

Hoja 7 de problemas y prácticas con R

Estadística Computacional I. Grado en Estadística

Departamento de Estadística e Investigación Operativa. Universidad de Sevilla

Ejercicio 1

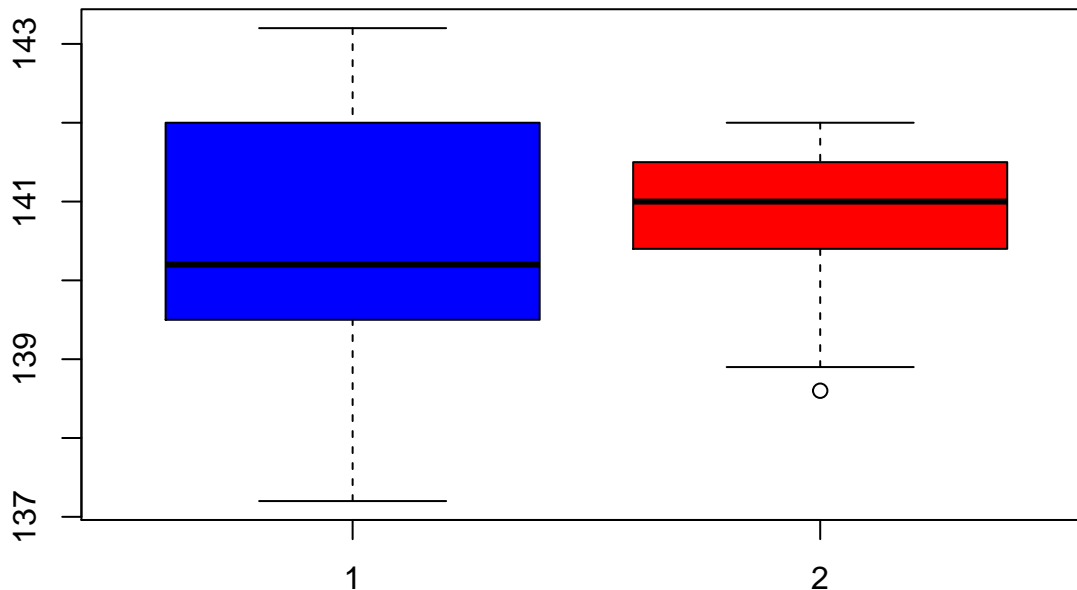
1. Realizar un contraste bootstrap unilateral de hipótesis para comparar las desviaciones típicas a partir de las siguientes muestras ($H_1 : \sigma_1 > \sigma_2$). Escribir las instrucciones R sin y con la librería boot.

```
x=c(137.9, 143, 143.2, 140, 140.2, 139.3, 141.4, 140.1, 142, 137.2, 139.5, 142.7, 141.3)
y=c(141.6, 138.9, 140, 141.9, 140.5, 138.6, 141.5, 141.5, 140.7, 141.5, 140.4, 142,141)
```

Solución

Parte 1

```
#1. Contraste bootstrap unilateral
#####
x<-c(137.9, 143, 143.2, 140, 140.2, 139.3, 141.4, 140.1, 142, 137.2,
     139.5, 142.7, 141.3)
y<-c(141.6, 138.9, 140, 141.9, 140.5, 138.6, 141.5, 141.5, 140.7, 141.5,
     140.4, 142,141)
nx<-length(x)
ny<-length(y)
boxplot(x,y,col=c("blue","red"))
```



```
xy<-c(x,y)
nxy<-length(xy)
```

```

B<-1999
set.seed(125)
T0<-sd(x)/sd(y)
T0

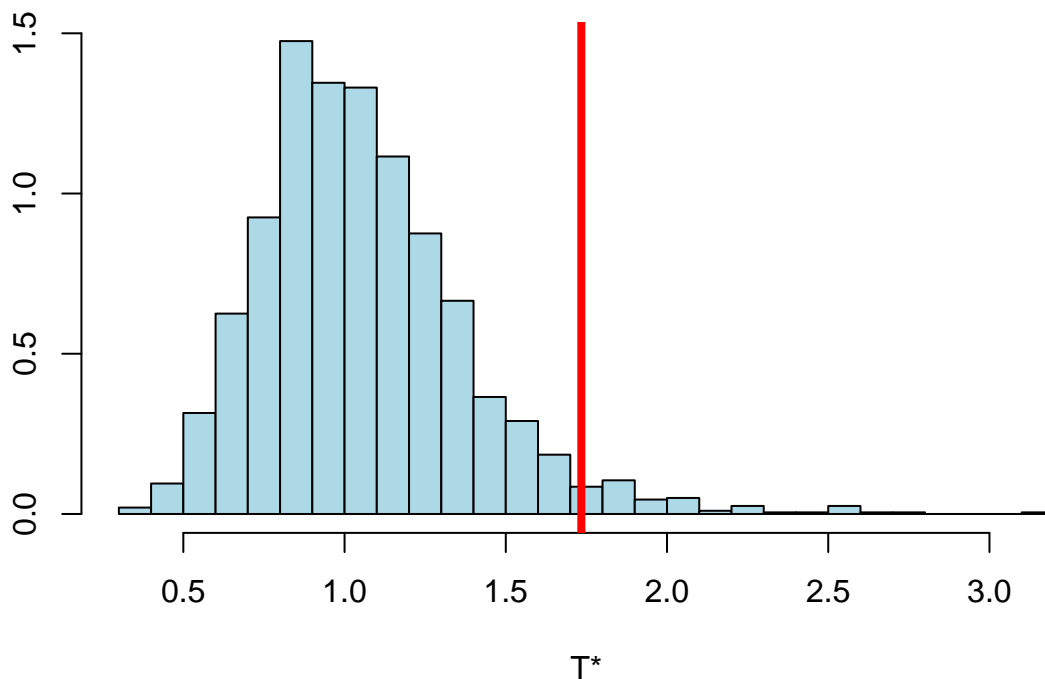
## [1] 1.73431

Tast<-numeric(B)
for (b in 1:B)
  Tast[b]<-sd(sample(xy,nx,replace=TRUE))/sd(sample(xy,ny,replace=TRUE))

hist(Tast,br=30,col="lightblue",prob=TRUE,
     main=paste(B,"muestras bootstrap"),
     xlab="T*",ylab="")
abline(v=T0, lwd=4,col="red")

```

1999 muestras bootstrap



```

cat("p valor bootstrap unilateral (>) = ",(sum(Tast>=T0)+1)/(B+1),"\\n")

```

```

## p valor bootstrap unilateral (>) = 0.0325

```

Parte 2

```

#Para un contraste bilateral se puede trabajar con log(sd(x)/sd(y))
hist(log(Tast),br=30,col="lightblue",prob=TRUE,
     main=paste(B,"muestras bootstrap"),
     xlab="log(T*)",ylab="")
abline(v=log(T0), lwd=4,col="red")
cat("p valor bootstrap unilateral (>) = ",(sum(log(Tast)>=log(T0))+1)/(B+1),"\\n")

```

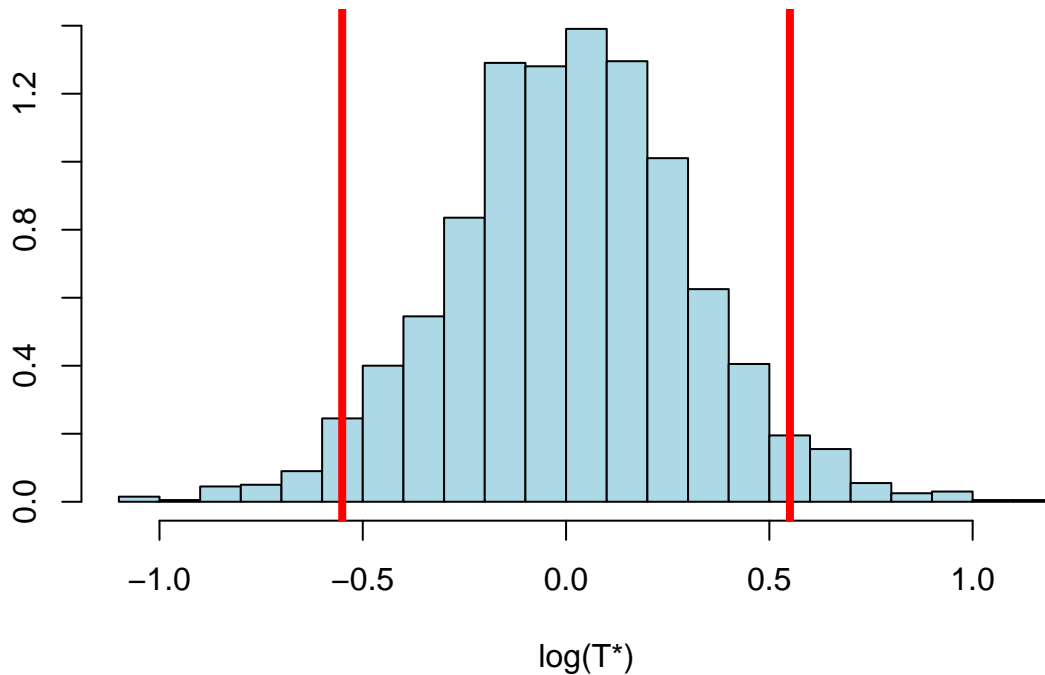
```

## p valor bootstrap unilateral (>) = 0.0325

```

```
abline(v=-log(T0), lwd=4,col="red")
```

1999 muestras bootstrap



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

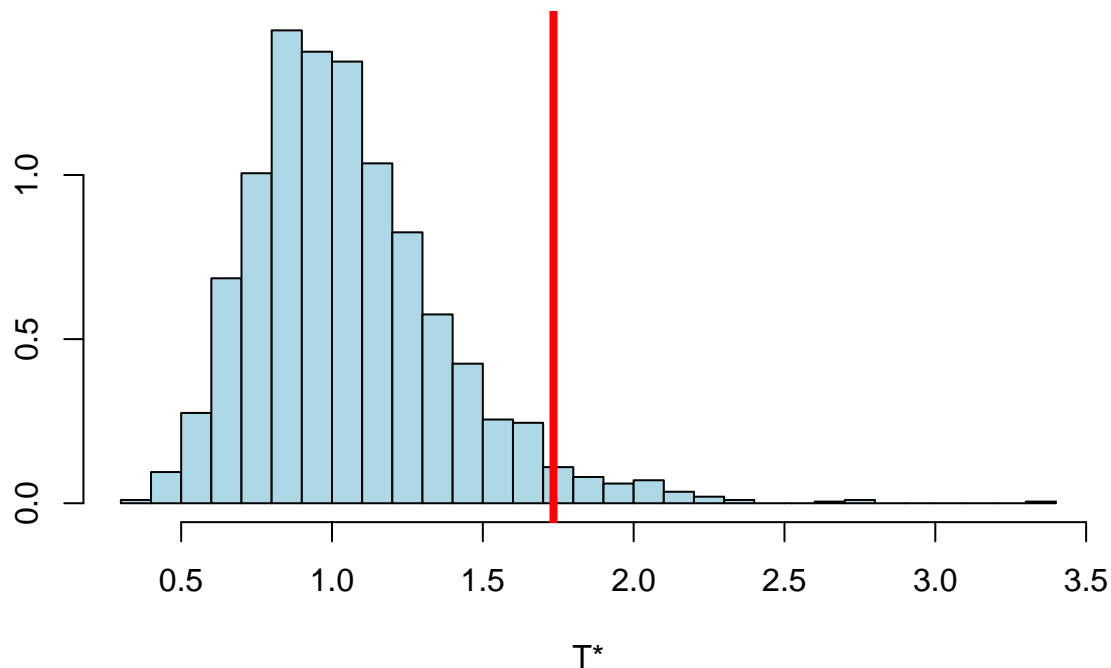
```
## p valor bootstrap bilateral = 0.063
```

Parte 3

```
#EJERCICIO: HACERLO CON LA LIBRERIA boot
library(boot)
cocboot=function(xy,indi,nx)
{
  sd(xy[indi[1:nx]])/sd(xy[indi[-c(1:nx)]])
}

resulboot=boot(xy,cocboot,nx=nx,R=1999)
hist(resulboot$t,br=30,col="lightblue",prob=TRUE,
     main=paste(B,"muestras bootstrap"),
     xlab="T*",ylab="")
abline(v=resulboot$t0, lwd=4,col="red")
```

1999 muestras bootstrap



```
cat("p valor bootstrap unilateral (>) = ",
    (sum(resulboot$t>=resulboot$t0)+1)/(B+1), "\n")
```

```
## p valor bootstrap unilateral (>) = 0.038
```

```
T0
```

```
## [1] 1.73431
```

```
resulboot$t0
```

```
## [1] 1.73431
```

```
summary(Tast)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.3414 0.8360 1.0135 1.0555 1.2233 3.1186
```

```
summary(resulboot$t)
```

```
##      V1
## Min.   :0.3874
## 1st Qu.:0.8283
## Median :1.0076
## Mean   :1.0558
## 3rd Qu.:1.2324
## Max.   :3.3592
```

```
cat("p valor bootstrap bilateral = ",
    (sum(abs(log(resulboot$t))>=abs(log(resulboot$t0)))+1)/(B+1), "\n")
```

```
## p valor bootstrap bilateral = 0.066
```

Ejercicio 2

- Realizar mediante procedimientos bootstrap el contraste de hipótesis unilateral relativo a la posibilidad de que la primera componente principal del fichero iris de R explique más del 70% de la varianza total. Hacerlo directamente y con la ayuda de **boot**.

