Métodos de Remuestreo

Tema 4. Remuestreos en Modelos Lineales y Series Temporales.

basado en

B. Efron, R. Tibshirani (1993). An Introduction to the bootstrap.

O. Kirchkamp (2017). Resampling methods.

Curso 2018/19

Introducción a la Regresión Lineal

- En el modelo clásico de regresión lineal se tiene un conjunto de n parejas de observaciones $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ tal que cada $\mathbf{x}_i$ es un par $\mathbf{x}_i = (\mathbf{c}_i, y_i)$.

- Cada $\mathbf{c}_i$ es un vector de dimensión p tal que $\mathbf{c}_i = (c_{i1}, c_{i2}, \ldots, c_{ip})$ se suele denominar como vector de *covariables* o *predictores*.

- y_i es un número real denominado *respuesta*.

- Se define la esperanza condicional de la respuesta y_i dado el predictor $\mathbf{c}_i$ como

$$\mu_i = E\left(y_i | \mathbf{c}_i\right)$$

para $i = 1, 2, \ldots, n$.

Introducción a la Regresión Lineal

- La suposición básica de los modelos lineales es que μ_i es una combinación lineal de los componentes del vector $\mathbf{c}_i$

$$\mu_i = \mathbf{c}_i \boldsymbol{\beta} = \sum_{j=1}^{p} c_{ij} \beta_j$$

- El vector de parámetros $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$ es desconocido de modo que se trata de estimarlo mediante los datos observados $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$.

- El término *lineal* se refiere a la forma lineal de la esperanza, no a que las términos de $\mathbf{c}_i$ puedan estar elevados a un exponente dado.

- La estructura habitual es (para $i = 1, 2, \ldots, n$)

$$y_i = \mathbf{c}_i \boldsymbol{\beta} + \varepsilon_i$$

Introducción a la Regresión Lineal

- Los términos de error ε_i se asume que proceden de una distribución desconocida F que tiene esperanza igual a 0:

$$F \rightarrow (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$$

tal que $E_F(\varepsilon_i) = 0$.

- Esto implica que

$$E\left(y_i|\mathbf{c}_i\right) = E\left(\mathbf{c}_i\boldsymbol{\beta} + \varepsilon_i|\mathbf{c}_i\right) = E\left(\mathbf{c}_i\boldsymbol{\beta}|\mathbf{c}_i\right) + E\left(\varepsilon_i|\mathbf{c}_i\right) = \mathbf{c}_i\boldsymbol{\beta}$$

- Ya que al ser ε_i independientes de $\mathbf{c}_i$ entonces

$$E\left(\varepsilon_i|\mathbf{c}_i\right) = E\left(\varepsilon_i\right) = 0$$

Introducción a la Regresión Lineal

- Para estimar los parámetros de la regresión β a partir de los datos originales, se toma un valor inicial, digamos $\mathbf{b}$ de β,

$$\mathrm{ECM}(\mathbf{b}) = \sum_{i=1}^{n} (y_i - \mathbf{c}_i \mathbf{b})^2$$

- De modo que el estimador de mínimos cuadrados de β es el valor $\widehat{\beta}$ de $\mathbf{b}$ que minimiza el error cuadrático medio

$$\boxed{\mathrm{ECM}(\widehat{\beta}) = \min_{\mathbf{b}} \left(\mathrm{ECM}(\mathbf{b})\right).}$$

Introducción a la Regresión Lineal

- Se define la llamada *matriz de diseño* como $\mathbf{C}$, de orden $n \times p$, tal que la fila $i-$ésima es $\mathbf{c}_i$, y se denomina $\mathbf{y}$ al vector $(y_1, y_2, \ldots, y_n)'$
- Entonces el estimador de mínimos cuadrados es la solución de las *ecuaciones normales*

$$\mathbf{C}'\mathbf{C}\widehat{\beta} = \mathbf{C}'\mathbf{y}$$

- es decir

$$\boxed{\widehat{\beta} = (\mathbf{C}'\mathbf{C})^{-1}\mathbf{C}'\mathbf{y}}$$

Introducción a la Regresión Lineal con R

- En R hay muchos paquetes estadísticos que permiten trabajar con métodos de regresión.

- La orden básica en R es `lm`.

- Ver, por ejemplo, como tutoriales:

  Curso completo sobre métodos de regresión con R:

  ```
  http://www.et.bs.ehu.es/~etptupaf/nuevo/ficheros/estad3/
  nreg1.pdf
  ```

  Tutorial corto sobre métodos de regresión con R:

  ```
  http://www.montefiore.ulg.ac.be/~kvansteen/GBIO0009-1/
  ac20092010/Class8/Using%20R%20for%20linear%20regression.pdf
  ```

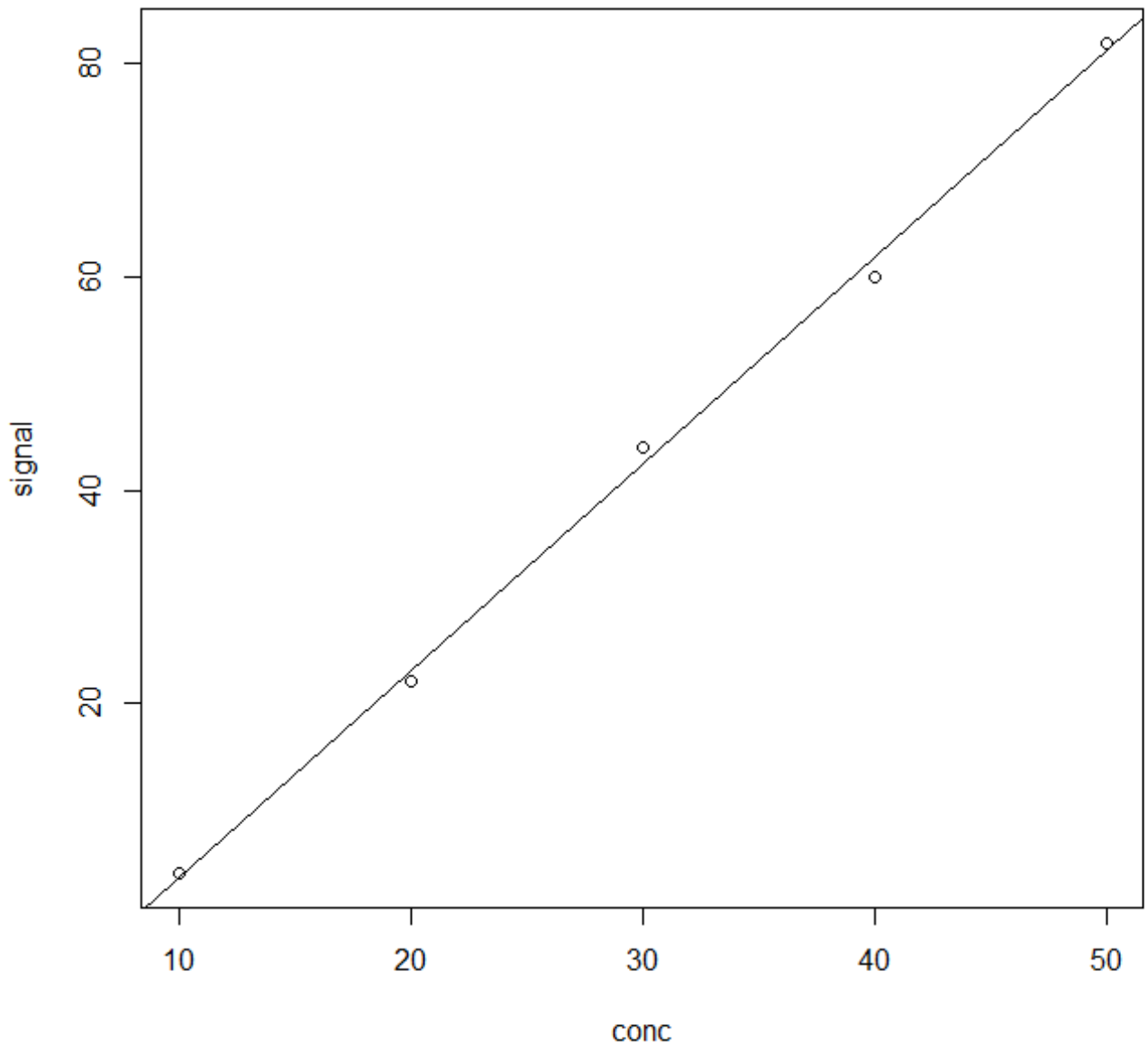
Introducción a la Regresión Lineal con R

