

Técnicas de Remuestreo

Ricardo Cao Abad (rcao@udc.es) y Rubén Fernández Casal (rfcasal@udc.es)

2021-11-25

Índice general

Prólogo	5
1 Motivación del principio Bootstrap	7
1.1 Introducción	7
1.2 El Bootstrap uniforme	13
1.3 Cálculo de la distribución Bootstrap: exacta y aproximada	18
1.4 Herramientas disponibles en R sobre bootstrap	23
2 Estimación de la precisión y el sesgo de un estimador	31
2.1 Estimación bootstrap de la precisión y el sesgo de un estimador	31
2.2 Motivación del método Jackknife	36
2.3 Estimación Jackknife de la precisión y el sesgo de un estimador	36
2.4 Relación Bootstrap/Jackknife en dicha estimación	40
3 Modificaciones del Bootstrap uniforme	43
3.1 Bootstrap paramétrico	43
3.2 Bootstrap simetrizado	47
3.3 Bootstrap suavizado	49
3.4 Bootstrap ponderado y bootstrap sesgado	54
3.5 Deficiencias del bootstrap uniforme	55
3.6 Validez de la aproximación Bootstrap	58
3.7 Bootstrap semiparamétrico y bootstrap residual	61
4 Intervalos de confianza bootstrap	67
4.1 Intervalos basados en la distribución normal asintótica	67
4.2 Método percentil (básico)	68
4.3 Método percentil- <i>t</i>	71
4.4 Método percentil- <i>t</i> simetrizado	72
4.5 Tabla resumen de los errores de cobertura	73
4.6 Ejemplos	74
5 Aplicaciones del Bootstrap en contrastes de hipótesis	81
5.1 Aproximación del p-valor mediante remuestreo	82
5.2 Contrast es bootstrap paramétricos	82
5.3 Contrast es de permutaciones	86
5.4 Contrast es bootstrap semiparamétricos	89
6 Bootstrap y estimación no paramétrica de la densidad	97
6.1 Estimación no paramétrica de la función de densidad	97
6.2 Sesgo, varianza y error cuadrático medio	97
6.3 Aproximación Bootstrap de la distribución del estimador de Parzen-Rosenblatt	99
6.4 El Bootstrap en la selección del parámetro de suavizado.	100
6.5 Estimación no paramétrica de la densidad en R	102
6.6 Ejemplos	104

7 Bootstrap y regresión no paramétrica	109
7.1 Estimador de Nadaraya-Watson	109
7.2 Métodos de remuestreo en regresión no paramétrica	111
7.3 Regresión polinómica local en R	114
7.4 Ejemplos	116
8 El Bootstrap con datos censurados	121
8.1 Introducción a los datos censurados	121
8.2 Remuestreos Bootstrap en presencia de censura	123
8.3 Relaciones entre los métodos de remuestreo bajo censura	124
8.4 Implementación en R (con los paquetes <code>boot</code> y <code>survival</code>)	125
8.5 Ejercicios	128
9 El Bootstrap con datos dependientes	131
9.1 Introducción a las condiciones de dependencia y modelos habituales de datos dependientes	131
9.2 El bootstrap en la estimación con datos dependientes	132
9.3 El bootstrap para la predicción con datos dependientes	137
9.4 Implementación en R	139
9.5 Implementación en R con el paquete <code>boot</code>	140
9.6 Ejercicios	143
9.7 Implementación en R con el paquete <code>forecast</code>	147
9.8 Spatial data	147
Referencias	149
A Enlaces	155
A.1 Forecasting: Principles and Practice	156
A.2 Spatial data	157
B Introducción al procesamiento en paralelo en R	159
B.1 Introducción	159
B.2 Paquetes en R	159
B.3 Ejemplos	160

Prólogo

Este libro contiene los apuntes de la asignatura de Técnicas de Remuestreo del Máster en Técnicas Estadísticas.

Este libro ha sido escrito en R-Markdown empleando el paquete `bookdown` y está disponible en el repositorio Github: [rubenfcasal/book_remuestreo](https://rubenfcasal.github.io/book_remuestreo). Se puede acceder a la versión en línea a través del siguiente enlace:

https://rubenfcasal.github.io/book_remuestreo.

donde puede descargarse en formato pdf.

Para ejecutar los ejemplos mostrados en el libro será necesario tener instalados los siguientes paquetes: `boot`, `bootstrap`, `survival`, `forecast`, `MASS`, `sm`, `snow`. Por ejemplo mediante el comando:

```
install.packages(c("boot", "bootstrap", "survival", "forecast", "MASS", "sm", "snow"))
```

Para generar el libro (compilar) serán necesarios paquetes adicionales, para lo que se recomendaría consultar el libro de “Escritura de libros con bookdown” en castellano.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional (esperamos poder liberarlo bajo una licencia menos restrictiva más adelante...).



Capítulo 1

Motivación del principio Bootstrap

Etimología: bootstrap = cinta de la bota (oreja lateral para calzarse las botas). Modismo anglosajón: to pull oneself up by one's bootstraps.

1.1 Introducción

El bootstrap es un procedimiento estadístico que sirve para aproximar la distribución en el muestreo (normalmente) de un estadístico. Para ello procede mediante remuestreo, es decir, obteniendo muestras mediante algún procedimiento aleatorio que utilice la muestra original.

Su ventaja principal es que no requiere hipótesis sobre el mecanismo generador de los datos. Sí las requiere, aunque suelen ser más relajadas, para obtener propiedades asintóticas del mismo. Por otra parte, su implementación en ordenador suele ser sencilla, en comparación con otros métodos. Su principal inconveniente es la necesidad de computación intensiva, debido a la fuerza bruta del método de Monte Carlo. Con la capacidad computacional actual, esta mayor carga computacional del bootstrap no suele ser un problema hoy en día. En raras ocasiones el bootstrap no necesita del uso de técnicas de Monte Carlo.

1.1.1 Breve nota histórica

Precursores teóricos remotos:

- Laplace (1810). Teoría límite de primer orden.
- Chebychev (final siglo XIX). Teoría límite de segundo orden.

Primeras contribuciones:

- Hubback (1878-1968). Esquemas de muestreo espacial para ensayos agrícolas.
- Mahalanobis (años 1930 y segunda guerra mundial). Precursor del bootstrap por bloques.

Otras contribuciones:

- Gurney, McCarthy, Hartigan (años 1960, 1970). Métodos de half-sampling para estimación de varianzas (U.S. Bureau of the Census).
- Maritz, Jarret, Simon (años 1970, 1980). Métodos de permutaciones relacionados con el bootstrap.

En la actualidad:

- Bradley Efron (Stanford University, 1979). Creador oficial del método. Acuñó su nombre. Fusionó la potencia de Monte Carlo con la resolución de problemas planteados de forma muy general.
- Peter Hall (1951-2016). Fue uno de los estadísticos contemporáneos más prolíficos. Dedicó al bootstrap gran parte de su producción a partir de los años 1980.

1.1.2 Paradigma inferencial y análogo bootstrap

Paradigma inferencial

Suponemos que $\mathbf{X} = (X_1, \dots, X_n)$ es una m.a.s. de una población con distribución F y que estamos interesados en hacer inferencia sobre $\theta = \theta(F)$. Para ello nos gustaría conocer la distribución en el muestreo de $R(\mathbf{X}, F)$, cierto estadístico función de la muestra y de la distribución poblacional. Por ejemplo:

$$R = R(\mathbf{X}, F) = \theta(F_n) - \theta(F) = \hat{\theta} - \theta,$$

siendo F_n la función de distribución empírica.

A veces podemos calcular directamente la distribución de $R(\mathbf{X}, F)$, aunque suele depender de cantidades poblacionales, no conocidas en la práctica. Por ejemplo, bajo normalidad $X_i \stackrel{i.i.d.}{\sim} \mathcal{N}(\mu, \sigma^2)$, si estamos interesados en

$$\theta(F) = \mu = \int x \, dF(x) = \int xf(x) \, dx$$

como $\theta(F_n) = \int x \, dF_n(x) = \sum \frac{1}{n} X_i = \bar{X}$, podríamos considerar el estadístico:

$$R = R(\mathbf{X}, F) = \bar{X} - \mu \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\right).$$

Aunque en la práctica la varianza no es normalmente conocida y habría que aproximarla (sería preferible considerar como estadístico la media estandarizada).

Otras veces sólo podemos llegar a aproximar la distribución de $R(\mathbf{X}, F)$ cuando $n \rightarrow \infty$. Por ejemplo, cuando estamos interesados en la media pero desconocemos la distribución de los datos.

Análogo bootstrap

El primer paso es reemplazar la distribución poblacional (desconocida) F por una estimación, \hat{F} , de la misma. Por ejemplo, podríamos considerar la distribución empírica $\hat{F} = F_n$ (bootstrap uniforme; Sección 1.2), o una aproximación paramétrica $\hat{F} = F_{\hat{\theta}}$ (bootstrap paramétrico; Sección 3.1).

Como ejemplo ilustrativo consideramos los datos simulados [Figura 1.1]:

```
set.seed(1)
muestra <- rnorm(100)
hist(muestra, freq = FALSE, xlim = c(-3, 3),
     main = '', xlab = 'x', ylab = 'densidad')
curve(dnorm, lty = 2, add = TRUE)
```

Como aproximación de la distribución poblacional, desconocida en la práctica, siempre podemos considerar la distribución empírica (o una versión suavizada: bootstrap suavizado; Sección 3.3). Alternativamente podríamos asumir un modelo paramétrico y estimar los parámetros a partir de la muestra [Figura 1.2].

```
# Distribución bootstrap uniforme
curve(ecdf(muestra)(x), xlim = c(-3, 3), ylab = "F(x)", type = "s")
# Distribución bootstrap paramétrico (asumiendo normalidad)
curve(pnorm(x, mean(muestra), sd(muestra)), lty = 2, add = TRUE)
# Distribución teórica
curve(dnorm, lty = 3, add = TRUE)
legend("bottomright", legend = c("Empírica", "Aprox. paramétrica", "Teórica"), lty = 1:3)
```

A partir de la aproximación \hat{F} podríamos generar, condicionalmente a la muestra observada, remuestras

$$\mathbf{X}^* = (X_1^*, \dots, X_n^*)$$

con distribución $X_i^* \sim \hat{F}$, que denominaremos remuestras bootstrap. Por lo que podemos hablar de la distribución en el remuestreo de

$$R^* = R(\mathbf{X}^*, \hat{F}),$$

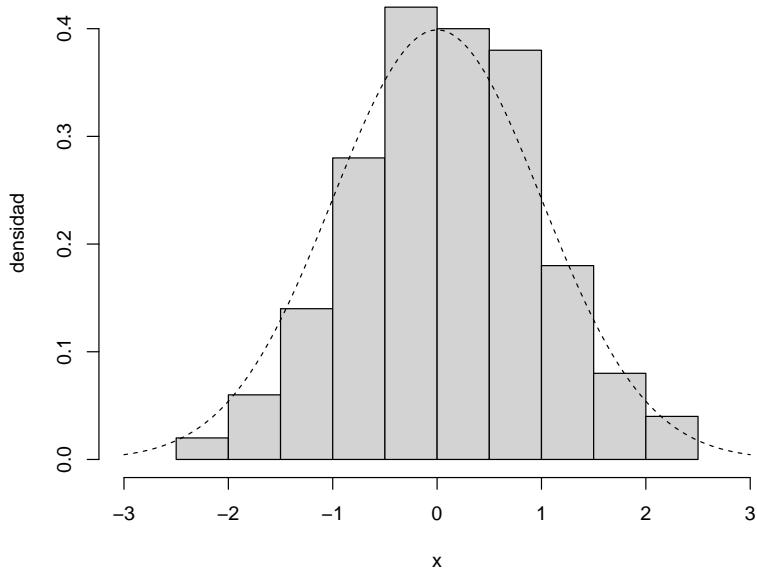


Figura 1.1: Distribución de la muestra simulada.

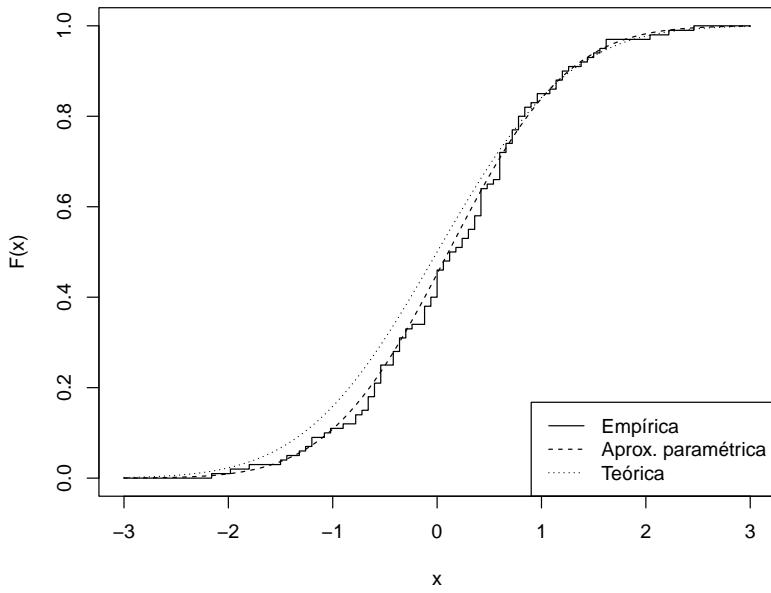


Figura 1.2: Distribución teórica de la muestra simulada y distintas aproximaciones.

llamada distribución bootstrap.

La idea original (Efron, 1979) es que la distribución de $\hat{\theta}_b^*$ en torno a $\hat{\theta}$ aproxima la distribución de $\hat{\theta}$ en torno a θ . Por tanto se pretende aproximar la distribución en el muestreo de R por la distribución bootstrap de R^* .

En raras ocasiones la distribución bootstrap de R^* es calculable directamente, pero siempre suele

poder aproximarse por Monte Carlo.

1.1.3 Implementación en la práctica

En el caso i.i.d., si empleamos como aproximación la distribución empírica $\hat{F} = F_n$, la generación de las muestras bootstrap puede hacerse mediante remuestreo (manteniendo el tamaño muestral). Habría que simular una muestra de tamaño n de una variable aleatoria discreta que toma los valores X_1, \dots, X_n todos ellos con probabilidad $\frac{1}{n}$:

- Para cada $i = 1, \dots, n$, $P^*(X_i^* = X_j) = \frac{1}{n}$, $j = 1, \dots, n$.

Existen multitud de algoritmos para simular variables discretas, pero en este caso de equiprobabilidad hay un procedimiento muy eficiente (método de la transformación cuantil con búsqueda directa) que se reduce a simular un número aleatorio U , con distribución $\mathcal{U}(0, 1)$, y hacer $X^* = X_{\lfloor nU \rfloor + 1}$, donde $\lfloor x \rfloor$ representa la parte entera de x , es decir, el mayor número entero que sea menor o igual que x . Empleando ese método, el procedimiento para generar la muestra bootstrap sería:

- Para cada $i = 1, \dots, n$ generar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$.

```
set.seed(1)
n <- length(muestra)
u <- runif(n)
muestra_boot <- muestra[floor(n*u) + 1]
head(muestra_boot)
```

```
## [1] -0.1557955 -0.0593134 -1.0441346 -0.5425200  0.9189774  0.2670988
```

En R es recomendable¹ emplear la función `sample` para generar muestras aleatorias con reemplazamiento del conjunto de datos original:

```
muestra_boot <- sample(muestra, replace = TRUE)
head(muestra_boot)
```

```
## [1] -1.4707524  0.7685329  0.3876716 -0.6887557  0.9189774  1.3586796
```

En el caso multidimensional, cuando trabajamos con un conjunto de datos con múltiples variables, podríamos emplear un procedimiento análogo, a partir de remuestras del vector de índices. Por ejemplo:

```
data(iris)
str(iris)
```

```
## 'data.frame': 150 obs. of  5 variables:
## $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
## $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
## $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
## $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
## $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
n <- nrow(iris)
# i_boot <- floor(n*runif(n)) + 1
# i_boot <- sample.int(n, replace = TRUE)
i_boot <- sample(n, replace = TRUE)
data_boot <- iris[i_boot, ]
str(data_boot)

## 'data.frame': 150 obs. of  5 variables:
## $ Sepal.Length: num  6.9 7.2 4.3 6.4 5.7 5.8 7.2 5.6 5.8 5.4 ...
## $ Sepal.Width : num  3.1 3.2 3 3.2 4.4 4 3 2.9 2.7 3.9 ...
## $ Petal.Length: num  5.4 6 1.1 5.3 1.5 1.2 5.8 3.6 5.1 1.3 ...
## $ Petal.Width : num  2.1 1.8 0.1 2.3 0.4 0.2 1.6 1.3 1.9 0.4 ...
```

¹De esta forma se evitan posibles problemas numéricos al emplear el método de la transformación cuantil cuando n es extremadamente grande (e.g. <https://stat.ethz.ch/pipermail/r-devel/2018-September/076817.html>).

```
## $ Species      : Factor w/ 3 levels "setosa","versicolor",..: 3 3 1 3 1 1 3 2 3 1 ...
```

Esta forma de proceder es la que emplea por defecto el paquete `boot` que describiremos más adelante (Sección 1.4.1).

Ejemplo 1.1 (Inferencia sobre la media con varianza conocida). Hemos observado 15 tiempos de vida de microorganismos: 0.143, 0.182, 0.256, 0.260, 0.270, 0.437, 0.509, 0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08. A partir de los cuales queremos obtener una estimación por intervalo de confianza de su vida media, suponiendo que la desviación típica es conocida e igual a 0.6 (en el Capítulo 4 se tratará con más detalle la construcción de intervalos de confianza).

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
           0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
sigma <- 0.6
summary(muestra)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##  0.1430  0.2650  0.6110  0.8053  1.1200  2.0800

sd(muestra)

## [1] 0.6237042
hist(muestra)
rug(muestra)
```

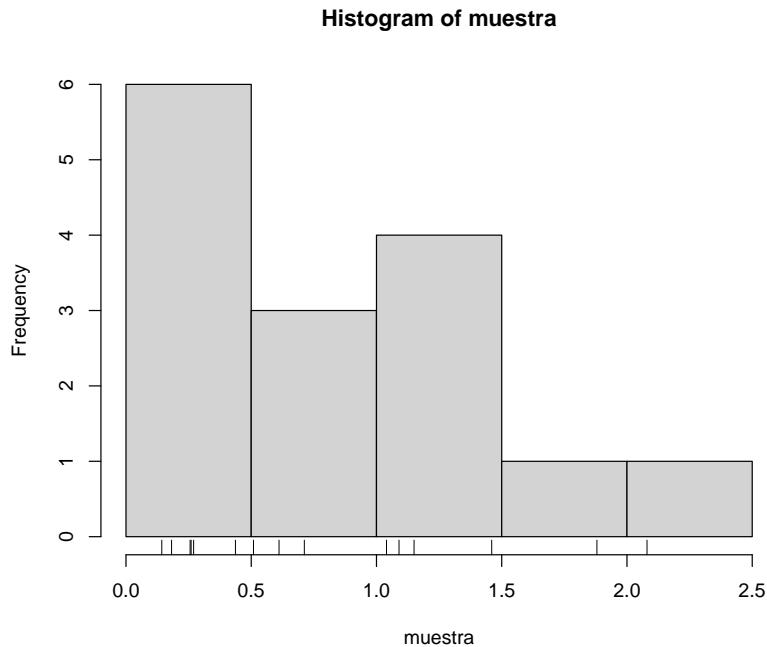


Figura 1.3: Distribución del tiempo de vida de microorganismos.

[Figura 1.3]

Contexto clásico

Suponemos que los datos $\mathbf{X} = (X_1, \dots, X_n)$ son una m.a.s. de una población con distribución F , con μ desconocida y σ conocida, y que estamos interesados en hacer inferencia sobre:

$$\theta(F) = \mu = \int x dF(x)$$

Para ello, un estadístico adecuado para este caso es:

$$R = R(\mathbf{X}, F) = \sqrt{n} \frac{\bar{X} - \mu}{\sigma},$$

con $\theta(F_n) = \int x dF_n(x) = \bar{X}$.

Bajo normalidad ($X \sim \mathcal{N}(\mu, \sigma^2)$), $R \sim N(0, 1)$. Si F no es normal, tan sólo sabemos que, bajo ciertas condiciones, $R \xrightarrow{d} \mathcal{N}(0, 1)$.

A partir de esta última aproximación, se obtiene el intervalo de confianza asintótico (de nivel $1 - \alpha$) para la media μ :

$$\hat{IC}_{1-\alpha}(\mu) = \left(\bar{X} - z_{1-\alpha/2} \frac{\sigma}{\sqrt{n}}, \bar{X} + z_{1-\alpha/2} \frac{\sigma}{\sqrt{n}} \right).$$

```
alfa <- 0.05
x_barra <- mean(muestra)
z <- qnorm(1 - alfa/2)
n <- length(muestra)
ic_inf <- x_barra - z*sigma/sqrt(n)
ic_sup <- x_barra + z*sigma/sqrt(n)
IC <- c(ic_inf, ic_sup)
IC
```

[1] 0.501697 1.108970

Contexto bootstrap

Consideramos la función de distribución empírica $\hat{F} = F_n$ como aproximación de la distribución poblacional (bootstrap uniforme). Para aproximar la distribución bootstrap del estadístico por Monte Carlo, se generan $B = 1000$ muestras bootstrap $\mathbf{X}^{*(b)} = (X_1^{*(b)}, \dots, X_n^{*(b)})$ de forma que $P^{*(b)}(X_i^* = X_j) = \frac{1}{n}$, $j = 1, \dots, n$, para $i = 1, \dots, n$ y $b = 1, \dots, B$. A partir de las cuales se obtienen las B réplicas bootstrap del estadístico:

$$R^{*(b)} = R(\mathbf{X}^{*(b)}, \hat{F}) = \sqrt{n} \frac{\bar{X}^{*(b)} - \bar{X}}{\sigma}, \quad b = 1, \dots, B,$$

con $\bar{X}^{*(b)} = \frac{1}{n} \sum X_i^{*(b)}$.

```
set.seed(1)
B <- 1000
estadistico_boot <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  x_barra_boot <- mean(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot - x_barra)/sigma
}
```

Las características de interés de la distribución en el muestreo de R se aproximan por las correspondientes de la distribución bootstrap de R^* . En este caso nos interesa aproximar los puntos críticos $x_{\alpha/2}$ y $x_{1-\alpha/2}$, tales que:

$$P(x_{\alpha/2} < R < x_{1-\alpha/2}) = 1 - \alpha.$$

Para lo que podemos emplear los cuantiles muestrales²:

²Se podrían considerar distintos estimadores del cuantil x_α (ver p.e. la ayuda de la función `quantile()`). Si empleamos directamente la distribución empírica, el cuantil se correspondería con la observación ordenada en la posición $B\alpha$ (se suele hacer una interpolación lineal si este valor no es entero), lo que equivale a emplear la función `quantile()` de R con el parámetro `type = 1`. Esta función considera por defecto la posición $1 + (B - 1)\alpha$ (`type = 7`). En el libro de Davison y Hinkley (1997), y en el paquete `boot`, se emplea $(B + 1)\alpha$ (equivalente a `type = 6`; lo que justifica que consideren habitualmente 99, 199 ó 999 réplicas bootstrap).

```
# Empleando la distribución empírica del estadístico bootstrap:
estadistico_boot_ordenado <- sort(estadistico_boot)
indice_inf <- floor(B * alfa/2)
indice_sup <- floor(B * (1 - alfa/2))
pto_crit <- estadistico_boot_ordenado[c(indice_inf, indice_sup)]
# Empleando la función `quantile`:
# pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2), type = 1)
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))
pto_crit

##      2.5%    97.5%
## -1.918622  2.075984
```

A partir de los cuales obtenemos la correspondiente estimación por IC bootstrap:

$$\hat{IC}_{1-\alpha}^{boot}(\mu) = \left(\bar{X} - x_{1-\alpha/2} \frac{\sigma}{\sqrt{n}}, \bar{X} - x_{\alpha/2} \frac{\sigma}{\sqrt{n}} \right).$$

```
# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2] * sigma/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1] * sigma/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%") # rev(names(IC_boot))
IC_boot

##      2.5%    97.5%
## 0.4837233 1.1025650
```

Nótese que este intervalo de confianza no está centrado en la media, al contrario que el obtenido con la aproximación tradicional. Aunque en este caso no se observan grandes diferencias ya que la distribución bootstrap obtenida es muy similar a la aproximación normal (ver Figura 1.4).

```
hist(estadistico_boot, freq = FALSE)
lines(density(estadistico_boot))
abline(v = pto_crit)
curve(dnorm, lty = 2, add = TRUE)
abline(v = c(-z, z), lty = 2)
```

1.2 El Bootstrap uniforme

Como ya se comentó anteriormente el bootstrap uniforme es aquel en el que se reemplaza la distribución poblacional (desconocida) por la distribución empírica:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}\{X_i \leq x\}.$$

Es decir $\hat{F} = F_n$ y, por lo tanto, $R^* = R(\mathbf{X}^*, F_n)$.

Conviene recordar algunas propiedades de la distribución empírica:

$$\begin{aligned} nF_n(x) &= \sum_{i=1}^n \mathbf{1}\{X_i \leq x\} \sim \mathcal{B}(n, F(x)), \\ E(nF_n(x)) &= nF(x) \implies E(F_n(x)) = F(x), \\ Var(nF_n(x)) &= nF(x)(1 - F(x)) \\ &\implies Var(F_n(x)) = \frac{F(x)(1 - F(x))}{n} \end{aligned}$$

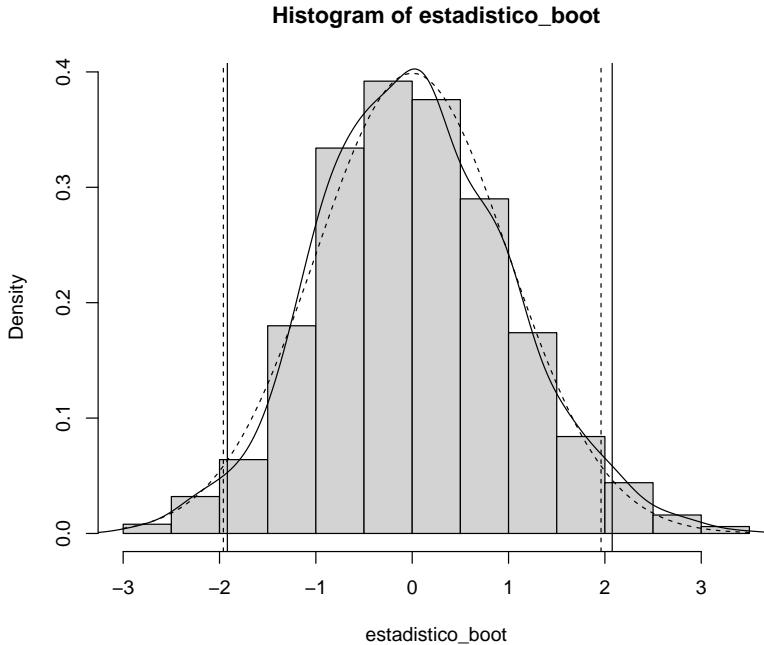


Figura 1.4: Distribución del estadístico bootstrap y aproximaciones de los cuantiles. Con línea discontinua se muestra la distribución normal asintótica.

Así pues, en este caso el algoritmo bootstrap uniforme (también llamado bootstrap naïve) es el siguiente:

1. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de F_n , es decir $P^*(X_i^* = X_j) = \frac{1}{n}, j = 1, \dots, n$
2. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, F_n)$

Como veremos más adelante, a veces (muy poco frecuentemente) es posible calcular exactamente la distribución bootstrap de R^* . Cuando eso no es posible, esa distribución es fácilmente aproximable por Monte Carlo, arrojando una gran cantidad, B , de réplicas de R^* . En ese caso, el algoritmo se convierte en:

1. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de F_n
2. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, F_n)$
4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Utilizar esas réplicas bootstrap para aproximar la distribución en el muestreo de R

1.2.1 Ejemplos

Ejemplo 1.2 (Inferencia sobre la media con varianza conocida, continuación).

En el Ejemplo 1.1 anteriormente visto de inferencia para la media con varianza conocida, el algoritmo bootstrap (basado en Monte Carlo) para aproximar la distribución en el muestreo de R empleado fue:

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$
2. Obtener $\bar{X}^* = \frac{1}{n} \sum X_i^*$
3. Calcular $R^* = \sqrt{n} \frac{\bar{X}^* - \bar{X}}{\sigma}$

4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Aproximar la distribución en el muestreo de R mediante la empírica de $R^{*(1)}, \dots, R^{*(B)}$

Como curiosidad podemos calcular la esperanza y la varianza de R y la esperanza y varianza bootstrap de R^* . Para R tenemos:

$$E(R) = \sqrt{n} \frac{E(\bar{X}) - \mu}{\sigma} = 0,$$

$$Var(R) = n \frac{Var(\bar{X})}{\sigma^2} = n \frac{\frac{1}{n}\sigma^2}{\sigma^2} = 1.$$

Para calcular esos mismos momentos de R^* , resultará útil obtener previamente la esperanza y varianza bootstrap de \bar{X}^* :

$$E^*(\bar{X}^*) = \frac{1}{n} \sum_{i=1}^n E^*(X_i^*) = \frac{1}{n} \sum_{i=1}^n E^*(X_1^*) = E^*(X_1^*) = \bar{X},$$

$$Var^*(\bar{X}^*) = \frac{1}{n^2} \sum_{i=1}^n Var^*(X_i^*) = \frac{1}{n^2} \sum_{i=1}^n Var^*(X_1^*) = \frac{1}{n} Var^*(X_1^*) = \frac{S_n^2}{n},$$

ya que

$$E^*(X_1^*) = \sum_{j=1}^n X_j P^*(X_1^* = X_j) = \sum_{j=1}^n \frac{1}{n} X_j = \bar{X},$$

$$Var^*(X_1^*) = E^*(X_1^{*2}) - [E^*(X_1^*)]^2 = \sum_{j=1}^n X_j^2 P^*(X_1^* = X_j) - \bar{X}^2$$

$$= \frac{1}{n} \sum_{j=1}^n X_j^2 - \bar{X}^2 = \frac{1}{n} \sum_{j=1}^n (X_j - \bar{X})^2 = S_n^2$$

Así pues, la esperanza y la varianza bootstrap de R^* resultan:

$$E^*(R^*) = \sqrt{n} \frac{E^*(\bar{X}^*) - \bar{X}}{\sigma} = 0,$$

$$Var^*(R^*) = n \frac{Var^*(\bar{X}^*)}{\sigma^2} = n \frac{\frac{1}{n} S_n^2}{\sigma^2} = \frac{S_n^2}{\sigma^2}.$$

Es curioso observar que la esperanza de R y la esperanza bootstrap de R^* coinciden (son ambas cero), pero no ocurre lo mismo con sus varianzas: la de R es 1 y la varianza bootstrap de R^* es S_n^2/σ^2 , que, aunque tiende a 1 (en probabilidad o de forma casi segura, bajo las condiciones adecuadas) cuando $n \rightarrow \infty$, no es igual a 1. Eso nos lleva a intuir que el método de remuestreo bootstrap propuesto quizás podría modificarse ligeramente para que imitase exactamente al caso no bootstrap también en la varianza. Puede comprobarse que eso se consigue remuestreando X^* de la distribución empírica de la muestra modificada: $(\tilde{X}_1, \dots, \tilde{X}_n)$, siendo

$$\tilde{X}_i = \bar{X} + \frac{\sigma}{S_n} (X_i - \bar{X}), \quad i = 1, \dots, n.$$

Efectivamente, bajo ese nuevo remuestreo, se tiene

$$E^*(\bar{X}^*) = E^*(X_1^*) = \bar{X} = \frac{1}{n} \sum_{i=1}^n \left[\bar{X} + \frac{\sigma}{S_n} (X_i - \bar{X}) \right]$$

$$= \bar{X} + \frac{1}{n} \frac{\sigma}{S_n} \sum_{i=1}^n (X_i - \bar{X}) = \bar{X},$$

$$Var^*(\bar{X}^*) = \frac{1}{n} Var^*(X_1^*) = \frac{\sigma^2}{n},$$

ya que

$$\begin{aligned} Var^*(X_1^*) &= E^*(X_1^{*2}) - [E^*(X_1^*)]^2 = \sum_{j=1}^n \tilde{X}_j^2 P^*(X_1^* = \tilde{X}_j) - \bar{\tilde{X}}^2 \\ &= \frac{1}{n} \sum_{j=1}^n \tilde{X}_j^2 - \bar{\tilde{X}}^2 = \frac{1}{n} \sum_{j=1}^n (\tilde{X}_j - \bar{\tilde{X}})^2 = \frac{1}{n} \sum_{j=1}^n \left[\frac{\sigma}{S_n} (X_j - \bar{X}) \right]^2 \\ &= \frac{\sigma^2}{S_n^2} \frac{1}{n} \sum_{j=1}^n (X_j - \bar{X})^2 = \frac{\sigma^2}{S_n^2} S_n^2 = \sigma^2. \end{aligned}$$

Como consecuencia

$$\begin{aligned} E^*(R^*) &= \sqrt{n} \frac{E^*(\bar{X}^*) - \bar{X}}{\sigma} = 0, \\ Var^*(R^*) &= n \frac{Var^*(\bar{X}^*)}{\sigma^2} = n \frac{\frac{\sigma^2}{n}}{\sigma^2} = 1. \end{aligned}$$

Esto es muy coherente con lo que nos diría la intuición pues, si la varianza poblacional, σ^2 , es conocida (ese es el motivo de que podamos usarla directamente en la definición del estadístico R), el plan de remuestreo bootstrap también ha de conocer σ^2 , es decir ha de diseñarse de modo que la distribución bootstrap de X^* tenga también varianza bootstrap σ^2 . Eso ocurre con el remuestreo uniforme de la muestra transformada $(\tilde{X}_1, \dots, \tilde{X}_n)$, pero no ocurre con el remuestreo naïve (a partir de la distribución empírica de la muestra original). Esto da pie a una de las consideraciones más importantes a la hora de diseñar un buen método de remuestreo bootstrap: ha de procurarse que **el bootstrap imite todas las condiciones que cumple la población original**.

El código para realizar remuestreo bootstrap uniforme sobre la empírica de la muestra perturbando es análogo:

```
# Remuestreo
B <- 1000
estadistico_boot <- numeric(B)
coeficiente <- sigma/sd(muestra)
muestra_perturbada <- x_barra + coeficiente * (muestra - x_barra)
for (k in 1:B) {
  remuestra <- sample(muestra_perturbada, n, replace = TRUE)
  x_barra_boot <- mean(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot - x_barra)/sigma
}

# Aproximación bootstrap de los ptos críticos
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))
# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2] * sigma/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1] * sigma/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.5024398 1.0827052
```

Ejemplo 1.3 (Inferencia sobre la mediana). Continuando con el ejemplo de los tiempos de vida de microorganismos, supongamos que queremos obtener una estimación por intervalo de confianza de su vida mediana a partir de los 15 valores observados.

Consideramos la mediana poblacional como parámetro de interés:

$$\theta = \theta(F) = F^{-1}\left(\frac{1}{2}\right) = \inf\left\{x \in \mathbb{R} : F(x) \geq \frac{1}{2}\right\}.$$

Dada una muestra $\mathbf{X} = (X_1, \dots, X_n) \sim F$, θ puede estimarse mediante la mediana muestral

$$\begin{aligned}\hat{\theta} &= \theta(F_n) = F_n^{-1}\left(\frac{1}{2}\right) = \inf\left\{x \in \mathbb{R} : F_n(x) \geq \frac{1}{2}\right\} \\ &= \begin{cases} X_{(m)} & \text{si } n = 2m - 1 \text{ es impar} \\ \frac{X_{(m)} + X_{(m+1)}}{2} & \text{si } n = 2m \text{ es par} \end{cases}\end{aligned}$$

siendo $X_{(1)}, \dots, X_{(n)}$ los estadísticos ordenados.

El estadístico interesante para realizar inferencia en este contexto es $R = \sqrt{n}(\hat{\theta} - \theta)$. Si la población de partida es continua, puede demostrarse que su distribución asintótica (i.e., cuando $n \rightarrow \infty$) viene dada por

$$R = \sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} \mathcal{N}\left(0, \frac{1}{f(\theta)^2}\right),$$

donde f es la función de densidad de la población. Como consecuencia, la utilización de esta distribución límite, $\mathcal{N}(0, 1/f(\theta)^2)$, para realizar inferencia sobre la mediana, además de comportar una aproximación de la distribución en el muestreo real, no puede utilizarse directamente porque la densidad (desconocida) aparece en la expresión de la varianza asintótica. Para ser utilizable en la práctica deberíamos estimar f , lo cual es un problema añadido.

Esta es pues una situación muy natural en la que usar un método bootstrap para aproximar la distribución de R . Consideremos como estimador de F la distribución empírica, F_n , y procedamos según un bootstrap uniforme (supongamos $n = 2m - 1$, impar, por simplicidad):

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{[nU_i]+1}$
2. Obtener $X_{(1)}^*, \dots, X_{(n)}^*$ los estadísticos ordenados de la remuestra bootstrap y quedarse con el que ocupa lugar central: $\hat{\theta}^* = \theta(F_n^*) = X_{(m)}^*$
3. Calcular $R^* = \sqrt{n}(X_{(m)}^* - X_{(m)})$
4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Aproximar la distribución en el muestreo de R mediante la empírica de $R^{*(1)}, \dots, R^{*(B)}$

El código implementando este algoritmo sería muy similar al de los casos anteriores:

```
x_mediana<- median(muestra)

# Remuestreo
B <- 1000
estadistico_boot <- numeric(B)
coeficiente <- sigma/sd(muestra)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  x_mediana_boot <- median(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_mediana_boot - x_mediana)
}

# Aproximación bootstrap de los ptos críticos
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))
# Construcción del IC
ic_inf_boot <- x_mediana - pto_crit[2]/sqrt(n)
ic_sup_boot <- x_mediana - pto_crit[1]/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

## 2.5% 97.5%
## 0.132 0.962
```

Sin embargo, como veremos más adelante, este caso de inferencia de la mediana es uno de los pocos casos en los que la distribución bootstrap se puede calcular de forma exacta, siendo dicha expresión utilizable en la práctica.

1.3 Cálculo de la distribución Bootstrap: exacta y aproximada

1.3.1 Distribución bootstrap exacta

En principio siempre es posible calcular la distribución en el remuestreo del estadístico bootstrap de forma exacta. Al menos para el bootstrap uniforme, que es el más habitual. El motivo es que la distribución de probabilidad de la que se remuestrea en el universo bootstrap es discreta y con un número finito de valores: X_1, \dots, X_n . Así pues, cada observación bootstrap, X_i^* , ha de tomar necesariamente alguno de esos n valores y, por tanto, el número de posibles remuestras, $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$, obtenibles mediante el bootstrap uniforme es finito, concretamente n^n . Aún siendo finito, este número es gigantescamente grande incluso para tamaños muestrales pequeños (salvo casos extremos del tipo $n = 2, \dots, 9$). Por ejemplo, para $n = 10$, tenemos 10^{10} (diez mil millones de) posibles remuestras bootstrap y para $n = 20$, tendríamos $20^{20} \simeq 10.4857 \cdot 10^{25}$ (algo más de cien cuatrillones). Incluso para estos tamaños muestrales el problema de cálculo de la distribución bootstrap exacta de \mathbf{X}^* es inabordable.

1.3.2 Vectores de remuestreo

Una forma alternativa de representar las posibles remuestras bootstrap es mediante los llamados vectores de remuestreo. Son utilizables en el caso de que el estadístico de interés sea funcional, es decir, cuando R depende de la muestra sólo a través de la distribución empírica o, lo que es lo mismo, el valor de R no cambia cuando realizamos una permutación arbitraria sobre los elementos de la muestra (los cambiamos de orden). Consideremos la remuestra bootstrap $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$ y denotemos por

$$N_j = \#\{i \in \{1, \dots, n\} : X_i^* = X_j\}.$$

Obviamente, si el orden en el que se han obtenido los elementos de la muestra no es importante, entonces el vector $\mathbf{N} = (N_1, \dots, N_n)$ contiene la misma información que la remuestra bootstrap \mathbf{X}^* . Esencialmente lo que hace el vector \mathbf{N} es contabilizar cuantas veces se repite cada elemento de la muestra original en la remuestra bootstrap. Con esta notación, el vector de remuestreo bootstrap, $\mathbf{P}^* = (P_1^*, \dots, P_n^*)$, se define como $P_i^* = \frac{N_i}{n}$, $i = 1, \dots, n$.

La distribución en el remuestreo de \mathbf{N} , bajo el bootstrap uniforme, es multinomial: $\mathbf{N} \sim \mathcal{M}_n(n, (\frac{1}{n}, \dots, \frac{1}{n}))$. Así que su masa de probabilidad, y por tanto la de \mathbf{P}^* , es fácilmente calculable:

$$\begin{aligned} P(N_1 = m_1, \dots, N_n = m_n) &= \frac{n!}{m_1! \cdots m_n!} \left(\frac{1}{n}\right)^{m_1} \cdots \left(\frac{1}{n}\right)^{m_n} \\ &= \frac{n!}{m_1! \cdots m_n! n^n}, \end{aligned}$$

para m_1, \dots, m_n enteros con $\sum_{i=1}^n m_i = n$, donde el número de átomos de probabilidad de \mathbf{N} es ahora $\binom{n+n-1}{n} = \binom{2n-1}{n}$.

En general $\binom{2n-1}{n} < n^n$, pues el hecho de que no importe el orden de las componentes de las remuestras bootstrap provoca un menor número de átomos de probabilidad. Aún así dicho cardinal es prohibitivamente grande incluso para tamaños muestrales pequeños: para $n = 10$, resultaría abordable pues $\binom{19}{10} = 92\,378$, pero para $n = 20$ tendríamos $\binom{39}{20} = 68\,923\,264\,410$. De toda esa enorme cantidad de átomos, el de más grande probabilidad resulta tener una probabilidad de $\frac{n!}{n^n}$, que es insignificanteamente pequeña para tamaños pequeños como $n = 20$, con $\frac{20!}{20^{20}} \sim 2.320\,2 \times 10^{-8}$. De todas formas, existen raras ocasiones en las que el número de átomos de probabilidad de \mathbf{P}^* resulta ser mucho menor que el de \mathbf{P} .

Para tamaños muestrales realmente pequeños es posible encontrar todos los átomos de probabilidad de la distribución bootstrap. Un ejemplo es la media muestral con, por ejemplo, $n = 3$.

Ejemplo 1.4 (Media muestral para una muestra de tamaño 3). Consideremos una muestra aleatoria simple de tamaño $n = 3$ de una población con distribución F y tomemos como parámetro de interés la media poblacional $\theta(F) = \mu = \int x dF(x)$. Tomemos como estadístico de interés $R = R(\mathbf{X}, F) = \bar{X}$. El análogo bootstrap de esta estadística es $R^* = R(\mathbf{X}^*, F_n) = \bar{X}^*$, cuya distribución en el remuestreo se puede calcular de forma exacta debido al reducido número de átomos de probabilidad que tiene. Esta es una distribución discreta con 10 posibles valores, cuyo valor más probable es precisamente \bar{X} que tiene una probabilidad bootstrap de $\frac{2}{9}$, como puede verse en la siguiente tabla.

\mathbf{X}^* (salvo permutaciones)	$\mathbf{N} = (m_1, m_2, m_3)$	$\mathbf{P}^* = (p_1, p_2, p_3)$	$\frac{3!}{m_1!m_2!m_3!3^3}$	\bar{X}^*
(X_1, X_1, X_1)	$(3, 0, 0)$	$(1, 0, 0)$	$\frac{1}{27}$	X_1
(X_2, X_2, X_2)	$(0, 3, 0)$	$(0, 1, 0)$	$\frac{1}{27}$	X_2
(X_3, X_3, X_3)	$(0, 0, 3)$	$(0, 0, 1)$	$\frac{1}{27}$	X_3
(X_1, X_1, X_2)	$(2, 1, 0)$	$(\frac{2}{3}, \frac{1}{3}, 0)$	$\frac{1}{9}$	$\frac{2X_1+X_2}{3}$
(X_1, X_1, X_3)	$(2, 0, 1)$	$(\frac{2}{3}, 0, \frac{1}{3})$	$\frac{1}{9}$	$\frac{2X_1+X_3}{3}$
(X_1, X_2, X_2)	$(1, 2, 0)$	$(\frac{1}{3}, \frac{2}{3}, 0)$	$\frac{1}{9}$	$\frac{X_1+2X_2}{3}$
(X_2, X_2, X_3)	$(0, 2, 1)$	$(0, \frac{2}{3}, \frac{1}{3})$	$\frac{1}{9}$	$\frac{2X_2+X_3}{3}$
(X_1, X_3, X_3)	$(1, 0, 2)$	$(\frac{1}{3}, 0, \frac{2}{3})$	$\frac{1}{9}$	$\frac{X_1+2X_3}{3}$
(X_2, X_3, X_3)	$(0, 1, 2)$	$(0, \frac{1}{3}, \frac{2}{3})$	$\frac{1}{9}$	$\frac{X_2+2X_3}{3}$
(X_1, X_2, X_3)	$(1, 1, 1)$	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$\frac{2}{9}$	$\frac{X_1+X_2+X_3}{3}$

En algunas ocasiones es factible encontrar expresiones cerradas para la distribución de R^* , más allá de las obvias que consisten en enumerar el ingente número de átomos de probabilidad de \mathbf{P}^* :

$$P^*(R^* = R((m_1, \dots, m_n), F_n))$$

Veamos un ejemplo.

1.3.3 Inferencia sobre la mediana

En el caso de la mediana, consideremos, por simplicidad el caso de tamaño muestral impar, $n = 2m - 1$. Supongamos también que no hay empates en los valores de la muestra (si los hubiese las expresiones serían más farragosas pero también calculables). La versión bootstrap del estadístico sobre el cual pivota la inferencia es $R^* = X_{(m)}^* - X_{(m)}$. Su distribución bootstrap podría calcularse si se obtuviese la de $X_{(m)}^*$. Pero ésta es factible de calcular por los pocos posibles valores que puede tomar el estadístico $X_{(m)}^*$ (tan sólo los valores de la muestra original) y por la sencillez del bootstrap uniforme. Veámoslo:

$$P^*(X_{(m)}^* > X_{(j)}) = P^*(\#\{X_i^* \leq X_{(j)}\} \leq m - 1),$$

pero

$$\#\{X_i^* \leq X_{(j)}\} \sim \mathcal{B}\left(n, \frac{j}{n}\right),$$

con lo cual

$$P^*(X_{(m)}^* > X_{(j)}) = \sum_{k=0}^{m-1} \binom{n}{k} \left(\frac{j}{n}\right)^k \left(\frac{n-j}{n}\right)^{n-k}$$

y, por lo tanto, si $j \geq 2$,

$$\begin{aligned} P^*(X_{(m)}^* = X_{(j)}) &= P^*(X_{(m)}^* > X_{(j-1)}) - P^*(X_{(m)}^* > X_{(j)}) \\ &= \sum_{k=0}^{m-1} \binom{n}{k} \left(\frac{j-1}{n}\right)^k \left(\frac{n-j+1}{n}\right)^{n-k} \\ &\quad - \sum_{k=0}^{m-1} \binom{n}{k} \left(\frac{j}{n}\right)^k \left(\frac{n-j}{n}\right)^{n-k} \\ &= \sum_{k=0}^{m-1} \binom{n}{k} \left[\left(\frac{j-1}{n}\right)^k \left(\frac{n-j+1}{n}\right)^{n-k} - \left(\frac{j}{n}\right)^k \left(\frac{n-j}{n}\right)^{n-k} \right]. \end{aligned}$$

Cuando $j = 1$, entonces

$$\begin{aligned} P^* \left(X_{(m)}^* = X_{(1)} \right) &= 1 - P^* \left(X_{(m)}^* > X_{(1)} \right) \\ &= 1 - \sum_{k=0}^{m-1} \binom{n}{k} \left(\frac{1}{n} \right)^k \left(\frac{n-1}{n} \right)^{n-k}. \end{aligned}$$

1.3.4 Distribución Bootstrap aproximada por Monte Carlo

Como ya se comentó anteriormente, al conocer el mecanismo que genera los datos en el bootstrap, siempre se podrá simular dicho mecanismo mediante el método de Monte Carlo. Por lo que el algoritmo general para la aproximación de Monte Carlo del bootstrap uniforme es:

1. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de F_n
2. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, F_n)$
4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Utilizar esas réplicas bootstrap para aproximar la distribución en el muestreo de R

Como se mostró en la Sección 1.1.3, el paso 1 se puede llevar a cabo simulando una distribución uniforme discreta mediante el método de la transformación cuantil:

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$

Aunque en R se recomienda emplear la función `sample`.

Ejemplo 1.5 (Inferencia sobre la media con varianza desconocida). Continuando con el ejemplo de los tiempos de vida de microorganismos, supongamos que queremos obtener una estimación por intervalo de confianza de su vida media a partir de los 15 valores observados pero en la situación mucho más realista de que la varianza sea desconocida.

Tenemos pues $\mathbf{X} = (X_1, \dots, X_n) \sim F$, con μ y σ desconocidas

El parámetro de interés es

$$\theta(F) = \mu = \int x \, dF(x)$$

que se estima mediante

$$\theta(F_n) = \int x \, dF_n(x) = \bar{X}.$$

Así pues, el estadístico en el que basar la inferencia es

$$R = R(\mathbf{X}, F) = \sqrt{n} \frac{\bar{X} - \mu}{S_{n-1}},$$

donde S_{n-1}^2 es la cuasivarianza muestral:

$$S_{n-1}^2 = \frac{1}{n-1} \sum_{j=1}^n (X_j - \bar{X})^2.$$

Bajo normalidad ($X \sim \mathcal{N}(\mu, \sigma^2)$), se sabe que $R \sim t_{n-1}$ y, en particular, $R \xrightarrow{d} \mathcal{N}(0, 1)$ cuando $n \rightarrow \infty$. Si F no es normal entonces la distribución de R ya no es una t_{n-1} , pero también es cierto que, bajo ciertas condiciones, $R \xrightarrow{d} \mathcal{N}(0, 1)$.

En el contexto bootstrap elegimos $\hat{F} = F_n$, con lo cual se trata de un bootstrap naïve o uniforme. El análogo bootstrap del estadístico R será

$$R^* = R(\mathbf{X}^*, F_n) = \sqrt{n} \frac{\bar{X}^* - \bar{X}}{S_{n-1}^*},$$

siendo

$$\begin{aligned}\bar{X}^* &= \frac{1}{n} \sum_{i=1}^n X_i^*, \\ S_{n-1}^{*2} &= \frac{1}{n-1} \sum_{i=1}^n (X_i^* - \bar{X}^*)^2.\end{aligned}$$

El algoritmo bootstrap (aproximado por Monte Carlo) procedería así:

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$
2. Obtener \bar{X}^* y S_{n-1}^{*2}
3. Calcular $R^* = \sqrt{n} \frac{\bar{X}^* - \bar{X}}{S_{n-1}^*}$
4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Aproximar la distribución en el muestreo de R mediante la distribución empírica de $R^{*(1)}, \dots, R^{*(B)}$

El código para implementar este método es similar al del caso de varianza conocida del Ejemplo 1.1:

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
           0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
alfa <- 0.05
x_barra <- mean(muestra)
cuasi_dt <- sd(muestra)

# Remuestreo
set.seed(1)
B <- 1000
remuestra <- numeric(n)
estadistico_boot <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  x_barra_boot <- mean(remuestra)
  cuasi_dt_boot <- sd(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot - x_barra)/cuasi_dt_boot
}

# Aproximación bootstrap de los ptos críticos
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))

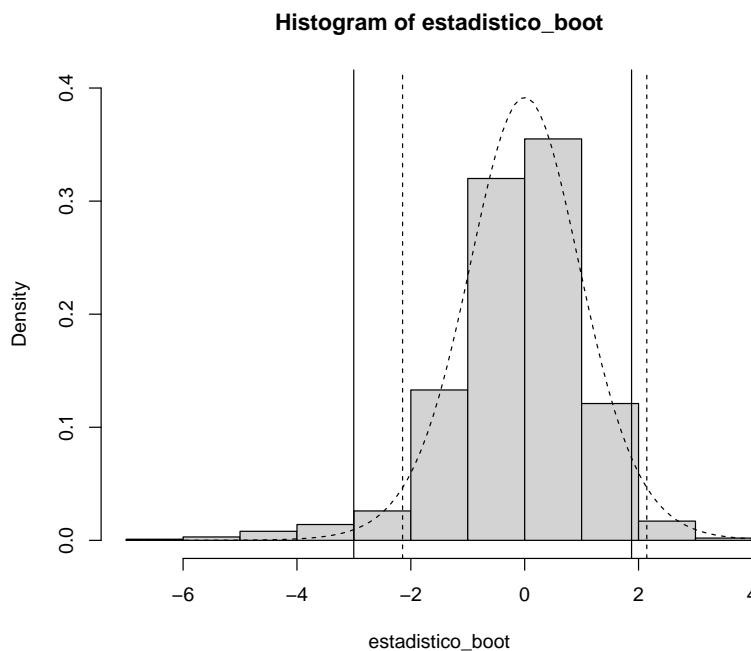
# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2] * cuasi_dt/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1] * cuasi_dt/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.5030131 1.2888063
```

Este procedimiento para la construcción de intervalos de confianza se denomina *método percentil-t* y se tratará en la Sección 4.3.

Como ejemplo adicional podemos comparar la aproximación de la distribución bootstrap del estadístico con la aproximación t_{n-1} basada en normalidad.

```
hist(estadistico_boot, freq = FALSE, ylim = c(0, 0.4))
abline(v = pto_crit)
curve(dt(x, n-1), add=TRUE, lty = 2)
pto_crit_t <- qt(1 - alfa/2, n-1)
abline(v = c(-pto_crit_t, pto_crit_t), lty = 2)
```



En este caso la distribución bootstrap del estadístico es más asimétrica, lo que se traducirá en diferencias entre las estimaciones por intervalos de confianza. Por ejemplo, podemos obtener la estimación basada en normalidad mediante la función `t.test()`:

```
t.test(muestra)$conf.int
```

```
## [1] 0.4599374 1.1507292
## attr(", "conf.level")
## [1] 0.95
```

1.3.5 Elección del número de réplicas Monte Carlo

Normalmente el valor de B se toma del orden de varias centenas o incluso millares. En los casos en los que el bootstrap se utiliza para estimar el sesgo o la varianza de un estimador, bastará tomar un número, B , de réplicas bootstrap del orden de $B = 100, 200, 500$. Sin embargo, cuando se trata de utilizar el bootstrap para realizar contrastes de hipótesis o construir intervalos de confianza son necesarios valores mayores, del tipo $B = 500, 1000, 2000, 5000$.

