Ejemplosbootstrap_I

Pedro Luque

```
##############################################
##ESTADISTICA COMPUTACIONAL I.               #
##GRADO EN ESTADISTICA                       #
##DOBLE GRADO EN MATEMATICAS Y ESTADISTICA   #
##EJEMPLOS BOOTSTRAP  (I)                    #
##############################################
source("ananor.r")   #Para analizar la normalidad


#####################################################
#PREVIO: cómo generar muestras bootstrap
#####################################################
#i) Muestras univariantes
#-----------------------
set.seed(123)
x=rnorm(5)
x
```

```
## [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774
```

```
#Se trata de generar una m.a.s. de los elementos de x,
#con reemplazmiento
#o sea x*=(x1*....x5*) donde las xi* son iid según x
#tres posibles muestras
sample(x)
```

```
## [1]  0.07050839 -0.23017749  0.12928774 -0.56047565  1.55870831
```

```
sample(x,rep=TRUE)
```

```
## [1]  1.5587083  0.1292877  1.5587083  1.5587083 -0.5604756
```

```
sample(x,rep=TRUE)
```

Valores repetidos

```
## [1]  0.07050839 -0.56047565 -0.56047565  0.12928774  1.55870831
```

```
sample(x,rep=TRUE)
```

```
## [1] -0.23017749 -0.23017749 -0.56047565  1.55870831  0.07050839
```

```
#En este caso habría 5^5=3125 muestras bootstrap posibles
#Se observa que en una muestra bootstrap usualmente habrá
#casos repetidos y casos que no aparecen
```

```
#ii) Muestras multivariantes
#---------------------------
#hay que seleccionar casos completos, por ejemplo seleccionando posiciones
#supongamos el siguiente conjunto de datos
```

```r
data(attitude)
datasub=attitude[1:10,1:4]
n=nrow(datasub)
datasub
```

```
##    rating complaints privileges learning
## 1      43         51         30       39
## 2      63         64         51       54
## 3      71         70         68       69
## 4      61         63         45       47
## 5      81         78         56       66
## 6      43         55         49       44
## 7      58         67         42       56
## 8      71         75         50       55
## 9      72         82         72       67
## 10     67         61         45       47
```

```r
datasub[sample(n,replace=TRUE),]
```

```
##      rating complaints privileges learning
## 1        43         51         30       39
## 7        58         67         42       56
## 5        81         78         56       66
## 10       67         61         45       47
## 7.1      58         67         42       56
## 9        72         82         72       67
## 9.1      72         82         72       67
## 10.1     67         61         45       47
## 7.2      58         67         42       56
## 5.1      81         78         56       66
```

Podemos generar tantas muestras como queramos

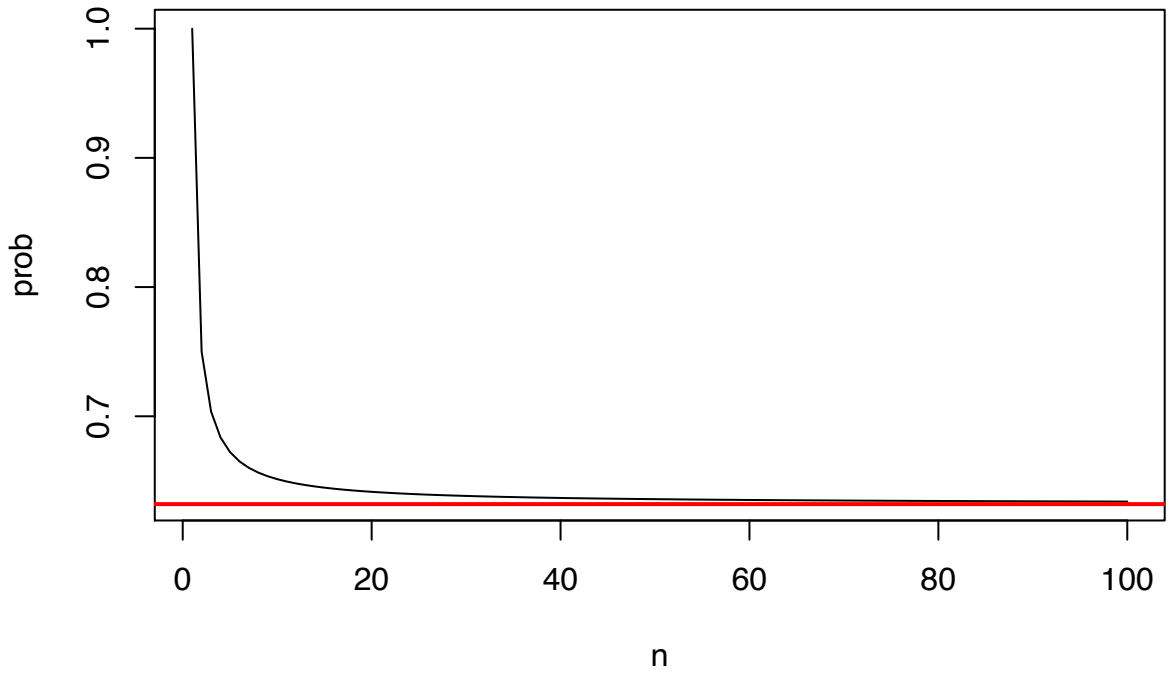
```r
#Las filas que aparecen con .1, .2, significa que aparecen repetidas
#en la muestra bootstrap

#iii) Probabilidad de que un caso pertenezca a una muestra
#     bootstrap= 1-(1-1/n)^n, tiende a 0.632
```

caso = fila

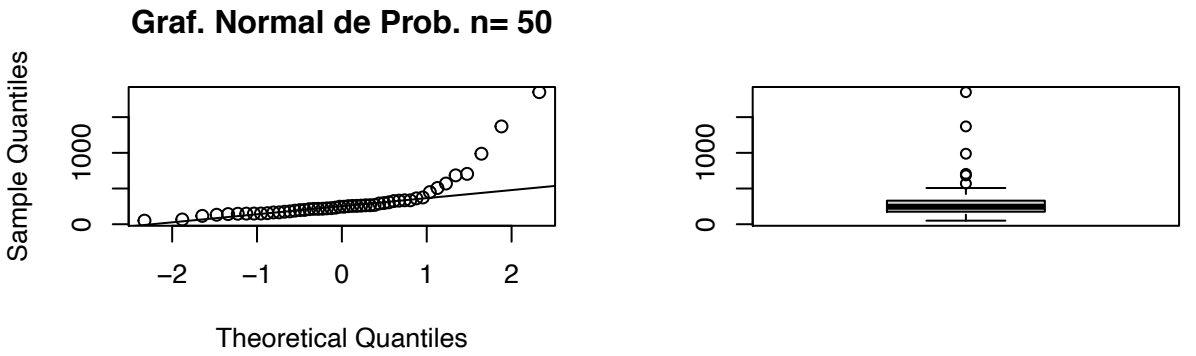
Para ver esto: Es el complementaario

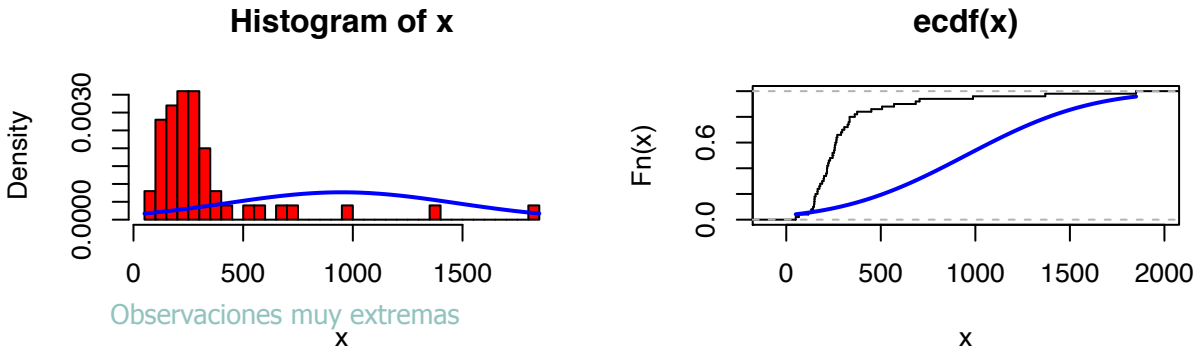
```r
n=seq(1,100,1)
prob=1-(1-1/n)^n

plot(n,prob,type="l")
abline(h=0.632,col="red",lwd=2)
```

A medida de que n crece, la probabilidad baja, pero nunca será menor que 0,632




```
#####################################################
#Ejemplo 1.
#Precios de venta (miles de euros) de viviendas
#en cierto lugar
#Estimador: media recortada
#Parámetro: Media poblacional
#Objetivo:  Estimar el sesgo y la varianza de este
#           estimador
#####################################################
x=c(142,175,197.5,149.4,705,232,50,146.5,155,1850,
    132.5,215,116.7,244.9,290,200,260,449.9,66.407,
    164.95,362,307,266,166,375,244.95,210.95,265,296,335,
    335,1370,256,148.5,987.5,324.5,215.5,684.5,270,330,
    222,179.8,257,252.95,149.95,225,217,570,507,190)

ananor(x)
```





Observaciones muy extremas

No parece que tenga comportamiento normal

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.60636, p-value = 2.424e-10
```

```
#La presencia de outliers afecta sensiblemente a
#la media muestral
#Alternativa: media recortada
```

Para evitar valores extremos

```
mean(x)
```

```
## [1] 329.2571
```

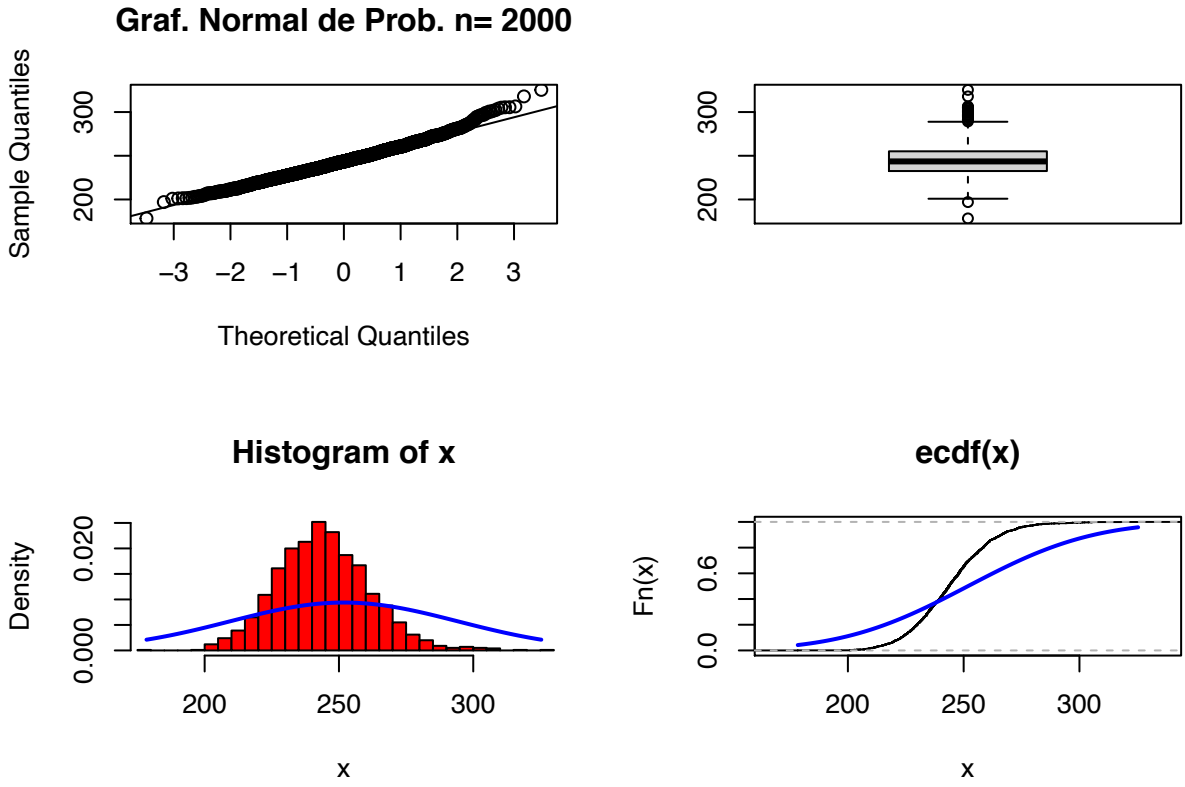
```
(medrec0= mean(x,trim=0.25))
```

Descarta el 25% de los valores por arriba y por abajo

```
## [1] 244.0019
```

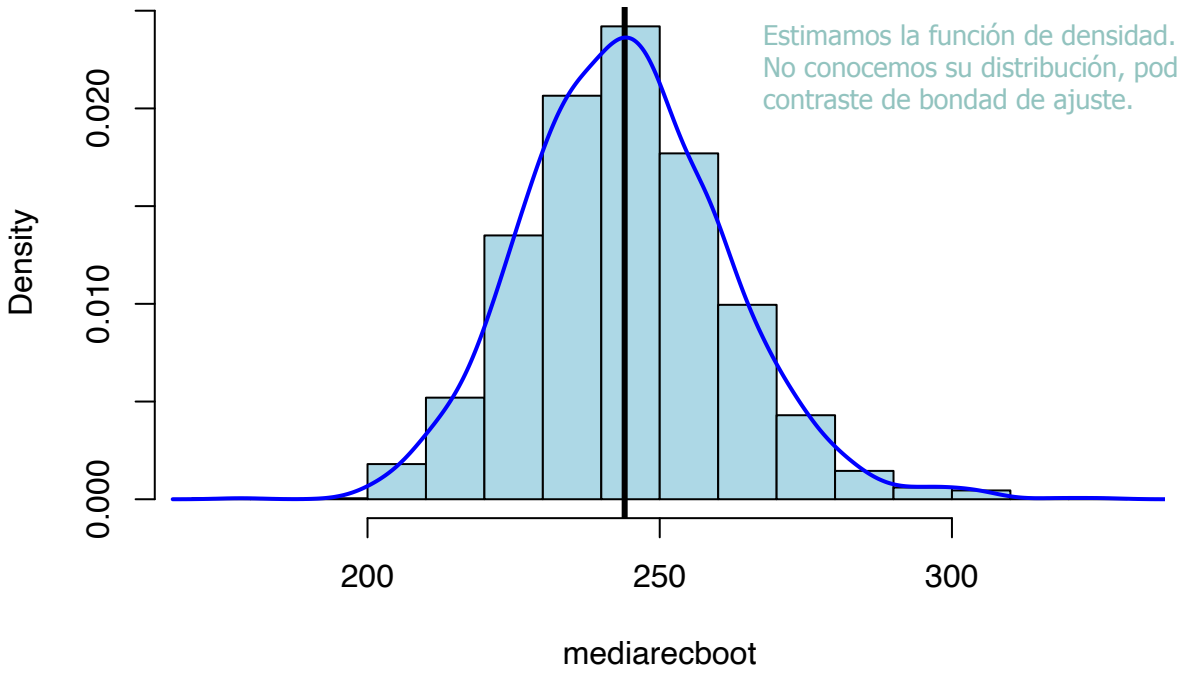
```
#se descartan el 25% de valores mayores
#y el 25% de valores menores

B=2000
#En el siguiente vector se guardarán los B valores
#del estadístico bootstrap
mediarecboot= numeric(B)
for (b in 1:B)
{   xboot= sample(x,replace=TRUE)
    mediarecboot[b]= mean(xboot,trim=0.25)
}  #siguiente b
```

```
#Suele ser de interés estudiar la forma de la distribución bootstrap
#sobre todo cuando vayamos a calcular IC
ananor(mediarecboot) #no parece que siga una distribución normal
```




```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.99146, p-value = 1.937e-09
```

```
hist(mediarecboot,br=20,prob=T,
     main="Distribuc. bootstrap de la media recortada 0.25",
     col="lightblue")
abline(v=medrec0,lwd=3)
lines(density(mediarecboot,bw="SJ"),col="blue",lwd=2)
```



Estimamos la función de densidad.
No conocemos su distribución, podríamos hacer un contraste de bondad de ajuste.