Supongamos un ejemplo muy simple sobre dos vectores de datos:

```
conc = c(10,20,30,40,50)
signal = c(4,22,44,60,82)

lm.r = lm(signal ~ conc)
summary(lm.r)

plot(conc, signal)
abline(lm.r)
```

Introducción a la Regresión Lineal con R

```
Call:
lm(formula = signal ~ conc)

Residuals:
   1    2    3    4    5
 0.4 -1.0  1.6 -1.8  0.8

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -15.80000    1.66933  -9.465   0.0025 **
conc          1.94000    0.05033  38.544 3.84e-05 ***
---
Signif. codes: 0 '***' 0.001 '**'
0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.592 on 3 degrees of freedom
Multiple R-squared:  0.998,  Adjusted R-squared: 0.9973
F-statistic:  1486 on 1 and 3 DF,  p-value: 3.842e-05
```


signal
20
40
60
80
10
20
30
40
50
conc

Introducción a la Regresión Lineal con R

```
# Coeficientes del modelo
coef(lm.r)
```

```
(Intercept)        conc
     -15.80        1.94
```

```
# Residuos del modelo
resid(lm.r)
```

```
   1    2    3    4    5
 0.4 -1.0  1.6 -1.8  0.8
```

Introducción a la Regresión Lineal con R

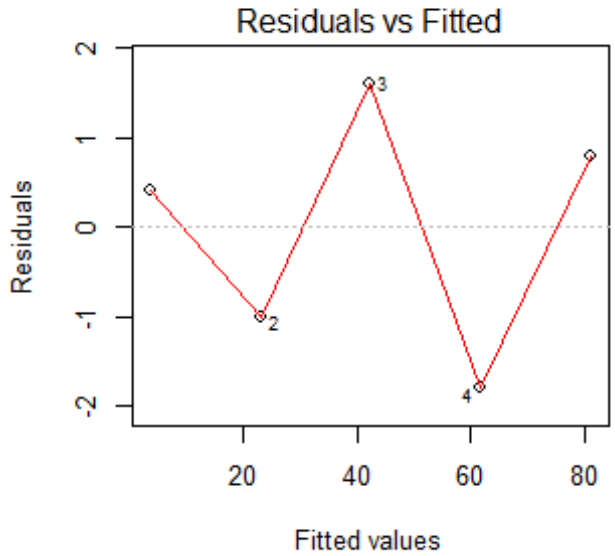
```
# Valores predichos
fitted(lm.r)
```

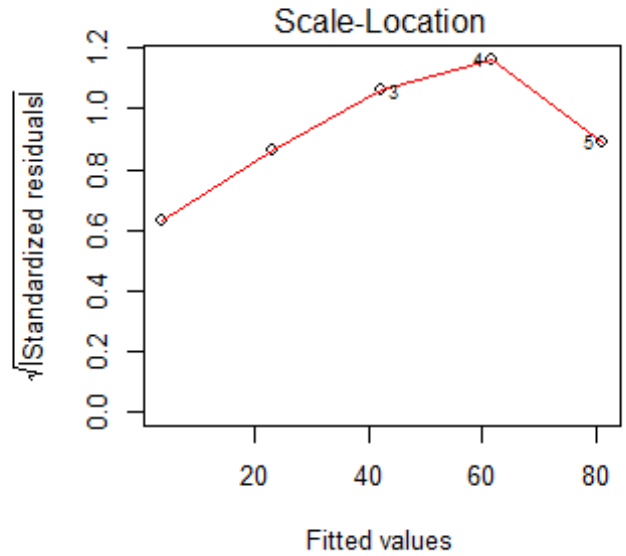
```
   1    2    3    4    5
 3.6 23.0 42.4 61.8 81.2
```

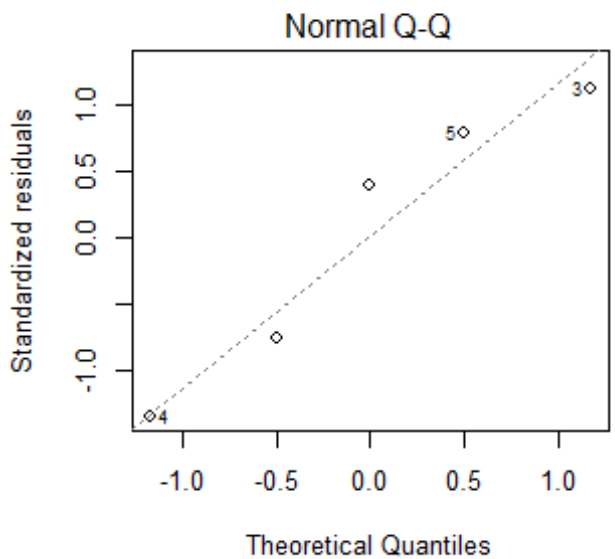
```
# Intervalos de confianza para los parametros
confint(lm.r)
```

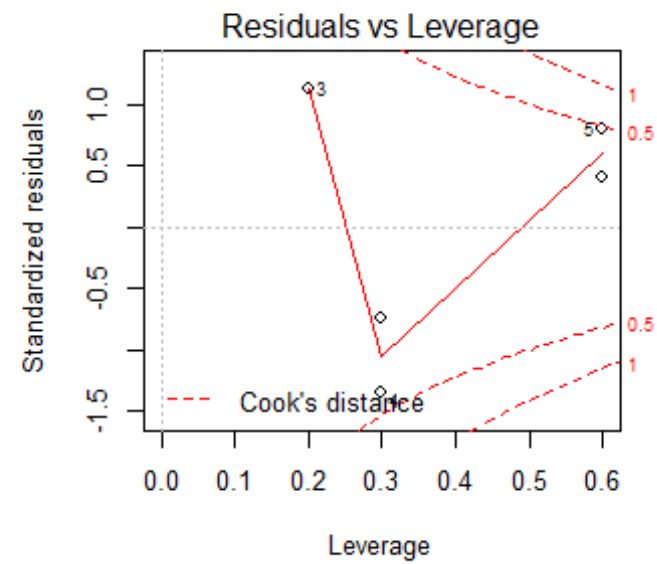
```
                2.5 %    97.5 %
(Intercept) -21.11256 -10.48744
conc          1.77982   2.10018
```

```
layout(matrix(1:4,2,2))
plot(lm.r)
```


Residuals vs Fitted
Residuals
Fitted values
2
1
0
-1
-2
20
40
60
80
3
2
4


Scale-Location
√|Standardized residuals|
Fitted values
0.0
0.2
0.4
0.6
0.8
1.0
1.2
20
40
60
80
4
3
5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0
3
5
4


Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance
1.0
0.5
0.0
-0.5
-1.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
3
5
1
0.5
0.5
1

Introducción a la Regresión Lineal con R

```
newconc = c(5, 15, 25, 35, 45)

# Prediccion de nuevas observaciones
predict(lm.r, data.frame(conc=newconc), level=0.9,
interval="confidence")
```

```
   fit        lwr        upr
1 -6.1 -9.502218 -2.697782
2 13.3 10.858090 15.741910
3 32.7 30.923250 34.476750
4 52.1 50.323250 53.876750
5 71.5 69.058090 73.941910
```

Bootstrap en Regresión Lineal

- Ninguno de los cálculos anteriores requiere el uso del bootstrap, pero la aplicación al modelo de regresión lineal simple sirve como base para otros modelos más complejos.

- El modelo de probabilidad $P \rightarrow \mathbf{x}$ para la regresión lineal tiene dos componentes: $P = (\beta, F)$ donde β es el vector de parámetros de la regresión y F es la distribución de los errores.