Solución

```
#2. ACP:
#####
#Porcentaje de varianza explicada por la primera C.P.
#?Puede aceptarse que es superior al 70%?
#H0:PE1<=70; H1:PE1>70
set.seed(357)
data(iris)
#?iris

ACP<- princomp(iris[,-5],cor=T)
summary(ACP)

## Importance of components:
##               Comp.1    Comp.2    Comp.3    Comp.4
## Standard deviation  1.7083611 0.9560494 0.38308860 0.143926497
## Proportion of Variance 0.7296245 0.2285076 0.03668922 0.005178709
## Cumulative Proportion 0.7296245 0.9581321 0.99482129 1.000000000

ACP$loadings

##
## Loadings:
##               Comp.1 Comp.2 Comp.3 Comp.4
## Sepal.Length  0.521  0.377  0.720  0.261
## Sepal.Width   -0.269  0.923 -0.244 -0.124
## Petal.Length   0.580         -0.142 -0.801
## Petal.Width    0.565         -0.634  0.524
##
##               Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var    0.25   0.25   0.25   0.25
## Cumulative Var    0.25   0.50   0.75   1.00

varcp<- ACP$sdev^2
sum(varcp)

## [1] 4

PE1_0<- 100*varcp[1]/sum(varcp)
dife0<- PE1_0-70
n<-nrow(iris)
indices<- 1:n

B<- 4999
difeBoot<- numeric(B)
for (i in 1:B)
{ if (i%%500==0) cat("Muestra bootstrap ",i,"\n")
  ACPboot<- princomp(iris[sample(indices,rep=T),-5],cor=T)
  varcpboot<- ACPboot$sdev^2
```

```

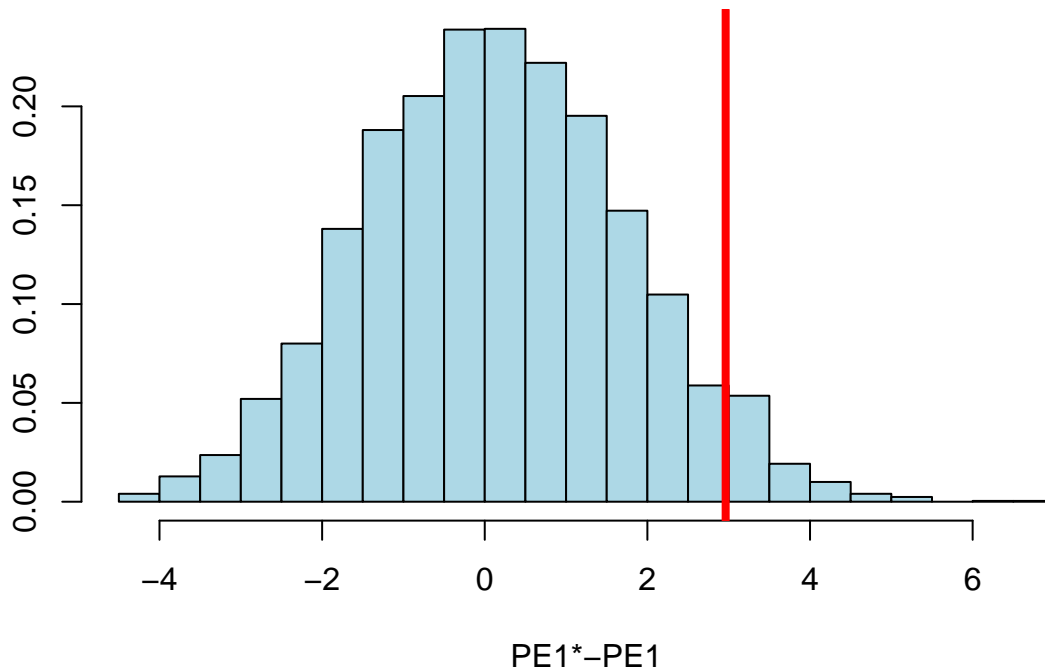
PE1Boot<- 100*varcpboot[1]/sum(varcpboot)
difeBoot[i]<- PE1Boot-PE1_0
}

## Muestra bootstrap 500
## Muestra bootstrap 1000
## Muestra bootstrap 1500
## Muestra bootstrap 2000
## Muestra bootstrap 2500
## Muestra bootstrap 3000
## Muestra bootstrap 3500
## Muestra bootstrap 4000
## Muestra bootstrap 4500

#EJERCICIO: histograma y p-valor
hist(difeBoot,br=30,col="lightblue",prob=TRUE,
     main=paste(B,"muestras bootstrap"),
     xlab="PE1*-PE1",ylab="")
abline(v=dife0, lwd=4,col="red")

```

4999 muestras bootstrap



```

cat("p valor bootstrap unilateral (>) = ",
    (sum(difeBoot>=dife0)+1)/(B+1),"\\n")

## p valor bootstrap unilateral (>) = 0.0472

#EJERCICIO: Con boot
library(boot)
calculoPE1boot=function(datos,indi)
{
  ACPBoot<- princomp(iris[indi,-5],cor=T)
  varcpboot<- ACPBoot$sdev^2

```

```

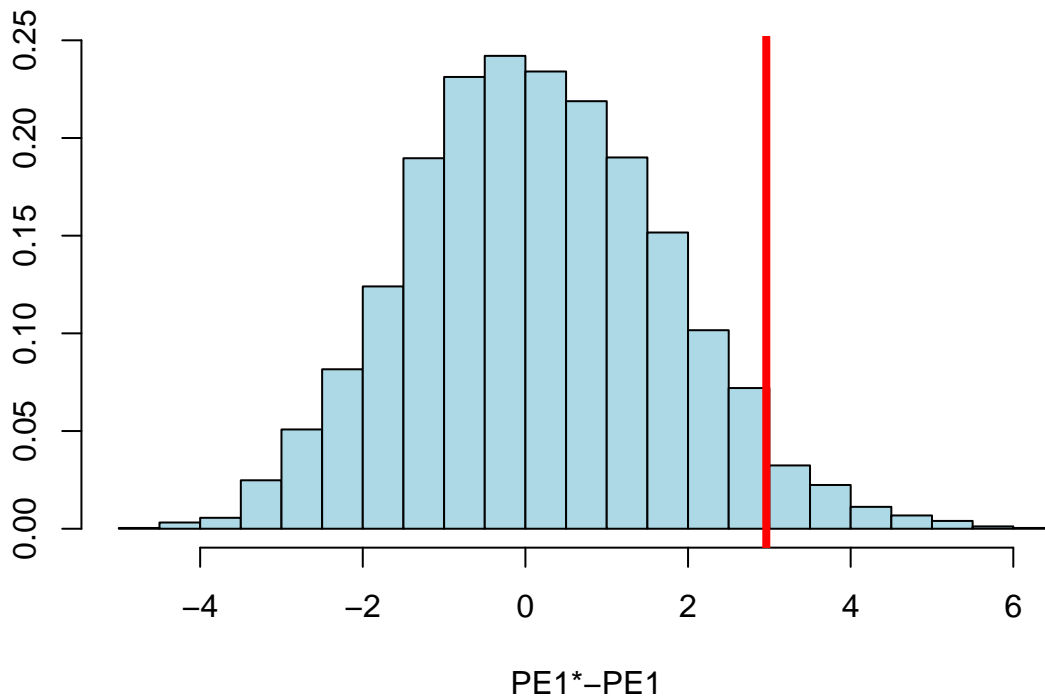
PE1<- 100*varcpboot[1]/sum(varcpboot)
PE1
}

resulboot=boot(iris,calculoPE1boot,R=4999)

hist(resulboot$t-resulboot$t0,br=30,col="lightblue",prob=TRUE,
     main=paste(B,"muestras bootstrap (boot)"),
     xlab="PE1*-PE1",ylab="")
abline(v=resulboot$t0-70, lwd=4,col="red")

```

4999 muestras bootstrap (boot)



```

cat("p valor bootstrap unilateral (>) = ",
    (sum( (resulboot$t-resulboot$t0) >= (resulboot$t0-70))+1)/(B+1),"\n")

```

```
## p valor bootstrap unilateral (>) = 0.0404
```

Ejercicio 3

3. Las siguientes instrucciones permiten definir dos variables en R que contienen las medidas de la corrosión (y) en 13 aleaciones de níquel-cobre, cada una de ellas con un contenido de hierro x . Es de interés el cambio en la corrosión cuando aumenta el nivel de hierro, comparado con la pérdida de corrosión cuando no hay hierro: β_1/β_0 en el modelo de regresión lineal. Calcular intervalos de confianza bootstrap para dicho cociente.

```
x<- c(0.01,0.48,0.71,0.95,1.19,0.01,0.48,1.44,0.71,1.96,0.01,1.44,1.96)
y<- c(127.6,124,110.8,103.9,101.5,130.01,122,92.3,113.1,83.7,128,91.4,86.2)
```

Se sabe que un estimador de la varianza de β_1/β_0 mediante el método delta es:

$$\left(\frac{\hat{\beta}_1}{\hat{\beta}_0}\right)^2 \left(\frac{\hat{v}(\hat{\beta}_1)}{\hat{\beta}_1^2} + \frac{\hat{v}(\hat{\beta}_0)}{\hat{\beta}_0^2} - \frac{2 \operatorname{cov}(\hat{\beta}_0, \hat{\beta}_1)}{\hat{\beta}_0 \hat{\beta}_1} \right)$$

La función utilizada para calcular el estadístico de interés debe incluir también esta estimación de su varianza, para usarla en el método *bootstrap-t*.

Solución

```
x<- c(0.01,0.48,0.71,0.95,1.19,0.01,0.48,1.44,0.71,1.96,0.01,1.44,1.96)
y<- c(127.6,124,110.8,103.9,101.5,130.01,122,92.3,113.1,83.7,128,91.4,86.2)
cor(x,y)
```

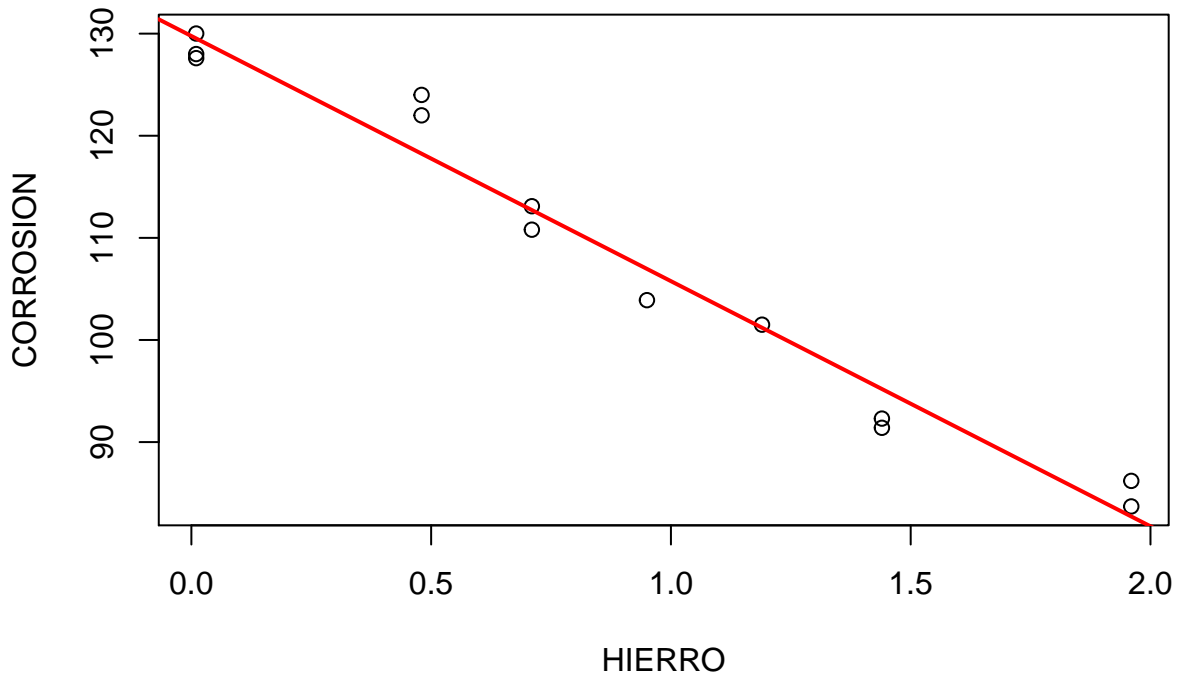
```
## [1] -0.9847401
```

```
plot(x,y,xlab="HIERRO",ylab="CORROSION",main="")
regrelin<- lm(y~x)
summary(regrelin)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.7987 -1.9277  0.2997  0.9845  5.7552
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  129.768      1.402   92.55  < 2e-16 ***
## x           -24.006      1.279  -18.77 1.06e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.056 on 11 degrees of freedom
## Multiple R-squared:  0.9697, Adjusted R-squared:  0.967
## F-statistic: 352.2 on 1 and 11 DF,  p-value: 1.057e-09
```



```
abline(regrelin,col="red",lwd=2)
```



```
coefici<- coef(regrelin)
print(T0<- coefici[2]/coefici[1])
```

```
##          x
## -0.1849942
```

```
alfa<-0.05
```

```
xydat<- data.frame(x,y)
```

```
library(boot)
```

```
#Para bootstrap de pares
```

```
calcuT<-function(xydat,indi)
```

```
{
```

```
  regre<-lm(y~x,data=xydat[indi,])
```

```
  coefi<- regre$coefficients
```

```
  T<-coefi[2]/coefi[1]
```

```
  V<-vcov(regre) #cov(beta0,beta1)
```

```
  auxi<- (V[1,1]/(coefi[1]^2)) + (V[2,2]/(coefi[2]^2))-
```

```
  2*V[1,2]/(coefi[1]*coefi[2])
```

```
  varT<- (T^2) * auxi #var(T), método delta para bootstrap stud (t)
```