Evidentemente, la función de distribución del estadístico de interés, $\psi(u) = P(R \leq u)$, se estimaría mediante la distribución empírica de las B realizaciones de la aproximación de Monte Carlo,

$$\hat{\psi}_B(u) = \frac{1}{B} \sum_{i=1}^B \mathbf{1}\{R^{*(i)} \leq u\},$$

de la verdadera distribución bootstrap exacta: $\hat{\psi}(u) = P^*(R^* \leq u)$. El error de MonteCarlo de $\hat{\psi}_B(u)$ con respecto a $\hat{\psi}(u)$ viene dado por su varianza Monte Carlo, pues su sesgo Monte Carlo es cero:

$$\begin{aligned} E^{MC}(\hat{\psi}_B(u)) &= \frac{1}{B} \sum_{i=1}^B E^{MC}(\mathbf{1}\{R^{*(i)} \leq u\}) = \frac{1}{B} \sum_{i=1}^B P^*(R^{*(i)} \leq u) \\ &= \frac{1}{B} \sum_{i=1}^B \hat{\psi}(u) = \hat{\psi}(u), \\ Var^{MC}(\hat{\psi}_B(u)) &= \frac{1}{B^2} \sum_{i=1}^B Var^{MC}(\mathbf{1}\{R^{*(i)} \leq u\}) \\ &= \frac{1}{B^2} \sum_{i=1}^B P^*(R^{*(i)} \leq u)[1 - P^*(R^{*(i)} \leq u)] = \\ &= \frac{1}{B^2} \sum_{i=1}^B \hat{\psi}(u)(1 - \hat{\psi}(u)) = \frac{1}{B} \hat{\psi}(u)(1 - \hat{\psi}(u)) \leq \frac{1}{4B} \end{aligned}$$

Así, el error de la aproximación de Monte Carlo al bootstrap exacto (raíz cuadrada de la varianza del Monte Carlo), puede acotarse por $\frac{1}{2\sqrt{B}}$ (para más detalles sobre la convergencia de una aproximación Monte Carlo ver p.e. el Capítulo 4 de Fernández-Casal y Cao, 2020).

1.4 Herramientas disponibles en R sobre bootstrap

En R hay una gran cantidad de paquetes que implementan métodos bootstrap. Por ejemplo, al ejecutar el comando `??bootstrap` (o `help.search('bootstrap')`) se mostrarán las funciones de los paquetes instalados que incluyen este término en su documentación (se puede realizar la búsqueda en todos los paquetes disponibles de R a través de <https://www.rdocumentation.org>).

De entre todos estas herramientas destacan dos librerías como las más empleadas:

- **bootstrap**: contiene las rutinas (`bootstrap`, `cross-validation`, `jackknife`) y los datos del libro “An Introduction to the Bootstrap” de B. Efron y R. Tibshirani, 1993, Chapman and Hall. La librería fue desarrollada originalmente en S por Rob Tibshirani y exportada a R por Friedrich Leisch. Es útil para desarrollar los ejemplos que se citan en ese libro.
- **boot**: incluye las funciones y conjuntos de datos utilizados en el libro “Bootstrap Methods and Their Applications” de A. C. Davison y D. V. Hinkley, 1997, Cambridge University Press. Esta librería fue desarrollada originalmente en S por Angelo J. Canty y posteriormente exportada a R (ver Canty, 2002). Este paquete es mucho más completo que el paquete **bootstrap**, forma parte de la distribución estándar de R y es el que emplearemos como referencia en este libro (ver Sección 1.4.1).

Por otra parte existen numerosas rutinas (scripts) realizadas en R por diversos autores, que están disponibles en Internet (por ejemplo, puede ser interesante realizar una búsqueda en <https://rseek.org>).

El bootstrap uniforme se puede implementar fácilmente. Por ejemplo, una rutina general para el caso univariante sería la siguiente:

```
#' @param x vector que contiene la muestra.
#' @param B número de réplicas bootstrap.
#' @param statistic función que calcula el estadístico.
boot.strap0 <- function(x, B=1000, statistic=mean){
  ndat <- length(x)
  x.boot <- sample(x, ndat*B, replace=TRUE)
  x.boot <- matrix(x.boot, ncol=B, nrow=ndat)
  stat.boot <- apply(x.boot, 2, statistic)
}
```

Podríamos aplicar esta función a la muestra de tiempos de vida de microorganismos con el siguiente código:

```
fstatistic <- function(x){
  # mean(x)
  # mean(x, trim=0.2)
  median(x)
  # max(x)
}

B <- 1000
set.seed(1)
stat.dat <- fstatistic(muestra)
stat.boot <- boot.bootstrap(muestra, B, fstatistic)

res.boot <- c(stat.dat, mean(stat.boot)-stat.dat, sd(stat.boot))
names(res.boot) <- c("Estadístico", "Sesgo", "Error Std.")
res.boot

## Estadístico      Sesgo  Error Std.
##   0.6110000  0.0609260  0.2481362
```

La función `boot.bootstrap()` anterior no es adecuada para el caso multivariante (por ejemplo cuando estamos interesados en regresión). Como se mostró en la Sección 1.1.3 sería preferible emplear remuestras del vector de índices. Por ejemplo:

```
'# @param datos vector, matriz o data.frame que contiene los datos.
'# @param B número de réplicas bootstrap.
'# @param statistic función con al menos dos parámetros,
'# los datos y el vector de índices de remuestreo,
'# y que devuelve el vector de estadísticos.
'# @param ... parámetros adicionales de la función statistic.
boot.bootstrap <- function(datos, B=1000, statistic, ...) {
  ndat <- NROW(datos)
  i.boot <- sample(ndat, ndat*B, replace=TRUE)
  i.boot <- matrix(i.boot, ncol=B, nrow=ndat)
  stat.boot <- drop(apply(i.boot, 2, function(i) statistic(datos, i, ...)))
}
```

El paquete `boot`, descrito a continuación, emplea una implementación similar.

1.4.1 El paquete `boot`

La función principal de este paquete es la función `boot()` que implementa distintos métodos de remuestreo para datos i.i.d.. En su forma más simple permite realizar bootstrap uniforme (que en la práctica también se denomina habitualmente *bootstrap noparamétrico*):

```
boot(data, statistic, R)
```

donde `data` es un vector, matriz o `data.frame` que contiene los datos, `R` es el número de réplicas bootstrap, y `statistic` es una función con al menos dos parámetros (con las opciones por defecto), los datos y el vector de índices de remuestreo, y que devuelve el vector de estadísticos.

Por ejemplo, para hacer inferencia sobre la mediana del tiempo de vida de microorganismos, podríamos emplear el siguiente código:

```
library(boot)
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
           0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)

statistic <- function(data, i){
```

```
# remuestra <- data[i]; median(remuestra)
median(data[i])
}

set.seed(1)
res.boot <- boot(muestra, statistic, R = 1000)
```

El resultado que devuelve esta función es un objeto de clase `boot`, una lista con los siguientes componentes:

```
names(res.boot)

## [1] "t0"         "t"          "R"          "data"        "seed"        "statistic"
## [7] "sim"        "call"       "stype"      "strata"     "weights"
```

Además de los parámetros de entrada (incluyendo los valores por defecto), contiene tres componentes adicionales:

- `t0`: el valor observado del estadístico (su evaluación en los datos originales).
- `t`: la matriz de réplicas bootstrap del estadístico (cada fila se corresponde con una remuestra).
- `seed`: el valor inicial de la semilla (`.Random.seed`) empleada para la generación de las réplicas.

Este tipo de objetos dispone de dos métodos principales: el método `print()` que muestra un resumen de los resultados (incluyendo aproximaciones bootstrap del sesgo y del error estándar de los estadísticos; ver Capítulo 2):

```
res.boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = muestra, statistic = statistic, R = 1000)
##
##
## Bootstrap Statistics :
##      original    bias   std. error
## t1*    0.611 0.058523  0.2526519
```

y el método `plot()` que genera gráficas básicas de diagnosis de los resultados (correspondientes al estadístico determinado por el parámetro `index`, por defecto = 1): [Figura 1.5]

```
plot(res.boot)
```

Es recomendable examinar la distribución bootstrap del estimador (o estadístico) para detectar posibles problemas. Como en este caso puede ocurrir que el estadístico bootstrap tome pocos valores distintos, lo que indicaría que el número de réplicas bootstrap es insuficiente o que hay algún problema con método de remuestreo empleado (en este caso la distribución objetivo es continua). Se darán más detalles sobre los posibles problemas del bootstrap uniforme en la Sección 3.5.

Además de estos métodos, las principales funciones de interés serían:

- `jack.after.boot()`: genera un gráfico para diagnósticar la influencia de las observaciones individuales en los resultados bootstrap (se representan los cuantiles frente a las diferencias en el estadístico al eliminar una observación; este gráfico también se puede obtener estableciendo `jack = TRUE` en `plot.boot()`).
- `boot.array()`: genera la matriz de índices a partir de la que se obtuvieron las remuestras (permite reconstruir las remuestras bootstrap).

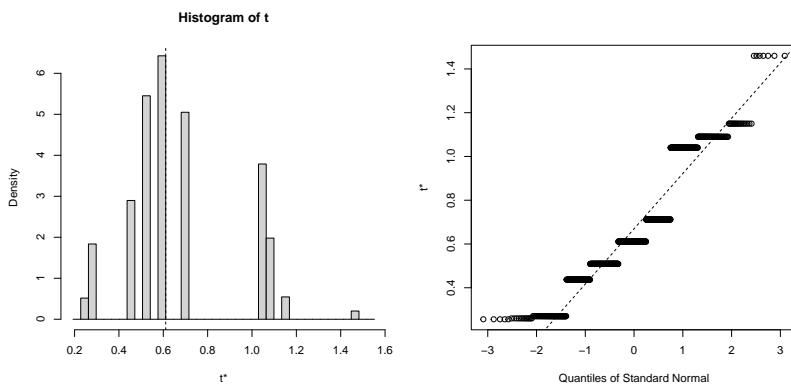


Figura 1.5: Gráficos de diagnóstico de los resultados bootstrap de la mediana de los tiempos de vida de microorganismos.

- `boot.ci()`: construye distintos tipos de intervalos de confianza (se tratarán en el Capítulo 4) dependiendo del parámetro `type`:
 - "`norm`": utiliza la distribución asintótica normal considerando las aproximaciones bootstrap del sesgo y de la varianza.
 - "`basic`": emplea el estadístico $R = \hat{\theta} - \theta$ para la construcción del intervalo de confianza.
 - "`stud`": calcula el intervalo a partir del estadístico estudentizado $R = (\hat{\theta} - \theta) / \sqrt{Var(\hat{\theta})}$.
 - "`perc`": utiliza directamente la distribución bootstrap del estadístico ($R = \hat{\theta}$).
 - "`bca`": emplea el método *BCa* ("bias-corrected and accelerated") propuesto por Efron (1987) (ver Sección 5.3.2 de Davison y Hinkley, 1997).
 - "`all`": calcula los cinco tipos de intervalos anteriores.

Como ya se comentó, la función `boot()` admite estadísticos multivariantes (haciendo que la función `statistic` devuelva un vector en lugar de un escalar), pero por defecto las funciones anteriores consideran el primer componente como el estadístico principal. Para obtener resultados de otros componentes del vector de estadísticos habrá que establecer el parámetro `index` igual al índice deseado. Además, en algunos casos (por ejemplo para la obtención de intervalos de confianza estudiantizados con la función `boot.ci()`) se supone, por defecto, que el segundo componente del vector de estadísticos contiene estimaciones de la varianza del estadístico para cada réplica bootstrap.

Ejemplo 1.6 (Inferencia sobre la media con varianza desconocida, continuación). Continuando con el Ejemplo 1.5 de inferencia sobre la media con varianza desconocida. Para obtener la estimación por intervalo de confianza del tiempo de vida medio de los microorganismos con el paquete `boot`, podríamos emplear el siguiente código:

```
library(boot)
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
           0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)

statistic <- function(data, i){
  remuestra <- data[i]
  c(mean(remuestra), var(remuestra)/length(remuestra))
}

set.seed(1)
```

```

res.boot <- boot(muestra, statistic, R = 1000)
res.boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
## Call:
## boot(data = muestra, statistic = statistic, R = 1000)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.8053333  0.003173267 0.158330646
## t2* 0.0259338 -0.002155755 0.007594682
boot.ci(res.boot)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot)
##
## Intervals :
## Level      Normal          Basic          Studentized
## 95%   ( 0.4918,  1.1125 )  ( 0.4825,  1.0980 )  ( 0.4715,  1.2320 )
##
## Level      Percentile       BCa
## 95%   ( 0.5127,  1.1282 )  ( 0.5384,  1.1543 )
## Calculations and Intervals on Original Scale

```

El intervalo marcado como **Studentized** se obtuvo empleando el mismo estadístico del Ejemplo 1.5.

Modificaciones del bootstrap uniforme

Estableciendo parámetros adicionales de la función `boot` se pueden llevar a cabo modificaciones del bootstrap uniforme. Algunos de estos parámetros son los siguientes:

- **strata**: permite realizar remuestreo estratificado estableciendo este parámetro como un vector numérico o factor que defina los grupos.
- **sim = c("ordinary", "parametric", "balanced", "permutation", "antithetic")**: permite establecer distintos tipos de remuestreo. Por defecto es igual a `"ordinary"` que se corresponde con el bootstrap uniforme, descrito anteriormente. Entre el resto de opciones destacaríamos `sim = "permutation"`, que permite realizar contrastes de permutaciones (remuestreo sin reemplazamiento), y `sim = "parametric"`, que permite realizar bootstrap paramétrico (Sección 3.1). En este último caso también habrá que establecer los parámetros `ran.gen` y `mle`, y la función `statistics` no empleará el segundo parámetro de índices.
- **ran.gen**: función que genera los datos. El primer argumento será el conjunto de datos original y el segundo un vector de parámetros adicionales (normalmente los valores de los parámetros de la distribución).
- **mle**: parámetros de la distribución (típicamente estimados por máxima verosimilitud) o parámetros adicionales para `ran.gen` ó `statistics`.

Además hay otros parámetros para el procesamiento en paralelo: `parallel = c("no", "multicore", "snow")`, `ncpus`, `c1`. En el Apéndice B se incluye una pequeña introducción al procesamiento en paralelo y se muestran algunos ejemplos sobre el uso de estos parámetros. También se puede consultar la ayuda de la función `boot()` (`?boot`).

El paquete `boot` también incluye otras funciones que implementan métodos bootstrap para otros tipos de datos, como la función `censboot()` para datos censurados (Capítulo 8) o la función `tsboot()` para series de tiempo (Capítulo 9).

Finalmente destacar que hay numerosas extensiones implementadas en otros paquetes utilizando el paquete `boot` (ver *Reverse dependencies* en la web de CRAN). Por ejemplo en la Sección 3.7 se ilustrará el uso de la función `Boot()` del paquete `car` para hacer inferencia sobre modelos de regresión.

1.4.2 Ejemplo: Bootstrap uniforme multidimensional

Como ya se mostró en las Secciones 1.1.3 y 1.4 podemos implementar el bootstrap uniforme en el caso multidimensional (denominado también *remuestreo de casos* o *bootstrap de las observaciones*) de modo análogo al unidimensional.

Consideraremos como ejemplo el conjunto de datos `Prestige` del paquete `carData`, y supongamos que queremos realizar inferencias sobre el coeficiente de correlación entre `prestige` (puntuación de ocupaciones obtenidas a partir de una encuesta) `income` (media de ingresos en la ocupación).

```
data(Prestige, package = "carData")
str(Prestige)

## 'data.frame': 102 obs. of 6 variables:
## $ education: num 13.1 12.3 12.8 11.4 14.6 ...
## $ income   : int 12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
## $ women    : num 11.16 4.02 15.7 9.11 11.68 ...
## $ prestige  : num 68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
## $ census   : int 1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
## $ type     : Factor w/ 3 levels "bc","prof","wc": 2 2 2 2 2 2 2 2 2 2 ...
# with(Prestige, cor(income, prestige))
cor(Prestige$income, Prestige$prestige)

## [1] 0.7149057
```

En el siguiente código se emplea el paquete `boot` para realizar bootstrap uniforme multidimensional sobre este estadístico:

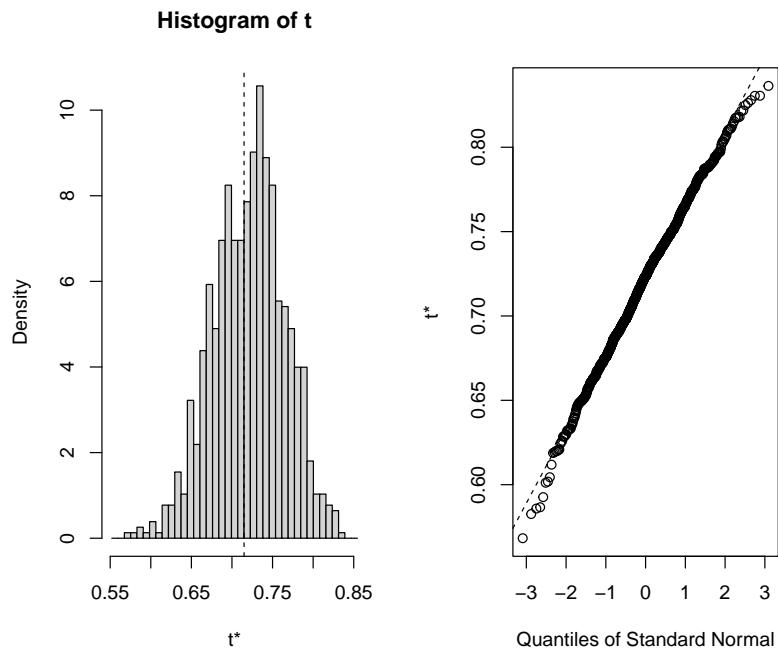
```
library(boot)

statistic <- function(data, i){
  remuestra <- data[i, ]
  cor(remuestra$income, remuestra$prestige)
}

set.seed(1)
B <- 1000
res.boot <- boot(Prestige, statistic, R = B)
res.boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Prestige, statistic = statistic, R = B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.7149057  0.006306905  0.04406473
```

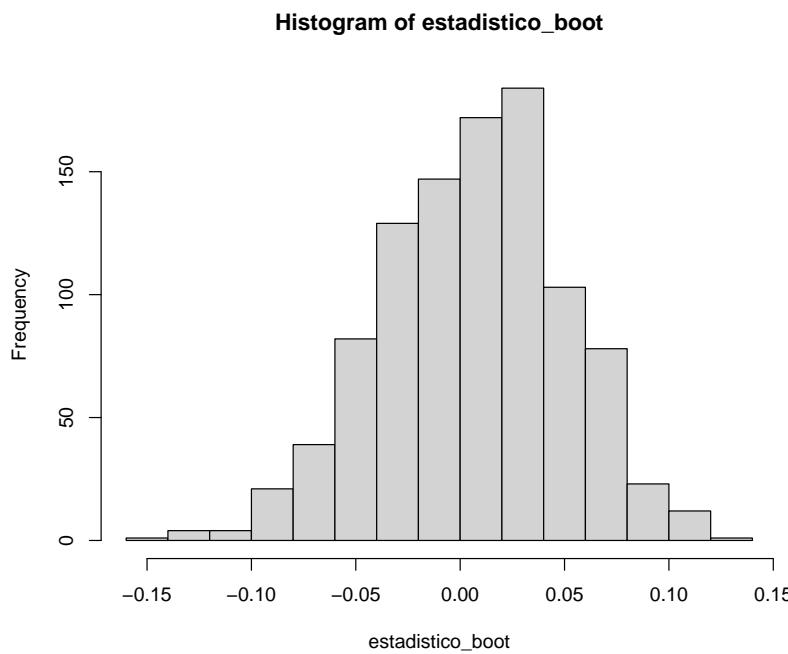
```
plot(res.boot)
```



En este caso podemos observar que la distribución bootstrap del estimador es asimétrica, por lo que asumir que su distribución es normal podría no ser adecuado (por ejemplo para la construcción de intervalos de confianza, que se tratarán en la Sección 4.6.3).

Como comentario final, nótese que en principio el paquete `boot` está diseñado para obtener réplicas bootstrap de un estimador, por lo que si lo que nos interesa es emplear otro estadístico habría que construirlo a partir de ellas (como hacen otras funciones secundarias como `boot.ci()`). Por ejemplo, si queremos emplear el estadístico $R = \hat{\theta} - \theta$ (bootstrap percentil básico o natural), podemos obtener la correspondiente distribución bootstrap (aproximada por Monte Carlo) con el siguiente código:

```
estadistico_boot <- res.boot$t - res.boot$theta
hist(estadistico_boot)
```



A partir de la distribución empírica del estadístico bootstrap $R^* = \hat{\theta}^* - \hat{\theta}$ aproximariamos la característica de interés de la distribución en el muestreo de $R = \hat{\theta} - \theta$. Por ejemplo, para aproximar $\psi(u) = P(R \leq u)$ emplearíamos la frecuencia relativa:

$$\hat{\psi}_B(u) = \frac{1}{B} \sum_{i=1}^B \mathbf{1}\{R^{*(i)} \leq u\}.$$

```
u <- 0
sum(estadistico_boot <= u)/B

## [1] 0.427
# Equivalente:
mean(estadistico_boot <= u)

## [1] 0.427
```

Capítulo 2

Estimación de la precisión y el sesgo de un estimador

Uno de los problemas más interesantes que pueden ser abordados desde la perspectiva de los métodos de remuestreo es el de la estimación del sesgo y la precisión de un estimador. En dicho contexto surgió el método Jackknife (bastante antes que el bootstrap), que, en ese sentido, puede considerarse el método de remuestreo más antiguo como tal.

2.1 Estimación bootstrap de la precisión y el sesgo de un estimador

Consideremos $\mathbf{X} = (X_1, \dots, X_n)$ una m.a.s. de una población con distribución F y supongamos que tenemos interés en realizar inferencia sobre un parámetro de la población $\theta = \theta(F)$. Consideremos un estimador, $\hat{\theta} = T(\mathbf{X})$, de dicho parámetro y definamos el estadístico

$$R = R(\mathbf{X}, F) = T(\mathbf{X}) - \theta(F) = \hat{\theta} - \theta.$$

El sesgo del estimador no es más que la esperanza del estadístico R y la varianza de $\hat{\theta}$ es también la varianza de R (pues θ no es aleatorio). Además, el error cuadrático medio del estimador también se puede escribir como el momento de orden 2 de R :

$$\begin{aligned} \text{Sesgo}(\hat{\theta}) &= E(\hat{\theta} - \theta) = E(R), \\ \text{Var}(\hat{\theta}) &= \text{Var}(\hat{\theta} - \theta) = \text{Var}(R), \\ \text{MSE}(\hat{\theta}) &= E[(\hat{\theta} - \theta)^2] = E(R^2). \end{aligned}$$

Dado que el principio bootstrap es útil para aproximar la distribución en muestreo del estadístico R , entonces también permitirá aproximar sus momentos (su esperanza, su varianza, la esperanza de su cuadrado) y así proceder según sigue:

1. Estimar la función de distribución de probabilidad mediante \hat{F}
2. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de \hat{F} y obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, \hat{F}) = T(\mathbf{X}^*) - \theta(\hat{F}) = \hat{\theta}^* - \theta(\hat{F})$
4. Repetir B veces los pasos 2-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Calcular el estimador bootstrap del sesgo:

$$\text{Sesgo}^*(\hat{\theta}^*) = \bar{R}^* = \frac{1}{B} \sum_{b=1}^B R^{*(b)}.$$

El algoritmo anterior es útil para aproximar por bootstrap el sesgo. Si se desea aproximar la varianza puede sustituirse al paso 5 por:

5. Calcular el estimador bootstrap de la varianza:

$$\text{Var}^*(\hat{\theta}^*) = \frac{1}{B} \sum_{b=1}^B (\bar{R}^{*(b)} - \bar{R}^*)^2$$

Si se trata de aproximar por bootstrap el error cuadrático medio, el paso 5 pasaría a ser:

5. Calcular el estimador bootstrap del error cuadrático medio: $\text{MSE}^*(\hat{\theta}^*) = \frac{1}{B} \sum_{b=1}^B R^{*(b)2}$

En el caso de la varianza podría ahorrarse algunos cálculos definiendo directamente $R = T(\mathbf{X}) = \hat{\theta}$ y, consecuentemente, $R^* = T(\mathbf{X}^*) = \hat{\theta}^*$. Así otro algoritmo de cálculo algo menos intensivo sería:

1. Estimar la función de distribución de probabilidad mediante \hat{F}
2. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de \hat{F} y obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $T(\mathbf{X}^*) = \hat{\theta}^*$
4. Repetir B veces los pasos 2-3 para obtener las réplicas bootstrap $\hat{\theta}^{*(1)}, \dots, \hat{\theta}^{*(B)}$
5. Calcular $\bar{\hat{\theta}}^* = \frac{1}{B} \sum_{b=1}^B \hat{\theta}^{*(b)}$ y, con ello, $\text{Var}^*(\hat{\theta}^*) = \frac{1}{B} \sum_{b=1}^B (\hat{\theta}^{*(b)} - \bar{\hat{\theta}}^*)^2$

Es interesante mencionar que esto permite aproximar por bootstrap (mediante Monte Carlo) la varianza de un estimador sin conocer una expresión explícita para dicha varianza teórica. En general, el estimador, \hat{F} , de F a utilizar en los pasos 1-2 de estos algoritmos se elige según proceda al caso. En el caso del bootstrap uniforme sería $\hat{F} = F_n$ y se puede proceder como se mostró en la Sección 1.1.3.

2.1.1 Ejemplo: la media muestral

Consideremos como parámetro de interés la media de la población, $\theta = \theta(F) = \mu = \int x dF(x)$, y tomemos como estimador la media muestral: $\hat{\theta} = \hat{\mu} = T(\mathbf{X}) = \bar{X} = \frac{1}{n} \sum_{j=1}^n X_j$. Supongamos que deseamos estudiar la varianza de este estimador: $\text{Var}(\hat{\theta}) = \text{Var}(\bar{X}) = \frac{\sigma^2}{n}$.

A la hora de aproximar por bootstrap $\text{Var}(\hat{\theta})$, si no disponemos de ninguna otra información adicional (como que la distribución sea de cierta familia paramétrica o que sea continua), parece razonable elegir como método de remuestreo el bootstrap uniforme. En tal caso el algoritmo bootstrap de Monte Carlo procedería de esta forma:

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$
2. Calcular $T(\mathbf{X}^*) = \bar{X}^* = \frac{1}{n} \sum_{i=1}^n X_i^*$
3. Repetir B veces los pasos 1-2 para obtener las réplicas bootstrap $\bar{X}^{*(1)}, \dots, \bar{X}^{*(B)}$
4. Calcular $\bar{\bar{X}}^* = \frac{1}{B} \sum_{b=1}^B \bar{X}^{*(b)}$ y $\text{Var}^*(\bar{X}^*) = \frac{1}{B} \sum_{b=1}^B (\bar{X}^{*(b)} - \bar{\bar{X}}^*)^2$

De todas formas, en este caso puede verse fácilmente que no es necesario realizar Monte Carlo. En efecto,

$$\begin{aligned} \text{Var}^*(\bar{X}^*) &= \frac{1}{n^2} \sum_{i=1}^n \text{Var}^*(X_i^*) = \frac{1}{n} \text{Var}^*(X_1^*) \\ &= \frac{1}{n} \left\{ E^*(X_1^{*2}) - [E^*(X_1^*)]^2 \right\} = \frac{1}{n} \left[\frac{1}{n} \sum_{j=1}^n X_j^2 - \left(\frac{1}{n} \sum_{j=1}^n X_j \right)^2 \right] \\ &= \frac{1}{n^2} \sum_{j=1}^n (X_j - \bar{X})^2 = \frac{S_n^2}{n}, \end{aligned}$$

que es precisamente el estimador plug-in de la varianza de la media muestral.

Ejemplo 2.1 (Aproximación bootstrap de la precisión de estimaciones del tiempo de vida medio de microorganismos). Continuando con el ejemplo de los tiempos de vida de microorganismos, supongamos que queremos estimar la precisión de dos estimadores de su vida media: media muestral y mediana muestral, a partir de los datos observados: 0.143, 0.182, 0.256, 0.260, 0.270, 0.437, 0.509, 0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08.

La estimación media muestral resulta $\bar{X} = 0.8053333$. Por su parte la estimación mediana muestral es $x_{(8)} = 0.611$.

La varianza del estimador media muestral, \bar{X} es $Var(\bar{X}) = \frac{\sigma^2}{n}$, desconocida en este caso. Su estimación bootstrap (idéntica a la plug-in) mediante un remuestreo uniforme es calculable sin necesidad de realizar Monte Carlo y resulta:

$$Var^*(\bar{X}^*) = \frac{1}{n^2} \sum_{j=1}^n (x_j - \bar{X})^2 = 0.024204877.$$

Con lo cual $\sqrt{Var^*(\bar{X}^*)} = \sqrt{0.024204877} = 0.15558$

Si consideramos ahora la mediana muestral (como estimador de la media), también sabemos que su distribución bootstrap puede calcularse de forma explícita, sin necesidad de realizar Monte Carlo. Su masa de probabilidad viene dada por:

$$\begin{aligned} P^*(X_{(8)}^* = x_{(1)}) &= 1 - \sum_{k=0}^{m-1} \binom{n}{k} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{n-k} \\ P^*(X_{(8)}^* = x_{(j)}) &= \sum_{k=0}^{m-1} \binom{n}{k} \left[\left(\frac{j-1}{n}\right)^k \left(\frac{n-j+1}{n}\right)^{n-k} - \left(\frac{j}{n}\right)^k \left(\frac{n-j}{n}\right)^{n-k} \right] \\ &\text{para } j = 2, \dots, n \end{aligned}$$

Con los datos concretos del ejemplo resulta:

$$\begin{aligned} P^*(X_{(8)}^* = 0.143) &= 1.639 \times 10^{-6}, & P^*(X_{(8)}^* = 0.182) &= 2.655 \times 10^{-4}, \\ P^*(X_{(8)}^* = 0.256) &= 3.973 \times 10^{-3}, & P^*(X_{(8)}^* = 0.260) &= 2.121 \times 10^{-2}, \\ P^*(X_{(8)}^* = 0.270) &= 6.278 \times 10^{-2}, & P^*(X_{(8)}^* = 0.437) &= 0.1249, \\ P^*(X_{(8)}^* = 0.509) &= 0.1832, & P^*(X_{(8)}^* = 0.611) &= 0.2073, \\ P^*(X_{(8)}^* = 0.712) &= 0.1832, & P^*(X_{(8)}^* = 1.04) &= 0.1249, \\ P^*(X_{(8)}^* = 1.09) &= 6.278 \times 10^{-2}, & P^*(X_{(8)}^* = 1.15) &= 2.121 \times 10^{-2}, \\ P^*(X_{(8)}^* = 1.46) &= 3.973 \times 10^{-3}, & P^*(X_{(8)}^* = 1.88) &= 2.655 \times 10^{-4}, \\ P^*(X_{(8)}^* = 2.08) &= 1.639 \times 10^{-6}. \end{aligned}$$

Como consecuencia,

$$\begin{aligned} E^*(X_{(8)}^*) &= \sum_{j=1}^{15} x_{(j)} P^*(X_{(8)}^* = x_{(j)}) = 0.65749924 \\ E^*(X_{(8)}^{*2}) &= \sum_{j=1}^{15} x_{(j)}^2 P^*(X_{(8)}^* = x_{(j)}) = 0.49500381 \\ Var^*(X_{(8)}^*) &= 0.49500381 - 0.65749924^2 = 6.2699 \times 10^{-2} \\ \sqrt{Var^*(X_{(8)}^*)} &= \sqrt{6.2699 \times 10^{-2}} = 0.25040 \end{aligned}$$

Las estimaciones bootstrap de los errores cuadráticos medios de ambos estimadores (como estimadores

de la media poblacional) son:

$$\begin{aligned} MSE^*(\bar{X}^*) &= (E^*(\bar{X}^*) - \bar{X})^2 + Var^*(\bar{X}^*) \\ &= Var^*(\bar{X}^*) = 0.024204877, \\ MSE^*(X_{(8)}^*) &= (E^*(X_{(8)}^*) - \bar{X})^2 + Var^*(X_{(8)}^*) = \\ &= (0.65749924 - 0.8053333)^2 + 6.2699 \times 10^{-2} \\ &= 0.084554. \end{aligned}$$

Una aproximación de Monte Carlo de estas varianzas bootstrap se puede llevar a cabo mediante el siguiente código:

```
# Para la muestra de TIEMPOS DE VIDA, estima (plug-in) la precisión
# de la media muestral (desumedia) y también approxima por Monte Carlo
# la estimación bootstrap de dicha precisión (desumediaboot) y
# también la precisión de la mediana muestral, de la cual no se
# conoce su expresión, (desumedianaboot).
# También estima el sesgo bootstrap de esos dos estimadores
# (sesgomediaboot y sesgomedianaboot, respectivamente).

muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
            0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
varmedia <- (1/(n^2)) * sum((muestra - mean(muestra))^2)
# Alternativamente: varmedia <- var(muestra)/n
desvmedia <- sqrt(varmedia)

# Remuestreo
B <- 1e+04
media <- numeric(B)
mediana <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  media[k] <- mean(remuestra)
  # remordenada <- sort(remuestra)
  # mediana[k] <- remordenada[8]
  mediana[k] <- median(remuestra)
}

# Aproximaciones precisión
varmediaboot <- (1/B) * sum((media - mean(media))^2)
desvmediaboot <- sqrt(varmediaboot)
varmedianaboot <- (1/B) * sum((mediana - mean(mediana))^2)
desvmedianaboot <- sqrt(varmedianaboot)
desvmedia

## [1] 0.1555792
desvmediaboot

## [1] 0.1572797
desvmedianaboot

## [1] 0.2518982
# Aproximaciones sesgo
sesgomediaboot <- mean(media) - mean(muestra)
```

```
sesgomedianaboot <- mean(mediana) - muestra[8]
sesgomediaboot
```

```
## [1] -0.001110327
sesgomedianaboot
```

```
## [1] 0.0449271
```

Empleando el paquete `boot` el código sería más simple:

```
library(boot)
statistic <- function(data, i){
  remuestra <- data[i]
  c(mean(remuestra), median(remuestra))
}

set.seed(1)
res.boot <- boot(muestra, statistic, R = B)
res.boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = muestra, statistic = statistic, R = B)
##
##
## Bootstrap Statistics :
##      original     bias   std. error
## t1* 0.8053333 7.115333e-05  0.1572396
## t2* 0.6110000 4.529410e-02  0.2511022
```

Lamentablemente la función `print.boot()` calcula las aproximaciones bootstrap del sesgo y de la precisión pero no las almacena. En el caso más simple podríamos obtenerlas con el siguiente código:

```
op <- with(res.boot, cbind(
  t0, apply(t, 2, mean, na.rm = TRUE) - t0,
  apply(t, 2, sd, na.rm = TRUE)
))
rownames(op) <- paste0("t", 1:ncol(res.boot$t), "*")
colnames(op) <- c("original", "bias ", " std. error")
op

##      original     bias   std. error
## t1* 0.8053333 7.115333e-05  0.1572396
## t2* 0.6110000 4.529410e-02  0.2511022
```

Como comentario adicional, si se emplea \hat{F} como aproximación de la distribución poblacional, lo habitual es que $\hat{\theta} = T(\mathbf{X}) = \theta(\hat{F})$ sea el estimador de $\theta(F)$ (por ejemplo $\hat{\theta} = \theta(F_n)$ en el caso del bootstrap uniforme). De esta forma, en el universo bootstrap el parámetro teórico se sustituye por el estimador y, por ejemplo, $R = \hat{\theta} - \theta$ se traduce en $R^* = \hat{\theta}^* - \hat{\theta}$. Este es el principal motivo por el que la mayoría de las herramientas utilizadas en la práctica siempre asumen que ocurre esto, como es el caso de los métodos implementados en el paquete `boot` (y derivados).

En el caso general $\hat{\theta} = T(\mathbf{X})$ puede ser distinto de $\theta(\hat{F})$ (por ejemplo, en el caso del bootstrap uniforme puede haber estimadores mejores que $\theta(F_n)$ y típicamente se suele considerar el mejor estimador disponible). Como se mostró al principio de la Sección 2.1, en ese caso se debería sustituir en el universo bootstrap el parámetro teórico por $\theta(\hat{F})$ y, por ejemplo, $R = \hat{\theta} - \theta$ se traduciría en $R^* = \hat{\theta}^* - \theta(\hat{F})$ (esto

puede ser especialmente importante en la aproximación del sesgo). Suponiendo que podemos obtener $\theta(\hat{F})$, podemos desarrollar un código que implemente este algoritmo sin mucha dificultad (se podría tomar como base la primera aproximación en el ejemplo anterior), pero no se puede hacer de forma directa con el paquete **boot** (que asume siempre que $\hat{\theta} = \theta(\hat{F})$). Habría que rehacer los cálculos que implementan las herramientas de este paquete (empleando directamente `res.boot$t` y, por ejemplo, cambiando `t0` en la aproximación del sesgo en el código anterior o procediendo de forma similar a como se hace en la Sección 6.6.3, donde la estimación del sesgo se podría considerar importante).

Finalmente, especialmente en el caso de métodos bootstrap más avanzados, puede resultar difícil obtener $\theta(\hat{F})$ y la aproximación habitual es reemplazarlo directamente por $\hat{\theta}$. Esta es una justificación más de la implementación que se suele hacer en la práctica.

2.2 Motivación del método Jackknife

El jackknife es probablemente el método de remuestreo, propiamente dicho, más antiguo. Fue propuesto por Quenouille (1949) para estimar el sesgo de un estimador. Tukey (1958) bautiza el método y lo utiliza para estimar la varianza de un estimador. En realidad el jackknife no suele utilizarse para aproximar la distribución de $R(\mathbf{X}, F)$, sino más bien para estimar características de dicha variable aleatoria, como su esperanza o su varianza.

La diferencia entre el bootstrap y el jackknife es muy fácil de expresar en términos de los vectores de remuestreo. Así, el bootstrap uniforme utiliza vectores de remuestreo de la forma $\mathbf{P}^* = (\frac{m_1}{n}, \dots, \frac{m_n}{n})$, con $m_i \in \mathbb{Z}^+, i = 1, \dots, n$, mientras que el jackknife considera vectores de remuestreo de la forma

$$\mathbf{P}_{(i)}^* = \left(\frac{1}{n-1}, \dots, 0, \underset{(i)}{1}, \dots, \frac{1}{n-1} \right).$$

En otras palabras todas las remuestras jackknife posibles son tantas como el tamaño muestral y cada una consiste en eliminar una observación de la muestra, quedándose con una remuestra de tamaño $n - 1$ en la que las demás observaciones aparecen exactamente con frecuencia 1.

Evidentemente, el número de posibles remuestras jackknife, n , es muchísimo más pequeño que el número de remuestras bootstrap, $\binom{2n-1}{n}$, lo cual permite calcular con rapidez las realizaciones del estadístico de interés en todas las posibles remuestras jackknife.

2.3 Estimación Jackknife de la precisión y el sesgo de un estimador

Cuando estamos interesados en el sesgo o la varianza de un estimador $\hat{\theta} = \theta(\mathbf{X})$ de un parámetro $\theta = \theta(F)$, el estadístico de interés suele definirse como $R = R(\mathbf{X}, F) = \hat{\theta} - \theta$. En este caso

$$\begin{aligned} \text{Sesgo}(\hat{\theta}) &= E(\hat{\theta}) - \theta = E(R), \\ \text{Var}(\hat{\theta}) &= \text{Var}(\hat{\theta} - \theta) = \text{Var}(R). \end{aligned}$$

Así pues trataremos de usar el jackknife para aproximar la esperanza y varianza de R , o, equivalentemente, el sesgo y la varianza de $\hat{\theta}$.

El conjunto de remuestras jackknife es

$$\mathcal{X}_{jackk} = \left\{ \mathbf{X}^* = \mathbf{X}_{(i)} = (X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n) : i = 1, \dots, n \right\}$$

y todas ellas se consideran con equiprobabilidad en el universo jackknife. Como primera tentativa estimaríamos el sesgo y la varianza jackknife mediante:

$$\begin{aligned} E^*(R^*) &= E_{jackk}^*(\hat{\theta}^*) - \theta(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^n \theta(\mathbf{X}_{(i)}) - \hat{\theta} = \overline{\theta(\mathbf{X}_{(.)})} - \hat{\theta}, \\ \text{Var}^*(R^*) &= \text{Var}_{jackk}^*(\hat{\theta}^*) = \frac{1}{n} \sum_{i=1}^n \left[\theta(\mathbf{X}_{(i)}) - \overline{\theta(\mathbf{X}_{(.)})} \right]^2, \end{aligned}$$

con $\overline{\theta(\mathbf{X}_{(\cdot)})} = \frac{1}{n} \sum_{j=1}^n \theta(\mathbf{X}_{(j)})$.

Sin embargo es evidente que las réplicas jackknife son mucho más parecidas a la muestra original de lo que lo son las remuestras bootstrap, en general. De hecho se puede demostrar que el valor absoluto de ese estimador jackknife del sesgo es siempre menor que el valor absoluto del sesgo bootstrap y que la estimación jackknife de la varianza que se acaba de proponer también es menor que la varianza bootstrap. En resumen, el método jackknife necesita de un **factor de elevación** para que las estimaciones que proporciona sean consistentes. La idea es elegir dicho factor de elevación como aquel que provoca que, al multiplicar los estadísticos anteriores por él, y considerando como parámetro a estimar la media o la varianza poblacional, el estimador jackknife finalmente resultante sea insesgado. Así, el factor de elevación resulta ser $n - 1$ y las estimaciones jackknife finales son

$$\begin{aligned} Sesgo_{jackk}^*(\hat{\theta}^*) &= (n-1) \left(\overline{\theta(\mathbf{X}_{(\cdot)})} - \hat{\theta} \right) = \frac{n-1}{n} \sum_{i=1}^n (\theta(\mathbf{X}_{(i)}) - \hat{\theta}), \\ Var_{jackk}^*(\hat{\theta}^*) &= \frac{n-1}{n} \sum_{i=1}^n [\theta(\mathbf{X}_{(i)}) - \overline{\theta(\mathbf{X}_{(\cdot)})}]^2. \end{aligned}$$

Tomando como parámetro de interés la media, $\theta = \mu$, tenemos que

$$\begin{aligned} \overline{\theta(\mathbf{X}_{(\cdot)})} - \hat{\theta} &= \frac{1}{n} \sum_{i=1}^n \theta(\mathbf{X}_{(i)}) - \hat{\theta} = \frac{1}{n} \sum_{i=1}^n \overline{X_{(i)}} - \bar{X} = \\ &= \frac{1}{n} \sum_{i=1}^n \frac{1}{n-1} \sum_{j=1, j \neq i}^n X_j - \bar{X} = \frac{1}{n(n-1)} \sum_{i,j=1, i \neq j}^n X_j - \bar{X} \\ &= \frac{1}{n(n-1)} \sum_{j=1}^n (n-1) X_j - \bar{X} = \frac{1}{n} \sum_{j=1}^n X_j - \bar{X} = 0, \end{aligned}$$

así que $\overline{\theta(\mathbf{X}_{(\cdot)})} - \hat{\theta}$ es un estimador insesgado del sesgo de \bar{X} (que es cero). De esta forma, utilizando un factor de elevación arbitrario, c , se tiene igualmente que

$$c \left(\overline{\theta(\mathbf{X}_{(\cdot)})} - \hat{\theta} \right) = 0,$$

así que es también un estimador insesgado de $Sesgo(\bar{X}) = 0$. Determinaremos el valor de c imponiendo que el estimador jackknife de la varianza de dicho estimador ($\hat{\theta} = \bar{X}$) es una estimador insesgado de la varianza de dicho estimador. Por una parte, es bien conocido que la varianza de $\hat{\theta}$ es $Var(\bar{X}) = \sigma^2/n$. Por otra parte, la estimación jackknife de la varianza de $\hat{\theta}$, con factor de elevación c es

$$\begin{aligned} Var_{jackk}^*(\bar{X}) &= \frac{c}{n} \sum_{i=1}^n [\overline{X_{(i)}} - \overline{\overline{X_{(\cdot)}}}]^2 \\ &= \frac{c}{n} \sum_{i=1}^n \left[\frac{1}{n-1} \sum_{j=1, j \neq i}^n X_j - \frac{1}{n} \sum_{k=1}^n \frac{1}{n-1} \sum_{j=1, j \neq k}^n X_j \right]^2 \\ &= \frac{c}{n} \sum_{i=1}^n \left[\frac{1}{n-1} \sum_{j=1, j \neq i}^n X_j - \frac{1}{n(n-1)} \sum_{k,j=1, j \neq k}^n X_j \right]^2 \\ &= \frac{c}{n} \sum_{i=1}^n \left[\frac{1}{n(n-1)} \sum_{j=1, j \neq i}^n X_j - \frac{1}{n} X_i \right]^2. \end{aligned}$$

La esperanza de esta cantidad resulta:

$$\begin{aligned}
& E \left[\frac{c}{n} \sum_{i=1}^n \left(\frac{1}{n(n-1)} \sum_{j=1, j \neq i}^n X_j - \frac{1}{n} X_i \right)^2 \right] \\
& = \frac{c}{n} \sum_{i=1}^n E \left[\left(\frac{1}{n(n-1)} \sum_{j=1, j \neq i}^n X_j - \frac{1}{n} X_i \right)^2 \right] \\
& = \frac{c}{n} \sum_{i=1}^n E \left[\left(\frac{1}{n(n-1)} \sum_{j=1, j \neq i}^n (X_j - \mu) - \frac{1}{n} (X_i - \mu) \right)^2 \right] \\
& = \frac{c}{n} \sum_{i=1}^n Var \left[\frac{1}{n(n-1)} \sum_{j=1, j \neq i}^n (X_j - \mu) - \frac{1}{n} (X_i - \mu) \right] \\
& = \frac{c}{n} \sum_{i=1}^n \left(\frac{1}{n^2(n-1)^2} \sum_{j=1, j \neq i}^n \sigma^2 + \frac{1}{n^2} \sigma^2 \right) \\
& = \frac{c}{n} \sum_{i=1}^n \left(\frac{1}{n^2(n-1)} \sigma^2 + \frac{1}{n^2} \sigma^2 \right) = \frac{c\sigma^2}{n(n-1)}.
\end{aligned}$$

Así pues, el sesgo del estimador jackknife de la varianza de la media muestral es

$$E [Var_{jackk}^*(\bar{X})] - \frac{\sigma^2}{n} = \frac{c\sigma^2}{n(n-1)} - \frac{\sigma^2}{n} = \frac{\sigma^2}{n} \left(\frac{c}{n-1} - 1 \right),$$

que vale cero si y solamente si $c = n - 1$. Dicho en otras palabras, tomando como factor de elevación $c = n - 1$, entonces, tanto el estimador jackknife del sesgo de \bar{X} como el estimador jackknife de la varianza de \bar{X} son estimadores insesgados, respectivamente, del sesgo y la varianza de \bar{X} .

Dichos estimadores resultan

$$\begin{aligned}
Sesgo_{jackk}^*(\bar{X}) &= (n-1) (\overline{\bar{X}_{(\cdot)}} - \bar{X}), \\
Var_{jackk}^*(\bar{X}) &= \frac{n-1}{n} \sum_{i=1}^n [\bar{X}_{(i)} - \overline{\bar{X}_{(\cdot)}}]^2.
\end{aligned}$$

Podemos realizar un razonamiento análogo cuando el parámetro de interés es la varianza poblacional, $\theta = \sigma^2$. En ese caso, considerando el estimador varianza muestral: $\hat{\theta} = S_n^2$, se tiene que su esperanza viene dada por

$$\begin{aligned}
E(S_n^2) &= E \left[\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2 \right] = \frac{1}{n} \sum_{i=1}^n E[(X_i - \bar{X})^2] \\
&= E \left[\left(X_1 - \frac{1}{n} \sum_{j=1}^n X_j \right)^2 \right] = E \left[\left((X_1 - \mu) - \frac{1}{n} \sum_{j=1}^n (X_j - \mu) \right)^2 \right] \\
&= Var \left[(X_1 - \mu) - \frac{1}{n} \sum_{j=1}^n (X_j - \mu) \right] \\
&= Var \left[\frac{n-1}{n} (X_1 - \mu) - \frac{1}{n} \sum_{j=1, j \neq 1}^n (X_j - \mu) \right] \\
&= \left(\frac{n-1}{n} \right)^2 \sigma^2 + \frac{1}{n^2} \sum_{j=2}^n \sigma^2 \\
&= \frac{(n-1)^2}{n^2} \sigma^2 + \frac{n-1}{n^2} \sigma^2 = \frac{n(n-1)}{n^2} \sigma^2 = \frac{n-1}{n} \sigma^2,
\end{aligned}$$

así que su sesgo es

$$\text{Sesgo}(S_n^2) = E(S_n^2) - \sigma^2 = -\frac{1}{n}\sigma^2.$$

Para un factor de elevación, c , el estimador jackknife del sesgo de este estimador es

$$\text{Sesgo}_{jackk}^*(S_n^2) = c \left(\overline{\theta(\mathbf{X}_{(\cdot)})} - \hat{\theta} \right) = c \left(\overline{S_{n,(\cdot)}^2} - S_n^2 \right).$$

Con lo cual la esperanza de este estimador resulta

$$E \left(c \left(\overline{S_{n,(\cdot)}^2} - S_n^2 \right) \right) = c \left[E \left(\overline{S_{n,(\cdot)}^2} \right) - E(S_n^2) \right]$$

Estudiemos por separado cada término:

$$\begin{aligned} \overline{S_{n,(\cdot)}^2} &= \frac{1}{n} \sum_{i=1}^n S_{n,(i)}^2 = \frac{1}{n} \sum_{i=1}^n \frac{1}{n-1} \sum_{j=1, j \neq i}^n (X_j - \overline{X}_{(i)})^2 \\ &= \frac{1}{n} \sum_{i=1}^n \frac{1}{n-1} \sum_{j=1, j \neq i}^n \left(X_j - \frac{1}{n-1} \sum_{k=1, k \neq i}^n X_k \right)^2 \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \left(\frac{n-2}{n-1} X_j - \frac{1}{n-1} \sum_{k=1, k \neq i, k \neq j}^n X_k \right)^2, \end{aligned}$$

con lo cual

$$\begin{aligned} E \left(\overline{S_{n,(\cdot)}^2} \right) &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n E \left[\left(\frac{n-2}{n-1} X_j - \frac{1}{n-1} \sum_{k=1, k \neq i, k \neq j}^n X_k \right)^2 \right] \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n E \left[\left(\frac{n-2}{n-1} (X_j - \mu) - \frac{1}{n-1} \sum_{k=1, k \neq i, k \neq j}^n (X_k - \mu) \right)^2 \right] \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \text{Var} \left[\frac{n-2}{n-1} (X_j - \mu) - \frac{1}{n-1} \sum_{k=1, k \neq i, k \neq j}^n (X_k - \mu) \right] \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \left[\frac{(n-2)^2}{(n-1)^2} \sigma^2 + \frac{1}{(n-1)^2} (n-2) \sigma^2 \right] \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j=1, j \neq i}^n \left[\frac{(n-2)(n-1)}{(n-1)^2} \sigma^2 \right] = \frac{n-2}{n-1} \sigma^2 \end{aligned}$$

Además, ya hemos visto anteriormente que $E(S_n^2) = (n-1)\sigma^2/n$, con lo cual la esperanza del estimador jackknife del sesgo es

$$\begin{aligned} c \left(\frac{n-2}{n-1} \sigma^2 - \frac{n-1}{n} \sigma^2 \right) &= c\sigma^2 \left(\frac{n-2}{n-1} - \frac{n-1}{n} \right) \\ &= c\sigma^2 \frac{n(n-2) - (n-1)^2}{n(n-1)} \\ &= c\sigma^2 \frac{-1}{n(n-1)} = -\frac{c\sigma^2}{n(n-1)}. \end{aligned}$$

De esta forma, el sesgo del estimador jackknife del sesgo de S_n^2 resulta ser

$$\begin{aligned} E \left[\text{Sesgo}_{jackk}^*(S_n^2) \right] - \text{Sesgo}(S_n^2) &= -\frac{c\sigma^2}{n(n-1)} - \left(-\frac{1}{n}\sigma^2 \right) \\ &= -\frac{\sigma^2}{n} \left(\frac{c}{n-1} - 1 \right), \end{aligned}$$

con lo cual este sesgo será cero si y sólo si $c = n - 1$.

Esto da pie al estimador jackknife del sesgo de la varianza muestral:

$$Sesgo_{jackk}^*(S_n^2) = (n - 1) \left(\overline{S_{n,(.)}^2} - S_n^2 \right).$$

Así pues, queda justificado, en el caso de estimación de los parámetros media y varianza, la razón de la elección del factor de elevación $c = n - 1$.

2.4 Relación Bootstrap/Jackknife en dicha estimación

Consideremos un parámetro de interés $\theta(F)$ y su correspondiente estimador que supondremos funcional, $\theta(F_n)$. En realidad, cuando calculamos cantidades como $\theta(\mathbf{X}_{(i)})$, lo que estamos haciendo es evaluar el funcional θ en otra función de distribución

$$F_{n,(i)}(x) = \frac{1}{n-1} \sum_{j=1, j \neq i}^n \mathbf{1}(X_j \leq x).$$

Dicho en terminología de vectores de remuestreo, el estimador habitual consiste en evaluar θ en el vector $\mathbf{p} = (\frac{1}{n}, \dots, \frac{1}{n})$, $\theta(\mathbf{p})$, mientras que el estimador construido con toda la muestra excepto el dato i -ésimo es la evaluación $\theta(\mathbf{p}_{(i)})$, siendo

$$\mathbf{p}_{(i)} = \left(\frac{1}{n-1}, \dots, \frac{1}{n-1}, 0, \frac{1}{n-1}, \dots, \frac{1}{n-1} \right).$$

En lo que sigue, consideraremos funcionales θ que definen estimadores lineales o cuadráticos en los vectores de remuestreo:

- Estimadores lineales:

$$\theta(\mathbf{p}) = a + \mathbf{b}^T \mathbf{p}$$

- Estimadores cuadráticos:

$$\theta(\mathbf{p}) = a + \mathbf{b}^T \mathbf{p} + \mathbf{p}^T C \mathbf{p}$$

Una forma alternativa de definir estos estimadores es

- Estimadores lineales:

$$\theta(\mathbf{p}) = \mathbf{b}^T (\mathbf{p} - \mathbf{p}_0)$$

- Estimadores cuadráticos:

$$\theta(\mathbf{p}) = (\mathbf{p} - \mathbf{p}_0)^T C (\mathbf{p} - \mathbf{p}_0)$$

Por ejemplo, puede demostrarse fácilmente que la media, \bar{X} , es un estimador lineal en el vector de remuestreo y que la varianza muestral, S_n^2 , es un estimador cuadrático en el vector de remuestreo.

Existen dos resultados que relacionan el sesgo bootstrap y el sesgo jackknife de cualquier estimador cuadrático y la varianza bootstrap y la varianza jackknife de cualquier estimador lineal.

Teorema 2.1

Si $\hat{\theta}$ es un estimador cuadrático, entonces

$$Sesgo_{jackk}(\hat{\theta}) = \frac{n}{n-1} Sesgo_{boot}(\hat{\theta})$$

Teorema 2.2

Si $\hat{\theta}$ es un estimador lineal, entonces

$$Var_{jackk}(\hat{\theta}) = \frac{n}{n-1} Var_{boot}(\hat{\theta})$$

Dicho en otras palabras, para cualquier estimador cuadrático, el sesgo jackknife es mayor que el sesgo bootstrap. Si el estimador es lineal, la varianza jackknife es mayor que la varianza bootstrap. En ambos casos, el factor multiplicador es $n/(n - 1)$.

