))
  + geom_boxplot()
  + stat_sum(colour="darkgray",alpha=0.5)
  + scale_size(breaks=1:2, range=c(3,6))
)
```




```
#Número medio de colonias según lugar
by(hormigas$Colonias,hormigas$Lugar,mean)
```

```
## hormigas$Lugar: bosque
## [1] 6.571429
## ------------------------------------------------------------
## hormigas$Lugar: campo
## [1] 9.454545
```

```
#Descriptivamente, el número medio de colonias es aproximadamente
#3 veces más eleveado en el campo
#El contraste permitirá determinar si esa
#diferencia es significativa

#Se pueden generar las permutaciones de muchas formas
#diferentes. En primer lugar, veamos cómo generar una permutación.
#La orden tranform siguiente genera una transformación de
#data frame hormigas. En este caso la variable Lugar no es modificada,
#mientras que los valores de la variable Colonias original
#son permutados de forma aleatoria
n=nrow(hormigas)
transform(hormigas,Colonias=Colonias[sample(n)])
```

```
##     Lugar Colonias
## 1   campo       10
## 2   campo        4
## 3   campo        9
## 4   campo       11
```

```
## 5   campo        6
## 6   campo       13
## 7   campo        6
## 8   campo        8
## 9   campo        7
## 10  campo       11
## 11  campo        7
## 12 bosque        9
## 13 bosque        9
## 14 bosque       10
## 15 bosque       10
## 16 bosque        5
## 17 bosque        9
## 18 bosque        5
## 19 bosque        4
## 20 bosque        9
## 21 bosque        8
## 22 bosque        5
## 23 bosque        4
## 24 bosque        7
## 25 bosque       10
```

```
#si se repite la orden anterior varias veces se
#puede comprobar cómo van cambiando los datos resultantes
#sample(n) genera una permutación de los casos
#que permite recorrerlos en otro orden
#las etiquetas de grupo se mantienen

#El número total de posibles permutaciones
#en este ejemplo es muy elevado
choose(nC+nB,nC)
```

```
## [1] 4457400
```

```
#es decir, el total de combinaciones de los nB+nC elementos
#tomados de nC en nC

#Cuando el número total de permutaciones es muy elevado
#Será preferible generar aleatoriamente un número B de permutaciones


#ii) Un test de permutaciones.
# Se puede utilizar el estadístico del test-t, de hecho se
# recomiendan estadísticos "normalizados", en este caso
# la diferencia de medias se divide por la estimación de
# la desviación típica
#-------------------------------------------------------------
#(diferencia de medias/ES)
(T0 <- t.test(Colonias~Lugar,data=hormigas,var.equal=TRUE)$statistic)
```