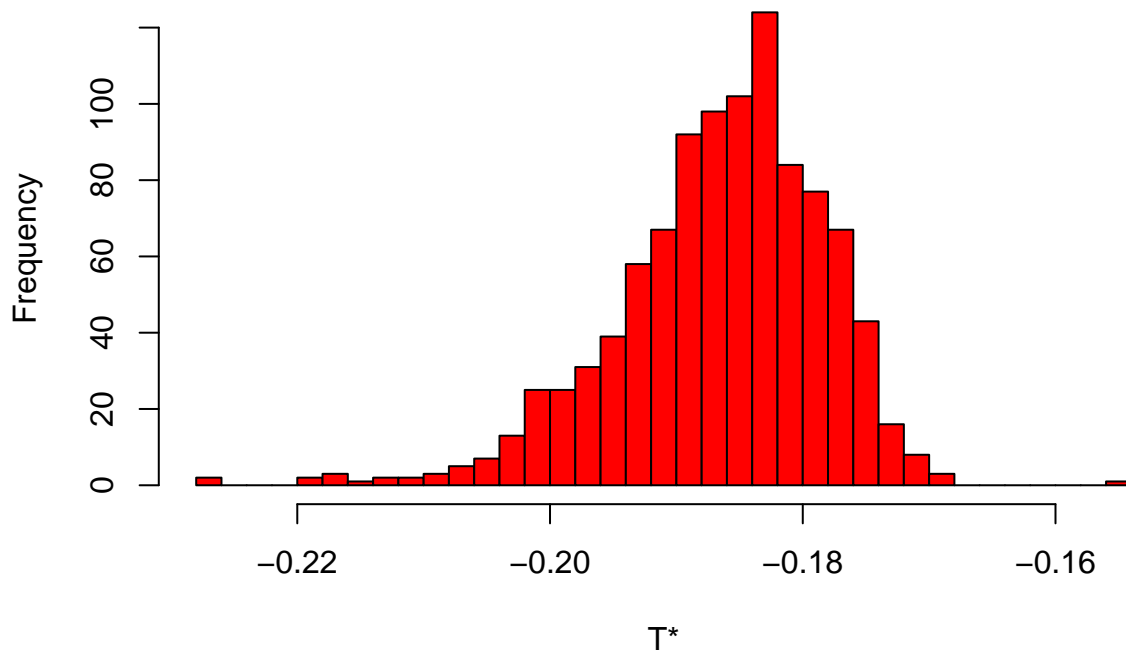
```
  c(T,varT)
```

```
}
```

```
Tbootpares<-boot(xydat,calcuT,1000) #Datos, estadístico, B
```

```
hist(Tbootpares$t[,1],br=30,main="Bootstrap de pares", col="red",xlab="T*")
```

Bootstrap de pares



```
boot.ci(Tbootpares, conf=1-alfa, type=c("norm","perc","stud","bca"),
        var.t = Tbootpares$t[,2])
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
## Based on 1000 bootstrap replicates
```

```
##
```

```
## CALL :
```

```
## boot.ci(boot.out = Tbootpares, conf = 1 - alfa, type = c("norm",
```

```
##      "perc", "stud", "bca"), var.t = Tbootpares$t[, 2])
```

```
##
```

```
## Intervals :
```

```
## Level      Normal      Studentized
```

```
## 95%   (-0.1996, -0.1673 )   (-0.2020, -0.1658 )
```

```
##
```

```
## Level      Percentile      BCa
```

```
## 95%   (-0.2048, -0.1737 )   (-0.2012, -0.1718 )
```

```
## Calculations and Intervals on Original Scale
```

```
#Bootstrap de residuos
```

```
calcuTres<-function(xydat,indi)
```

```
{ xydat$y<- predict(regrelin)+residuals(regrelin)[indi]
```

```
  regre<-lm(y~x,data=xydat)
```

```
  coefi<- regre$coefficients
```

```
  T<-coefi[2]/coefi[1]
```

```
  V<-vcov(regre)
```

```
  auxi<- (V[1,1]/(coefi[1]^2)) + (V[2,2]/(coefi[2]^2))-
```

```
  2*V[1,2]/(coefi[1]*coefi[2])
```

```
  varT<- (T^2) * auxi      #var(T), m?todo delta para bootstrap stud (t)
```

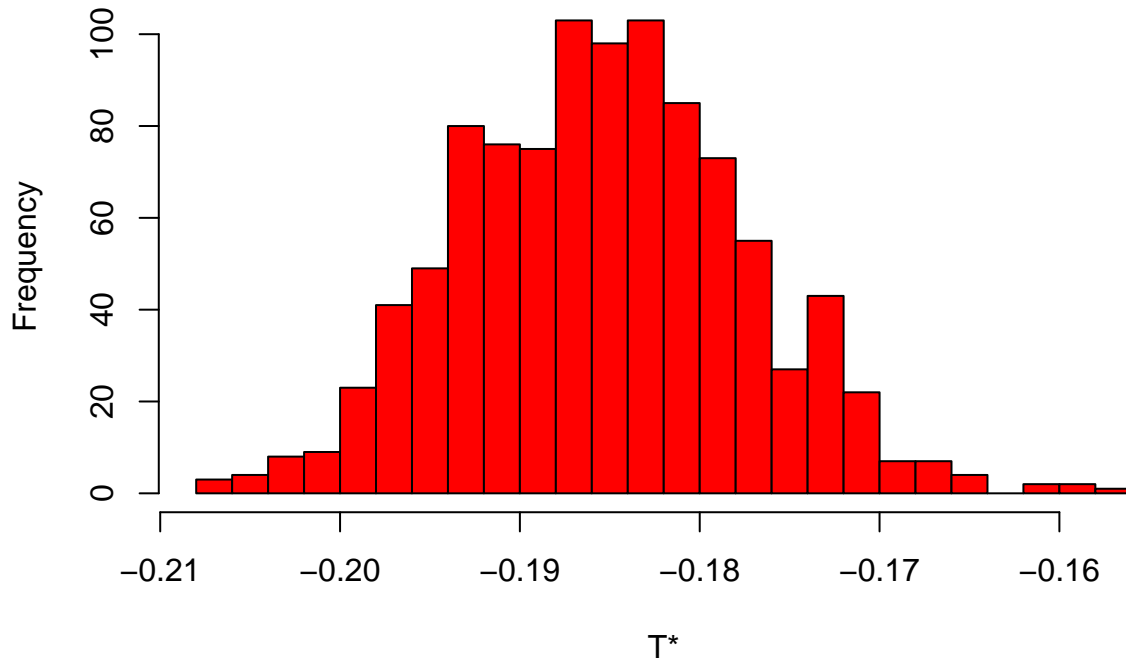
```
  c(T,varT)
```

```
}
```

```
Tbootresi<-boot(xydat,calcuTres,1000)
```

```
hist(Tbootresi$t[,1],br=30,main="Bootstrap de residuos", col="red",xlab="T*")
```

Bootstrap de residuos



```
boot.ci(Tbootresi, conf=1-alfa, type=c("norm","perc","stud","bca"),
        var.t = Tbootresi$t[,2])
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Tbootresi, conf = 1 - alfa, type = c("norm",
##      "perc", "stud", "bca"), var.t = Tbootresi$t[, 2])
##
## Intervals :
## Level      Normal          Studentized
## 95%  (-0.2001, -0.1691 )  (-0.2042, -0.1650 )
##
## Level      Percentile      BCa
## 95%  (-0.2000, -0.1701 )  (-0.1999, -0.1700 )
## Calculations and Intervals on Original Scale
```

Ejercicio 4

4. El fichero “salmon.dat” contiene datos anuales sobre una población de salmones. Las variables que aparecen son: R = “recruits” número de salmones que entran, y S = “spawners” número de salmones que están poniendo huevos, que mueren en cuanto lo hacen. Para que la población se estabilice se requiere: $R = S$ (en otro caso, o hay demasiados salmones para los mismos recursos, o bien no se repone la población).

Se desea calcular un I.C. bootstrap para el punto donde $R = S$, trabajando con el modelo de Beverton-Holt:

$$R = \frac{1}{\beta_0 + \beta_1/S}$$

Solución

```
#Se desea calcular un I.C. para el punto donde R=S  
#####  
salmon<- read.table("datos/salmon.dat",header=T)    #acceso a los datos  
salmon
```

```
##      R      S  
## 1    68    56  
## 2    77    62  
## 3   299   445  
## 4   220   279  
## 5   142   138  
## 6   287   428  
## 7   276   319  
## 8   115   102  
## 9    64    51  
## 10  206   289  
## 11  222   351  
## 12  205   282  
## 13  233   310  
## 14  228   266  
## 15  188   256  
## 16  132   144  
## 17  285   447  
## 18  188   186  
## 19  224   389  
## 20  121   113  
## 21  311   412  
## 22  166   176  
## 23  248   313  
## 24  161   162  
## 25  226   368  
## 26   67    54  
## 27  201   214  
## 28  267   429  
## 29  121   115  
## 30  301   407  
## 31  244   265  
## 32  222   301  
## 33  195   234  
## 34  203   229
```