- En principio, se dispone del estimador de $\widehat{\beta}$ de mínimos cuadrados. Pero hace falta estimar F.

- Si β fuese ***conocido*** entonces se podrían calcular los errores como $\varepsilon_i = y_i - \mathbf{c}_i\beta$ para $i = 1, 2, \ldots, n$ y se estimaría F mediante su distribución empírica.

Bootstrap en Regresión Lineal

- No se conoce β pero se puede usar $\widehat{\beta}$ para calcular los errores aproximados o *residuos*

$$\widehat{\varepsilon}_i = y_i - \mathbf{c}_i \widehat{\beta}$$

para $i = 1, 2, \ldots, n$

- Se usa la distribución empírica de $\widehat{\varepsilon}_i$

$$\widehat{F} \quad \rightarrow \quad \text{probabilidad igual a } 1/n \quad \text{en } \widehat{\varepsilon}_i$$

para $i = 1, 2, \ldots, n$, de modo que $\widehat{F}$ tiene esperanza igual a 0.

Bootstrap en Regresión Lineal

- A partir de $\widehat{P} = (\widehat{\beta}, \widehat{F})$ se calculan los muestras bootstrap $\widehat{P} \rightarrow \mathbf{x}^*$
- Para generar $\mathbf{x}^*$ se toma primero una muestra aleatoria de términos de error

$$\widehat{F} \rightarrow (\varepsilon_1^*, \varepsilon_2^*, \ldots, \varepsilon_n^*) = \varepsilon^*$$

- Cada ε_i^* es igual a cualquiera de los n valores de $\widehat{\varepsilon}_j$ con probabilidad $1/n$
- Así, las respuestas boostrap se generan mediante

$$y_i^* = \mathbf{c}_i \widehat{\beta} + \varepsilon_i^*$$

para $i = 1, 2, \ldots, n$ donde $\widehat{\beta}$ es el mismo para todo i.

Bootstrap en Regresión Lineal

- En conjunto, las muestras bootstrap son $\mathbf{x}_i^* = (\mathbf{c}_i, y_i^*)$
- Se observa que los valores $\mathbf{c}_i$ (vector de covariables) son iguales tanto en los datos originales como en los datos bootstrap. Esto se debe a que $\mathbf{c}_i$ son valores *fijos* y no aleatorios.
- El estimador bootstrap $\widehat{\beta}^*$ es el valor que minimiza el error cuadrático residual

$$\sum_{i=1}^{n} \left(y_i^* - \mathbf{c}_i \widehat{\beta}^* \right)^2 = \min_{\mathbf{b}} \sum_{i=1}^{n} (y_i^* - \mathbf{c}_i \mathbf{b})^2$$

- y con las ecuaciones normales aplicadas a los datos bootstrap se obtiene

$$\widehat{\beta}^* = (\mathbf{C}'\mathbf{C})^{-1} \mathbf{C}'\mathbf{y}^*$$

Bootstrap en Regresión Lineal

- El error estándar de los componentes de $\widehat{\beta}^*$ se obtiene de manera directa

$$\begin{aligned} \mathrm{Var}\left(\widehat{\beta}^*\right) &= \left(\mathbf{C}'\mathbf{C}\right)^{-1}\mathbf{C}'\mathrm{Var}(\mathbf{y}^*)\mathbf{C}\left(\mathbf{C}'\mathbf{C}\right)^{-1} \\ &= \widehat{\sigma}_F^2\left(\mathbf{C}'\mathbf{C}\right)^{-1} \end{aligned}$$

- ya que $\mathrm{Var}(\mathbf{y}^*) = \widehat{\sigma}_F^2\mathbf{I}$ donde $\mathbf{I}$ es la matriz identidad.

- Así, el estimador bootstrap del error estándar es igual al usual en regresión lineal.

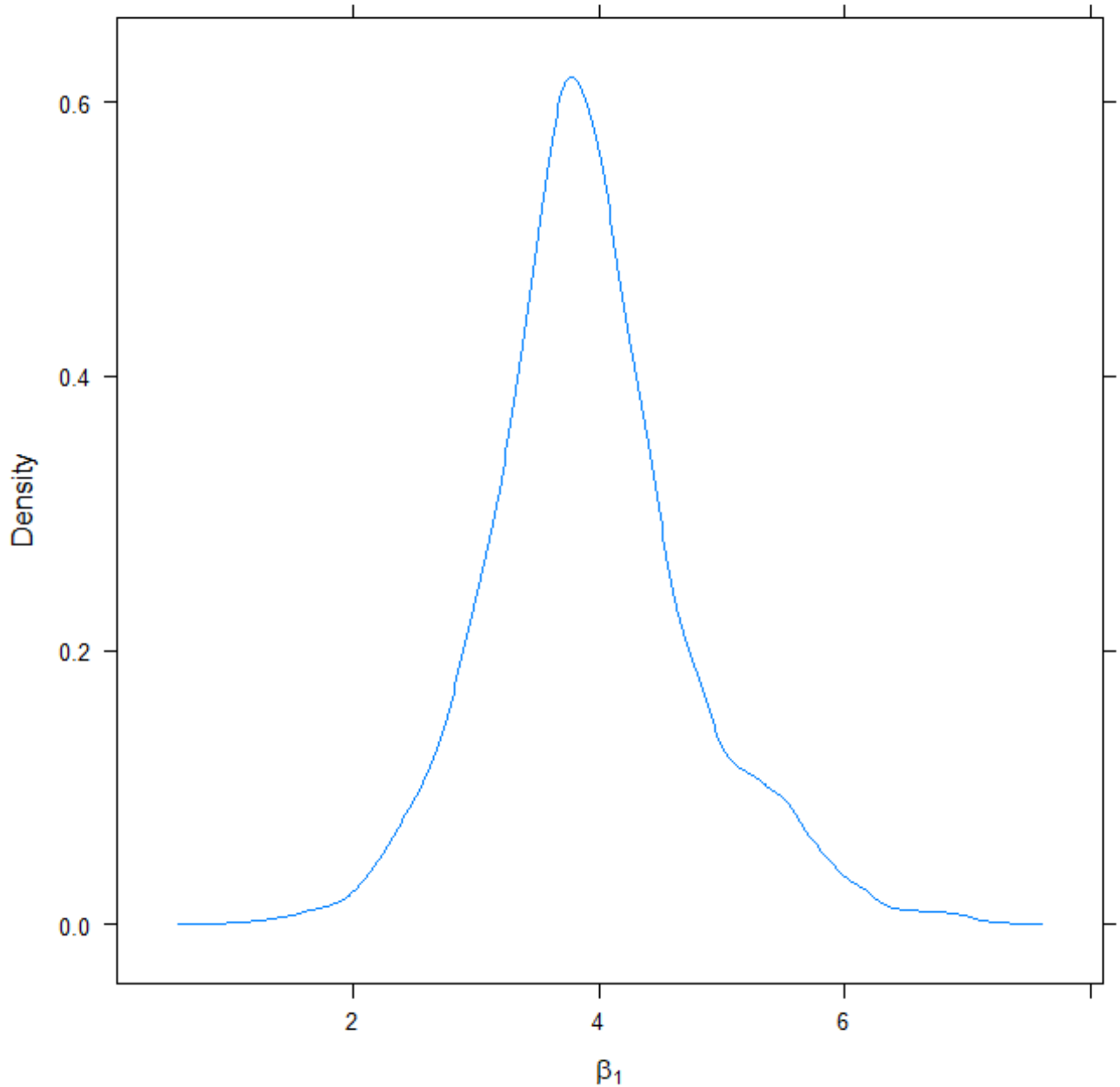
Ejemplo de Regresión bootstrap con residuos

- Se simula un modelo de regresión lineal con errores distribuidos según una normal

```
N = 15
sd = 1.5
x = rnorm(N)
y = 3*x + sd*rnorm(N)^2
est = lm(y ~ x)

kk = residuals(est)
beta = coef(est)
```