```
##         t
## -3.463845
```

```
#Atención: Sale negativo porque está basado en la diferencia
#bosque-campo, esto se tendrá en cuenta al calcular el p-valor
```

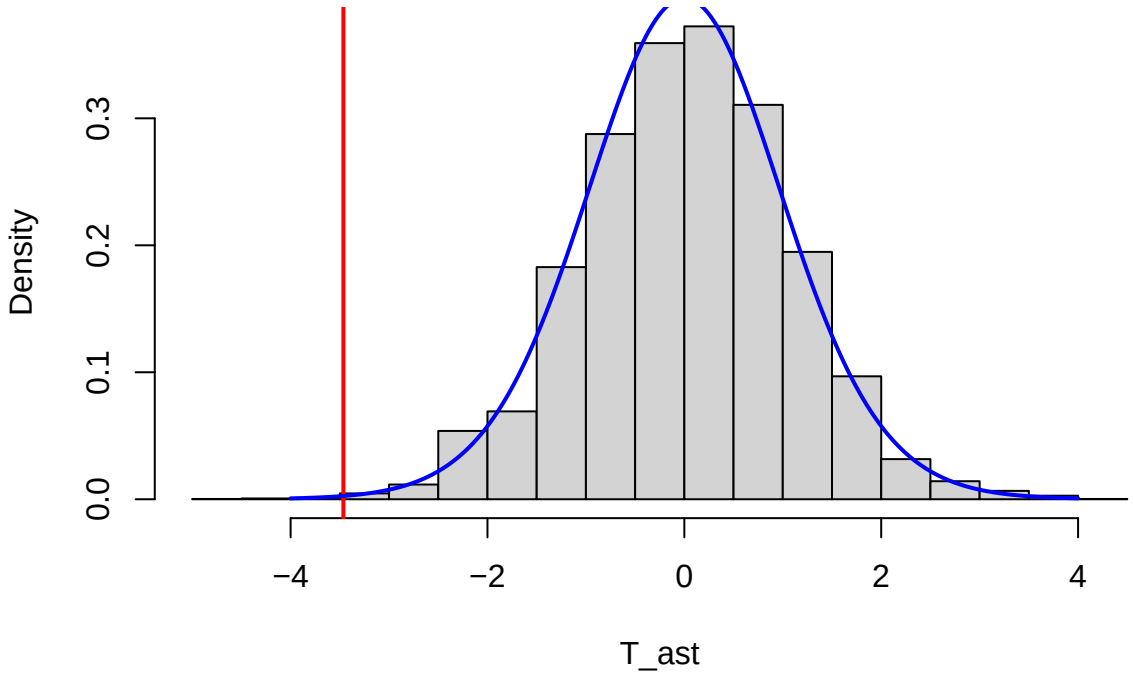
```r
set.seed(101) ##Para que sea reproducible el resultado
n=nrow(hormigas)
B <- 9999
T_ast <- numeric(B) ## Reservar espacio
for (b in 1:B) {
  if (b%%1000 ==0) cat("Perm.", b,"de",B,"\n")
  ## Permutar los casos y guardar en dataperm
  perm <- sample(n)
  dataperm <- transform(hormigas,Colonias=Colonias[perm])
  ## Calcular diferencia
  T_ast[b] <-  t.test(Colonias~Lugar,data=dataperm,var.equal=TRUE)$statistic
}
```

```
## Perm. 1000 de 9999
## Perm. 2000 de 9999
## Perm. 3000 de 9999
## Perm. 4000 de 9999
## Perm. 5000 de 9999
## Perm. 6000 de 9999
## Perm. 7000 de 9999
## Perm. 8000 de 9999
## Perm. 9000 de 9999
```

```r
hist(T_ast,br=30,col="lightgrey",freq=FALSE,
     main="Test de permutaciones (t-test)")
abline(v=T0,col="red",lwd=2)
#p-valor
(sum(T_ast<=T0)+1)/(B+1)
```

```
## [1] 0.0014
```

```r
#En este ejemplo la distribución generada
#tiene bastante similitud con la teórica
#del estadístico de la prueba t.test
gl=nC+nB-2
curve(dt(x,df=gl),-4,4,1000,add=TRUE,col="blue",lwd=2)
```




```
#iii) Generar las permutaciones con la función combn
#-----------------------------------------------------
#Controlando las combinaciones generadas
#podemos evitar que se repitan o incluso
#generar todas
#la función combn (ver help), p.e. comb(n,nc) genera todas las
#combinaciones de n elementos tomados de nc en nc
#Nótese que cada muestra permutada corresponde a una
#de esas combinaciones
#Se usa la función t para que salgan por filas
ind_comb <- t(combn(nrow(hormigas), sum(hormigas$Lugar=="campo")))
dim(ind_comb)
```