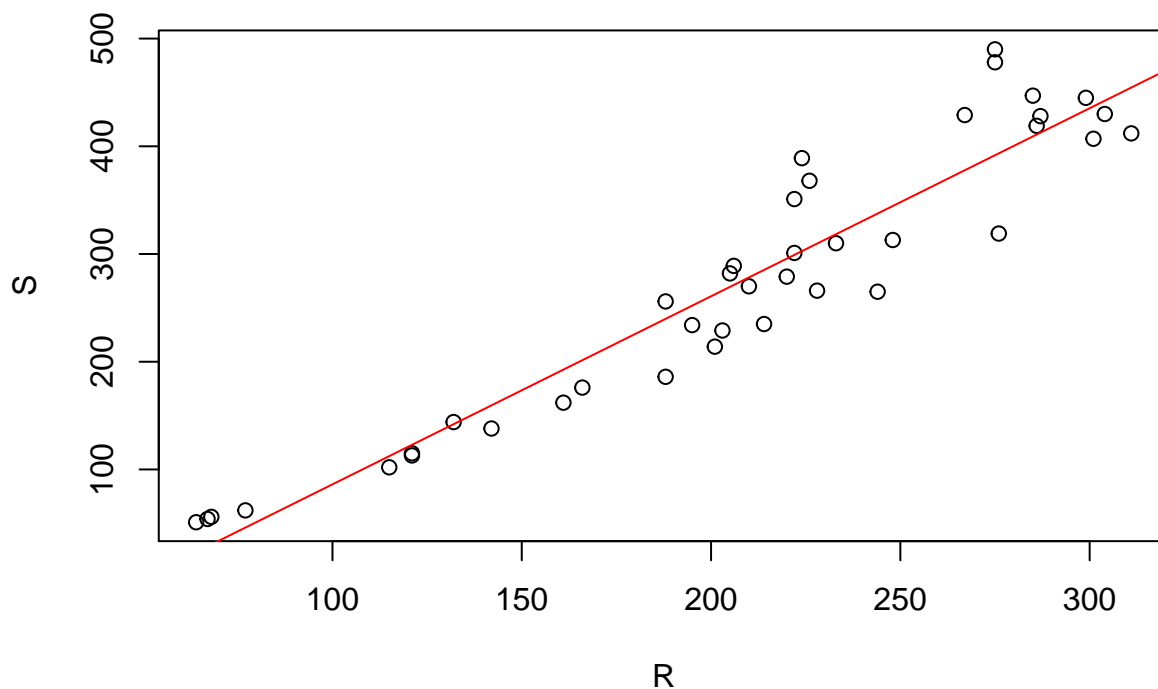
```
## 35 210 270
## 36 275 478
## 37 286 419
## 38 275 490
## 39 304 430
## 40 214 235

attach(salmon)
plot(R,S,main="Datos salmon")
regre1<- lm(S~R)
summary(regre1)

##
## Call:
## lm(formula = S ~ R)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -74.441 -30.233  -8.031   19.681   98.305
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -88.39825    20.90427  -4.229 0.000142 ***
## R              1.74579     0.09576  18.231 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.12 on 38 degrees of freedom
## Multiple R-squared:  0.8974, Adjusted R-squared:  0.8947
## F-statistic: 332.4 on 1 and 38 DF,  p-value: < 2.2e-16

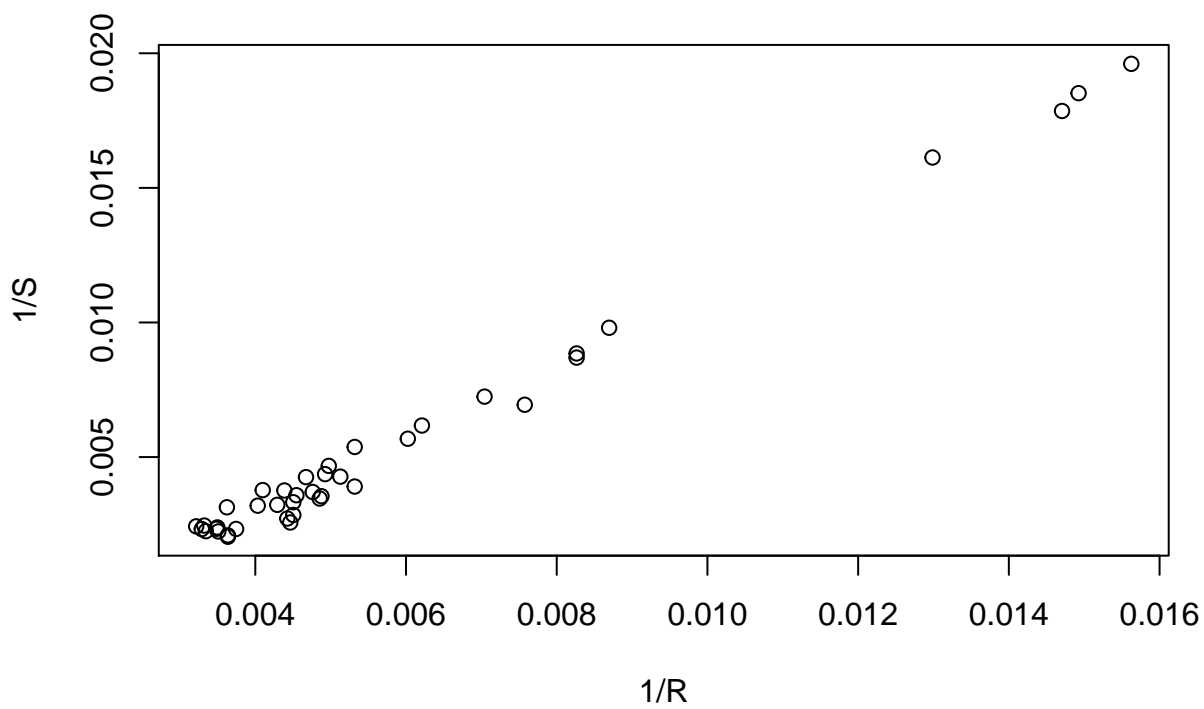
abline(regre1,col="red")
```

Datos salmon



```
plot(1/R, 1/S, main="Datos salmón. Transformac. inversa")
```

Datos salmón. Transformac. inversa



```
rt<- 1/R
st<- 1/S
regre2<- lm(st~rt)    #Modelo de Beverton-Holt
summary(regre2)
```

```
##
## Call:
## lm(formula = st ~ rt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.001e-03 -2.545e-04  5.886e-05  3.120e-04  7.956e-04
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0028000  0.0001556  -18.00  <2e-16 ***
## rt          1.4184178  0.0233598   60.72  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0004793 on 38 degrees of freedom
## Multiple R-squared:  0.9898, Adjusted R-squared:  0.9895
## F-statistic: 3687 on 1 and 38 DF,  p-value: < 2.2e-16

cor(R,1/predict(regre2))^2

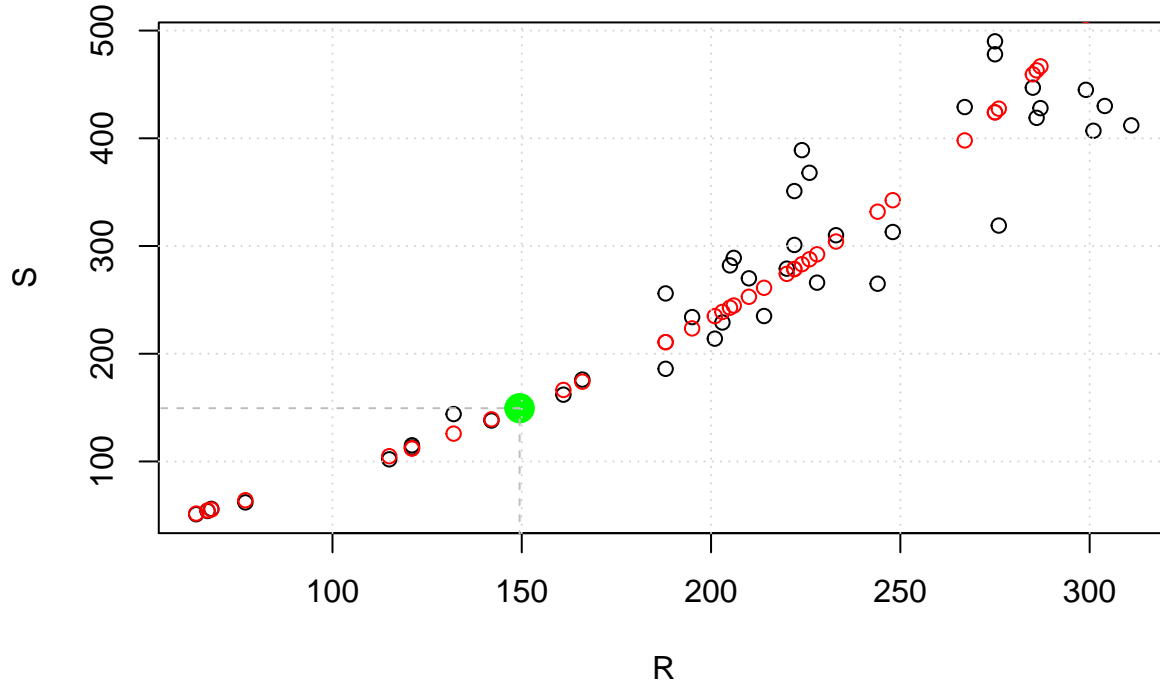
## [1] 0.9426389

plot(R,S,main="Datos Salmon. Modelo de Beverton-Holt")
points(R,1/predict(regre2),col="red")
#Punto de estabilizac., resolviendo en el modelo 1/R=prediccion(R)
coefi<- coef(regre2)
Restab0<- (1-coefi[2])/coefi[1]
c(Restab0,1/predict(regre2,data.frame(rt=1/Restab0)))

##      rt      rt
## 149.4358 149.4358

points(Restab0,Restab0,col="green",lwd=8)
#lines(Restab,Restab,lty=2,type="h",col="grey") Equivale a la siguiente
segments(x0=Restab0, y0=0, x1 = Restab0, y1 = Restab0,lty=2,col="grey")
segments(0, Restab0, Restab0, Restab0,lty=2,col="grey")
grid()
```

Datos Salmon. Modelo de Beverton–Holt



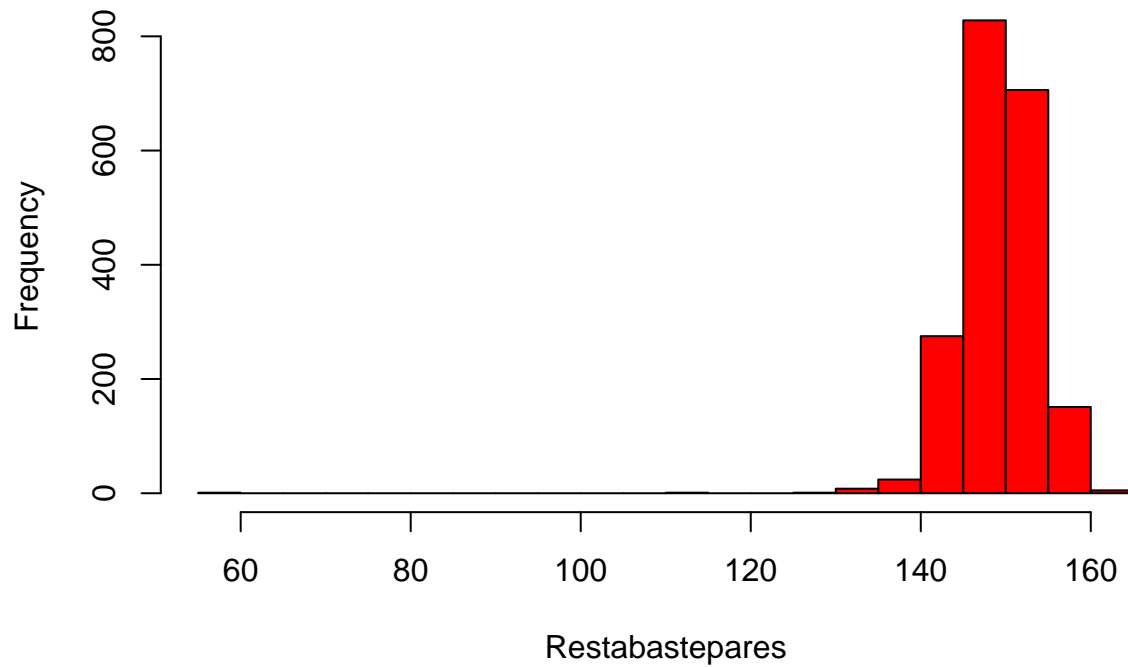
```
# Bootstrap

alfa=0.05
calcuRestabpares<-function(xydat,indi)
{coefi<- lsfit(xydat[indi,1],xydat[indi,2])$coefficients
  (1-coefi[2])/coefi[1]
}
library(boot)
xydat<- data.frame(rt,st)

##bootstrap de pares
bootpares<-boot(xydat,calcuRestabpares,2000) #Datos, estadístico, B
Restabastepares<-c(bootpares$t) #Lo de c() para que no sea matriz

hist(Restabastepares,br=30,main="R estabilizac., B. pares", col="red")
```


R estabilizac., B. pares

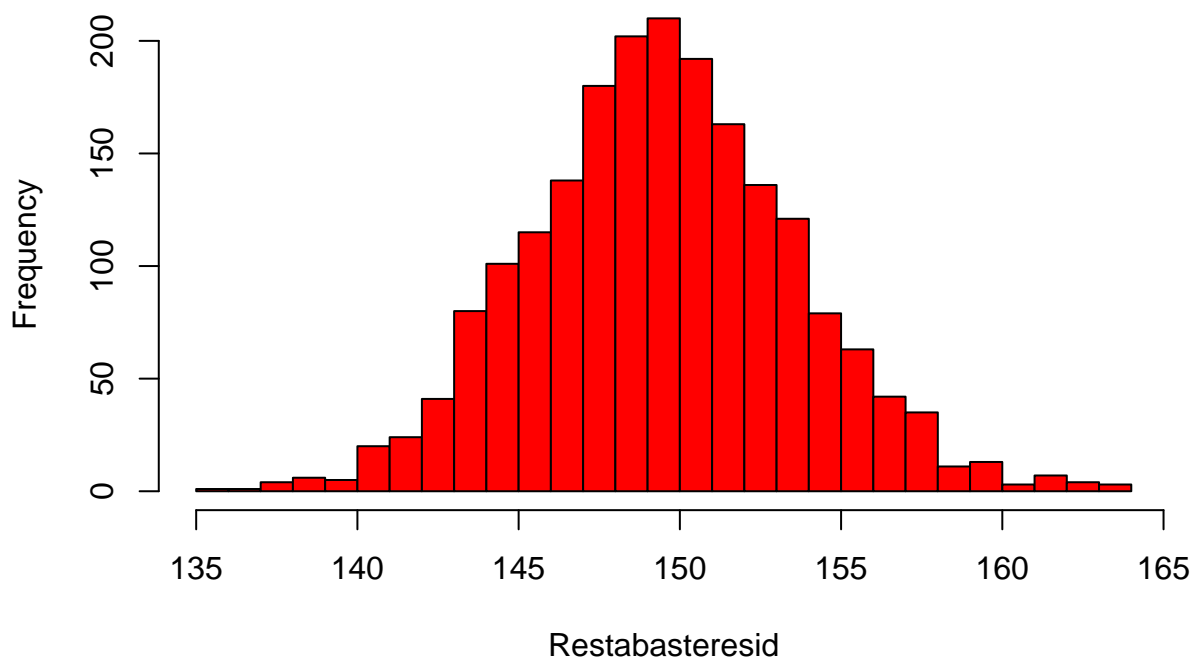


```
ICbootpares<- boot.ci(bootpares, conf=1-alfa,  
                      t=Restabastepares,t0=Restab0,  
                      type=c("norm","perc","bca"))
```

```
##Bootstrap de residuos  
calcuRestabresid<-function(xydat,indi)  
{  
  coefi<-lsfit(xydat[,1],  
               predict(regre2)+(residuals(regre2)[indi]))$coefficients  
  (1-coefi[2])/coefi[1]  
}  
  
bootresi<-boot(xydat,calcuRestabresid,2000)  
Restabasteresid<-c(bootresi$t)
```

```
hist(Restabasteresid,br=30,main="R estabilizac., B. residuos", col="red")
```