Ejemplo 2.2 (Aproximación jackknife de la precisión de estimaciones del tiempo de vida medio de microorganismos). Consideremos la muestra de tiempos de vida de microorganismos ya tratada. El siguiente código permite calcular los estimadores jackknife del sesgo y de la precisión tanto de la media como de la mediana muestral.

```
# Para la muestra de TIEMPOS DE VIDA, estima (plug-in) la precisión
# de la media muestral (desvmedia) y también estima mediante el jackknife
# dicha precisión (desvmediajackk) y también la precisión de la mediana
# muestral, de la cual no se conoce su expresión, (desvmedianajackk).
# También estima el sesgo jackknife de esos dos estimadores
# (sesgomediajackk y sesgomedianajackk, respectivamente).

muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
            0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
varmedia <- (1/(n^2)) * sum((muestra - mean(muestra))^2)
desvmedia <- sqrt(varmedia)

# Jackknife
media <- numeric(n)
mediana <- numeric(n)
for (i in 1:n) {
    imuestra <- muestra[-i]
    media[i] <- mean(imuestra)
    # remordenada <- sort(imuestra)
    # mediana[i] <- (remordenada[7] + remordenada[8])/2
    mediana[i] <- median(imuestra)
}
## [1] 0.1555792
desvmediajackk

## [1] 0.1610397
desvmedianajackk

## [1] 0.1834505
# Aproximaciones sesgo
sesgomediajackk <- (n - 1) * (mean(media) - mean(muestra))
# sesgomedianajackk <- (n - 1) * (mean(mediana) - muestra[8])
sesgomedianajackk <- (n - 1) * (mean(mediana) - median(muestra))
sesgomediajackk

## [1] 0
```

```
sesgomedianajackk
```

```
## [1] -0.003733333
```

También podríamos emplear la función `jackknife` del paquete `bootstrap`:

```
library(bootstrap)
```

```
resmedia <- jackknife(muestra, mean)
resmedia
```

```
## $jack.se
```

```
## [1] 0.1610397
```

```
##
```

```
## $jack.bias
```

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

```
##
```

```
## $jack.values
```

```
## [1] 0.8526429 0.8498571 0.8445714 0.8442857 0.8435714 0.8316429 0.8265000
```

```
## [8] 0.8192143 0.8120000 0.7885714 0.7850000 0.7807143 0.7585714 0.7285714
```

```
## [15] 0.7142857
```

```
##
```

```
## $call
```

```
## jackknife(x = muestra, theta = mean)
```

```
resmediana <- jackknife(muestra, median)
resmediana
```

```
## $jack.se
```

```
## [1] 0.1834505
```

```
##
```

```
## $jack.bias
```

```
## [1] -0.003733333
```

```
##
```

```
## $jack.values
```

```
## [1] 0.6615 0.6615 0.6615 0.6615 0.6615 0.6615 0.6615 0.6105 0.5600 0.5600
```

```
## [11] 0.5600 0.5600 0.5600 0.5600 0.5600
```

```
##
```

```
## $call
```

```
## jackknife(x = muestra, theta = median)
```

Estos resultados pueden compararse con los obtenidos en el Ejemplo 2.1 empleando bootstrap. En general las aproximaciones jackknife son adecuadas para el caso de estadísticos “suaves”, como la media, pero pueden ser inconsistentes cuando no lo son, como es el caso de la mediana.

Capítulo 3

Modificaciones del Bootstrap uniforme

El bootstrap uniforme (o naïve) es aquel en el que remuestreamos a partir de la función de distribución empírica. Eso es muy razonable cuando no tenemos ninguna información adicional sobre la función de distribución poblacional, ya que la distribución empírica es el estimador máximo verosímil no paramétrico de la función de distribución poblacional. Sin embargo, cuando en el contexto en el que nos encontramos conozcamos alguna propiedad adicional de dicha distribución poblacional, entonces debemos incorporarla en el método de remuestreo, dando lugar a otro método bootstrap que ya no debemos llamar uniforme o naïve. Veremos algunos de ellos.

3.1 Bootstrap paramétrico

Supongamos que sabemos que la función de distribución poblacional pertenece a cierta familia paramétrica. Es decir $F = F_\theta$ para algún vector d -dimensional $\theta \in \Theta$. En ese caso parece lógico estimar θ a partir de la muestra (denotemos $\hat{\theta}$ un estimador de θ , por ejemplo el de máxima verosimilitud) y obtener remuestras de $F_{\hat{\theta}}$ no de F_n . Entonces, el bootstrap uniforme se modifica de la siguiente forma, dando lugar al llamado bootstrap paramétrico:

1. Dada la muestra $\mathbf{X} = (X_1, \dots, X_n)$, calcular $\hat{\theta}$
2. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de $F_{\hat{\theta}}$
3. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
4. Calcular $R^* = R(\mathbf{X}^*, F_{\hat{\theta}})$

Así utilizaremos las distribuciones en el remuestreo de R^* para aproximar la distribución en el muestreo de R . Lógicamente, cuando no sea posible obtener una expresión explícita para la distribución bootstrap de R^* utilizaremos una aproximación de Monte Carlo de la misma:

1. Dada la muestra $\mathbf{X} = (X_1, \dots, X_n)$, calcular $\hat{\theta}$
2. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de $F_{\hat{\theta}}$
3. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
4. Calcular $R^* = R(\mathbf{X}^*, F_{\hat{\theta}})$
5. Repetir B veces los pasos 2-4 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
6. Utilizar esas réplicas bootstrap para aproximar la distribución en el muestreo de R

En general, para llevar a cabo el paso 2, debemos poder simular valores de la distribución $F_{\hat{\theta}}$ (en el caso del bootstrap uniforme se trataba de simular valores de la distribución empírica, lo cual es muy sencillo y rápido). Para ello podemos utilizar el método de inversión, que consiste en simular un valor

U procedente de una distribución $\mathcal{U}(0, 1)$ (es decir, U es un número aleatorio uniforme) y devolver $X^* = F_{\hat{\theta}}^{-1}(U)$. Así, podríamos escribir el paso 2 de una forma más detallada:

2. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = F_{\hat{\theta}}^{-1}(U_i)$

No en todos los modelos paramétricos es fácil de calcular la inversa $F_{\hat{\theta}}^{-1}$. En algunos modelos paramétricos (como el caso de la distribución normal) ni siquiera tenemos una fórmula explícita para $F_{\theta}(x)$, con lo cual difícilmente podremos calcular explícitamente su inversa. En casos como esos es frecuente recurrir a otros métodos para simular la distribución en cuestión. Normalmente existen rutinas incorporadas a la mayoría de los lenguajes de programación y software estadístico (como R) que permiten simular directamente la mayoría de las distribuciones paramétricas habituales.

Ejemplo 3.1 (Inferencia sobre la media con varianza conocida, continuación).

Continuando con el ejemplo de tiempo de vida de microorganismos, podemos modificar fácilmente el código mostrado en el Ejemplo 1.1, de forma que se emplee bootstrap paramétrico (normal), con desviación típica conocida, para calcular un intervalo de confianza para la media poblacional.

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
             0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
sigma <- 0.6

alfa <- 0.05
x_barra <- mean(muestra)

# Remuestreo
set.seed(1)
B <- 1000
estadistico_boot <- numeric(B)
for (k in 1:B) {
  # u <- rnorm(n)
  # remuestra <- u * sigma + x_barra
  remuestra <- rnorm(n, x_barra, sigma)
  x_barra_boot <- mean(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot - x_barra)/sigma
}

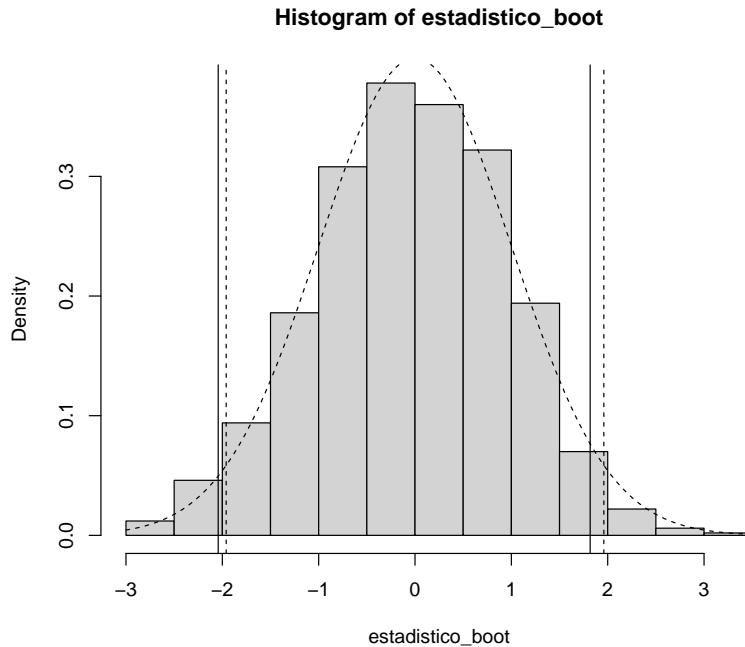
# Aproximación Monte Carlo de los ptos críticos
# Empleando la distribución empírica del estadístico bootstrap:
# estadistico_boot_ordenado <- sort(estadistico_boot)
# indice_inf <- floor(B * alfa/2)
# indice_sup <- floor(B * (1 - alfa/2))
# pto_crit <- estadistico_boot_ordenado[c(indice_inf, indice_sup)]
# Empleando la función `quantile`:
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))

# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2] * sigma/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1] * sigma/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.5236922 1.1217871
```

En este caso concreto la distribución bootstrap del estadístico sería conocida (normal estándar) y realmente no sería necesario emplear la aproximación Monte Carlo:

```
hist(estadistico_boot, freq = FALSE)
abline(v = pto_crit)
curve(dnorm, add=TRUE, lty = 2)
pto_crit_teor <- qnorm(1 - alfa/2)
abline(v = c(-pto_crit_teor, pto_crit_teor), lty = 2)
```



```
ic_inf_boot_teor <- x_barra - pto_crit_teor * sigma/sqrt(n)
ic_sup_boot_teor <- x_barra + pto_crit_teor * sigma/sqrt(n)
IC_boot_teor <- c(ic_inf_boot_teor, ic_sup_boot_teor)
names(IC_boot_teor) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot_teor

##      2.5%    97.5%
## 0.501697 1.108970
```

Para emplear el paquete `boot`, como se comentó en la Sección 1.4.1, habría que establecer en la llamada a la función `boot()` los argumentos: `sim = "parametric"`, `mle` igual a los parámetros necesarios para la simulación y `ran.gen = function(data, mle)`, una función de los datos originales y de los parámetros que devuelve los datos generados. En este caso además, la función `statistic` no necesita el vector de índices como segundo parámetro. Por ejemplo, para calcular el intervalo de confianza para la media del tiempo de vida de los microorganismos, podríamos utilizar el siguiente código:

```
library(boot)
ran.gen.norm <- function(data, mle) {
  # Función para generar muestras aleatorias normales
  # con desviación típica sigma = 0.6,
  # mle contendrá la media de los datos originales
  out <- rnorm(length(data), mle, sigma)
  out
}

statistic <- function(data){
  c(mean(data), sigma^2/length(data))
}
```

```

set.seed(1)
res.boot <- boot(muestra, statistic, R = B, sim = "parametric",
                  ran.gen = ran.gen.norm, mle = mean(muestra))

boot.ci(res.boot, type = "stud")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = "stud")
##
## Intervals :
## Level      Studentized
## 95%   ( 0.5208,  1.1232 )
## Calculations and Intervals on Original Scale

```

Aunque los resultados dependerán en gran medida de que el modelo paramétrico sea adecuado para describir la variabilidad de los datos (en este caso no es muy razonable que el modelo admita tiempos de vida negativos). Si, por ejemplo, consideramos que un modelo exponencial es más adecuado: [Figura 3.1]

```

# Distribución bootstrap uniforme
curve(ecdf(muestra)(x), xlim = c(-.5, 3), ylab = "F(x)", type = "s")
# Distribución bootstrap paramétrico normal
curve(pnorm(x, mean(muestra), 0.6), lty = 2, add = TRUE)
# Distribución bootstrap paramétrico exponencial
curve(pexp(x, 1/mean(muestra)), lty = 3, add = TRUE)
legend("bottomright", legend = c("Empírica", "Aprox. normal", "Aprox. exponencial"), lty = 1:3)

```

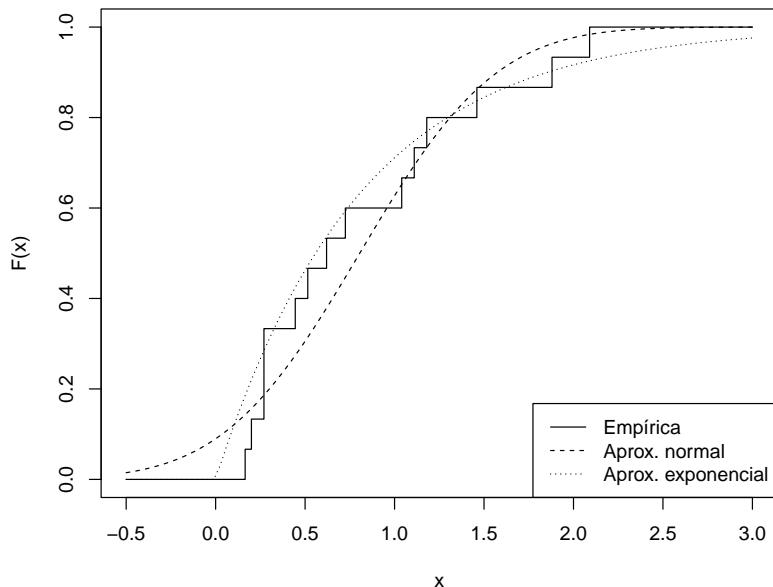


Figura 3.1: Distribución empírica de la muestra de tiempos de vida de microorganismos y aproximaciones paramétricas.

Solo tendríamos que cambiar la función que genera los datos:

```

ran.gen.exp <- function(data, mle) {
  # Función para generar muestras aleatorias exponenciales
  # mle contendrá la media de los datos originales
  out <- rexp(length(data), 1/mle)
  out
}

```

Una de las principales aplicaciones del bootstrap paramétrico es el contraste de hipótesis que se tratará en la Sección 5.2.

3.2 Bootstrap simetrizado

Supongamos que conocemos que la función de distribución poblacional es simétrica entorno a cierto valor. Eso significa que existe un valor c tal que $F(c - h) = 1 - F(c + h)$ para todo $h > 0$. Equivalentemente, una variable aleatoria es simétrica entorno a c si su función de distribución verifica

$$F(x) = 1 - F(2c - x)$$

para todo $x \in \mathbb{R}$. Puede demostrarse que dicho centro de simetría, c , ha de ser la media de la distribución, μ , en caso de que exista. Esta información (la simetría) sobre la distribución poblacional también se debe incorporar en el bootstrap. Así, para estimar la función de distribución poblacional, F , supuesto que es simétrica entorno a μ , es razonable utilizar una versión simetrizada de la distribución empírica, F_n^{sim} . Ese estimador empírico simetrizado de la función de distribución es el que otorga igual masa de probabilidad a una muestra artificialmente construida simetrizando, alrededor de la media muestral, la muestra original:

$$Y_i = \begin{cases} X_i & \text{si } i = 1, \dots, n \\ 2\bar{X} - X_{i-n} & \text{si } i = n+1, \dots, 2n \end{cases}$$

con lo cual

$$F_n^{sim}(x) = \frac{1}{2n} \sum_{i=1}^{2n} \mathbf{1}(Y_i \leq x).$$

Puede demostrarse fácilmente que

$$F_n^{sim}(x) = \frac{1}{2} (F_n(x) + 1 - F_n(2\bar{X} - x)).$$

Al diseñar el plan de remuestreo debemos utilizar F_n^{sim} (bootstrap simetrizado), en lugar de F_n (bootstrap uniforme).

1. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de F_n^{sim} , es decir $P^*(X_i^* = Y_j) = \frac{1}{2n}, j = 1, \dots, 2n$
2. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, F_n^{sim})$

Como veremos más adelante, a veces (muy poco frecuentemente) es posible calcular exactamente la distribución bootstrap de R^* . Cuando eso no es posible, esa distribución es fácilmente aproximable por Monte Carlo, arrojando una gran cantidad, B , de réplicas de R^* . En ese caso, el algoritmo se convierte en:

1. Para cada $i = 1, \dots, n$ arrojar X_i^* a partir de F_n^{sim}
2. Obtener $\mathbf{X}^* = (X_1^*, \dots, X_n^*)$
3. Calcular $R^* = R(\mathbf{X}^*, F_n^{sim})$
4. Repetir B veces los pasos 1-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$
5. Utilizar esas réplicas bootstrap para aproximar la distribución en el muestreo de R

Para llevar a cabo el paso 1 podemos proceder de dos formas equivalentes. La primera consiste en definir explícitamente la muestra simetrizada en torno a la media, \mathbf{Y} , y luego obtener uno de los valores de dicha muestra con equiprobabilidad. El paso 1 quedaría de la siguiente forma:

1. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y hacer $X_i^* = Y_{\lfloor 2nU_i \rfloor + 1}$

Alternativamente podemos proceder con el paso 1 utilizando el hecho de que la función de distribución $F_n^{sim}(x)$ resultar ser la distribución de una variable aleatoria obtenida en dos etapas: en la primera etapa se genera un valor según la empírica, $F_n(x)$, y en la segunda se decide (con equiprobabilidad) si el valor obtenido no se altera o bien si se refleja alrededor de la media muestral, \bar{X} (equivalentemente, la distribución simetrizada es una mixtura de la distribución empírica $F_n(x)$ y de su versión “reflejada” $1 - F_n(2\bar{X} - x)$ y se puede simular mediante el método de composición; ver p.e. Fernández-Casal y Cao, 2020, Sección 5.4). Así el paso 1 resulta:

1. Para cada $i = 1, \dots, n$ arrojar $U_i, V_i \sim \mathcal{U}(0, 1)$. Si $V_i \leq \frac{1}{2}$ entonces hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1}$ y en caso contrario hacer $X_i^* = 2\bar{X} - X_{\lfloor nU_i \rfloor + 1}$

La utilización de $F_n^{sim}(x)$ en lugar de $F_n(x)$ altera las propiedades conocidas de la distribución (empírica) de la que se remuestrea en el bootstrap uniforme. Así, en primer lugar, $F_n^{sim}(x)$ es simétrica (como se desea) con lo cual todos los momentos impares de esta distribución con respecto a \bar{X} son cero. En particular la media de $F_n^{sim}(x)$ es

$$\begin{aligned}\int x \, dF_n^{sim}(x) &= \frac{1}{2n} \sum_{i=1}^{2n} Y_i = \frac{1}{2n} \left[\sum_{i=1}^n X_i + \sum_{i=1}^n (2\bar{X} - X_i) \right] \\ &= \frac{1}{2n} (n\bar{X} + 2n\bar{X} - n\bar{X}) = \bar{X}.\end{aligned}$$

También se conservan los momentos centrales de orden par:

$$\begin{aligned}\int (x - \bar{X})^{2k} \, dF_n^{sim}(x) &= \frac{1}{2n} \sum_{i=1}^{2n} (Y_i - \bar{X})^{2k} \\ &= \frac{1}{2n} \left[\sum_{i=1}^n (X_i - \bar{X})^{2k} + \sum_{i=1}^n [(2\bar{X} - X_i) - \bar{X}]^{2k} \right] \\ &= \frac{1}{2n} \left[\sum_{i=1}^n (X_i - \bar{X})^{2k} + \sum_{i=1}^n (\bar{X} - X_i)^{2k} \right] \\ &= \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^{2k}.\end{aligned}$$

En particular, la varianza de $F_n^{sim}(x)$ coincide con la de $F_n(x)$, que es S_n^2 .

En general, cuando la distribución de partida es simétrica, es más adecuado utilizar el bootstrap simetrizado que el bootstrap uniforme. Aún así, cuando se realiza inferencia sobre algún estadístico (como $\sqrt{n}(\bar{X} - \mu)/\sigma$) cuya distribución asintótica ya es simétrica (como la normal), la aproximación bootstrap uniforme para distribuciones de partida simétricas, ya es especialmente buena y, por tanto, la ganancia del bootstrap simetrizado aporta una mejora difícil de detectar en la práctica. Ese no es el caso de otros estadísticos (como los asociados a inferencia sobre la varianza) con distribución más alejada de la simetría.

Ejercicio 3.1 (Inferencia sobre la media con varianza conocida empleando bootstrap simetrizado). Modificar adecuadamente el código del Ejemplo 1.1, para implementar un método bootstrap simetrizado, con el objeto de calcular un intervalo de confianza para la media con desviación típica conocida. ¿Qué diferencias se observan entre los intervalos obtenidos por el bootstrap uniforme y por el simetrizado?

3.3 Bootstrap suavizado

Cuando la distribución poblacional, F , es continua es lógico incorporar dicha información al bootstrap. Eso significa que la función de distribución tiene una función de densidad asociada, relacionadas mediante la expresión: $f(x) = F'(x)$. Para ello, debemos utilizar un método bootstrap que remuestre de un universo bootstrap continuo. En otras palabras debemos utilizar un estimador de la función de densidad y remuestrear de él.

Pasamos a considerar brevemente el problema de estimar, no paramétricamente, la función de densidad, f , de una población, a partir de una muestra, (X_1, X_2, \dots, X_n) , procedente de la misma. En ese contexto es bien conocido el método histograma (basado en el cual sería posible idear un método bootstrap) aunque es más recomendable utilizar el estimador tipo núcleo propuesto por Parzen (1962) y Rosenblatt (1956), que viene dado por

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right) = \frac{1}{n} \sum_{i=1}^n K_h(x - X_i),$$

donde

$$K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right),$$

K es una función núcleo (normalmente una densidad simétrica en torno al cero) y $h > 0$ es una parámetro de suavizado, llamado ventana, que regula el tamaño del entorno que se usa para llevar a cabo la estimación. Este estimador generaliza el bien conocido histograma y, más concretamente, su versión histograma móvil. Así, eligiendo como función K la densidad de una $\mathcal{U}(-1, 1)$, el estimador de Parzen-Rosenblatt resulta:

$$\begin{aligned} \frac{1}{nh} \sum_{i=1}^n \frac{1}{2} \mathbf{1}\left\{\frac{x - X_i}{h} \in (-1, 1)\right\} &= \frac{1}{2nh} \sum_{i=1}^n \mathbf{1}\{X_i \in (x - h, x + h)\} \\ &= \frac{\#\{X_i \in (x - h, x + h)\}}{2nh}, \end{aligned}$$

que no es más que la frecuencia relativa de datos X_i en el intervalo $(x - h, x + h)$ dividida entre la longitud del intervalo en cuestión ($2h$).

Es habitual exigir que la función núcleo K sea no negativa y su integral sea uno:

$$K(u) \geq 0, \forall u, \int_{-\infty}^{\infty} K(u) du = 1.$$

Además también es frecuente exigir que K sea una función simétrica ($K(-u) = K(u)$).

Aunque la elección de la función K no tiene gran impacto en las propiedades del estimador (salvo sus condiciones de regularidad: continuidad, diferenciabilidad, etc.) la elección del parámetro de suavizado sí es muy importante para una correcta estimación. En otras palabras, el tamaño del entorno usado para la estimación no paramétrica debe ser adecuado (ni demasiado grande ni demasiado pequeño).

En R podemos emplear la función `density()` del paquete base para obtener una estimación tipo núcleo de la densidad (con la ventana determinada por el parámetro `bw`), aunque podríamos emplear implementaciones de otros paquetes (en la Sección 6.5 se incluyen más detalles). Por ejemplo, considerando el conjunto de datos `precip` (que contiene el promedio de precipitación, en pulgadas de lluvia, de 70 ciudades de Estados Unidos), podríamos utilizar el siguiente código [Figura 3.2]:

```
x <- precip
npden <- density(x)
# npden <- density(x, bw = "SJ")

# plot(npden)
bandwidth <- npden$bw
hist(x, freq = FALSE, main = "Kernel density estimation",
```

```
xlab = paste("Bandwidth =", formatC(bandwidth)), lty = 2,
border = "darkgray", xlim = c(0, 80), ylim = c(0, 0.08))
lines(npden, lwd = 2)
rug(x, col = "darkgray")
```

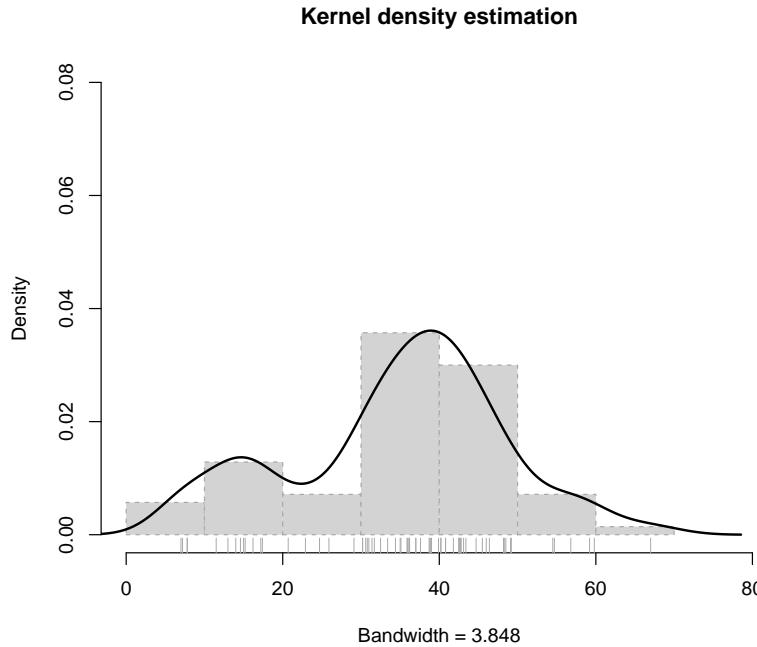


Figura 3.2: Estimación tipo núcleo de la densidad de ‘precip’.

La sensibilidad del estimador tipo núcleo al parámetro de suavizado puede observarse ejecutando el siguiente código (ver Figura 3.3, bandwidth-movie.gif):

```
bws <- 2^seq(log2(bandwidth * 0.01), log2(bandwidth * 20), len = 50)
bws <- c(bws, rev(bws))
for (bw in bws)
  plot(density(x, bw = bw) , main = "Kernel density estimation",
        xlab = paste("Bandwidth =", formatC(bw)),
        xlim = c(0, 80), ylim = c(0, 0.08))
```

La función de distribución asociada al estimador tipo núcleo de la función de densidad viene dada por

$$\begin{aligned}\hat{F}_h(x) &= \int_{-\infty}^x \hat{f}_h(y) dy = \int_{-\infty}^x \frac{1}{n} \sum_{i=1}^n \frac{1}{h} K\left(\frac{y-X_i}{h}\right) dy \\ &= \frac{1}{nh} \sum_{i=1}^n \int_{-\infty}^x K\left(\frac{y-X_i}{h}\right) dy \\ &= \frac{1}{n} \sum_{i=1}^n \int_{-\infty}^{\frac{x-X_i}{h}} K(u) du = \frac{1}{n} \sum_{i=1}^n \mathbb{K}\left(\frac{x-X_i}{h}\right)\end{aligned}$$

donde \mathbb{K} es la función de distribución asociada al núcleo K , es decir

$$\mathbb{K}(t) = \int_{-\infty}^t K(u) du.$$

Por ejemplo, en el caso de del conjunto de datos de precipitaciones, el siguiente código compara la estimación tipo núcleo de la distribución con la empírica [Figura 3.4]:

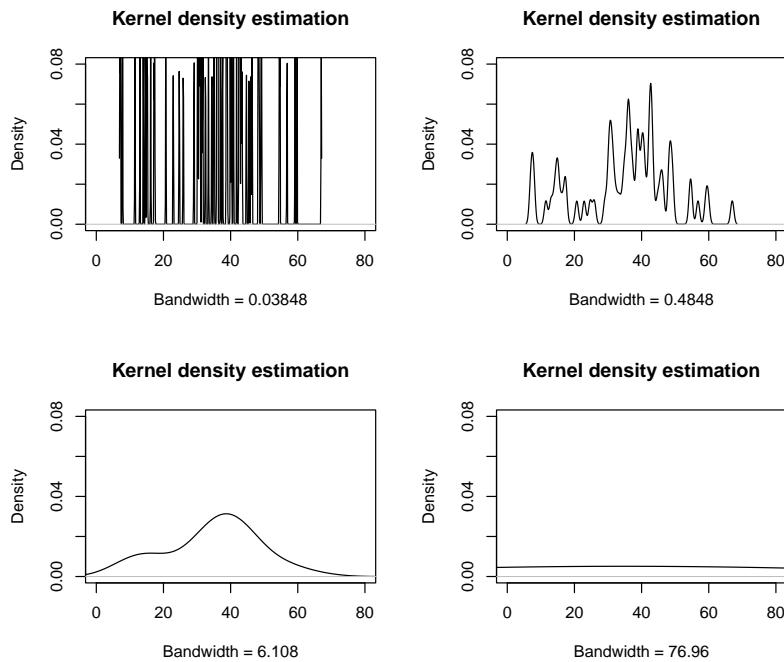


Figura 3.3: Efecto de cambio en la ventana en la estimación tipo núcleo de la densidad.

```

Fn <- ecdf(precip)
curve(Fn, xlim = c(0, 75), ylab = "F(x)", type = "s")
Fnp <- function(x) sapply(x, function(y) mean(pnorm(y, precip, bandwidth)))
curve(Fnp, lty = 2, add = TRUE)
legend("bottomright", legend = c("Empírica", "Tipo núcleo"), lty = 1:2)

```

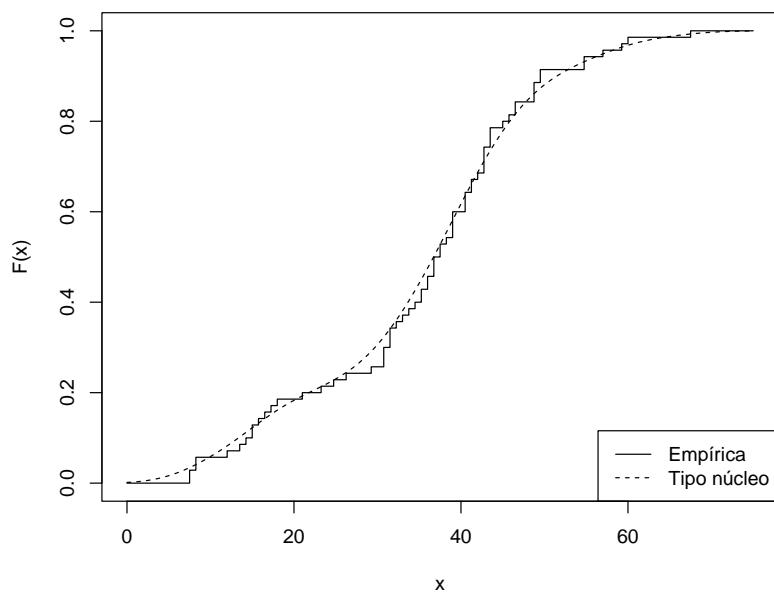


Figura 3.4: Estimación empírica y tipo núcleo de la función de distribución de 'precip'.

El método bootstrap suavizado procede de la siguiente forma:

1. A partir de la muestra (X_1, X_2, \dots, X_n) y utilizando un valor $h > 0$ como parámetro de suavizado, se calcula el estimador de Parzen-Rosenblatt \hat{f}_h
2. Se arrojan remuestras bootstrap $\mathbf{X}^* = (X_1^*, X_2^*, \dots, X_n^*)$ a partir de la densidad \hat{f}_h
3. Calcular $R^* = R(\mathbf{X}^*, \hat{F}_h)$
4. Repetir B veces los pasos 2-3 para obtener las réplicas bootstrap $R^{*(1)}, \dots, R^{*(B)}$

Para llevar a cabo un bootstrap que remuestree a partir del estimador $\hat{f}_h(x)$ es útil pensar en dicho estimador como una combinación lineal convexa de funciones de densidad, $K_h(x - X_i)$, cada una con coeficiente $\frac{1}{n}$ en dicha combinación lineal. Gracias a esa representación podemos simular valores, X^* , procedentes de $\hat{f}_h(x)$ en dos pasos (empleando el denominado método de composición; ver p.e. Fernández-Casal y Cao, 2020, Sección 5.4).

En un primer paso elegiremos (aleatoriamente y con equiprobabilidad) cuál de los índices $i \in \{1, \dots, n\}$ vamos a considerar y en un segundo paso simularemos X^* a partir de la densidad $K_h(\cdot - X_i)$. Esta última fase puede relacionarse fácilmente con la simulación de un valor, V , con densidad K , sin más que hacer $X_i + hV$. Así, el paso 2 del algoritmo previo puede llevarse a cabo mediante el siguiente procedimiento:

2. Para cada $i = 1, \dots, n$ arrojar $U_i \sim \mathcal{U}(0, 1)$ y V_i con densidad K y hacer $X_i^* = X_{\lfloor nU_i \rfloor + 1} + hV_i$

La equivalencia de ambas presentaciones del paso 2 viene dada por el siguiente razonamiento. Denotando $U \sim \mathcal{U}(0, 1)$, $I = \lfloor nU \rfloor + 1$ y $V \sim K$, independiente de U , se tiene:

$$\begin{aligned} P^*(X^* \leq x) &= \sum_{i=1}^n P^*(X^* \leq x|_{I=i}) P^*(I = i) \\ &= \sum_{i=1}^n P^*(X_i + hV \leq x|_{I=i}) P^*(I = i) \\ &= \sum_{i=1}^n P^*\left(V \leq \frac{x - X_i}{h} \Big|_{X_i}\right) \frac{1}{n} = \frac{1}{n} \sum_{i=1}^n \mathbb{K}\left(\frac{x - X_i}{h}\right), \end{aligned}$$

cuya función de densidad es, como ya sabemos, $\hat{f}_h(x)$. Esto justifica la presentación alternativa del paso 2, de forma que el bootstrap suavizado puede pensarse, a partir del bootstrap uniforme ($X_i^* = X_{\lfloor nU_i \rfloor + 1}$) añadiendo al mismo una perturbación (hV_i) cuya magnitud viene dada por el parámetro de suavizado (h) y cuya forma imita a la de una variable aleatoria (V_i) con densidad K .

Por ejemplo, la función `density()` emplea por defecto un núcleo gaussiano, y como se muestra en la ayuda de esta función, podemos emplear un código como el siguiente para obtener `nsim` simulaciones (ver Figura 3.5):

```
## simulation from a density() fit:
# a kernel density fit is an equally-weighted mixture.
nsim <- 1e6
set.seed(1)
# x_boot <- sample(x, nsim, replace = TRUE)
# x_boot <- x_boot + bandwidth * rnorm(nsim)
x_boot <- rnorm(nsim, sample(x, nsim, replace = TRUE), bandwidth)

plot(npden, main = "")
lines(density(x_boot), col = "blue", lwd = 2, lty = 2)
```

Es fácil percibirse de que los posibles valores que puede tomar una observación X_i^* de cada remuestra bootstrap son infinitos, pues la variable V puede tomar infinitos posibles valores, según la densidad de probabilidad K . Esto significa que la distribución en el remuestreo de la remuestra bootstrap, \mathbf{X}^* , es mucho más complicada que para el bootstrap uniforme. En particular no es discreta y por tanto

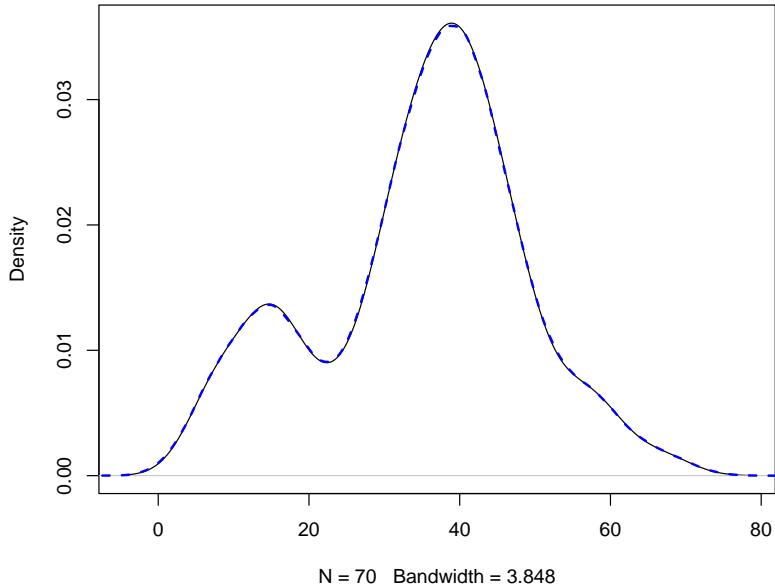


Figura 3.5: Estimaciones tipo núcleo de las densidades de ‘precip’ y de una simulación.

no puede caracterizarse a partir de vectores de remuestreo sobre la muestra original. Un problema importante es la elección del parámetro de suavizado, h , en este procedimiento de remuestreo. En la práctica es razonable elegir h como un valor bastante pequeño, en relación con la desviación típica de la muestra. Es fácil observar que en el caso extremo $h = 0$ este método de remuestreo se reduce al bootstrap uniforme.

Ejemplo 3.2 (Inferencia sobre la media con varianza conocida, continuación). Continuando con el ejemplo de tiempo de vida de microorganismos, podemos modificar fácilmente el código mostrado en el Ejemplo 1.1, para implementar bootstrap suavizado con función núcleo gaussiana, para calcular un intervalo de confianza para la media poblacional con desviación típica conocida:

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
            0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
sigma <- 0.6

alfa <- 0.05
x_barra <- mean(muestra)

# Remuestreo
set.seed(1)
B <- 1000
# h <- 1e-08
h <- bw.SJ(muestra)/2
estadistico_boot <- numeric(B)
for (k in 1:B) {
  # remuestra <- sample(muestra, n, replace = TRUE)
  # remuestrasu <- remuestra + h * rnorm(n, 0, 1)
  remuestrasu <- rnorm(n, sample(muestra, n, replace = TRUE), h)
  x_barra_boot <- mean(remuestrasu)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot - x_barra)/sigma}
```

```

}

# Aproximación bootstrap de los ptos críticos
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))

# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2] * sigma/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1] * sigma/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.4668897 1.0798549

```

Con el paquete `boot`, la recomendación es implementarlo como un bootstrap paramétrico:

```

library(boot)
ran.gen.smooth <- function(data, mle) {
  # Función para generar muestras aleatorias mediante
  # bootstrap suavizado con función núcleo gaussiana,
  # mle contendrá la ventana.
  n <- length(data)
  h <- mle
  out <- rnorm(n, sample(data, n, replace = TRUE), h)
  out
}

statistic <- function(data){
  c(mean(data), sigma^2/length(data))
}

set.seed(1)
res.boot <- boot(muestra, statistic, R = B, sim = "parametric",
                  ran.gen = ran.gen.smooth, mle = h)

boot.ci(res.boot, type = "stud")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = "stud")
##
## Intervals :
## Level   Studentized
## 95%   ( 0.4664,  1.0830 )
## Calculations and Intervals on Original Scale

```

3.4 Bootstrap ponderado y bootstrap sesgado

Mediante el nombre bootstrap ponderado se incluyen todos aquellos métodos de remuestreo bootstrap en los que la distribución de la que se remuestrea es discreta y asigna probabilidades sólo a los datos de la muestra:

$$\hat{F}(X_i) - \hat{F}(X_i^-) = p_i, \text{ para } i = 1, \dots, n$$

siendo $p_i \geq 0$ y $\sum_{i=1}^n p_i = 1$. En el caso particular $p_i = \frac{1}{n}$ para todo $i = 1, \dots, n$, se tiene el bootstrap uniforme. Veremos más adelante casos particulares de métodos bootstrap ponderados en el contexto

de datos censurados y también para datos dependientes.

El bootstrap ponderado da lugar al bootstrap sesgado cuando los pesos, p_i , se eligen de forma que el vector \mathbf{p} minimice la distancia al vector de pesos del bootstrap uniforme $(\frac{1}{n}, \dots, \frac{1}{n})$, sujeto a una serie de restricciones inherentes al problema en estudio. Este método fue propuesto por Hall (1998).

3.5 Deficiencias del bootstrap uniforme

Supongamos un contexto paramétrico en el que la distribución poblacional, F , es la $\mathcal{U}(0, \theta)$. Nuestro interés será hacer inferencia acerca del parámetro θ , para lo cual, dada una muestra observada, $\mathbf{X} = (X_1, X_2, \dots, X_n)$, consideraremos el estimador máximo verosímil en este contexto: $\hat{\theta} = X_{(n)}$. Para realizar dicha inferencia estaremos interesados en aproximar la distribución de $R(\mathbf{X}, F) = \hat{\theta} - \theta$.

La función de distribución en el muestreo, $G(x)$, de $\hat{\theta}$ puede calcularse de forma sencilla:

$$\begin{aligned} G(x) &= P(\hat{\theta} \leq x) = P(X_{(n)} \leq x) = P(X_i \leq x, \forall i \in \{1, \dots, n\}) \\ &= \prod_{i=1}^n P(X_i \leq x) = F(x)^n = \left(\frac{x}{\theta}\right)^n, \text{ si } x \in [0, \theta] \end{aligned}$$

con lo cual su función de densidad viene dada por

$$g(x) = \frac{n}{\theta} \left(\frac{x}{\theta}\right)^{n-1}, \text{ si } x \in [0, \theta].$$

Tomando, por ejemplo, $\theta = 1$ y $n = 50$, esta función de densidad resulta [Figura 3.6]:

```
theta <- 1
n <- 50
curve(n/theta * (x/theta)^(n - 1), 0, theta, ylab = "Density")
```

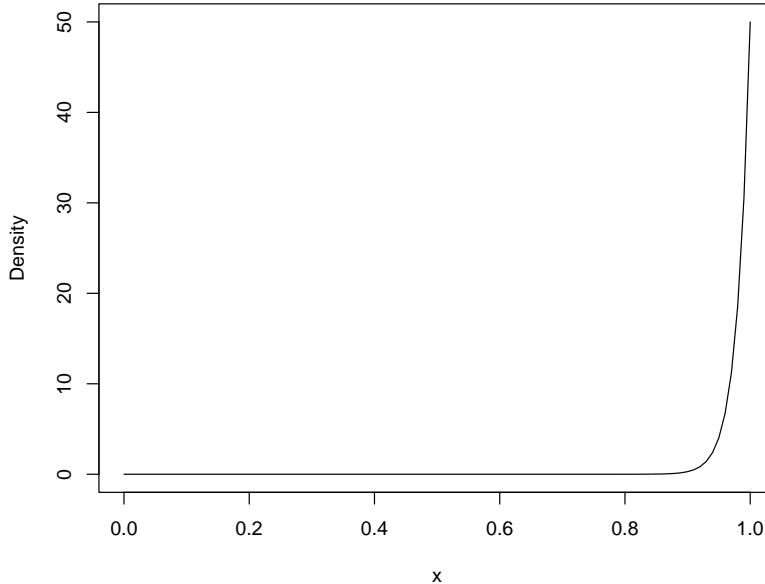


Figura 3.6: Función de densidad del máximo de una muestra procedente de una uniforme.

Como consecuencia podemos hallar fácilmente el sesgo del estimador $\hat{\theta}$, ya que

$$E(\hat{\theta}) = \int_0^\theta x \frac{n}{\theta} \left(\frac{x}{\theta}\right)^{n-1} dx = \left[\frac{n}{n+1} \frac{x^{n+1}}{\theta^n} \right]_{x=0}^{x=\theta} = \frac{n}{n+1} \theta,$$

con lo cual

$$Sesgo(\hat{\theta}) = E(\hat{\theta}) - \theta = -\frac{\theta}{n+1}.$$

Se ve claramente que $\hat{\theta}$ es un estimador sesgado de θ , puesto que se tiene que $\hat{\theta} \leq \theta$ con probabilidad 1.

Si deseamos aproximar mediante bootstrap la distribución en el muestreo de $\hat{\theta}$ (o la de R) y utilizamos un bootstrap uniforme (naïve), la versión bootstrap del estimador resulta ser $\hat{\theta}^* = X_{(n)}^*$, siendo $\mathbf{X}^* = (X_1^*, X_2^*, \dots, X_n^*)$ una remuestra bootstrap obtenida a partir de la distribución empírica F_n . La distribución en el remuestreo de $\hat{\theta}^*$ resulta un poco más complicada pues es discreta y sólo puede tomar cualquiera de los valores de la muestra.

Suponiendo que no hay empates en las observaciones de la muestra, es fácil darse cuenta de que

$$P^*(\hat{\theta}^* \leq X_{(j)}) = P^*(X_{(n)}^* \leq X_{(j)}) = P^*(X_i^* \leq X_{(j)}, 1 \leq i \leq n) = \left(\frac{j}{n}\right)^n$$

y, por tanto, su masa de probabilidad viene dada por

$$P^*(\hat{\theta}^* = X_{(j)}) = \left(\frac{j}{n}\right)^n - \left(\frac{j-1}{n}\right)^n, \quad j = 1, \dots, n.$$

En particular,

$$P^*(\hat{\theta}^* = X_{(n)}) = 1 - \left(1 - \frac{1}{n}\right)^n \rightarrow 1 - \frac{1}{e} \simeq 0.6321,$$

con lo cual la distribución en remuestreo de $R^* = R(\mathbf{X}^*, F_n) = \hat{\theta}^* - X_{(n)}$ tiene un átomo de probabilidad en el valor 0 cuya probabilidad tiende a $1 - \frac{1}{e}$ cuando el tamaño muestral tiende a infinito, es decir

$$\lim_{n \rightarrow \infty} P^*(R^* = 0) = 1 - \frac{1}{e},$$

cosa que no ocurre con la distribución en el muestreo de R , que es continua con densidad:

$$g_R(x) = \frac{n}{\theta} \left(\frac{x + \theta}{\theta}\right)^{n-1}, \quad \text{si } x \in [-\theta, 0].$$

De esta forma vemos que el bootstrap uniforme (no paramétrico) es inconsistente.

Ejemplo 3.3 (Inferencia sobre el máximo de una distribución uniforme).

El siguiente código implementa el método bootstrap uniforme (también llamado naïve) para aproximar la distribución del estadístico $R = \hat{\theta} - \theta$, para una muestra de tamaño $n = 50$, proveniente de una población con distribución $\mathcal{U}(0, 1)$ [Figura 3.7]:

```
theta <- 1
n <- 50
set.seed(1)
muestra <- runif(50) * theta
theta_est <- max(muestra)
# Remuestreo
B <- 2000
maximo <- numeric(B)
estadistico <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  maximo[k] <- max(remuestra)
  estadistico[k] <- maximo[k] - theta_est
```

```

}
# Distribución estadístico
xlim <- c(-theta/2, 0) # c(-theta, 0)
hist(estadistico, freq = FALSE, main = "", lty = 2,
      border = "darkgray", xlim = xlim)
lines(density(estadistico))
rug(estadistico, col = "darkgray")
curve(n/theta * ((x + theta)/theta)^(n - 1), col = "blue", lty = 2, lwd = 2, add = TRUE)

```

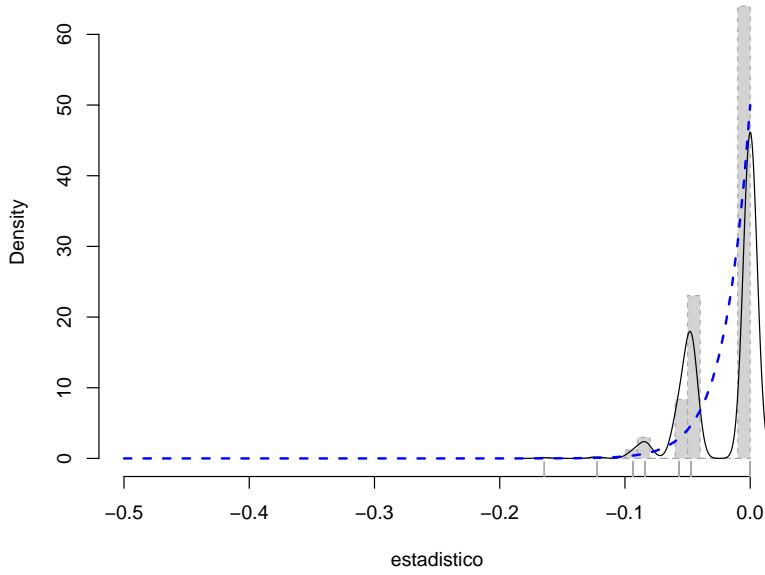


Figura 3.7: Distribución de las réplicas bootstrap (uniforme) del estadístico y distribución poblacional.

3.5.1 Ejemplo (método alternativo)

En este contexto, al conocer la familia paramétrica ($\mathcal{U}(0, \theta)$) a la cual pertenece la distribución de la población de partida, lo natural sería utilizar un bootstrap paramétrico, consistente en obtener las remuestras bootstrap a partir de una distribución uniforme con parámetro estimado:

$$\mathbf{X}^* = (X_1^*, X_2^*, \dots, X_n^*), \text{ con } X_i^* \sim \mathcal{U}(0, \hat{\theta}).$$

En estas circunstancias es muy sencillo obtener la distribución en el remuestreo de $\hat{\theta}^*$, ya que su deducción es totalmente paralela a la de la distribución en el muestreo de $\hat{\theta}$. Así, la función de densidad de $\hat{\theta}^*$ es

$$\hat{g}(x) = \frac{n}{\hat{\theta}} \left(\frac{x}{\hat{\theta}} \right)^{n-1}, \text{ si } x \in [0, \hat{\theta}].$$

Con lo cual, al utilizar un bootstrap paramétrico, la distribución en el remuestreo de $R^* = R(\mathbf{X}^*, F_{\hat{\theta}}) = \hat{\theta}^* - \hat{\theta}$ imita a la distribución en muestreo de $R = R(\mathbf{X}, F) = \hat{\theta} - \theta$.

Ejemplo 3.4 (Inferencia sobre el máximo de una distribución uniforme, continuación).

Para emplear el bootstrap paramétrico (que remuestrea de una distribución uniforme con parámetro estimado) podríamos emplear un código muy similar al del Ejemplo 3.3 [Figura 3.8]:

```

# Remuestreo
B <- 2000
maximo <- numeric(B)
estadistico <- numeric(B)
for (k in 1:B) {
  remuestra <- runif(n) * theta_est
  maximo[k] <- max(remuestra)
  estadistico[k] <- maximo[k] - theta_est
}
# Distribución estadístico
xlim <- c(-theta/2, 0) # c(-theta, 0)
hist(estadistico, freq = FALSE, main = "", lty = 2,
  border = "darkgray", xlim = xlim)
lines(density(estadistico))
rug(estadistico, col = "darkgray")
curve(n(theta * ((x + theta)/theta)^(n - 1), col = "blue", lty = 2, lwd = 2, add = TRUE)

```

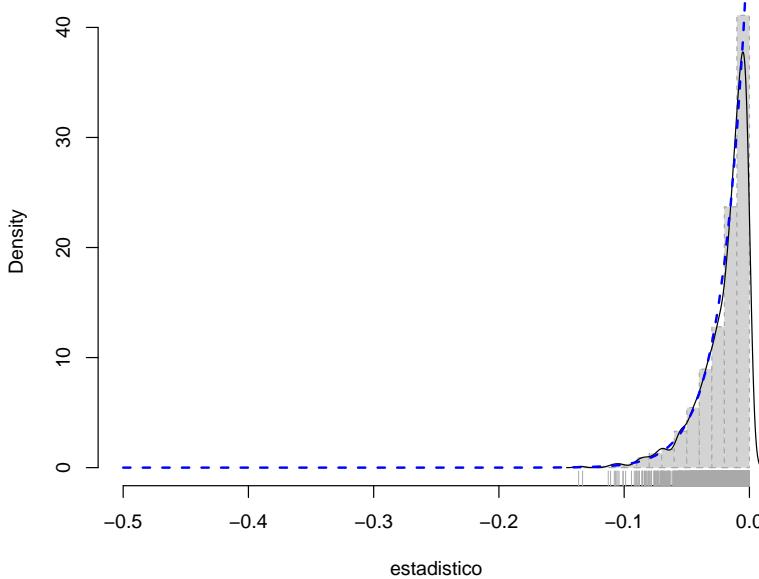


Figura 3.8: Distribución bootstrap paramétrica y distribución poblacional.

3.6 Validez de la aproximación Bootstrap

Trataremos ahora de dar una justificación teórica del buen funcionamiento del bootstrap uniforme. Para ello, por simplicidad, nos centraremos en el problema de aproximar la distribución en el muestreo del estadístico

$$R = R(\mathbf{X}, F) = \sqrt{n} \frac{\bar{X} - \mu}{\sigma},$$

donde $\mathbf{X} = (X_1, X_2, \dots, X_n)$ es una m.a.s. procedente de una distribución F , con media μ y desviación típica σ . Sabemos que, bajo ciertas condiciones, el teorema central del límite permite obtener la distribución asintótica de R , que es una $\mathcal{N}(0, 1)$, es decir

$$\lim_{n \rightarrow \infty} P(R \leq u) = \Phi(u), \quad \forall u \in \mathbb{R},$$

siendo Φ la función de distribución de una normal estándar, cuya función de densidad denotaremos por ϕ .

Para ver cómo de buena es la aproximación por normal del estadístico R , debemos razonar cómo de rápida es la convergencia en el límite anteriormente expuesto. La respuesta a esa pregunta viene dada por el Teorema de Cramer que usa los llamados desarrollos de Edgeworth de un estadístico para aproximarla por una suma de términos, el primero es la función de distribución normal estándar y los siguientes irán tiendiendo a cero sucesivamente más rápido cuando el tamaño muestral tiende a infinito. Enunciamos ese resultado.

Teorema 3.1 (Cramer)

Consideremos variables aleatorias $X_1, X_2, \dots, X_n, \dots$ independientes e idénticamente distribuidas procedentes de una distribución F , con media μ y desviación típica σ . Supongamos que existe cierto j , natural, para el cual $E(|X|^{j+2}) < \infty$, y que $\lim_{|t| \rightarrow \infty} |\alpha(t)| < 1$, siendo $\alpha(t) = E(e^{itX})$ la función característica de la población. Entonces:

$$\begin{aligned} P(R \leq u) &= P\left(\sqrt{n}\frac{\bar{X} - \mu}{\sigma} \leq u\right) \\ &= \Phi(u) + n^{-\frac{1}{2}} p_1(u) \phi(u) + \dots + n^{-\frac{j-1}{2}} p_{j-1}(u) \phi(u) + O(n^{-\frac{j}{2}}), \end{aligned}$$

siendo los $p_i(u)$ polinomios de grado $3i - 1$ cuyos coeficientes dependen de los momentos de X de orden menor o igual que $i + 2$. En particular

$$\begin{aligned} p_1(u) &= -\frac{1}{6} \frac{k_3}{\sigma^3} (u^2 - 1), \\ p_2(u) &= -u \left[\frac{1}{24} \frac{k_4}{\sigma^4} (u^2 - 3) + \frac{1}{72} \left(\frac{k_3}{\sigma^3} \right)^2 (u^4 - 10u^2 + 15) \right], \end{aligned}$$

siendo k_j el j -ésimo cumulante, es decir el términos que acompaña a $\frac{(it)^j}{j!}$ en el desarrollo en serie del logaritmo de la función característica:

$$\log \alpha(t) = \sum_{j=1}^{\infty} k_j \frac{(it)^j}{j!}.$$

Además dichos polinomios tienen paridad alternada, es decir, p_1 es simétrico, p_2 es antisimétrico, p_3 es simétrico, y así sucesivamente:

$$p_1(-u) = p_1(u), \quad p_2(-u) = -p_2(u), \quad p_3(-u) = p_3(u), \dots$$

Existen ecuaciones que relacionan todos los cumulantes hasta cierto orden con todos los momentos poblacionales hasta ese mismo orden. Dichas ecuaciones permiten expresar los cumulantes en función de los momentos y viceversa.