```
## [1] 4457400      11
```

```
nrow(ind_comb) ## total
```

```
## [1] 4457400
```

```
choose(nC+nB,nC) #comprobación
```

```
## [1] 4457400
```

```
#Ver algunas
ind_comb[sample(nrow(ind_comb),20),]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    1    4    5    6    8   11   12   15   16    21    22
##  [2,]    2    3    5   12   13   15   17   20   21    23    24
##  [3,]    5    7    9   10   14   16   19   20   21    22    25
##  [4,]    3    5    7   11   12   13   14   15   19    22    25
##  [5,]    4    6    8    9   10   15   16   18   20    23    24
```

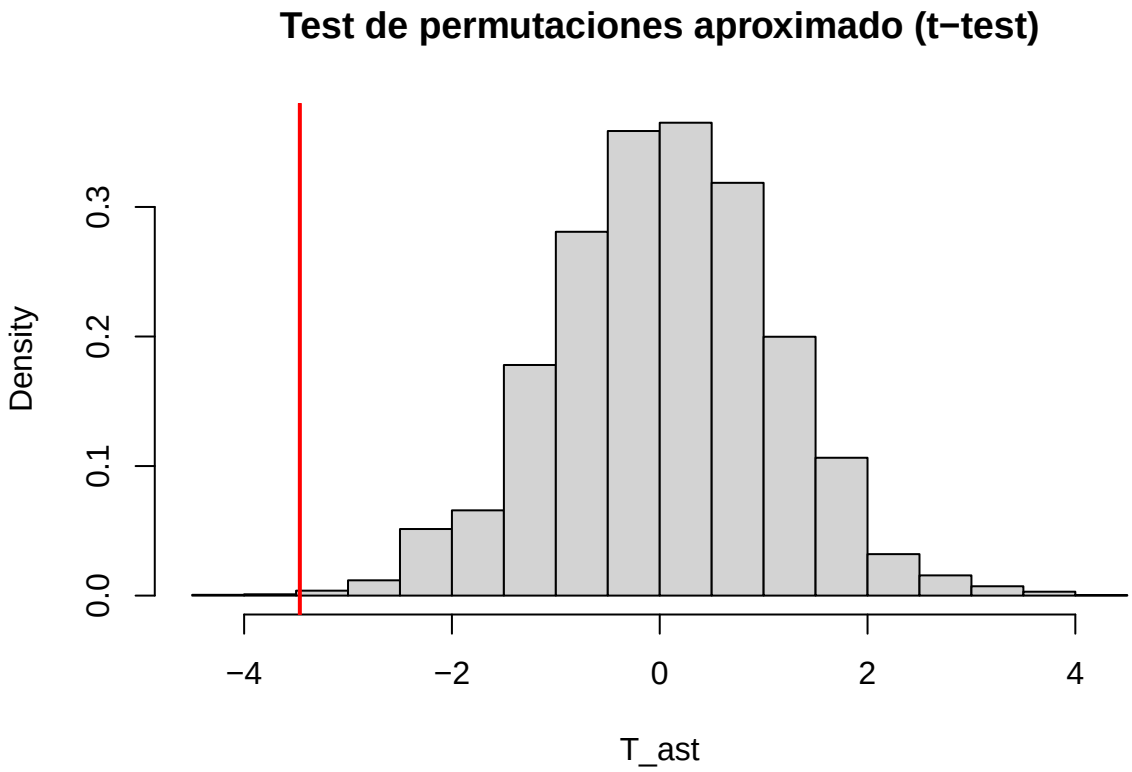
```
##  [6,]    1    4    6   12   14   16   17   18   21    23    25
##  [7,]    4    5    6    8   11   14   16   18   19    21    25
##  [8,]    1    4   11   12   14   15   16   19   21    23    24
##  [9,]    1    4    6   10   12   15   16   18   19    23    25
## [10,]    1    5    7    8   11   12   13   17   21    22    23
## [11,]    2    5    7   10   11   12   13   15   16    21    25
## [12,]    3    8   11   16   17   18   19   20   21    22    23
## [13,]    2    3    4    7    8   13   14   15   17    18    24
## [14,]    1    4    8   10   12   13   14   16   18    24    25
## [15,]    4    8   10   12   14   15   17   19   20    21    22
## [16,]    2    3    6    7   11   14   15   17   18    19    20
## [17,]    2    6    8   11   13   15   16   18   19    20    23
## [18,]    1    3    6    8   11   13   15   17   20    21    25
## [19,]    1    2    6   12   13   14   16   19   21    22    25
## [20,]    1    3    4    6   10   12   13   14   15    16    24
```

```
#cada fila representa los elementos asignados a la clase Campo
#(la que tiene 11 casos)