R estabilizac., B. residuos



```
ICbootresid<- boot.ci(bootresi,
                      conf=1-alfa, t=Restabasteresid,t0=Restab0,
                      type=c("norm","perc","bca"))
ICbootresid
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootresi, conf = 1 - alfa, type = c("norm",
##      "perc", "bca"), t0 = Restab0, t = Restabasteresid)
##
## Intervals :
## Level      Normal      Percentile      BCa
## 95%  (141.3, 157.5 ) (141.3, 157.5 ) (141.2, 157.4 )
## Calculations and Intervals on Original Scale
```

```
ICbootpares
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootpares, conf = 1 - alfa, type = c("norm",
##      "perc", "bca"), t0 = Restab0, t = Restabastepares)
##
## Intervals :
## Level      Normal      Percentile      BCa
## 95%  (140.5, 159.0 ) (140.7, 157.0 ) (140.7, 157.1 )
## Calculations and Intervals on Original Scale
```

```
#Estimac. corrigiendo el sesgo y estimac. de la varianza
sesgoB<- mean(Restabasteresid)-Restab0
Rcorreg<-Restab0-sesgoB
sd(Restabasteresid)
```

```
## [1] 4.111221
```

```
#IC normal
c(Rcorreg-qnorm(1-alfa/2)*sd(Restabasteresid),
  Rcorreg+qnorm(1-alfa/2)*sd(Restabasteresid))
```

```
##      rt      rt
## 141.3492 157.4649
```

Ejercicio 5

5. Diseñar un estudio empírico para analizar la efectividad de la transformación Z de Fisher en el cálculo de intervalos de confianza para el coeficiente de correlación lineal poblacional. Por ejemplo, generar 100 muestras de tamaño 10 de una Normal bivalente de media 0, desviaciones típicas 1 y correlación 0.6 y calcular los I.C. al 95% con los métodos Percentil, Normal y BCa. Comparar los cubrimientos y las longitudes medias utilizando procedimientos numéricos y gráficos.

Solución

```
#5. Estudio la efectividad de la transformac. Z de Fisher
#####
library(MASS) #para mvrnorm
library(boot)

z.transform <- function(r) {.5*log((1+r)/(1-r))}
z.inversa <- function(z) (exp(2*z)-1)/(exp(2*z)+1)
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

set.seed(13579)
M<- 100 #n?mero de muestras, en la pr?ctica se tomar?a M=10000
n<-10 #tama?o muestral
pho<- 0.6

correl<- numeric(M)
Zcor<- numeric(M)
InterPerc<- matrix(NA,M,2)
InterNorm<- matrix(NA,M,2)
InterBca<- matrix(NA,M,2)
InterPercZ<- matrix(NA,M,2)
InterNormZ<- matrix(NA,M,2)
InterBcaZ<- matrix(NA,M,2)

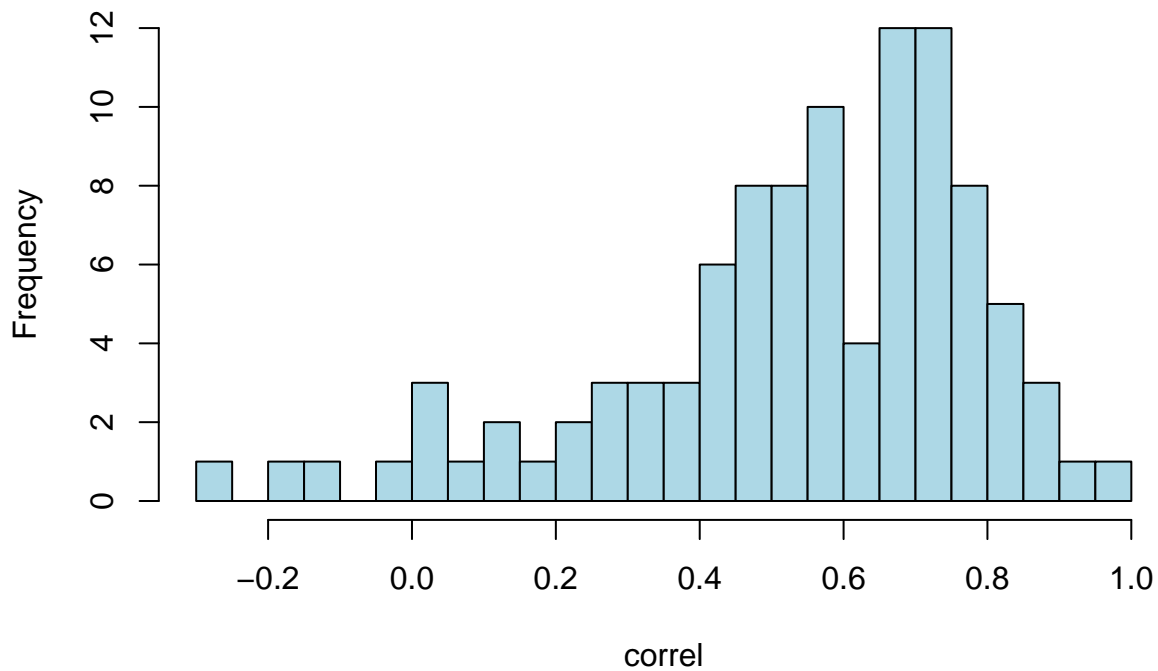
for (i in 1:M)
{ if (i%%10==0) cat("Muestra ",i,"\n")
  xy<-data.frame(mvrnorm(n, c(0,0), matrix(c(1,pho,pho,1),2,2)))
  correl[i]<- cor(xy)[1,2]
  xy.boot1 <- boot(xy, cor.fun, i1=1,i2=2, R=999)
  xy.boot2 <- boot(xy, zcor.fun,i1=1,i2=2, R=999)
  InterPercZ[i,]<-z.inversa(boot.ci(xy.boot2, type="perc")$percent[4:5])
  InterNormZ[i,]<-z.inversa(boot.ci(xy.boot2, type="norm")$normal[2:3])
  InterBcaZ[i,]<-z.inversa(boot.ci(xy.boot2, type="bca")$bca[4:5])
  InterPerc[i,]<-boot.ci(xy.boot1, type="perc")$percent[4:5]
  InterNorm[i,]<-boot.ci(xy.boot2, type="norm")$normal[2:3]
  InterBca[i,]<-boot.ci(xy.boot2, type="bca")$bca[4:5]
}
```

```
## Muestra 10
## Muestra 20
## Muestra 30
## Muestra 40
## Muestra 50
## Muestra 60
## Muestra 70
## Muestra 80
## Muestra 90
## Muestra 100
```

```
Zcorrel<-z.transform(correl)
```

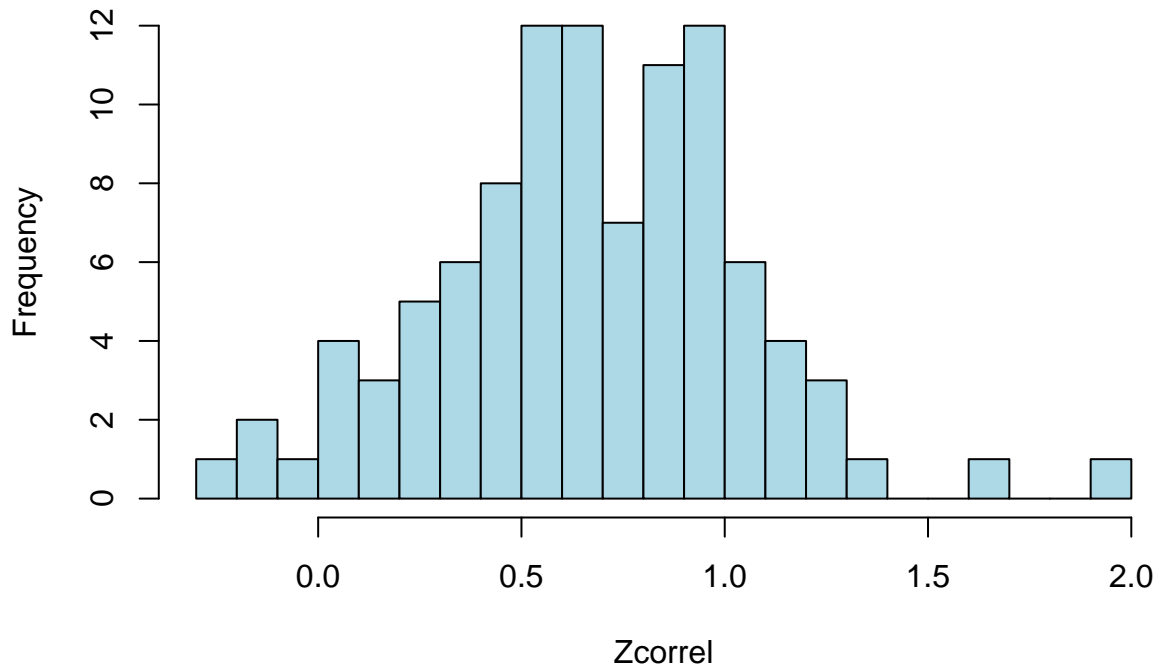
```
hist(correl,br=20,
      main=paste(M,"muestras. Coeficiente de correlac."),col="lightblue")
```

100 muestras. Coeficiente de correlac.



```
hist(Zcorrel,br=20,
      main=paste(M,"muestras. Transf. Z del Coeficiente de correlac."),
      cex.main=0.8,col="lightblue")
```

100 muestras. Transf. Z del Coeficiente de correlac.



```
#Como se sabe que r toma valores en [-1,1]:
corrige<- function(x)
{
  if (x[1]< -1) x[1]= -1
  if (x[2]> 1) x[2]= 1
  x
}
InterPercZ<-t(apply(InterPercZ,1,corrige))
InterNormZ<-t(apply(InterNormZ,1,corrige))
InterBcaZ<-t(apply(InterBcaZ,1,corrige))
InterPerc<-t(apply(InterPerc,1,corrige))
InterNorm<-t(apply(InterNorm,1,corrige))
InterBca<-t(apply(InterBca,1,corrige))

#Función para calcular la cobertura y
#la longitud de un conjunto de intervalos
rendimiento<- function(x,pho)
{
  cubri<-mean(apply(x,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} ))*100
  longi<-mean(x[,2]-x[,1])
  c(cubri,longi)
}
Resultados<-matrix(c(rendimiento(InterPerc,pho),
                      rendimiento(InterNorm,pho),
                      rendimiento(InterBca,pho),
                      rendimiento(InterPercZ,pho),
                      rendimiento(InterNormZ,pho),
                      rendimiento(InterBcaZ,pho)),byrow=T,6,2)

colnames(Resultados)<- c("Cubrimiento","Long.Media")
```