Como consecuencia de este resultado teórico, el grado de aproximación entre la distribución de R y la normal estándar límite es $O(n^{-\frac{1}{2}})$. Sin embargo, puede razonarse fácilmente que este orden de aproximación mejorará cuando utilizamos el bootstrap uniforme, en lugar de la normal estándar, para aproximar la distribución de R . Un desarrollo de Edgeworth para la distribución en el remuestreo de R^* permite obtener la siguiente expresión:

$$\begin{aligned} P^*(R^* \leq u) &= \Phi(u) + n^{-\frac{1}{2}} \hat{p}_1(u) \phi(u) + \dots + n^{-\frac{j-1}{2}} \hat{p}_{j-1}(u) \phi(u) \\ &\quad + O_P(n^{-\frac{j}{2}}), \end{aligned}$$

donde los polinomios $\hat{p}_i(u)$ tienen la misma estructura que los $p_i(u)$ pero reemplazando los cumulantes teóricos por los empíricos y la desviación típica teórica por la empírica. Así pues el grado de

aproximación entre cada polinomio $\hat{p}_i(u)$ y su análogo teórico $p_i(u)$ es $\hat{p}_i(u) - p_i(u) = O_P(n^{-\frac{1}{2}})$. Como consecuencia, puede obtenerse el orden de aproximación entre la distribución en el muestreo de R y la distribución en el remuestreo de R^* :

$$\begin{aligned} P(R \leq u) - P^*(R^* \leq u) &= n^{-\frac{1}{2}} [p_1(u) - \hat{p}_1(u)] \phi(u) + O_P(n^{-1}) \\ &= O_P(n^{-1}), \end{aligned}$$

que es mejor que el orden de aproximación de la normal estándar límite. Dichos órdenes pueden resumirse en la siguiente tabla.

Aproximación	Orden
Normal límite	$O(n^{-\frac{1}{2}})$
Boot. uniforme	$O_P(n^{-1})$

Usando razonamiento similares pueden encontrarse los órdenes de aproximación, tanto de la normal límite, como del bootstrap uniforme y del bootstrap simetrizado, cuando la distribucional de partida es simétrica. En ese caso, $p_1(u) = 0$, ya que k_3 es cero debido a la simetría de la distribución poblacional. Sin embargo $\hat{p}_1(u)$ no es cero cuando se usa el bootstrap uniforme, aunque sí lo es en el caso del bootstrap simetrizado. La siguiente tabla recoge los órdenes de las distintas aproximaciones.

Aproximación	Orden
Normal límite	$O(n^{-1})$
Boot. uniforme	$O_P(n^{-1})$
Boot. simetrizado	$O_P(n^{-\frac{3}{2}})$

El siguiente resultado permite generalizar los desarrollos de Edgeworth (Teorema 3.1) a otros estadísticos (estandarizados o studentizados) obtenidos para otros estimadores arbitrarios, $\hat{\theta}$, no necesariamente iguales a la media muestral.

Teorema 3.2 (Bhattacharya-Ghosh)

Consideremos variables aleatorias $X_1, X_2, \dots, X_n, \dots$ independientes e idénticamente distribuidas procedentes de una distribución F . Sea $\theta = \theta(F)$ un parámetro de dicha distribución y $\hat{\theta}$ un estimador de dicho parámetro. Supongamos además que

$$\sqrt{n}(\hat{\theta} - \theta) \rightarrow \mathcal{N}(0, \sigma_\theta^2),$$

en distribución. Entonces, bajo ciertas condiciones de regularidad (pueden verse en Bhattacharya y Ghosh, 1978) se tiene:

$$\begin{aligned} P\left(\sqrt{n}\frac{\hat{\theta} - \theta}{\sigma_\theta} \leq u\right) &= \Phi(u) + n^{-\frac{1}{2}}p_1(u)\phi(u) + \dots \\ &\quad + n^{-\frac{j-1}{2}}p_{j-1}(u)\phi(u) + O(n^{-\frac{j}{2}}), \\ P\left(\sqrt{n}\frac{\hat{\theta} - \theta}{\hat{\sigma}_\theta} \leq u\right) &= \Phi(u) + n^{-\frac{1}{2}}q_1(u)\phi(u) + \dots \\ &\quad + n^{-\frac{j-1}{2}}q_{j-1}(u)\phi(u) + O(n^{-\frac{j}{2}}), \end{aligned}$$

siendo los $p_i(u)$ y $q_i(u)$ polinomios de grado $3i - 1$ con paridad alternada, es decir, p_1 y q_1 son simétricos, p_2 y q_2 son antisimétricos, p_3 y q_3 son simétricos y así sucesivamente.

3.7 Bootstrap semiparamétrico y bootstrap residual

En ocasiones nos pueden interesar modelos semiparamétricos, en los que se asume una componente paramétrica pero no se especifica por completo la distribución de los datos. Una de las situaciones más habituales es en regresión, donde se puede considerar un modelo para la tendencia pero sin asumir una forma concreta para la distribución del error.

Nos centraremos en el caso de regresión y consideraremos como base el siguiente modelo general:

$$Y = m(\mathbf{X}) + \varepsilon, \quad (3.1)$$

donde Y es la respuesta, $\mathbf{X} = (X_1, X_2, \dots, X_p)$ es el vector de variables explicativas, $m(\mathbf{x}) = E(Y|_{\mathbf{x}=\mathbf{x}})$ es la media condicional, denominada función de regresión (o tendencia), y ε es un error aleatorio de media cero y varianza σ^2 , independiente de \mathbf{X} (errores homocedásticos independientes).

Supondremos que el objetivo es, a partir de una muestra:

$$\{(X_{1i}, \dots, X_{pi}, Y_i) : i = 1, \dots, n\},$$

realizar inferencias sobre la distribución condicional $Y|_{\mathbf{X}=\mathbf{x}}$.

El modelo (3.1) se corresponde con el denominado *diseño aleatorio*, mas general. Alternativamente se podría asumir que los valores de las variables explicativas no son aleatorios (por ejemplo han sido fijados por el experimentador), hablaríamos entonces de *diseño fijo*. Para realizar inferencias sobre modelos de regresión con errores homocedásticos se podrían emplear dos algoritmos bootstrap (e.g. Canty, 2002, y subsecciones siguientes). El primero consistiría en utilizar directamente bootstrap uniforme, remuestreando las observaciones, y sería adecuado para el caso de diseño aleatorio. La otra alternativa, que podría ser más adecuada para el caso de diseño fijo, sería lo que se conoce como *remuestreo residual*, *remuestreo basado en modelos* o *bootstrap semiparamétrico*. En esta aproximación se mantienen fijos los valores de las variables explicativas y se remuestrean los residuos. Una de las aplicaciones del bootstrap semiparamétrico es el contraste de hipótesis en regresión, que se tratará en la Sección 5.4.

Se puede generalizar el modelo (3.1) de diversas formas, por ejemplo asumiendo que la distribución del error depende de X únicamente a través de la varianza (error heterocedástico independiente). En este caso se suele reescribir como:

$$Y = m(\mathbf{X}) + \sigma(\mathbf{X})\varepsilon,$$

siendo $\sigma^2(\mathbf{x}) = Var(Y|_{\mathbf{X}=\mathbf{x}})$ la varianza condicional y suponiendo adicionalmente que ε tiene varianza uno. Se podría modificar el bootstrap residual para este caso pero habría que modelizar y estimar la varianza condicional. Alternativamente se podría emplear el denominado *Wild Bootstrap* que se describirá en la Sección 7.2.1 para el caso de modelos de regresión no paramétricos.

En esta sección nos centraremos en el caso de regresión lineal:

$$m_\beta(\mathbf{x}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p,$$

siendo $\beta = (\beta_0, \beta_1, \dots, \beta_p)^T$ el vector de parámetros (desconocidos). Su estimador mínimo cuadrático es:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{Y},$$

siendo $\mathbf{Y} = (Y_1, \dots, Y_n)^T$ el vector de observaciones de la variable Y y X la denominada *matriz del diseño* de las variables regresoras, cuyas filas son los valores observados de las variables explicativas.

En regresión lineal múltiple, bajo las hipótesis estructurales del modelo de normalidad y homocedasticidad, se dispone de resultados teóricos que permiten realizar inferencias sobre características de la distribución condicional. Si alguna de estas hipótesis no es cierta se podrían emplear aproximaciones basadas en resultados asintóticos, pero podrían ser poco adecuadas para tamaños muestrales no muy grandes. Alternativamente se podría emplear bootstrap. Con otros métodos de regresión, como los modelos no paramétricos descritos en el Capítulo 7, es habitual emplear bootstrap para realizar inferencias sobre la distribución condicional.

En esta sección se empleará el conjunto de datos `Prestige` del paquete `carData`, considerando como variable respuesta `prestige` (puntuación de ocupaciones obtenidas a partir de una encuesta) y como variables explicativas: `income` (media de ingresos en la ocupación) y `education` (media de los años de educación). Para ajustar el correspondiente modelo de regresión lineal podemos emplear el siguiente código:

```
data(Prestige, package = "carData")
# ?Prestige
modelo <- lm(prestige ~ income + education, data = Prestige)
summary(modelo)

##
## Call:
## lm(formula = prestige ~ income + education, data = Prestige)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -19.4040  -5.3308   0.0154   4.9803  17.6889 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -6.8477787  3.2189771  -2.127  0.0359 *  
## income       0.0013612  0.0002242   6.071 2.36e-08 *** 
## education    4.1374444  0.3489120  11.858 < 2e-16 *** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 7.81 on 99 degrees of freedom
## Multiple R-squared:  0.798, Adjusted R-squared:  0.7939 
## F-statistic: 195.6 on 2 and 99 DF,  p-value: < 2.2e-16
```

Como ejemplo, consideraremos que el objetivo es realizar inferencias sobre el coeficiente de determinación ajustado:

```
res <- summary(modelo)
names(res)

## [1] "call"          "terms"        "residuals"     "coefficients" 
## [5] "aliased"       "sigma"        "df"           "r.squared"    
## [9] "adj.r.squared" "fstatistic"   "cov.unscaled" 
res$adj.r.squared

## [1] 0.7939201
```

3.7.1 Remuestreo de las observaciones

Como ya se comentó, en regresión podríamos emplear bootstrap uniforme multidimensional para el caso de diseño aleatorio, aunque hay que tener en cuenta que con este método la distribución en el remuestreo de $Y^*|_{X^*=X_i}$ es degenerada.

En este caso, podríamos realizar inferencias sobre el coeficiente de determinación ajustado empleando el siguiente código:

```
library(boot)

case.stat <- function(data, i) {
  fit <- lm(prestige ~ income + education, data = data[i, ])
  summary(fit)$adj.r.squared
}
```

```

set.seed(1)
boot.case <- boot(Prestige, case.stat, R = 1000)
boot.case

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Prestige, statistic = case.stat, R = 1000)
##
##
## Bootstrap Statistics :
##      original     bias   std. error
## t1* 0.7939201 0.002495631 0.0315275

# plot(boot.case)
boot.ci(boot.case, type = c("basic", "perc", "bca"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.case, type = c("basic", "perc", "bca"))
##
## Intervals :
## Level      Basic          Percentile        BCa
## 95%  ( 0.7331,  0.8570 )  ( 0.7308,  0.8547 )  ( 0.7203,  0.8497 )
## Calculations and Intervals on Original Scale

```

3.7.2 Bootstrap residual

Como ya se comentó, en el caso de diseño fijo podemos realizar un remuestreo de los residuos:

$$\mathbf{r} = \mathbf{Y} - \hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{Y}}$$

obteniéndose las réplicas bootstrap:

$$\mathbf{Y}^* = \hat{\mathbf{Y}} + \mathbf{r}^*.$$

Por ejemplo, adaptando el código en Canty (2002) para este conjunto de datos, podríamos emplear:

```

pres.dat <- Prestige
pres.dat$fit <- fitted(modelo)
pres.dat$res <- residuals(modelo)

mod.stat <- function(data, i) {
  data$prestige <- data$fit + data$res[i]
  fit <- lm(prestige ~ income + education, data = data)
  summary(fit)$adj.r.squared
}

set.seed(1)
boot.mod <- boot(pres.dat, mod.stat, R = 1000)
boot.mod

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
```

```

## Call:
## boot(data = pres.dat, statistic = mod.stat, R = 1000)
##
## 
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.7939201 0.004401997 0.02671996
# plot(boot.mod)
boot.ci(boot.mod, type = c("basic", "perc", "bca"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.mod, type = c("basic", "perc", "bca"))
##
## Intervals :
## Level      Basic          Percentile        BCa
## 95%  ( 0.7407,  0.8464 )  ( 0.7415,  0.8471 )  ( 0.7244,  0.8331 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable

```

Sin embargo, la variabilidad de los residuos no reproduce la de los verdaderos errores, por lo que podría ser preferible (especialmente si el tamaño muestral es pequeño) emplear la modificación descrita en Davison y Hinkley (1997, Alg. 6.3, p. 271). Teniendo en cuenta que:

$$\mathbf{r} = (\mathbf{I} - \mathbf{H}) \mathbf{Y},$$

siendo $\mathbf{H} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ la matriz de proyección. La idea es remuestrear los residuos reescalados (de forma que su varianza sea constante) y centrados $e_i - \bar{e}$, siendo:

$$e_i = \frac{r_i}{\sqrt{1 - h_{ii}}},$$

donde h_{ii} es el valor de influencia o leverage, el elemento i -ésimo de la diagonal de \mathbf{H} .

En R podríamos obtener estos residuos mediante los comandos:

```

pres.dat$sres <- residuals(modelo)/sqrt(1 - hatvalues(modelo))
pres.dat$sres <- pres.dat$sres - mean(pres.dat$sres)

```

Sin embargo puede ser más cómodo emplear la función `Boot()` del paquete `car` (que internamente llama a la función `boot()`), como se describe en el apéndice “Bootstrapping Regression Models in R” del libro “An R Companion to Applied Regression” de Fox y Weisberg (2018), disponible aquí.

Esta función es de la forma:

```

Boot(object, f = coef, labels = names(f(object)), R = 999,
      method = c("case", "residual"))

```

donde:

- `object`: es un objeto que contiene el ajuste de un modelo de regresión.
- `f`: es la función de estadísticos (utilizando el ajuste como argumento).
- `method`: especifica el tipo de remuestreo: remuestreo de observaciones (“`case`”) o de residuos (“`residual`”), empleando la modificación descrita anteriormente.

Ejercicio 3.2.

Emplear la función `Boot()` del paquete `car` para hacer inferencia sobre el coeficiente de determinación ajustado del modelo de regresión lineal que explica `prestige` a partir de `income` y `education` (obtener una estimación del sesgo y de la predicción, y una estimación por intervalo de confianza de este estadístico).

```
library(car)

# set.seed(DNI)
# ...
```


Capítulo 4

Intervalos de confianza bootstrap

Consideremos el problema de construcción, mediante bootstrap, de un intervalo de confianza bilateral, con nivel de confianza $1 - \alpha$, para un parámetro θ de la distribución F . Una vez elegido el método bootstrap adecuado a la información disponible en el contexto del que se trate, un aspecto importante es el de la posible corrección de los intervalos de confianza bootstrap, aproximados por el método de Monte Carlo, al objeto de que la probabilidad de cobertura sea lo más parecida posible al nivel nominal $1 - \alpha$. Comenzaremos analizando el error de cobertura de los intervalos de confianza clásicos, los basados en la distribución normal asintótica.

4.1 Intervalos basados en la distribución normal asintótica

Consideremos primeramente el caso más sencillo (y poco realista) de construcción de un intervalo de confianza para la media, μ , con desviación típica, σ , conocida. El estadístico usado para construir el intervalo de confianza es

$$R = \sqrt{n} \frac{\bar{X} - \mu}{\sigma}$$

que cuando $n \rightarrow \infty$ tiende en distribución a una $N(0, 1)$. El intervalo de confianza basado en dicha aproximación normal es $\hat{I} = (\bar{X} - \frac{\sigma}{\sqrt{n}}z_{\alpha/2}, \bar{X} + \frac{\sigma}{\sqrt{n}}z_{\alpha/2})$. Mediante el desarrollo de Edgeworth dado por el Teorema de Cramer es fácil obtener una cota para el error de cobertura de dicho intervalo:

$$\begin{aligned} P(\mu \in \hat{I}) - (1 - \alpha) &= P(R < z_{\alpha/2}) - P(R \leq -z_{\alpha/2}) \\ &\quad - (\Phi(z_{\alpha/2}) - \Phi(-z_{\alpha/2})) \\ &= n^{-\frac{1}{2}} p_1(z_{\alpha/2}) \phi(z_{\alpha/2}) + O(n^{-1}) \\ &\quad - (n^{-\frac{1}{2}} p_1(-z_{\alpha/2}) \phi(-z_{\alpha/2}) + O(n^{-1})) \\ &= O(n^{-1}), \end{aligned}$$

ya que por la simetría de las funciones $p_1(u)$ y $\phi(u)$ se tiene

$$p_1(-z_{\alpha/2}) \phi(-z_{\alpha/2}) = p_1(z_{\alpha/2}) \phi(z_{\alpha/2}).$$

De esta forma, el orden del error de cobertura del intervalo de confianza bilateral con desviación típica poblacional conocida es $O(n^{-1})$. Puede obtenerse fácilmente el orden del error de cobertura de los intervalos unilaterales que resulta ser $O(n^{-\frac{1}{2}})$.

En el caso más realista en que la desviación típica, σ , sea desconocida, el intervalo de confianza resulta

$$\hat{I}_0 = \left(\bar{X} - \frac{S_n}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{S_n}{\sqrt{n}} z_{\alpha/2} \right).$$

Ahora, el estadístico en el que se basa la inferencia resulta:

$$R_1 = \sqrt{n} \frac{\bar{X} - \mu}{S_n}.$$

Un desarrollo de Edgeworth del tipo del obtenido en el Teorema de Bhattacharya-Ghosh permite acotar el error de cobertura de este intervalo:

$$\begin{aligned} P(\mu \in \hat{I}_0) - (1 - \alpha) &= P(R_1 < z_{\alpha/2}) - P(R_1 \leq -z_{\alpha/2}) \\ &\quad - (\Phi(z_{\alpha/2}) - \Phi(-z_{\alpha/2})) \\ &= n^{-\frac{1}{2}} q_1(z_{\alpha/2}) \phi(z_{\alpha/2}) + O(n^{-1}) \\ &\quad - (n^{-\frac{1}{2}} q_1(-z_{\alpha/2}) \phi(-z_{\alpha/2}) + O(n^{-1})) \\ &= O(n^{-1}). \end{aligned}$$

De esta forma, el orden del error de cobertura del intervalo de confianza bilateral con desviación típica desconocida es $O(n^{-1})$. El orden del error de cobertura para el intervalo de confianza unilateral resulta $O(n^{-\frac{1}{2}})$.

Si el parámetro de interés fuese otro arbitrario: $\theta = \theta(F)$, no necesariamente la media, puede obtenerse, análogamente un intervalo de confianza basado en la normal asintótica:

$$\hat{I}_0 = \left(\hat{\theta} - \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} z_{\alpha/2}, \hat{\theta} + \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} z_{\alpha/2} \right),$$

que está basado en el estadístico

$$R_1 = \sqrt{n} \frac{\hat{\theta} - \theta}{\hat{\sigma}_{\theta}}.$$

De forma análoga a lo ya razonado para la media muestral, puede deducirse que el error de cobertura del intervalo de confianza bilateral tiene un orden de $O(n^{-1})$, mientras que para intervalos unilaterales el orden es $O(n^{-\frac{1}{2}})$.

4.2 Método percentil (básico)

Este método se basa en la construcción del intervalo de confianza, mediante bootstrap, a partir del estadístico no estandarizado

$$R_2 = \sqrt{n} (\hat{\theta} - \theta).$$

Una vez realizado el correspondiente remuestreo (uniforme, suavizado, simetrizado, ...), a partir de cierto estimador, \hat{F} , de la distribución poblacional, F , la distribución en el muestreo de R_2 se aproxima mediante la distribución bootstrap de

$$R_2^* = \sqrt{n} (\hat{\theta}^* - \theta(\hat{F})).$$

Así se obtienen valores $x_{\alpha/2}$ y $x_{1-\alpha/2}$, siendo x_{β} , tal que $P^*(R_2^* \leq x_{\beta}) = \beta$, y a partir de ellos sabemos que

$$\begin{aligned} 1 - \alpha &= 1 - \frac{\alpha}{2} - \frac{\alpha}{2} = P^*(R_2^* < x_{1-\alpha/2}) - P^*(R_2^* \leq x_{\alpha/2}) \\ &= P^*(x_{\alpha/2} < R_2^* < x_{1-\alpha/2}), \end{aligned}$$

con lo cual decimos que también ha de ser aproximadamente igual a $1 - \alpha$ la siguiente probabilidad

$$\begin{aligned} P(x_{\alpha/2} < R_2 < x_{1-\alpha/2}) &= P(x_{\alpha/2} < \sqrt{n} (\hat{\theta} - \theta) < x_{1-\alpha/2}) \\ &= P\left(\hat{\theta} - \frac{x_{1-\alpha/2}}{\sqrt{n}} < \theta < \hat{\theta} - \frac{x_{\alpha/2}}{\sqrt{n}}\right). \end{aligned}$$

Ello da pie a definir el intervalo de confianza bootstrap calculado por el método percentil como

$$\hat{I}_1 = \left(\hat{\theta} - \frac{x_{1-\alpha/2}}{\sqrt{n}}, \hat{\theta} - \frac{x_{\alpha/2}}{\sqrt{n}} \right).$$

Para estudiar el error de cobertura de este intervalo de confianza conviene ver antes qué grado de aproximación existe entre la distribución en el muestreo de R_2 y la distribución bootstrap de R_2^* . A partir del Teorema de Bhattacharya-Ghosh se tiene

$$\begin{aligned} P(R_2 \leq v) &= P\left(\sqrt{n}(\hat{\theta} - \theta) \leq v\right) = P\left(\sqrt{n}\frac{\hat{\theta} - \theta}{\sigma_\theta} \leq \frac{v}{\sigma_\theta}\right) \\ &= \Phi\left(\frac{v}{\sigma_\theta}\right) + O(n^{-\frac{1}{2}}) \\ P^*(R_2^* \leq v) &= P^*\left(\sqrt{n}(\hat{\theta}^* - \theta(\hat{F})) \leq v\right) \\ &= P^*\left(\sqrt{n}\frac{\hat{\theta}^* - \theta(\hat{F})}{\sigma_{\hat{\theta}}} \leq \frac{v}{\sigma_{\hat{\theta}}}\right) = \Phi\left(\frac{v}{\sigma_{\hat{\theta}}}\right) + O_P(n^{-\frac{1}{2}}). \end{aligned}$$

Como consecuencia

$$P^*(R_2^* \leq v) - P(R_2 \leq v) = \Phi\left(\frac{v}{\sigma_{\hat{\theta}}}\right) - \Phi\left(\frac{v}{\sigma_\theta}\right) + O_P(n^{-\frac{1}{2}}) = O_P(n^{-\frac{1}{2}}),$$

ya que, típicamente, $\sigma_{\hat{\theta}} - \sigma_\theta = O_P(n^{-\frac{1}{2}})$. En resumen, la distribución en el muestreo de R_2 y la distribución bootstrap de R_2^* se aproximan, una a la otra, a la velocidad $O_P(n^{-\frac{1}{2}})$, cuando $n \rightarrow \infty$.

El error de cobertura del intervalo de confianza bilateral calculado mediante bootstrap por el método percentil es

$$\begin{aligned} P(\theta \in \hat{I}_1) - (1 - \alpha) &= P(R_2 < x_{1-\alpha/2}) - P(R_2 \leq x_{\alpha/2}) \\ &\quad - [P^*(R_2^* < x_{1-\alpha/2}) - P^*(R_2^* \leq x_{\alpha/2})] \\ &= P(R_2 < x_{1-\alpha/2}) - P^*(R_2^* < x_{1-\alpha/2}) \\ &\quad - [P(R_2 \leq x_{\alpha/2}) - P^*(R_2^* \leq x_{\alpha/2})] \\ &= O_P(n^{-\frac{1}{2}}) \end{aligned}$$

De esta forma el error de cobertura para los intervalos de confianza bilaterales bootstrap obtenidos mediante el método percentil es $O(n^{-\frac{1}{2}})$. Puede deducirse que ese es también el orden para los intervalos unilaterales obtenidos por este método. Así pues el orden del error de cobertura para el método percentil cuando se construyen intervalos de confianza unilaterales coincide con el de los construidos usando la normal asintótica pero el orden del error de cobertura de los intervalos bilaterales bootstrap construidos por el método percentil es peor que el de los basados en la normal asintótica, que es del orden $O(n^{-1})$.

Ejemplo 4.1 (Inferencia sobre la media con varianza desconocida, continuación). Continuando con el ejemplo de los tiempos de vida de microorganismos (sin asumir varianza conocida), supongamos que queremos obtener una estimación por intervalo de confianza de su vida media empleando este método. El código necesario sería muy similar al del Ejemplo 1.5:

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
            0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)
n <- length(muestra)
alfa <- 0.05
x_barra <- mean(muestra)
```

```

# Remuestreo
set.seed(1)
B <- 1000
remuestra <- numeric(n)
x_barra_boot <- numeric(B)
estadistico_boot <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  x_barra_boot[k] <- mean(remuestra)
  estadistico_boot[k] <- sqrt(n) * (x_barra_boot[k] - x_barra)
}

# Aproximación bootstrap de los ptos críticos
pto_crit <- quantile(estadistico_boot, c(alfa/2, 1 - alfa/2))

# Construcción del IC
ic_inf_boot <- x_barra - pto_crit[2]/sqrt(n)
ic_sup_boot <- x_barra - pto_crit[1]/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.4837233 1.1025650

```

Aunque en este caso también podemos obtener el intervalo a partir de las réplicas bootstrap del estimador:

```

pto_crit <- quantile(x_barra_boot, c(alfa/2, 1 - alfa/2))
ic_inf_boot <- 2*x_barra - pto_crit[2]
ic_sup_boot <- 2*x_barra - pto_crit[1]
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.4837233 1.1025650

```

Esta forma de proceder es la que emplea el paquete `boot` para obtener el que denomina intervalo de confianza *bootstrap básico* (estableciendo `type="basic"` en la llamada a la función `boot.ci()`):

```

library(boot)
statistic <- function(data, i) mean(data[i])

set.seed(1)
res.boot <- boot(muestra, statistic, R = 1000)
res <- boot.ci(res.boot, type = "basic")
res

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   ( 0.4825,  1.0980 )
## Calculations and Intervals on Original Scale

```

```
IC_boot <- res$basic[4:5]
IC_boot
```

```
## [1] 0.4824717 1.0980120
```

Además del paquete `boot`, otros autores también denominan a este método *bootstrap básico* (*bootstrap percentil básico* o incluso *bootstrap natural*), y utilizan la terminología *bootstrap percentil* cuando se emplea directamente el estimador como estadístico ($R = \hat{\theta}$) para realizar inferencia. Con el paquete `boot` habrá que establecer `type="perc"` en la llamada a la función `boot.ci()` para obtener el intervalo correspondiente:

```
boot.ci(res.boot, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%    ( 0.5127,  1.1282 )
## Calculations and Intervals on Original Scale
```

En este método se emplean directamente los cuantiles de las réplicas bootstrap del estadístico:

```
# IC_boot <- quantile(res.boot$t, c(alfa/2, 1 - alfa/2)) # type = 7
IC_boot <- quantile(res.boot$t, c(alfa/2, 1 - alfa/2), type = 6)
IC_boot

##      2.5%      97.5%
## 0.5126517 1.1281950
```

Asintóticamente ambos métodos son equivalentes, aunque en general es preferible (evita sesgos) el bootstrap percentil básico.

4.3 Método percentil-*t*

Este método bootstrap, construye un intervalo de confianza bootstrap a partir del estadístico studentizado:

$$R_1 = \sqrt{n} \frac{\hat{\theta} - \theta}{\hat{\sigma}_\theta}.$$

Su distribución en el muestreo se aproxima mediante la distribución bootstrap de

$$R_1^* = \sqrt{n} \frac{\hat{\theta}^* - \theta(\hat{F})}{\hat{\sigma}_\theta^*}.$$

En este caso, los valores $x_{\alpha/2}$ y $x_{1-\alpha/2}$, se obtienen a partir de esta última distribución bootstrap, es decir, x_β se define a partir de $P^*(R_1^* \leq x_\beta) = \beta$. Como

$$1 - \alpha = P^*(x_{\alpha/2} < R_1^* < x_{1-\alpha/2}),$$

razonamos que también ha de ser aproximadamente igual a $1 - \alpha$ la siguiente probabilidad

$$\begin{aligned} P(x_{\alpha/2} < R_1 < x_{1-\alpha/2}) &= P\left(x_{\alpha/2} < \sqrt{n} \frac{\hat{\theta} - \theta}{\hat{\sigma}_\theta} < x_{1-\alpha/2}\right) \\ &= P\left(\hat{\theta} - \frac{\hat{\sigma}_\theta}{\sqrt{n}} x_{1-\alpha/2} < \theta < \hat{\theta} - \frac{\hat{\sigma}_\theta}{\sqrt{n}} x_{\alpha/2}\right). \end{aligned}$$

Con lo cual, el intervalo de confianza bootstrap calculado por el método percentil-*t* se define como

$$\hat{I}_2 = \left(\hat{\theta} - \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} x_{1-\alpha/2}, \hat{\theta} - \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} x_{\alpha/2} \right).$$

Utilizando el Teorema de Bhattacharya-Ghosh puede acotarse el error de aproximación entre la distribución en el muestreo de R_1 y la distribución bootstrap de R_1^* :

$$\begin{aligned} P^*(R_1^* \leq u) - P(R_1 \leq u) &= P^*\left(\sqrt{n}\frac{\hat{\theta}^* - \theta(\hat{F})}{\hat{\sigma}_{\theta}^*} \leq u\right) - P\left(\sqrt{n}\frac{\hat{\theta} - \theta}{\hat{\sigma}_{\theta}} \leq u\right) \\ &= \Phi(u) + n^{-\frac{1}{2}}\hat{q}_1(u)\phi(u) + O_P(n^{-1}) \\ &\quad - [\Phi(u) + n^{-\frac{1}{2}}q_1(u)\phi(u) + O(n^{-1})] \\ &= n^{-\frac{1}{2}}[\hat{q}_1(u) - q_1(u)]\phi(u) + O_P(n^{-1}) \\ &= O_P(n^{-1}), \end{aligned}$$

ya que los coeficientes que aparecen en el polinomio $\hat{q}_1(u)$ son estimadores \sqrt{n} -consistentes de los coeficientes del polinomio $q_1(u)$. Los de éste último dependen de los momentos poblacionales y los del primero son sus correspondientes versiones empíricas.

Así pues, el error de cobertura del intervalo de confianza bootstrap bilateral calculado por el método percentil-*t* es

$$\begin{aligned} P(\theta \in \hat{I}_2) - (1 - \alpha) &= P(R_1 < x_{1-\alpha/2}) - P(R_1 \leq x_{\alpha/2}) \\ &\quad - [P^*(R_1^* < x_{1-\alpha/2}) - P^*(R_1^* \leq x_{\alpha/2})] \\ &= P(R_1 < x_{1-\alpha/2}) - P^*(R_1^* < x_{1-\alpha/2}) \\ &\quad - [P(R_1 \leq x_{\alpha/2}) - P^*(R_1^* \leq x_{\alpha/2})] \\ &= O_P(n^{-1}) \end{aligned}$$

Se tiene entonces que el error de cobertura para los intervalos de confianza bilaterales bootstrap obtenidos mediante el método percentil-*t* es $O(n^{-1})$. Puede deducirse que ese es también el orden para los intervalos unilaterales obtenidos por este método. Así pues el orden del error de cobertura para el método percentil-*t* cuando se construyen intervalos de confianza unilaterales mejora al de los intervalos unilaterales basados en la normal asintótica, con un error de cobertura de orden $O(n^{-\frac{1}{2}})$. En el caso de los intervalos de confianza bilaterales, el orden del error de cobertura usando la normal asintótica o bien el bootstrap por el método percentil-*t* es el mismo, $O(n^{-1})$ en ambos casos.

En el Ejemplo 1.5, se implementó este método para obtener una estimación por intervalo de confianza del tiempo de vida medio de microorganismos. En el Ejemplo 1.6 se mostró como calcular este intervalo empleando el paquete **boot** (haciendo que la función **statistic** devuelva también la varianza del estadístico y estableciendo **type="stud"** en la llamada a la función **boot.ci()**).

4.4 Método percentil-*t* simetrizado

Es un método análogo al percentil-*t*. Sólo difiere de él en la forma de seleccionar los cuantiles de la distribución bootstrap. En lugar de tomar cuantiles que dejen colas iguales ($\frac{\alpha}{2}$ a la izquierda y a la derecha, respectivamente), se eligen los cuantiles de forma que sean simétricos. Así, dado el estadístico R_1 y su versión bootstrap R_1^* , se considera el valor $y_{1-\alpha}$ que cumple $P^*(|R_1^*| \leq y_{1-\alpha}) = 1 - \alpha$. Así se tiene que

$$1 - \alpha = P^*(-y_{1-\alpha} \leq R_1^* \leq y_{1-\alpha}).$$

De esa forma se razona que también ha de ser aproximadamente igual a $1 - \alpha$ la siguiente probabilidad

$$\begin{aligned} P(-y_{1-\alpha} < R_1 < y_{1-\alpha}) &= P\left(-y_{1-\alpha} < \sqrt{n}\frac{\hat{\theta} - \theta}{\hat{\sigma}_{\theta}} < y_{1-\alpha}\right) \\ &= P\left(\hat{\theta} - \frac{\hat{\sigma}_{\theta}}{\sqrt{n}}y_{1-\alpha} < \theta < \hat{\theta} + \frac{\hat{\sigma}_{\theta}}{\sqrt{n}}y_{1-\alpha}\right). \end{aligned}$$

Con lo cual, el intervalo de confianza bootstrap calculado por el método percentil-*t* simetrizado se define como

$$\hat{I}_3 = \left(\hat{\theta} - \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} y_{1-\alpha}, \hat{\theta} + \frac{\hat{\sigma}_{\theta}}{\sqrt{n}} y_{1-\alpha} \right).$$

Utilizando el Teorema de Bhattacharya-Ghosh con desarrollos hasta el orden n^{-1} se tiene:

$$\begin{aligned} P^*(R_1^* \leq u) - P(R_1 \leq u) &= P^*\left(\sqrt{n} \frac{\hat{\theta}^* - \theta(\hat{F})}{\hat{\sigma}_{\theta}^*} \leq u\right) - P\left(\sqrt{n} \frac{\hat{\theta} - \theta}{\hat{\sigma}_{\theta}} \leq u\right) \\ &= \Phi(u) + n^{-\frac{1}{2}} \hat{q}_1(u) \phi(u) + n^{-1} \hat{q}_2(u) \phi(u) + O_P(n^{-\frac{3}{2}}) \\ &\quad - [\Phi(u) + n^{-\frac{1}{2}} q_1(u) \phi(u) + n^{-1} q_2(u) \phi(u) + O(n^{-\frac{3}{2}})] \\ &= n^{-\frac{1}{2}} [\hat{q}_1(u) - q_1(u)] \phi(u) + n^{-1} [\hat{q}_2(u) - q_2(u)] \phi(u) + O_P(n^{-\frac{3}{2}}). \end{aligned}$$

Como consecuencia, el error de cobertura del intervalo de confianza bootstrap bilateral calculado por el método percentil-*t* simetrizado es

$$\begin{aligned} P(\theta \in \hat{I}_3) - (1 - \alpha) &= P(-y_{1-\alpha} < R_1 < y_{1-\alpha}) - P^*(-y_{1-\alpha} < R_1^* < y_{1-\alpha}) \\ &= P(R_1 < y_{1-\alpha}) - P^*(R_1^* < y_{1-\alpha}) - \\ &\quad - [P(R_1 \leq -y_{1-\alpha}) - P^*(R_1^* \leq -y_{1-\alpha})] \\ &= n^{-\frac{1}{2}} [q_1(y_{1-\alpha}) - \hat{q}_1(y_{1-\alpha})] \phi(y_{1-\alpha}) \\ &\quad + n^{-1} [q_2(y_{1-\alpha}) - \hat{q}_2(y_{1-\alpha})] \phi(y_{1-\alpha}) \\ &\quad - n^{-\frac{1}{2}} [q_1(-y_{1-\alpha}) - \hat{q}_1(-y_{1-\alpha})] \phi(-y_{1-\alpha}) \\ &\quad - n^{-1} [q_2(-y_{1-\alpha}) - \hat{q}_2(-y_{1-\alpha})] \phi(-y_{1-\alpha}) + O_P(n^{-\frac{3}{2}}) \\ &= 2n^{-1} [q_2(y_{1-\alpha}) - \hat{q}_2(y_{1-\alpha})] \phi(y_{1-\alpha}) + O_P(n^{-\frac{3}{2}}) \\ &= O_P(n^{-\frac{3}{2}}) \end{aligned}$$

ya que los polinomios $q_1(u)$ y $\hat{q}_1(u)$ son simétricos, $q_2(u)$ y $\hat{q}_2(u)$ son antisimétricos, la función $\phi(u)$ es simétrica y los coeficientes que aparecen en el polinomio $\hat{q}_2(u)$ son estimadores \sqrt{n} -consistentes de los coeficientes del polinomio $q_2(u)$. Como consecuencia, el error de cobertura para los intervalos de confianza bilaterales bootstrap obtenidos mediante el método percentil-*t* simetrizado es $O(n^{-\frac{3}{2}})$. Puede deducirse que el orden para los intervalos unilaterales obtenidos por este método es $O(n^{-1})$. Así pues el orden del error de cobertura para el método percentil-*t* simetrizado cuando se construyen intervalos de confianza unilaterales mejora al de los intervalos unilaterales basados en la normal asintótica, con un error de cobertura de orden $O(n^{-\frac{1}{2}})$, e iguala al orden del error de cobertura de los obtenidos mediante el percentil-*t*. En el caso de los intervalos de confianza bilaterales, el orden del error de cobertura usando el método percentil-*t* simetrizado es $O(n^{-\frac{3}{2}})$, el cual mejora el orden $O(n^{-1})$, que es el que presentan los intervalos basados en la normal asintótica o bien en el método percentil-*t*.

4.5 Tabla resumen de los errores de cobertura

Tipo de I.C.	Unilateral	Bilateral
Percentil	$O_P(n^{-\frac{1}{2}})$	$O_P(n^{-\frac{1}{2}})$
Percentil- <i>t</i>	$O_P(n^{-1})$	$O_P(n^{-1})$
Percentil- <i>t</i> simetrizado	$O_P(n^{-1})$	$O_P(n^{-\frac{3}{2}})$

4.6 Ejemplos

4.6.1 IC bootstrap para la media mediante el método percentil-*t* simetrizado

Continuando con el ejemplo de los tiempos de vida de microorganismos (sin asumir varianza conocida), para obtener una estimación por intervalo de confianza de su vida media empleando el método bootstrap percentil-*t* simetrizado se podría utilizar (ver Ejemplo 1.5):

```
muestra <- c(0.143, 0.182, 0.256, 0.26, 0.27, 0.437, 0.509,
            0.611, 0.712, 1.04, 1.09, 1.15, 1.46, 1.88, 2.08)

n <- length(muestra)
alfa <- 0.05
x_barra <- mean(muestra)
cuasi_dt <- sd(muestra)

# Remuestreo
set.seed(1)
B <- 1000
remuestra <- numeric(n)
estadistico_boot <- numeric(B)
for (k in 1:B) {
  remuestra <- sample(muestra, n, replace = TRUE)
  x_barra_boot <- mean(remuestra)
  cuasi_dt_boot <- sd(remuestra)
  estadistico_boot[k] <- sqrt(n) * abs(x_barra_boot - x_barra)/cuasi_dt_boot
}

# Aproximación bootstrap del pto crítico
pto_crit <- quantile(estadistico_boot, 1 - alfa)

# Construcción del IC
ic_inf_boot <- x_barra - pto_crit * cuasi_dt/sqrt(n)
ic_sup_boot <- x_barra + pto_crit * cuasi_dt/sqrt(n)
IC_boot <- c(ic_inf_boot, ic_sup_boot)
names(IC_boot) <- paste0(100*c(alfa/2, 1-alfa/2), "%")
IC_boot

##      2.5%    97.5%
## 0.4334742 1.1771924
```

4.6.2 Estudio de simulación

El siguiente código permite realizar estudios de simulación comparando las probabilidades de cobertura y las longitudes de los intervalos de confianza clásicos (basados en normalidad), bootstrap percentil, bootstrap percentil-*t* y bootstrap percentil-*t* simetrizado para la media, en el caso de muestras procedentes de una distribución $\exp(\lambda)$. En este caso se obtienen las estimaciones Monte Carlo a partir de 500 simulaciones con $\lambda = 0.01$, tamaño muestral $n = 100$ y $B = 1000$ réplicas bootstrap para un nivel de confianza nominal del 90% ($\alpha = 0.10$).

```
t.ini <- proc.time()
rate <- 0.01
mu <- 1/rate
n <- 100

alfa <- 0.1
namesI <- paste0(100*c(alfa/2, 1-alfa/2), "%")
```

```

B <- 1000
percentil <- numeric(B)
percentilt <- numeric(B)
percentilts <- numeric(B)

nsim <- 500
resultados <- array(dim = c(nsim, 2, 4))
dimnames(resultados) <- list(NULL, c("Cobertura", "Longitud"),
                           c("Normal", "Percentil", "Percentil-t", "Percentil-t simetrizado"))
# Bucle simulación
set.seed(1)
for (isim in 1:nsim) {
  # Aproximación clásica
  muestra <- rexp(n, rate = 0.01)
  media <- mean(muestra)
  desv <- sd(muestra)
  z <- qnorm(1 - alfa/2)
  ic_inf <- media - z*desv/sqrt(n)
  ic_sup <- media + z*desv/sqrt(n)
  I0 <- c(ic_inf, ic_sup)
  # names(I0) <- namesI
  resultados[isim, 1, 1] <- (I0[1] < mu) && (mu < I0[2])
  resultados[isim, 2, 1] <- I0[2] - I0[1]

  # Remuestreo bootstrap
  for (k in 1:B) {
    remuestra <- sample(muestra, n, replace = TRUE)
    percentil[k] <- sqrt(n) * (mean(remuestra) - media)
    percentilt[k] <- percentil[k]/sd(remuestra)
    percentilts[k] <- abs(percentilt[k])
  }

  # Aproximación bootstrap percentil
  pto_crit <- quantile(percentil, c(alfa/2, 1 - alfa/2))
  # Construcción del IC
  ic_inf_boot <- media - pto_crit[2]/sqrt(n)
  ic_sup_boot <- media - pto_crit[1]/sqrt(n)
  I1 <- c(ic_inf_boot, ic_sup_boot)
  # names(I1) <- namesI
  resultados[isim, 1, 2] <- (I1[1] < mu) && (mu < I1[2])
  resultados[isim, 2, 2] <- I1[2] - I1[1]

  # Aproximación bootstrap percentil-t
  pto_crit <- quantile(percentilt, c(alfa/2, 1 - alfa/2))
  # Construcción del IC
  ic_inf_boot <- media - pto_crit[2] * desv/sqrt(n)
  ic_sup_boot <- media - pto_crit[1] * desv/sqrt(n)
  I2 <- c(ic_inf_boot, ic_sup_boot)
  # names(I2) <- namesI
  resultados[isim, 1, 3] <- (I2[1] < mu) && (mu < I2[2])
  resultados[isim, 2, 3] <- I2[2] - I2[1]

  # Aproximación bootstrap percentil-t simetrizado
  pto_crit <- quantile(percentilts, 1 - alfa)
  # Construcción del IC
  ic_inf_boot <- media - pto_crit * desv/sqrt(n)
}

```

```

ic_sup_boot <- media + pto_crit * desv/sqrt(n)
I3 <- c(ic_inf_boot, ic_sup_boot)
# names(I3) <- namesI
resultados[isim, 1, 4] <- (I3[1] < mu) && (mu < I3[2])
resultados[isim, 2, 4] <- I3[2] - I3[1]
}

t.fin <- proc.time() - t.ini
t.fin

##    user  system elapsed
##  15.87   0.02  16.27
apply(resultados, c(2, 3), mean)

##          Normal Percentil Percentil-t Percentil-t simetrizado
## Cobertura  0.8800   0.86600     0.88800             0.88800
## Longitud   32.5022  32.13928     33.5653            33.49959
# knitr::kable(t(apply(resultados, c(2, 3), mean)), digits = 3)

```

Aproximación	Cobertura	Longitud
Normal	0.892	32.243
Percentil	0.886	32.051
Percentil-t	0.912	33.395
Percentil-t simetrizado	0.904	33.342

La siguiente tabla recoge las probabilidades de cobertura, estimadas por Monte Carlo, en una ejecución con tamaño muestral $n = 100$, $N = 10000$ trials y $B = 1000$ réplicas bootstrap para un nivel de confianza nominal del 90% ($\alpha = 0.10$).

Aproximación	Cobertura IC
Normal	88.60%
Boot. percentil	88.60%
Boot. percentil-t	89.76%
Boot. percentil-t simetrizado	89.46%

En la Sección B.3.2 del Apéndice B se incluye un estudio similar, empleando computación en paralelo para comparar las probabilidades de cobertura y las longitudes de los intervalos de confianza implementados en la función `boot.ci()`.

4.6.3 IC bootstrap para el coeficiente de correlación

Supongamos que queremos estudiar la correlación entre dos variables X e Y a partir del coeficiente de correlación lineal de Pearson:

$$\rho = \frac{\text{Cov}(X, Y)}{\sigma(X)\sigma(Y)},$$

cuyo estimador natural es el coeficiente de correlación muestral:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}},$$

que podemos calcular en R empleando la función `cor()`.

Para realizar inferencias sobre el coeficiente de correlación, como aproximación más simple, se puede considerar que la distribución muestral de r es aproximadamente normal y emplear el estadístico:

$$\frac{r - \rho}{\sqrt{\frac{1-r^2}{n-2}}} \underset{aprox}{\sim} t_{n-2} \quad (4.1)$$

Pero esta aproximación solo sería válida en el caso de muestras grandes (o si la distribución bivariante de (X, Y) es aproximadamente normal) cuando la correlación entre las variables es débil o moderada. En caso contrario la distribución muestral de r puede ser muy asimétrica y los resultados obtenidos con el estadístico anterior no ser muy adecuados (esto concuerda con lo observado en la Sección 1.4.2, al emplear bootstrap uniforme multidimensional para hacer inferencia sobre $R = r - \rho$). Para evitar este problema se suelen obtener intervalos de confianza para ρ empleando la transformación Z de Fisher (1915):

$$Z = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) = \operatorname{arctanh}(r),$$

que es una transformación (aprox.) normalizadora y estabilizadora de la varianza. Suponiendo que (X, Y) es normal bivariante y que hay independencia entre las observaciones:

$$Z \sim \mathcal{N} \left(\frac{1}{2} \ln \left(\frac{1+\rho}{1-\rho} \right), \frac{1}{n-3} \right).$$

El intervalo de confianza asintótico se obtiene empleando la aproximación normal tradicional en la escala Z y aplicando posteriormente la transformación inversa:

$$r = \frac{\exp(2Z) - 1}{\exp(2Z) + 1} = \tanh(Z).$$

Esta aproximación está implementada en la función `cor.test()` del paquete base `stat` de R¹, además de que también realiza el contraste $H_0 : \rho = 0$ empleando el estadístico (4.1).

Continuando con el ejemplo de la Sección 1.4.2, para obtener un intervalo de confianza para el coeficiente de correlación lineal entre las variables `income` y `prestige` del conjunto de datos `Prestige`, podríamos emplear el siguiente código:

```
data(Prestige, package="carData")
# with(Prestige, cor.test(income, prestige))
cor.test(Prestige$income, Prestige$prestige)

##
## Pearson's product-moment correlation
##
## data: Prestige$income and Prestige$prestige
## t = 10.224, df = 100, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6044711 0.7983807
## sample estimates:
##      cor
## 0.7149057
```

La función `boot.ci()` del paquete `boot` también permite obtener intervalos de confianza calculados en una escala transformada del estadístico, mediante los parámetros:

- `h`: función vectorial que define la transformación. Los intervalos se calculan en la escala de $h(t)$ y se aplica la función inversa (si se especifica) para transformarlos a la escala original.
- `hinv`: (opcional) función inversa de la transformación (si no se especifica solo se calculan los intervalos en la escala transformada).

¹Se puede obtener el código tecleando en la consola `stats:::cor.test.default`.

- `hdot`: (opcional) función derivada de la transformación (empleada por algunos métodos para aproximar la varianza en la escala transformada mediante el método delta).

Por ejemplo, para considerar la transformación Z de Fisher en este caso, se podría emplear el siguiente código:

```
library(boot)

statistic <- function(data, i){
  remuestra <- data[i, ]
  cor(remuestra$income, remuestra$prestige)
}

set.seed(1)
res.boot <- boot(Prestige, statistic, R = 1000)

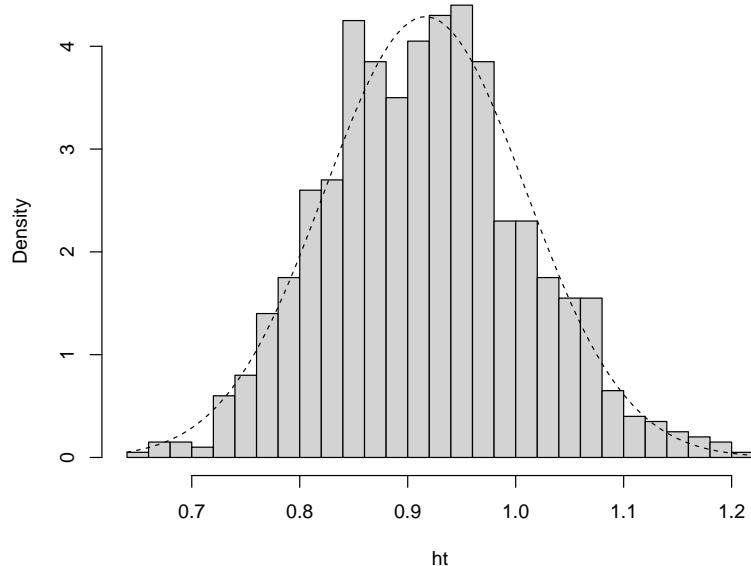
h <- function(t) atanh(t)
hdot <- function(t) 1/(1 - t^2)
hinv <- function(t) tanh(t)

# boot.ci(res.boot, type = "norm", h = h)
# boot.ci(res.boot, type = "norm", h = h, hinv = hinr)
boot.ci(res.boot, type = "norm", h = h, hdot = hdot, hinr = hinr)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = "norm", h = h, hdot = hdot,
##          hinr = hinr)
##
## Intervals :
## Level      Normal
## 95%   ( 0.6016,  0.7858 )
## Calculations on Transformed Scale;  Intervals on Original Scale
```

Esto sería en principio preferible a trabajar en la escala original, ya que la distribución bootstrap en la escala transformada se aproximaría más a la normalidad:

```
ht <- h(res.boot$t)
hist(ht, freq = FALSE, breaks = "FD",
     main = "Distribución bootstrap en la escala transformada")
curve(dnorm(x, mean=mean(ht), sd=sd(ht)), lty = 2, add = TRUE)
```

Distribución bootstrap en la escala transformada

Capítulo 5

Aplicaciones del Bootstrap en contrastes de hipótesis

El objetivo de los contrastes de hipótesis es, a partir de la información que proporciona una muestra, decidir (tratando de controlar el riesgo de equivocarse al no disponer de toda la información) entre dos hipótesis sobre alguna característica de interés de la población: hipótesis nula (H_0) e hipótesis alternativa (H_1).

Entre los distintos tipos de contrastes de hipótesis (e.g. paramétricos, no paramétricos, ...), nos centraremos principalmente en los contrastes de bondad de ajuste. En este caso interesará distinguir principalmente entre hipótesis nulas simples (especifican un único modelo) y compuestas (especifican un conjunto/familia de modelos).

Para realizar el contraste se emplea un estadístico $D(X_1, \dots, X_n; H_0)$, que mide la discrepancia entre la muestra observada y la hipótesis nula, con distribución conocida (o que se puede aproximar) bajo H_0 . Por ejemplo, en el caso de una hipótesis nula paramétrica es habitual emplear un estadístico estandarizado de la forma:

$$D(X_1, \dots, X_n; H_0) = \frac{\hat{\theta} - \theta_0}{\hat{\sigma}_{\hat{\theta}}}$$

(o algún tipo de razón de verosimilitudes).

La regla de decisión depende de la hipótesis alternativa y del riesgo asumible al rechazar H_0 siendo cierta:

$$P(\text{rechazar } H_0 \mid H_0 \text{ cierta}) = \alpha,$$

denominado nivel de significación. Se determina una región de rechazo (RR) a partir de los valores que tiende a tomar el estadístico cuando H_1 es cierta, de forma que¹:

$$P(D \in RR \mid H_0 \text{ cierta}) = \alpha.$$

Se rechaza la hipótesis nula cuando el valor observado del estadístico $\hat{d} = D(x_1, \dots, x_n; H_0)$ pertenece a la región de rechazo.

Para medir el nivel de evidencia en contra de H_0 se emplea el *p*-valor del contraste (también denominado valor crítico o tamaño del contraste), el menor valor del nivel de significación para el que se rechaza H_0 (que se puede interpretar también como la probabilidad de obtener una discrepancia mayor o igual que \hat{d} cuando H_0 es cierta).

El cálculo del *p*-valor dependerá por tanto de la hipótesis alternativa. Por ejemplo, si el estadístico del contraste tiende a tomar valores grandes cuando H_0 es falsa (contraste unilateral derecho):

$$p = P(D \geq \hat{d} \mid H_0).$$

¹Aunque cuando la hipótesis nula es compuesta: $P(D \in RR \mid H_0 \text{ cierta}) \leq \alpha$.

En otros casos (contrastos bilaterales) hay evidencias en contra de H_0 si el estadístico toma valores significativamente grandes o pequeños. En estos casos la distribución del estadístico del contraste bajo H_0 suele ser simétrica en torno al cero, por lo que:

$$p = 2P(D \geq |\hat{d}| \mid H_0).$$

Pero si esta distribución es asimétrica:

$$p = 2 \min \left\{ P(D \leq \hat{d} \mid H_0), P(D \geq \hat{d} \mid H_0) \right\}.$$

La regla de decisión a partir del p -valor es siempre la misma. Rechazamos H_0 , al nivel de significación α , si $p \leq \alpha$, en cuyo caso se dice que el contraste es estadísticamente significativo (rechazamos H_0 con mayor seguridad cuanto más pequeño es el p -valor). Por tanto, la correspondiente variable aleatoria \mathcal{P} debería verificar:

$$P(\mathcal{P} \leq \alpha \mid H_0) = \alpha.$$

Es decir, la distribución del p -valor bajo H_0 debería ser $\mathcal{U}(0, 1)$ (si la distribución del estadístico del contraste es continua).

5.1 Aproximación del p-valor mediante remuestreo

En los métodos tradicionales de contrastes de hipótesis se conoce o se puede aproximar la distribución del estadístico del contraste bajo H_0 . Muchas de estas aproximaciones están basadas en resultados asintóticos y pueden no ser adecuadas para tamaños muestrales pequeños. En ese caso, o si no se dispone de estas herramientas, se puede recurrir a métodos de remuestreo para aproximar el p -valor. Uno de los procedimientos más antiguos es el denominado *contraste de permutaciones* (Fisher, 1935; Pitman, 1937; Welch, 1937). Aunque el bootstrap paramétrico y el semiparamétrico son los procedimientos de remuestreo más empleados para aproximar la distribución del estadístico de contraste bajo la hipótesis nula.

La idea es obtener remuestras de una aproximación de la distribución del estadístico bajo H_0 . En el bootstrap paramétrico y semiparamétrico se estima la distribución de los datos bajo la hipótesis nula, \hat{F}_0 , y se obtienen réplicas del estadístico a partir de remuestras de esta distribución (no sería adecuado emplear directamente la distribución empírica). En el caso de los contrastes de permutaciones las remuestras se obtienen directamente de los datos, remuestreando sin reemplazamiento los valores de la respuesta (y manteniendo fijas las covariables).

Finalmente, se emplean las réplicas bootstrap del estadístico d_1^*, \dots, d_B^* para aproximar el p -valor. Por ejemplo, en el caso de un contraste unilateral en el que el estadístico del contraste tiende a tomar valores grandes si la hipótesis nula es falsa, se podría emplear como aproximación:

$$p_{boot} = \frac{1}{B} \# \{d_i^* \geq \hat{d}\}.$$

Mientras que en el caso bilateral, asumiendo que la distribución del estadístico no es necesariamente simétrica, habría que emplear:

$$p_{boot} = \frac{2}{B} \min \left(\# \{d_i^* \leq \hat{d}\}, \# \{d_i^* \geq \hat{d}\} \right).$$

5.2 Contrastos bootstrap paramétricos

En los casos en los que la hipótesis nula especifica por completo la distribución (hipótesis nula simple) o solo desconocemos los valores de algunos parámetros (hipótesis nula paramétrica compuesta) podemos emplear bootstrap paramétrico para obtener las remuestras bootstrap de los datos (realmente en el primer caso se trataría de simulaciones Monte Carlo). Siempre hay que tener en cuenta que las réplicas bootstrap del estadístico se deberían obtener empleando el mismo procedimiento utilizado en la muestra (p.e. reestimando los parámetros si es el caso).

5.2.1 Ejemplo: contraste de Kolmogorov-Smirnov

Se trata de un contraste de bondad de ajuste (similar a la prueba de Cramer-von Mises o a la de Anderson-Darling, implementadas en el paquete `goftest` de R, que son en principio mejores). A partir de X_1, \dots, X_n m.a.s. de X con función de distribución F , se pretende contrastar:

$$\begin{cases} H_0 : F = F_0 \\ H_1 : F \neq F_0 \end{cases}$$

siendo F_0 una función de distribución continua. El estadístico empleado para ello compara la función de distribución bajo H_0 (F_0) con la empírica (F_n):

$$\begin{aligned} D_n &= \sup_x |F_n(x) - F_0(x)| \\ &= \max_{1 \leq i \leq n} \{|F_n(X_{(i)}) - F_0(X_{(i)})|, |F_n(X_{(i-1)}) - F_0(X_{(i)})|\} \\ &= \max_{1 \leq i \leq n} \left\{ \frac{i}{n} - F_0(X_{(i)}), F_0(X_{(i)}) - \frac{i-1}{n} \right\} \\ &= \max_{1 \leq i \leq n} \{D_{n,i}^+, D_{n,i}^-\}, \end{aligned}$$

y su distribución bajo H_0 no depende F_0 (es de distribución libre), si H_0 es simple y F_0 es continua. Esta distribución está tabulada (para tamaños muestrales grandes se utiliza la aproximación asintótica) y se rechaza H_0 si el valor observado d del estadístico es significativamente grande:

$$p = P(D_n \geq d) \leq \alpha.$$

Este método está implementado en la función `ks.test()` del paquete base de R:

```
ks.test(x, y, ...)
```

donde `x` es un vector que contiene los datos, `y` es una función de distribución (o una cadena de texto que la especifica; también puede ser otro vector de datos para el contraste de dos muestras) y `...` representa los parámetros de la distribución.

Si H_0 es compuesta, el procedimiento habitual es estimar los parámetros desconocidos por máxima verosimilitud y emplear \hat{F}_0 en lugar de F_0 . Sin embargo, al proceder de esta forma es de esperar que \hat{F}_0 se aproxime más que F_0 a la distribución empírica, por lo que los cuantiles de la distribución de D_n pueden ser demasiado conservativos (los p -valores tenderán a ser mayores de lo que deberían) y se tenderá a aceptar la hipótesis nula.

Para evitar este problema, en el caso de contrastar normalidad se desarrolló el test de Lilliefors, implementado en la función `lillie.test()` del paquete `nortest` (también hay versiones en este paquete para los métodos de Cramer-von Mises y Anderson-Darling). Como ejemplo analizaremos el comportamiento de ambos métodos para contrastar normalidad considerando 1000 pruebas con muestras de tamaño 30 de una $\mathcal{N}(0, 1)$ (estudiaremos el *tamaño de los contrastes*).