```
#Estimación del Sesgo:
sesgob=mean(mediarecboot)-  medrec0
sesgob
```

```
## [1] 0.187225
```

Estimación del sesgo pequeña

```
#Sesgo bajo comparado con medrec0
#Al igual que en el Jackknife se puede corregir la
#estimación
medrec0  - sesgob
```

```
## [1] 243.8147
```

```
#ES:
varBoot= var(mediarecboot); varBoot
```

```
## [1] 300.7658
```

```
ESBoot= sd(mediarecboot); ESBoot
```

```
## [1] 17.3426
```

```
##########################################
##Ejemplo 2. Estudiar mediante el bootstrap
##el Recorrido intercuartílico
##########################################

#Una función que devuelva el RI a partir
#de una muestra x
RI= function(x)
{
  Cuartiles= summary(x)[c(2,5)]
  res= Cuartiles[2]-Cuartiles[1]
```

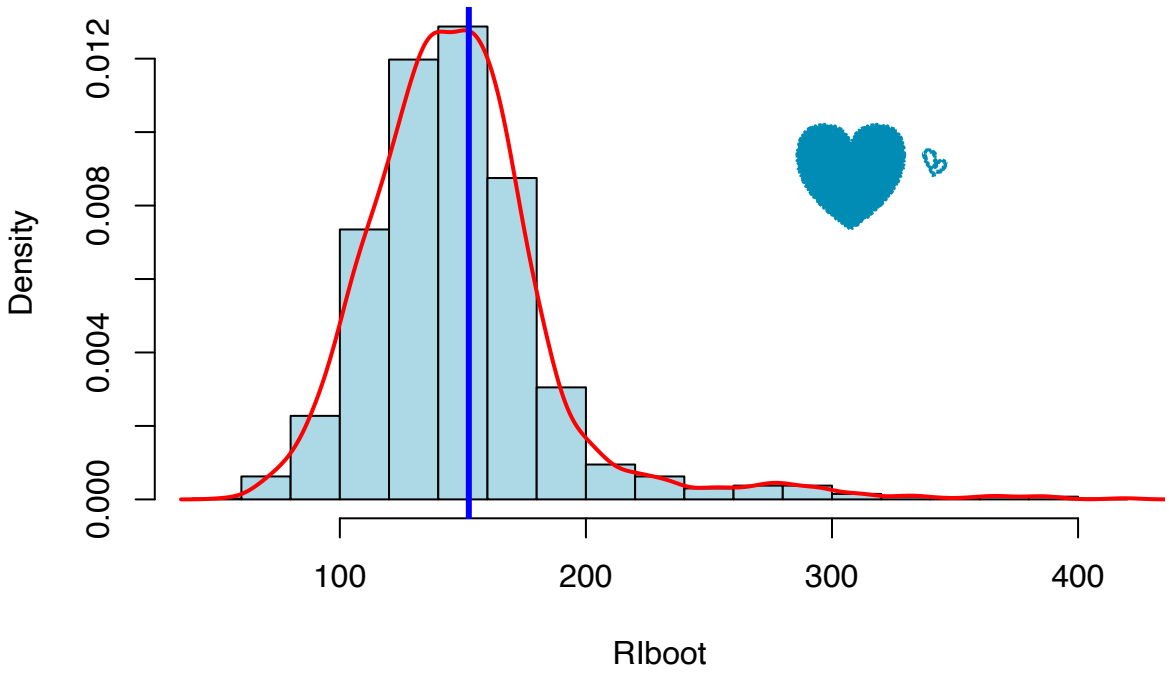
Rango intercuartílico de la muestra

```
  names(res)= "R.I."
  res
}

RI0=RI(x); RI0   Rango Intercuartílico de la muestra Original
```

```
##    R.I.
## 152.425
```

```
B=2000
RIboot= numeric(B)
for (b in 1:B)
{   xboot= sample(x,replace=TRUE)
    RIboot[b]= RI(xboot)
}  #siguiente b
hist(RIboot,br=20,prob=TRUE,
     main="Distribuc. bootstrap del Recorrido Intercuart.",
     col="lightblue",cex.main=0.8)
lines(density(RIboot,bw="SJ"),col="red",lwd=2)
abline(v=RI0,lwd=3,col="blue")
```

Remuestreamos sobre esa muestra original que habíamos seleccionado de los datos del ejercicio 1




```
# sesgo
sesgob=mean(RIboot)-  RI0
#Estimador corregido:
RI0-sesgob
```

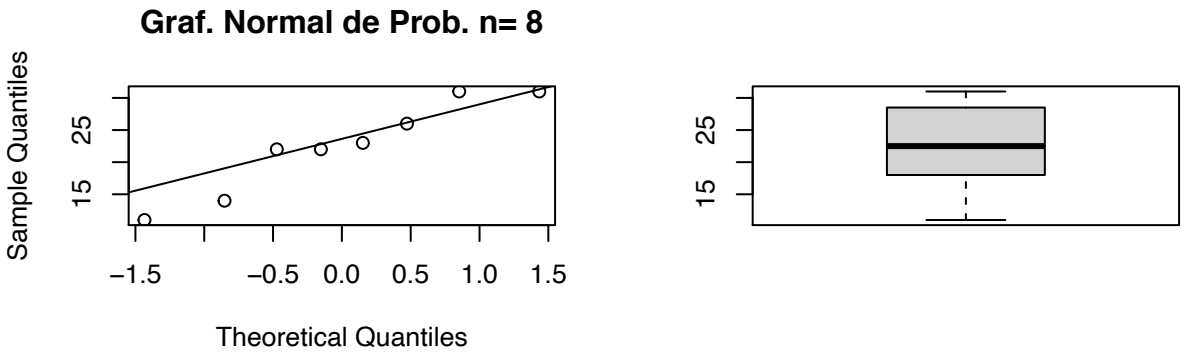
```
##     R.I.
## 156.5924
```

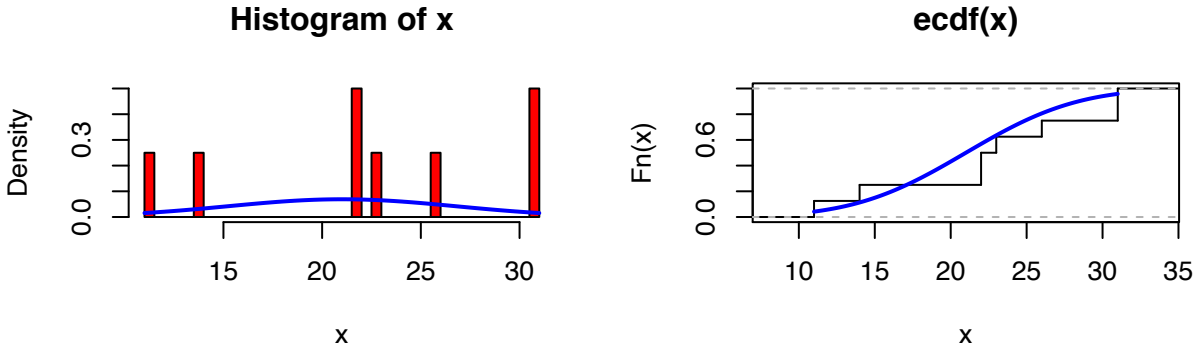
```
#ES
ESBoot= sd(RIboot); ESBoot
```

Error estandar ó cuasidesviación típica de las muestras obtenidas

```
## [1] 39.28767
```

```
#####################################
#Ejemplo 3. Cantidad de vitaminas en 8
#lotes de una mezcla de maíz y soja
#Un contraste de hipótesis mediante el
#bootstrap
#####################################
x=c(26, 31, 23, 22, 11, 22, 14, 31)
ananor(x)
```






```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.91858, p-value = 0.4184
```

```
#según Shapiro-Wilk no se rechaza la normalidad
#pero lo vamos a hacer con el bootstrap

#interesa que la media sea superior a 18, por
#lo que se plantea el siguiente contraste
#H0: E[X]<=18, H1:E[X]>18
#Vamos a usar el estadístico del test-t
t.test(x,mu=18)$statistic
```

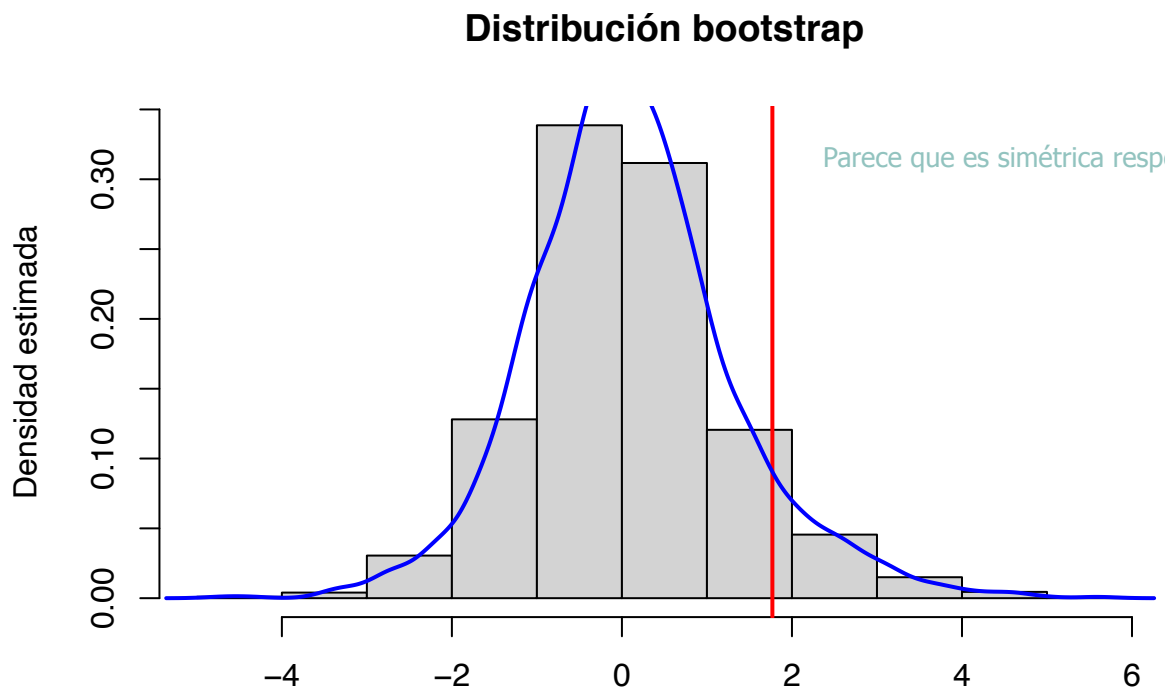
```
##        t
## 1.769914
```

```
#o sea:
(mean(x)-18)*sqrt(length(x))/sd(x)
```

```
## [1] 1.769914
```

```
t0= t.test(x,mu=18)$statistic
B=1999
#Aquí van los B valores del estadístico bootstrap:
taste= numeric(B)
for (b in 1:B)
{
  xboot= sample(x,rep=T)   media de la muestra original
  taste[b]= t.test(xboot,mu=mean(x))$statistic  Estadístico no usando mu=18 de H0
}
#Importante: el estadístico bootstrap compara la media
#de la muestra bootstrap con la media de la muestra disponible
#t.test(xboot,mu=mean(x))  no se pone mu=18
#en las transparencias y el script Bootstrap_ilustrarConsejos.r
#se ilustra este hecho  Mejora el resultado

hist(taste,prob=T,main="Distribución bootstrap",
     xlab="",ylab="Densidad estimada",col="lightgrey")
abline(v=t0,col="red",lwd=2)
lines(density(taste,bw="SJ"),col="blue",lwd=2) Estimación de la función de densidad con el método del núcleo SJ
```

Si la distribución es normal, el estadístico se tiene que distribuir



Parece que es simétrica respecto a 0

```
#Calcular la estimación del p valor
cat("P[T*>=T0]~",(sum(taste>t0)+1)/(B+1),"\n")  En cuantas muestras el estadístico bootstrap es mayor que t0
```

```
## P[T*>=T0]~ 0.083
```

8'3 %

```
#Contraste paramétrico
t.test(x,mu=18,alternative="greater")  No tenemos pruebas para concluir que la media es mayor que 18
```

```
##
##  One Sample t-test
##
## data:  x
```

```
## t = 1.7699, df = 7, p-value = 0.06003
## alternative hypothesis: true mean is greater than 18
## 95 percent confidence interval:
##  17.68304      Inf
## sample estimates:
## mean of x
##      22.5
```

```
#Para el contraste bilateral, el p-valor aproximado
#se calcularía como sigue
```

Calor absoluto de las obs a la derecha del rojo y a la izquierda del valor rojo negativo

```
cat("P[|T*|>=|T0|]~",(sum(abs(taste)>=abs(t0))+1)/(B+1),"\n")
```

```
## P[|T*|>=|T0|]~ 0.1265
```

No rechazo el contraste

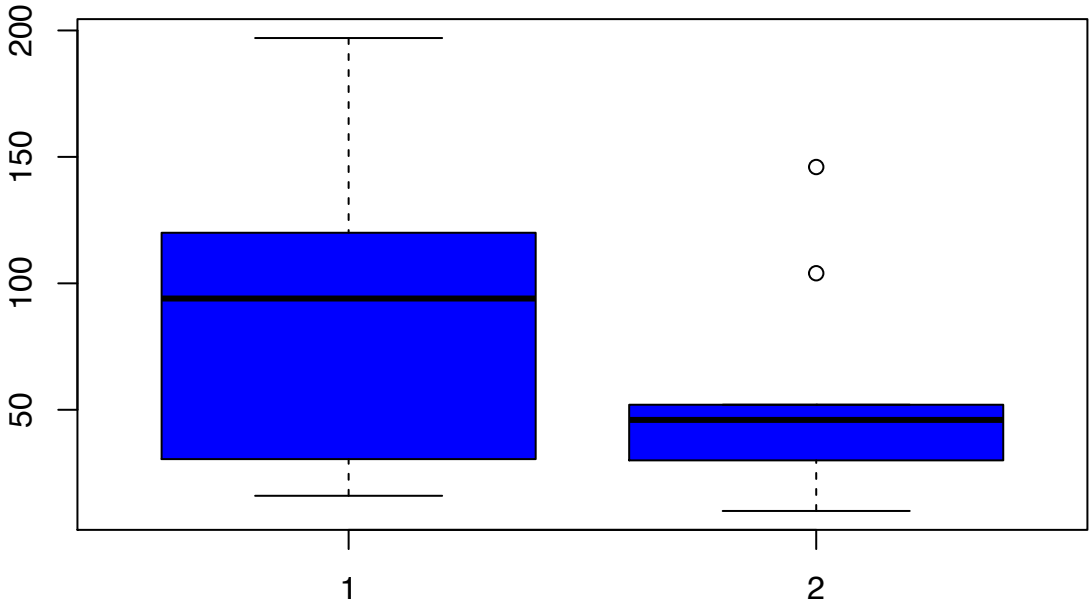
```
##########################################
##Ejemplo 4.Contraste bootstrap bilateral
##comparando las medias de dos poblaciones
##########################################
x=c(94,38,23,197,99,16,141)
y=c(52,10,40,104,51,27,146,30,46)
nx=length(x)
ny=length(y)
summary(x)
```

