EjemplosJackknife

Pedro Luque

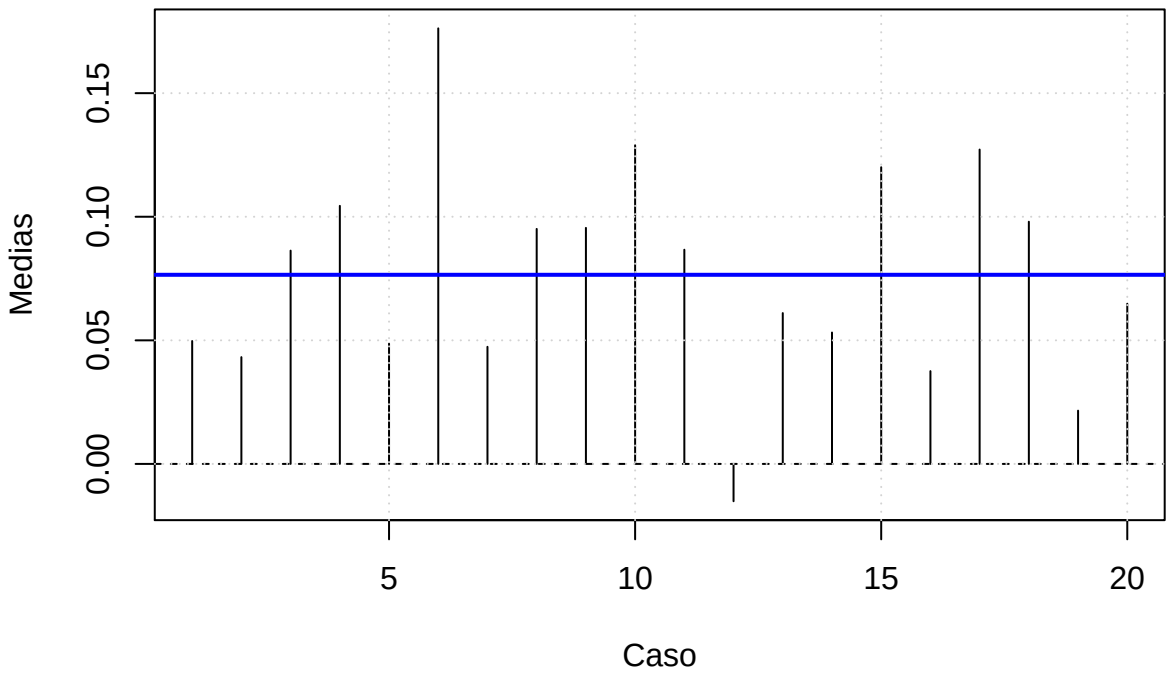
```
##########################################
##ESTADISTICA COMPUTACIONAL I             #
##GRADO EN ESTADISTICA                    #
##DOBLE GRADO EN MATEMATICAS Y ESTADISTICA #
##EJEMPLOS JACKKNIFE                      #
##########################################

###############################################
##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
```