```
# Valores iniciales
library(nortest)
set.seed(1)
nx <- 30
mx <- 0
sx <- 1
nsim <- 1000
# Realizar contrastes
pvalor.ks <- numeric(nsim)
pvalor.lil <- numeric(nsim)
for(isim in 1:nsim) {
  rx <- rnorm(nx, mx, sx)
  pvalor.ks[isim] <- ks.test(rx, "pnorm", mean(rx), sd(rx))$p.value
  pvalor.lil[isim] <- lillie.test(rx)$p.value
}
```

Bajo la hipótesis nula el p -valor debería de seguir una distribución uniforme, por lo que podríamos generar el correspondiente histograma para estudiar el tamaño del contraste. Alternativamente podríamos representar su función de distribución empírica, que se correspondería con la proporción de rechazos para los distintos niveles de significación.

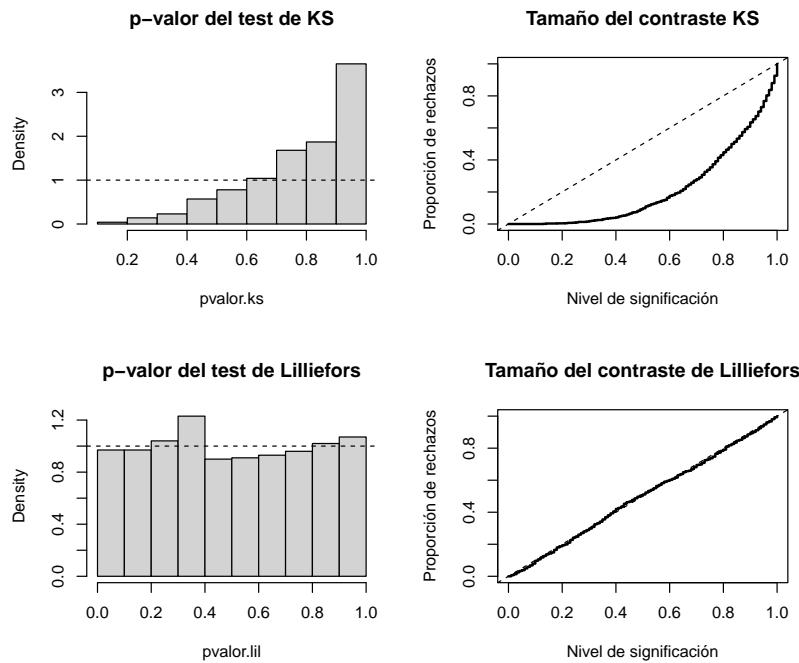
```
old.par <- par(mfrow=c(2, 2))

# Histograma
hist(pvalor.ks, freq=FALSE, main = "p-valor del test de KS")
abline(h=1, lty=2) # curve(dunif(x,0,1), add=TRUE)

# Distribución empírica
curve(ecdf(pvalor.ks))(x), type = "s", lwd = 2,
      main = 'Tamaño del contraste KS', ylab = 'Proporción de rechazos',
      xlab = 'Nivel de significación')
abline(a=0, b=1, lty=2) # curve(punif(x, 0, 1), add = TRUE)

# Histograma
hist(pvalor.lil, freq=FALSE, main = "p-valor del test de Lilliefors")
abline(h=1, lty=2) # curve(dunif(x,0,1), add=TRUE)

# Distribución empírica
curve(ecdf(pvalor.lil))(x), type = "s", lwd = 2,
      main = 'Tamaño del contraste de Lilliefors', ylab = 'Proporción de rechazos',
      xlab = 'Nivel de significación')
abline(a=0, b=1, lty=2) # curve(punif(x, 0, 1), add = TRUE)
```



```
par(old.par)
```

En el caso del contraste de Kolmogorov-Smirnov (KS) se observa que el p -valor tiende a tomar valores grandes y por tanto se rechaza la hipótesis nula muchas menos veces de las que se debería.

En el caso de otras distribuciones se puede emplear bootstrap paramétrico para aproximar la distribución del estadístico del contraste. Es importante recordar que el bootstrap debería imitar el procedimiento empleado sobre la muestra, por lo que en este caso también habría que estimar los parámetros en cada remuestra (en caso contrario aproximariamos la distribución de D_n).

Por ejemplo, la siguiente función implementaría el contraste KS de bondad de ajuste de una variable exponencial aproximando el p -valor mediante bootstrap paramétrico:

```

ks.exp.boot <- function(x, nboot = 10^3) {
  DNAME <- deparse(substitute(x))
  METHOD <- "Kolmogorov-Smirnov Test of pexp by bootstrap"
  n <- length(x)
  RATE <- 1/mean(x)
  ks.exp.stat <- function(x, rate = 1/mean(x)) { # se estima el parámetro
    DMinus <- pexp(sort(x), rate=rate) - (0:(n - 1))/n
    DPlus <- 1/n - DMinus
    Dn = max(c(DMinus, DPlus))
  }
  STATISTIC <- ks.exp.stat(x, rate = RATE)
  names(STATISTIC) <- "Dn"
  # PVAL <- 0
  # for(i in 1:nboot) {
  #   rx <- rexp(n, rate = RATE)
  #   if (STATISTIC <= ks.exp.stat(rx)) PVAL <- PVAL + 1
  # }
  # PVAL <- PVAL/nboot
  # PVAL <- (PVAL + 1)/(nboot + 1) # Alternativa para aproximar el p-valor
  rx <- matrix(rexp(n*nboot, rate = RATE), ncol=n)
  PVAL <- mean(STATISTIC <= apply(rx, 1, ks.exp.stat))
  return(structure(list(statistic = STATISTIC, alternative = "two.sided",
                        p.value = PVAL, method = METHOD, data.name = DNAME),
                  class = "htest"))
}

```

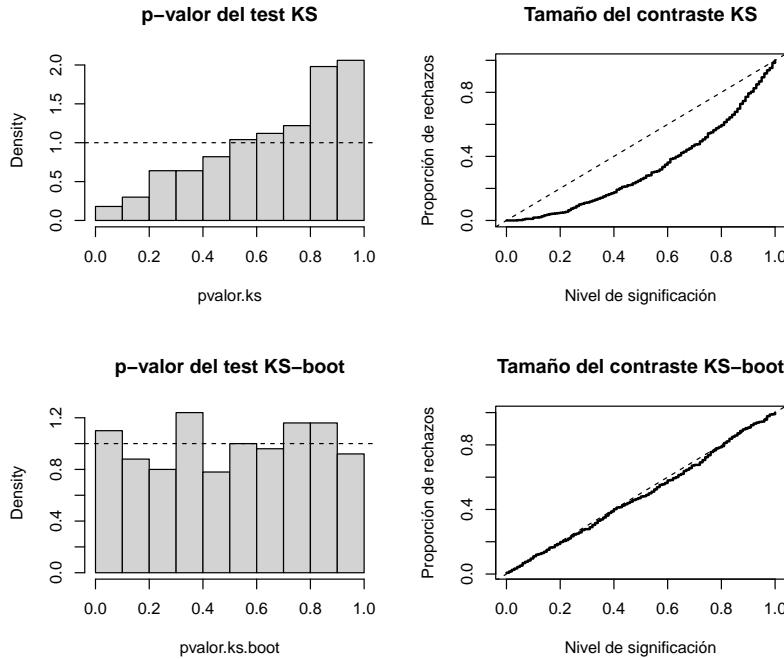
Como ejemplo estudiaremos el caso de contrastar una distribución exponencial considerando 500 pruebas con muestras de tamaño 30 de una $Exp(1)$ y 200 réplicas bootstrap (para disminuir el tiempo de computación).

```

# Valores iniciales
set.seed(1)
nx <- 30
ratex <- 1
nsim <- 500
# Realizar contrastes
pvalor.ks <- numeric(nsim)
pvalor.ks.boot <- numeric(nsim)
for(isim in 1:nsim) {
  rx <- rexp(nx, ratex)
  pvalor.ks[isim] <- ks.test(rx, "pexp", 1/mean(rx))$p.value
  pvalor.ks.boot[isim] <- ks.exp.boot(rx, nboot = 200)$p.value
}
# Generar gráficos
old.par <- par(mfrow=c(2, 2))
# Histograma
hist(pvalor.ks, freq=FALSE, main = "p-valor del test KS")
abline(h=1, lty=2) # curve(dunif(x,0,1), add=TRUE)
# Distribución empírica
curve(ecdf(pvalor.ks)(x), type = "s", lwd = 2,
      main = 'Tamaño del contraste KS', ylab = 'Proporción de rechazos',
      xlab = 'Nivel de significación')
abline(a=0, b=1, lty=2) # curve(punif(x, 0, 1), add = TRUE)
# Histograma
hist(pvalor.ks.boot, freq=FALSE, main = "p-valor del test KS-boot")
abline(h=1, lty=2) # curve(dunif(x,0,1), add=TRUE)
# Distribución empírica

```

```
curve(ecdf(pvalor.ks.boot))(x), type = "s", lwd = 2,
      main = 'Tamaño del contraste KS-boot', ylab = 'Proporción de rechazos',
      xlab = 'Nivel de significación')
abline(a=0, b=1, lty=2) # curve(punif(x, 0, 1), add = TRUE)
```



```
par(old.par)
```

5.3 Contrastes de permutaciones

Supongamos que a partir de una muestra $\{(\mathbf{X}_i, Y_i) : i = 1, \dots, n\}$ estamos interesados en contrastar la hipótesis nula de independencia entre \mathbf{X} e Y :

$$H_0 : F_{Y|\mathbf{X}} = F_Y$$

o equivalentemente que \mathbf{X} no influye en la distribución de Y .

En este caso los valores de la respuesta serían intercambiables bajo la hipótesis nula, por lo que podríamos obtener las remuestras manteniendo fijos los valores² \mathbf{X}_i y permutando los Y_i . Es decir:

1. Generar Y_i^* , con $i = 1, \dots, n$, mediante muestreo sin reemplazamiento de $\{Y_i : i = 1, \dots, n\}$.
2. Considerar la remuestra bootstrap $\{(\mathbf{X}_i, Y_i^*) : i = 1, \dots, n\}$.

Se pueden realizar contrastes de este tipo con el paquete `boot` estableciendo el parámetro `sim = "permutation"` al llamar a la función `boot()` (el argumento `i` de la función `statistic` contendrá permutaciones del vector de índices). Puede ser también de interés el paquete `coin`, que implementa muchos contrastes de este tipo.

5.3.1 Ejemplo: Inferencia sobre el coeficiente de correlación lineal

En esta sección consideraremos como ejemplo el conjunto de datos `dogs` del paquete `boot`, que contiene observaciones sobre el consumo de oxígeno cardíaco (`mvo`) y la presión ventricular izquierda (`1vp`) de 7 perros domésticos.

²Nótese que no se hace ninguna suposición sobre el tipo de covariables, podrían ser categóricas, numéricas o una combinación de ambas.

```
library(boot)
data('dogs', package = "boot")
# plot(dogs)
```

Supongamos que estamos interesados en estudiar la correlación lineal entre las variables `mvo` (X) y `lvp` (Y). Para ello podemos considerar el coeficiente de correlación lineal de Pearson:

$$\rho = \frac{\text{Cov}(X, Y)}{\sigma(X)\sigma(Y)}$$

Su estimador es el coeficiente de correlación muestral:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}},$$

que podemos calcular en R empleando la función `cor()`:

```
cor(dogs$mvo, dogs$lvp)
```

```
## [1] 0.8536946
# with(dogs, cor(mvo, lvp))
```

Para realizar inferencias sobre ρ podemos emplear la función `cor.test()`:

```
cor.test(dogs$mvo, dogs$lvp)
```

```
##
## Pearson's product-moment correlation
##
## data: dogs$mvo and dogs$lvp
## t = 3.6655, df = 5, p-value = 0.01451
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.2818014 0.9780088
## sample estimates:
##      cor
## 0.8536946
# with(dogs, cor.test(mvo, lvp))
```

Esta función realiza el contraste $H_0 : \rho = 0$ empleando el estadístico:

$$\frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \underset{\text{aprox}}{\sim} t_{n-2},$$

bajo la hipótesis nula de que la verdadera correlación es cero. Alternativamente se pueden realizar contrastes unilaterales estableciendo el parámetro `alternative` igual a `"less"` o `"greater"`. Por ejemplo, para contrastar $H_0 : \rho \leq 0$ podríamos emplear:

```
cor.test(dogs$mvo, dogs$lvp, alternative = "greater")
```

```
##
## Pearson's product-moment correlation
##
## data: dogs$mvo and dogs$lvp
## t = 3.6655, df = 5, p-value = 0.007255
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
## 0.4195889 1.0000000
## sample estimates:
```

```
##      cor
## 0.8536946
```

Para realizar el contraste con la función `boot` podríamos emplear el siguiente código:

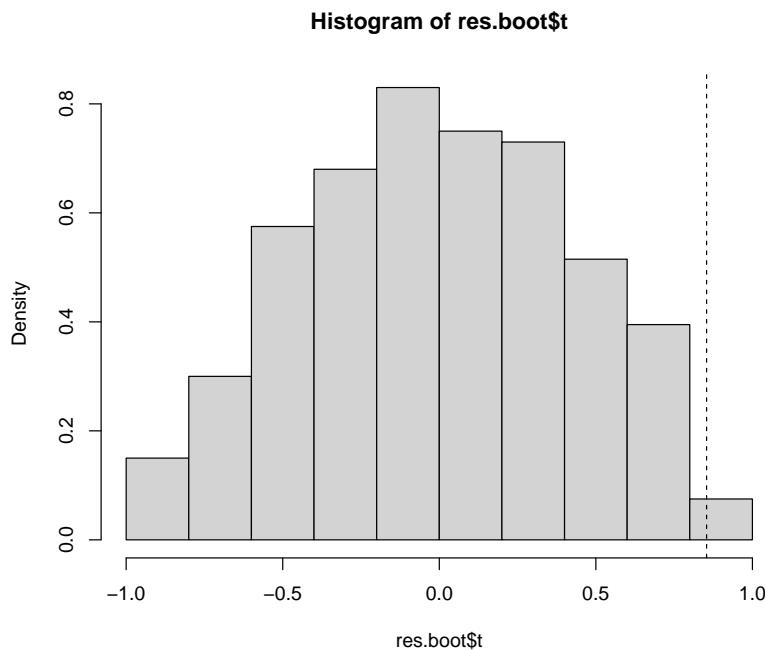
```
library(boot)

statistic <- function(data, i) cor(data$mvo, data$lvp[i])

set.seed(1)
res.boot <- boot(dogs, statistic, R = 1000, sim = "permutation")
# res.boot
```

Posteriormente emplearíamos las réplicas (almacenadas en `res.boot$t`) y el valor observado del estadístico del contraste (almacenado en `res.boot$t0`) para aproximar el *p*-valor:

```
hist(res.boot$t, freq = FALSE)
abline(v = res.boot$t0, lty = 2)
```



Por ejemplo, para el contraste unilateral $H_0 : \rho \leq 0$ (`alternative = "greater"`), obtendríamos:

```
pval.greater <- mean(res.boot$t >= res.boot$t0)
pval.greater
```

```
## [1] 0.009
```

Mientras que para realizar el contraste bilateral $H_0 : \rho = 0$ (`alternative = "two.sided"`), sin asumir que la distribución del estadístico de contraste es simétrica:

```
pval.less <- mean(res.boot$t <= res.boot$t0)
pval <- 2*min(pval.less, pval.greater)
pval
```

```
## [1] 0.018
```

5.4 Contrastes bootstrap semiparamétricos

Este tipo de aproximación se emplearía en el caso de que la hipótesis nula (o la alternativa) especifique un modelo semiparamétrico, con una componente paramétrica y otra no paramétrica. Típicamente se incluye el error en la componente no paramétrica, y podríamos emplear el bootstrap residual (también denominado semiparamétrico o basado en modelos) descrito en la Sección 3.7.2.

En esta sección consideraremos como ejemplo el conjunto de datos `Prestige` del paquete `carData`, considerando como variable respuesta `prestige` (puntuación de ocupaciones obtenidas a partir de una encuesta) y como variables explicativas: `income` (media de ingresos en la ocupación) y `education` (media de los años de educación).

```
data(Prestige, package = "carData")
# ?Prestige
```

5.4.1 Ejemplo: Inferencia sobre modelos de regresión

En la mayoría de los casos nos interesa contrastar un **modelo reducido** frente a un **modelo completo** (que generaliza el modelo reducido). Por ejemplo, en el caso de modelos lineales (estimados por mínimos cuadrados) se dispone del test F para realizar los contrastes de este tipo, que emplea el estadístico:

$$F = \frac{n - q}{q - q_0} \frac{RSS_0 - RSS}{RSS},$$

siendo n el número de observaciones, RSS y q la suma de cuadrados residual y el número de parámetros distintos del modelo completo, y RSS_0 y q_0 los correspondientes al modelo reducido. Este estadístico sigue una distribución $\mathcal{F}_{q-q_0, n-q}$ bajo H_0 y las hipótesis habituales del modelo lineal (ε_i i.i.d. $\mathcal{N}(0, \sigma^2)$).

El contraste de regresión sería un caso particular. Por ejemplo, para contrastar si `income` y `education` influyen linealmente en `prestige` podemos emplear el siguiente código:

```
modelo <- lm(prestige ~ income + education, data = Prestige)
summary(modelo)

##
## Call:
## lm(formula = prestige ~ income + education, data = Prestige)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.4040  -5.3308   0.0154   4.9803  17.6889
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.8477787  3.2189771 -2.127  0.0359 *
## income       0.0013612  0.0002242  6.071 2.36e-08 ***
## education    4.1374444  0.3489120 11.858 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 99 degrees of freedom
## Multiple R-squared:  0.798, Adjusted R-squared:  0.7939
## F-statistic: 195.6 on 2 and 99 DF,  p-value: < 2.2e-16
```

También podemos obtener el valor observado del estadístico F a partir de los resultados del método `summary.lm()`:

```
res <- summary(modelo)
# names(res)
stat <- res$fstatistic[1]
df <- res$fstatistic[2]
```

```
dfr <- res$fstatistic[3]
res$fstatistic

##      value    numdf    dendf
## 195.5505   2.0000 99.0000
```

o haciendo los cálculos a mano:

```
n <- nrow(Prestige)
q <- 3
q0 <- 1
rss0 <- with(Prestige, sum((prestige - mean(prestige))^2))
rss <- sum(residuals(modelo)^2)
inc.mse <- (rss0 - rss)/(q - q0) # Incremento en varabilidad explicada
msr <- rss/(n - q) # Variabilidad residual
inc.mse/msr

## [1] 195.5505
```

Desde el punto de vista de comparación de modelos, el modelo reducido bajo la hipótesis nula es:

```
modelo0 <- lm(prestige ~ 1, data = Prestige)
```

y podemos realizar el contraste mediante la función `anova()`

```
anova(modelo0, modelo)

## Analysis of Variance Table
##
## Model 1: prestige ~ 1
## Model 2: prestige ~ income + education
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     101 29895.4
## 2      99 6038.9  2    23857 195.55 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Para aproximar la distribución de este estadístico bajo H_0 podríamos adaptar el bootstrap semiparamétrico³ descrito en la Sección 3.7.2:

```
library(boot)

pres.dat <- Prestige
# pres.dat$fit0 <- mean(Prestige$prestige)
# pres.dat$fit0 <- predict(modelo0)
pres.dat$res0 <- with(Prestige, prestige - mean(prestige))
# pres.dat$res0 <- residuals(modelo0)

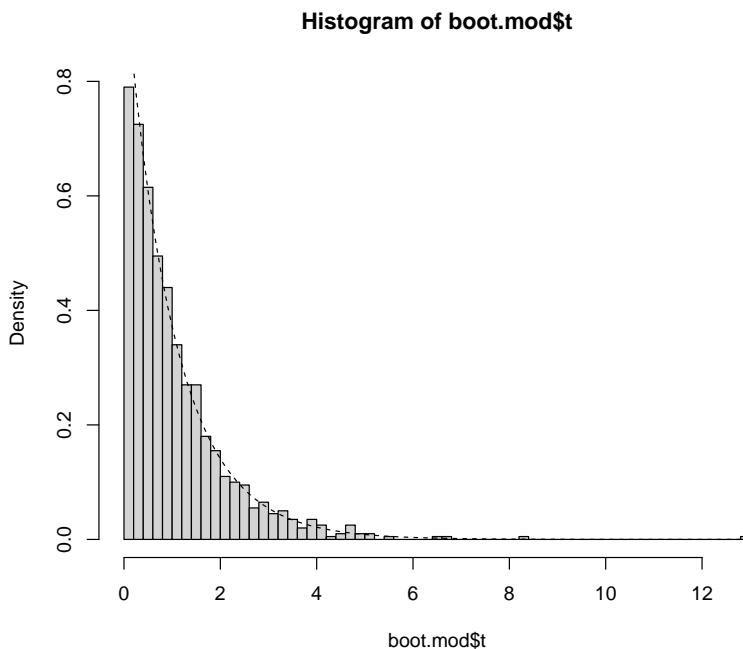
mod.stat <- function(data, i) {
  data$prestige <- mean(data$prestige) + data$res0[i]
  fit <- lm(prestige ~ income + education, data = data)
  summary(fit)$fstatistic[1]
}

set.seed(1)
boot.mod <- boot(pres.dat, mod.stat, R = 1000)
boot.mod

##
```

³En este caso también podríamos emplear un contraste de permutaciones.

```
##  
##  
## Call:  
## boot(data = pres.dat, statistic = mod.stat, R = 1000)  
##  
##  
## Bootstrap Statistics :  
##      original     bias    std. error  
## t1* 195.5505 -194.4866    1.096335  
hist(boot.mod$t, breaks = "FD", freq = FALSE)  
curve(pf(x, df, dfr, lower.tail = FALSE), lty = 2, add = TRUE)
```

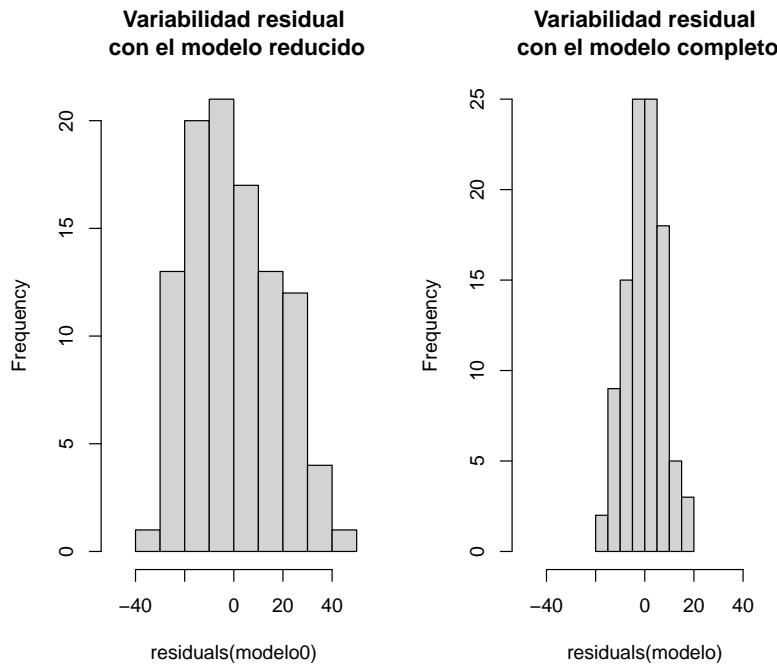


```
# pval <- mean(boot.mod$t >= boot.mod$t0)  
pval <- mean(boot.mod$t >= stat)  
pval
```

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

Procediendo de esta forma sin embargo estaríamos sobreestimando la variabilidad del error cuando la hipótesis nula es falsa (la variabilidad no explicada por la tendencia es asumida por el error), lo que disminuirá la potencia del contraste. Para mejorar la potencia, siguiendo la idea propuesta por González-Manteiga y Cao (1993), se pueden remuestrear los residuos del modelo completo. De esta forma reproduciríamos la variabilidad del error de forma consistente tanto bajo la hipótesis alternativa como bajo la nula.

```
old.par <- par(mfrow=c(1,2))  
hist(residuals(modelo0), xlim = c(-50, 50),  
     main = 'Variabilidad residual\n con el modelo reducido')  
hist(residuals(modelo), xlim = c(-50, 50),  
     main = 'Variabilidad residual\n con el modelo completo')
```



```
par(old.par)
```

Adicionalmente, como se mostró en la Sección 3.7.2, se puede emplear la modificación propuesta en Davison y Hinkley (1997, Alg. 6.3, p. 271) y remuestrear los residuos reescalados y centrados.

```

pres.dat <- Prestige
# pres.dat$fit0 <- mean(Prestige$prestige)
# pres.dat$fit0 <- predict(modelo0)
# pres.dat$res <- residuals(modelo)
pres.dat$sres <- residuals(modelo)/sqrt(1 - hatvalues(modelo))
pres.dat$sres <- pres.dat$sres - mean(pres.dat$sres)

mod.stat <- function(data, i) {
  # data$prestige <- mean(data$prestige) + data$res[i]
  data$prestige <- mean(data$prestige) + data$sres[i]
  fit <- lm(prestige ~ income + education, data = data)
  summary(fit)$fstatistic[1]
}

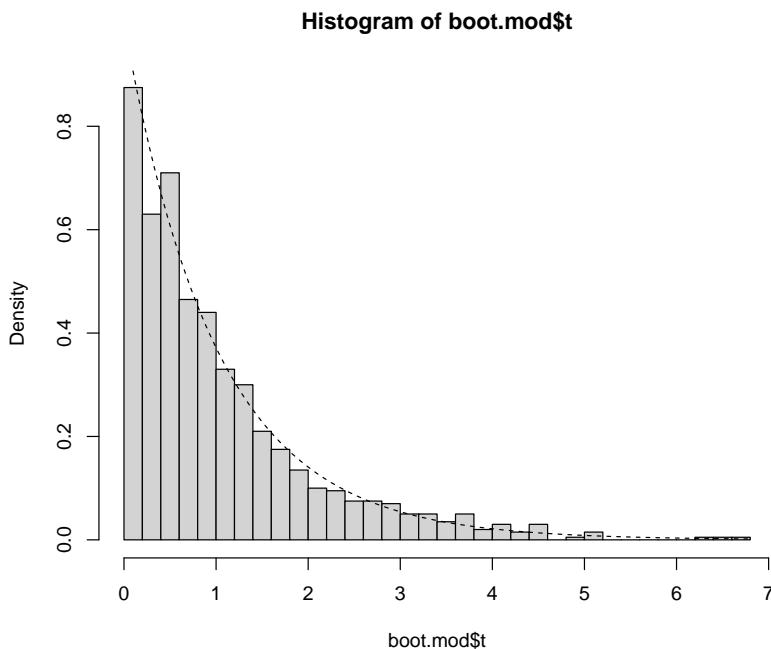
set.seed(1)
boot.mod <- boot(pres.dat, mod.stat, R = 1000)
boot.mod

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = pres.dat, statistic = mod.stat, R = 1000)
##
##
## Bootstrap Statistics :
##       original    bias   std. error
## t1*  0.01164396 1.029746   1.029715

```

En la aproximación del p -valor hay que tener en cuenta que al modificar los residuos `boot.mod$t0` no va a coincidir con el valor observado del estadístico, almacenado en `stat` (por tanto habría que ignorar `original` y `bias` en `Bootstrap Statistics`; la función `Boot()` del paquete `car` corrige este problema).

```
hist(boot.mod$t, breaks = "FD", freq = FALSE)
curve(pf(x, df, dfr, lower.tail = FALSE), lty = 2, add = TRUE)
```



```
pval <- mean(boot.mod$t >= stat)
pval
```

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

En el caso de modelos no lineales (o otros tipos de modelos lineales) puede ser complicado aproximar los grados de libertad para el cálculo del estadístico F , pero si empleamos bootstrap, vamos a obtener los mismos resultados considerando como estadístico:

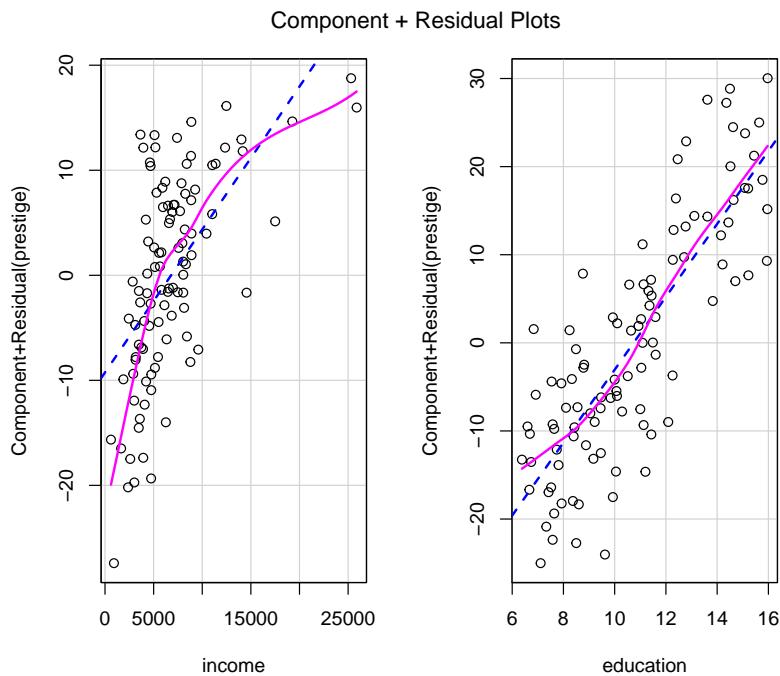
$$\tilde{F} = \frac{RSS_0 - RSS}{RSS},$$

que se puede interpretar también como una medida del incremento en la variabilidad residual al considerar el modelo reducido (ya que únicamente difieren en una constante). En este caso también se suelen emplear los residuos sin reescalar, ya que también puede ser difícil encontrar la transformación adecuada.

5.4.2 Ejercicio

Al estudiar el efecto de las variables explicativas en el modelo anterior, podríamos pensar que no es adecuado asumir un efecto lineal de alguna de las variables explicativas. Por ejemplo, si generamos los gráficos parciales de residuos obtendríamos:

```
# library(car)
crPlots(modelo)
```



En este caso podría ser razonable considerar un efecto cuadrático de la variable `income`⁴

```
modelo <- lm(prestige ~ income + I(income^2) + education, data = Prestige)
summary(modelo)
```

```
##
## Call:
## lm(formula = prestige ~ income + I(income^2) + education, data = Prestige)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -15.732  -4.900  -0.057   4.598  18.459 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.135e+01  3.272e+00 -3.470 0.000775 ***
## income       3.294e-03  5.669e-04  5.810 7.79e-08 ***
## I(income^2) -7.967e-08  2.169e-08 -3.673 0.000390 ***
## education    3.809e+00  3.407e-01 11.179 < 2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 7.36 on 98 degrees of freedom
## Multiple R-squared:  0.8224, Adjusted R-squared:  0.817 
## F-statistic: 151.3 on 3 and 98 DF,  p-value: < 2.2e-16
```

Para comparar el ajuste de este modelo respecto al del anterior, podemos realizar un contraste empleando la función `anova()`:

```
modelo0 <- lm(prestige ~ income + education, data = Prestige)
anova(modelo0, modelo)
```

⁴Para ajustar un modelo polinómico puede ser recomendable, especialmente si el grado del polinomio es alto, emplear la función `poly()` ya que utiliza polinomios ortogonales. En el caso cuadrático, al emplear `y ~ x + I(x^2)` estaremos considerando $1, x, x^2$, mientras que `y ~ poly(x, 2)` considerará polinomios de Legendre de la forma $1, x, \frac{1}{2}(3x^2 - 1)$. En este caso concreto, obtendríamos una parametrización equivalente empleando `modelo <- lm(prestige ~ poly(income, 2) + education, data = Prestige)`.

```
## Analysis of Variance Table
##
## Model 1: prestige ~ income + education
## Model 2: prestige ~ income + I(income^2) + education
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     99 6038.9
## 2     98 5308.0  1      730.8 13.492 0.0003904 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Contrastar si el efecto de `income` es lineal mediante bootstrap residual, empleando como estadístico el incremento en la variabilidad residual con el modelo reducido y remuestreando los residuos del modelo completo (sin reescalar). Aproximar el nivel crítico del contraste y el valor que tendría que superar el estadístico para rechazar H_0 con un nivel de significación $\alpha = 0.05$.

```
library(boot)

# set.seed(DNI)
# ...
```


Capítulo 6

Bootstrap y estimación no paramétrica de la densidad

En este capítulo se presentarán diversos métodos bootstrap adecuados para realizar inferencia en algunos problemas en el contexto de la estimación no paramétrica de la densidad. Concretamente, se abordará el problema de construcción de intervalos de confianza para la funciones de densidad en un punto dado, así como la selección del parámetro de suavizado para el estimador tipo núcleo de la función de densidad. A continuación se incluye una breve introducción a estos métodos no paramétricos de estimación de curvas.

6.1 Estimación no paramétrica de la función de densidad

Como ya se introdujo en la Sección 3.3, si (X_1, X_2, \dots, X_n) es una muestra aleatoria simple (m.a.s.), de una población con función de distribución F , absolutamente continua, y función de densidad f , el estimador tipo núcleo propuesto por Parzen (1962) y Rosenblatt (1956) viene dado por

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-X_i}{h}\right) = \frac{1}{n} \sum_{i=1}^n K_h(x-X_i),$$

donde $K_h(u) = \frac{1}{h}K\left(\frac{u}{h}\right)$, K es una función núcleo (normalmente una densidad simétrica en torno al cero) y $h > 0$ es una parámetro de suavizado, llamado ventana, que regula el tamaño del entorno que se usa para llevar a cabo la estimación. Es habitual exigir que la función núcleo K sea no negativa y su integral sea uno:

$$K(u) \geq 0, \quad \forall u, \quad \int_{-\infty}^{\infty} K(u) du = 1.$$

Además también es frecuente exigir que K sea una función simétrica ($K(-u) = K(u)$).

6.2 Sesgo, varianza y error cuadrático medio

6.2.1 Sesgo

Mediante cálculos sencillos puede obtenerse el sesgo del estimador de Parzen-Rosenblatt:

$$\begin{aligned} \text{Sesgo}(\hat{f}_h(x)) &= E(\hat{f}_h(x)) - f(x) = \int \frac{1}{h} K\left(\frac{x-y}{h}\right) f(y) dy - f(x) \\ &= (K_h * f)(x) - f(x), \end{aligned}$$

siendo $*$ el operador convolución:

$$(f * g)(x) = \int f(x-y) g(y) dy.$$

A partir de la expresión del sesgo puede obtenerse otra asintótica para el mismo:

$$E(\hat{f}_h(x)) - f(x) = \frac{d_K}{2} h^2 f''(x) + O(h^4),$$

con $d_K = \int t^2 K(t) dt$.

6.2.2 Varianza

La varianza puede tratarse análogamente:

$$\begin{aligned} Var(\hat{f}_h(x)) &= \frac{1}{nh^2} Var\left(K\left(\frac{x-X_1}{h}\right)\right) \\ &= \frac{1}{nh^2} \left[\int K\left(\frac{x-y}{h}\right)^2 f(y) dy - \left(\int K\left(\frac{x-y}{h}\right) f(y) dy\right)^2 \right] \\ &= \frac{1}{n} \left[((K_h)^2 * f)(x) - ((K_h * f)(x))^2 \right] \\ &= \frac{1}{nh} \left[(K^2)_h * f \right](x) - \frac{1}{n} [(K_h * f)(x)]^2. \end{aligned}$$

Su expresión asintótica resulta:

$$Var(\hat{f}_h(x)) = \frac{c_K}{nh} f(x) - \frac{1}{n} f(x)^2 + O\left(\frac{h}{n}\right),$$

con $c_K = \int K(t)^2 dt$.

6.2.3 Error cuadrático medio

Como consecuencia el error cuadrático medio del estimador es:

$$\begin{aligned} MSE(\hat{f}_h(x)) &= E(\hat{f}_h(x) - f(x))^2 = Sesgo(\hat{f}_h(x))^2 + Var(\hat{f}_h(x)) \\ &= [(K_h * f)(x) - f(x)]^2 + \frac{1}{nh} \left[(K^2)_h * f \right](x) \\ &\quad - \frac{1}{n} [(K_h * f)(x)]^2. \end{aligned}$$

Además, su expresión asintótica es:

$$MSE(\hat{f}_h(x)) = \frac{d_K^2}{4} h^4 f''(x)^2 + \frac{c_K}{nh} f(x) - \frac{1}{n} f(x)^2 + O(h^6) + O\left(\frac{h}{n}\right).$$

6.2.4 Error cuadrático medio integrado (MISE)

Una medida global (no para un x particular) del error cometido por el estimador es el error cuadrático medio integrado:

$$\begin{aligned} MISE(\hat{f}_h(x)) &= \int E[(\hat{f}_h(x) - f(x))^2] dx = \int MSE(\hat{f}_h(x)) dx = \\ &= \int [(K_h * f)(x) - f(x)]^2 dx + \frac{c_K}{nh} - \frac{1}{n} \int [(K_h * f)(x)]^2 dx. \end{aligned}$$

Una expresión asintótica para el mismo es la siguiente:

$$\begin{aligned} MISE(\hat{f}_h(x)) &= \frac{d_K^2}{4} h^4 \int f''(x)^2 dx + \frac{c_K}{nh} - \frac{1}{n} \int f(x)^2 dx \\ &\quad + O(h^6) + O\left(\frac{h}{n}\right). \end{aligned}$$

En ella se puede ver el efecto negativo de tomar ventanas (h) demasiado grandes o demasiado pequeñas.

6.3 Aproximación Bootstrap de la distribución del estimador de Parzen-Rosenblatt

Antes de proceder a abordar el bootstrap en este contexto conviene presentar la distribución asintótica del estimador y otras aproximaciones posibles. Pueden encontrarse más detalles sobre estos resultados en Cao (1990).

6.3.1 Distribución asintótica del estimador de Parzen-Rosenblatt

Las condiciones mínimas necesarias para que el sesgo y la varianza del estimador tiendan a cero cuando el tamaño muestral tiende a infinito son $h \rightarrow 0$, $nh \rightarrow \infty$. En tales circunstancias se tiene

$$\sqrt{nh} (\hat{f}_h(x) - f(x)) \xrightarrow{d} \mathcal{N}(B, V).$$

Además, puede probarse que el valor asintóticamente óptimo de h , en el sentido del MSE , es $h = c_0 n^{-1/5}$, con

$$c_0 = \left(\frac{c_K f(x)}{d_K^2 f''(x)^2} \right)^{1/5}.$$

Con esa elección de h los valores de media y varianza de la distribución normal límite son

$$\begin{aligned} B &= \frac{1}{2} c_0^{5/2} d_K f''(x), \\ V &= c_K f(x). \end{aligned}$$

Para utilizar la distribución asintótica anterior en la construcción de intervalos de confianza para $f(x)$ podemos

1. Estimar B y V y utilizarlos en la correspondiente distribución normal (**metodo plug-in**).
2. Diseñar un plan de remuestreo y utilizar el **método bootstrap**.

6.3.2 Aproximación plug-in

Pasa por estimar B y V mediante

$$\begin{aligned} \hat{B} &= \frac{1}{2} \hat{c}_0^{5/2} d_K \hat{f}_g''(x), \\ \hat{V} &= c_K \hat{f}_h(x), \end{aligned}$$

siendo g una ventana adecuada para estimar la derivada segunda de la función de densidad. Utilizando la desigualdad de Berry-Esséen se obtiene el siguiente orden de convergencia:

$$\sup_{z \in R} \left| P \left[\sqrt{nh} (\hat{f}_h(x) - f(x)) \leq z \right] - \Phi \left(\frac{z - \hat{B}}{\hat{V}^{1/2}} \right) \right| = O_P(n^{-1/5}),$$

que empeora la tasa teórica de la aproximación normal basada en la media y varianza exactas ($B_n = E[\sqrt{nh}(\hat{f}_h(x) - f(x))]$ y $V_n = \text{Var}[\sqrt{nh}(\hat{f}_h(x) - f(x))]$):

$$\sup_{z \in R} \left| P \left[\sqrt{nh} (\hat{f}_h(x) - f(x)) \leq z \right] - \Phi \left(\frac{z - B_n}{V_n^{1/2}} \right) \right| = O(n^{-2/5}),$$

aunque no la de la normal asintótica, $\mathcal{N}(B, V)$, cuya tasa es igualmente de orden $O_P(n^{-1/5})$.

6.3.3 Aproximación bootstrap

Se procede según el siguiente plan de remuestreo.

1. A partir de la muestra (X_1, X_2, \dots, X_n) y utilizando una **ventana piloto** g , se calcula el estimador de Parzen-Rosenblatt \hat{f}_g .
2. Se arrojan remuestras bootstrap $(X_1^*, X_2^*, \dots, X_n^*)$ a partir de la densidad \hat{f}_g .
3. Se construye el análogo bootstrap del estimador de Parzen-Rosenblatt

$$\hat{f}_h^*(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i^*}{h}\right).$$

4. Se aproxima la distribución en el muestreo de $\sqrt{nh}(\hat{f}_h(x) - f(x))$ por la distribución en el remuestreo de $\sqrt{nh}(\hat{f}_h^*(x) - \hat{f}_g(x))$.

Si nuestro interés estuviese en el sesgo o la varianza de $\hat{f}_h(x)$ entonces utilizariamos, en el paso 4 del algoritmo anterior, los análogos bootstrap del sesgo o la varianza: $E^*(\hat{f}_h^*(x) - \hat{f}_g(x))$ o $Var^*(\hat{f}_h^*(x))$.

En el algoritmo anterior, la ventana g ha de ser asintóticamente mayor que h . De hecho, una elección razonable para g es aquella que minimiza $E\left[\left(\hat{f}_g''(x) - f''(x)\right)^2\right]$. Asintóticamente esa ventana viene dada por

$$g \simeq \left(\frac{5f(x) \int K''(t)^2 dt}{d_K^2 f^{(4)}(x)^2 n} \right)^{1/9}.$$

El orden de convergencia de para la aproximación bootstrap viene dado por

$$\begin{aligned} & \sup_{z \in R} |P[\sqrt{nh}(\hat{f}_h(x) - f(x)) \leq z] - P^*[\sqrt{nh}(\hat{f}_h^*(x) - \hat{f}_g(x)) \leq z]| \\ &= O_P(n^{-2/9}), \end{aligned}$$

que mejora los ofrecidos por la aproximación normal teórica y el método plug-in.

6.4 El Bootstrap en la selección del parámetro de suavizado.

6.4.1 Expresión asintótica de la ventana óptima

El *MISE* tiene una expresión asintótica que puede usarse como criterio para obtener un valor óptimo del parámetro de suavizado:

$$MISE(h) = AMISE(h) + O(h^6) + O\left(\frac{h}{n}\right),$$

con

$$AMISE(h) = \frac{d_K^2}{4} h^4 \int f''(x)^2 dx + \frac{c_K}{nh} - \frac{1}{n} \int f(x)^2 dx.$$

El parámetro de suavizado que minimiza el *AMISE* es

$$h_{AMISE} = \left(\frac{c_K}{nd_K^2 \int f''(x)^2 dx} \right)^{1/5}.$$

Existen multitud de métodos encaminados a dar respuesta al problema de selección del parámetro de suavizado. Entre ellos destacamos los métodos plug-in, los de validación cruzada (suavizada o no) y, desde luego, los métodos bootstrap (ver, por ejemplo, Marron (1992)).

6.4.2 Análogo bootstrap del *MISE*

La idea básica (Cao (1993)) consiste en diseñar un plan de remuestreo, del tipo bootstrap suavizado, para estimar el *MISE*:

1. A partir de la muestra (X_1, X_2, \dots, X_n) y utilizando una ventana piloto g , se calcula el estimador de Parzen-Rosenblatt \hat{f}_g .
2. Se arrojan remuestras bootstrap $(X_1^*, X_2^*, \dots, X_n^*)$ de la densidad \hat{f}_g .
3. Para cada $h > 0$, se obtiene el análogo bootstrap del estimador de Parzen-Rosenblatt

$$\hat{f}_h^*(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i^*}{h}\right).$$

4. Se construye la versión bootstrap del *MISE*:

$$MISE^*(h) = \int E^*\left[\left(\hat{f}_h^*(x) - \hat{f}_g(x)\right)^2\right] dx.$$

5. Se minimiza $MISE^*(h)$ en $h > 0$ y se obtiene el selector bootstrap:

$$h_{MISE}^* = \arg \min_{h>0} MISE^*(h)$$

6.4.3 Expresión cerrada para $MISE^*$

A diferencia de lo que es habitual, en este contexto es posible obtener una expresión cerrada para el análogo bootstrap del *MISE*:

$$\begin{aligned} MISE^*(h) &= \int \left[(K_h * \hat{f}_g)(x) - \hat{f}_g(x) \right]^2 dx \\ &\quad + \frac{c_K}{nh} - \frac{1}{n} \int \left[(K_h * \hat{f}_g)(x) \right]^2 dx \\ &= \frac{c_K}{nh} - \frac{1}{n^3} \sum_{i,j=1}^n \left[(K_h * K_g) * (K_h * K_g) \right] (X_i - X_j) \\ &\quad + \frac{1}{n^2} \sum_{i,j=1}^n \left[(K_h * K_g - K_g) * (K_h * K_g - K_g) \right] (X_i - X_j). \end{aligned}$$

6.4.4 Elección de la ventana piloto

De nuevo ocurre que el problema de elección óptima de la ventana piloto, g , viene ligado al de estimación óptima de la curvatura de la función de densidad. Así, una buena elección de g es la que minimiza

$$E \left[\left(\int \hat{f}_g''(x)^2 dx - \int f''(x)^2 dx \right)^2 \right].$$

El valor asintótico de dicha ventana g es

$$g_0 = \left(\frac{\int K''(t)^2 dt}{nd_K \int f^{(3)}(x)^2 dx} \right)^{1/7}.$$

6.4.5 Resultados teóricos

Utilizando cualquier ventana piloto determinística que cumpla $\frac{g-g_0}{g_0} = O(n^{-1/14})$, se tiene

$$\begin{aligned} \frac{h_{MISE}^* - h_{MISE}}{h_{MISE}} &= O_P(n^{-5/14}), \\ \frac{MISE(h_{MISE}^*) - MISE(h_{MISE})}{MISE(h_{MISE})} &= O_P(n^{-5/7}). \end{aligned}$$

Mediante técnicas más sofisticadas que permiten que g dependa de h pueden obtenerse tasas ligeramente mejores:

$$\frac{h_{MISE}^* - h_{MISE}}{h_{MISE}} = O_P(n^{-1/2}).$$

6.4.6 Caso particular de núcleo gaussiano

Cuando el núcleo K es la función de densidad de una $\mathcal{N}(0, 1)$:

- K_h es la densidad de una $\mathcal{N}(0, h^2)$
- K_g es la densidad de una $\mathcal{N}(0, g^2)$
- $K_h * K_g$ es la densidad de una $\mathcal{N}(0, h^2 + g^2)$
- $(K_h * K_g) * (K_h * K_g)$ es la densidad de una $\mathcal{N}(0, 2h^2 + 2g^2)$
- $(K_h * K_g) * K_g$ es la densidad de una $\mathcal{N}(0, h^2 + 2g^2)$
- $K_g * K_g$ es la densidad de una $\mathcal{N}(0, 2g^2)$

con lo cual

$$\begin{aligned} MISE^*(h) &= \frac{c_K}{nh} - \frac{1}{n^3} \sum_{i,j=1}^n K_{\sqrt{2h^2+2g^2}}(X_i - X_j) \\ &\quad + \frac{1}{n^2} \sum_{i,j=1}^n \left[K_{\sqrt{2h^2+2g^2}}(X_i - X_j) \right. \\ &\quad \left. - 2K_{\sqrt{h^2+2g^2}}(X_i - X_j) + K_{\sqrt{2g^2}}(X_i - X_j) \right]. \end{aligned}$$

6.4.7 Comparación con otros selectores

El método bootstrap presentado es muy semejante al de validación cruzada suavizada (SCV) propuesto por Hall, Marron y Park (1992). En estudios de simulación comparativos (ver Cao, Cuevas y González-Manteiga (1993)) puede verse como este método ofrece resultados muy competitivos con otros métodos de selección del parámetro de suavizado. En general es el que mejor comportamiento ofrece junto con el método plug-in tipo solve-the-equation de Sheather y Jones (1991) y el método SCV.

Otros selectores bootstrap con mucho peor comportamiento son:

- Hall (1990), en el que se remuestrea de la distribución empírica, con lo cual no se imita el sesgo.
- Faraway y Jhun (1990), que eligen g como la ventana de validación cruzada, que resulta ser demasiado pequeña.
- Taylor (1989), que elige $g = h$, con lo cual $MISE^*(h) \rightarrow 0$, cuando $h \rightarrow \infty$, lo cual produce un mínimo global de $MISE^*$ inconsistente con h_{MISE} .

6.5 Estimación no paramétrica de la densidad en R

Como ya se comentó en la Sección 3.3, en R podemos emplear la función `density()` del paquete base para obtener una estimación tipo núcleo de la densidad. Los principales parámetros (con los valores por defecto) son los siguientes:

```
density(x, bw = "nrd0", adjust = 1, kernel = "gaussian", n = 512, from, to)
```

- **bw:** ventana, puede ser un valor numérico o una cadena de texto que la determine (en ese caso llamará internamente a la función `bw.xxx()` donde `xxx` se corresponde con la cadena de texto). Las opciones son:

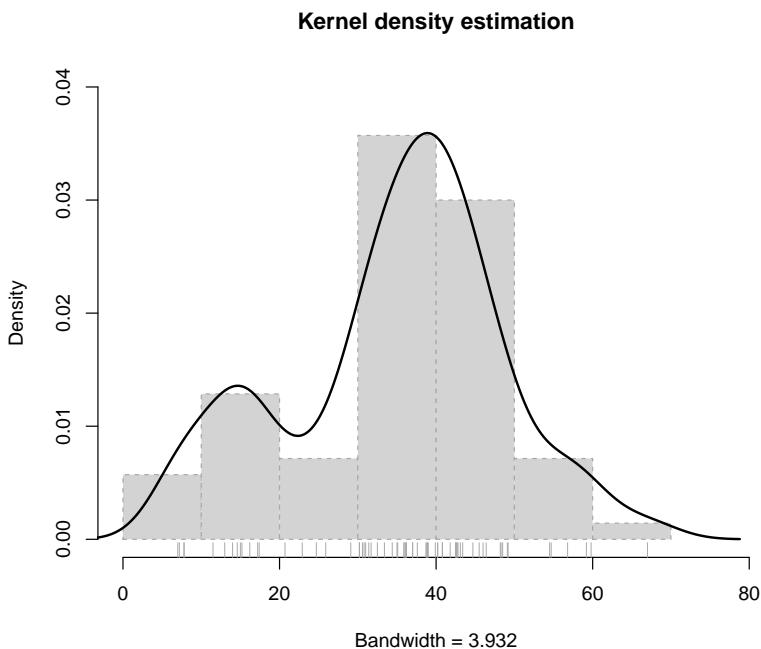
- `"nrd0"`, `"nrd"`: Reglas del pulgar de Silverman (1986, page 48, eqn (3.31)) y Scott (1992), respectivamente. Como es de esperar que la densidad objetivo no sea tan suave como la normal, estos criterios tenderán a seleccionar ventanas que producen un sobresuavizado de las observaciones.

- "ucv", "bcv": Métodos de validación cruzada insesgada y sesgada, respectivamente.
- "sj", "sj-ste", "sj-dpi": Métodos de Sheather y Jones (1991), “solve-the-equation” y “direct plug-in”, respectivamente.
- **adjust**: parámetro para reescalado de la ventana, las estimaciones se calculan con la ventana $\text{adjust} * \text{bw}$.
- **kernel**: cadena de texto que determina la función núcleo, las opciones son: "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine" y "optcosine".
- **n, from, to**: permiten establecer la rejilla en la que se obtendrán las estimaciones (si $n > 512$ se emplea `fft()` por lo que se recomienda establecer **n** a un múltiplo de 2; por defecto **from** y **to** se establecen como **cut** = 3 veces la ventana desde los extremos de las observaciones).

Utilizaremos como punto de partida el código empleado en la Sección 3.3. Considerando el conjunto de datos `precip` (que contiene el promedio de precipitación, en pulgadas de lluvia, de 70 ciudades de Estados Unidos).