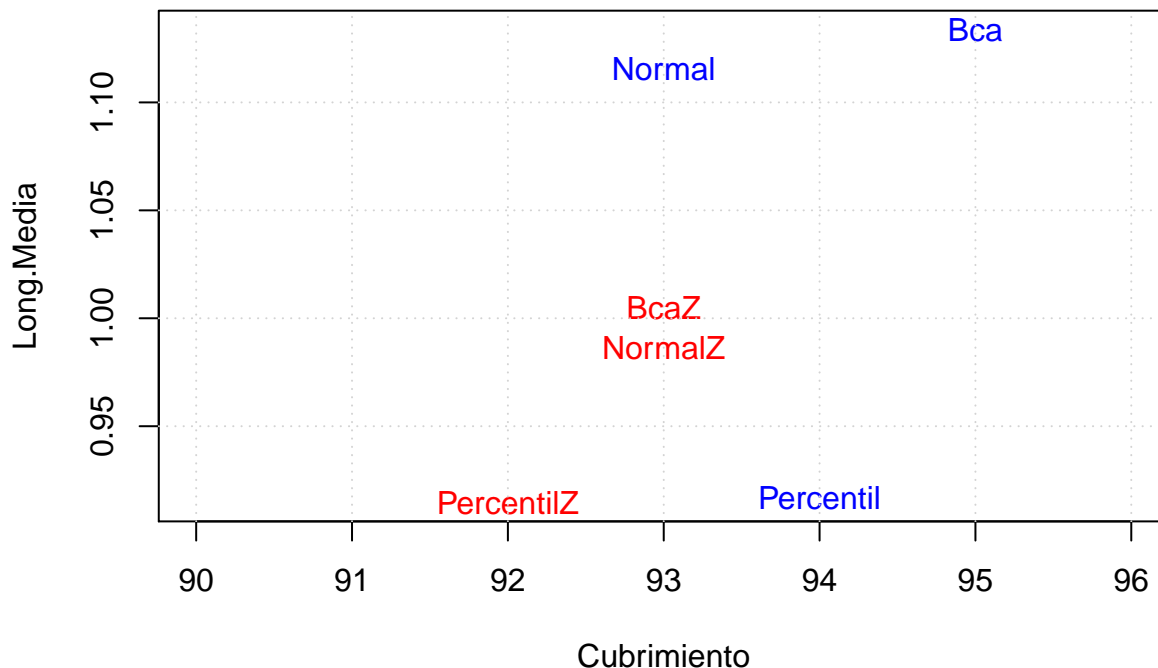
```
rownames(Resultados)<- c("Percentil","Normal","Bca",
                          "PercentilZ","NormalZ","BcaZ")
```

Resultados

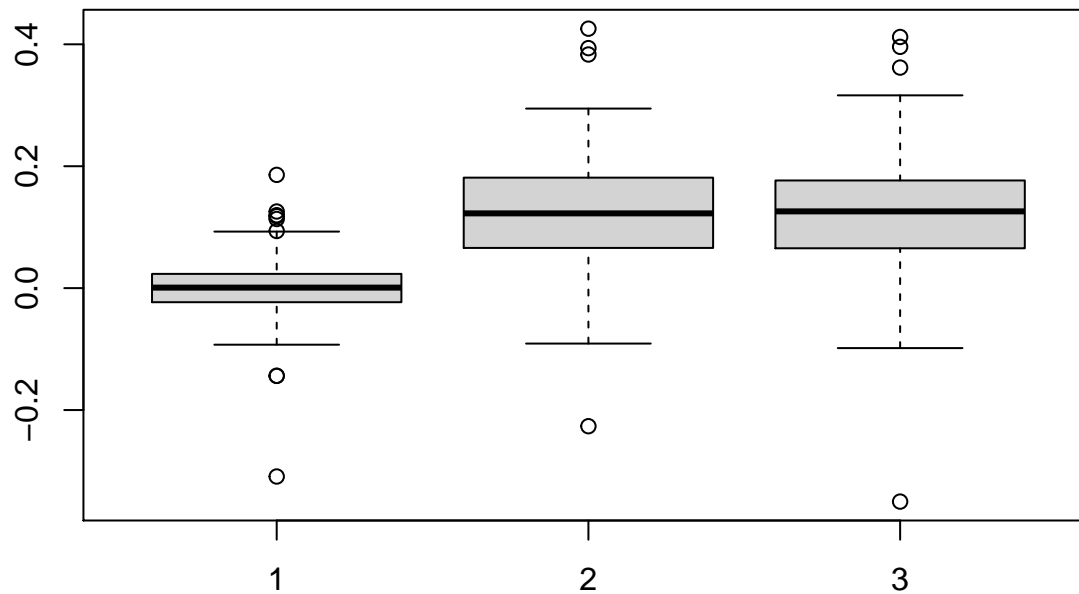
```
##          Cubrimiento Long.Media
## Percentil          94  0.9168274
## Normal            93  1.1156342
## Bca               95  1.1337364
## PercentilZ        92  0.9146747
## NormalZ           93  0.9863495
## BcaZ              93  1.0049938
```

```
plot(Resultados,type="n",
      main="Resultados del experimento",xlim=c(90,96))
text(Resultados,labels=rownames(Resultados),
      col=c(rep("blue",3),rep("red",3)))
grid()
```

Resultados del experimento



```
#Longitudes de los intervalos, y diferencias
#según se aplique o no la transf. Z
longiPerc<- InterPerc[,2]-InterPerc[,1]
longiPercZ<- InterPercZ[,2]-InterPercZ[,1]
difePerc<- longiPerc-longiPercZ
longiNorm<- InterNorm[,2]-InterNorm[,1]
longiNormZ<- InterNormZ[,2]-InterNormZ[,1]
difeNorm<- longiNorm-longiNormZ
longiBca<- InterBca[,2]-InterBca[,1]
longiBcaZ<- InterBcaZ[,2]-InterBcaZ[,1]
difeBca<- longiBca-longiBcaZ
boxplot(difePerc,difeNorm,difeBca)
```



*#Para cada una de las M muestras se han aplicado
#diversos m?todos. Las posibles comparaciones
#corresponden a muestras relacionadas*
t.test(difePerc)

```
##
## One Sample t-test
##
## data: difePerc
## t = 0.34905, df = 99, p-value = 0.7278
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
## -0.01008431 0.01438964
## sample estimates:
## mean of x
## 0.002152665
```

t.test(difeNorm)

```
##
## One Sample t-test
##
## data: difeNorm
## t = 13.593, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
## 0.1104123 0.1481570
## sample estimates:
## mean of x
## 0.1292846
```

t.test(difeBca)

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



```
## t = 12.08, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
## 0.1075953 0.1498898
## sample estimates:
## mean of x
## 0.1287425

#Comparar las proporciones
#Aquí habría que aplicar Test de McNemar
#por el mismo motivo de muestras relacionadas
#Primero se construyen tablas cruzando la
#cobertura (T/F) de cada m?todo
CubPerc=apply(InterPerc,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )
CubPercZ=apply(InterPercZ,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )
CubNorm=apply(InterNorm,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )
CubNormZ=apply(InterNormZ,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )
CubBca=apply(InterBca,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )
CubBcaZ=apply(InterBcaZ,1,cubre<-function(a) { pho<=a[2] && pho>= a[1]} )

( TablaPerc=table(CubPerc,CubPercZ) )

##          CubPercZ
## CubPerc FALSE TRUE
##  FALSE      6    0
##  TRUE       2   92

mcnemar.test(TablaPerc)

##
## McNemar's Chi-squared test with continuity correction
##
## data:  TablaPerc
## McNemar's chi-squared = 0.5, df = 1, p-value = 0.4795

#H0:P[T/No Z]=P[T/Z]
( TablaNorm=table(CubNorm,CubNormZ) )

##          CubNormZ
## CubNorm FALSE TRUE
##  FALSE      4    3
##  TRUE       3   90

mcnemar.test(TablaNorm)

##
## McNemar's Chi-squared test
##
## data:  TablaNorm
## McNemar's chi-squared = 0, df = 1, p-value = 1

( TablaBca=table(CubBca,CubBcaZ) )

##          CubBcaZ
## CubBca  FALSE TRUE
##  FALSE      5    0
##  TRUE       2   93
```

```

mcnemar.test(TablaBca)

##
## McNemar's Chi-squared test with continuity correction
##
## data: TablaBca
## McNemar's chi-squared = 0.5, df = 1, p-value = 0.4795
#Si las muestras fueran distintas para los
#diferentes m?todos, muestras independientes:

prop.test(Resultados[c(1,4),1], c(M,M))

##
## 2-sample test for equality of proportions with continuity correction
##
## data: Resultados[c(1, 4), 1] out of c(M, M)
## X-squared = 0.076805, df = 1, p-value = 0.7817
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.06066751 0.10066751
## sample estimates:
## prop 1 prop 2
## 0.94 0.92

prop.test(Resultados[c(2,5),1], c(M,M))

##
## 2-sample test for equality of proportions without continuity
## correction
##
## data: Resultados[c(2, 5), 1] out of c(M, M)
## X-squared = 8.114e-30, df = 1, p-value = 1
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.07072185 0.07072185
## sample estimates:
## prop 1 prop 2
## 0.93 0.93

prop.test(Resultados[c(3,6),1], c(M,M))

##
## 2-sample test for equality of proportions with continuity correction
##
## data: Resultados[c(3, 6), 1] out of c(M, M)
## X-squared = 0.088652, df = 1, p-value = 0.7659
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.0557684 0.0957684
## sample estimates:
## prop 1 prop 2
## 0.95 0.93

#En este caso, si hay avisos,
#usar el test exacto de Fisher
totales<- Resultados[,1]*M/100

```

```

mcnemar.test(matrix(c(totales[1],
                      M-totales[1],
                      totales[4],
                      M-totales[4]),
                    byrow=T,ncol=2))

##
## McNemar's Chi-squared test with continuity correction
##
## data: matrix(c(totales[1], M - totales[1], totales[4], M - totales[4]), byrow = T, ncol = 2)
## McNemar's chi-squared = 73.724, df = 1, p-value < 2.2e-16

fisher.test(matrix(c(totales[1],
                    M-totales[1],
                    totales[4],
                    M-totales[4]),
                  byrow=T,ncol=2),alternative="greater")

##
## Fisher's Exact Test for Count Data
##
## data: matrix(c(totales[1], M - totales[1], totales[4], M - totales[4]), byrow = T, ncol = 2)
## p-value = 0.3914
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.4726768      Inf
## sample estimates:
## odds ratio
##  1.360218

fisher.test(matrix(c(totales[2],
                    M-totales[2],
                    totales[5],
                    M-totales[5]),
                  byrow=T,ncol=2),alternative="greater")

##
## Fisher's Exact Test for Count Data
##
## data: matrix(c(totales[2], M - totales[2], totales[5], M - totales[5]), byrow = T, ncol = 2)
## p-value = 0.6086
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.3446103      Inf
## sample estimates:
## odds ratio
##          1

fisher.test(matrix(c(totales[3],
                    M-totales[3],
                    totales[5],
                    M-totales[5]),
                  byrow=T,ncol=2),alternative="greater")

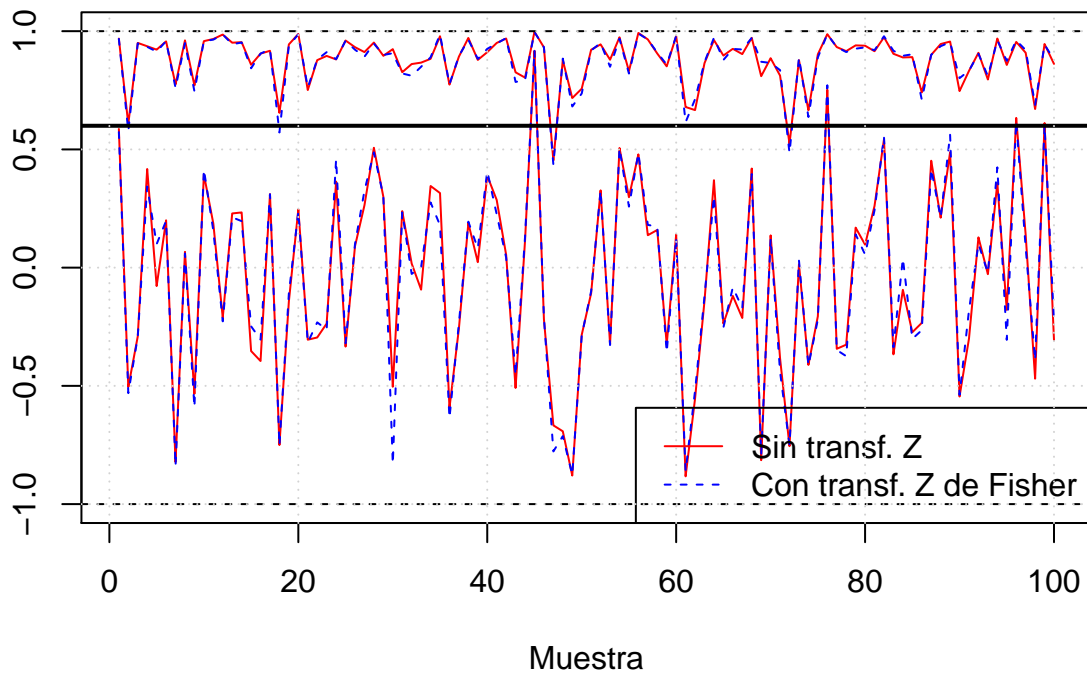
##
## Fisher's Exact Test for Count Data

```

```
##
## data: matrix(c(totales[3], M - totales[3], totales[5], M - totales[5]), byrow = T, ncol = 2)
## p-value = 0.3837
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
## 0.4530872      Inf
## sample estimates:
## odds ratio
## 1.427561
```

```
#Dibujo de los I.C. Percentil
plot(InterPerc[,1],ylim=c(-1,1),type="l",
     main="IC-Bootstrap (Percentil)",col="red",xlab="Muestra", ylab="")
lines(InterPerc[,2],col="red")
lines(InterPercZ[,1],col="blue",lty=2)
lines(InterPercZ[,2],col="blue",lty=2)
legend("bottomright",col=c("red","blue"), lty=1:2,
      legend=c("Sin transf. Z","Con transf. Z de Fisher"))
abline(h=pho,lwd=2)
abline(h=1,lty=2);abline(h=-1,lty=2); grid()
```