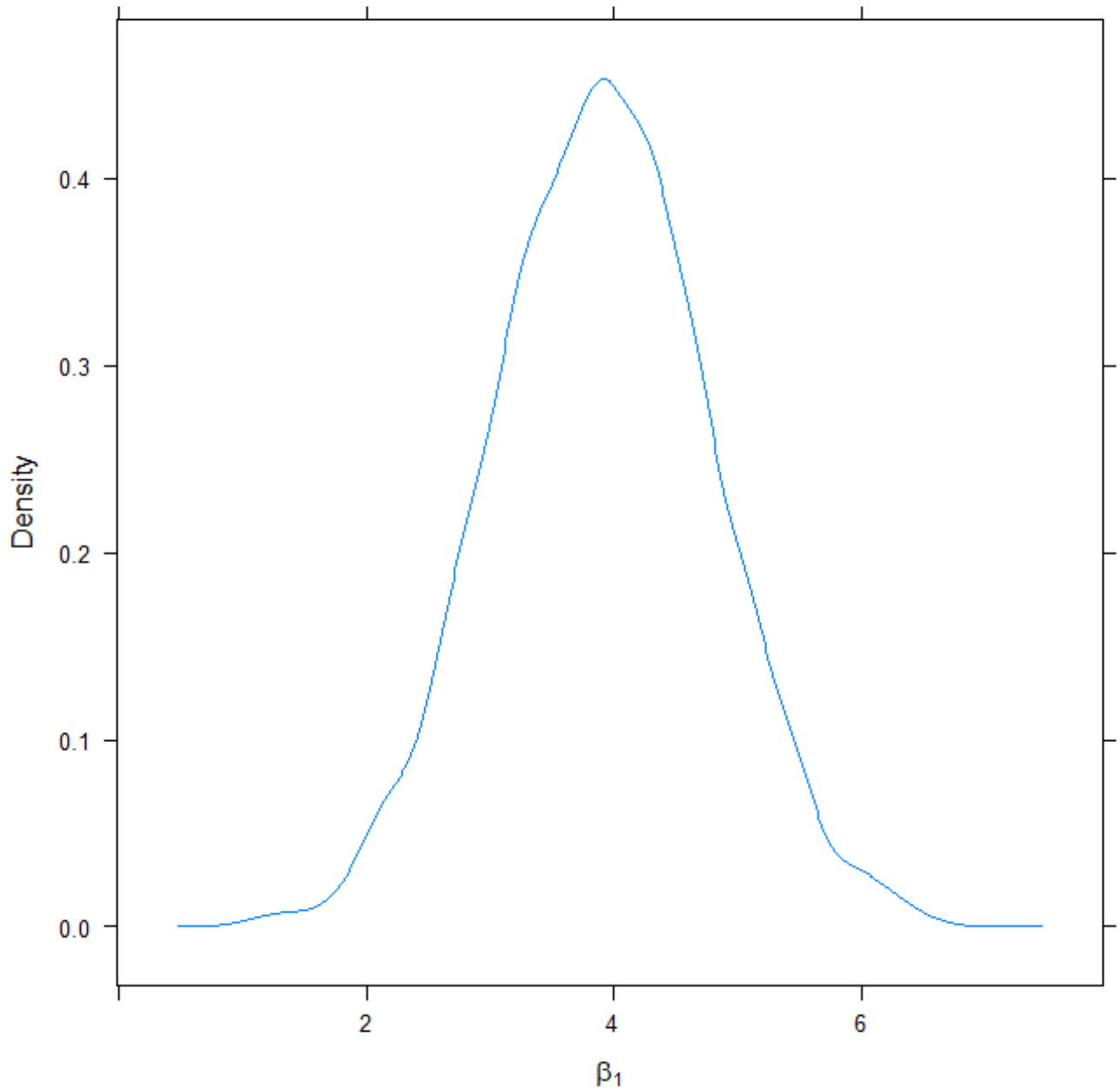
Ejemplo de Regresión bootstrap con residuos

```
bootResid = replicate(5000,
    {epsilon = sample(kk, replace=TRUE);
    coef(lm((cbind(1,x)%*%beta + epsilon) ~ x))[2]}
)

library(latticeExtra)

densityplot(~bootResid, plot.points=FALSE,
auto.key=list(columns=2), xlab=expression(beta[1]))
```


0.6
0.4
0.2
0.0
Density
2
4
6
β1

Ejemplo de Regresión bootstrap con residuos

```
# Otra opcion simulando directamente
# desde el modelo de regresion estimado

bootResid2=replicate(5000,
coef(lm(simulate(est)[,1] ~ x))[2])

densityplot(~bootResid2, plot.points=FALSE,
auto.key=list(columns=2), xlab=expression(beta[1]))
```


Density
0.0
0.1
0.2
0.3
0.4
2
4
6
β1

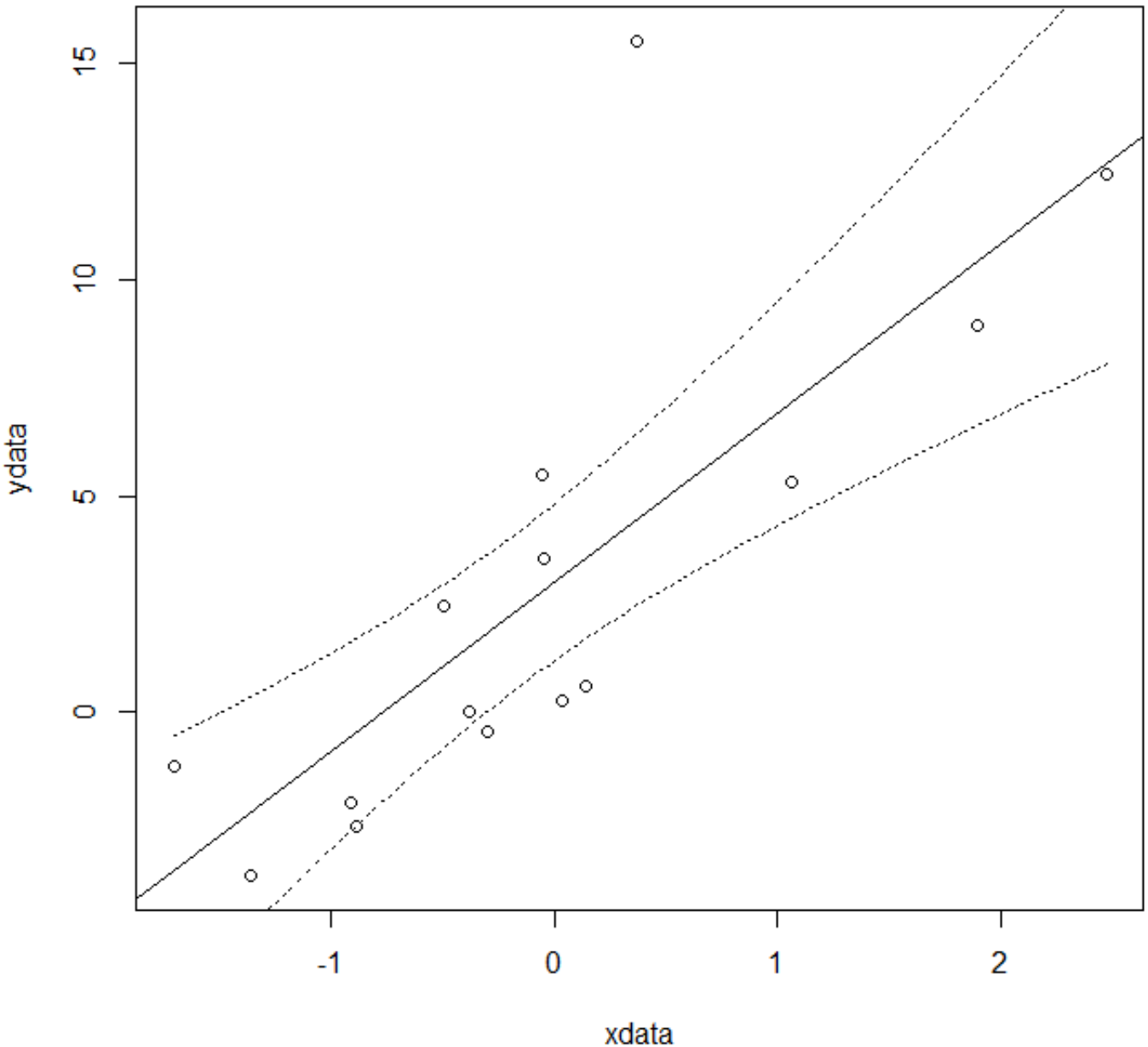
Regresión bootstrap con la librería simpleboot

```
library(simpleboot)

lmodel = lm(y ~ x)
# Bootstrap con residuos
lboot2 = lm.boot(lmodel, R=1000, rows=FALSE)
summary(lboot2)
```

```
BOOTSTRAP OF LINEAR MODEL  (method = residuals)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)             x
      2.993         3.905

Bootstrap SD's:
(Intercept)             x
  0.9092530     0.8108823
```