```
##    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   30.00   46.00   56.22   52.00  146.00
```

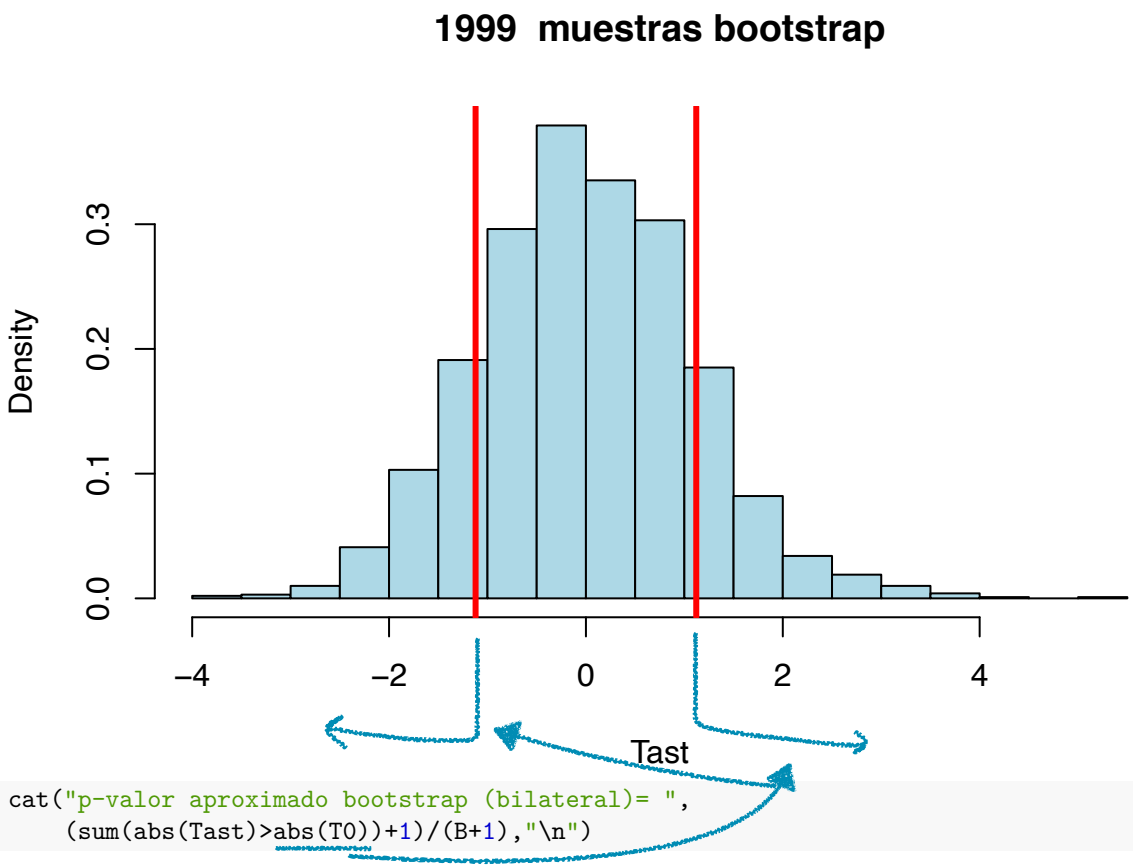
```
boxplot(x,y,col="blue")
```




```
B=1999
T0=t.test(x,y,var.equal=TRUE)$statistic
Tast=numeric(B)
```

Supongo hipótesis de homocedasteceidad, y ambas distribuciones prodecen de la misma distribución por lo que ambas varianzas han de ser iguales. Hago el t-test, por lo que obtengo el estadístico t-student

```
#las muestras bootstrap se forman con elementos
#extraídos de forma indistinta de cualquiera de
#las submuestras originales
xy=c(x,y)
for (b in 1:B)
{xast=sample(xy,nx,replace=TRUE) #Muestreo bajo HO
 yast=sample(xy,ny,replace=TRUE) #Muestreo bajo HO
 Tast[b]=t.test(xast,yast,var.equal=TRUE)$statistic
}

hist(Tast,br=30,prob=TRUE,
     main=paste(B," muestras bootstrap"),
     col="lightblue")
abline(v=T0, lwd=3,col="red")
abline(v=-T0, lwd=3,col="red")
```




```
cat("p-valor aproximado bootstrap (bilateral)= ",
    (sum(abs(Tast)>abs(T0))+1)/(B+1),"\n")
```

```
## p-valor aproximado bootstrap (bilateral)=  0.285
```

No rechazo

```
##########################################
##Ejemplo 5. Intervalo de confianza
##para la diferencia de medias de dos poblaciones
#mismos datos del ejemplo anterior
##########################################
x=c(94,38,23,197,99,16,141)
y=c(52,10,40,104,51,27,146,30,46)
nx=length(x)
ny=length(y)
```

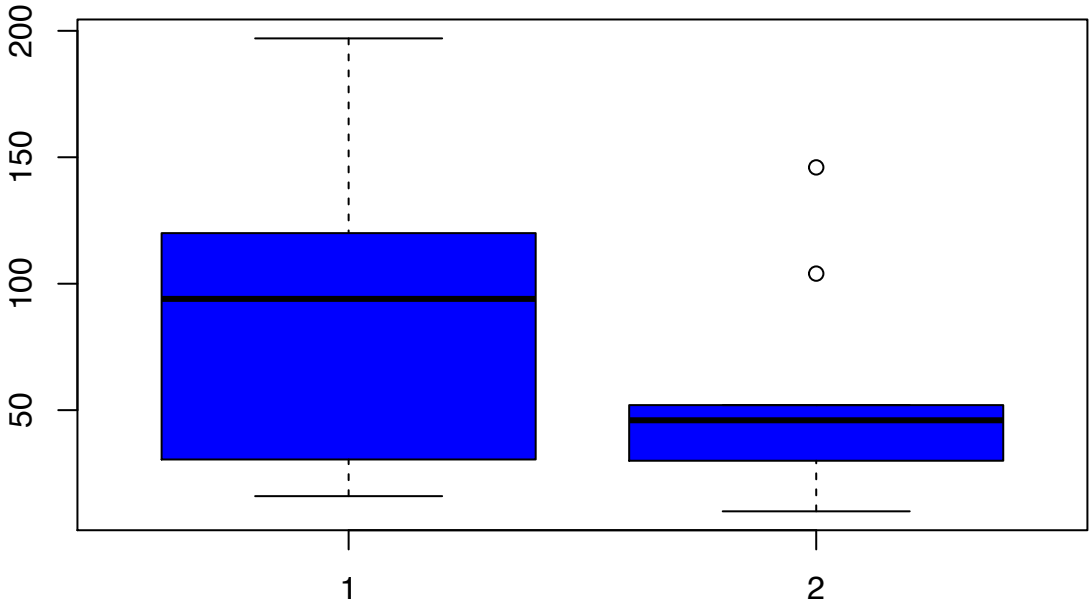
```
summary(x)
```

```
##    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   30.00   46.00   56.22   52.00  146.00
```

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




```
B=2000
dife0=mean(x)-mean(y)
dife0
```

```
## [1] 30.63492
```

La de x es bastante mayor que la de y (30ud)

```
difeast=numeric(B)

#Para calcular IC, como no hay una hipótesis nula
#H0, se muestrea por separado
for (b in 1:B)
{xast=sample(x,replace=TRUE) #Muestra boot de x
 yast=sample(y,replace=TRUE) #Muestra boot de y
 difeast[b]=mean(xast)-mean(yast)
}

hist(difeast,br=30,prob=TRUE,
     main=paste(B," muestras bootstrap"),
     xlab="diferencia de medias",
     col="lightblue")
#Si la forma de la distribución boostrap es razonablemente
#normal, se pueden usar el método normal y/o el método percentil
#para calcular IC

#Método percentil, simplemente los cuantiles de difeast
```

No tengo hipótesis, sólo quiero IC

Calculo el estadístico para cada una de las muestras

```
alfa= 0.05
ICperc=quantile(difeast,prob=c(alfa/2,1-alfa/2))
cat("ICBootstrap (Percentil) para la diferencia de medias al 95% :\n(",
   ICperc[1],",",
  ICperc[2],")\n")
```

Método percentil: calculo cuantiles

```
## ICBootstrap (Percentil) para la diferencia de medias al 95% :
## ( -18.38611 , 83.63611 )
```

Ojo, es una errata y coge de otro

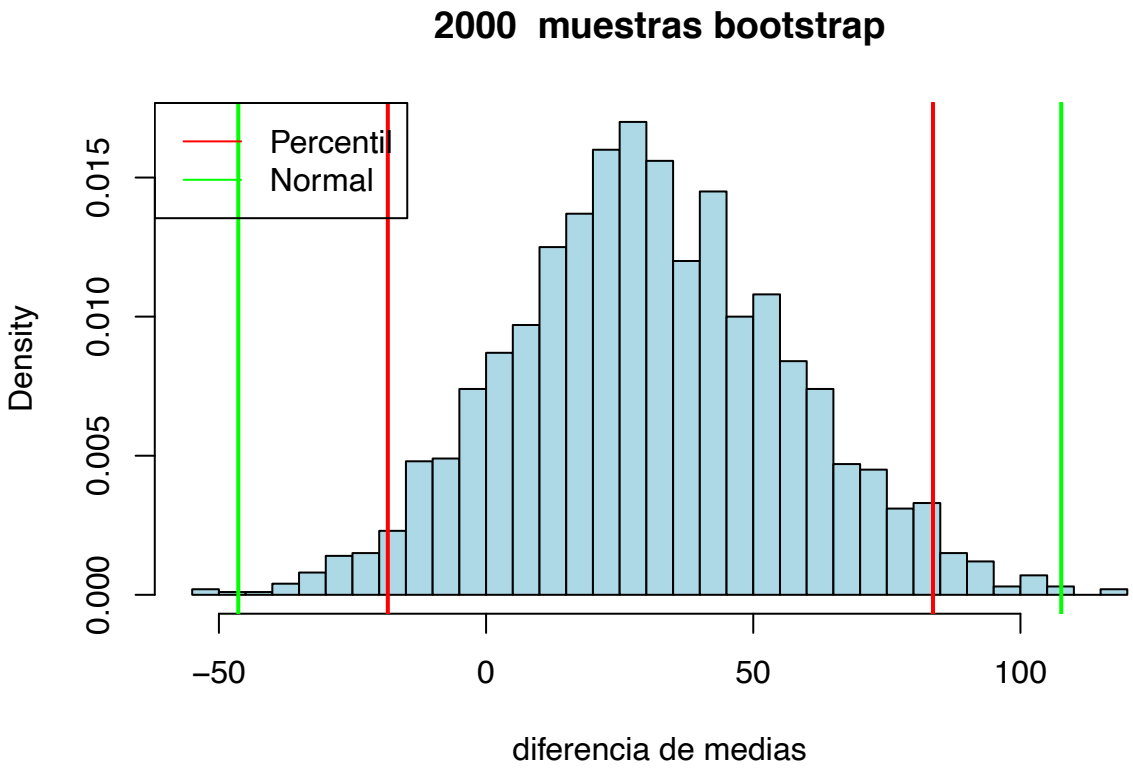
calcular ESBoot = sd(dife0)= 27.03379

```
#Método normal
cuantil= qnorm(1-alfa/2)
ICnormal=c(dife0- cuantil*ESBoot,dife0+ cuantil*ESBoot)
cat("ICBootstrap (Normal) para la diferencia de medias al 95% :\n(",
    ICnormal[1],",",
    ICnormal[2],")\n")
```

```
## ICBootstrap (Normal) para la diferencia de medias al 95% :
## ( -46.3675 , 107.6373 )
```

(-22,83)

```
abline(v=ICperc,col="red",lwd=2)
abline(v=ICnormal,col="green",lwd=2)

legend("topleft",col=c("red","green"),lty=1,
       legend=c("Percentil","Normal"))
```




```
#####################################################################
##Ejemplo 6. Una tabla de contingencia, contraste de independencia
##y cálculo de IC para la diferencia de probabilidades
#####################################################################
#    Infarto:    SI     NO    Total
# Fuman          12     12   | 24
```

```
# No Fuman        3    16   | 19
#H0:P[Infarto=SI/Fumar]=P[Infarto=SI/No Fuman]
#H1:P[Infarto=SI/Fumar]!=P[Infarto=SI/No Fuman]
fuman=24
fumaneinf=12
nofuman=19
nofumaneinf=3

RR=(fumaneinf/fuman)/(nofumaneinf/nofuman)
cat(" Frec. relativa de infartos en fumadores=",fumaneinf/fuman,"\n",
    "Frec. relativa de infartos en no fumadores=",nofumaneinf/nofuman,"\n",
    "Riesgo relativo de sufrir infarto Fumador/No fumador= ",RR,"\n")
```

```
##  Frec. relativa de infartos en fumadores= 0.5
##  Frec. relativa de infartos en no fumadores= 0.1578947
##  Riesgo relativo de sufrir infarto Fumador/No fumador=  3.166667
```

Casi un tercio la segunda

```
#Construir una tabla con los datos
tabla= matrix(c(fumaneinf,fuman-fumaneinf,
                nofumaneinf,nofuman-nofumaneinf),byrow=T,2,2)
rownames(tabla)= c("Fuman","No Fuman")
colnames(tabla)=c("Infarto","No Infarto")
tabla
```

```
##          Infarto No Infarto
## Fuman         12         12
## No Fuman       3         16
```

La muestra de partida bootstrap es la tabla

```
100*prop.table(tabla,1)
```

```
##           Infarto No Infarto
## Fuman    50.00000   50.00000
## No Fuman 15.78947   84.21053
```

```
100*prop.table(tabla,2)
```

```
##          Infarto No Infarto
## Fuman         80   42.85714
## No Fuman      20   57.14286
```

```
#Se puede hacer el contraste de independencia con el test-chicuadrado
chisq.test(tabla)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tabla
## X-squared = 4.0616, df = 1, p-value = 0.04387
```

Con un nivel de sig del 5%, esto nos lleva a rechazar la independencia y la igualdad de probabilidades

```
#Si fallaran las reglas de Cochran
#en tablas 2x2:frecuencias esperadas >5
#podemos recurrir al bootstrap

#5.1 Contraste de independencia bootstrap
#----------------------------------------
#Para aplicar el bootstrap se puede elegir el estadistico del test
#chi-cuadrado; como depende de  los valores observados, que
```

```r
#son fijos, y de los esperados, que se calculan bajo H0,
#se puede muestrear sobre los valores esperados,
#que al dividir por n dan las probabilidades bajo H0
#Recordemos que el estadístico es la suma de las (obs-esp)^2/esp
#donde las frecuencias esperadas (bajo H0) están disponibles en
#el elemento expected

resul = chisq.test(tabla)
resul
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tabla
## X-squared = 4.0616, df = 1, p-value = 0.04387
```