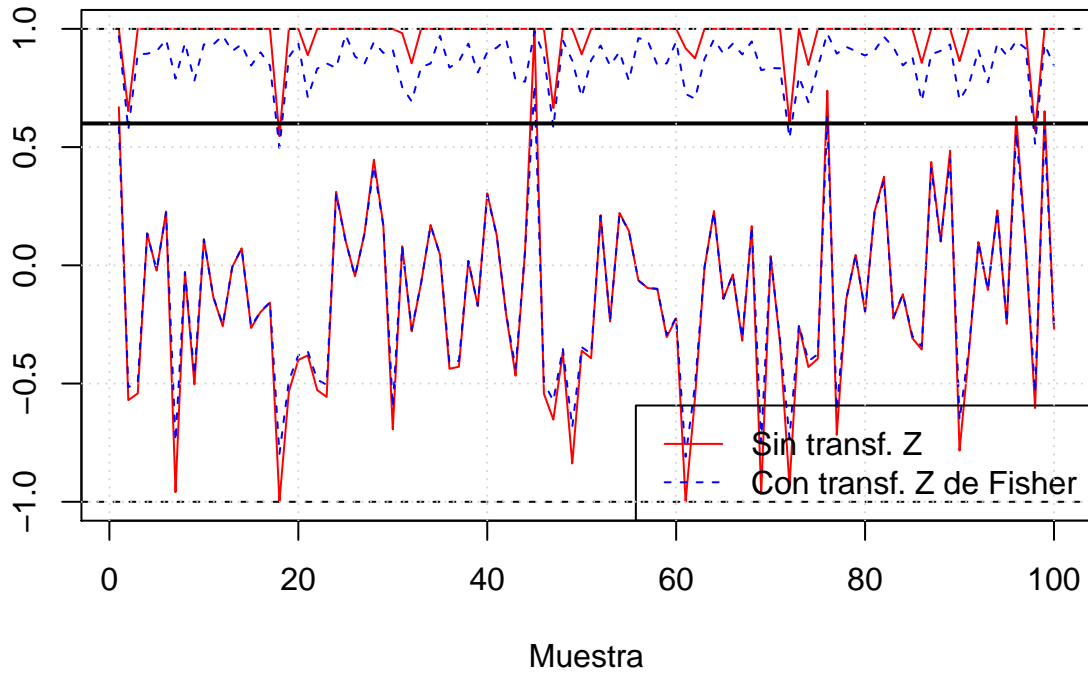
IC-Bootstrap (Percentil)



```
#Dibujo de los I.C. Normal
plot(InterNorm[,1],ylim=c(-1,1),type="l",
     main="IC-Bootstrap (Normal)",
     col="red",xlab="Muestra", ylab="")
lines(InterNorm[,2],col="red")
lines(InterNormZ[,1],col="blue",lty=2)
lines(InterNormZ[,2],col="blue",lty=2)
legend("bottomright",col=c("red","blue"), lty=1:2,
      legend=c("Sin transf. Z","Con transf. Z de Fisher"))
abline(h=pho,lwd=2)
```

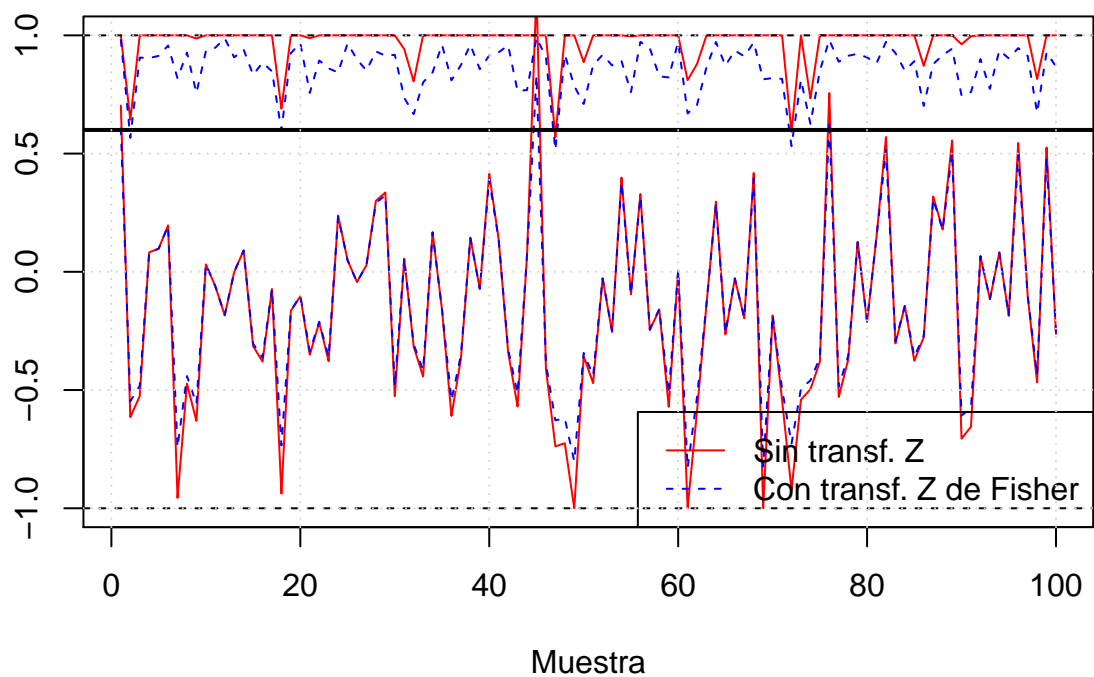
```
abline(h=1,lty=2);abline(h=-1,lty=2); grid()
```

IC-Bootstrap (Normal)



```
#Dibujo de los I.C. BCa
plot(InterBca[,1],ylim=c(-1,1),type="l",
     main="IC-Bootstrap (BCa)",
     col="red",xlab="Muestra", ylab="")
lines(InterBca[,2],col="red")
lines(InterBcaZ[,1],col="blue",lty=2)
lines(InterBcaZ[,2],col="blue",lty=2)
legend("bottomright",col=c("red","blue"), lty=1:2,
      legend=c("Sin transf. Z","Con transf. Z de Fisher"))
abline(h=pfo,lwd=2)
abline(h=1,lty=2);abline(h=-1,lty=2); grid()
```

IC-Bootstrap (BCa)



*#Los basados en la transformac., en general
#tienen menor longitud, y el extremo superior es <1*

Ejercicio 6

6. Estimar el sesgo de la razón (cociente de las medias de las variables x y u) en el fichero “city” de R mediante el bootstrap balanceado (fichero disponible en la librería “boot”):
- Escribiendo directamente las instrucciones.
 - Empleando la librería boot.

Solución

Apartado (a)

```
#6. Bootstrap Balanceado
#####
library(boot)
data(city)
city
```

```
##      u    x
## 1  138 143
## 2   93 104
## 3   61  69
## 4  179 260
## 5   48  75
## 6   37  63
## 7   29  50
## 8   23  48
## 9   30 111
## 10   2  50
```

```
?city
#6.1.
print(R<- mean(city$x)/mean(city$u) )
```

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

```
B<- 999
Raste<- numeric(B)
n<- nrow(city)
lista<- rep(1:n,B)
table(lista)
```

```
## lista
##  1  2  3  4  5  6  7  8  9 10
## 999 999 999 999 999 999 999 999 999 999
```

```
listaper<- matrix(sample(lista),ncol=n,byrow=TRUE)
head(listaper)
```

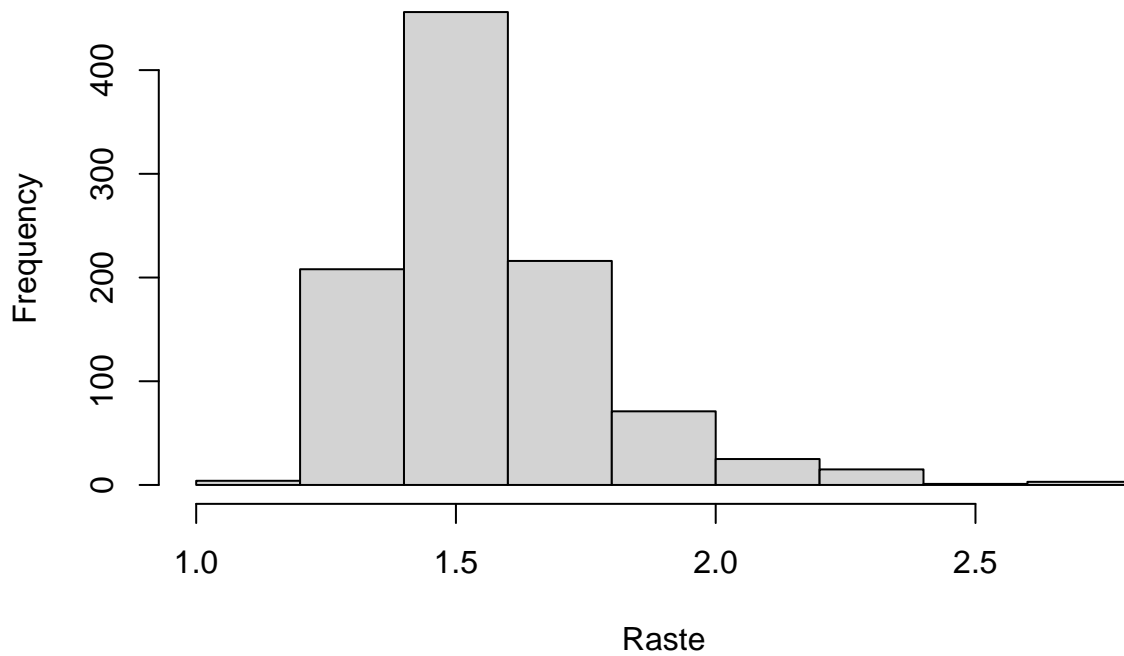
```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    5    4    9   10    5    7    7    8   10    5
## [2,]    1    7    7    4   10    8    3    2    4    9
## [3,]    2    4    8    2    3    2    6    6    8    2
## [4,]    7    5    7   10    2   10    1    8    5    7
## [5,]    3    6    1    2    9    2    1    5    5    1
## [6,]    4    3    3    5    9    1    5    5    4    3
```

```
tail(listaper)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [994,]  5   10  2    2    3    2    2    8    1    3
## [995,]  7    8  2    1    3    1    8    7    3    3
## [996,]  2    3  6    9    3    3    4    6    1    4
## [997,]  9    5  3   10    7    6    2    3    7    5
## [998,]  9    7 10    8    6    5    1    2    7    3
## [999,]  8    7  7    5    1    7    8    9    3    5
```

```
for (b in 1:B)
{indiB<-listaper[b,]
  Raste[b]<- mean(city$x[indiB])/mean(city$u[indiB])
}
hist(Raste)
```

Histogram of Raste



```
mean(Raste)-R
```

```
## [1] 0.03863468
```

Apartado (b)

```
#6.2.Usando boot
ratio <- function(d,indi) mean(d$x[indi])/mean(d$u[indi])
boot.bal<- boot(data=city, statistic=ratio, R = 999, sim="balanced")
mean(boot.bal$t)-boot.bal$t0
```

```
## [1] 0.0384459
```


Ejercicio 7 Transparencia pag 51

7. Estimar el error de clasificación para el modelo de análisis discriminante lineal sobre los datos “cats” de la librería MASS.