Regresión bootstrap con la librería simpleboot

```
# Grafico de los datos y de la recta de regresion
# original junto con

# + 1.96 veces el error estandar bootstrap
# - 1.96 veces el error estandar bootstrap

plot(lboot2)
```


ydata
xdata
15
10
5
0
-1
0
1
2

Bootstrap en regresión basado en pares de valores

- Hay otro método alternativo para aplicar el bootstrap en regresión, que es remuestreando las parejas de valores $\mathbf{x}_i = (\mathbf{c}_i, y_i)$
- De este modo, una muestra bootstrap consiste en

$$x^* = \{(\mathbf{c}_{i_1}, y_{i_1}), (\mathbf{c}_{i_2}, y_{i_2}), \ldots, (\mathbf{c}_{i_n}, y_{i_n})\}$$

  para $i_1, i_2, \ldots, i_n$, que es una muestra aleatoria de números enteros entre 1 y n.

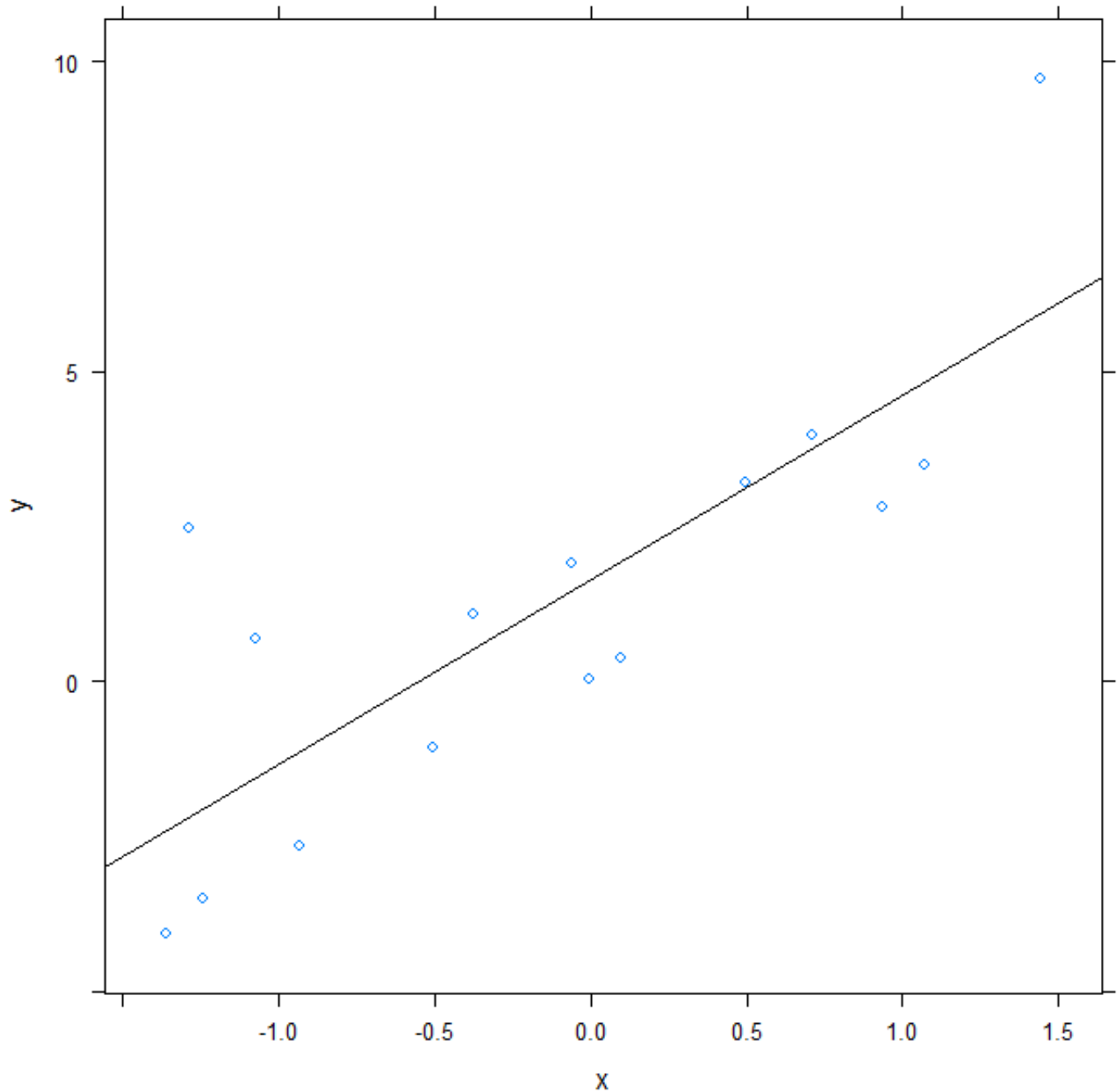
- ¿Qué método es mejor, el que remuestrea residuos o el que remuestrea parejas?

Bootstrap en regresión basado en pares de valores

- La respuesta depende de cómo se considere el modelo de regresión.
- Si en el modelo se asume que el error correspondiente a la diferencia entre y_i y la media $\mu_i = c_i\beta$ no depende de $\mathbf{c}_i$, esto implica que tiene la misma distribución F sin importar cuál sea el valor de $\mathbf{c}_i$.
- El bootstrap con parejas es menos sensible a la suposición anterior y lo único que se requiere es que las parejas originales $\mathbf{x}_i = (\mathbf{c}_i, y_i)$ se remuestreen de manera aleatoria de una distribución F en los vectores $p + 1$ dimensionales $(\mathbf{c}, y)$.