#Todas las permutaciones posibles
#No ejecutar estas instrucciones
#puede tardar entre 2 y 3 horas según
#el equipo informático
#Tarda bastante!!!
#B=nrow(ind_comb)
#T_ast <- numeric(B) ## Reservar espacio
#for (b in 1:B) {
#  if (b%%50 ==0) cat("Perm.", b,"de",B,"\n")
#  ## Permutar la variable respuesta
#  cc=ind_comb[b,]
#  dataperm <-  transform(hormigas,Colonias=c(Colonias[cc],Colonias[-cc]))
  ## Calcular diferencia
#  T_ast[b] <-  t.test(Colonias~Lugar,data=dataperm,var.equal=TRUE)$statistic
#}
#hist(T_ast,br=30,col="lightgrey",freq=FALSE,
#     main="Test de permutaciones exacto (t-test)")
#abline(v=T0,col="red",lwd=2)
#(sum(T_ast<=T0)+1)/(B+1)

#Seleccionar un número B, asegurando de paso que no se repite ninguna
B=9999
T_ast <- numeric(B) ## Reservar espacio
indices=sample(nrow(ind_comb),B)
for (b in 1:B) {
  if (b%%1000 ==0) cat("Perm.", b,"de",B,"\n")
  ## Permutar la variable respuesta
  cc=ind_comb[indices[b],]
  dataperm <-  transform(hormigas,Colonias=c(Colonias[cc],Colonias[-cc]))
  ## Calcular estadístico
  T_ast[b] <-  t.test(Colonias~Lugar,data=dataperm,var.equal=TRUE)$statistic
}
```

```
## Perm. 1000 de 9999
## Perm. 2000 de 9999
## Perm. 3000 de 9999
```

```
## Perm. 4000 de 9999
## Perm. 5000 de 9999
## Perm. 6000 de 9999
## Perm. 7000 de 9999
## Perm. 8000 de 9999
## Perm. 9000 de 9999
```

```
hist(T_ast,br=30,col="lightgrey",freq=FALSE,
     main="Test de permutaciones aproximado (t-test)")
abline(v=T0,col="red",lwd=2)
```




```
#p-valor:
(sum(T_ast<=T0)+1)/(B+1)
```

```
## [1] 0.002
```

```
#####################################################################
#2. Contraste bilateral de varianzas  de dos poblaciones     #
#####################################################################
#H0: Fx=Fy; H1: var(X)!=var(Y)
##x: tratamiento; y: control
x<-c(94,38,23,197,99,16,141)
y<-c(59,10,49,104,51,33,146,39,46)
nx<-length(x)
ny<-length(y)
choose(nx+ny,nx)
```

```
## [1] 11440
```

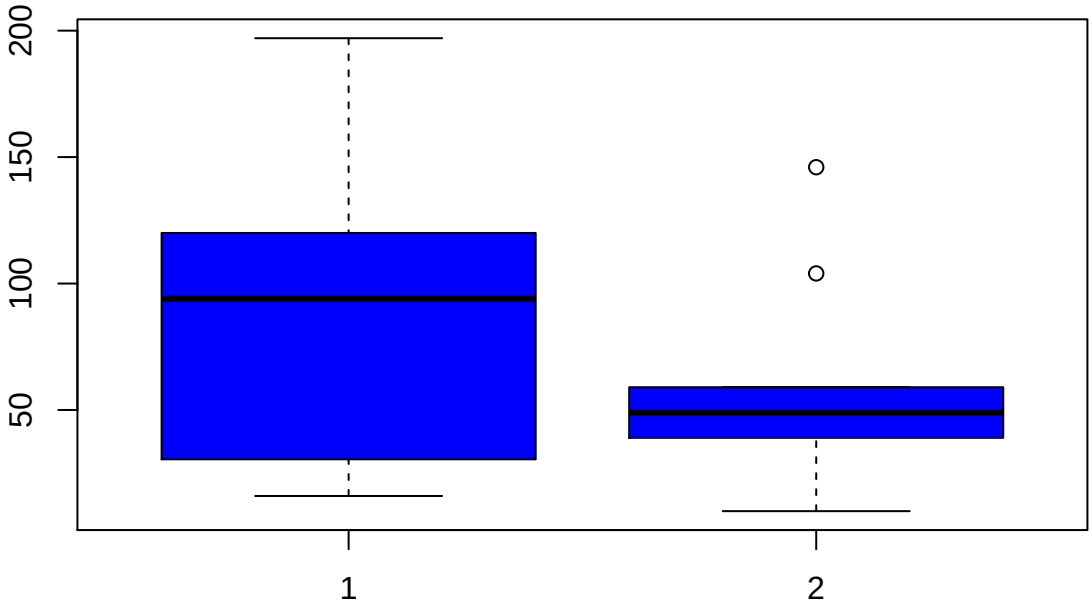
```
#En este caso se pueden generar todas las permutaciones
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   30.50   94.00   86.86  120.00  197.00
```