```r
(p.hat = resul$expected / sum(tabla))
```

```
##            Infarto No Infarto
## Fuman    0.1946998  0.3634397
## No Fuman 0.1541374  0.2877231
```

Divido por n, el total de observaciones
y hallo las estimaciones de las probs conjuntas
de cada una de las celdas

```r
#Probabilidades conjuntas estimadas bajo H0
```

Son indepe

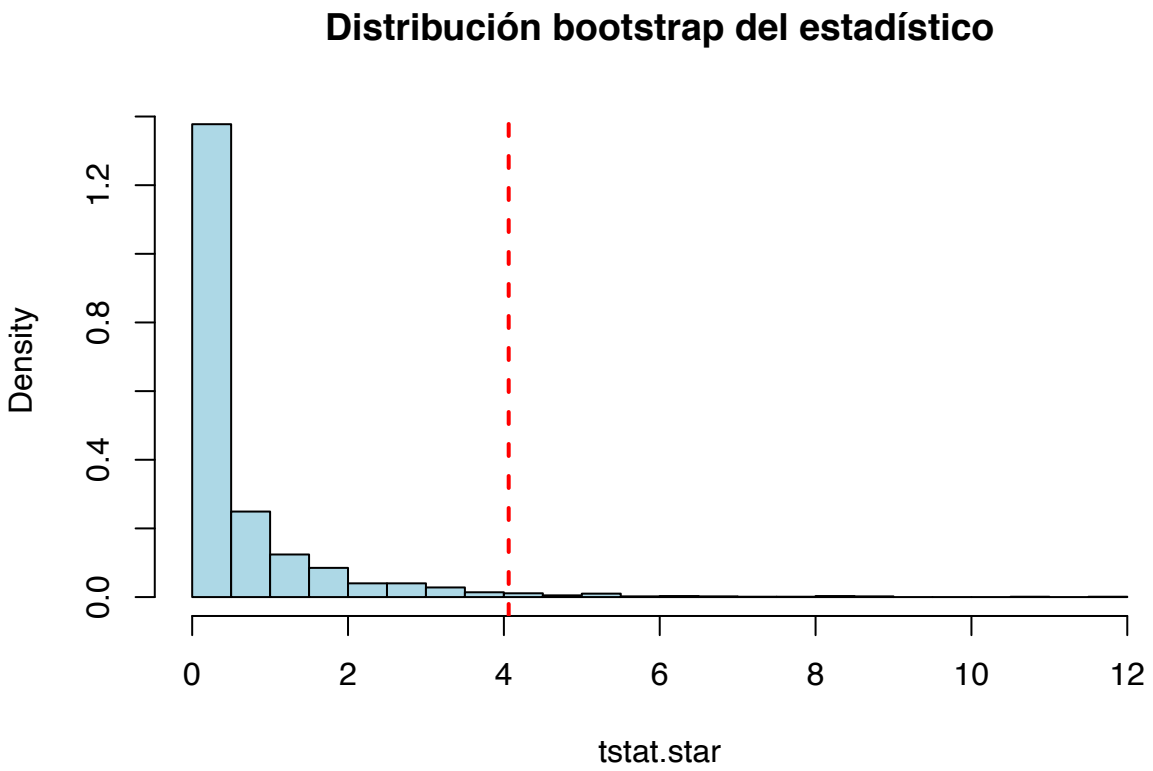
```r
#Estadístico calculado en la tabla original:
(tstat.hat = resul$statistic)
```

```
## X-squared
##  4.061616
```

```r
#Cada muestra bootstrap se puede obtener muestreando con
#reemplazamiento, con tamaño n, el conjunto de las cuatro posiciones
#de la tabla, cada posición con probabilidad la que aparece en p.hat
n = sum(tabla)
k = length(tabla)   #Número de celdas=nrow(x)*ncol(x)  2*2=4
B = 1999
tstat.star = numeric(B)   #Estadístico bootstrap
for (i in 1:B) {
  #Muestra bootstrap de la tabla conjunta, p.hat (bajo H0):
  y.star = sample(1:k, n, replace = TRUE,
                  prob = as.numeric(p.hat))
  #as.numeric(p.hat) pasa la matriz a un vector
  #recorriendo las columnas
  #Disponer como tabla la muestra y.star
  #el siguiente tabulate guarda un 0 para aquellas posiciones
  #que correspondan a elementos no observados en y.star
  #tabulate va a contar cuántas veces aparece el 1,2,3,4 (k)
  tabla.star = matrix(tabulate(y.star, k), 2, 2)
  suppressWarnings({resul.boot = chisq.test(tabla.star)})
  tstat.star[i]=resul.boot$statistic
  #podría dar errores si una fila o columna está vacía
}

cat("P-valor bootstrap =",
    (sum(tstat.star >= tstat.hat) + 1) / (B + 1),"\n")
```

```
## P-valor bootstrap = 0.0215
```

```
hist(tstat.star,br=20,prob=TRUE,
     col="lightblue",main="Distribución bootstrap del estadístico")
abline(v = tstat.hat, lty = 2,lwd=2,col="red")
```



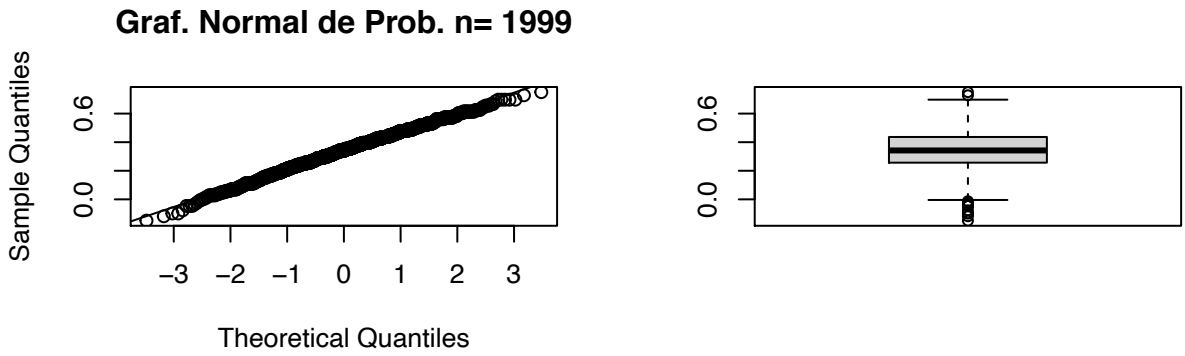

```
#5.2 Intervalos de Confianza boostrap
#--------------------------------------
#IC-bootstrap (normal y percentil) para la diferencia de probabilidades
#P[Infarto=SI/Fumar]-P[Infarto=SI/No Fuman]
#En este caso se muestrea por separado !!!
alfa=0.05
difeboot=numeric(B)
dife0= (fumaneinf/fuman)-(nofumaneinf/nofuman)
cat("Diferencia de proporciones observadas en la tabla=",dife0,"\n")
```

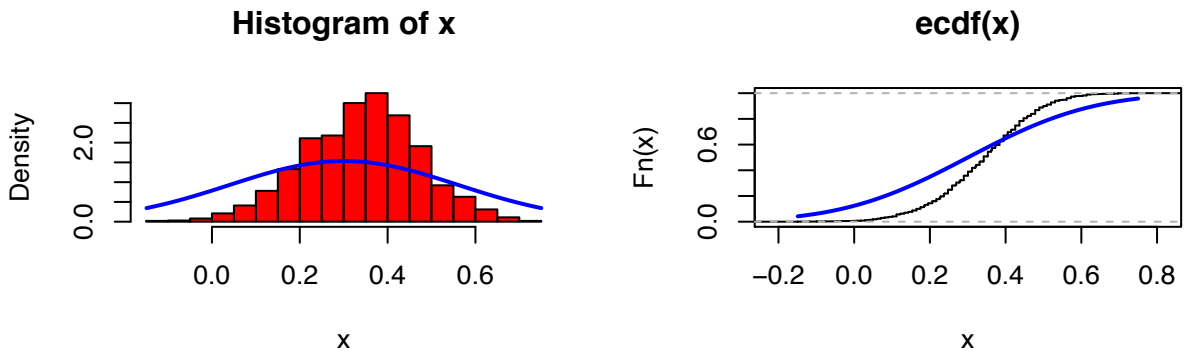
Método de muestreo es por separado, porque tengo por un lado los fumadores y por otro los no fumadores

```
## Diferencia de proporciones observadas en la tabla= 0.3421053
```

```
for (i in 1:B)
{
  #Se muestrea por separado, por ejemplo en los fumadores
  #se le ha asignado un número de orden entre 1 fuman
  #comparando con el total de fumaneinf obtenemos la proporción a
  a=sum(sample(1:fuman,rep=TRUE)<=fumaneinf)/fuman
  b=sum(sample(1:nofuman,rep=TRUE)<=nofumaneinf)/nofuman
  difeboot[i]= a-b
}
ananor(difeboot)
```

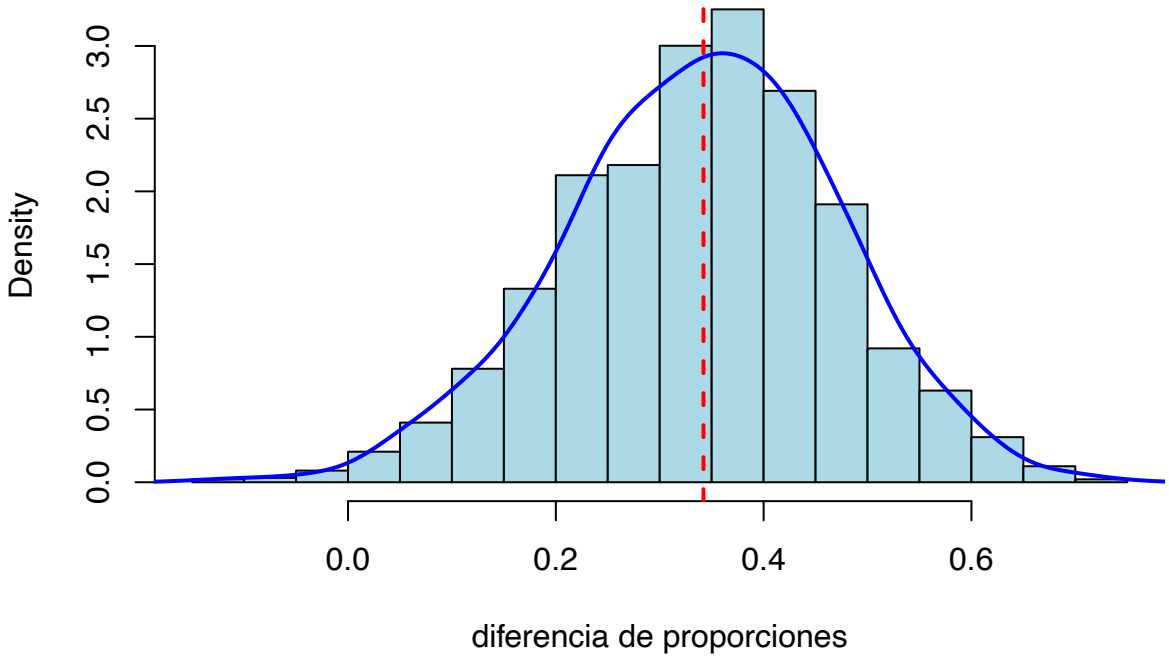
Genero un valor entre 1 y 24 con reemp y miro si es menor o igual que 12.
Cuento de la muestra cuantos han sido menor o igual q 12 (han tenido infarto) y divido por los que fuman: obtengo la frecuencia relativa.
12 es el valor frontera.






```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.99716, p-value = 0.001022
```

```
hist(difeboot,br=20,prob=TRUE,
     xlab="diferencia de proporciones",
     col="lightblue",main="Distribuc. bootstrap")
abline(v = dife0, lty = 2,lwd=2,col="red")
lines(density(difeboot,bw="SJ"),col="blue",lwd=2)
```




```
#Parece que hay ligera asimetría a la izquierda, se podría
#utilizar el método BCa que se verá en el script siguiente
cat(" Intervalo de confianza normal 95%= (",
    dife0-qnorm(1-alfa/2)*sd(difeboot), ", ",
    dife0+qnorm(1-alfa/2)*sd(difeboot),") \n",
    "Intervalo de confianza Percentil 95%= (",
    quantile(difeboot,probs=c(alfa/2,1-alfa/2)), ") \n")
```

Metodo percentil: quantiles de nivel alfa/2 y 1-alfa/2

```
##  Intervalo de confianza normal 95%= ( 0.08429555 ,  0.599915 )
##  Intervalo de confianza Percentil 95%= ( 0.07017544 0.5833333 )
```

Ejemplosbootstrap_II

Pedro Luque

Ejemplo 6

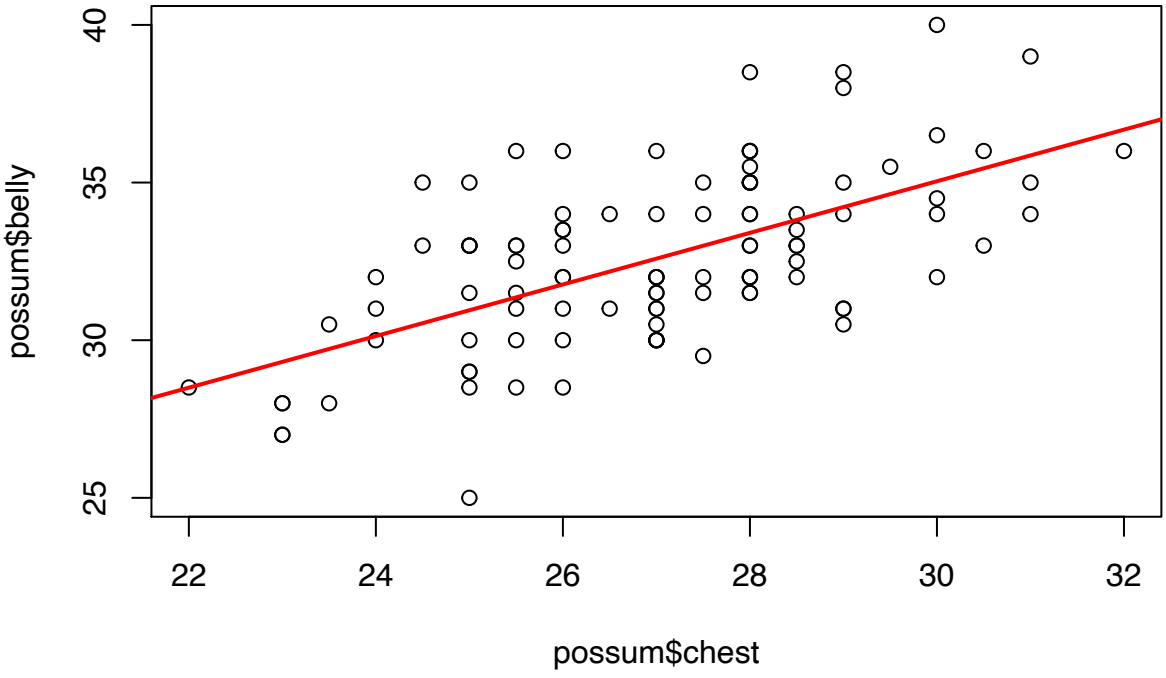
```
source("ananor.r")
#######################################################################
#Ejemplo 6. Coeficiente de correlac. lineal, transform. Z de Fisher
#Vamos a usar el paquete boot
#######################################################################
library(DAAG)  #Para acceder a estos datos
```