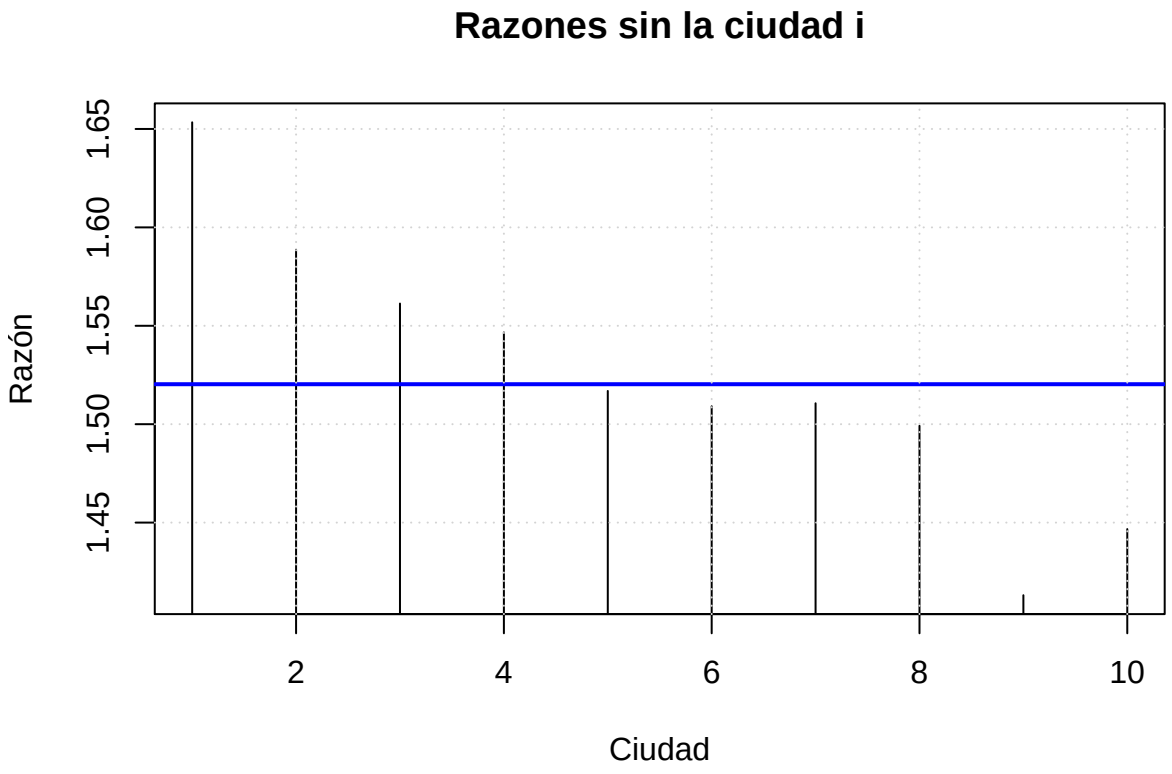
```r
##y la varianza del estimador de la razón (cociente
#de los totales de las variables x y u) en el fichero
#city de R (fichero disponible en la librería boot):
#########################################################
library(boot)
data(city)
#city
#?city
#EStimador
print(R<- sum(city$x)/sum(city$u)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

```
## [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
```

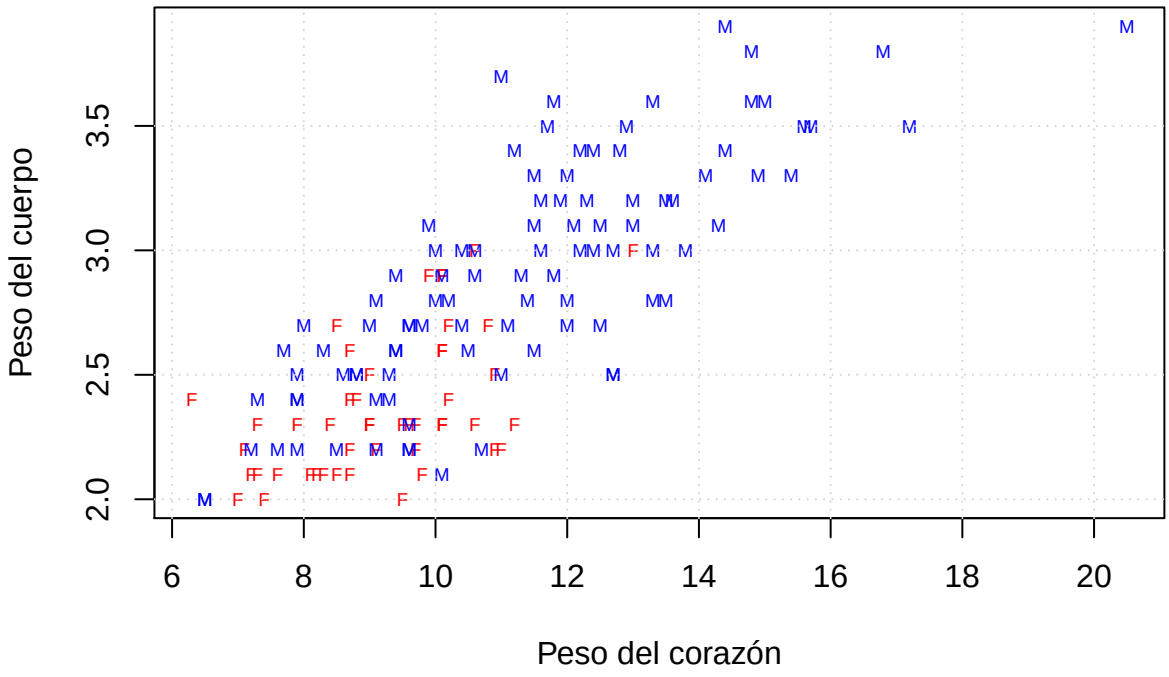
```r
#Estimador incluyendo la corrección del sesgo
Rjack<- R-sesgoj; Rjack
```

```
## [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()
```




```
#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)
```