```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   39.00   49.00   59.67   59.00  146.00
```

```
boxplot(x,y,col="blue")
```




```
ind_comb <- t(combn(nx+ny, ny))
dim(ind_comb)
```

```
## [1] 11440     9
```

```
#cada fila, los individuos asignados a y
#Estadístico: vamos a usar el estadístico del test
#de Levene de comparación de varianzas
#bajo normalidad y homocedasticidad tendría
#una distribución F Snedecor
#suponiendo que falle alguna de esas hipótesis
#se recurre al remuestreo mediante permutaciones

#Primero, un data frame con la información necesaria
datos=data.frame(grupo=c(rep(1,nx),rep(2,ny)),X=c(x,y))
datos$grupo=factor(datos$grupo) #necesario para leveneTest
datos
```

```
##    grupo   X
## 1      1  94
## 2      1  38
## 3      1  23
## 4      1 197
## 5      1  99
## 6      1  16
## 7      1 141
## 8      2  59
```

```
## 9      2  10
## 10     2  49
## 11     2 104
## 12     2  51
## 13     2  33
## 14     2 146
## 15     2  39
## 16     2  46
```

```
library(car)  #para poder usar leveneTest
```

```
## Loading required package: carData
```

```
leveneTest(c(x,y),
           factor(c(rep("Tratamiento",length(x)),
                    rep("Control",length(y)))))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  2.1294 0.1666
##       14
```

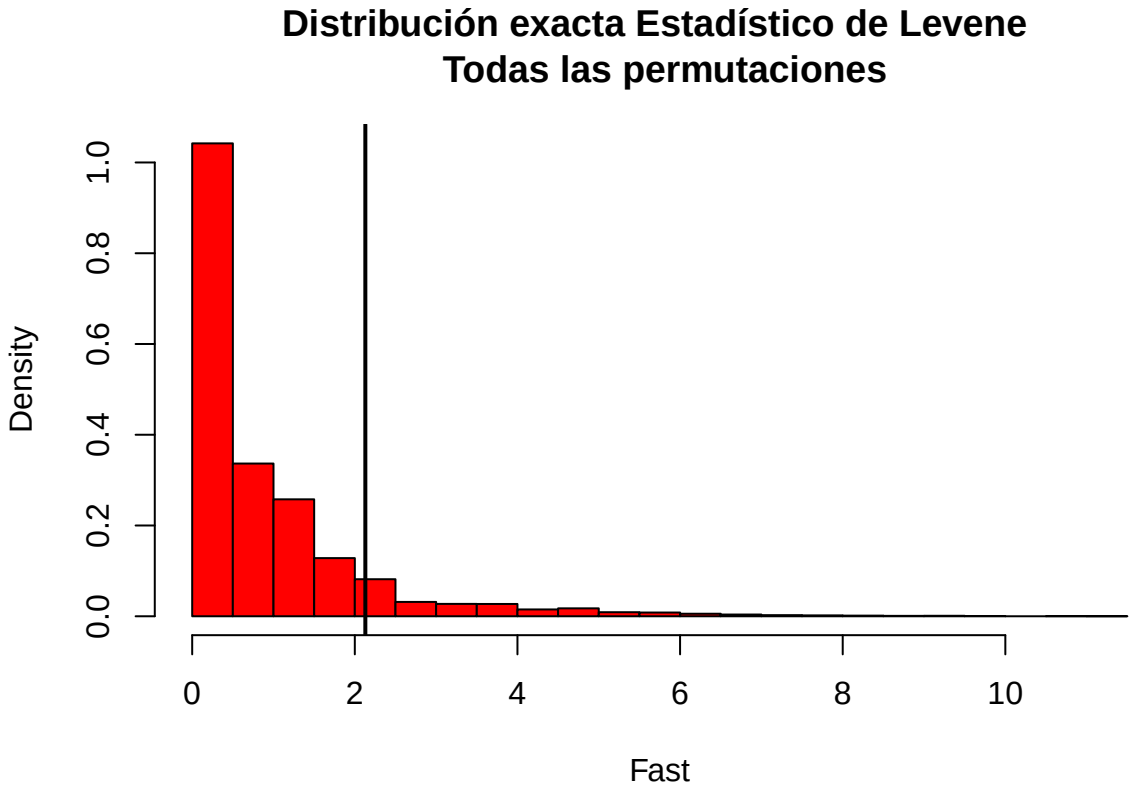
```
F0=leveneTest(X~grupo,data=datos)$`F value`[1]
F0
```