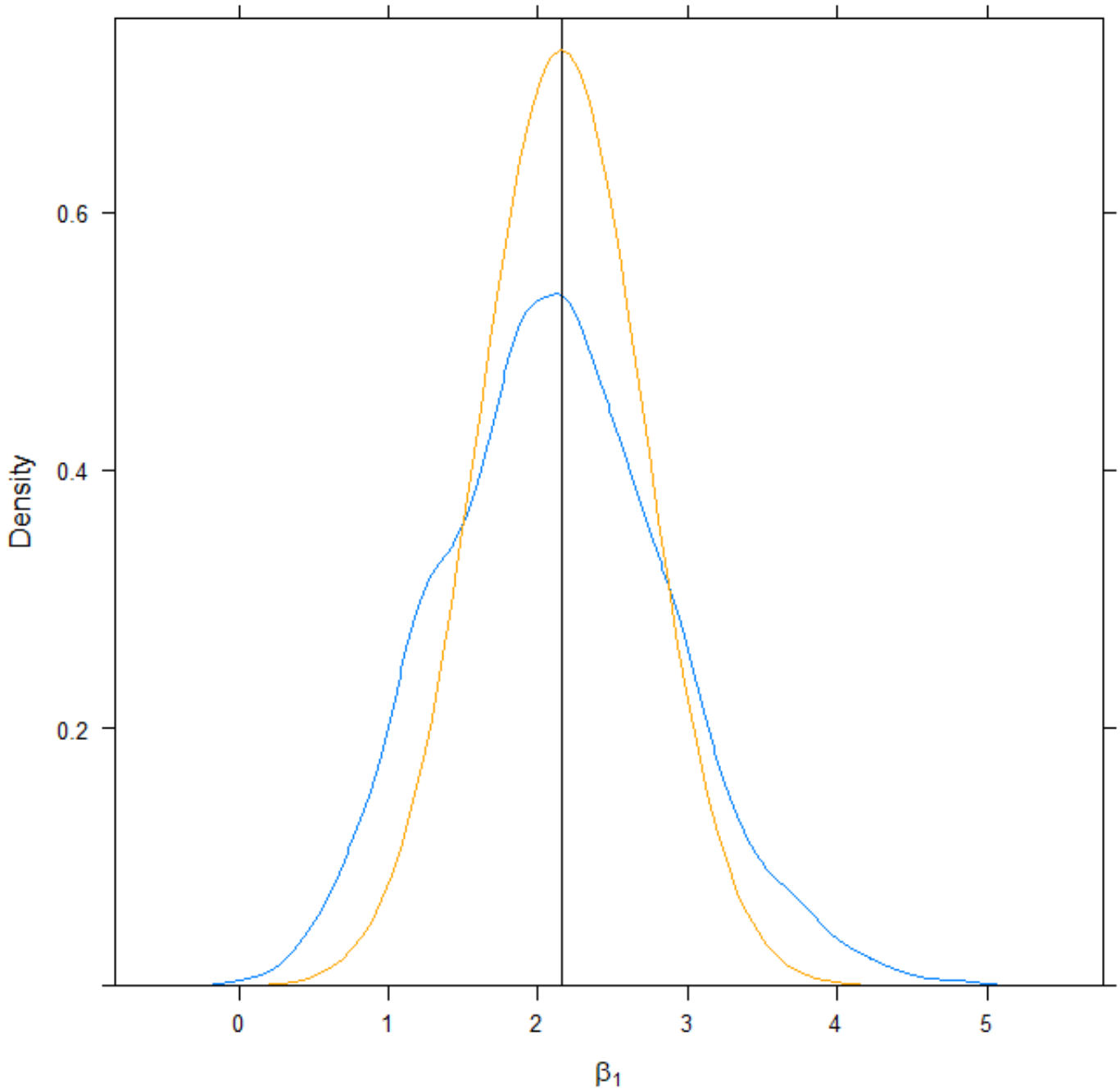
Bootstrap en regresión basado en pares de valores

```
N = 15
sd = 1.5
x = rnorm(N)
y = 3*x + sd*rnorm(N)^2
est = lm(y ~ x)

library(latticeExtra)
xyplot(y ~ x) + layer(panel.abline(est))
```


10
5
0
y
-1.0
-0.5
0.0
0.5
1.0
1.5
x

Bootstrap en regresión basado en pares de valores

```
bootPair = replicate(5000,{
ind = sample(1:N, replace=TRUE);
coef(lm(y[ind] ~ x[ind]))[2]
})

# est tiene los valores de la recta de regresion original
betaEst = coef(est)[2]
sdBeta = sqrt(vcov(est)[2, 2])

# Grafico de la distribucion del estadistico beta original
# y el estadistico beta remuestreado

densityplot(bootPair, plot.points=FALSE,ylim=c(0,.75),
 xlab=expression(beta[1])) +
layer(panel.abline(v=betaEst))+
layer(panel.mathdensity(args=list(mean=betaEst,sd=sdBeta),
col="orange", n=100))
```


Density
0.6
0.4
0.2
0
1
2
3
4
5
β1

Bootstrap en regresión basado en pares de valores

Si se compara el valor del error estándar bootstrap $\hat{\sigma}_\beta$ con respecto al error estándar del modelo de regresión original:

```
sd(bootPair)
```

```
[1] 0.7883879
```

```
sqrt(vcov(est)[2,2])
```

```
[1] 0.8749494
```

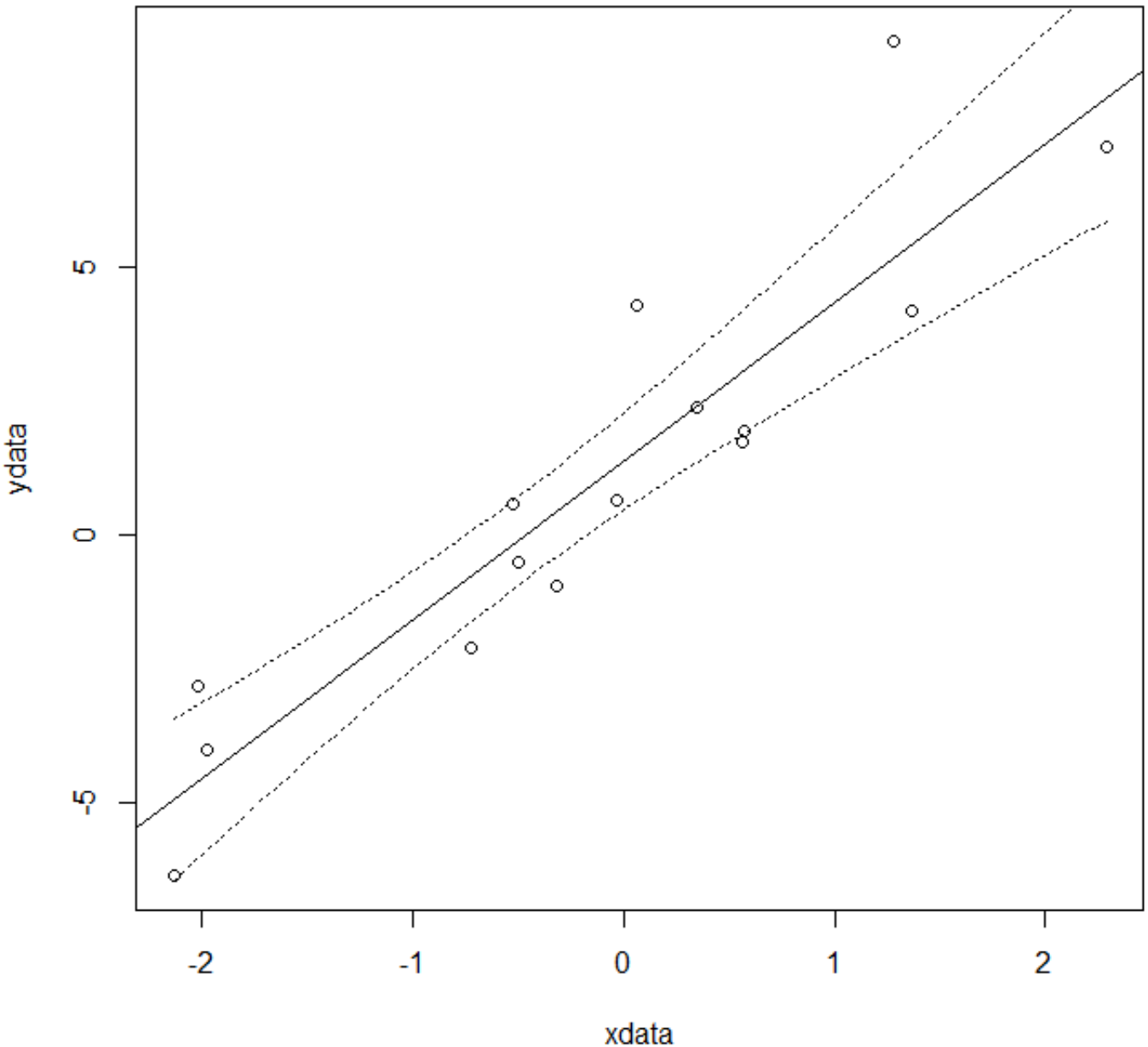
Regresión bootstrap con la librería simpleboot

```
N = 15
sd = 1.5
x = rnorm(N)
y = 3*x + sd*rnorm(N)^2

library(simpleboot)
lmodel = lm(y ~ x)

lboot2 = lm.boot(lmodel, R=1000)
summary(lboot2)
```