Solución

#7. ESTIMACION DEL ERROR DE CLASIFICACION

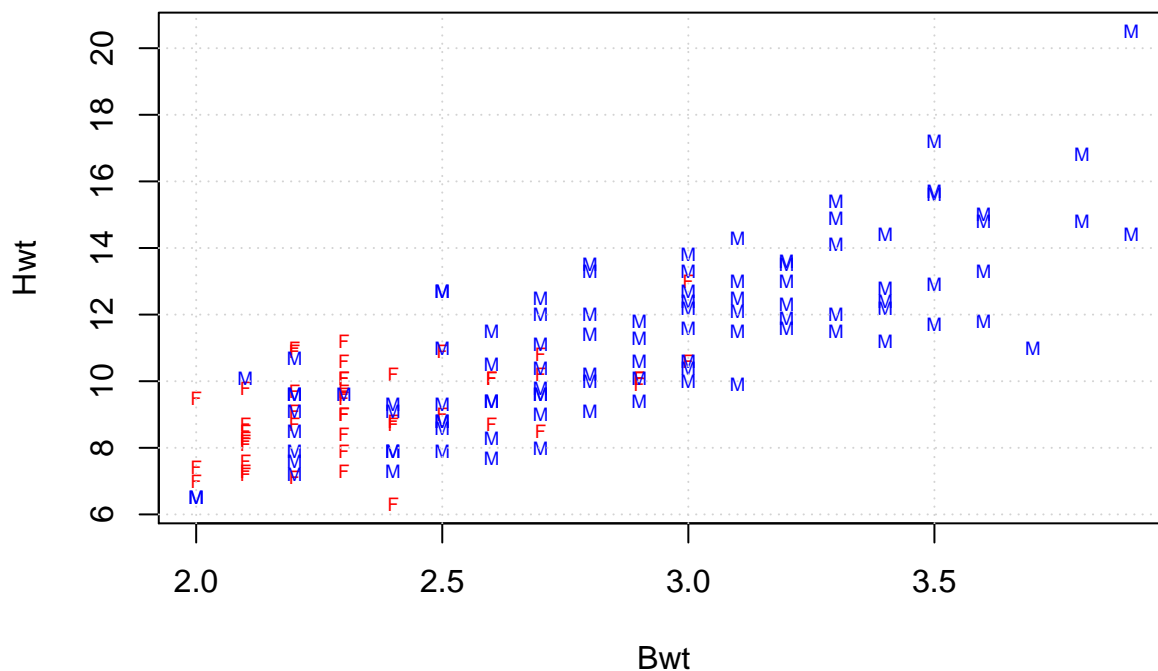
#####

```
library(MASS)
data(cats)
summary(cats)
```

```
## Sex      Bwt      Hwt
## F:47  Min.   :2.000  Min.   : 6.30
## M:97  1st Qu.:2.300  1st Qu.: 8.95
##      Median :2.700  Median :10.10
##      Mean   :2.724  Mean   :10.63
##      3rd Qu.:3.025  3rd Qu.:12.12
##      Max.   :3.900  Max.   :20.50
```

```
?cats
colores<- c("red","blue")
plot(cats[,2:3],type="n",main="Datos Cats")
text(cats[,2:3],as.character(cats$Sex),
     col=colores[cats$Sex],cex=0.6)
grid()
```

Datos Cats



```
cats.lda<-lda(Sex~.,cats)
cats.lda
```

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

##
##      F  M
## F 31 16
## M 12 85
erroremp<- mean(cats$Sex!=predict(cats.lda)$class)
cat("Error empírico=",100*erroremp ,"% \n")

## Error empírico= 19.44444 %
#lda admite CV=TRUE que implementa n-VC
#o sea, Jackknife
cats.ldaJ<-lda(Sex~.,cats,CV=TRUE)
str(cats.ldaJ)

## List of 5
## $ class      : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
## $ posterior: num [1:144, 1:2] 0.769 0.772 0.783 0.707 0.708 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : chr [1:144] "1" "2" "3" "4" ...
## .. ..$ : chr [1:2] "F" "M"
## $ terms      :Classes 'terms', 'formula' language Sex ~ Bwt + Hwt
## .. ..- attr(*, "variables")= language list(Sex, Bwt, Hwt)
## .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
## .. ..- attr(*, "dimnames")=List of 2
## .. .. ..$ : chr [1:3] "Sex" "Bwt" "Hwt"
## .. .. ..$ : chr [1:2] "Bwt" "Hwt"
## .. ..- attr(*, "term.labels")= chr [1:2] "Bwt" "Hwt"
## .. ..- attr(*, "order")= int [1:2] 1 1
## .. ..- attr(*, "intercept")= int 1
## .. ..- attr(*, "response")= int 1
## .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
## .. ..- attr(*, "predvars")= language list(Sex, Bwt, Hwt)
## .. ..- attr(*, "dataClasses")= Named chr [1:3] "factor" "numeric" "numeric"
## .. .. ..- attr(*, "names")= chr [1:3] "Sex" "Bwt" "Hwt"
## $ call       : language lda(formula = Sex ~ ., data = cats, CV = TRUE)
## $ xlevels    : Named list()
```

```

table(cats$Sex,cats.ldaJ$class)

##
##      F  M
##  F 31 16
##  M 14 83

errorJ<- mean(cats$Sex!=cats.ldaJ$class)
cat("Error Jackknife=",100*errorJ ,"% \n")

## Error Jackknife= 20.83333 %

#Cómo se generan muestras bootstrap de conjuntos
#de datos. Por ejemplo
datos=cats[c(1:5,140:144),]
datos

##      Sex Bwt  Hwt
## 1      F 2.0  7.0
## 2      F 2.0  7.4
## 3      F 2.0  9.5
## 4      F 2.1  7.2
## 5      F 2.1  7.3
## 140     M 3.7 11.0
## 141     M 3.8 14.8
## 142     M 3.8 16.8
## 143     M 3.9 14.4
## 144     M 3.9 20.5

datos[sample(1:nrow(datos),rep=TRUE),]

##      Sex Bwt  Hwt
## 4      F 2.1  7.2
## 142     M 3.8 16.8
## 142.1    M 3.8 16.8
## 4.1      F 2.1  7.2
## 144     M 3.9 20.5
## 1      F 2.0  7.0
## 3      F 2.0  9.5
## 141     M 3.8 14.8
## 142.2    M 3.8 16.8
## 4.2      F 2.1  7.2

#Estimaciones Bootstrap:
#####
B<- 2000
errorboot<-numeric(B)
error00B<- numeric(B)
n<- nrow(cats)
indin<- 1:n
for (b in 1:B)
{
  if (b%500==0) cat("Muestra bootstrap número ",b,"\n")
  #Generar muestra bootstrap de los índices
  indiB<- sample(indin,rep=T)
  #Obtener los índices no incluidos en imuestrab
  indi00B<-setdiff(indin,indiB)

```

```

#Construir modelo lda sobre la muestra bootstrap
cats.lda.boot<-lda(Sex~.,cats[indiB,])
#Calcular tasa de error en la muestra original
errorboot[b]<- mean(cats$Sex!=predict(cats.lda.boot,cats)$class)
#Obtener predicciones OOB
prediOOB<- predict(cats.lda.boot,cats[indiOOB,])$class
#Calcular la tasa de error OOB
errorOOB[b]<- mean(cats$Sex[indiOOB]!=prediOOB)
}

```

```

## Muestra bootstrap número 500
## Muestra bootstrap número 1000
## Muestra bootstrap número 1500
## Muestra bootstrap número 2000

```

```

errorB<- mean(errorboot) #no recomendable
errorB

```

```
## [1] 0.2225938
```

```

errorOOB<- mean(errorOOB)
error632B<-0.368*erroremp+0.632*errorOOB

```

```

#Calcular cada elemento Lij
#directamente:
matrizL<- matrix(NA,n,n)
for (i in 1:n)
  for (j in 1:n)
    matrizL[i,j]<- (cats[i,]$Sex!=predict(cats.lda,cats[j,])$class)
print(Noinf<- mean(matrizL))

```

```
## [1] 0.4300733
```

```

#O bien, en un problema de clasificación,
#con error 0-1, Noinf se puede calcular
#de forma más eficiente:
p1<- mean(cats$Sex=="M")
q1<- mean(predict(cats.lda)$class=="M")
p1*(1-q1)+(1-p1)*q1

```

```
## [1] 0.4300733
```

```

#Redefinición de errorOOB (aquí no hace falta)
errorOOB

```

```
## [1] 0.2323681
```

```
Noinf
```

```
## [1] 0.4300733
```

```
erroremp
```

```
## [1] 0.1944444
```

```

errorOOB=min(errorOOB,Noinf)
errorOOB

```

```
## [1] 0.2323681
```

```

#Cálculo de tsr y w
(tsr<- (error00B-erroremp)/(Noinf-erroremp))

## [1] 0.1609467
(w<- 0.632/(1-0.368*tsr))

## [1] 0.671789
#Posible redefinición de tsr
if ( (error00B<= erroremp) | (Noinf <=erroremp) ) tsr=0
tsr

## [1] 0.1609467
#(error632masB<-(1-w)*erroremp+w*error00B)
#Definición general (la línea anterior no vale
#si se redefinen tsr o error00B)
error632masB=error632B+
  (error00B-erroremp)*(0.368*0.632*tsr)/(1-0.368*tsr)
error632masB

## [1] 0.2199212
op_ant = options(digits=4)
cat(" Error Empírico=\t",100*erroremp ,"% \n",
    "Error Jackknife=\t",100*errorJ ,"% \n",
    "Error 00B=\t\t", 100*error00B,"% \n",
    "Error 0.632Boot=\t", 100*error632B,"% \n",
    "Error 0.632+Boot=\t", 100*error632masB,"% \n")

## Error Empírico= 19.44 %
## Error Jackknife= 20.83 %
## Error 00B= 23.24 %
## Error 0.632Boot= 21.84 %
## Error 0.632+Boot= 21.99 %

options(op_ant)

```

Ejercicio 8

8. Estimar el error de predicción para los datos “Renta.txt”, siendo “rentsqm” (precio del alquiler por m^2) la variable dependiente de un modelo de regresión lineal. Utilizar el criterio RECM.

Solución

Parte 1

```
#8. ESTIMACION DEL ERROR DE PREDICCION (REGRESION)
#####
#1. Leer los datos
datos<-read.table("datos/Renta.txt",header=T)
dim(datos)

## [1] 118    6

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
```

Parte 2

```
#2. Modelo de regresión lineal múltiple
modelo=lm(rentsqm~.,data=datos)
#Calcular MSE y RMSE empírico
error_emp=mean(residuals(modelo)^2)
RMSE_emp=sqrt(error_emp)
```

Parte 3

```
#3. Estimaciones Jackknife y bootstrap
#-----

#3.1. 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(rentsqr~.,data=datos[-i,])
  prediJ[i]<-predict(modelo.i,datos[i,])
}

resi_J=datos$rentsqr - prediJ
RMSE_J<-sqrt( mean(resi_J^2) )

```

Parte 4

```

#3.2. Bootstrap:
#Se pueden calcular los estimadores de ECM (MSE)
#y al final visualizar su raíz cuadrada
B<-2000
errorboot<-numeric(B)
errorOOB<- numeric(B)
indin<-1:n

for(b in 1:B){
  if (b%500==0) cat("Muestra bootstrap número ",b,"\n")
  #Generar muestra bootstrap de los índices
  indiB<- sample(indin,rep=T)
  #Obtener los índices no incluidos en imuestrab
  indiOOB<-setdiff(indin,indiB)
  #Construir modelo lda sobre la muestra bootstrap
  modelo.boot<-lm(rentsqr~.,data=datos[indiB,])
  #Calcular ECM en la muestra original
  suppressWarnings({ errorboot[b]<- mean((datos$rentsqr[indiB]-predict(modelo.boot,datos))^2) })
  #Obtener predicciones OOB
  suppressWarnings({ prediOOB<- predict(modelo.boot,datos[indiOOB,]) })
  #Calcular ECM OOB
  errorOOB[b]<- mean((datos$rentsqr[indiOOB]-prediOOB)^2)
}

## Muestra bootstrap número 500
## Muestra bootstrap número 1000
## Muestra bootstrap número 1500
## Muestra bootstrap número 2000

errorB<- mean(errorboot)
errorOOB<- mean(errorOOB)
error632B<-0.368*error_emp+0.632*errorOOB

#Calcular cada elemento L_ij
#Al no ser un problema de clasificación,
#se debe calcular directamente:
matrizL<- matrix(NA,n,n)
for (i in 1:n)
  for (j in 1:n)
    matrizL[i,j]<- (datos$rentsqr[i]-predict(modelo,datos[j,]))^2 #error cuad.
print(Noinf<- mean(matrizL))

## [1] 3.164884

```

```

#Redefinición de error00B
#(por si hiciera falta)
error00B #Noinf< error_emp ?

## [1] 0.944598
Noinf      #error_emp<Noinf< error00B?

## [1] 3.164884
error_emp

## [1] 0.804228
error00B=min(error00B,Noinf)
error00B

## [1] 0.944598
#Cálculo de tsr y w
(tsr<- (error00B-error_emp)/(Noinf-error_emp))

## [1] 0.05946229
(w<- 0.632/(1-0.368*tsr))

## [1] 0.6461389
#Posible redefinición de tsr
if ( (error00B<= error_emp) | (Noinf <= error_emp) ) tsr=0
tsr

## [1] 0.05946229
#(error632masB<-(1-w)*erroremp+w*error00B)
#Definición general (la línea anterior no vale
#si se redefinen tsr o error00B)
error632masB=error632B+
  (error00B-error_emp)*(0.368*0.632*tsr)/(1-0.368*tsr)
error632masB

## [1] 0.8949265
cat("  RMSE Empírico=\t",sqrt(RMSE_emp), "\n",
    "  RMSE Jackknife=\t", RMSE_J, "\n",
    "  RMSE 00B=\t\t", sqrt(error00B),"\n",
    "  RMSE 0.632Boot=\t", sqrt(error632B),"\n",
    "  RMSE 0.632+Boot=\t", sqrt(error632masB),"\n")

##  RMSE Empírico= 0.9469887
##  RMSE Jackknife= 0.9483709
##  RMSE 00B= 0.9719043
##  RMSE 0.632Boot= 0.944956
##  RMSE 0.632+Boot= 0.9460056

```