```
## [1] 2.129375
```

```
B=nrow(ind_comb)
Fast<-numeric(B) #vector donde almacenar los B valores

for (b in 1:B)
{
  if (b%%1000 ==0) cat("Perm.", b,"de",B,"\n")
  cc=ind_comb[b,]
  dataperm <-  transform(datos,X=c(X[cc],X[-cc]))
  ## Calcular estadístico
  Fast[b] <- leveneTest(X~grupo,data=dataperm)$`F value`[1]
}
```

```
## Perm. 1000 de 11440
## Perm. 2000 de 11440
## Perm. 3000 de 11440
## Perm. 4000 de 11440
## Perm. 5000 de 11440
## Perm. 6000 de 11440
## Perm. 7000 de 11440
## Perm. 8000 de 11440
## Perm. 9000 de 11440
## Perm. 10000 de 11440
## Perm. 11000 de 11440
```

```
hist(Fast,br=30,col="red",freq=FALSE,
     main="Distribución exacta Estadístico de Levene \n Todas las permutaciones")
abline(v=F0,lwd=2)
```




```
#p-valor exacto
mean(Fast>=F0)
```

```
## [1] 0.104458
```

```
#####################################################
#3. COMPARACIÓN DE MÁS DE DOS MUESTRAS              #
#   INDEPENDIENTES (ANOVA)                          #
#####################################################
#Se ha realizado un estudio experimental sobre 4 fertilizantes
#6 parcelas han sido tratadas con cada uno, y se ha medido la
#cosecha posteriormente recogida
#¿Son iguales las cosechas medias para los 4 fertilizantes?
datos=read.table("Cosechas.txt",header=TRUE,row.names=1)
datos
```

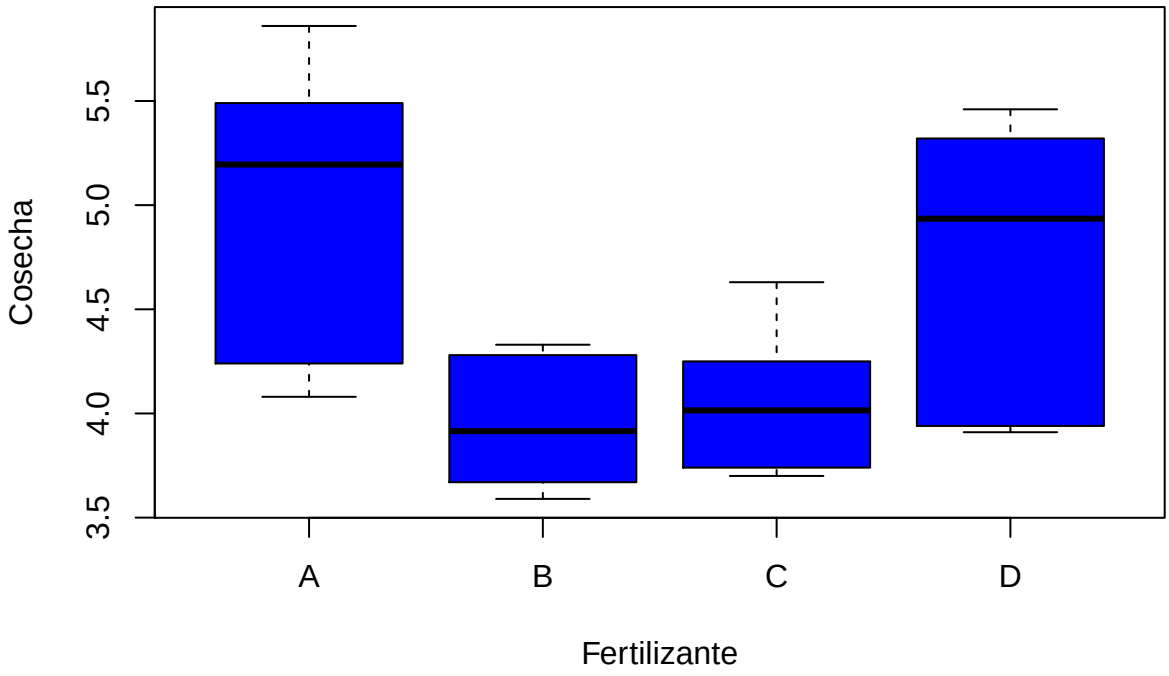
```
##    Fertilizante Cosecha
## 1             A    5.30
## 2             A    4.24
## 3             A    4.08
## 4             A    5.09
## 5             A    5.49
## 6             A    5.86
## 7             B    3.67
## 8             B    4.28
## 9             B    4.08
## 10            B    4.33
## 11            B    3.59
## 12            B    3.75
## 13            C    3.74
```