Regresión bootstrap con la librería simpleboot

```
BOOTSTRAP OF LINEAR MODEL  (method = rows)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      1.336        2.421

Bootstrap SD's:
(Intercept)            x
  0.6074143    0.5885158
```

```
# Grafico de la recta de regresion con las bandas
# +/- 1.96 veces el error estandar bootstrap

plot(lboot2)
```


ydata
5
0
-5
-2
-1
0
1
2
xdata

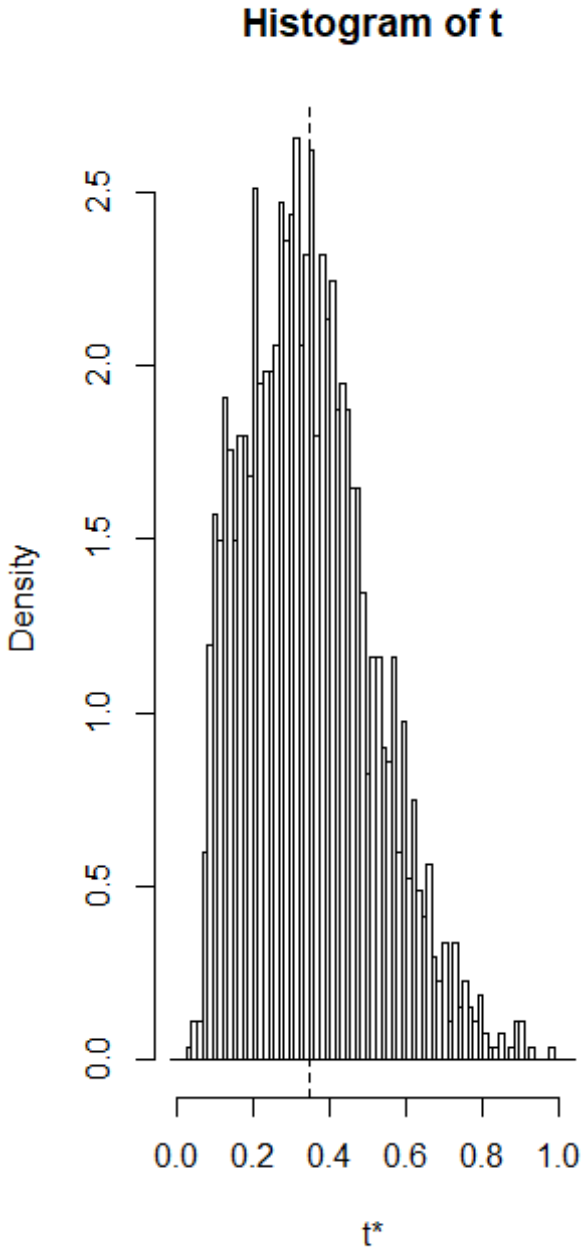
Regresión bootstrap con la librería boot

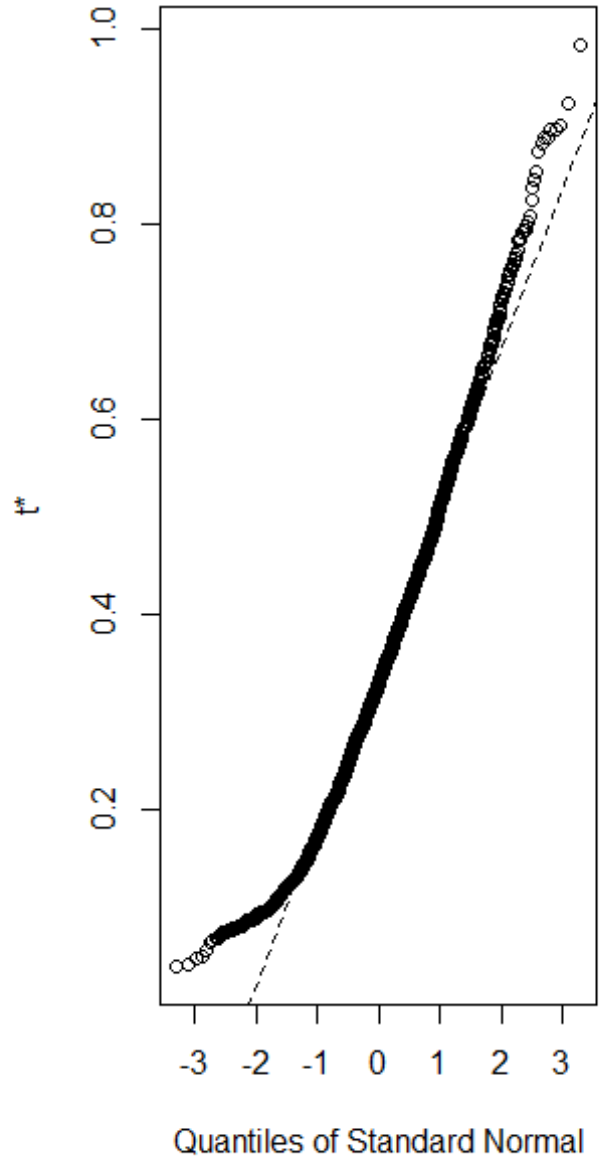
```
N = 15
sd = 0.5
x = rnorm(N)
y = 10*x + sd*rnorm(N)^2
datos = data.frame(y,x)

# Regresion basada en parejas o filas
boot.reg = function(data, i){
   mod = lm(y ~ x, data=data[i, ])
   coef(mod)
}
```

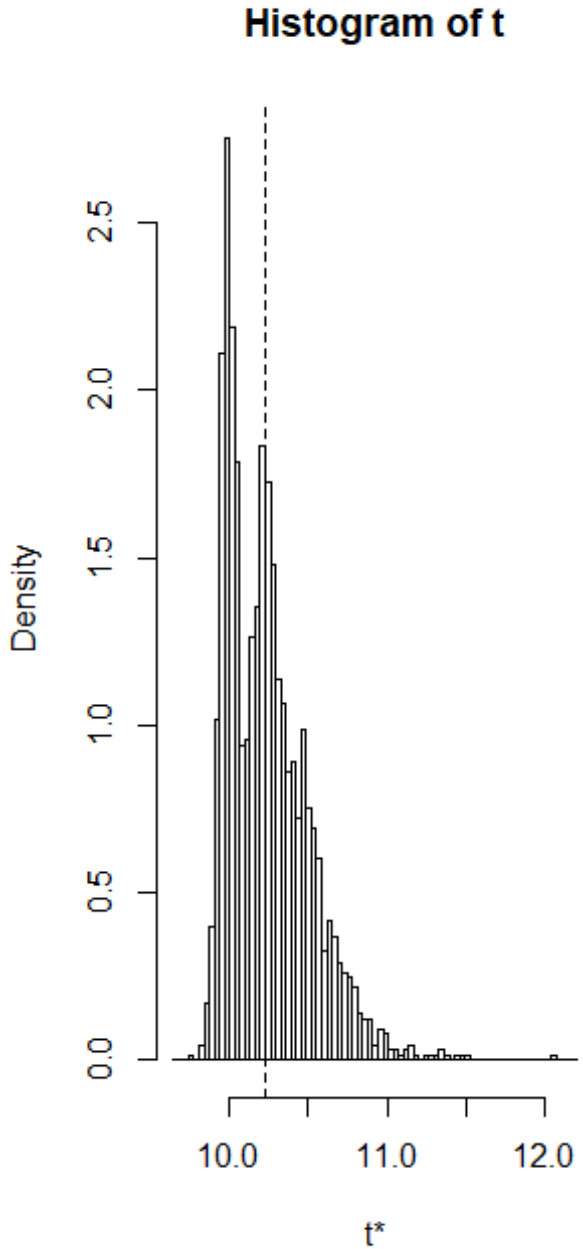
Regresión bootstrap con la librería boot