```
x <- precip
h <- bw.SJ(x)
npden <- density(x, bw = h)
# npden <- density(x, bw = "SJ")

# plot(npden)
hist(x, freq = FALSE, main = "Kernel density estimation",
      xlab = paste("Bandwidth =", formatC(h)), lty = 2,
      border = "darkgray", xlim = c(0, 80), ylim = c(0, 0.04))
lines(npden, lwd = 2)
rug(x, col = "darkgray")
```



Alternativamente podríamos emplear implementaciones en otros paquetes de R. Uno de los más empleados es `ks` (Duong, 2019), que admite estimación incondicional y condicional multidimensional. También se podrían emplear los paquetes `KernSmooth` (Wand y Ripley, 2019), `sm` (Bowman y Azzalini, 2019), `np` (Tristen y Jeffrey, 2019), `kedd` (Guidoum, 2019), `features` (Duong y Matt, 2019) y `npsp` (Fernández-Casal, 2019), entre otros.

6.6 Ejemplos

En esta sección nos centraremos en el bootstrap en la estimación tipo núcleo de la densidad para la aproximación de la precisión y el sesgo, y también para el cálculo de intervalos de confianza.

6.6.1 Bootstrap y estimación del sesgo

La idea sería aproximar la distribución del error de estimación $\hat{f}_h(x) - f(x)$ por la distribución bootstrap de $\hat{f}_h^*(x) - \hat{f}_g(x)$ (bootstrap percentil básico).

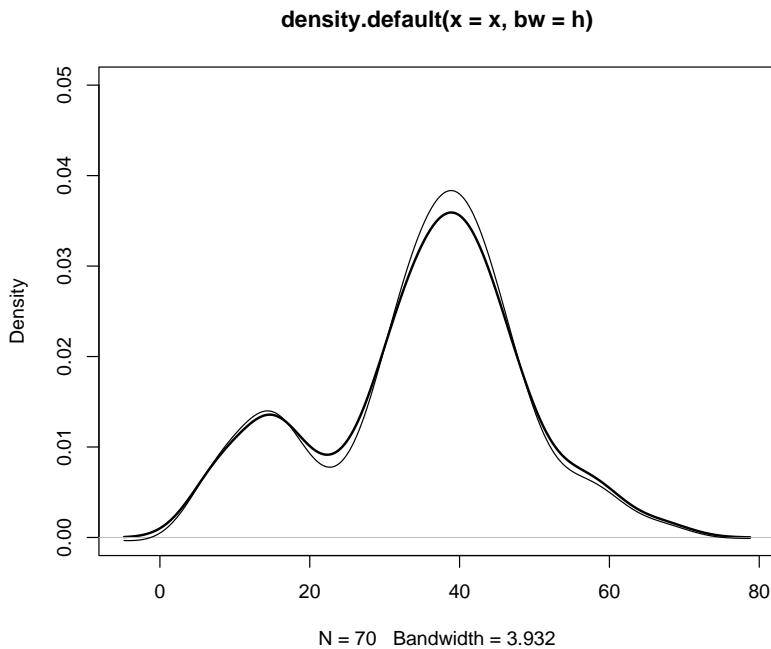
Como se comentó en la Sección 6.3 la ventana g debería ser asintóticamente mayor que h (de orden $n^{-1/5}$) y la recomendación sería emplear la ventana óptima para la estimación de $f''(x)$, de orden $n^{-1/9}$. Sin embargo, en la práctica es habitual emplear $g = h$ para evitar la selección de esta ventana (lo que además facilita emplear herramientas como el paquete `boot`). Otra alternativa podría ser asumir que $g \simeq n^{1/5}h/n^{1/9}$ como se hace a continuación.

```
# Remuestreo
set.seed(1)
n <- length(x)
g <- h * n^(4/45) # h*n^(-1/9)/n^(-1/5)
# g <- h
range_x <- range(npden$x) # Para fijar las posiciones de estimación
B <- 1000
stat_den_boot <- matrix(nrow = length(npden$x), ncol = B)
npdeng <- density(x, bw = g, from = range_x[1], to = range_x[2])

for (k in 1:B) {
  # x_boot <- sample(x, n, replace = TRUE) + rnorm(n, 0, g)
  x_boot <- rnorm(n, sample(x, n, replace = TRUE), g)
  den_boot <- density(x_boot, bw = h, from = range_x[1], to = range_x[2])$y
  # Si se quiere tener en cuenta la variabilidad debida a la selección de
  # la ventana habría que emplear el mismo criterio en la función `density`.
  stat_den_boot[, k] <- den_boot - npdeng$y
}

# Calculo del sesgo y error estándar
bias <- apply(stat_den_boot, 1, mean)
std_err <- apply(stat_den_boot, 1, sd)

# Representar estimación y corrección de sesgo bootstrap
plot(npden, type="l", ylim = c(0, 0.05), lwd = 2)
# lines(npden$x, pmax(npden$y - bias, 0))
lines(npden$x, npden$y - bias)
```

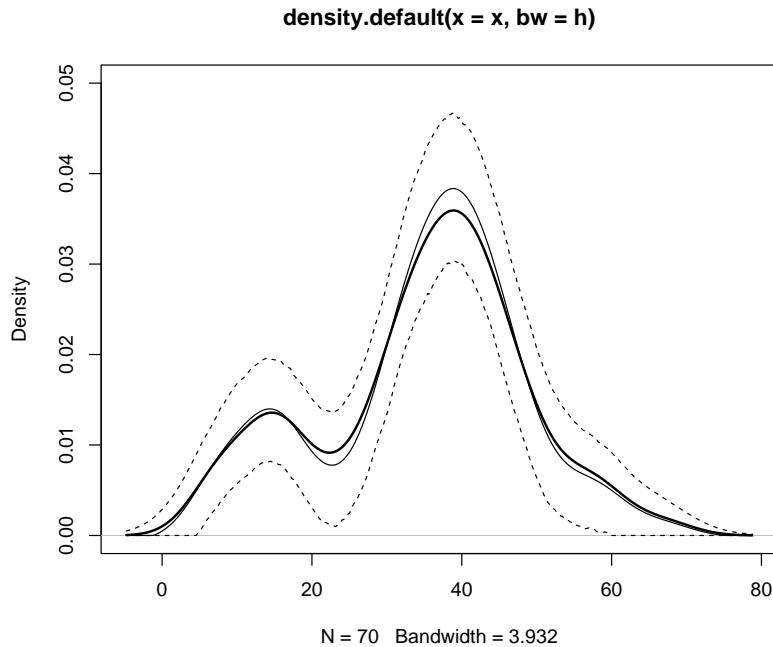


6.6.2 Estimación por intervalos de confianza

Empleando la aproximación descrita en la Sección 4.2 podemos calcular de estimaciones por intervalo de confianza (puntuales) por el método percentil (básico).

```
alfa <- 0.05
pto_crit <- apply(stat_den_boot, 1, quantile, probs = c(alfa/2, 1 - alfa/2))
# ic_inf_boot <- npden$y - pto_crit[2, ]
ic_inf_boot <- pmax(npden$y - pto_crit[2, ], 0)
ic_sup_boot <- npden$y - pto_crit[1, ]

plot(npden, type="l", ylim = c(0, 0.05), lwd = 2)
lines(npden$x, pmax(npden$y - bias, 0)) # Ojo: no es una densidad
lines(npden$x, ic_inf_boot, lty = 2)
lines(npden$x, ic_sup_boot, lty = 2)
```



6.6.3 Implementación con el paquete boot

Como también se comentó en la Sección 3.3, la recomendación es implementar el bootstrap suavizado como un bootstrap paramétrico:

```
library(boot)

# Los objetos necesarios para el cálculo del estadístico
# hay que pasarlos a través del argumento `data` de `boot`.
# range_x <- range(npden$x)
data.precip <- list(x = x, h = h, range_x = range_x)

ran.gen.smooth <- function(data, mle) {
  # Función para generar muestras aleatorias mediante
  # bootstrap suavizado con función núcleo gaussiana,
  # mle contendrá la ventana
  n <- length(data$x)
  g <- mle
  xboot <- rnorm(n, sample(data$x, n, replace = TRUE), g)
  out <- list(x = xboot, h = data$h, range_x = data$range_x)
}

statistic <- function(data)
  density(data$x, bw = data$h, from = range_x[1], to = range_x[2])$y

set.seed(1)
res.boot <- boot(data.precip, statistic, R = B, sim = "parametric",
  ran.gen = ran.gen.smooth, mle = g)

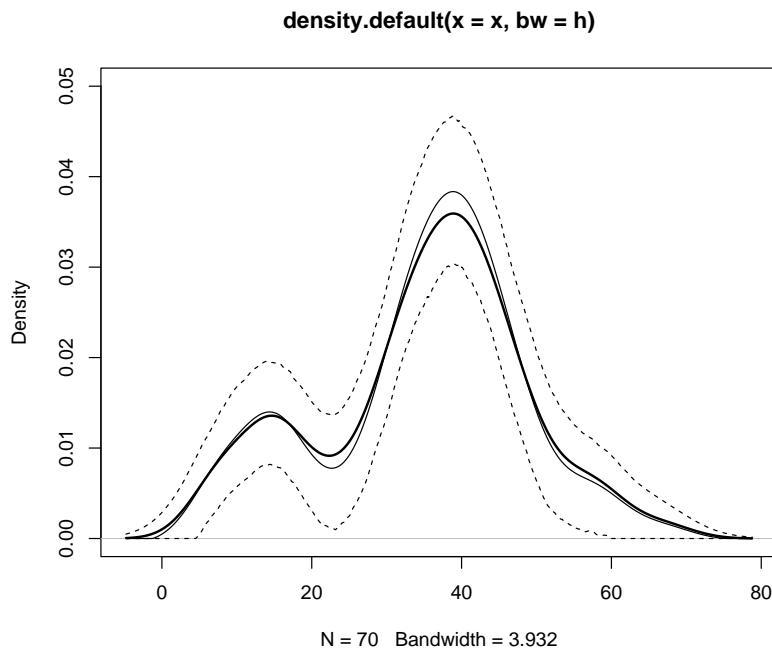
# Calculo del sesgo y error estándar
# Empleamos npdeng$y en lugar de resboot$t0
bias <- apply(res.boot$t, 2, mean, na.rm = TRUE) - npdeng$y
std_err <- apply(res.boot$t, 2, sd, na.rm = TRUE)
```

Además, la función `boot.ci()` solo permite el cálculo del intervalo de confianza para cada valor de x de forma independiente (parámetro `index`). Por lo que podría ser recomendable obtenerlo a partir de las réplicas bootstrap del estimador:

```
# Método percentil básico calculado directamente
# a partir de las réplicas bootstrap del estimador
alfa <- 0.05
pto_crit <- apply(res.boot$t, 2, quantile, probs = c(alfa/2, 1 - alfa/2))
ic_inf_boot <- pmax(npden$y + npdeng$y - pto_crit[2, ], 0)
ic_sup_boot <- npden$y + npdeng$y - pto_crit[1, ]

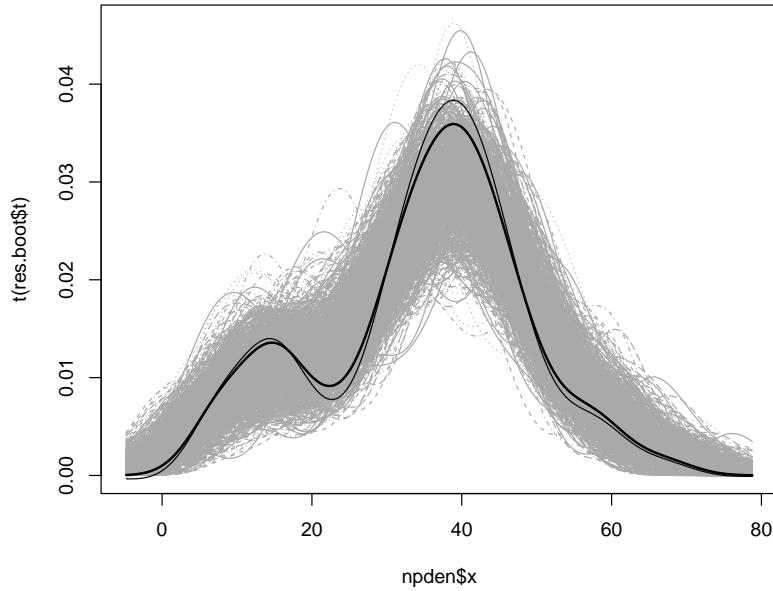
plot(npden, ylim = c(0, 0.05), lwd = 2)
lines(npden$x, pmax(npden$y - bias, 0))

lines(npden$x, ic_inf_boot, lty = 2)
lines(npden$x, ic_sup_boot, lty = 2)
```



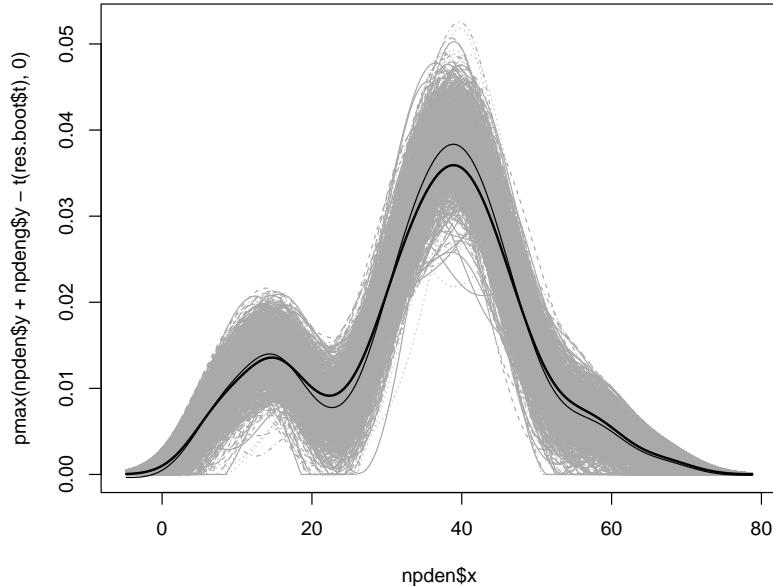
En la práctica, en muchas ocasiones se trabaja directamente con las réplicas bootstrap del estimador. Por ejemplo, es habitual generar envolventes como medida de la precisión de la estimación (que se interpretan de forma similar a una banda de confianza):

```
matplotlib(npden$x, t(res.boot$t), type = "l", col = "darkgray")
lines(npden, lwd = 2)
lines(npden$x, npden$y - bias)
```



Pero la recomendación es emplear bootstrap básico (o percentil-*t*) en lugar de bootstrap percentil (directo) en la presencia de sesgo:

```
# matplot(npden$x, npden$y + npdeng$y - t(res.boot$t), type = "l", col = "darkgray")
matplot(npden$x, pmax(npden$y + npdeng$y - t(res.boot$t), 0), type = "l", col = "darkgray")
lines(npden, lwd = 2)
lines(npden$x, npden$y - bias)
```



Capítulo 7

Bootstrap y regresión no paramétrica

Por simplicidad nos centraremos en el caso bivariante, el caso multivariante sería análogo. Supongamos que $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ es una m.a.s. de una población bidimensional (X, Y) , con $E(|Y|) < \infty$, y que el objetivo es realizar inferencias sobre la distribución condicional $Y|_{X=x}$, principalmente estimar la función de regresión de Y dada X :

$$m(x) = E(Y|_{X=x}).$$

En esta sección se introducirá la estimación no paramétrica de la función de regresión. En primer lugar se considerará el *estimador de Nadaraya-Watson* (en la Sección 7.3 se mostrarán generalizaciones de este estimador desde un punto de vista práctico en R). En la siguiente sección se introducirán distintos métodos de remuestreo, diseñados inicialmente para este estimador, y resultados para ellos.

7.1 Estimador de Nadaraya-Watson

La función de regresión $m(x) = E(Y|_{X=x})$ puede escribirse así:

$$\begin{aligned} m(x) &= \int y f_{2|1}(y|x) dy = \int y \frac{f(x,y)}{f_1(x)} dy = \frac{\int y f(x,y) dy}{f_1(x)} \\ &= \frac{\int y f_{1|2}(x|y) f_2(y) dy}{f_1(x)} = \frac{\Psi(x)}{f_1(x)}, \end{aligned}$$

siendo $f_1(x)$ la función de densidad marginal de X y

$$\Psi(x) = \int y f_{1|2}(x|y) f_2(y) dy = E(Y f_{1|2}(x|Y)).$$

Las funciones $\Psi(x)$ y $f_1(x)$ pueden estimarse mediante el método núcleo:

$$\begin{aligned} \hat{f}_{1,h}(x) &= \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-X_i}{h}\right), \\ \hat{\Psi}_h(x) &= \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-X_i}{h}\right) Y_i, \end{aligned}$$

resultando así el estimador tipo núcleo de Nadaraya-Watson (ver Nadaraya (1964) y Watson (1964)):

$$\hat{m}_h(x) = \frac{\hat{\Psi}_h(x)}{\hat{f}_{1,h}(x)} = \frac{\frac{1}{n} \sum_{i=1}^n K_h(x-X_i) Y_i}{\frac{1}{n} \sum_{i=1}^n K_h(x-X_i)},$$

donde $K_h(x - X_i) = \frac{1}{h} K\left(\frac{x - X_i}{h}\right)$.

Para este estimador se pueden probar propiedades semejantes a las mencionadas para el estimador de Parzen-Rosenblatt de la función de densidad.

En esta sección se presentarán métodos de remuestreo bootstrap adecuados para el contexto de la función de regresión. El objetivo es aproximar la distribución en el muestreo del estimador de Nadaraya-Watson. Los resultados reflejan el comportamiento de los métodos de remuestreo bootstrap, tanto en un aspecto condicional a la muestra de la variable explicativa como incondicionalmente.

7.1.1 Distribución asintótica del estimador de Nadaraya-Watson

Antes de proceder a abordar el bootstrap en este contexto conviene presentar la distribución asintótica del estimador de Nadaraya-Watson, dado por

$$\hat{m}_h(x) = \frac{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i) Y_i}{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i)}.$$

De forma semejante al caso de la densidad, puede comprobarse que las condiciones mínimas necesarias para la consistencia del estimador, en términos del parámetro de suavizado, son $h \rightarrow 0$, $nh \rightarrow \infty$, cuando $n \rightarrow \infty$. En tales circunstancias se tiene

$$\sqrt{nh}(\hat{m}_h(x) - m(x)) \xrightarrow{d} \mathcal{N}(B, V).$$

Además, puede probarse que el valor asintóticamente óptimo de h , en el sentido del *MSE*, es de la forma $h = c_0 n^{-1/5}$. En tal caso, los valores de media y varianza de la distribución normal límite son

$$\begin{aligned} B &= \frac{1}{2} c_0^{5/2} d_K \frac{m''(x) f(x) + 2m'(x) f'(x)}{f(x)}, \\ V &= c_K \frac{\sigma^2(x)}{f(x)}, \end{aligned}$$

siendo $f(x)$ la función de densidad marginal de X y $\sigma^2(x) = \text{Var}(Y|_{X=x})$ la varianza condicional de Y dado $X = x$.

Al igual que en el caso de la densidad, para utilizar la distribución asintótica anterior en la construcción de intervalos de confianza para $m(x)$ podemos

1. Estimar B y V y utilizarlos en la correspondiente distribución normal (**metodo plug-in**).
2. Diseñar un plan de remuestreo y utilizar el **método bootstrap**.

7.1.2 Órdenes de convergencia de la distribución del estimador de Nadaraya-Watson a su distribución asintótica

Los órdenes de convergencia de la aproximación de la distribución (condicional o incondicional) del estadístico a la distribución normal límite vienen dados por:

$$\begin{aligned} \sup_{z \in R} \left| P^{Y|x} [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - \Phi\left(\frac{z - B}{V^{1/2}}\right) \right| &= O_P(n^{-1/5}), \\ \sup_{z \in R} \left| P [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - \Phi\left(\frac{z - B}{V^{1/2}}\right) \right| &= O(n^{-2/5}), \end{aligned}$$

donde $P^{Y|x}(A)$ denota $P(A|_{X_1, X_2, \dots, X_n})$.

7.1.3 Aproximación plug-in

Consiste en estimar B y V mediante estimadores apropiados de $f(x)$, $f'(x)$, $m(x)$, $m'(x)$, $m''(x)$ y $\sigma^2(x)$. Usando, para cada una de estas seis curvas, selectores de los parámetros de suavizado encaminados a aproximar las ventanas óptimas para cada una de ellas (proceso bastante laborioso), pueden obtenerse estimadores del sesgo, \hat{B} , y la varianza, \hat{V} , que cumplen $\hat{B} - B = O_P(n^{-2/9})$ y $\hat{V} - V = O_P(n^{-2/5})$. Como consecuencia se tienen los siguientes órdenes de convergencia (condicional e incondicional) para la aproximación plug-in:

$$\begin{aligned}\sup_{z \in R} \left| P^{Y|x} [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - \Phi\left(\frac{z - \hat{B}}{\hat{V}^{1/2}}\right) \right| &= O_P(n^{-1/5}), \\ \sup_{z \in R} \left| P[\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - \Phi\left(\frac{z - \hat{B}}{\hat{V}^{1/2}}\right) \right| &= O_P(n^{-2/9}).\end{aligned}$$

que iguala y empeora, respectivamente, la tasa teórica de la aproximación normal límite (ver Cao (1991)).

7.2 Métodos de remuestreo en regresión no paramétrica

En este caso se podría emplear también bootstrap uniforme (Sección 3.7.1) y el bootstrap residual, de forma totalmente análoga a como se mostró en la Sección 3.7.2. Sin embargo, en el caso heterocedástico es habitual emplear *Wild bootstrap* y en el caso de diseño aleatorio podría ser recomendable emplear *bootstrap suavizado en la variable explicativa*.

7.2.1 Wild bootstrap

Este método de remuestreo bootstrap, propuesto por Wu (1986) y estudiado por Härdle y Marron (1991), procede del siguiente modo:

1. A partir del estimador de Nadaraya-Watson de $m(x)$ y tomando el parámetro ventana de partida, h , se construyen los residuos $r_i = Y_i - \hat{m}_h(X_i)$, $i = 1, 2, \dots, n$.
2. Para cada índice $i = 1, 2, \dots, n$, se arroja, condicionalmente a la muestra observada, $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$, un error bootstrap $\hat{\varepsilon}_i^*$ de una distribución de probabilidad que cumpla, $E^*(\hat{\varepsilon}_i^*) = 0$, $E^*(\hat{\varepsilon}_i^{*2}) = r_i^2$ y $E^*(\hat{\varepsilon}_i^{*3}) = r_i^3$. Aunque la condición del momento de orden 3 no es estrictamente necesaria, es útil para las demostraciones de validez del método.
3. Usando una ventana piloto g , asintóticamente mayor que h (i.e. $g/h \rightarrow \infty$), se arrojan análogos bootstrap de las observaciones de la variable respuesta: $Y_i^* = \hat{m}_g(X_i) + \hat{\varepsilon}_i^*$, $i = 1, 2, \dots, n$.
4. A partir de la remuestra bootstrap $\{(X_1, Y_1^*), (X_2, Y_2^*), \dots, (X_n, Y_n^*)\}$ se construye el análogo bootstrap del estimador de Nadaraya-Watson:

$$\hat{m}_h^*(x) = \frac{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i) Y_i^*}{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i)}.$$

5. Se approxima la distribución en el muestreo de $\sqrt{nh}(\hat{m}_h(x) - m(x))$ por la distribución en el remuestreo de $\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x))$.

El paso 2 suele llevarse a cabo encontrando una variable aleatoria, V^* , que cumpla $E^*(V^*) = 0$, $E^*(V^{*2}) = 1$ y $E^*(V^{*3}) = 1$, arrojando una muestra de tamaño n de la misma, $(V_1^*, V_2^*, \dots, V_n^*)$, y luego definiendo $\hat{\varepsilon}_i^* = r_i V_i^*$ para $i = 1, 2, \dots, n$.

Una de las elecciones más habituales para la distribución de V^* es la distribución discreta con masa de probabilidad en dos puntos ($P^*(V^* = a) = p$ y $P^*(V^* = b) = 1 - p$) que es solución del sistema

de tres ecuaciones dadas por los tres primeros momentos:

$$\begin{aligned} ap + b(1-p) &= 0, \\ a^2p + b^2(1-p) &= 1, \\ a^3p + b^3(1-p) &= 1. \end{aligned}$$

Esto da lugar al llamado bootstrap de la sección aurea (golden section bootstrap), con $a = \frac{1-\sqrt{5}}{2}$, $b = \frac{1+\sqrt{5}}{2}$, $p = \frac{5+\sqrt{5}}{10}$, es decir

$$\begin{aligned} P^* \left(V^* = \frac{1-\sqrt{5}}{2} \right) &= \frac{5+\sqrt{5}}{10} \\ P^* \left(V^* = \frac{1+\sqrt{5}}{2} \right) &= \frac{5-\sqrt{5}}{10} \end{aligned}$$

La elección de la ventana g , que aparece en el paso 3, guarda relación con la estimación de $m''(x)$, pues esa es la cantidad crítica a la hora de estimar B y V . Tomando una ventana piloto de orden óptimo en ese sentido, $g_0 \simeq d_0 n^{-1/9}$, se obtienen las siguientes tasas de convergencia (condicionales e incondicionales) para la aproximación dada por el wild bootstrap:

$$\begin{aligned} \sup_{z \in R} |P^{Y|x} [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - P^* [\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x)) \leq z]| &= O_P(n^{-2/9}), \\ \sup_{z \in R} |P [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - P^* [\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x)) \leq z]| &= O_P(n^{-1/5}). \end{aligned}$$

7.2.2 Bootstrap suavizado en la variable explicativa

La idea es tratar, por un lado, de considerar la variabilidad inherente a la variable explicativa (en el wild bootstrap esa parte de la remuestra se mantiene fija) y, por otro, que la distribución en el remuestreo de $Y^*|_{X^*=X_i}$ no sea degenerada (como sí lo sería en un bootstrap naïve bidimensional).

El plan de remuestreo, propuesto por Cao y González-Manteiga (1993) consta de los siguientes pasos:

1. Dada la muestra $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$, se construye un estimador (empírico en la variable respuesta y suavizado en la explicativa) de la distribución conjunta de (X, Y) :

$$\hat{F}_g(x, y) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{Y_i \leq y\}} \int_{-\infty}^x K_g(t - X_i) dt.$$

2. Se arrojan remuestras bootstrap, $\{(X_1^*, Y_1^*), (X_2^*, Y_2^*), \dots, (X_n^*, Y_n^*)\}$, con distribución $\hat{F}_g(x, y)$.
3. Se construye el análogo bootstrap del estimador de Nadaraya-Watson:

$$\hat{m}_h^*(x) = \frac{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i^*) Y_i^*}{\frac{1}{n} \sum_{i=1}^n K_h(x - X_i^*)}.$$

4. Se utiliza la distribución en el remuestreo de $\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x))$ para aproximar la distribución del estadístico de interés: $\sqrt{nh}(\hat{m}_h(x) - m(x))$.

La ventana piloto, g , óptima vuelve a ser de orden $n^{-1/9}$, es decir asintóticamente mayor que h .

La distribución bidimensional de la que se remuestrea en el paso 2, $\hat{F}_g(x, y)$, puede sustituirse por una distribución suavizada en ambas variables:

$$\tilde{F}_g(x, y) = \frac{1}{n} \sum_{i=1}^n \int_{-\infty}^y K_g(s - Y_i) ds \int_{-\infty}^x K_g(t - X_i) dt.$$

Esto es lo mismo que remuestrear de la densidad bidimensional

$$\hat{f}_g(x, y) = \frac{1}{n} \sum_{i=1}^n K_g(x - X_i) K_g(y - Y_i),$$

que es el estimador tipo núcleo de Parzen-Rosenblatt de la variable bidimensional (X, Y) .

Cálculos sencillos permiten demostrar que si (X^*, Y^*) tiene distribución $\hat{F}_g(x, y)$, entonces,

- X^* tiene densidad marginal bootstrap $\hat{f}_g(x)$.
- La distribución marginal bootstrap de Y^* es la empírica de las Y_i : $\hat{F}_n^Y(y) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{Y_i \leq y\}}$.
- La función de regresión del plan de remuestreo bootstrap coincide con la estimación de Nadaraya-Watson con ventana g , es decir,

$$E^*(Y^*|_{X^*=x}) = \hat{m}_g(x).$$

- De hecho, la distribución condicional $Y^*|_{X^*=x}$ es

$$\hat{F}_g(y|x) = \frac{\frac{1}{n} \sum_{i=1}^n K_g(x - X_i) \mathbf{1}_{\{Y_i \leq y\}}}{\frac{1}{n} \sum_{i=1}^n K_g(x - X_i)},$$

es decir, el estimador tipo núcleo Nadaraya-Watson de la distribución condicional.

Esta última observación da pie a diseñar un método que permita simular valores de (X^*, Y^*) , tal y como se requiere en el paso 2 del plan de remuestreo. Para ello basta con simular X^* a partir del estimador de Parzen-Rosenblatt construido con la muestra de la variable explicativa (es decir, el bootstrap suavizado clásico) y luego simular Y^* a partir de la distribución discreta que da a cada dato Y_i la probabilidad

$$w_i(X^*) = \frac{\frac{1}{n} K_g(X^* - X_i)}{\frac{1}{n} \sum_{j=1}^n K_g(X^* - X_j)}, \quad i = 1, 2, \dots, n.$$

Las tasas de convergencia de la aproximación bootstrap proporcionadas por este método resultan:

$$\begin{aligned} \sup_{z \in R} & \left| P^{Y|x} [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - P^{Y^*|x^*} [\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x)) \leq z] \right| \\ &= O_{P^*}(n^{-2/9}), \text{ en probabilidad } P, \\ \sup_{z \in R} & \left| P [\sqrt{nh}(\hat{m}_h(x) - m(x)) \leq z] - P^* [\sqrt{nh}(\hat{m}_h^*(x) - \hat{m}_g(x)) \leq z] \right| \\ &= O_P(n^{-2/9}). \end{aligned}$$

7.2.3 Resumen comparativo

La siguiente tabla recoge un resumen de las tasas de convergencia obtenidas con cada una de las aproximaciones estudiadas:

Aproximación	condicional	incondicional
Normal teórica	$O_P(n^{-1/5})$	$O(n^{-2/5})$
Plug-in	$O_P(n^{-1/5})$	$O_P(n^{-2/9})$
Wild bootstrap	$O_P(n^{-2/9})$	$O_P(n^{-1/5})$
Bootstrap suavizado en la variable explicativa	$O_{P^*}(n^{-2/9})$ en probabilidad P	$O_P(n^{-2/9})$

Exceptuando las tasas de convergencia de la normal teórica (aproximación inutilizable en la práctica) se observa que el método bootstrap suavizado en la variable explicativa presenta órdenes que igualan o mejoran al resto de los métodos, tanto en un aspecto condicional como incondicionalmente. Así, incondicionalmente los dos remuestreos bootstrap son los que ofrecen una mejor tasa de convergencia ($n^{-2/9}$, frente a $n^{-1/5}$ de la aproximación plug-in). En el sentido incondicional el bootstrap suavizado en la variable explicativa y la aproximación plug-in son los que presentan un mejor orden ($n^{-2/9}$, frente a $n^{-1/5}$ del wild bootstrap).

7.3 Regresión polinómica local en R

El estimador de Nadaraya-Watson de $m(x)$ descrito en la Sección 7.1 es un caso particular de una clase más amplia de estimadores no paramétricos, denominados estimadores polinómicos locales.

En el caso univariante, para cada x_0 se ajusta un polinomio:

$$\beta_0 + \beta_1(x - x_0) + \cdots + \beta_p(x - x_0)^p$$

por mínimos cuadrados ponderados, con pesos $w_i = \frac{1}{h} K(\frac{x-x_0}{h})$.

- La estimación en x_0 es $\hat{m}_h(x_0) = \hat{\beta}_0$.
- Adicionalmente¹: $\widehat{m}_h^{(r)}(x_0) = r! \hat{\beta}_r$.

Por tanto, la estimación polinómica local de grado p , $\hat{m}_h(x) = \hat{\beta}_0$, se obtiene al minimizar:

$$\min_{\beta_0, \beta_1, \dots, \beta_p} \sum_{i=1}^n \left\{ Y_i - \beta_0 - \beta_1(x - X_i) - \dots - \beta_p(x - X_i)^p \right\}^2 K_h(x - X_i).$$

Explícitamente:

$$\hat{m}_h(x) = \mathbf{e}_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x \mathbf{Y} \equiv s_x^T \mathbf{Y},$$

donde $\mathbf{e}_1 = (1, \dots, 0)^T$, X_x es la matriz con $(1, x - X_i, \dots, (x - X_i)^p)$ en la fila i , $W_x = \text{diag}(K_h(x_1 - x), \dots, K_h(x_n - x))$ es la matriz de pesos, e $\mathbf{Y} = (Y_1, \dots, Y_n)^T$ es el vector de observaciones de la respuesta.

Se puede pensar que se obtiene aplicando un suavizado polinómico a (X_i, Y_i) :

$$\hat{m} = S \mathbf{Y},$$

siendo S la matriz de suavizado con $s_{X_i}^T$ en la fila i .

Habitualmente se considera:

- $p = 0$: Estimador Nadaraya-Watson.
- $p = 1$: Estimador lineal local.

Asintóticamente el estimador lineal local tiene un sesgo menor que el de Nadaraya-Watson (pero del mismo orden) y la misma varianza (e.g. Fan and Gijbels, 1996). Sin embargo, su principal ventaja es que se ve menos afectado por el denominado efecto frontera (*edge effect*).

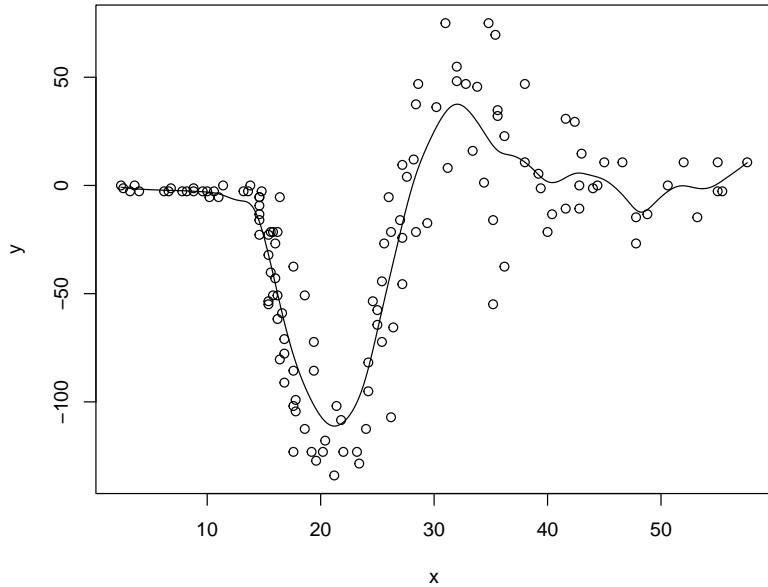
Aunque el paquete base de R incluye herramientas para la estimación tipo núcleo de la regresión (`lowess()`, `ksmooth()`), recomiendan el uso del paquete `KernSmooth` (Wand y Ripley, 2019). Otros paquetes incluyen más funcionalidades: `sm` (Bowman y Azzalini, 2019), `np` (Tristen y Jeffrey, 2019), `npsp` (Fernández-Casal, 2019), entre otros.

Como ejemplo emplearemos el conjunto de datos MASS::`mcycle` que contiene mediciones de la aceleración de la cabeza en una simulación de un accidente de motocicleta, utilizado para probar cascos protectores.

```
data(mcycle, package = "MASS")
x <- mcycle$times
y <- mcycle$accel

library(KernSmooth)
h <- dpill(x, y) # Método plug-in de Ruppert, Sheather y Wand (1995)
fit <- locpoly(x, y, bandwidth = h) # Estimación lineal local
plot(x, y)
lines(fit)
```

¹Se puede pensar que se están estimando los coeficientes de un desarrollo de Taylor de $m(x_0)$.



Hay que tener en cuenta que el paquete `KernSmooth` no implementa los métodos `predict()` y `residuals()`:

```
est <- approx(fit, xout = x)$y # est <- predict(fit)
resid <- y - est # resid <- residuals(fit)
```

Tampoco calcula medidas de bondad de ajuste, aunque podríamos obtener fácilmente un (pseudo) R-cuadrado:

```
r.squared <- 1 - sum(resid^2)/sum((y - mean(y))^2)
r.squared
```

```
## [1] 0.8023864
```

7.3.1 Estimación de la varianza

En el caso heterocedástico, se puede obtener una estimación de la varianza $\sigma^2(x)$ mediante suavizado local de los residuos al cuadrado (Fan y Yao, 1998). Mientras que en el caso homocedástico, se puede obtener una estimación de la varianza a partir de la suma de cuadrados residual y la matriz de suavizado:

$$\hat{\sigma}^2 = \frac{RSS}{df_e},$$

siendo $RSS = \sum_{i=1}^n (Y_i - \hat{m}(X_i))^2$ y $df_e = \text{tr}(I - S)$ (de forma análoga al caso lineal), o alternativamente $df_e = \text{tr}((I - S^T)(I - S))$, es una aproximación de los grados de libertad del error.

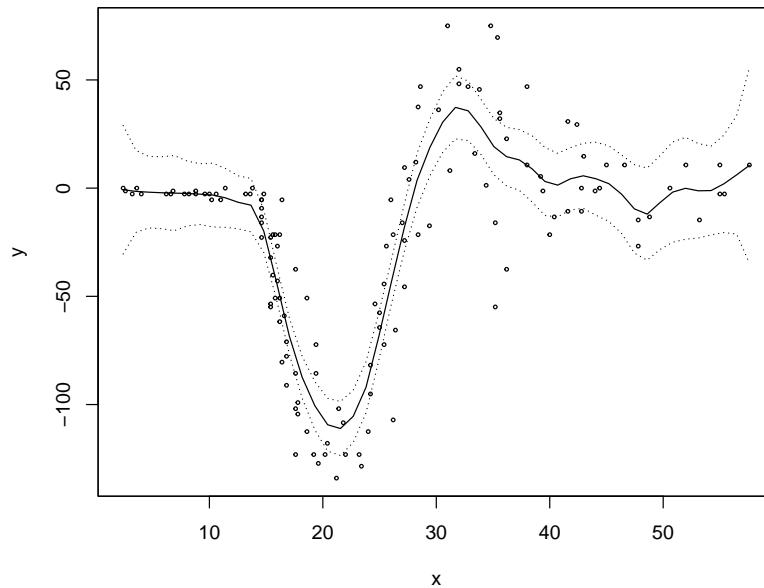
Adicionalmente:

$$\widehat{\text{Var}}(\hat{m}_h(x_i)) = \hat{\sigma}^2 \sum_{j=1}^n s_{ij}^2$$

Uno de los pocos paquetes de R que implementan la estimación de la varianza y el cálculo de intervalos de confianza es el paquete² `sm` (Bowman y Azzalini, 2019).

²El paquete `np` calcula estimaciones similares aunque no documenta la aproximación que emplea. También implementa bootstrap uniforme y bootstrap por bloques.

```
library(sm)
hcv <- hcv(x, y) # Método de validación cruzada
fit.sm <- sm.regression(x, y, h = hcv, display = "se")
```



```
fit.sm$sigma
```

```
## [1] 22.82508
```

Alternativamente se podría emplear bootstrap.

7.4 Ejemplos

En esta sección nos centraremos en el bootstrap en la estimación tipo núcleo de la función de regresión, para la aproximación de la precisión y el sesgo, y también para el cálculo de intervalos de confianza y de predicción.

7.4.1 Bootstrap residual

El modelo ajustado de regresión se puede emplear para estimar la respuesta media $m(x_0)$ cuando la variable explicativa toma un valor concreto x_0 . En este caso también podemos emplear el bootstrap residual (Sección 3.7.2) para realizar inferencias acerca de la media. La idea sería aproximar la distribución del error de estimación $\hat{m}_h(x_0) - m(x_0)$ por la distribución bootstrap de $\hat{m}_h^*(x_0) - \hat{m}_g(x_0)$.

Para reproducir adecuadamente el sesgo del estimador, la ventana g debería ser asintóticamente mayor que h (de orden $n^{-1/5}$). Análogamente al caso de la densidad, la recomendación sería emplear la ventana óptima para la estimación de $m''(x_0)$, de orden $n^{-1/9}$ (Sección 7.2.1). Sin embargo, en la práctica es habitual emplear $g = h$ para evitar la selección de esta ventana (lo que además facilita emplear herramientas como el paquete `boot`). Otra alternativa podría ser asumir que $g \simeq n^{1/5}h/n^{1/9}$ como se hace a continuación.

```
n <- length(x)
g <- h * n^(4/45) # h*n^(-1/9)/n^(-1/5)
# g <- h
fitg <- locpoly(x, y, bandwidth = g) # puntos de estimación/predicción
# fitg$y <- predict(fitg, newdata = fitg$x)
```

```

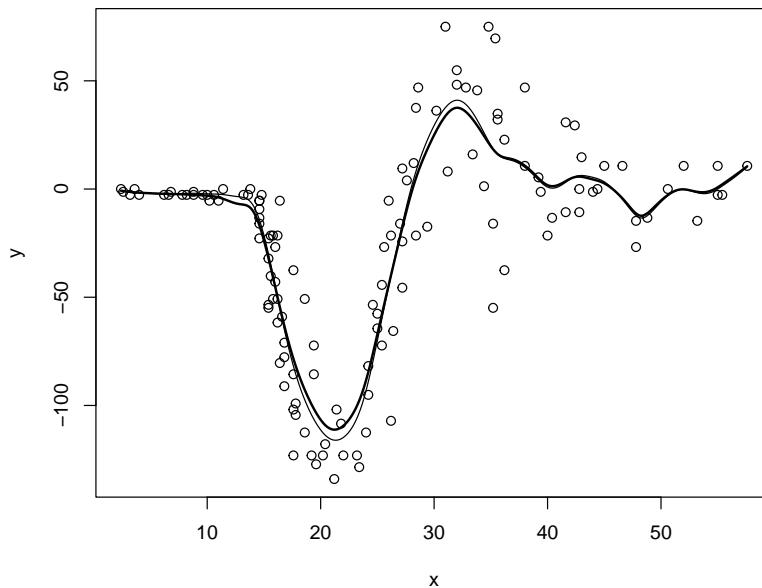
estg <- approx(fitg, xout = x)$y # puntos observaciones
# estg <- predict(fitg)
# resid2 <- y - estg # resid2 <- residuals(fitg)

# Remuestreo
set.seed(1)
B <- 2000
stat_fit_boot <- matrix(nrow = length(fit$x), ncol = B)
resid0 <- resid - mean(resid)
for (k in 1:B) {
  y_boot <- estg + sample(resid0, replace = TRUE)
  fit_boot <- locpoly(x, y_boot, bandwidth = h)$y
  stat_fit_boot[ , k] <- fit_boot - fitg$y
}

# Calculo del sesgo y error estándar
bias <- apply(stat_fit_boot, 1, mean)
std.err <- apply(stat_fit_boot, 1, sd)

# Representar estimación y corrección de sesgo bootstrap
plot(x, y)
lines(fit, lwd = 2)
lines(fit$x, fit$y - bias)

```



NOTA: De forma análoga al caso lineal (Sección 3.7.2), se podrían reescalar los residuos a partir de la matriz de suavizado (empleando los paquetes `sm` o `npsp`).

7.4.2 Intervalos de confianza y predicción

De forma análoga al caso de la estimación de la densidad mostrado en la Sección 6.6.2, podemos calcular estimaciones por intervalo de confianza (puntuales) por el método percentil (básico):

```

alfa <- 0.05
pto_crit <- apply(stat_fit_boot, 1, quantile, probs = c(alfa/2, 1 - alfa/2))

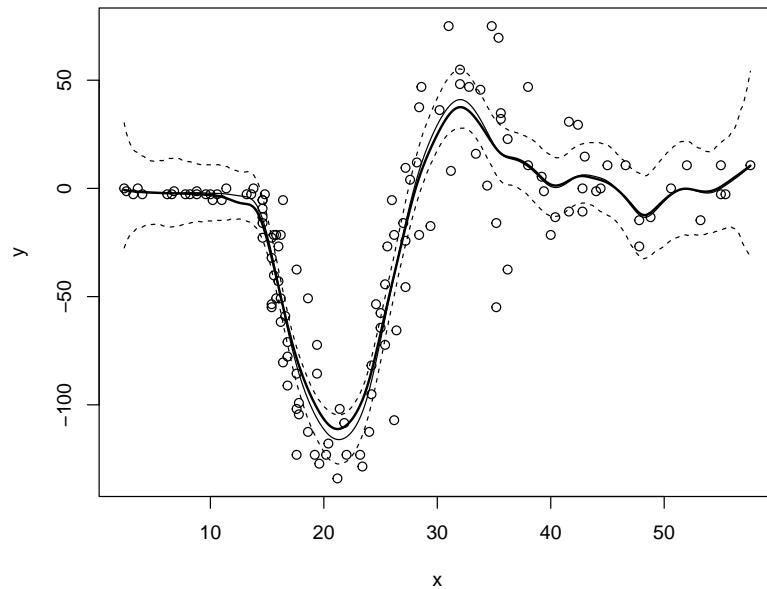
```

```

ic_inf_boot <- fit$y - pto_crit[2, ]
ic_sup_boot <- fit$y - pto_crit[1, ]

plot(x, y)
lines(fit, lwd = 2)
lines(fit$x, fit$y - bias)
lines(fit$x, ic_inf_boot, lty = 2)
lines(fit$x, ic_sup_boot, lty = 2)

```



El modelo ajustado también es empleado para predecir una nueva respuesta individual $Y(x_0)$ para un valor concreto x_0 de la variable explicativa.

En el caso de errores independientes $\hat{Y}(x_0) = \hat{m}_h(x_0)$, pero si estamos interesados en realizar inferencias sobre el error de predicción $r(x_0) = Y(x_0) - \hat{Y}(x_0)$, a la variabilidad de $\hat{m}_h(x_0)$ debida a la muestra, se añade la variabilidad del error $\varepsilon(x_0)$.

La idea sería aproximar la distribución del error de predicción:

$$r(x_0) = Y(x_0) - \hat{Y}(x_0) = m(x_0) + \varepsilon(x_0) - \hat{m}_h(x_0)$$

por la distribución bootstrap de:

$$r^*(x_0) = Y^*(x_0) - \hat{Y}^*(x_0) = \hat{m}_g(x_0) + \varepsilon^*(x_0) - \hat{m}_h^*(x_0)$$

```

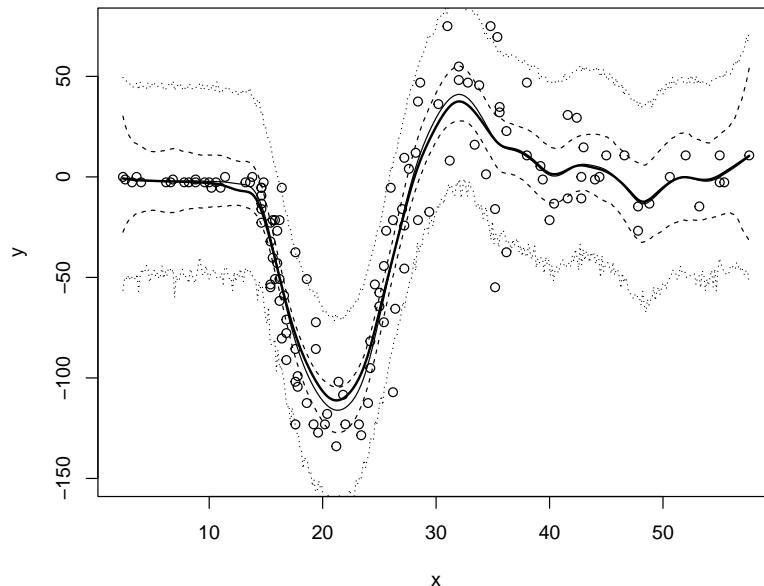
# Remuestreo
set.seed(1)
n_pre <- length(fit$x)
stat_pred_boot <- matrix(nrow = n_pre, ncol = B)
for (k in 1:B) {
  y_boot <- estg + sample(resid0, replace = TRUE)
  fit_boot <- locpoly(x, y_boot, bandwidth = h)$y
  pred_boot <- fitg$y + sample(resid0, n_pre, replace = TRUE)
  stat_pred_boot[, k] <- pred_boot - fit_boot
}

# Cálculo de intervalos de predicción

```

```
# por el método percentil (básico)
alfa <- 0.05
pto_crit_pred <- apply(stat_pred_boot, 1, quantile, probs = c(alfa/2, 1 - alfa/2))
ip_inf_boot <- fit$y + pto_crit_pred[1, ]
ip_sup_boot <- fit$y + pto_crit_pred[2, ]

plot(x, y, ylim = c(-150, 75))
lines(fit, lwd = 2)
lines(fit$x, fit$y - bias)
lines(fit$x, ic_inf_boot, lty = 2)
lines(fit$x, ic_sup_boot, lty = 2)
lines(fit$x, ip_inf_boot, lty = 3)
lines(fit$x, ip_sup_boot, lty = 3)
```



En este caso puede no ser recomendable considerar errores i.i.d., sería de esperar heterocedasticidad (e incluso dependencia temporal). El bootstrap residual se puede extender al caso heterocedástico y/o dependencia (e.g. Castillo-Páez *et al.*, 2019, 2020).

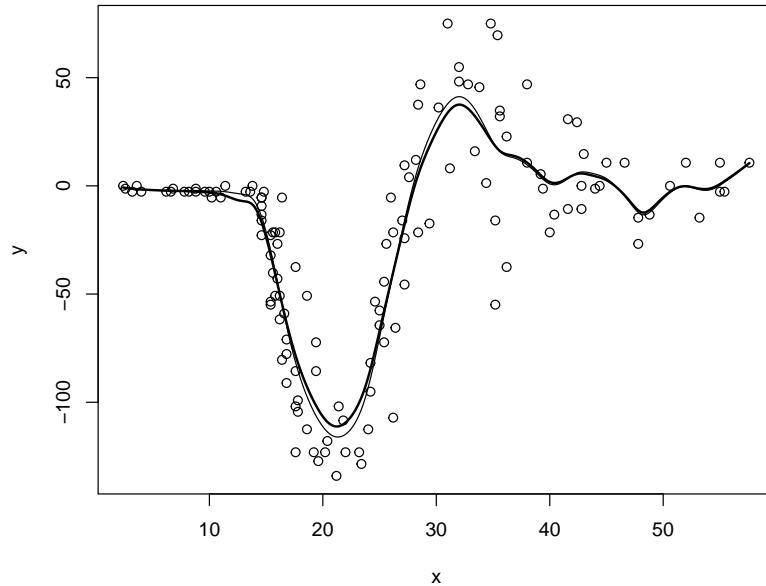
7.4.3 Wild bootstrap

Como se describe en la Sección 7.2.1 en el caso heterocedástico se puede emplear wild bootstrap.