```
## 14            C    4.63
## 15            C    4.25
## 16            C    4.13
## 17            C    3.70
## 18            C    3.90
## 19            D    5.32
## 20            D    4.91
## 21            D    5.46
## 22            D    3.94
## 23            D    4.96
## 24            D    3.91
```

```
summary(datos)
```

```
##  Fertilizante          Cosecha
##  Length:24          Min.   :3.590
##  Class :character   1st Qu.:3.908
##  Mode  :character   Median :4.245
##                     Mean   :4.442
##                     3rd Qu.:4.992
##                     Max.   :5.860
```

```
attach(datos)
boxplot(Cosecha~Fertilizante,col="blue", main= "",
        xlab="Fertilizante", ylab="Cosecha")
```




```
tapply(Cosecha, Fertilizante, mean)
```

```
##        A        B        C        D
## 5.010000 3.950000 4.058333 4.750000
```

```
sapply(split(Cosecha,Fertilizante),shapiro.test)
```

```
##           A                         B
## statistic 0.9163325                 0.8822613
```

```
## p.value   0.4793463                     0.2796031
## method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
## data.name "X[[i]]"                      "X[[i]]"
##           C                             D
## statistic 0.9298846                     0.8527951
## p.value   0.5792109                     0.1657838
## method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
## data.name "X[[i]]"                      "X[[i]]"
```

```
#En principio se acepta la normalidad en cada muestra
#si bien los tamaños muestras son muy reducidos
library(car)
leveneTest(Cosecha~Fertilizante)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  3  1.0729  0.383
##       20
```

```
#Si optamos por el test paramétrico:

anavar<-aov(Cosecha~Fertilizante)
summary(anavar)
```

```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## Fertilizante  3  4.841  1.6135   5.466 0.00656 **
## Residuals    20  5.903  0.2952
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Habría evidencia estadística contra
#H0:F1=F2=F3=F4 siendo Fi f.Distribución Fertilizante i
#H1: al menos dos mui son distintas

#Mediante permutaciones, vamos a usar el mismo
#estadístico, pero sin emplear la distribución
#F-Snedecor
#Cuántas permutaciones hay?
factorial(24)/(4*factorial(6))
```