```
## Loading required package: lattice
```

```
data(possum)  #comadrejas
?possum
z.transform = function(r) {.5*log((1+r)/(1-r))}
z.inversa = function(z) (exp(2*z)-1)/(exp(2*z)+1)
plot(possum$chest,possum$belly)  #torso y barriga
cor(possum$chest,possum$belly)
```

```
## [1] 0.6061696
```

```
regre= lm(possum$belly~possum$chest)
abline(regre,col="red",lwd=2)
```



```
summary(regre)
```

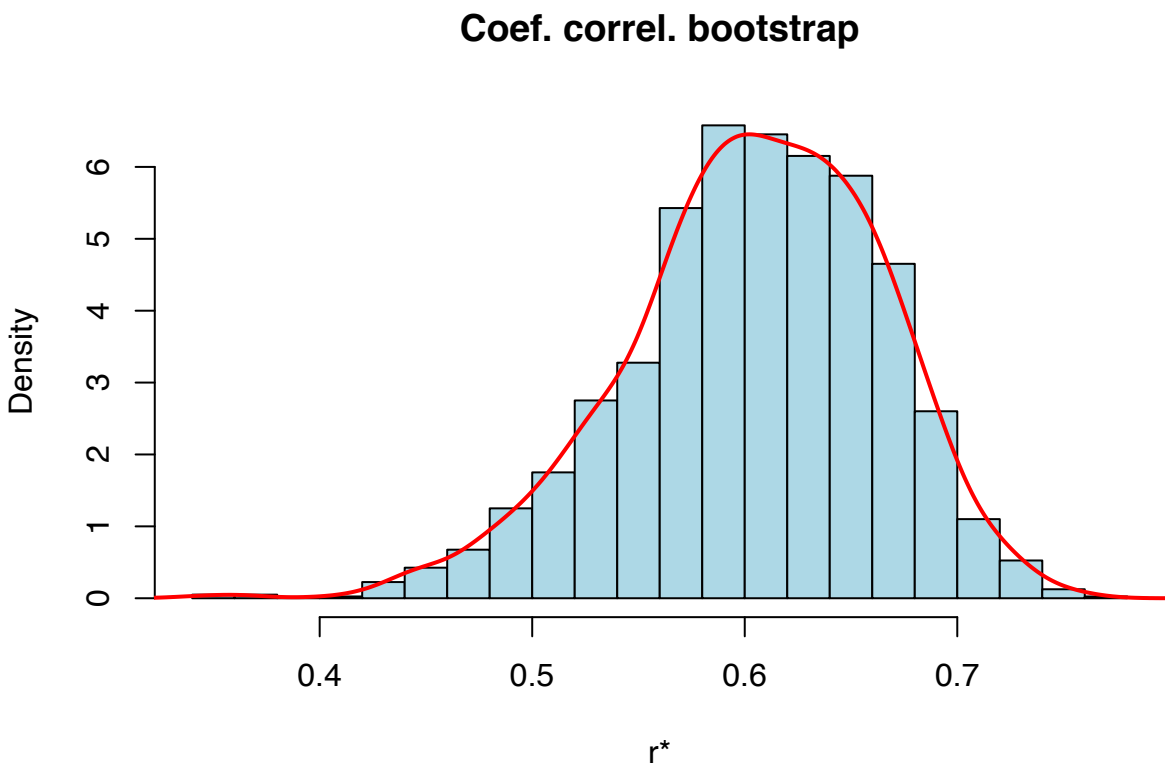
```
##
## Call:
## lm(formula = possum$belly ~ possum$chest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9496 -1.5865 -0.3819  1.6065  5.0950
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.4885     2.8790   3.643 0.000426 ***
## possum$chest   0.8184     0.1063   7.697 9.19e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.207 on 102 degrees of freedom
## Multiple R-squared:  0.3674, Adjusted R-squared:  0.3612
## F-statistic: 59.25 on 1 and 102 DF,  p-value: 9.187e-12
```

```
library(boot)
```

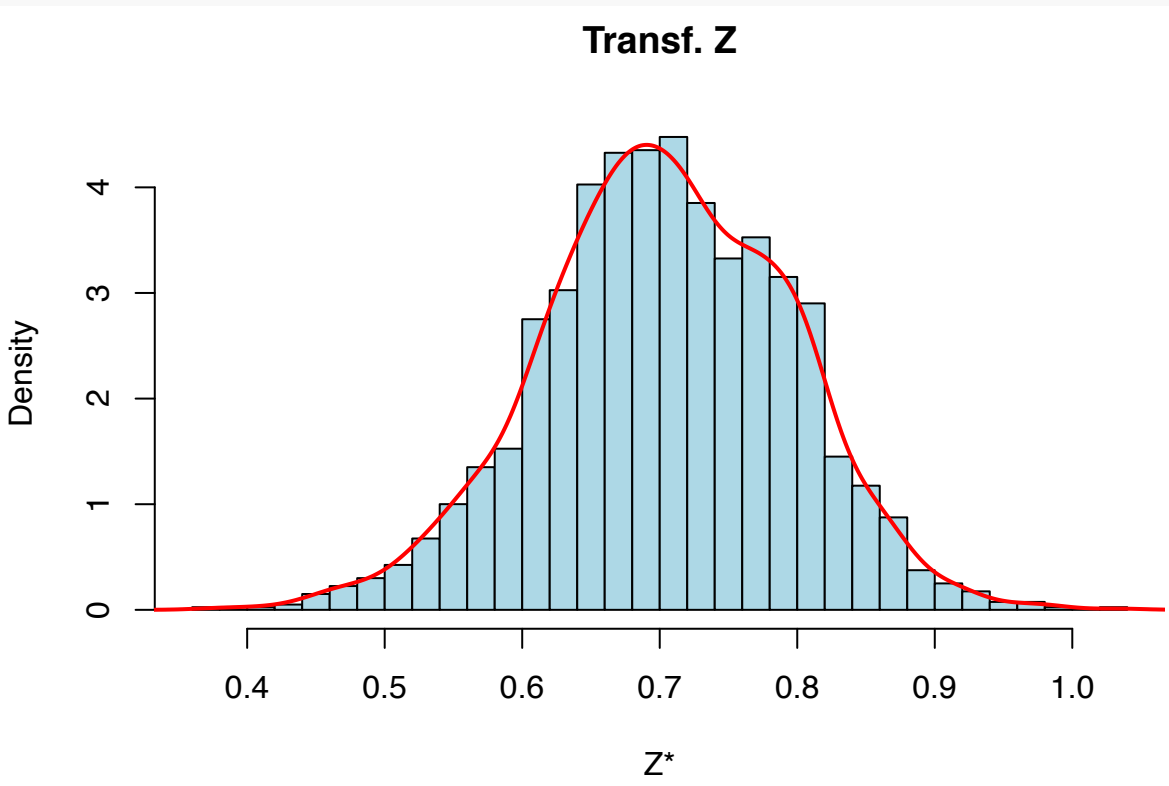
```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
#Para usar este paquete, hay que definir una funcion que dependa
#de los indices para seleccionar elementos
#i1 e i2 son las posiciones de las dos variables cuyo coef.corr.
#se desea estudiar

cor.fun = function(datos,i1,i2, indices) {
x = datos[indices,i1]
y = datos[indices,i2]
cor(x, y)} #r directamente

zcor.fun = function(datos,i1,i2, indices) {
x = datos[indices,i1]
y = datos[indices,i2]
z.transform(cor(x, y))} #Con transf. Z de Fisher

possum.boot1 = boot(possum, cor.fun, i1=13,i2=14, R=1999)
possum.boot2 = boot(possum, zcor.fun,i1=13,i2=14, R=1999)
hist(possum.boot1$t,prob=T,br=30,col="lightblue",
     main="Coef. correl. bootstrap",xlab="r*")
lines(density(possum.boot1$t,bw="SJ"),col="red",lwd=2)
```




```
hist(possum.boot2$t,br=30,prob=T,col="lightblue",
     main="Transf. Z",xlab="Z*")   #Más simétrica
lines(density(possum.boot2$t,bw="SJ"),col="red",lwd=2)
```




```
#IC percentil, normal y BCa con transformac. inversa
z.inversa(boot.ci(possum.boot2, type="perc")$percent[4:5])
```

```
## [1] 0.4781624 0.7021606
```

```
z.inversa(boot.ci(possum.boot2, type="norm")$normal[2:3])
```

```
## [1] 0.4816745 0.7058855
```

```
z.inversa(boot.ci(possum.boot2, type="bca")$bca[4:5])
```

```
## [1] 0.4775746 0.7019402
```

```
#Sin aplicar la transformac.:
boot.ci(possum.boot1, type="perc")$percent[4:5]
```

```
## [1] 0.4755813 0.7087909
```

```
boot.ci(possum.boot2, type="norm")$normal[2:3]
```

```
## [1] 0.5251624 0.8789353
```

```
boot.ci(possum.boot2, type="bca")$bca[4:5]
```

```
## [1] 0.5198376 0.8711150
```

EjemplosJackknife

Pedro Luque

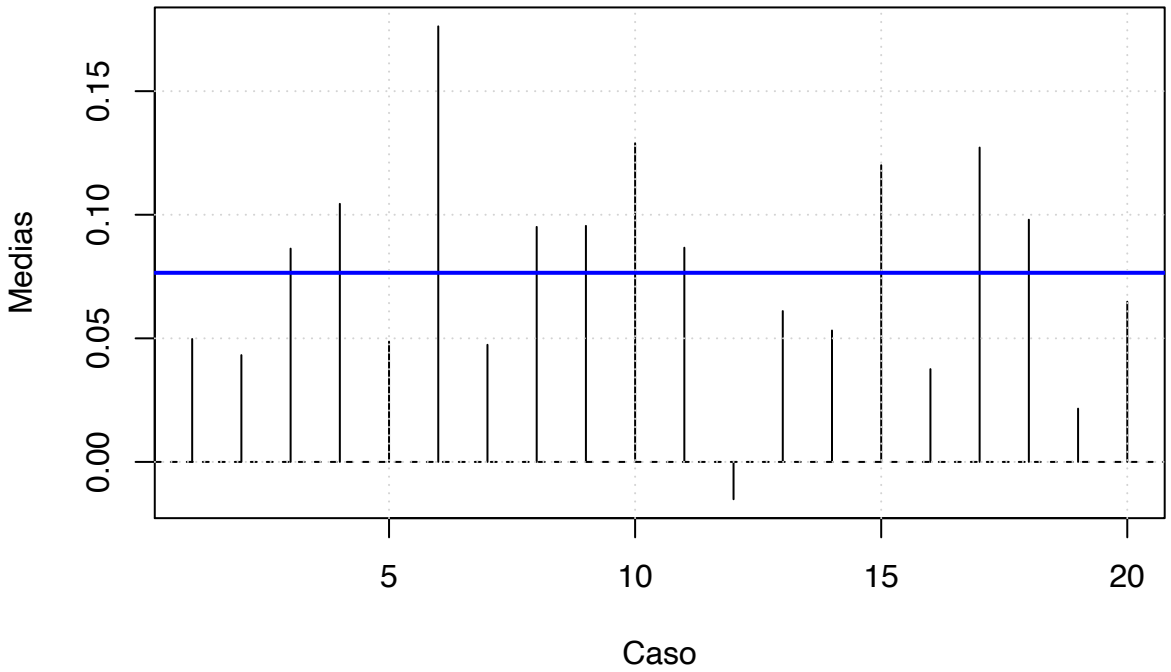
```
##########################################
##ESTADISTICA COMPUTACIONAL I             #
##GRADO EN ESTADISTICA                    #
##DOBLE GRADO EN MATEMATICAS Y ESTADISTICA #
##EJEMPLOS JACKKNIFE                      #
##########################################

###############################################
##Ejemplo 1. Estimac. jackknife del sesgo
##y la varianza de la media muestral
##como estimador de la media poblacional
###############################################

#Primero, generar aleatoriamente los datos
set.seed(12345)
x<-rnorm(20)
x
```

```
##  [1]  0.5855288  0.7094660 -0.1093033 -0.4534972  0.6058875 -1.8179560
##  [7]  0.6300986 -0.2761841 -0.2841597 -0.9193220 -0.1162478  1.8173120
## [13]  0.3706279  0.5202165 -0.7505320  0.8168998 -0.8863575 -0.3315776
## [19]  1.1207127  0.2987237
```

```
#Calcular el estadístico en cada una de las
#submuestras de tamaño n-1
n<-length(x)
ti<-numeric(n) #Aquí se guardarán los n valores a generar
t<-mean(x)
for (i in 1:n)
 ti[i]<-mean(x[-i])
ti
```

```
##  [1]  0.04972670  0.04320369  0.08629682  0.10441228  0.04865520  0.17622590
##  [7]  0.04738093  0.09508001  0.09549979  0.12892938  0.08666232 -0.01510399
## [13]  0.06103728  0.05316420  0.12004569  0.03754928  0.12719441  0.09799546
## [19]  0.02155913  0.06482171
```

```
plot(ti,type="h",xlab="Caso",ylab="Medias",
     main="Medias sin el caso i")
abline(h=0,lty=2)
abline(h=t,col="blue",lwd=2)
grid()
```




```
#Estimación Jackknife del sesgo de la media:
sesgoj<- (n-1)*(mean(ti)-t); sesgoj
```

```
## [1] 0
```

```
#Por tanto coincide con el valor real
#del sesgo de la media aritmética

#Aquí no hace falta, pero el estimador incluyendo
#la corrección del sesgo
tjack<- t-sesgoj; tjack
```

```
## [1] 0.07651681
```

```
#Estimación Jackknife de la varianza
#según las transparencias, se define como
#(n-1)*varianza(ti),  varianza dividiendo por n
#como la orden var de R divide por (n-1), o sea,
#calcula la cuasiv, podemos obtener var mediante (n-1)*cuasivar/n
varj<- (((n-1)^2)/n)*var(ti); varj
```

```
## [1] 0.03477242
```

```
var(x)/n #cuasivar/n
```

```
## [1] 0.03477242
```

```
#El estimador jack de var(xmedia) coincide
#con el estimador insesgado cuasivar/n