```
# Remuestreo
set.seed(1)
B <- 1000
fit_boot <- matrix(nrow = n_pre, ncol = B)
for (k in 1:B) {
    rwild <- sample(c((1 - sqrt(5))/2, (1 + sqrt(5))/2), n, replace = TRUE,
                     prob = c((5 + sqrt(5))/10, 1 - (5 + sqrt(5))/10))
    y_boot <- estg + resid*rwild
    fit_boot[ , k] <- locpoly(x, y_boot, bandwidth = h)$y
    # OJO: bootstrap percentil directo
}
```

```
# Calculo del sesgo y error estándar
bias <- apply(fit_boot, 1, mean, na.rm = TRUE) - fitg$y
std.err <- apply(fit_boot, 1, sd, na.rm = TRUE)

# Representar estimación y corrección de sesgo bootstrap
plot(x, y)
lines(fit, lwd = 2)
lines(fit$x, fit$y - bias)
```



7.4.4 Ejercicio

Siguiendo con el conjunto de datos MASS::mcycle, emplear wild bootstrap para obtener estimaciones por intervalo de confianza de la función de regresión de `accel` a partir de `times` mediante bootstrap percentil básico. Comparar los resultados con los obtenidos mediante bootstrap residual (comentar las diferencias y cuál de las aproximaciones sería más adecuada para este caso).

Capítulo 8

El Bootstrap con datos censurados

En este capítulo se hace una introducción a los datos censurados y se presentan diversos métodos de remuestreo para este contexto, analizando la validez de los mismos.

8.1 Introducción a los datos censurados

Considérese una variable de interés, X , no negativa que no siempre es posible observar (por ejemplo un tiempo de vida) pues, en ocasiones, ocurre otro fenómeno previo, cuyo tiempo hasta su ocurrencia, C , puede ser anterior a la variable de interés (es decir, $C < X$). Cuando X es un tiempo de vida ante una enfermedad mortal, la variable C suele representar el tiempo hasta el fin del estudio, el tiempo hasta que el individuo fallezca por otra causa o el tiempo hasta que se produce una pérdida de seguimiento. Es habitual definir el indicador de no censura $\delta = \mathbf{1}_{\{X \leq C\}}$. Si $C < X$ diremos que la observación es censurada y sólo seremos capaces de observar C junto con el valor de δ . Cuando $X \leq C$ entonces somos capaces de observar la variable de interés y además el valor de δ .

En resumen, en lugar de observar la muestra (X_1, X_2, \dots, X_n) , sólo podemos observar

$$((T_1, \delta_1), (T_2, \delta_2), \dots, (T_n, \delta_n)),$$

siendo $T_i = \min\{X_i, C_i\}$ los tiempos de vida observados y

$$\delta_i = \mathbf{1}_{\{X_i \leq C_i\}} = \mathbf{1}_{\{T_i = X_i\}}$$

los indicadores de censura, para $i = 1, 2, \dots, n$. En el modelo de censura aleatoria por la derecha, que es el más habitual, se supone que X_i y C_i ($i = 1, 2, \dots, n$) son independientes. Además (X_1, X_2, \dots, X_n) son mutuamente independientes, como también lo son (C_1, C_2, \dots, C_n) .

Denotando por F (respectivamente G y H) la función de distribución de la variable aleatoria X (respectivamente C y T), la condición de independencia implica que

$$1 - H(t) = (1 - F(t))(1 - G(t)).$$

8.1.1 Estimador de Kaplan-Meier

Puede verse fácilmente que, bajo censura aleatoria por la derecha, la distribución empírica $F_n(t) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{T_i \leq t\}}$ deja de ser consistente. En este contexto el estimador no paramétrico de máxima verosimilitud de la función de distribución es el estimador límite-producto, propuesto por Kaplan y Meier (1958), obtenido a partir de la función de supervivencia ($S(t) = 1 - F(t)$):

$$\hat{S}(t) = 1 - \hat{F}(t) = \prod_{T_{(i)} \leq t} \left(\frac{n-i}{n-i+1} \right)^{\delta_{(i)}},$$

siendo $(T_{(1)}, T_{(2)}, \dots, T_{(n)})$ la muestra de estadísticos ordenados de los tiempos de vida observados y $(\delta_{(1)}, \delta_{(2)}, \dots, \delta_{(n)})$ los correspondientes concomitantes.

Ejemplo 8.1 (Estimación de Kaplan-Meier). Se observan los datos censurados: (2.1,0), (3.2,1), (1.2,1), (4.3,0), (1.8,1), (3.9,1), (2.7,0), (2.5,1). El estimador resulta:

$$\hat{F}(t) = \begin{cases} 0 & \text{si } t < 1.2 \\ 0.125 & \text{si } 1.2 \leq t < 1.8 \\ 0.25 & \text{si } 1.8 \leq t < 2.5 \\ 0.4 & \text{si } 2.5 \leq t < 3.2 \\ 0.6 & \text{si } 3.2 \leq t < 3.9 \\ 0.8 & \text{si } 3.9 \leq t \end{cases}$$

En R se recomienda emplear el paquete **survival** para el análisis de datos censurados. Podemos utilizar la función **survfit()** para obtener la estimación Kaplan-Meier de la función de supervivencia (y a partir de ella la de la distribución). En este caso podríamos utilizar el siguiente código [Figura 8.1]:

```
datcen <- data.frame(t = c(2.1, 3.2, 1.2, 4.3, 1.8, 3.9, 2.7, 2.5),
                      cen = c(0, 1, 1, 0, 1, 1, 0, 1))

library(survival)
fit <- survfit(Surv(t, cen)~1, data = datcen)
summary(fit)

## Call: survfit(formula = Surv(t, cen) ~ 1, data = datcen)
##
##    time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    1.2      8      1     0.875   0.117    0.6734      1
##    1.8      7      1     0.750   0.153    0.5027      1
##    2.5      5      1     0.600   0.182    0.3315      1
##    3.2      3      1     0.400   0.203    0.1477      1
##    3.9      2      1     0.200   0.174    0.0363      1

old.par <- par(mfrow = c(1, 2))
plot(fit, main = "Método plot de un objeto 'survfit' ")
legend("bottomleft", c("supervivencia", "conf.int"), lty = 1:2)

with(fit, {
  plot(c(0, time), c(1, surv), type = "s", lty = 2,
       main = "Estimaciones funciones supervivencia y distribución",
       xlab = "t", ylab = "", ylim = c(0, 1))
  lines(c(0, time), 1 - c(1, surv), type = "s")
  legend("bottomright", c("supervivencia", "distribución"), lty = 2:1)
})
```

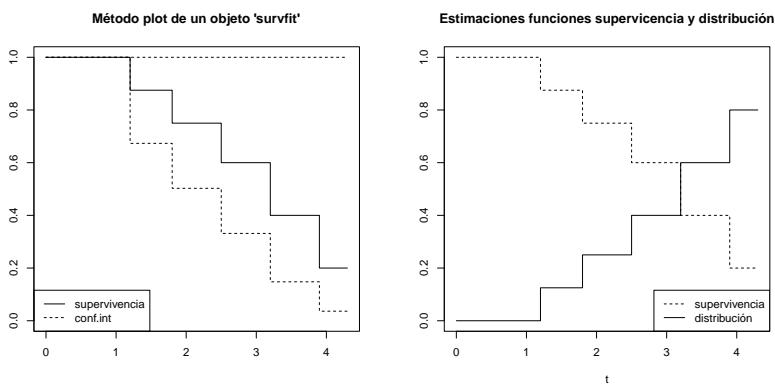


Figura 8.1: Estimaciones Kaplan-Meier de la función de supervivencia y de la función de distribución.

```
par(old.par)
```

8.1.2 Distribución asintótica del estimador de Kaplan-Meier

El estimador de Kaplan-Meier sólo otorga pesos positivos a los datos no censurados, aunque la forma de distribuirse los datos censurados en medio de los no censurados afecta a los pesos de estos últimos. Por otra parte, en ausencia de censura (es decir $\delta_i = 1$, $i = 1, 2, \dots, n$), el estimador de Kaplan-Meier coincide con la distribución empírica.

La obtención de la propiedades de sesgo y varianza asintóticos y distribución límite del estimador de Kaplan-Meier es mucho más laboriosa que en el caso de la distribución empírica, en un contexto sin censura. Esto es así porque el estimador de Kaplan-Meier deja de ser una suma de variables iid, como sí ocurre con la empírica.

Breslow y Crowley (1974) obtienen el siguiente resultado para la distribución límite para el estimador de Kaplan-Meier:

$$\sqrt{n} (\hat{F}(t) - F(t)) \xrightarrow{d} \mathcal{N}(0, \sigma^2(t)),$$

siendo

$$\begin{aligned} \sigma^2(t) &= (1 - F(t))^2 \int_0^t \frac{dH_1(u)}{(1 - H(u))^2}, \quad t \leq H^{-1}(1), \\ H_1(u) &= P(X \leq u, X \leq C) = P(T \leq u, \delta = 1). \end{aligned}$$

También existen resultados de convergencia en distribución del proceso estocástico

$$\left\{ \sqrt{n} (\hat{F}(t) - F(t)) : t \in [0, H^{-1}(1)] \right\}$$

a un proceso gaussiano límite.

8.2 Remuestreos Bootstrap en presencia de censura

Estos métodos tratan del mecanismo bootstrap para aproximar la distribución de un estadístico, $R(\mathbf{T}, \delta)$, siendo $\mathbf{T} = (T_1, T_2, \dots, T_n)$ y $\delta = (\delta_1, \delta_2, \dots, \delta_n)$. Los dos siguientes métodos de remuestreo fueron propuestos por Efron (1981).

8.2.1 El bootstrap simple

Procede de la siguiente forma:

1. Construir la distribución empírica bidimensional, $F_n^{T, \delta}$, de la muestra $\{(T_1, \delta_1), (T_2, \delta_2), \dots, (T_n, \delta_n)\}$.
2. Arrojar remuestras $\{(T_1^*, \delta_1^*), (T_2^*, \delta_2^*), \dots, (T_n^*, \delta_n^*)\}$ a partir de dicha distribución empírica. Esto es tanto como decir que

$$P^*((T^*, \delta^*) = (T_i, \delta_i)) = \frac{1}{n}, \text{ para } i = 1, 2, \dots, n.$$

3. Evaluar el estadístico de interés en el vector que contiene la remuestra bootstrap: $R^* = R(\mathbf{T}^*, \delta^*)$, con $\mathbf{T}^* = (T_1^*, T_2^*, \dots, T_n^*)$ y $\delta^* = (\delta_1^*, \delta_2^*, \dots, \delta_n^*)$.
4. Aproximar la distribución en el muestreo del estadístico $R(\mathbf{T}, \delta)$ por la distribución en el remuestreo de $R(\mathbf{T}^*, \delta^*)$.

Este método es de muy rápida implementación y ejecución.

8.2.2 El bootstrap obvio

Para detallar el método es necesario definir el estimador de Kaplan-Meier, $\hat{G}(t)$, de la variable censurante, a partir de

$$1 - \hat{G}(t) = \prod_{T_{(i)} \leq t} \left(\frac{n-i}{n-i+1} \right)^{1-\delta_{(i)}}.$$

Observemos que este estimador es totalmente semejante al de Kaplan-Meier de la variable de interés pero simplemente reemplazando cada valor $\delta_{(i)}$ por $1 - \delta_{(i)}$.

El mecanismo de remuestreo procede como sigue:

1. Construir los estimadores de Kaplan-Meier de las distribuciones de la variable de interés, $\hat{F}(t)$, y de la variable censurante, $\hat{G}(t)$.
2. Para cada índice $i = 1, 2, \dots, n$, arrojar observaciones bootstrap independientes, X_i^* con distribución \hat{F} y C_i^* con distribución \hat{G} .
3. Definir $T_i^* = \min\{X_i^*, C_i^*\}$ y $\delta_i^* = \mathbf{1}_{\{X_i^* \leq C_i^*\}}$, para $i = 1, 2, \dots, n$, y considerar la remuestra bootstrap (\mathbf{T}^*, δ^*) , con $\mathbf{T}^* = (T_1^*, T_2^*, \dots, T_n^*)$ y $\delta^* = (\delta_1^*, \delta_2^*, \dots, \delta_n^*)$.
4. Aproximar la distribución en el muestreo del estadístico $R(\mathbf{T}, \delta)$ por la distribución en el remuestreo de su análogo bootstrap, $R(\mathbf{T}^*, \delta^*)$.

Obviamente, este método de remuestreo imita fielmente el modelo de datos censurados por la derecha. Su ejecución es considerablemente más lenta que la del método simple, pues necesita de la construcción de los estimadores de Kaplan-Meier, de la obtención de remuestras a partir de ellos y de algunos cálculos adicionales.

8.3 Relaciones entre los métodos de remuestreo bajo censura

8.3.1 Equivalencia entre el bootstrap simple y el obvio

Es fácil demostrar que el bootstrap simple y el obvio son planes de remuestreo equivalentes (cuando se supone que en la muestra no existe ninguna observación no censurada que esté empatada con otra censurada). Esta equivalencia se establece en el sentido de que la distribución bootstrap de (T^*, δ^*) es la misma para cualquiera de los dos métodos.

Así, si (T^*, δ^*) se genera mediante el método obvio, entonces

$$\begin{aligned} P^*(T^* > t) &= P^*(X^* > t, C^* > t) \\ &= P^*(X^* > t) P^*(C^* > t) = (1 - \hat{F}(t)) (1 - \hat{G}(t)) \\ &= \left[\prod_{T_{(i)} \leq t} \left(\frac{n-i}{n-i+1} \right)^{\delta_{(i)}} \right] \left[\prod_{T_{(i)} \leq t} \left(\frac{n-i}{n-i+1} \right)^{1-\delta_{(i)}} \right] \\ &= \prod_{T_{(i)} \leq t} \frac{n-i}{n-i+1} = \prod_{i=1}^{\#\{T_{(j)} \leq t\}} \frac{n-i}{n-i+1} \\ &= \frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \dots \cdot \frac{n - \#\{T_{(j)} \leq t\}}{n - \#\{T_{(j)} \leq t\} + 1} \\ &= \frac{n - \#\{T_{(j)} \leq t\}}{n} = 1 - H_n(t) = \frac{\#\{T_{(j)} > t\}}{n}, \end{aligned}$$

siendo $H_n(t)$ la distribución empírica de la muestra (T_1, T_2, \dots, T_n) .

Esto demuestra que la distribución bootstrap marginal de T^* es la misma para ambos remuestreos. Sólo resta probar pues que la distribución condicionada $\delta^*|_{T^*=T_i}$ es idéntica en ambos casos. Pero esto

es inmediato ya que, en los dos remuestreos esa distribución condicionada es la degenerada en el valor δ_i .

8.3.2 El bootstrap de Reid

Es otro método alternativo propuesto por Reid (1981). Consta de los siguientes pasos:

1. Construir el estimador de Kaplan-Meier, $\hat{F}(t)$, de la muestra original.
2. Arrojar remuestras bootstrap (todas formadas por observaciones no censuradas, T_i^* , $i = 1, 2, \dots, n$) a partir de $\hat{F}(t)$.
3. Aproximar la distribución en el muestreo de $R(\mathbf{T}, \delta)$, por la distribución bootstrap de $R(\mathbf{T}^*, \mathbf{1})$, siendo $\mathbf{1}$ el vector formado por n unos.

8.3.3 Validez de los planes de remuestreo

Akritas (1986) demuestra que los procesos bootstrap

$$\begin{aligned} & \sqrt{n} (\hat{F}_{Efron}^*(t) - \hat{F}(t)) \\ & \sqrt{n} (\hat{F}_{Reid}^*(t) - \hat{F}(t)) \end{aligned}$$

tienden a sendos procesos límite distintos. Aquí \hat{F}_{Efron}^* denota la versión bootstrap del estimador de Kaplan-Meier bajo el remuestreo de Efron (cualquiera de ellos, ya que el remuestreo simple y el obvio son equivalentes) y \hat{F}_{Reid}^* es la correspondiente versión bootstrap del estimador de Kaplan-Meier bajo el remuestreo de Reid (una distribución empírica, al fin y al cabo, porque en el remuestreo de Reid todas las observaciones son no censuradas).

Además el proceso límite del estimador de Kaplan-Meier, $\sqrt{n} (\hat{F}(t) - F(t))$, es el mismo que el del bootstrap de Efron. Como consecuencia el remuestreo de Efron es consistente y el de Reid es inconsistente.

8.4 Implementación en R (con los paquetes `boot` y `survival`)

La función `censboot()` del paquete `boot` implementa distintos métodos de remuestreo para datos censurados. Por defecto utiliza el bootstrap simple (`sim = "ordinary"`) y su uso es prácticamente igual al del bootstrap uniforme con la función `boot()` (descrita en la Sección 1.4.1), la única diferencia es que la función `statistic` solo tiene los datos como único parámetro (aunque en este caso podríamos emplear también la función `boot()`).

8.4.1 Bootstrap simple

Como ejemplo utilizaremos el conjunto de datos `channing` del paquete `boot`, que contiene la edad de entrada y de partida o muerte de las personas que pasaron por el centro de retiro ‘Channing House’ (Palo Alto, California), desde su apertura en 1964 hasta el 1 de julio de 1975 (ver Sección 3.5 y ‘Practical 3.2’, de Davison y Hinkley, 1997). En primer lugar consideraremos únicamente la muestra de hombres:

```
# Datos
library(boot)
data(channing)

# Calcular edad (de partida o muerte) en años
channing$age <- (channing$entry + channing$time)/12

# Seleccionar hombres (y de paso hacer que `index = c(1, 2)` para `censboot()`)
chan <- subset(channing, sex=="Male", c(age, cens))

# Estimación supervivencia
library(survival)
```

```

chan.F <- survfit(Surv(age, cens)~1, data = chan)
chan.F

## Call: survfit(formula = Surv(age, cens) ~ 1, data = chan)
##
##      n  events  median 0.95LCL 0.95UCL
##    97.0    46.0    87.0    85.8    90.4

# plot(chan.F)
# Estimaciones de interés
with(chan.F,
  c(s75 = max(surv[time > 75]), s85 = max(surv[time > 85]),
    p75 = min(time[surv <= 0.75]), p50 = min(time[surv <= 0.5])))
)

##          s75          s85          p75          p50
## 0.9160745  0.6347541 82.4166667 87.0000000

# Bootstrap
# library(boot)
chan.stat <- function(data) {
  s <- survfit(Surv(age, cens)~1, data = data)
  with(s, c(s75 = max(surv[time > 75]), s85 = max(surv[time > 85]),
            p75 = min(time[surv <= 0.75]), p50 = min(time[surv <= 0.5])))
}
set.seed(1)
chan.boot <- censboot(chan, chan.stat, R = 199) # sim = "ordinary"
chan.boot

##
## CASE RESAMPLING BOOTSTRAP FOR CENSORED DATA
##
##
## Call:
## censboot(data = chan, statistic = chan.stat, R = 199)
##
##
## Bootstrap Statistics :
##      original      bias   std. error
## t1*  0.9160745 -0.006995081  0.03133656
## t2*  0.6347541 -0.003538939  0.05595748
## t3* 82.4166667 -0.040619765  1.22517032
## t4* 87.0000000  0.195142379  1.07267425

```

8.4.2 Otros métodos de remuestreo

La función `censboot()` implementa otros dos métodos de remuestreo, `sim = c("cond", "weird")`, aunque en ambos casos hay que establecer en el parámetro `F.surv` la estimación de Kaplan-Meier de la supervivencia y, si `sim = "cond"`, la correspondiente a la variable censurante en `G.surv`. Se recomienda estimarlas con la función `survfit()` del paquete `survival` (ver Figura 8.2):

```

# Estimación supervivencia variable censurante
chan.G <- survfit(Surv(age, 1-cens)~1, data = chan)
# Representación
old.par <- par(mfrow = c(1, 2))
plot(chan.F, main = "Supervivencia (edad)", mark.time = TRUE,
     xlim = c(60, 100))
plot(chan.G, main = "Supervivencia variable censurante (partida)",
     mark.time = TRUE, xlim = c(60, 100))

```

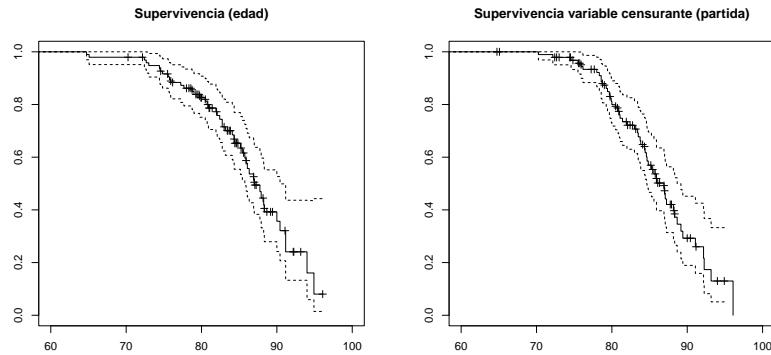


Figura 8.2: Estimaciones de la supervivencia (izquierda; indicando los tiempos de las observaciones censuradas) y de la variable censurante (derecha; indicando los de las no censuradas).

```
par(old.par)
```

En el *bootstrap condicional* (`sim = "cond"`) se condiciona el muestreo al patrón de censura observado (en lugar de fijarlo a **1** como en el bootstrap de Reid; Sección 8.3.2). El mecanismo es similar al del bootstrap obvio (Sección 8.2.2):

1. Construir los estimadores de Kaplan-Meier de las distribuciones de la variable de interés, $\hat{F}(t)$, y de la variable censurante, $\hat{G}(t)$.
2. Para cada índice $i = 1, 2, \dots, n$, generar X_i^* independientes con distribución \hat{F} .
3. Si la i -ésima observación está censurada ($\delta_i = 0$) se toma $C_i^* = T_i$ y si no ($\delta_i = 1$) se genera un valor de la estimación de la distribución de la variable censurante condicionada a $C > T_i$:

$$\hat{G}\left(t \mid t_{>T_i}\right) = \frac{\hat{G}(t) - \hat{G}(T_i)}{1 - \hat{G}(T_i)}.$$

4. Definir $T_i^* = \min\{X_i^*, C_i^*\}$ y $\delta_i^* = \mathbf{1}_{\{X_i^* \leq C_i^*\}}$, para $i = 1, 2, \dots, n$, y considerar la remuestra bootstrap (\mathbf{T}^*, δ^*) , con $\mathbf{T}^* = (T_1^*, T_2^*, \dots, T_n^*)$ y $\delta^* = (\delta_1^*, \delta_2^*, \dots, \delta_n^*)$.

El otro método (`sim = "weird"`) es el denominado *weird bootstrap* (Andersen et al., 1993) que emplea la estimación de Nelson-Aalen de la función de riesgo acumulada para generar los valores (e.g. Sección 3.5.2 de Davison y Hinkley, 1997).

El siguiente código muestra un ejemplo de la aplicación de ambos métodos:

```
chan.boot2 <- censboot(chan, chan.stat, R = 199, F.surv = chan.F,
                        G.surv = chan.G, sim = "cond")
chan.boot2

##
## CONDITIONAL BOOTSTRAP FOR CENSORED DATA
##
##
## Call:
## censboot(data = chan, statistic = chan.stat, R = 199, F.surv = chan.F,
##          G.surv = chan.G, sim = "cond")
##
##
## Bootstrap Statistics :
##      original      bias    std. error
```

```

## t1* 0.9160745 0.002340590 0.02796666
## t2* 0.6347541 -0.001057025 0.05382609
## t3* 82.4166667 0.019681742 1.21793756
## t4* 87.0000000 0.286013400 0.95041994

chan.boot3 <- censboot(chan, chan.stat, R = 199, F.surv = chan.F,
                      sim = "weird")
chan.boot3

##
## WEIRD BOOTSTRAP FOR CENSORED DATA
##
##
## Call:
## censboot(data = chan, statistic = chan.stat, R = 199, F.surv = chan.F,
##           sim = "weird")
##
##
## Bootstrap Statistics :
##          original      bias    std. error
## t1* 0.9160745 -4.082326e-05 0.02825475
## t2* 0.6347541  1.639197e-03 0.05086460
## t3* 82.4166667 2.303183e-02 1.14079354
## t4* 87.0000000 2.458124e-01 0.96511648

```

8.5 Ejercicios

Ejercicio 8.1 (Bootstrap censurado por estratos). Analizar el conjunto de datos `channing` completo, teniendo en cuenta el sexo como estrato (i.e. `Surv(age, cens) ~ sex` y `strata = chan$sex`)

```

# Datos
data(channing)
# Calcular edad (de partida o muerte) en años
channing$age <- (channing$entry + channing$time)/12
# Seleccionar variables
chan <- channing[c("age", "cens", "sex")]

# Estimación supervivencia
library(survival)
chan.F <- survfit(Surv(age, cens) ~ sex, data = chan)
chan.F

## Call: survfit(formula = Surv(age, cens) ~ sex, data = chan)
##
##          n events median 0.95LCL 0.95UCL
## sex=Female 365     130      88    86.7    89.5
## sex=Male   97      46      87    85.8    90.4

plot(chan.F, lty = 1:2, xlim = c(60, 100))

res <- summary(chan.F)
# res
str(res)

## List of 19
## $ n            : int [1:2] 365 97
## $ time         : num [1:146] 67 68.5 69.2 70 70.4 ...
## $ n.risk       : num [1:146] 364 359 355 353 352 346 344 340 335 334 ...

```

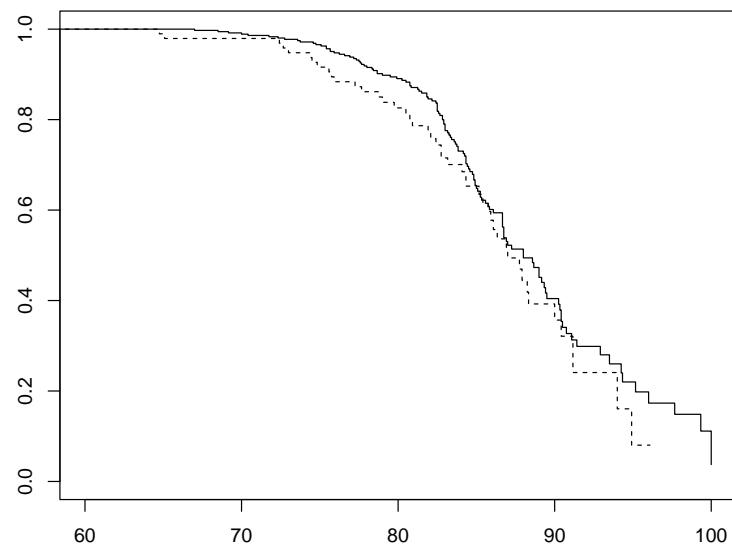


Figura 8.3: Estimaciones de la supervivencia.

```

## $ n.event      : num [1:146] 1 1 1 1 1 1 1 1 1 1 ...
## $ n.censor     : num [1:146] 2 3 3 1 0 6 0 3 4 0 ...
## $ surv         : num [1:146] 0.997 0.994 0.992 0.989 0.986 ...
## $ std.err       : num [1:146] 0.00274 0.0039 0.00479 0.00554 0.00619 ...
## $ cumhaz        : num [1:146] 0.00275 0.00553 0.00835 0.01118 0.01402 ...
## $ std.chaz      : num [1:146] 0.00275 0.00391 0.00482 0.00559 0.00627 ...
## $ strata        : Factor w/ 2 levels "sex=Female","sex=Male": 1 1 1 1 1 1 1 1 1 ...
## $ type          : chr "right"
## $ logse         : logi TRUE
## $ conf.int      : num 0.95
## $ conf.type     : chr "log"
## $ lower         : num [1:146] 0.992 0.987 0.982 0.978 0.974 ...
## $ upper         : num [1:146] 1 1 1 1 0.998 ...
## $ call          : language survfit(formula = Surv(age, cens) ~ sex, data = chan)
## $ table         : num [1:2, 1:9] 365 97 365 97 365 ...
## ..- attr(*, "dimnames")=List of 2
## ... $ : chr [1:2] "sex=Female" "sex=Male"
## ... $ : chr [1:9] "records" "n.max" "n.start" "events" ...
## $ rmean.endtime: num [1:2] 101 101
## - attr(*, "class")= chr "summary.survfit"

# Estimaciones de interés
res$table[, c("*rmean", "median")]

##           *rmean median
## sex=Female 88.46153    88
## sex=Male   86.89935    87
as.numeric(res$table[, c("*rmean", "median")])

## [1] 88.46153 86.89935 88.00000 87.00000

```

Ejercicio 8.2 (Bootstrap censurado con riesgo proporcional de Cox). Reproducir el ejemplo en Canty

(2002, Rnews_2002-3) del modelo de riesgo proporcional de Cox (Cox, 1972):

```
# Datos
data(melanoma)
mel <- melanoma[melanoma$ulcer == 1, ]
mel$cens <- 1 * (mel$status == 1)
# Estimación supervivencia
library(survival)
# Modelo de riesgo proporcional de Cox
mel.cox <- coxph(Surv(time, cens) ~ thickness, data = mel)
mel.cox

## Call:
## coxph(formula = Surv(time, cens) ~ thickness, data = mel)
##
##          coef exp(coef) se(coef)   z      p
## thickness 0.09968  1.10481  0.04052 2.46 0.0139
##
## Likelihood ratio test=5  on 1 df, p=0.02541
## n= 90, number of events= 41
# summary(mel.cox)
# Estadísticos de interés
mel.cox$coefficients

## thickness
## 0.09967665
```

Capítulo 9

El Bootstrap con datos dependientes

En este capítulo se presentan gran cantidad de métodos bootstrap para realizar inferencia, así como predicción, en el contexto de datos dependientes. En primer lugar se hace una introducción a las condiciones habituales de dependencia y a los modelo paramétricos de dependencia, para luego centrarse en los métodos de remuestreo en ambos contextos.

En cada uno de los dos contextos (estimación y predicción) se estudiarán dos situaciones drásticamente diferentes. En la primera de ellas consideraremos modelos en los que la estructura de dependencia está explícitamente modelizada (normalmente a través de una ecuación de autorregresión), mientras que la segunda trata el caso en que no existe ninguna especificación explícita de la estructura de dependencia (simplemente se asumen condiciones mixing, por ejemplo). Una revisión sobre los resultados principales puede verse en Cao (1999).

9.1 Introducción a las condiciones de dependencia y modelos habituales de datos dependientes

9.1.1 Situaciones de dependencia general

Consideramos un proceso estocástico en tiempo discreto y con espacio de estados continuo (p. ej. \mathbb{R}), $\{X_t\}_{t \in \mathbb{Z}}$, del cual observamos parte de su trayectoria: (X_1, X_2, \dots, X_n) , es decir una muestra de datos dependientes. Este tipo de procesos estocásticos suelen llamarse series temporales.

Normalmente supondremos que el proceso $\{X_t\}_{t \in \mathbb{Z}}$ es estacionario. En ocasiones se requerirá además que sea fuertemente mixing:

$$\sup_{A \in \mathcal{F}_1^n, B \in \mathcal{F}_{n+k}^\infty} |P(A \cap B) - P(A)P(B)| \leq \alpha_k, \text{ con } \alpha_k \rightarrow 0,$$

o bien uniformemente mixing:

$$|P(A \cap B) - P(A)P(B)| \leq \phi_k P(A), \forall A \in \mathcal{F}_1^n, \forall B \in \mathcal{F}_{n+k}^\infty, \text{ con } \phi_k \rightarrow 0,$$

siendo \mathcal{F}_s^t la σ -álgebra generada por las variables aleatorias X_s, \dots, X_t .

Estas condiciones establecen que la dependencia entre las variables aleatorias que conforman las observaciones de la muestra se atenúa a medida que sus instantes temporales se distancian.

9.1.2 Modelos paramétricos de dependencia

Los modelos paramétricos más habituales para datos dependientes y estacionarios son los autorregresivos ($AR(p)$):

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + a_t, t \in \mathbb{Z},$$

de medias móviles ($MA(q)$):

$$X_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_p a_{t-q}, \quad t \in \mathbb{Z}$$

y la mezcla de ambos ($ARMA(p, q)$):

$$\begin{aligned} X_t = & \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} \\ & + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}, \end{aligned}$$

En las anteriores expresiones los $\{a_t\}_{t \in \mathbb{Z}}$ representan una sucesión de variables independientes con la misma distribución (ruido blanco), habitualmente con distribución normal.

9.2 El bootstrap en la estimación con datos dependientes

El objetivo de esta sección es mostrar distintos métodos de remuestreo para realizar inferencia sobre los parámetros de una serie temporal. Comenzaremos tratando los modelos de dependencia explícita para luego abordar la situación en que tan sólo existen condiciones generales de dependencia.

9.2.1 Modelos paramétricos de dependencia

Consideremos uno de los casos más simples, dado por el modelo $AR(p)$:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + a_t,$$

donde $\{a_t\}$ es una sucesión de variables aleatorias independientes de media cero (ruido blanco), de tal forma que a_t es independiente del pasado de X_t : $\{X_{t-1}, X_{t-2}, \dots\}$.

En el contexto de estimación error cuadrático medio de predicción (PMSE), Stine (1987) propone un método bootstrap que mejora el estimador clásico de sustitución del PMSE del mejor predictor lineal estimado (ver capítulo 2 de Fuller (1976)), cuando la distribución del ruido blanco no tiene porqué ser normal. El método procede como sigue:

1. Obtener una estimación de los coeficientes de autorregresión:

$$\hat{\phi}_1, \hat{\phi}_2, \dots, \hat{\phi}_p.$$

En el artículo de Stine estos estimadores se obtienen por el método de mínimos cuadrados.

2. Calcular los residuos (para aquellos índices que sea posible):

$$\hat{a}_t = X_t - \hat{\phi}_1 X_{t-1} - \hat{\phi}_2 X_{t-2} - \cdots - \hat{\phi}_p X_{t-p}, \quad t = p+1, p+2, \dots, n.$$

Estos valores son sustitutos de los errores inobservables a_t .

3. Calcular la distribución empírica de los residuos corregidos (recentrados y reescalados):

$$F_n^{\hat{a}}(x) = \frac{1}{n-p} \sum_{t=p+1}^n 1_{\{\hat{a}'_t \leq x\}},$$

donde $\hat{a}'_t = \hat{a}_t - \bar{a}$ y $\bar{a} = \frac{1}{n-p} \sum_{t=p+1}^n \hat{a}_t$.

4. Arrojar a_t^* , $t = 1, 2, \dots, n+k$ observaciones iid con distribución $F_n^{\hat{a}}$.

5. Fijar los primeros p valores de las réplicas bootstrap de la serie:

$$X_1^*, X_2^*, \dots, X_p^*$$

igual a cero (o con igual probabilidad de los $n-p+1$ bloques posibles de observaciones consecutivas de la serie original) y definir:

$$X_t^* = \hat{\phi}_1 X_{t-1}^* + \hat{\phi}_2 X_{t-2}^* + \cdots + \hat{\phi}_p X_{t-p}^* + a_t^*, \quad t = p+1, \dots, n+k.$$

6. A partir de la remuestra bootstrap (hasta el instante n), calcular las versiones bootstrap, $\hat{\phi}_1^*, \hat{\phi}_2^*, \dots, \hat{\phi}_p^*$, de los estimadores ϕ y obtener \widehat{X}_{n+k}^* , la predicción de X_{n+k}^* , usando la versión bootstrap de los estimadores de los parámetros y las últimas observaciones de la remuestra bootstrap.
7. Aproximar el PMSE mediante su análogo bootstrap:

$$PMSE^* = E^* \left[(\widehat{X}_{n+k}^* - X_{n+k}^*)^2 \right].$$

Ferretti y Romo (1996) demuestran la consistencia de un bootstrap basado en los residuos (en el sentido de convergencia débil de la distribución bootstrap) para contrastes de raíz unitaria en series temporales del tipo $AR(1)$, tanto en el caso de errores iid como cuando el error sigue también un modelo $AR(1)$. Heimann y Kreiss (1996) dan un resultado similar, sólo para el caso de errores iid, cuando el tamaño muestral de las remuestas bootstrap es m_n , de forma que $\frac{m_n}{n} \rightarrow 0$ (subremuestreo).

Las ideas generales sobre el bootstrap para modelos autorregresivos pueden extenderse al bootstrap de series temporales autorregresivas y de media móvil. Consideremos un modelo $ARMA(p, q)$:

$$\begin{aligned} X_t &= \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} \\ &\quad + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots - \theta_q a_{t-q}, \end{aligned}$$

o, equivalentemente,

$$\phi(B)X_t = \theta(B)a_t,$$

donde

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p \theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q$$

y B es el operador retardo: $BX_t = X_{t-1}$. La diferencia principal con respecto al caso autorregresivo es que ahora se necesitan estimar los coeficientes la parte de media móvil, al objeto de calcular los residuos, \hat{a}_t . Así, el algoritmo bootstrap para una serie $AR(p)$ puede adaptarse a este caso de manera inmediata. En este contexto, Kreiss y Franke (1992) usan la representación $MA(\infty)$ del proceso de error en términos de la series original,

$$a_t = \theta(B)^{-1}\phi(B)X_t,$$

para construir los residuos (utilizando los parámetros estimados en la fórmula anterior) y demuestran la validez asintótica del bootstrap (en el sentido de la distancia de Mallows) para aproximar la distribución en el muestreo del M -estimador de los parámetros de un modelo $ARMA(p, q)$.

Paparoditis (1996) demuestra la validez del bootstrap para la inferencia acerca de los parámetros de un proceso $ARMA$ multidimensional de orden infinito. El autor propone arrojar réplicas bootstrap de modelos $ARMA$ de orden finito creciente, de forma que ese orden tienda a infinito a cierta tasa, según crece el tamaño muestral. Bühlmann (1997) desarrolla ideas semejantes en el contexto de procesos $AR(\infty)$, introduciendo el llamado *sieve bootstrap*. Este método se ha extendido estimación no paramétrica de la regresión cuando la variable explicativa sigue un modelo $AR(\infty)$ (ver Bühlmann (1998)).

9.2.2 Situaciones de dependencia general

En esta sección se estudia el caso en que no se asume ningún tipo de estructura autorregresiva sobre el proceso estocástico. De hecho, asumiremos condiciones generales de dependencia, como condiciones mixing o de m -dependencia, por ejemplo.

El problema de no tener una ecuación explícita que relacione el valor actual de la serie con sus valores pasados provoca que no sea posible diseñar un plan de remuestreo a partir de un modelo de dependencia explícita.

9.2.3 El bootstrap por bloques

La primeras propuestas para evitar el problema de carecer de una expresión explícita para modelizar la dependencia corresponden a Künsch (1989) y Liu y Singh (1992), que propusieron de forma independiente el llamado bootstrap por bloques (moving blocks bootstrap o MBB). El método procede del siguiente modo:

1. Fijar un entero positivo, b , el tamaño del bloque, y tomar k igual al menor entero mayor o igual que $\frac{n}{b}$.
2. Definir los bloques (o submuestras): $B_{i,b} = (X_i, X_{i+1}, \dots, X_{i+b-1})$, o simplemente B_i , para $i = 1, 2, \dots, q$ ($q = n - b + 1$).
3. Arrojar k observaciones (bloques), $\xi_1, \xi_2, \dots, \xi_k$, con distribución equiprobable sobre el conjunto de posibles bloques: $\{B_1, B_2, \dots, B_q\}$. Cada ξ_i es un vector b -dimensional $(\xi_{i,1}, \xi_{i,2}, \dots, \xi_{i,b})$.
4. Definir \mathbf{X}^* como el vector formado por las n primeras componentes de

$$(\xi_{1,1}, \xi_{1,2}, \dots, \xi_{1,b}, \xi_{2,1}, \xi_{2,2}, \dots, \xi_{2,b}, \dots, \xi_{k,1}, \xi_{k,2}, \dots, \xi_{k,b}).$$

Si tomamos $b = 1$, entonces $k = n$ y se obtiene el bootstrap ordinario. Por otra parte, si $b = n$, tenemos $k = 1$ y se obtiene el remuestreo degenerado, ya que todas las réplicas bootstrap coincidirían con la muestra original.

Künsch (1989) y Liu y Singh (1992) demuestran la validez asintótica de este método bajo condiciones poco restrictivas sobre el grado de dependencia y el tamaño del bloque. Por ejemplo, Liu y Singh (1992) demuestran que si el proceso estocástico es m -dependiente (i.e., (X_t, X_{t+1}, \dots) y (X_s, X_{s-1}, \dots)) son independientes siempre que $s + m < t$, si T es un funcional dos veces Frechet diferenciable y el tamaño del bloque satisface $b \rightarrow \infty$ y $b \log n/n \rightarrow 0$, entonces

$$\sup_{x \in \mathbb{R}} |P^* \{ \sqrt{n} (T(F_n^*) - T(F_n)) \leq x \} - P \{ \sqrt{n} (T(F_n) - T(F)) \leq x \}| \rightarrow 0,$$

en probabilidad. Naik-Nimbalkar y Rajarshi (1994) demuestran la consistencia del proceso empírico MBB bajo la condición de que $b = O(n^{1/2-\varepsilon})$, para algún $\varepsilon \in (\frac{1}{4}, \frac{1}{2})$. Bühlmann (1994) lo extiende al caso multivariante y debilita la condición sobre ε , siendo $\varepsilon \in (0, \frac{1}{2})$.

Carlstein, Do, Hall, Hesterberg y Künsch (1998) propusieron una modificación del MBB. Su idea consiste en seleccionar las remuestras de bloques de acuerdo a una cadena de Markov. El primer bloque de la remuestra bootstrap se genera igual que para el MBB ordinario. Una vez que se ha seleccionado en la remuestra el bloque B_i , el siguiente bloque de la remuestra bootstrap se elige dentro de todos los posibles bloques, B_j , poniendo más probabilidad a aquellos que son precedidos por un bloque, B_{j-1} , cuyo último valor, X_{j+b-2} , es más cercano al último valor, X_{i+b-1} , del bloque B_i . En el caso $j = 1$, esta regla no tiene sentido, ya que no existe un bloque anterior al B_1 , así que, en ese caso los autores proponen hacer que la probabilidad dependa de la distancia entre X_1 (el primer valor del bloque B_1) y el valor siguiente al último del bloque B_i , es decir X_{i+b} . De nuevo esto sólo es posible si $i < q$. Si $i = q$ usan X_1 en lugar de X_{i+b} . Estas probabilidades se calculan usando pesos de tipo núcleo. Estos autores demuestran la consistencia de esta versión del MBB para el estimador bootstrap de la varianza de la media muestral.

9.2.4 Elección de b

Un asunto importante en el método bootstrap por bloques es la elección del tamaño del bloque, b . Hall, Horowitz y Jing (1995) considera este problema en el contexto de la estimación bootstrap del sesgo y la varianza. Obtienen una expresión asintótica para el error cuadrático medio:

$$n^{-2}(C_1 b^{-2} + C_2 n^{-1} b),$$

donde C_1 y C_2 son constantes desconocidas que dependen del problema de estimación del que se trate. Está claro entonces que el tamaño óptimo del bloque (en el sentido del error cuadrático medio) es de orden $n^{1/3}$.

Un resultado importante de utilidad para probar la validez del MBB en muchos contextos es el dado por Radulović (1996). Este autor demuestra que siempre que una sucesión de variables aleatorias fuertemente mixing satisface el Teorema Central del Límite, dicho resultado también es válido para la versión bootstrap por bloques.

9.2.5 El bootstrap estacionario

Consideremos el bootstrap por bloques para una muestra, (X_1, X_2, \dots, X_n) , de tamaño $n = 100$ y el tamaño del bloque $b = 10$. Podemos calcular fácilmente las distribuciones bootstrap conjuntas de (X_{10}^*, X_{11}^*) y (X_9^*, X_{10}^*) :

$$\begin{aligned} P^* \{(X_{10}^*, X_{11}^*) = (X_i, X_j)\} &= \frac{1}{91^2}, \quad \text{para } i = 10, 11, \dots, 100; \\ &\quad j = 1, 2, \dots, 91 \\ P^* \{(X_9^*, X_{10}^*) = (X_i, X_j)\} &= \frac{1}{91}, \quad \text{para } i = 9, 10, \dots, 99; j = i + 1. \end{aligned}$$

Como estas distribuciones bootstrap son diferentes entonces el MBB no es estacionario.

Con el fin de remediar la falta de estacionariedad del MBB, Politis y Romano (1994a) proponen el llamado bootstrap estacionario (stationary bootstrap, SB). El método necesita de la elección de un número $p \in [0, 1]$ y puede presentarse de dos formas equivalentes:

SB1:

1. Arrojar X_1^* de F_n , la distribución empírica construida con las muestra (X_1, X_2, \dots, X_n) .
2. Una vez que se ha arrojado el valor $X_i^* = X_j$ (para algún $j \in \{1, 2, \dots, n-1\}$) con $i < n$, se define la siguiente observación bootstrap, X_{i+1}^* , como X_{j+1} , con probabilidad $1-p$ y arrojada de la función de distribución empírica de la muestra, con probabilidad p . En el caso $j = n$, la observación X_{j+1} se reemplaza por X_1 .

SB2:

1. Definir los bloques circulares $B_{i,b} = (X_i, X_{i+1}, \dots, X_{i+b-1})$ con $b \in \mathbb{N}$, $i = 1, 2, \dots, n$ y $X_t = X_{((t-1)\bmod n)+1}$ si $t > n$.
2. Arrojar realizaciones iid, L_1, L_2, \dots , con distribución geométrica de parámetro p , i.e.

$$P(L_1 = m) = p(1-p)^{m-1}, m = 1, 2, \dots$$

3. Obtener enteros aleatorios, I_1, I_2, \dots , con distribución equiprobable sobre el conjunto $\{1, 2, \dots, n\}$.
4. Definir $X_1^*, X_2^*, \dots, X_n^*$ como los n primeros valores obtenidos al unir los bloques $B_{I_1, L_1}, B_{I_2, L_2}, \dots$

A continuación se comentan algunos aspectos interesantes en relación con el SB.

- El número mínimo de bloques necesario, k , en el método de remuestreo SB2, de forma que el conjunto de bloques $B_{I_1, L_1}, B_{I_2, L_2}, \dots, B_{I_k, L_k}$ tenga, al menos, n observaciones, coincide con el menor entero k para el cual $\sum_{i=1}^k L_i \geq n$.
- Eligiendo $p = 1$ se tiene el bootstrap clásico. La elección $p = 0$ corresponde a una permutación circular aleatoria de la muestra, que conduce a una distribución bootstrap degenerada si el estadístico es funcional (i.e., si es sólo función de la distribución empírica, pero no depende del orden de los datos).
- Condicionalmente a la muestra observada, el proceso bootstrap, $\{X_i^*\}$, es estacionario. Más aún, si no hay datos empatados, entonces el proceso bootstrap es un proceso de Markov. En general, es un proceso markoviano de orden $r + 1$, donde

$$r = \max \{b \in \mathbb{N} / \exists i, j, i \neq j \text{ con } B_{i,b} = B_{j,b}\}.$$

- Observando el esquema de remuestreo SB2 resulta fácil generalizar el método a casos en los que la distribución de L_i no es geométrica y la distribución de los I_i no tiene porqué ser equiprobable. En tales casos, debe ponerse mucho cuidado en la elección de esas distribuciones para no destruir la estacionariedad del proceso bootstrap. Con esta generalización del remuestreo SB2, el MBB puede pensarse como un caso particular, tomando

$$P(L_i = m) = \begin{cases} 1 & \text{si } m = b \\ 0 & \text{si } m \neq b \end{cases}$$

$$P(I_i = j) = \begin{cases} 1/q & \text{si } j = 1, 2, \dots, q \\ 0 & \text{si } j = q + 1, q + 2, \dots, n \end{cases}$$

con $q = n - b + 1$.

- Como el tamaño medio del bloque en el SB es $\frac{1}{p}$, en cierto sentido el valor p juega el papel inverso del tamaño del bloque en el MBB ($p = 1$ es comparable con $b = 1$ y $p = 0$ con $b \rightarrow \infty$).

Dado un proceso estocástico estrictamente estacionario con función de autocovarianza γ , cumpliendo $\gamma(0) + \sum_r |r\gamma(r)| < \infty$, con momento finito de orden $d + 2$ (para algún $d > 0$) y la siguiente condición para los coeficientes mixing:

$$\sum_k \alpha_k^{\frac{d}{d+2}} < \infty,$$

Politis y Romano (1994a) demostraron la validez asintótica del bootstrap estacionario:

$$\sup_{x \in \mathbb{R}} |P^* \{ \sqrt{n}(\bar{X}_n^* - \bar{X}_n) \leq x \} - P \{ \sqrt{n}(\bar{X}_n - \mu) \leq x \}| \rightarrow 0,$$

en probabilidad, siempre que $p \rightarrow 0$ y $np \rightarrow \infty$. Estos autores también dan una idea acerca de cómo generalizar este resultado a estadísticos funcionales, $T(F_n)$, donde T es un funcional Frechet diferenciable. Politis y Romano (1994c) también demostraron que el método funciona para una amplia clase de estimadores, incluyendo los de mínima distancia.

9.2.6 El método del submuestreo

Politis y Romano (1994b) proporcionan un método bootstrap que es válido bajo condiciones minimales. Estos autores presentan dos versiones de este método: una para datos independientes y otra para datos dependientes.

Para enunciar el método del submuestreo de forma unificada consideremos las observaciones, X_1, X_2, \dots, X_n , que provienen o bien de (a) variables aleatorias iid con distribución F o (b) un proceso estocástico fuertemente mixing. Consideremos un parámetro $\theta = \theta(F)$, $T_n = T_n(X_1, X_2, \dots, X_n)$ un estimador de él, y $J_n(\cdot, F)$ la función de distribución en el muestreo de $\tau_n(T_n - \theta)$. Se fija un entero $b < n$ y se define:

- en el caso iid, $S_{n,i} = T_b(Y_i)$, $i = 1, 2, \dots, N$, donde Y_1, Y_2, \dots, Y_n son todas las $N = \binom{n}{b}$ posibles submuestras de tamaño b (sin reemplazamiento) de la muestra original.
- en el caso de datos dependientes, $S_{n,i} = T_b(B_{i,b})$, $i = 1, 2, \dots, N$, donde $B_{i,b}$, $i = 1, 2, \dots, N$, con $N = n - b + 1$, son todos los posibles bloques de tamaño b .

Este método propone usar la función de distribución empírica de los valores $\tau_b(S_{n,i} - T_n)$,

$$L_n(x) = \frac{1}{N} \sum_{i=1}^N \mathbf{1}_{\{\tau_b(S_{n,i} - T_n) \leq x\}}$$

como aproximación de la distribución en el muestreo de $\tau_n(T_n - \theta)$. El resultado demostrado por Politis y Romano (1994b) afirma que siempre que $\tau_b/\tau_n \rightarrow 0$, $b \rightarrow \infty$ y $b/n \rightarrow 0$, la condición $\tau_n(T_n - \theta) \xrightarrow{d} J(\cdot, F)$ implica que $L_n(x) \rightarrow J(x, F)$ para cada x , punto de continuidad de $J(\cdot, F)$ y $\|L_n(\cdot) - J_n(\cdot, F)\|_\infty \rightarrow 0$ en probabilidad (si $J(\cdot, F)$ es continua). A grandes rasgos este resultado garantiza que, bajo condiciones minimales sobre el tamaño del bloque, el método del submuestreo es siempre asintóticamente válido, siempre que el estadístico de interés tenga una distribución límite.

9.3 El bootstrap para la predicción con datos dependientes

Dado un proceso estocástico en tiempo discreto, $\{X_t\}_{t \in \mathbb{Z}}$, un problema importante en este contexto es predecir un valor futuro del proceso. Habiendo observado una trayectoria del proceso, hasta el tiempo n : X_1, X_2, \dots, X_n , la cuestión es encontrar un predictor, tan preciso como sea posible, para el valor del proceso a k retardos, X_{n+k} . Puede construirse un predictor puntual o un intervalo de predicción, que es típicamente más informativo.

9.3.1 Modelos de dependencia paramétrica

Al igual que en el caso de estimación, cuando la estructura de dependencia sigue un modelo paramétrico, esta información puede usarse para adaptar el bootstrap ordinario al contexto de predicción. La mayor parte de los mecanismos bootstrap presentados en la sección anterior para la estimación en el contexto paramétrico son también válidos para la predicción con muy pocos cambios.

Stine (1987) propone un método bootstrap (ya presentado antes) para estimar el error cuadrático medio de predicción del mejor predictor lineal estimado en el contexto de un modelo $AR(p)$. Usa versiones bootstrap de los parámetros estimados y la remuestra bootstrap para obtener

$$\widehat{X}_{n+j}^* = \widehat{\phi}_1^* \widehat{X}_{n+j-1}^* + \widehat{\phi}_2^* \widehat{X}_{n+j-2}^* + \cdots + \widehat{\phi}_p^* \widehat{X}_{n+j-p}^*, \quad j = 1, 2, \dots, k,$$

con $\widehat{X}_t^* = X_t^*$ para $t \leq n$, cuya distribución bootstrap se usa para estimar la verdadera distribución en el muestreo del predictor.

Thombs y Schucany (1990) proponen método bootstrap primero hacia atrás y luego hacia adelante para obtener intervalos de predicción a k retardos para procesos $AR(p)$. El método procede como sigue:

1. Construir los residuos hacia atrás:

$$\widehat{e}_i = X_i - \widehat{\phi}_1 X_{i+1} - \widehat{\phi}_2 X_{i+2} - \cdots - \widehat{\phi}_p X_{i+p}, \quad i = 1, 2, \dots, n-p,$$

y calcular su versión corregida, \widehat{e}'_i (tal y como se hace en el método de Stine en la sección anterior).

2. Arrojar errores bootstrap hacia atrás, \widehat{e}_i^* , de la función de distribución empírica de los residuos hacia atrás corregidos.
3. Definir réplicas bootstrap hacia atrás:

$$X_i^* = \widehat{\phi}_1 X_{i+1}^* + \widehat{\phi}_2 X_{i+2}^* + \cdots + \widehat{\phi}_p X_{i+p}^* + \widehat{e}_i^*, \quad i = n-p, \dots, 1,$$

con $X_i^* = X_i$ para $t = n-p+1, n-p+2, \dots, n$.

4. Calcular versiones bootstrap de los estimadores, $\widehat{\phi}_1^*, \widehat{\phi}_2^*, \dots, \widehat{\phi}_p^*$.

5. Construir residuos hacia adelante:

$$\widehat{a}_i = X_i - \widehat{\phi}_1 X_{i-1} - \widehat{\phi}_2 X_{i-2} - \cdots - \widehat{\phi}_p X_{i-p}, \quad i = p+1, p+2, \dots, n,$$

y su versión corregida \widehat{a}'_i .

6. Arrojar errores bootstrap hacia adelante, \widehat{a}_i^* , de la función de distribución empírica de los residuos hacia adelante corregidos.
7. Definir las réplicas bootstrap hacia adelante:

$$X_{n+j}^* = \widehat{\phi}_1^* X_{n+j-1}^* + \widehat{\phi}_2^* X_{n+j-2}^* + \cdots + \widehat{\phi}_p^* X_{n+j-p}^* + \widehat{a}_{n+j}^*, \quad j = 1, 2, \dots, k.$$

Thombs y Schucany (1990) prueban la validez asintótica del bootstrap demostrando que, cuando el tamaño muestral, n , tiende a infinito,

$$P^*(X_{n+k}^* \leq x) - P(X_{n+k} \leq x | X_{n-p+1}, X_{n-p+2}, \dots, X_n) \rightarrow 0,$$

de forma casi segura, para casi todo x . Este resultado implica la validez asintótica del intervalo de predicción bootstrap $(x_{\alpha/2}^*, x_{1-\alpha/2}^*)$, donde x_β^* se define mediante $P^*(X_{n+k}^* \leq x_\beta^*) = \beta$. Algunos estudios de simulación muestran los beneficios de este método sobre los métodos clásicos cuando la distribución del error no es normal.

García-Jurado, González-Manteiga, Prada-Sánchez, Febrero-Bande y Cao (1995) demuestran la validez del bootstrap de Thombs y Schucany para modelos $ARI(p, d)$. Supongamos que $X_t \sim ARI(p, d)$, la idea principal de esta extensión es la siguiente:

1. Construir la serie de diferencias, $Y_t = \nabla^d X_t$, donde

∇ es el operador diferencia definido por $\nabla X_t = X_t - X_{t-1}$. Obviamente Y_t tiene una estructura $AR(p)$.

2. Aplicar el bootstrap de Thombs y Schucany a esta serie para obtener la serie bootstrap $\{Y_t^*\}$.
3. Calcular réplicas bootstrap X_t^* mediante d integraciones de la serie Y_t^* , fijando las primeras observaciones bootstrap: $X_t^* = X_t$ para $t \leq n$.

Cao, Febrero-Bande, González-Manteiga, Prada-Sánchez y García-Jurado (1997) estudian un método bootstrap, alternativo al de Thombs y Schucany, que es computacionalmente más rápido y también consistente. Puede resumirse en los siguientes pasos:

1. Construir la distribución empírica de los residuos hacia adelante corregidos, $F_n^{\hat{a}'}$.
2. Generar \hat{a}_i^* con distribución $F_n^{\hat{a}'}$.
3. Construir réplicas bootstrap futuras

$$X_{n+j}^* = \hat{\phi}_1 X_{n+j-1}^* + \hat{\phi}_2 X_{n+j-2}^* + \cdots + \hat{\phi}_p X_{n+j-p}^* + \hat{a}_{n+j}^*, \quad j = 1, 2, \dots, k,$$

donde $X_i^* = X_i$ para $i = n, n-1, \dots, n-p+1$.

Estos autores demuestran la validez asintótica de este método bootstrap (en el mismo sentido que Thombs y Schucany) y de una versión suavizada en la cual se reemplaza $F_n^{\hat{a}'}$ por $K_h * F_n^{\hat{a}'}$, en el paso 2. Pascual, Romo y Ruiz (2001) proponen una variante de este método en la que se incorpora la variabilidad en la estimación de los parámetros de la serie.

9.3.2 Situaciones de dependencia general

Cuando la estructura de dependencia de la serie no es explícita los métodos bootstrap existentes para la estimación (como el MBB, el SB o el método de submuestreo) no funcionan para la predicción. El motivo es que estos métodos no estiman consistentemente la distribución condicional

$$X_{n+k}|_{X_1, X_2, \dots, X_n}.$$

Esta situación es completamente diferente del caso en que la dependencia se modeliza paramétricamente, ya que en ese otro caso los métodos bootstrap usados para la estimación permanecen válidos, en general, en el contexto de predicción.

Es poco menos que imposible estimar la distribución condicional anterior sin hacer ninguna suposición sobre el tipo de dependencia. Sin embargo se puede llevar a cabo una estimación cuando se supone que el proceso estocástico es markoviano de orden p , porque entonces,

$$X_{n+k}|_{X_1, X_2, \dots, X_n} =^d X_{n+k}|_{X_{n-p+1}, X_{n-p+2}, \dots, X_n}$$

y, por tanto,

$$F_k(y|_x) = F_k(y|_{x_1, x_2, \dots, x_p}) = P(X_{n+k} \leq y|_{X_{n-p+1}=x_1, X_{n-p+2}=x_2, \dots, X_n=x_p})$$

puede estimarse por medio de un estimador no paramétrico de la distribución condicional, basado en estimadores no paramétricos de la regresión, como, por ejemplo, mediante el estimador tipo núcleo:

$$\widehat{F}_{k,H}(y|_{\mathbf{x}}) = \frac{\sum_{i=1}^{q-k} K_H(\mathbf{x} - B_{i,p}) \cdot 1_{\{X_{i+p+k-1} \leq y\}}}{\sum_{i=1}^{q-k} K_H(\mathbf{x} - B_{i,p})},$$

donde $q = n - p + 1$, $K_H(\mathbf{u}) = \det(H)^{-1}K(H^{-1}\mathbf{z})$, K es una función núcleo, H es una matriz ventana diagonal definida positiva y $B_{i,p}$, $i = 1, 2, \dots, q$ son los bloques muestrales de tamaño p . Este estimador podría usarse para calcular intervalos predicción aproximados para X_{n+k} dados los valores observados del proceso hasta el instante n .

En el caso $p = 1$ ($\{X_t\}$ es un proceso de Markov) el estimador núcleo puede escribirse como

$$\widehat{F}_{k,h}(y|_{\mathbf{x}}) = \frac{\sum_{i=1}^{n-k} K_h(x - X_i) \cdot 1_{\{X_{i+k} \leq y\}}}{\sum_{i=1}^{n-k} K_h(x - X_i)},$$

donde $K_h(u) = h^{-1}K(u/h)$ y $h > 0$. Usar este estimador para calcular el intervalo de predicción de nivel α :

$$(\widehat{F}_{k,h}^{-1}(\alpha/2|_{\mathbf{x}}), \widehat{F}_{k,h}^{-1}(1 - \alpha/2|_{\mathbf{x}})),$$

es equivalente a llevar a cabo un método bootstrap de forma que

$$P(X_{n+k}^* = X_{i+k}) = \hat{p}_i = \frac{K_h(X_n - X_i)}{\sum_{j=1}^{n-k} K_h(X_n - X_j)}, i = 1, 2, \dots, n - k.$$

Teniendo esto en cuenta ese mecanismo bootstrap puede describirse como sigue:

1. Construir los bloques muestrales de tamaño $k + 1$: $B_{i,k+1}$, $i = 1, 2, \dots, n - k$.
2. Calcular los valores \hat{p}_i , $i = 1, 2, \dots, n - k$.
3. Arrojar un bloque del conjunto $\{B_{1,k+1}, B_{2,k+1}, \dots, B_{n-k,k+1}\}$ con probabilidades \hat{p}_i , $i = 1, 2, \dots, n - k$ y definir X_{n+k}^* como la última observación de los bloques generados.

Está claro que la precisión de este mecanismo bootstrap depende de las propiedades del estimador tipo núcleo de la distribución condicional. Así, por ejemplo, bajo las condiciones impuestas en el Teorema 1 de Gannoun (1990) se obtiene que

$$\sup_{x \in C} \sup_{y \in \mathbb{R}} |P(X_{n+k}^* \leq y|_{X_n=x}) - P(X_{n+k} \leq y|_{X_n=x})| \rightarrow 0$$

en probabilidad.

Como consecuencia los intervalos de predicción bootstrap tienen probabilidad de cobertura asintóticamente correcta, uniformemente, en probabilidad, sobre la última observación de la muestra. Este resultado puede extenderse fácilmente para procesos de Markov de orden $p > 1$.

9.4 Implementación en R

Para simular una serie de tiempo en R se puede emplear la función `arima.sim()` del paquete base `stats`. Por ejemplo, podemos generar una serie autoregresiva con: [Figura 9.1]

```
# Parámetros
nsim <- 200    # Número de simulaciones
xvar <- 1       # Varianza
xmed <- 0       # Media
rho <- 0.5      # Coeficiente AR
nburn <- 10     # Período de calentamiento (burn-in)
evar <- xvar*(1 - rho^2) # Varianza del error
# Simulación
set.seed(1)
```

```
ry <- arima.sim(list(order = c(1,0,0), ar = rho),
                 n = nsim, sd = sqrt(evar)) # n.start = nburn
plot(ry)
```

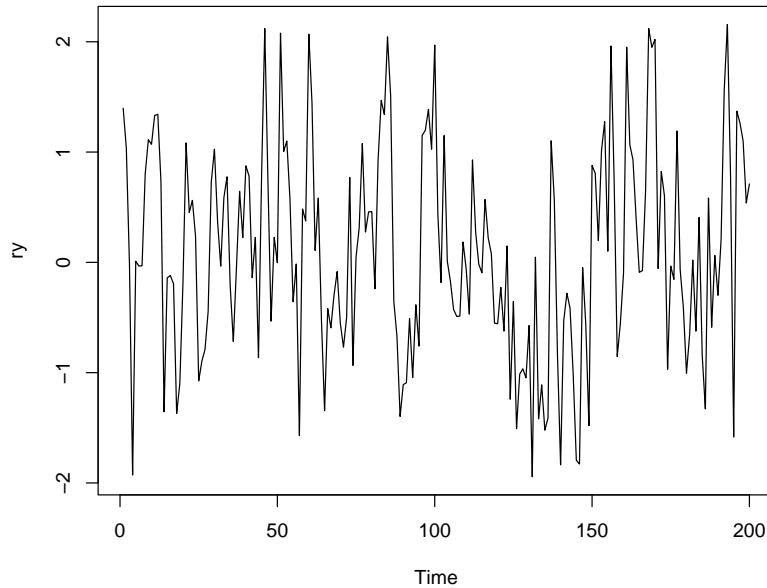


Figura 9.1: Simulación de un modelo autoregresivo.

En este caso el periodo de calentamiento se establece mediante el parámetro `n.start` (que se fija automáticamente a un valor adecuado). La recomendación es fijar la varianza de las series simuladas si se quieren comparar resultados considerando distintos parámetros de dependencia.

Otras opciones:

- `rand.gen = rnorm`
- `innov = rand.gen(n, ...)`
- `n.start = NA`
- `start.innov = rand.gen(n.start, ...)`

Ejemplo (`?arima.sim`): [Figura 9.2]

```
ry2 <- arima.sim(n = 63, list(ar = c(0.8897, -0.4858),
                           ma = c(-0.2279, 0.2488)),
                  rand.gen = function(n, ...) sqrt(0.1796) * rt(n, df = 5))

plot(ry2)
```

9.5 Implementación en R con el paquete `boot`

La función `tsboot()` del paquete `boot` implementa distintos métodos de remuestreo para series de tiempo.