```
## [1] 2.154335e+20
```

```
#Por tanto se recurre a generar un número reducido B
#de permutaciones de forma aleatoria
#No usamos combn ya que es muy improbable en ese caso
#que se repitan permutaciones entre las generadas
#Vamos a usar el estadístico F del ANOVA paramétrico
#pero sin utilizar la distribución teórica F Snedecor
F0<- anova(anavar)[1,4]  #F para la muestra inicial
B<-9999
Fperm<-numeric(B)  #estadístico F para cada muestra permutada
for (b in 1:B)
{if (b%%1000 ==0) cat("Perm.", b,"de",B,"\n")
  Fertilizaperm<-sample(Fertilizante) #permutar fertilizantes
  anavast<-aov(Cosecha~  Fertilizaperm)
```

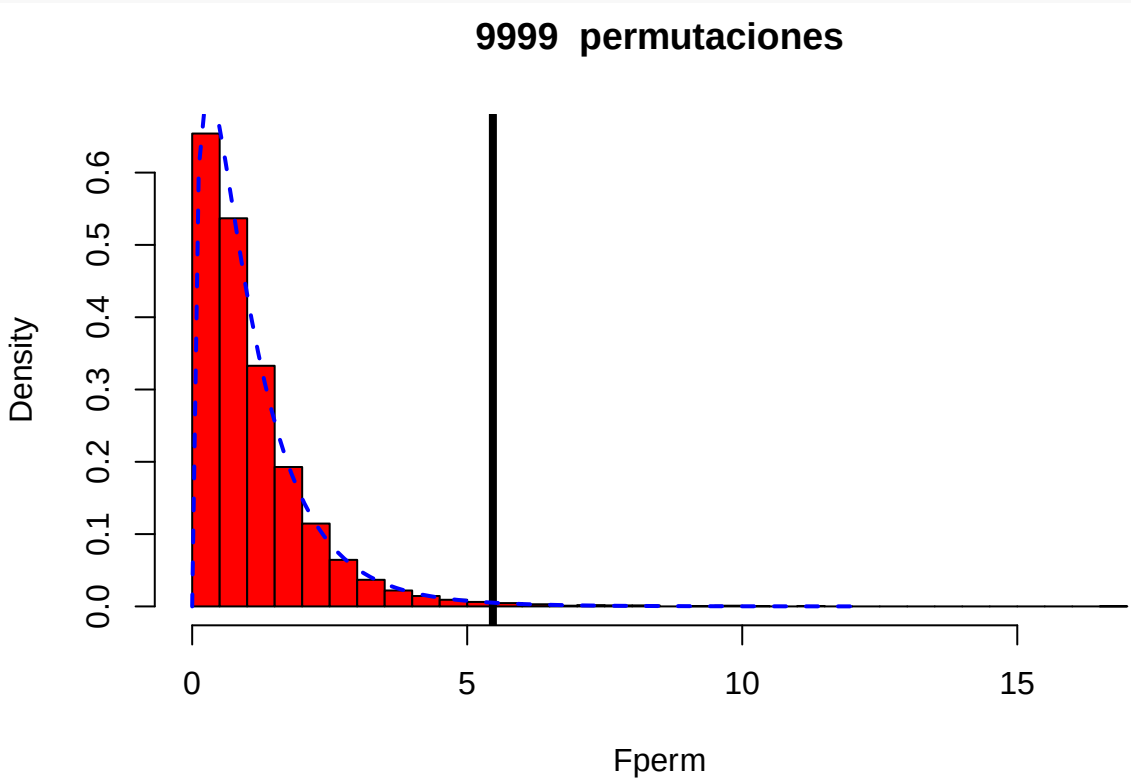
```
  Fperm[b]<-anova(anavast)[1,4]
}
```

```
## Perm. 1000 de 9999
## Perm. 2000 de 9999
## Perm. 3000 de 9999
## Perm. 4000 de 9999
## Perm. 5000 de 9999
## Perm. 6000 de 9999
## Perm. 7000 de 9999
## Perm. 8000 de 9999
## Perm. 9000 de 9999
```

```
hist(Fperm,br=30,col="red",fre=FALSE,
     main=paste(B," permutaciones"))
abline(v=F0, lwd=4)
cat("p-valor aproximado = ",
    (sum(F>=F0)+1)/(B+1),"\n")
```

```
## p-valor aproximado =  1e-04
```

```
curve(df(x,3,20),0,12,col="blue",lwd=2,lty=2,add=TRUE)
```




```
#la distribución del estadístico resultante es muy similar
#a la F-Senedecor teórica
```