#########################################################
##Ejemplo 2. Estimac. jackknife del sesgo
```

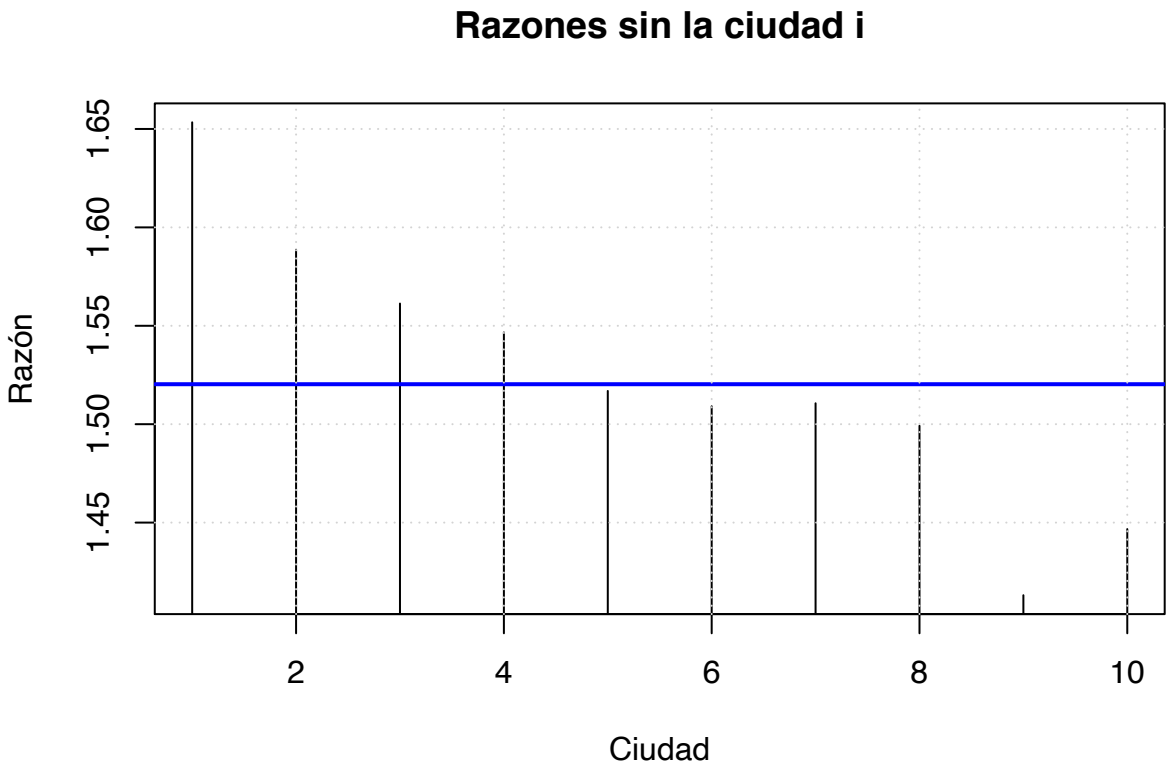
```r
##y la varianza del estimador de la razón (cociente
#de los totales de las variables x y u) en el fichero
#city de R (fichero disponible en la librería boot):
#########################################################
library(boot)
data(city)
#city
#?city
#EStimador
print(R<- sum(city$x)/sum(city$u) )
```

```
## [1] 1.520312
```

```r
#La población total de estas ciudades aumentó el 52,03%
#en el periodo 1920-1930

#Jackknife
n<-nrow(city)
Ri<-numeric(n) #Aquí se guardarán los n valores a generar
for (i in 1:n)
  Ri[i]<- sum(city$x[-i])/sum(city$u[-i])
Ri
```

```
##  [1] 1.653386 1.588665 1.561313 1.546638 1.516892 1.509121 1.510638 1.499190
##  [9] 1.413115 1.446708
```

```r
plot(Ri,type="h",xlab="Ciudad",ylab="Razón",
     main="Razones sin la ciudad i")
abline(h=R,lwd=2,col="blue")
grid()
```



```r
#Estimación Jackknife del sesgo
sesgoj<- (n-1)*(mean(Ri)-R); sesgoj
```

```
## [1] 0.03828722
```

```r
#Estimador incluyendo la corrección del sesgo
Rjack<- R-sesgoj; Rjack
```

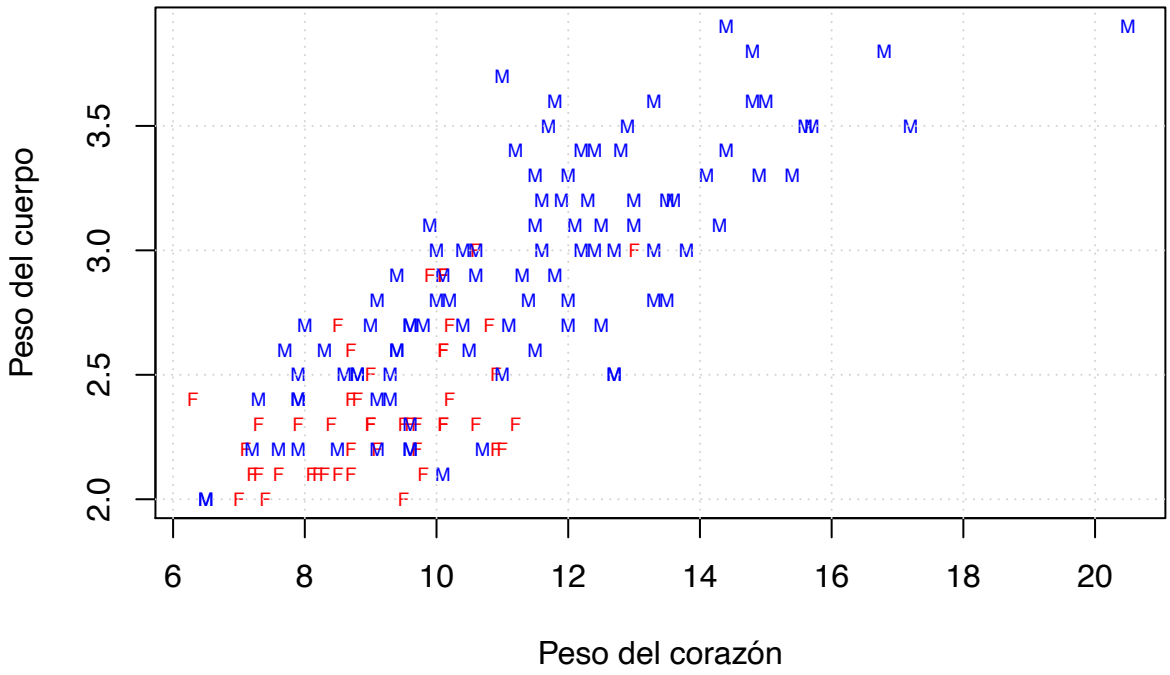
```
## [1] 1.482025
```

```r
#Corrección: la población total de estas ciudades aumentó el 48,2%%
#en el periodo 1920-1930

#Estimación Jackknife de la varianza del estadístico razón
varj<- (((n-1)^2)/n)*var(Ri); varj
```

```
## [1] 0.03794353
```

```r
##########################################################
#Ejemplo 3. Estimac. jackknife de la tasa de acierto
#de la FLD de Fisher
##########################################################
library(MASS)
data(cats)
#?cats
#Se tiene 144 gatos, de cada uno se conoce el peso del cuerpo
#y el peso del corazón
#El objetivo es construir una regla de clasificación que, a partir
#del conocimiento de ambas variables, proporcione una estimación del
#sexo del gato
colores<-c("red","blue")
attach(cats)
plot(Hwt,Bwt,type="n",main="Fichero Gatos",
     xlab="Peso del corazón",
     ylab="Peso del cuerpo")
text(Hwt,Bwt,Sex,cex=0.6, col=colores[Sex])
grid()
```

Fichero Gatos




```
#Analisis Lineal Discriminante de Fisher
modeloADFemp <- lda(Sex~Hwt+Bwt,cats)
modeloADFemp
```

```
## Call:
## lda(Sex ~ Hwt + Bwt, data = cats)
##
## Prior probabilities of groups:
##         F         M
## 0.3263889 0.6736111
##
## Group means:
##         Hwt      Bwt
## F  9.202128 2.359574
## M 11.322680 2.900000
##
## Coefficients of linear discriminants:
##             LD1
## Hwt -0.02986042
## Bwt  2.53019769
```

```
#La regla de clasificación se basa en la combinación lineal
#definida por los coeficientes que aparecen bajo LD1
#se accede mediante predict(modeloADFemp)$x
#si es negativa se clasifica en F, en otro caso M

#Se pretende estimar el rendimiento del modelo de clasificación
#Sin embargo, el rendimiento de un modelo predictivo no puede
#medirse sobre los mismos datos usados para construirlo,
#por ejemplo si calculamos
```

```r
predi=predict(modeloADFemp)$class
tablaemp=table(Real=cats$Sex,Pred=predi)
tablaemp
```

```
##     Pred
## Real  F  M
##    F 31 16
##    M 12 85
```

```r
#Porcentaje de acierto:
100*(sum(diag(tablaemp)/sum(tablaemp)))
```

```
## [1] 80.55556
```

```r
#La estimación de la capacidad de generalización que ofrece
#este procedimiento, 80,55% es optimista, en general cabe esperar
#un rendimiento inferior

#Una alternativa para estimar ese rendimiento es el jackknife
#para ello hay que establecer CV=TRUE:
modeloADF <- lda(Sex~Hwt+Bwt,cats, CV=TRUE)
names(modeloADF)
```

```
## [1] "class"     "posterior" "terms"     "call"      "xlevels"
```

```r
#CV=TRUE significa utilizar Jackknife, de ese modo, el objeto resultante modeloADF
#contiene en el elemento class la clasificación de cada caso utilizando
#el modelo construido sobre el resto de casos

prediJack<- modeloADF$class
prediJack
```

```
##   [1] F F F F F F F F F F F F F F F F F F F F F F F F F F F M M M F M M M
##  [38] M M M M M M M M M M F F F F F F F F F F F F M M M F F M M M M M M M M M M
##  [75] M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M
## [112] M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M M
## Levels: F M
```

```r
#La siguiente tabla permite resumir el rendimiento
#del modelo, se suele conocer como matriz de confusión
tabla<- table(Real=Sex,Pred=prediJack); tabla
```

```
##     Pred
## Real  F  M
##    F 31 16
##    M 14 83
```

```r
#Aciertos:
#  31 gatas son clasificadas (mediante VC) correctamente como gatas
#  83 gatos son clasificados (mediante VC) correctamente como gatos
#Errores:
#  16 gatas con clasificadas (mediante VC) erróneamente como gatos
#  14 gatos con clasificadas (mediante VC) erróneamente como gatas

#Como estos son totales absolutos, es preferible expresarlos de forma relativa:
#Porcentajes de acierto
100*diag(prop.table(tabla,1))
```

```
##        F        M
## 65.95745 85.56701
```

```
#Se estima que asi el 66% de las gatas son clasificadas correctamente,
#mientras que se estima que el 85,6% de los gatos son clasificados correctamente

100*sum(diag(tabla))/sum(tabla)
```

```
## [1] 79.16667
```

```
#Interpretación:
#Con este modelo cabe esperar un acierto global del 79.17%


#Otra forma de calcular estos indicadores sin construir la tabla anterior,
#que suele conocerse como matriz de confusión
#Tasa de acierto
100*mean(Sex==  prediJack)
```

```
## [1] 79.16667
```

```
#Acierto dentro de M
100*sum(Sex=="M" & Sex==prediJack) /sum(Sex=="M")
```

```
## [1] 85.56701
```

```
#Acierto dentro de F
100*sum(Sex=="F" & Sex==prediJack) /sum(Sex=="F")
```

```
## [1] 65.95745
```

```
#Cómo calcular directamente las predicciones Jackknife
#sin usar CV=TRUE
n=nrow(cats)
clasiJ=character(n)
for (i in 1:n)
{
  modeloADFi = lda(Sex~Hwt+Bwt,cats[-i,])
  clasiJ[i]=as.character(predict(modeloADFi,newdata=cats[i,],
                                 prior=modeloADFemp$prior)$class)
  #hay que usar las probabilidades a priori estimadas en el modelo sobre los n casos
  #para que concuerde con las estimaciones con CV=TRUE
}
#Comprobación de estos cálculos
head(data.frame(clasiJ,prediJack))
```

```
##   clasiJ prediJack
## 1      F         F
## 2      F         F
## 3      F         F
## 4      F         F
## 5      F         F
## 6      F         F
```

```
#...
tail(data.frame(clasiJ,prediJack))
```

```
##     clasiJ prediJack
## 139      M         M
```

```
## 140      M         M
## 141      M         M
## 142      M         M
## 143      M         M
## 144      M         M
```

```
table(clasiJ,prediJack)
```

```
##       prediJack
## clasiJ  F  M
##      F 45  0
##      M  0 99
```

```
######################################################################
#Ejemplo 4. Problema de regresión lineal múltiple
######################################################################

### Leer los datos
datos=read.table("Renta.txt",header=T)
dim(datos)
```

```
## [1] 118   6
```

```
names(datos)
```

```
## [1] "rentsqm"  "yearc"    "locat"    "bath"     "kitchen"  "cheating"
```

```
#Son datos sobre alquileres de pisos
#Variable objetivo, rentsqm precio del alquiler por m2
#Variables predictoras: año de construcción, calidad de la localización,
#Si tiene baño (1=Sí), Si tien cocina (1=Sí)
#y si tiene calefacción (1=Sí)
summary(datos)
```

```
##     rentsqm           yearc          locat            bath
##  Min.   : 5.146   Min.   :1918   Min.   :1.000   Min.   :0.00000
##  1st Qu.: 8.533   1st Qu.:1939   1st Qu.:1.000   1st Qu.:0.00000
##  Median : 9.344   Median :1959   Median :2.000   Median :0.00000
##  Mean   : 9.396   Mean   :1957   Mean   :1.771   Mean   :0.04237
##  3rd Qu.:10.405   3rd Qu.:1971   3rd Qu.:2.000   3rd Qu.:0.00000
##  Max.   :12.613   Max.   :1995   Max.   :3.000   Max.   :1.00000
##     kitchen          cheating
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.0339   Mean   :0.8983
##  3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

```
datos$locat=factor(datos$locat)
datos$bath=factor(datos$bath)
datos$kitchen=factor(datos$kitchen)
datos$cheating=factor(datos$cheating)
levels(datos$locat)=c("Mala","Regular","Buena")
levels(datos$bath)=levels(datos$kitchen)=levels(datos$cheating)=c("NO","SI")
summary(datos)
```

```
##     rentsqm           yearc          locat    bath     kitchen  cheating
```

```
##  Min.   : 5.146   Min.   :1918   Mala   :47   NO:113   NO:114   NO: 12
##  1st Qu.: 8.533   1st Qu.:1939   Regular:51   SI:  5   SI:  4   SI:106
##  Median : 9.344   Median :1959   Buena  :20
##  Mean   : 9.396   Mean   :1957
##  3rd Qu.:10.405   3rd Qu.:1971
##  Max.   :12.613   Max.   :1995
```

```
### Modelo de regresión lineal múltiple

modelo=lm(rentsqm~.,data=datos)
summary(modelo)
```

```
##
## Call:
## lm(formula = rentsqm ~ ., data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.44223 -0.68077  0.04873  0.68761  1.91992
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -79.205474   8.251974  -9.598 3.02e-16 ***
## yearc          0.045078   0.004249  10.610  < 2e-16 ***
## locatRegular   0.334578   0.188820   1.772  0.07915 .
## locatBuena     0.599265   0.256146   2.340  0.02110 *
## bathSI         1.383707   0.437834   3.160  0.00203 **
## kitchenSI     -0.236235   0.476846  -0.495  0.62129
## cheatingSI     0.098487   0.294107   0.335  0.73836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9245 on 111 degrees of freedom
## Multiple R-squared:  0.5949, Adjusted R-squared:  0.573
## F-statistic: 27.17 on 6 and 111 DF,  p-value: < 2.2e-16
```

```
#Calcular MSE, o sea, error cuadrático medio y RMSE empírico
MSE_emp=mean(residuals(modelo)^2)
RMSE_emp=sqrt(MSE_emp)
MSE_emp
```

```
## [1] 0.803969
```

```
RMSE_emp
```

```
## [1] 0.8966432
```

```
#Estimaciones Jackknife

#Se pueden calcular con cv.lm
#o bien recorriendo los n modelos
#cada uno se construye dejando fuera
#el caso i, donde se aplica el
#modelo para calcular prediJ[i]
n=nrow(datos)
prediJ = numeric(n)
for(i in 1:n){
```

```
  modelo.i = lm(rentsqm~.,data=datos[-i,])
  prediJ[i]<-predict(modelo.i,datos[i,])
}

resi_J=datos$rentsqm - prediJ
MSE_J=mean(resi_J^2)
RMSE_J=sqrt( MSE_J )
R2J=cor(datos$rentsqm ,prediJ)^2
MSE_J
```

```
## [1] 0.9136945
```