```
library(boot)
# ?tsboot
```

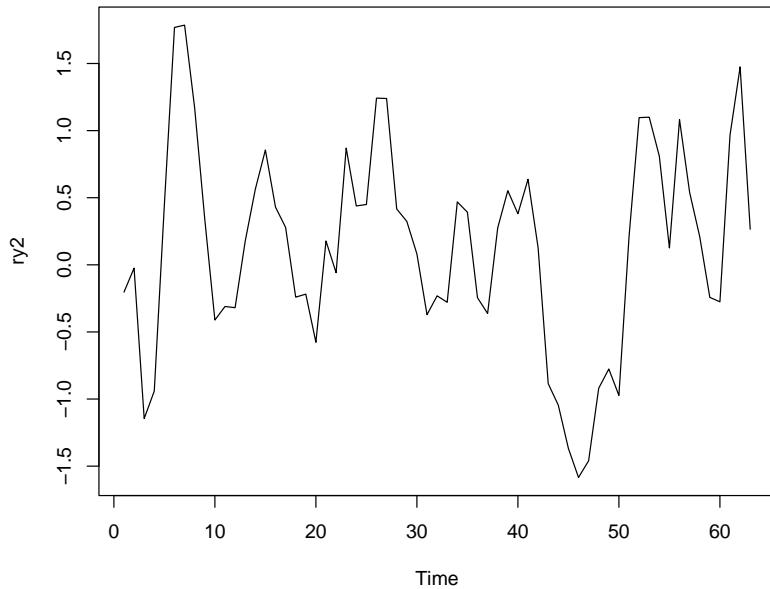


Figura 9.2: Simulación de un modelo autoregresivo con errores con distribución t^* de Student.

9.5.1 Rnews 1

Canty, A. J. (2002). Resampling methods in R: the boot package. Rnews: The Newsletter of the R Project, 2 (3), pp. 2-7.

"`tsboot()` can do either of these methods by specifying `sim="fixed"` or `sim="geom"` respectively. A simple call to `tsboot` includes the time series, a function for the `statistic` (the first argument of this function being the time series itself), the number of bootstrap replicates, the simulation type and the (mean) block length".

```
# Datos
data(lynx)
# ?lynx
# Boot
library(boot)
lynx.fun <- function(tsb) {
  fit <- ar(tsb, order.max = 25)
  c(fit$order, mean(tsb))
}
# tsboot
set.seed(1)
tsboot(log(lynx), lynx.fun, R = 199, sim = "geom", l = 20)

##
## STATIONARY BOOTSTRAP FOR TIME SERIES
##
## Average Block Length of 20
##
## Call:
## tsboot(tseries = log(lynx), statistic = lynx.fun, R = 199, l = 20,
##       sim = "geom")
##
```

```
##
## Bootstrap Statistics :
##      original     bias   std. error
## t1* 11.000000 -6.46733668  2.4675036
## t2*  6.685933 -0.01494926  0.1163515
```

9.5.2 Rnews 2

Canty, A. J. (2002). Resampling methods in R: the boot package. Rnews: The Newsletter of the R Project, 2 (3), pp. 2-7.

"An alternative to the block bootstrap is to use model based resampling. In this case a model is fitted to the time series so that the errors are i.i.d. The observed residuals are sampled as an i.i.d. series and then a bootstrap time series is reconstructed. In constructing the bootstrap time series from the residuals, it is recommended to generate a long time series and then discard the initial burn-in stage. Since the length of burn-in required is problem specific, `tsboot` does not actually do the resampling. Instead the user should give a function which will return the bootstrap time series. This function should take three arguments, the time series as supplied to `tsboot`, a value `n.sim` which is the length of the time series required and the third argument containing any other information needed by the random generation function such as coefficient estimates. When the random generation function is called it will be passed the arguments `data`, `n.sim` and `ran.args` passed to `tsboot` or their defaults.

One problem with the model-based bootstrap is that it is critically dependent on the correct model being fitted to the data. Davison and Hinkley (1997) suggest post-blackening as a compromise between the block bootstrap and the model-based bootstrap. In this method a simple model is fitted and the residuals are found. These residuals are passed as the dataset to `tsboot` and are resampled using the block (or stationary) bootstrap. To create the bootstrap time series the resampled residuals should be put back through the fitted model filter. The function `ran.gen` can be used to do this'.

```
# Datos
lynx1 <- log(lynx)
# Modelo
lynx.ar <- ar(lynx1)
# Residuos
lynx.res <- with(lynx.ar, resid[!is.na(resid)])
lynx.res <- lynx.res - mean(lynx.res)
# Boot
library(boot)
lynx.ord <- c(lynx.ar$order, 0, 0)
lynx.mod <- list(order = lynx.ord, ar = lynx.ar$ar)
lynx.args <- list(mean = mean(lynx1), model = lynx.mod)
lynx.black <- function(res, n.sim, ran.args) {
  m <- ran.args$mean
  ts.mod <- ran.args$model
  m + filter(res, ts.mod$ar, method = "recursive")
}
# tsboot
set.seed(1)
tsboot(lynx.res, lynx.fun, R = 199, l = 20,
       sim = "fixed", n.sim = 114,
       ran.gen = lynx.black, ran.args = lynx.args)

##
```

POST-BLACKENED BLOCK BOOTSTRAP FOR TIME SERIES

##

```

## Fixed Block Length of 20
##
## Call:
## tsboot(tseries = lynx.res, statistic = lynx.fun, R = 199, l = 20,
##        sim = "fixed", n.sim = 114, ran.gen = lynx.black, ran.args = lynx.args)
##
## Bootstrap Statistics :
##      original   bias   std. error
## t1* 0.0000e+00 9.819095 3.46664295
## t2* 6.1989e-18 6.683323 0.09551445

```

9.6 Ejercicios

Ejercicio 9.1 (Practical 8.1, Lynx data: Davison y Hinkley, 1997). Reproducir el “Practical 8.1 (Lynx data)” en Davison, A. C., y Hinkley, D. V. (1997). Bootstrap methods and their application. Cambridge university press, <http://statwww.epfl.ch/davison/BMA> (caché):

”Dataframe lynx contains the Canadian lynx data, to the logarithm of which we fit the autoregressive model that minimizes AIC:

[Figura 9.3]

```
ts.plot(log(lynx))
```

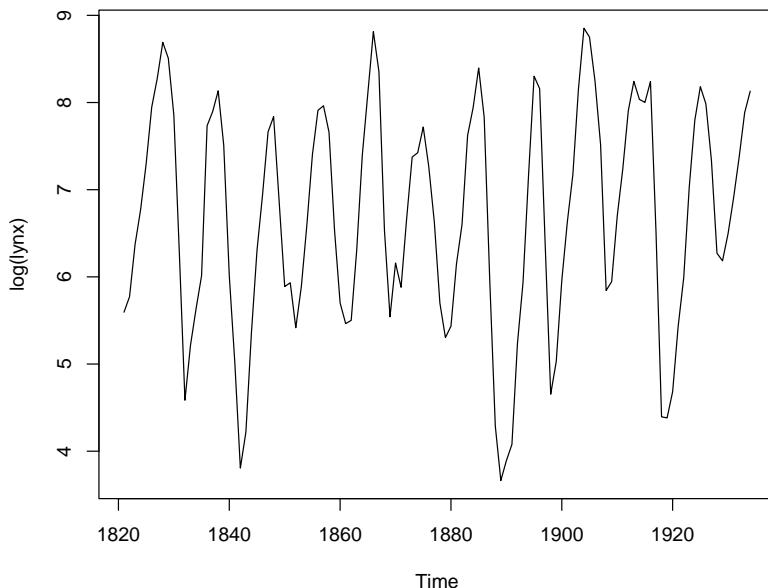


Figura 9.3: Lynx data (logarithmic scale).

```

lynx.ar <- ar(log(lynx))
lynx.ar$order

```

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

The best model is AR(11). How well determined is this, and what is the variance of the series average? We bootstrap to see, using `lynx.fun` (given below), which calculates the order of the fitted autoregressive model, the series average, and saves the series itself.

Here are results for fixed-block bootstraps with block length $l = 20$:

```
set.seed(DNI)
lynx.fun <- function(tsb) {
  ar.fit <- ar(tsb, order.max=25)
  c(ar.fit$order, mean(tsb), tsb)
}
lynx.1 <- tsboot(log(lynx), lynx.fun, R=99, l=20, sim="fixed")
lynx.1
ts.plot(ts(lynx.1$t[1,3:116], start=c(1821,1)),
         main="Block simulation, l=20")
boot.array(lynx.1)[1,]
table(lynx.1$t[,1])
var(lynx.1$t[,2])
qnorm(lynx.1$t[,2]);
abline(mean(lynx.1$t[,2]),sqrt(var(lynx.1$t[,2])),lty=2)
```

To obtain similar results for the stationary bootstrap with mean block length $l = 20$:

```
.Random.seed <- lynx.1$seed
lynx.2 <- tsboot(log(lynx), lynx.fun, ...
# lynx.2
```

See if the results look different from those above. Do the simulated series using blocks look like the original? Compare the estimated variances under the two resampling schemes. Try different block lengths, and see how the variances of the series average change.

For model-based resampling we need to store results from the original model:

Check the orders of the fitted models for this scheme.

For post-blackening we need to define yet another function:

Compare these results with those above, and try the post-blackened bootstrap with `sim="geom"`. (Sections 8.2.2, 8.2.3)'.

Ejercicio 9.2 (Practical 8.2, Beaver data: Davison y Hinkley, 1997). Reproducir el “Practical 8.2 (Beaver data)” en Davison, A. C., y Hinkley, D. V. (1997). Bootstrap methods and their application. Cambridge university press, <http://statwww.epfl.ch/davison/BMA> (caché):

”The data in beaver consist of a time series of $n = 100$ observations on the body temperature y_1, \dots, y_n and an indicator x_1, \dots, x_n of activity of a female beaver, Castor canadensis.

[Figura 9.4]

```
# ?beaver
plot(beaver)

class(beaver)

## [1] "mts" "ts"

str(beaver)

## Time-Series [1:100, 1:4] from 1 to 100: 307 307 307 307 307 ...
## - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "day" "time" "temp" "activ"
```

We want to estimate and give an uncertainty measure for the body temperature of the beaver. The simplest model that allows for the clear autocorrelation of the series is ...

To fit the original model and to generate a new series:

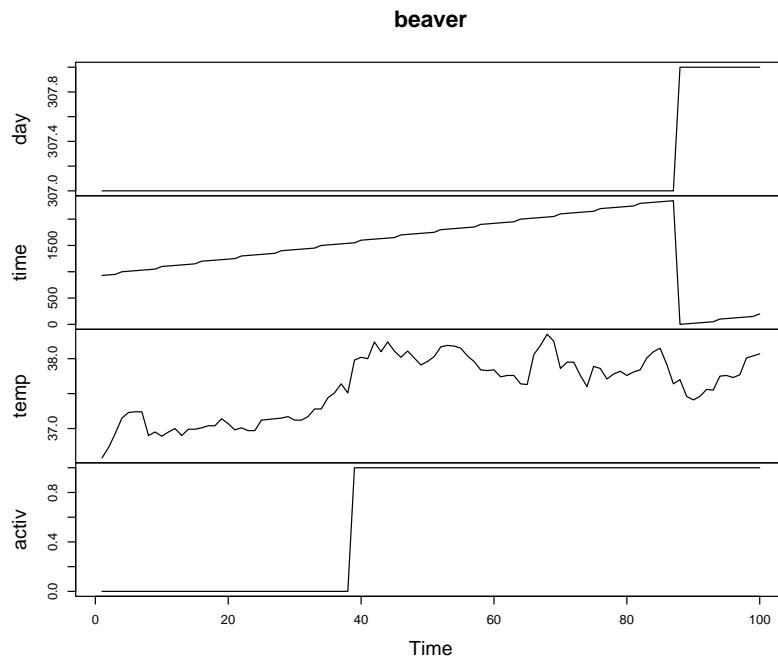


Figura 9.4: Beaver data

```

data <- beaver
# fit <- function( data ){
#   # X <- cbind(rep(1,100),data$activ) # ErrorR
#   X <- cbind(rep(1, 100), data[, "activ"])
#   para <- list(X = X, data = data)
#   # assign("para",para, frame=1) # ErrorR
#   # assign("para", para, envir = .GlobalEnv)
#   para <- para
#   # arima.mle # ErrorR
#
#   -----
#   d <- arima.mle(x = para$data$temp,
#                   model = list(ar = c(0.8)), xreg = para$X)
#   res <- arima.diag(d, plot = F, std.resid = T)$std.resid
#   res <- res[!is.na(res)]
#   list(
#     paras = c(d$model$ar, d$reg.coef, sqrt(d$sigma2)),
#     res = res - mean(res),
#     fit = X %*% d$reg.coef)
# }
#
# beaver.args <- fit(beaver)
# white.noise <- function(n.sim, ts)
#   sample(ts, size = n.sim, replace = TRUE)
# beaver.gen <- function(ts, n.sim, ran.args){
#   tsb <- ran.args$res
#   fit <- ran.args$fit
#   coeff <- ran.args$paras
#   ts$temp <- fit + coeff[4] * arima.sim(model = list(ar = coeff[1]),
#   n = n.sim, rand.gen = white.noise, ts = tsb)
#   ts

```

```
# }
# new.beaver <- beaver.gen( beaver, 100, beaver.args )
```

Now we are able to generate data, we can bootstrap and see the results of `beaver.boot` as follows:

```
# beaver.fun <- function(ts) fit(ts)$paras
# beaver.boot <- tsboot( beaver, beaver.fun, R=99, sim="model",
#                      n.sim=100, ran.gen=beaver.gen, ran.args=beaver.args)
# names(beaver.boot)
# beaver.boot$t0
# beaver.boot$t[1:10, ]
```

showing the original value of `beaver.fun` and its value for the first 10 replicate series. Are the estimated mean temperatures for the $R = 99$ simulations normal? Use `boot.ci` to obtain normal and basic bootstrap confidence intervals for the resting and active temperatures. In this analysis we have assumed that the linear model with AR(1) errors is appropriate. How would you proceed if it were not? (Section 8.2; Reynolds, 1994)'.

Ejercicio 9.3 (Practical 8.3, Sunspot data: Davison y Hinkley, 1997). Reproducir el “Practical 8.3 (Sunspot data)” en Davison, A. C., y Hinkley, D. V. (1997). Bootstrap methods and their application. Cambridge university press, <http://statwww.epfl.ch/davison/BMA> (caché):

”Consider scrambling the phases of the sunspot data. To see the original data,

[Figura 9.5]

```
data(sunspot.year) # WarningR: data set 'sunspot' not found
# ?sunspot.year
yl <- c(-50, 200)
plot(sunspot.year, ylim = yl)
abline(h = 0, lty = 2)
```

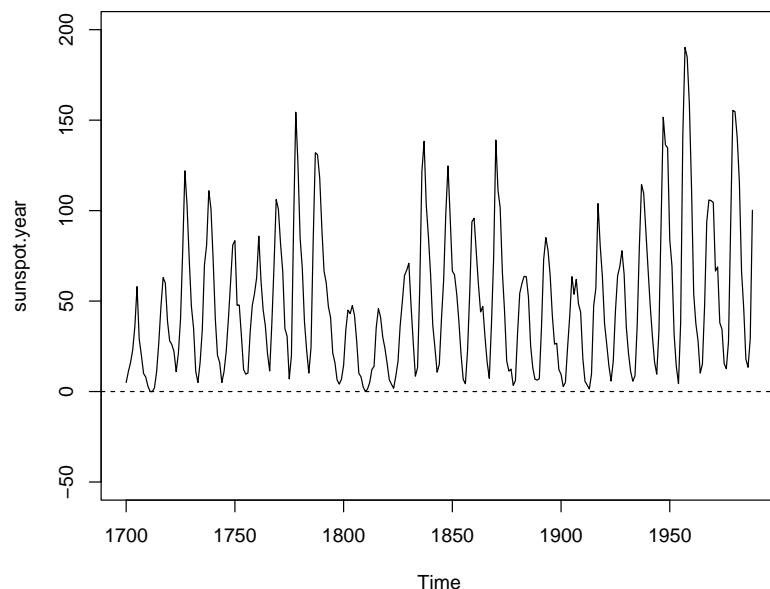


Figura 9.5: Sunspot data (yearly numbers).

two replicates generated using ordinary phase scrambling, and two phase scrambled series whose marginal distribution is the same as that of the original data:

```
set.seed(DNI)
sunspot.1 <- tsboot(sunspot.year, ...
.R.Random.seed <- sunspot.1$seed # set.seed(DNI)
sunspot.2 <- tsboot(sunspot.year, ...)
```

What features of the original data are preserved by the two algorithms? (You may find it helpful to experiment with different shapes for the figures.) (Section 8.2.4; Problem 8.4; Theiler et al, 1992)'.

9.7 Implementación en R con el paquete `forecast`

9.7.1 Bootstrap condicional (a partir de un modelo ajustado)

En la práctica normalmente se ajusta un modelo a los datos observados y posteriormente se obtienen las simulaciones condicionadas empleando el modelo ajustado.

Por ejemplo, en el caso de series de tiempo, se puede emplear la función `simulate` del paquete `forecast`: [Figura 9.6]

```
library(forecast)
# ?co2
data <- window(co2, 1990) # datos de co2 desde 1990 (hasta 1997)
plot(data, ylab = expression("Atmospheric concentration of CO"[2]),
      xlim=c(1990,2000), ylim=c(350, 375))
data2 <- window(co2, 1990, 1996) # datos de co2 desde 1990 hasta 1996
fit <- ets(data2)
# Simulación condicional
set.seed(1)
ry <- simulate(fit, 12*4)
lines(ry, col="red")

plot(forecast(fit, h=12*4), col="blue")
lines(ry, col="red")
```

[Figura 9.7]

Ver enlaces en apéndice A.1.

9.8 Spatial data

Ver enlaces en apéndice A.2.

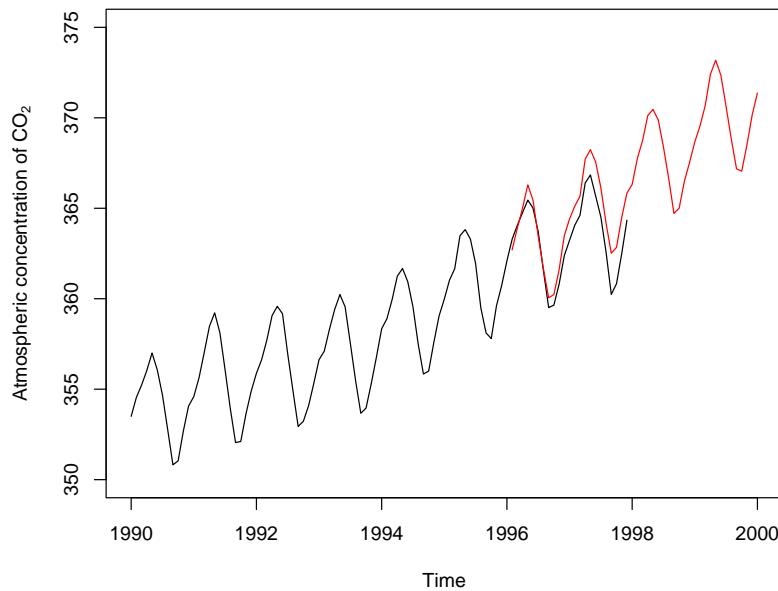


Figura 9.6: Datos de co2 (1990-1997) y simulación condicional (a partir de las observaciones desde 1990 hasta 1996).

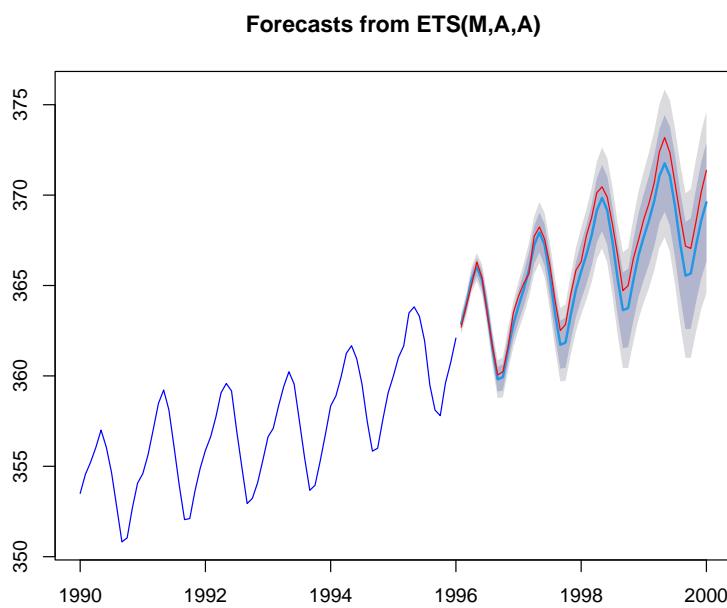


Figura 9.7: Predicción de los valores de co2 y simulación condicional (ambas a partir de las observaciones entre 1990 y 1996).

Referencias

- Akritas, M. G. (1986). Bootstrapping the Kaplan–Meier estimator. *J. Amer. Stat. Assoc.* **81**, 1032–1038.
- Beran, R. (1987). Prepivoting to reduce level error of confidence sets. *Biometrika* **74**, 457–468.
- Bhattacharya, R.N. and Ghosh, J.K. (1978). On the validity of the formal Edgeworth expansion. *Ann. Statist.* **6**, 434–451.
- Bickel, P.J. and Freedman, D.A. (1981). Some Asymptotic theory for the bootstrap. *Ann. Statist.* **12** 2, 470–482.
- Bock M., Bowman A.W. and Ismail B. (2007). Estimation and inference for error variance in bivariate nonparametric regression. *Statistics and Computing* **17**, 39–47.
- Bowman A.W. and Azzalini A. (2019). *R package ‘sm’: nonparametric smoothing methods* (version 2.2). <http://www.stats.gla.ac.uk/~adrian/sm>.
- Bowman, A., Hall, P. and Prvan, T. (1998). Bandwidth selection for the smoothing of distribution functions. *Biometrika* **85** 4, 799–808.
- Breslow, N. and Crowley, J. (1974). A large sample study of the life table and product limit estimates under random censorship. *Ann. Statist.* **2**, 437–453.
- Bühlmann, P. (1994). Blockwise bootstrap empirical processes for stationary sequences. *Ann. Statist.* **22**, 995–1012.
- Bühlmann, P. (1997). Sieve bootstrap for time series. *Bernoulli* **3**, 123–148.
- Bühlmann, P. (1998). Sieve bootstrap for smoothing in nonstationary time series. *Ann. Statist.* **26**, 48–83.
- Canty, A. J. (2002). Resampling methods in R: the boot package. *The Newsletter of the R Project* **2**, 2–7.
- Cao, R. (1990). Órdenes de convergencia para las aproximaciones normal y bootstrap en la estimación no paramétrica de la función de densidad. *Trabajos de Estadística*, vol. **5**, 2, 23–32.
- Cao, R. (1991). Rate of convergence for the wild bootstrap in nonparametric regression. *Ann. Statist.* **19**, 2226–2231.
- Cao, R. (1993). Bootstrapping the mean integrated squared error. *Jr. Mult. Anal.* **45**, 137–160.
- Cao, R. (1999). An overview of bootstrap methods for estimating and predicting in time series. *Test* **8**, 95–116.
- Cao, R., Cuevas, A. and González-Manteiga, W. (1993). A comparative study of several smoothing methods in density estimation. *Comp. Statist. Data Anal.* **17**, 153–176.
- Cao, R., Febrero-Bande, M., González-Manteiga, W., Prada-Sánchez, J.M. and García-Jurado, I. (1997). Saving computer time in constructing consistent bootstrap prediction intervals for autoregressive processes. *Commun. Statist. Simula.* **26**, 961–978.

- Cao, R. and González-Manteiga, W. (1993). Bootstrap methods in regression smoothing. *Journal of Nonparametric Statistics* **2**, 379-388.
- Cao, R. and Prada-Sánchez, J.M. (1993). Bootstrapping the mean of a symmetric population. *Statistics & Probability Letters* **17**, 43-48.
- Castillo-Páez, S., Fernández-Casal, R. and García-Soidán, P. (2019). A nonparametric bootstrap method for spatial data, *Computational Statistics and Data Analysis* **137**, 1-15.
- Castillo-Páez S., Fernández-Casal R. and García-Soidán P. (2020). Nonparametric bootstrap approach for unconditional risk mapping under heteroscedasticity. *Spatial Statistics*. In Press.
- Carlstein, E., Do, K. A., Hall, P., Hesterberg, T., and Künsch, H. R. (1998). Matched-block bootstrap for dependent data . *Bernoulli* **4** 3, 305-328.
- Cox, D. R. (1972). Regression Models and Life Tables (with discussion), *Journal of the Royal Statistical Society series B* **34**, 187–220.
- Davison, A.C. and Hinkley, D.V. (1997). *Bootstrap Methods and their Application*. Cambridge University Press.
- Efron, B. (1979). Bootstrap Methods: Another look at the Jackknife. *Ann. Statist.* **7**, 1-26.
- Efron, B. (1981). Censored data and the bootstrap. *J. Amer. Statist. Assoc.* **76**, 312–319.
- Efron, B. (1982). The Jackknife, the Bootstrap and other Resampling Plans. CBMS-NSF. *Regional Conference series in applied mathematics*.
- Efron, B. (1983). Estimating the error rate of a prediction rule: improvements on cross-validation. *J. Amer. Stat. Assoc.* **78**, 316-331.
- Efron, B. (1987). Better Bootstrap confidence intervals (with discussion). *J. Amer. Stat. Assoc.* **82**, 171-200.
- Efron, B. (1990). More Efficient Bootstrap Computations. *J. Amer. Statist. Assoc.* **85**, 79-89.
- Efron, B. and Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* **1**, 54-77.
- Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall.
- Fan J. and Gijbels I. (1996). *Local Polynomial Modelling and Its Applications*. Chapman and Hall, London.
- Fan J. and Yao Q. (1998). Efficient estimation of conditional variance functions in stochastic regression. *Biometrika*, **85**, 645–660.
- Faraway, J.J. and Jhun, M. (1990). Bootstrap choice of bandwidth for density estimation. *Jr. Amer. Statist. Assoc.* **85**, 1119–1122.
- Fernández-Casal, R. y Cao, R. (2020). *Simulación Estadística*. <https://rubenfcasal.github.io/simbook>.
- Ferretti, N. and Romo, J. (1996). Unit root bootstrap test for AR(1) models. *Biometrika* **83**, 849-860.
- Fisher, R.A. (1935). *The design of experiments*. Edinburgh: Oliver and Boyd.
- Fox, J., and Weisberg, S. (2018). *An R companion to applied regression*. Sage publications.
- Freedman, D.A. (1981). Bootstrapping regression models. *Ann. Statis.* **9** 6, 118-1228.
- Fuller, W.A. (1976). *Introduction to statistical time series*. New York: Wiley.
- Gannoun, A. (1990). Estimation non paramétrique de la médiane conditionnelle. Application à la prévision. *C. R. Acad. Sci. Paris* **310**, 295-298.
- García-Jurado, I. González-Manteiga, W., Prada-Sánchez, J.M., Febrero-Bande, M. and Cao, R. (1995). Predicting using Box-Jenkins, nonparametric and bootstrap techniques. *Technometrics* **37**, 303-310.

- González-Manteiga, W., and Cao, R. (1993). Testing the hypothesis of a general linear model using nonparametric regression estimation. *Test*, **2**, 161-188.
- González-Manteiga, W. and Prada-Sánchez, J.M. (1985). Una aplicación de los métodos de suavización no paramétricos en la técnica bootstrap. *Proceedings Jornadas Hispano-Lusas de Matemáticas*. Murcia, 1985.
- González-Manteiga, W., Prada-Sánchez, J.M. and Romo, J. (1994). The Bootstrap-A Review. *Computational Statistics* **9**, 165-205.
- Hall, P. (1986). On the bootstrap and confidence intervals. *Ann. Statist.* **14**, 1431-1452.
- Hall, P. (1988-a) Theoretical comparison of bootstrap confidence intervals. *Ann. Statist.* **16**, 927-953.
- Hall, P. (1988-b). Rate of convergence in bootstrap approximations. *Ann. Probab.* **16** 4, 1665-1684.
- Hall, P. (1990). Using the bootstrap to estimate mean squared error and select smoothing parameter in nonparametric problems. *J. Multivariate Anal.* **32**, 177-203.
- Hall, P. (1992). *The Bootstrap and Edgeworth Expansion*. Springer Verlag.
- Hall, P., Horowitz, J.L. and Jing, B-Y. (1995). On blocking rules for the bootstrap with dependent data. *Biometrika* **82**, 561-574.
- Hall, P., Marron, J.S. and Park, B. (1992). Smoothed cross-validation. *Probab. Theor. Rel. Fields* **92**, 1-20.
- Hall, P. and Martin, M.A. (1988). On bootstrap resampling and iteration. *Biometrika* **75**, 661-671.
- Härdle, W. and Mammen, E. (1993). Comparing nonparametric versus parametric regression fits. *Ann. Statist.* **21**, 1926-1947.
- Härdle, W. and Marron, J. S. (1991). Bootstrap simultaneous error bars for nonparametric regression. *Ann. Statist.* **19**, 778-796.
- Hartigan, J.A. (1969). Using subsample values as typical values. *J. Amer. Statist. Assoc.* **64**, 1303-1317.
- Heimann, G. and Kreiss, J-P. (1996). Bootstrapping general first order autoregression. *Statist. Prob. Lett.* **30**, 87-98.
- Kaplan, E. L. and P. Meier, Nonparametric estimation from incomplete observations, *J. Amer. Stat. Assoc.* **53** (1958) 457-481.
- Kreiss, J-P. and Franke, J. (1992). Bootstrapping stationary autoregressive moving average models. *J. Time Ser. Anal.* **13**, 297-317.
- Künsch, H.R. (1989). The jackknife and the bootstrap for general stationary observations. *Ann. Statist.* **17**, 1217-1241.
- Liu, R.Y. and Singh, K. (1992). Moving blocks jackknife and bootstrap capture weak dependence. In *Exploring the limits of bootstrap* (R. LePage and L Billard, Eds.), pp. 225-248. New York: Wiley.
- Mammen, E. (1992). *When does Bootstrap Work?*. Springer Verlag.
- Maritz, J.S. (1979). A note on exact robust confidence intervals for location. *Biometrika* **66**, 163-166.
- Marron, J.S. (1992). Bootstrap bandwidth selection. In *Exploring the limits of the bootstrap*, LePage, R. and Billard, L. eds., pp. 249-262. New York: Wiley.
- Nadaraya, E.A. (1964). On estimating regression. *Theor. Probab. Appl.* **9**, 141-142.
- Naik-Nimbalkar, U.V. and Rajarshi, M.B. (1994). Validity of blockwise bootstrap for empirical processes with stationary observations. *Ann. Statist.* **22**, 980-994.
- Navidi, W. (1989). Edgeworth expansions for bootstrapping regression models. *Ann. Statist.* **17** 4, 1472-1478.
- Paparoditis, E. (1996). Bootstrapping autoregressive and moving average parameters estimates of infinite order vector autoregressive processes. *J. Mult. Anal.* **57**, 277-296.

- Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Statist.* **33**, 1065–1076.
- Pascual, L., Romo, J. and Ruiz, E. (2001). Effects of parameter estimation on prediction densities: a bootstrap approach. *Int. J. Forecasting* **17**, 83-103
- Pitman, E.J.G. (1937). Significance tests which may be applied to samples from any populations. *Journal of the Royal Statistical Society, Series B*, 4, 119–130.
- Politis, D.N. and Romano, J.R. (1994a). The stationary bootstrap. *J. Amer. Statist. Assoc.* **89**, 1303-1313.
- Politis, D.N. and Romano, J.R. (1994b). Large sample confidence regions based on subsamples under minimal assumptions. *Ann. Statist.* **22**, 2031-2050.
- Politis, D.N. and Romano, J.R. (1994c). Limit theorems for weakly dependent Hilbert space valued random variables with application to the stationary bootstrap. *Statist. Sin.* **4**, 461-476.
- Politis, D.N., Romano, J.P. and Wolf, M. (1999). *Subsampling*. Springer Verlag.
- Prada-Sánchez, J.M. and Cotos-Yáñez, T. (1997). A Simulation Study of Iterated and Non-iterated Bootstrap Methods for Bias Reduction and Confidence Interval Estimation. *Comm. Statist .-Simula.* **26** 3, 927-946.
- Prada-Sánchez, J.M. and Otero-Cepeda, X.L. (1989). The use of smooth bootstrap techniques for estimating the error rate of a prediction rule. *Comm. Statist .-Simula.* **18** 3, 1169-1186.
- Quenouille, M. (1949). Approximate test of correlation in time series. *J. Roy. Statist. Soc. Ser. B* **11**, 18-84.
- Radulović, D. (1996). The bootstrap for the mean of strong mixing sequences under minimal conditions. *Statist. Prob. Lett.* **28**, 65-72.
- Reid, N. (1981). Estimating the median survival time. *Biometrika* **68**, 601–608.
- Rizzo, M.L. (2008). *Statistical Computing with R*. Chapman&Hall/CRC
- Rosenblatt, M. (1956). Remarks on some nonparametric estimate of a density function. *Ann. Math. Statist.* **27**, 832–837.
- Rubin, D.B. (1981). The Bayesian Bootstrap. *Ann. Statist.* **9** 1, 130-134.
- Rubinstein, R.Y. (1981). *Simulation and the Monte Carlo Method*. Wiley.
- Schucany, W., Gray, H. and Owen, O. (1971). On bias reduction in estimation. *J. Amer. Stat. Assoc.* **66**, 524-533.
- Shao, J. (1999). *Mathematical Statistics*. Springer.
- Shao, J. (2006). *Mathematical Statistics: exercises and solutions*. Springer.
- Sheather, S.J. and Jones, M.C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Jr. Royal Statist. Soc. Ser. B* **53**, 683–690.
- Silverman, B.W. (1986). *Density Estimation*. Chapman and Hall.
- Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap*. Springer Verlag.
- Sing, K. (1981). On the asymptotic accuracy of Efron's bootstrap. *Ann. Statist.* **9** 6, 1187-1195.
- Stine, R.A. (1987). Estimating properties of autoregressive forecasts. *J. Amer. Stat. Assoc.* **82**, 1072-1078.
- Taylor, C. C. (1989). Bootstrap choice of the smoothing parameter in kernel density estimation. *Biometrika* **76**, 705–712.
- Thombs, L.A. and Schucany, W.R. (1990). Bootstrap prediction intervals for autoregression. *J. Amer. Stat. Assoc.* **85**, 486-492.

- Tukey, J. (1958). Bias and confidence in not quite large samples, abstract, *Ann. Math. Statist.* **29**, 614.
- Wand M.P. and Jones M.C. (1995) *Kernel Smoothing*. Chapman and Hall, London.
- Watson, G.S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics Ser. A* **26**, 359-372.
- Welch, B.L. (1937). On the z-test in randomized blocks and Latin squares. *Biometrika*, 29, 21-52.
- Wu, C.-F. J. (1986). Jackknife, bootstrap and other resampling methods in regression analysis. *Ann. Statist.* **14**, 1261-1350.

Apéndice A

Enlaces

Recursos para el aprendizaje de R (<https://rubenfcasal.github.io/post/ayuda-y-recursos-para-el-aprendizaje-de-r>): A continuación se muestran algunos recursos que pueden ser útiles para el aprendizaje de R y la obtención de ayuda...

Ayuda online:

- Ayuda en línea sobre funciones o paquetes: RDocumentation
- Buscador RSeek
- StackOverflow

Cursos: algunos cursos gratuitos:

- Coursera:
 - Introducción a Data Science: Programación Estadística con R
 - Mastering Software Development in R
- DataCamp:
 - Introducción a R
- Stanford online:
 - Statistical Learning
- Curso UCA: Introducción a R, R-commander y shiny
- Udacity: Data Analysis with R
- Swirl Courses: se pueden hacer cursos desde el propio R con el paquete swirl.

Para información sobre cursos en castellano se puede recurrir a la web de R-Hispano en el apartado formación. Algunos de los cursos que aparecen en entradas antiguas son gratuitos. Ver: Cursos MOOC relacionados con R.

Libros

- *Iniciación:*
 - 2011 - The Art of R Programming. A Tour of Statistical Software Design, (No Starch Press)
 - R for Data Science (online, O'Reilly)
 - Hands-On Programming with R: Write Your Own Functions and Simulations, by Garrett Grolemund (O'Reilly)
- *Avanzados:*
 - 2008 - Software for Data Analysis: Programming with R - Chambers (Springer)

- Advanced R by Hadley Wickham (online: 1^a ed, 2^a ed, Chapman & Hall)
- R packages by Hadley Wickham (online, O'Reilly)
- **Bookdown:** el paquete `bookdown` de R permite escribir libros empleando R Markdown y compartirlos. En <https://bookdown.org> está disponible una selección de libros escritos con este paquete (un listado más completo está disponible aquí). Algunos libros en este formato en castellano son:
 - Prácticas de Simulación (disponible en el repositorio de GitHub [rubenfcasal/simbook](https://github.com/rubenfcasal/simbook)).
 - Escritura de libros con bookdown (disponible en el repositorio de GitHub [rubenfcasal/bookdown_intro](https://github.com/rubenfcasal/bookdown_intro)).
 - R para profesionales de los datos: una introducción.
 - Estadística Básica Edulcorada.

Material online: en la web se puede encontrar mucho material adicional, por ejemplo:

- CRAN: Other R documentation
- RStudio: Online learning, Webinars, shiny, tidyverse.
 - CheatSheets: RMarkdown, Shiny, dplyr, tidyr, stringr.
- Blogs en inglés:
 - <https://www.r-bloggers.com/>
 - <https://www.littlemissdata.com/blog/rstudioconf2019>
 - RStudio: <https://blog.rstudio.com>
 - Microsoft Revolutions: <https://blog.revolutionanalytics.com>
- Blogs en castellano:
 - <https://www.datanalytics.com>
 - <http://oscarperpinan.github.io/R>
 - <http://rubenfcasal.github.io>
- Listas de correo:
 - Listas de distribución de r-project.org: <https://stat.ethz.ch/mailman/listinfo>
 - Búsqueda en R-help: <http://r.789695.n4.nabble.com/R-help-f789696.html>
 - Búsqueda en R-help-es: <https://r-help-es.r-project.narkive.com>
 - <https://grokbase.com/g/r/r-help-es>
 - Archivos de R-help-es: <https://stat.ethz.ch/pipermail/r-help-es>

A.1 Forecasting: Principles and Practice

Forecasting: Principles and Practice, 2^a ed by Rob J. Hyndman and George Athanasopoulos:

- 3.1 Some simple forecasting methods
- 3.6 The forecast package in R
- 11.4 Bootstrapping and bagging
- Appendix: For instructors

A.2 Spatial data

- Apuntes de simulación:
 - 9 Simulación de Distribuciones Multidimensionales
 - 9.3 Simulación condicional e incondicional
- Castillo-Páez, S., Fernández-Casal, R., García-Soidán, P. A nonparametric bootstrap method for spatial data, Computational Statistics and Data Analysis, 137 (2019) 1-15.
- Poster bootstrap condicional (pdf)
- npsp, post en castellano.
- Tesis geoestadística espacio-temporal (pdf)

Apéndice B

Introducción al procesamiento en paralelo en R

En este apéndice se pretenden mostrar las principales herramientas para el procesamiento en paralelo disponibles en R y dar una idea de su funcionamiento. Para más detalles se recomienda ver CRAN Task View: High-Performance and Parallel Computing with R (además de HPC, High Performance Computing, también incluye herramientas para computación distribuida)¹.

B.1 Introducción

Emplearemos la siguiente terminología:

- **Núcleo**: término empleado para referirse a un procesador lógico de un equipo (un equipo puede tener un único procesador con múltiples núcleos que pueden realizar operaciones en paralelo). También podría referirse aquí a un equipo (nodo) de una red (clúster de equipos; en la práctica cada uno puede tener múltiples núcleos). *Un núcleo puede ejecutar procesos en serie.*
- **Clúster**: colección de núcleos en un equipo o red de equipos. *Un clúster puede ejecutar varios procesos en paralelo.*

Por defecto la versión oficial de R emplea un único núcleo, aunque se puede compilar de forma que realice cálculos en paralelo (e.g. librería LAPACK). También están disponibles otras versiones de R que ya implementan por defecto procesamiento en paralelo (multithread):

- Microsoft R Open: versión de R con rendimiento mejorado.

Métodos simples de paralelización²:

- *Forking*: Copia el proceso de R a un nuevo núcleo (se comparte el entorno de trabajo). Es el más simple y eficiente pero **no está disponible en Windows**.
- *Socket*: Lanza una nueva versión de R en cada núcleo, como si se tratase de un cluster de equipos comunicados a través de red (hay que crear un entorno de trabajo en cada núcleo). Disponible en todos los sistemas operativos.

B.2 Paquetes en R

Hay varios paquetes que se pueden usar para el procesamiento paralelo en R, entre ellos podríamos destacar:

¹También puede ser de interés la presentación R y HPC (uso de R en el CESGA) de Aurelio Rodríguez en las VI Xornadas de Usuarios de R en Galicia

²Realmente las herramientas estándar son *OpenMP* para el procesamiento en paralelo con memoria compartida en un único equipo y *MPI* para la computación distribuida en múltiples nodos.

- **parallel**: forma parte de la instalación base de R y fusiona los paquetes **multicore** (forking) y **snow** (sockets; Simple Network of Workstations). Además incluye herramientas para la generación de números aleatorios en paralelo (cada proceso empleará una secuencia y los resultados serán reproducibles).

Incluye versiones “paralelizadas” de la familia ***apply()**: **mclapply()** (forking), **parLapply()** (socket), ...

- **foreach**: permite realizar iteraciones y admite paralelización con el operador **%dopar%**, aunque requiere paquetes adicionales como **doSNOW** o **doParallel** (recomendado).
- **rslurm**: permite la ejecución distribuida en clústeres Linux que implementen SLURM (Simple Linux Utility for Resource Management), un gestor de recursos de código abierto muy empleado.

B.3 Ejemplos

Si se emplea el paquete **parallel** en sistemas tipo Unix (Linux, Mac OS X, ...), se podría evaluar en paralelo una función llamando directamente a **mclapply()**. Por defecto empleará todos los núcleos disponibles, pero se puede especificar un número menor mediante el argumento **mc.cores**.

```
library(parallel)
ncores <- detectCores()
ncores

## [1] 8

func <- function(k) {
  i_boot <- sample(nrow(iris), replace = TRUE)
  lm(Petal.Width ~ Petal.Length, data = iris[i_boot, ])$coefficients
}

RNGkind("L'Ecuyer-CMRG") # Establecemos Pierre L'Ecuyer's RngStreams...
set.seed(1)

system.time(res.boot <- mclapply(1:100, func)) # En Windows llama a lapply() (mc.cores = 1)

##    user  system elapsed
##   0.06    0.00    0.06

# res.boot <- mclapply(1:100, func, mc.cores = ncores - 1) # En Windows genera un error si mc.cores ...
```

En Windows habría que crear previamente un cluster, llamar a una de las funciones **par*apply()** y finalizar el cluster:

```
cl <- makeCluster(ncores - 1, type = "PSOCK")
clusterSetRNGStream(cl, 1) # Establecemos Pierre L'Ecuyer's RngStreams con semilla 1...

system.time(res.boot <- parSapply(cl, 1:100, func))

##    user  system elapsed
##   0.01    0.00    0.06

# stopCluster(cl)

str(res.boot)

## num [1:2, 1:100] -0.415 0.429 -0.363 0.42 -0.342 ...
## - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2] "(Intercept)" "Petal.Length"
##   ..$ : NULL
```

Esto también se puede realizar en Linux (`type = "FORK"`), aunque podríamos estar trabajando ya en un cluster de equipos...

También podríamos emplear balance de carga si el tiempo de computación es variable (e.g. `parLapplyLB()` o `clusterApplyLB()`) pero no sería recomendable si se emplean números pseudo-aleatorios (los resultados no serían reproducibles).

Además, empleando las herramientas del paquete `snow` se puede representar el uso del cluster (*experimental* en Windows):

```
# library(snow)
ctime <- snow::snow.time(snow::parSapply(cl, 1:100, func))
ctime
plot(ctime)
```

Hay que tener en cuenta la sobrecarga adicional debida a la comunicación entre nodos al paralelizar (especialmente con el enfoque de socket).

B.3.1 Procesamiento en paralelo con la función `boot()`

La función `boot::boot()` incluye parámetros para el procesamiento en paralelo: `parallel = c("no", "multicore", "snow")`, `ncpus`, `cl`. Si `parallel = "snow"` se crea un clúster en la máquina local durante la ejecución, salvo que se establezca con el parámetro `cl`.

Veamos un ejemplo empleando una muestra simulada:

```
n <- 100
rate <- 0.01
mu <- 1/rate
muestra <- rexp(n, rate = rate)
media <- mean(muestra)
desv <- sd(muestra)

library(boot)
statistic <- function(data, i){
  remuestra <- data[i]
  c(mean(remuestra), var(remuestra)/length(remuestra))
}
B <- 2000
set.seed(1)

system.time(res.boot <- boot(muestra, statistic, R = B))

##    user  system elapsed
##    0.07    0.00    0.08

# system.time(res.boot <- boot(muestra, statistic, R = B, parallel = "snow"))
system.time(res.boot <- boot(muestra, statistic, R = B, parallel = "snow", cl = cl))

##    user  system elapsed
##    0.05    0.00    0.04
```

B.3.2 Estudio de simulación

Si se trata de un estudio más complejo, como por ejemplo un estudio de simulación en el que se emplea bootstrap, la recomendación sería tratar de paralelizar en el nivel superior para minimizar la sobrecarga debida a la comunicación entre nodos.

Por ejemplo, a continuación se realiza un estudio similar al mostrado en la Sección 4.6.2 pero comparando las probabilidades de cobertura y las longitudes de los intervalos de confianza implementados en la función `boot.ci()`.

```

t.ini <- proc.time()

nsim <- 500

getSimulation <- function(isim, B = 2000, n = 30, alfa = 0.1, mu = 100) {
  rate <- 1/mu # 0.01
  resnames <- c("Cobertura", "Longitud")
  # intervals <- c("Normal", "Percentil", "Percentil-t", "Percentil-t simetrizado")
  intervals <- c("Normal", "Basic", "Studentized", "Percentil", "BCa")
  names(intervals) <- c("normal","basic", "student", "percent", "bca")
  intervals <- intervals[1:4]
  resultados <- array(dim = c(length(resnames), length(intervals)))
  dimnames(resultados) <- list(resnames, intervals)
  # for (isim in 1:nsim) { # isim <- 1
  muestra <- rexp(n, rate = 0.01)
  media <- mean(muestra)
  desv <- sd(muestra)
  # boot()
  library(boot)
  statistic <- function(data, i){
    remuestra <- data[i]
    c(mean(remuestra), var(remuestra)/length(remuestra))
  }
  res.boot <- boot(muestra, statistic, R = B)
  res <- boot.ci(res.boot, conf = 1 - alfa)
  # Intervales
  res <- sapply(res[names(intervals)], function(x) {
    l <- length(x)
    x[c(l-1, l)]
  })
  # resultados
  resultados[1, ] <- apply(res, 2,
                            function(ic) (ic[1] < mu) && (mu < ic[2])) # Cobertura
  resultados[2, ] <- apply(res, 2, diff) # Longitud
  resultados
}

parallel::clusterSetRNGStream(cl)
result <- parLapply(cl, 1:nsim, getSimulation)
# stopCluster(cl)

# result
t.fin <- proc.time() - t.ini
print(t.fin)

##      user    system elapsed
##      0.03    0.00    9.80

resnames <- c("Cobertura", "Longitud")
intervals <- c("Normal", "Basic", "Studentized", "Percentil", "BCa")
names(intervals) <- c("normal","basic", "student", "percent", "bca")
intervals <- intervals[1:4]
resultados <- sapply(result, function(x) x)
dim(resultados) <- c(length(resnames), length(intervals), nsim)
dimnames(resultados) <- list(resnames, intervals, NULL)

res <- t(apply(resultados, c(1, 2), mean))

```

```
res  
##          Cobertura Longitud  
## Normal      0.866 57.05639  
## Basic       0.860 56.97389  
## Studentized 0.900 65.72609  
## Percentil    0.868 56.97389  
knitr::kable(res, digits = 3)
```

	Cobertura	Longitud
Normal	0.866	57.056
Basic	0.860	56.974
Studentized	0.900	65.726
Percentil	0.868	56.974

El último paso es finalizar el cluster:

```
stopCluster(cl)
```