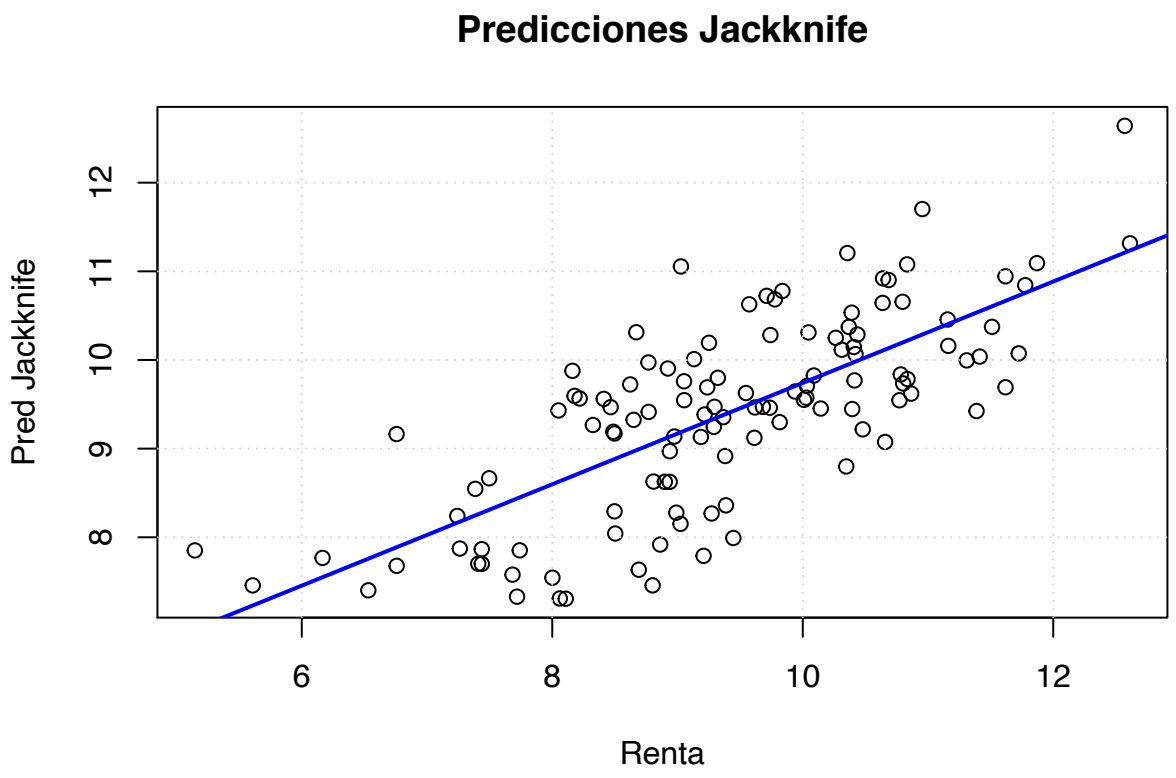
```
RMSE_J
```

```
## [1] 0.9558737
```

```
R2J
```

```
## [1] 0.541302
```

```
plot(datos$rentsqm ,prediJ,xlab="Renta",ylab="Pred Jackknife",
     main="Predicciones Jackknife")
grid()
abline(lsfit(x=datos$rentsqm ,y=prediJ),col="blue",lwd=2)
```



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
```

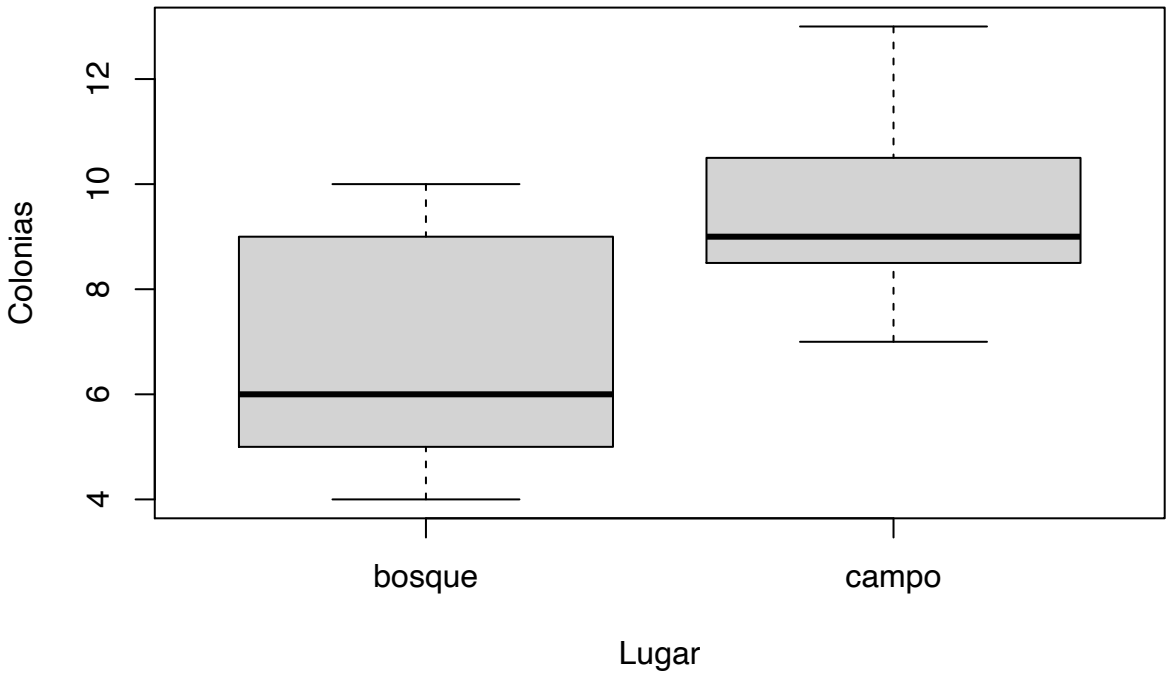
```
##     Lugar Colonias
## 1   campo        9
## 2   campo        9
## 3   campo       10
## 4   campo        9
## 5   campo       11
## 6   campo        7
## 7   campo       11
## 8   campo        8
```

```
## 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)
```

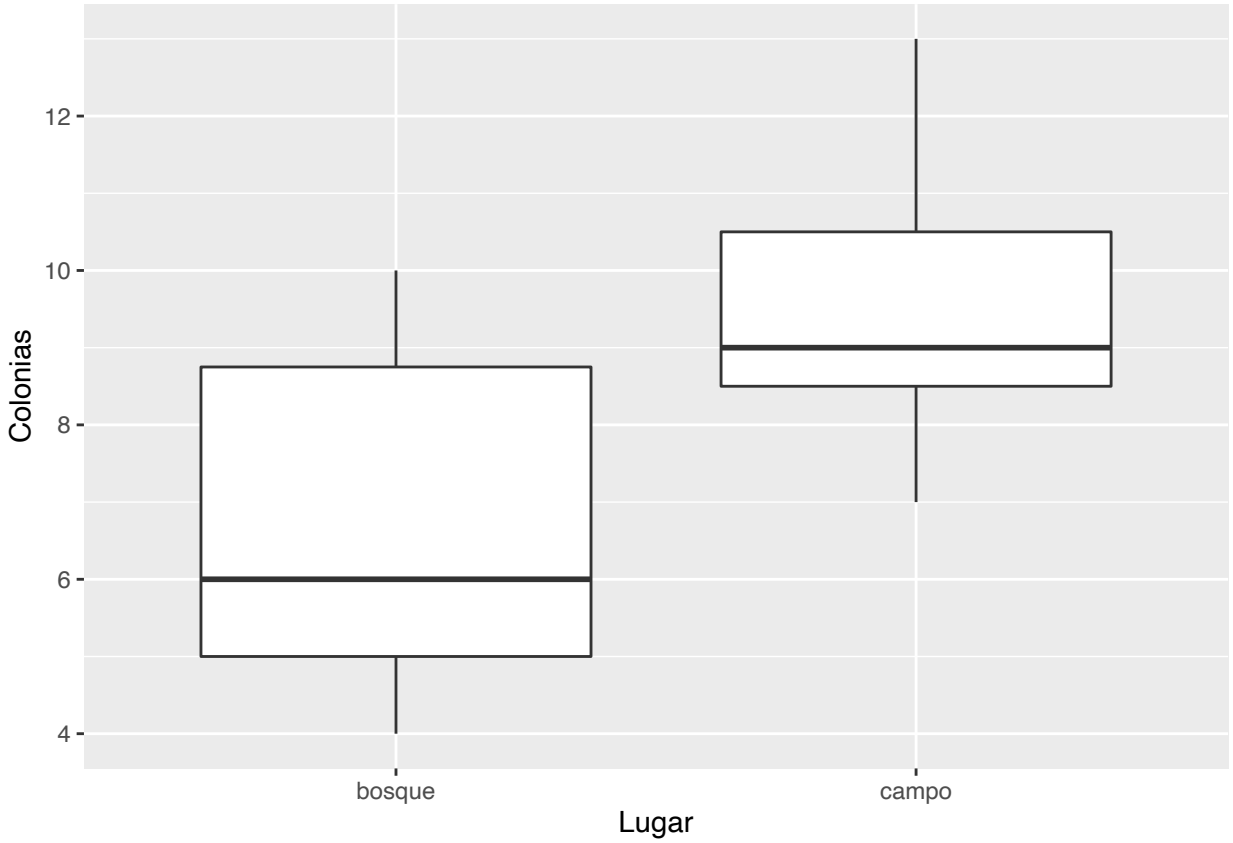
```
##
## bosque  campo
##     14     11
```

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


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

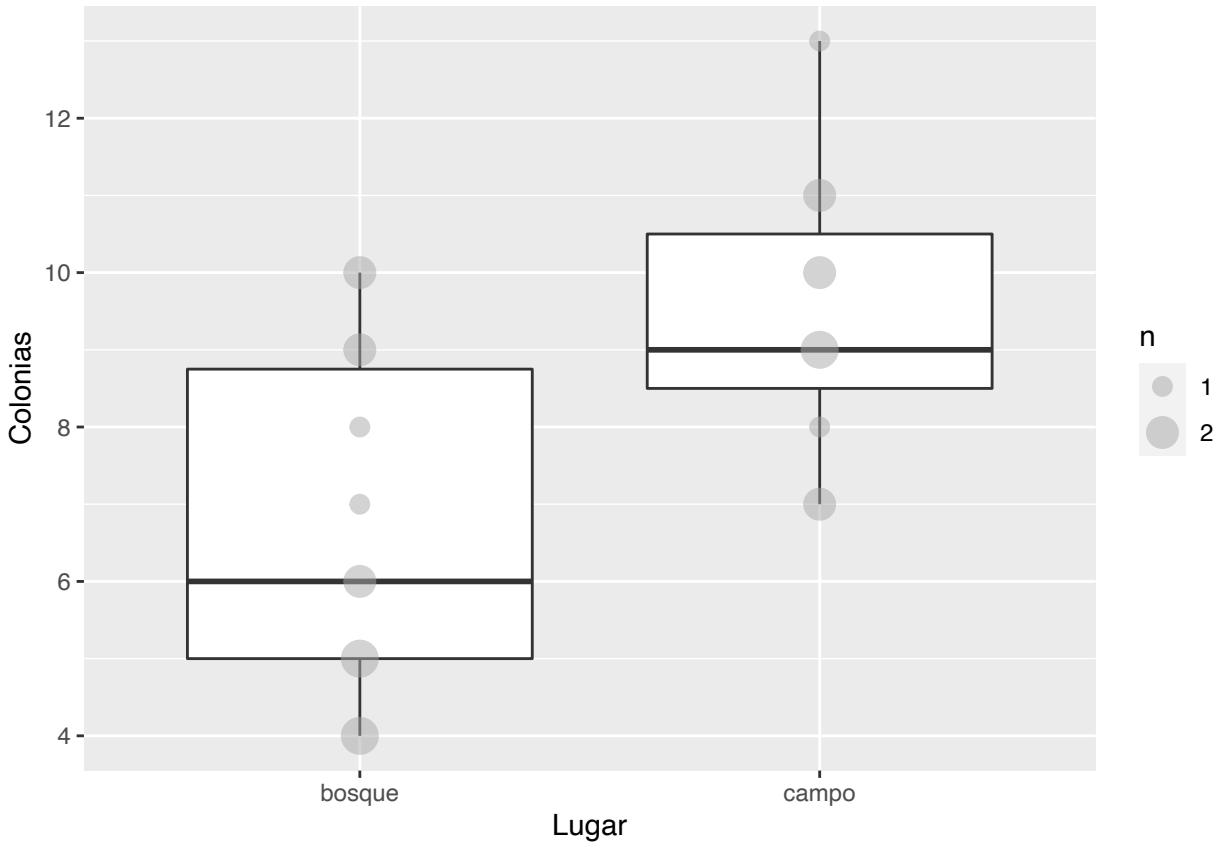



```
#o bien mediante ggplot
library(ggplot2)
(ggplot(hormigas,aes(Lugar,Colonias))
  + geom_boxplot()
```

```
)
```




```
#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
```

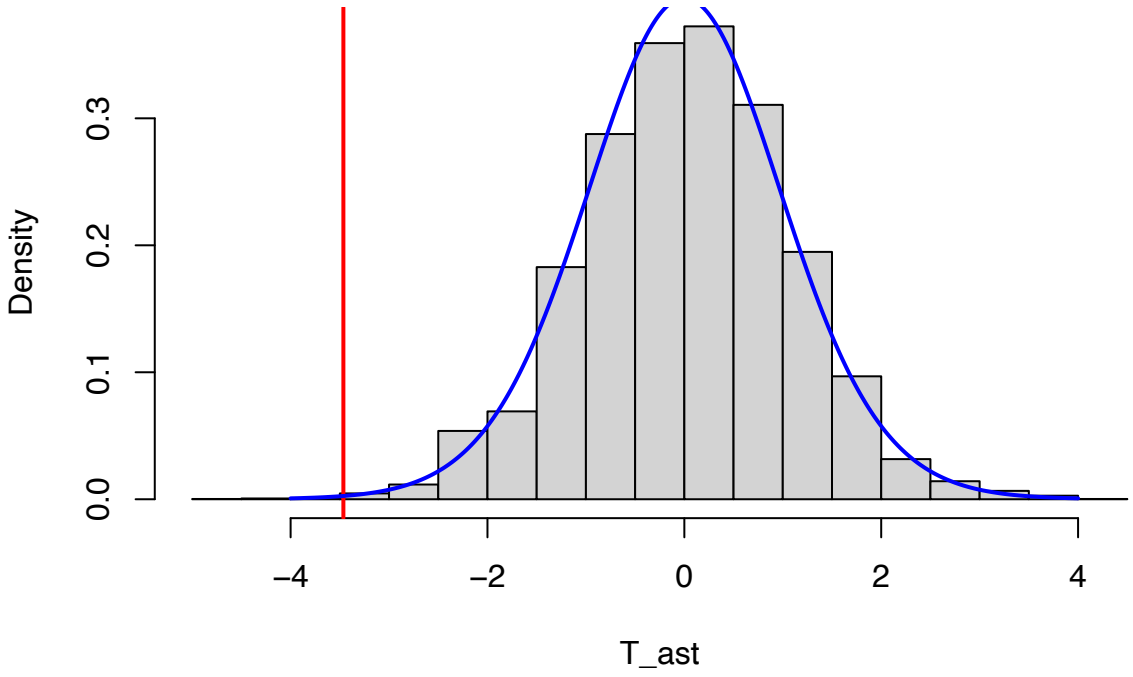
```r
set.seed(101) ##Para que sea reproducible el resultado
n=nrow(hormigas)
B <- 9999
T_ast <- numeric(B) ## Reservar espacio
for (b in 1:B) {
  if (b%%1000 ==0) cat("Perm.", b,"de",B,"\n")
  ## Permutar los casos y guardar en dataperm
  perm <- sample(n)
  dataperm <- transform(hormigas,Colonias=Colonias[perm])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

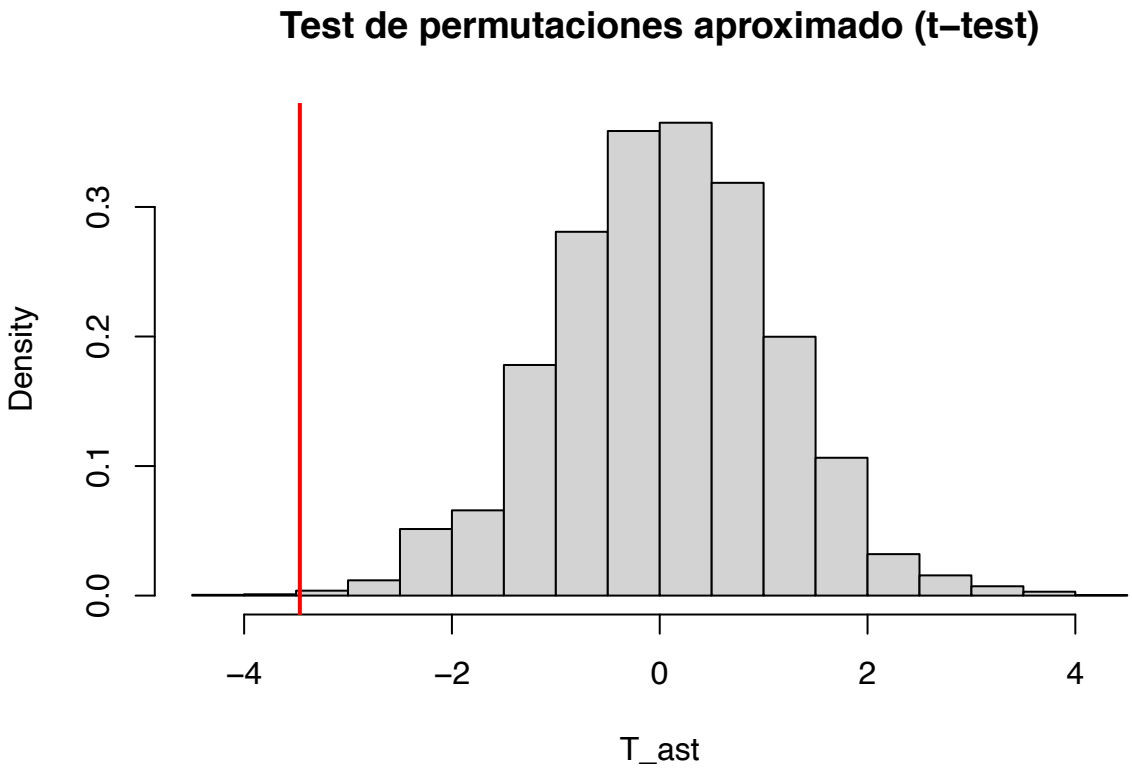
```
##  [6,]    1    4    6   12   14   16   17   18   21    23    25
##  [7,]    4    5    6    8   11   14   16   18   19    21    25
##  [8,]    1    4   11   12   14   15   16   19   21    23    24
##  [9,]    1    4    6   10   12   15   16   18   19    23    25
## [10,]    1    5    7    8   11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

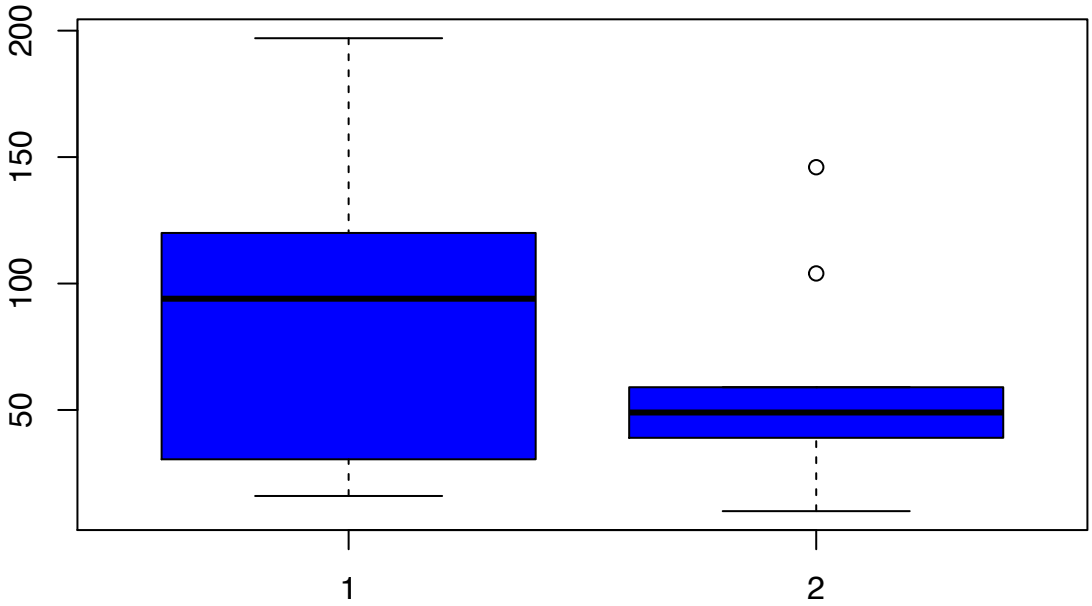
```
#En este caso se pueden generar todas las permutaciones
summary(x)
```

```
##    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
```

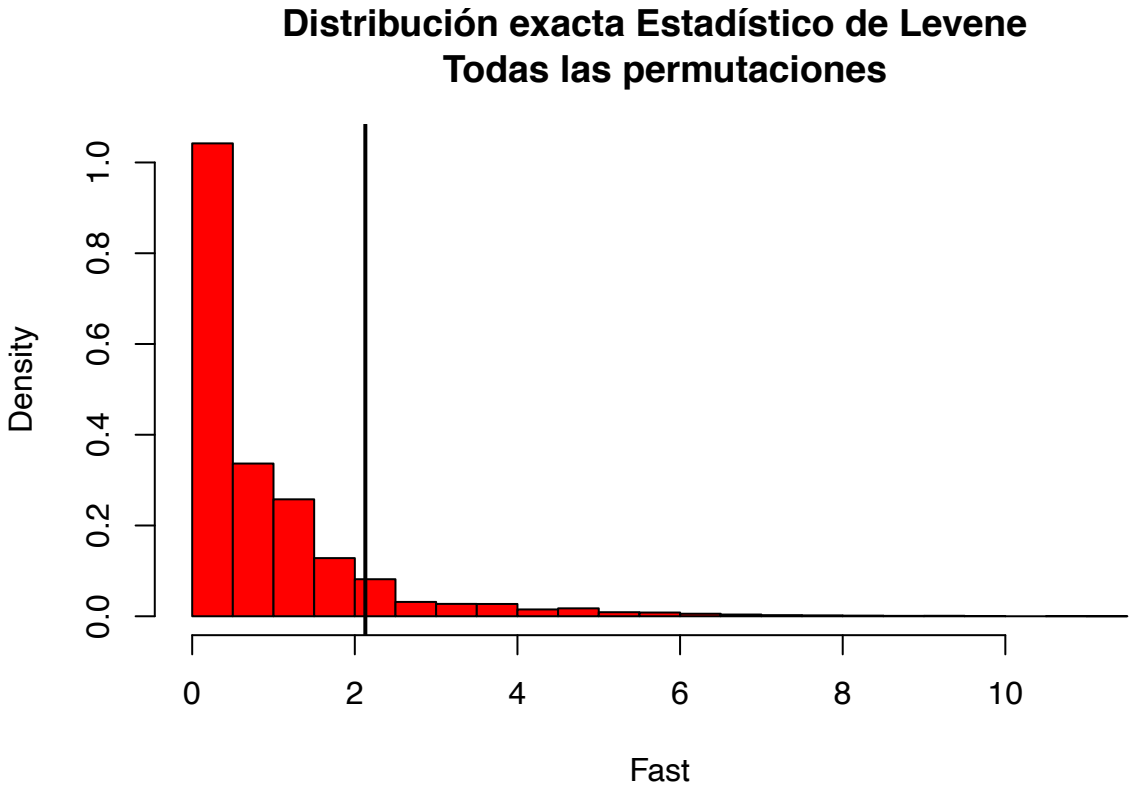
```
F0=leveneTest(X~grupo,data=datos)$`F value`[1]
F0
```

```
## [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
```

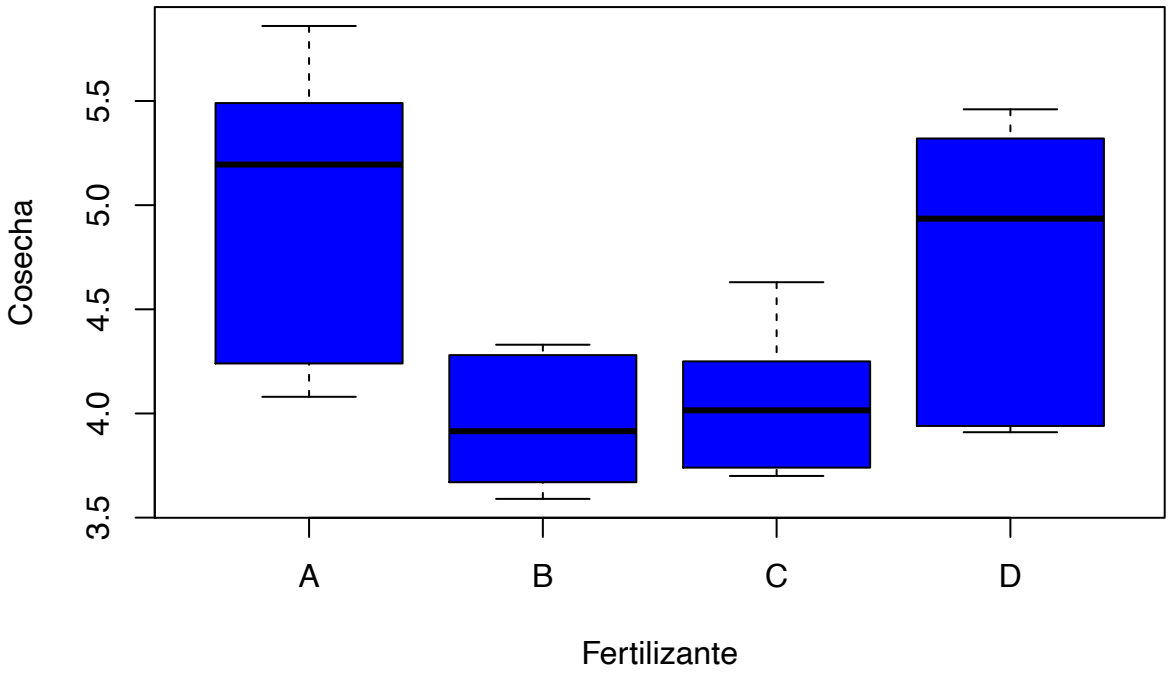
```
##    Fertilizante Cosecha
## 1             A    5.30
## 2             A    4.24
## 3             A    4.08
## 4             A    5.09
## 5             A    5.49
## 6             A    5.86
## 7             B    3.67
## 8             B    4.28
## 9             B    4.08
## 10            B    4.33
## 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)
```

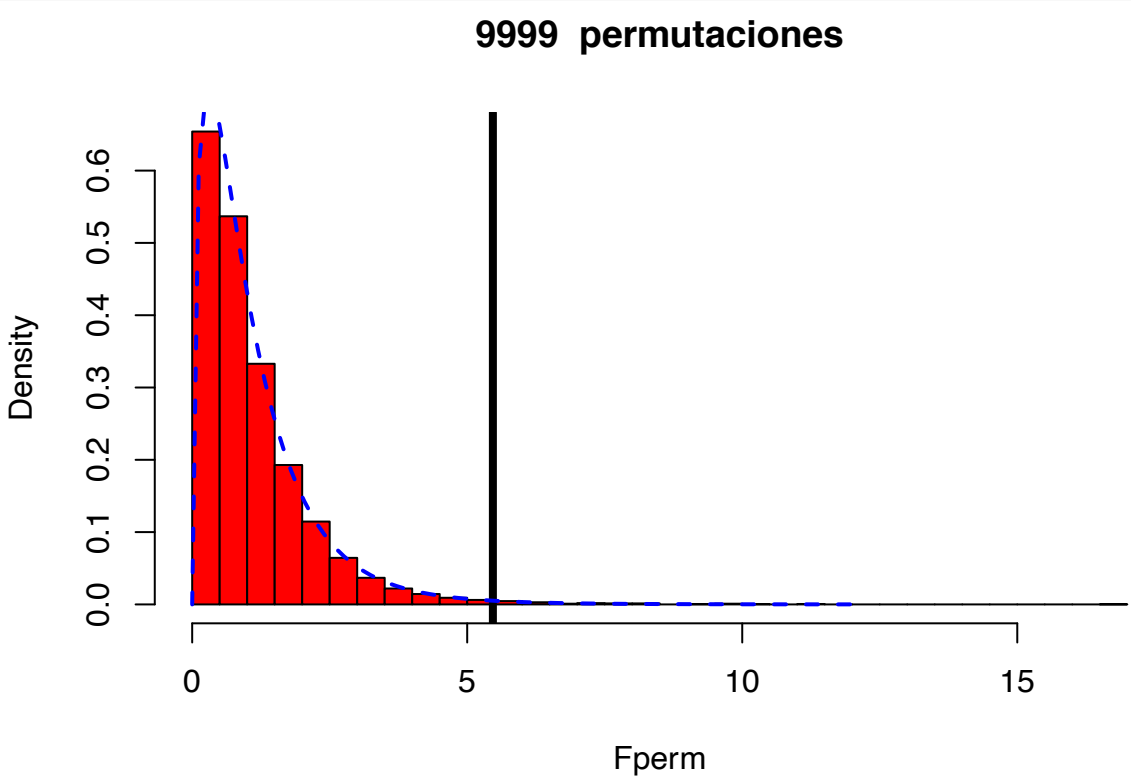
```
  Fperm[b]<-anova(anavast)[1,4]
}
```

```
## 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
```