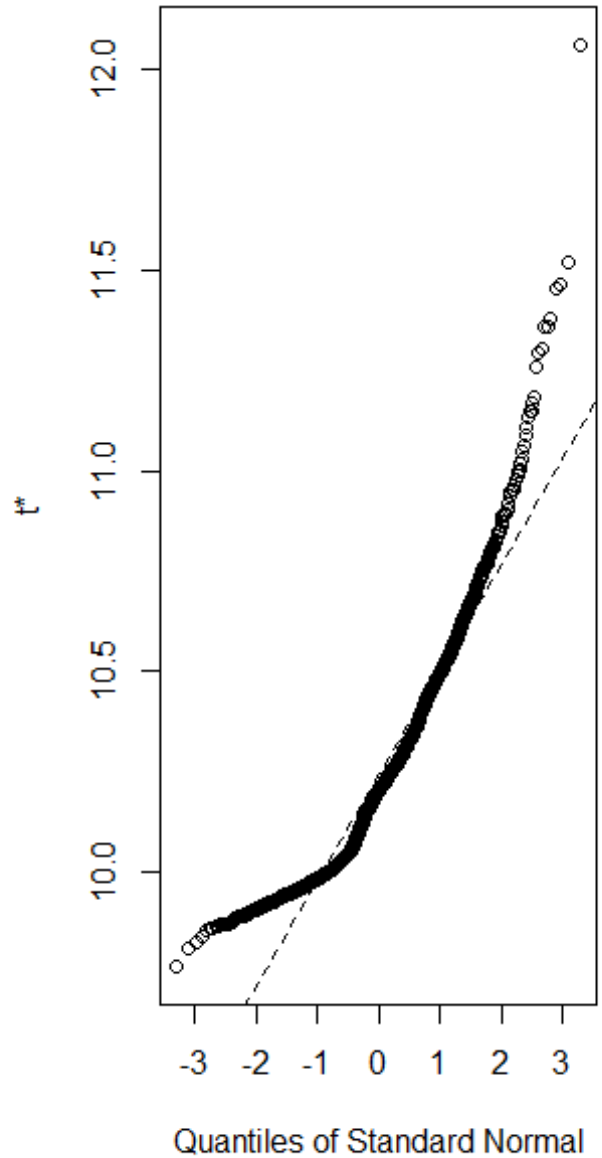
```
library(boot)

boot.1 = boot(data=datos, statistic=boot.reg, R=2000)
boot.1

plot(boot.1, index=1)
plot(boot.1, index=2)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = datos, statistic = boot.reg, R = 2000)

Bootstrap Statistics :
      original        bias    std. error
t1*  0.3473924 -0.002623727   0.1637179
t2* 10.2273750  0.010208366   0.2636508
```


Histogram of t
Density
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
t*


t*
0.2
0.4
0.6
0.8
1.0
-3
-2
-1
0
1
2
3
Quantiles of Standard Normal


Histogram of t
Density
0.0
0.5
1.0
1.5
2.0
2.5
10.0
11.0
12.0
t*


t*
10.0
10.5
11.0
11.5
12.0
-3
-2
-1
0
1
2
3
Quantiles of Standard Normal

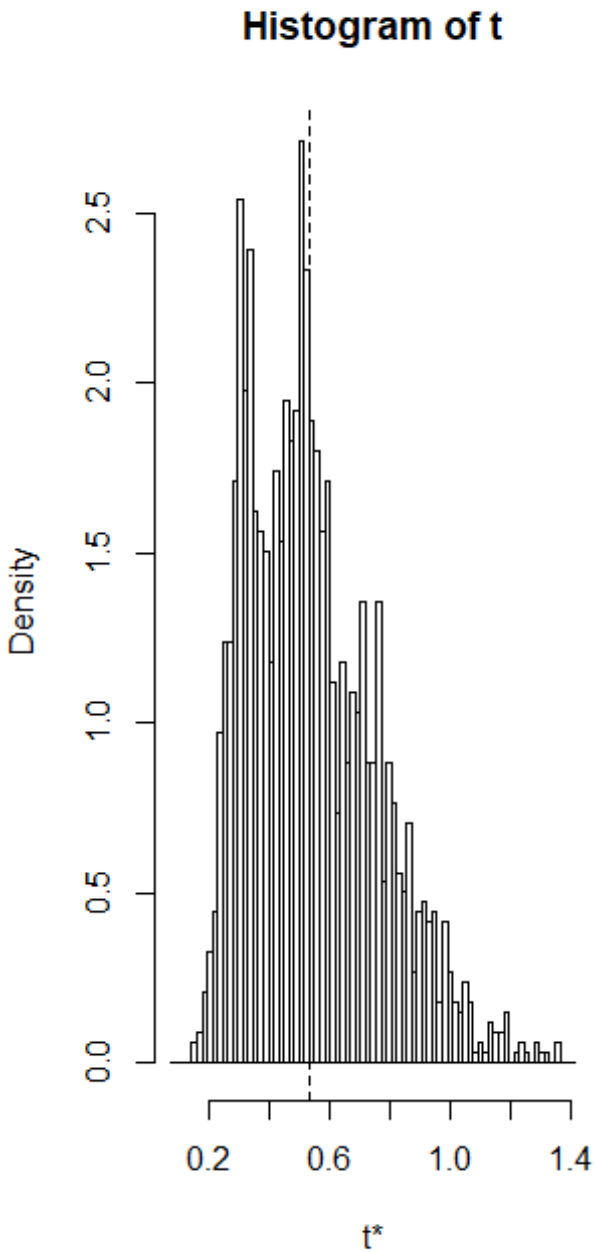
Regresión bootstrap con la librería boot

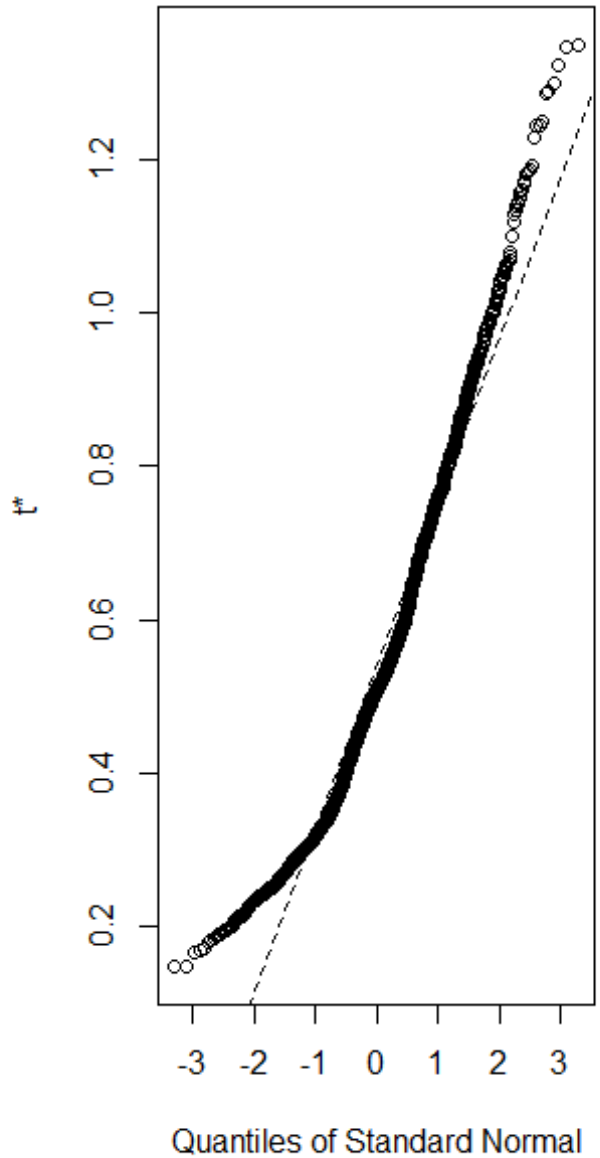
```
# Regresion basada en residuos
boot.reg2 = function(losdatos, i){
  modelo = lm(y ~ x, data=losdatos)
  yhat = fitted(modelo)
  e = resid(modelo)
  y.star = yhat + e[i]
  modelB = lm(y.star ~ x)
  coef(modelB)
}
```

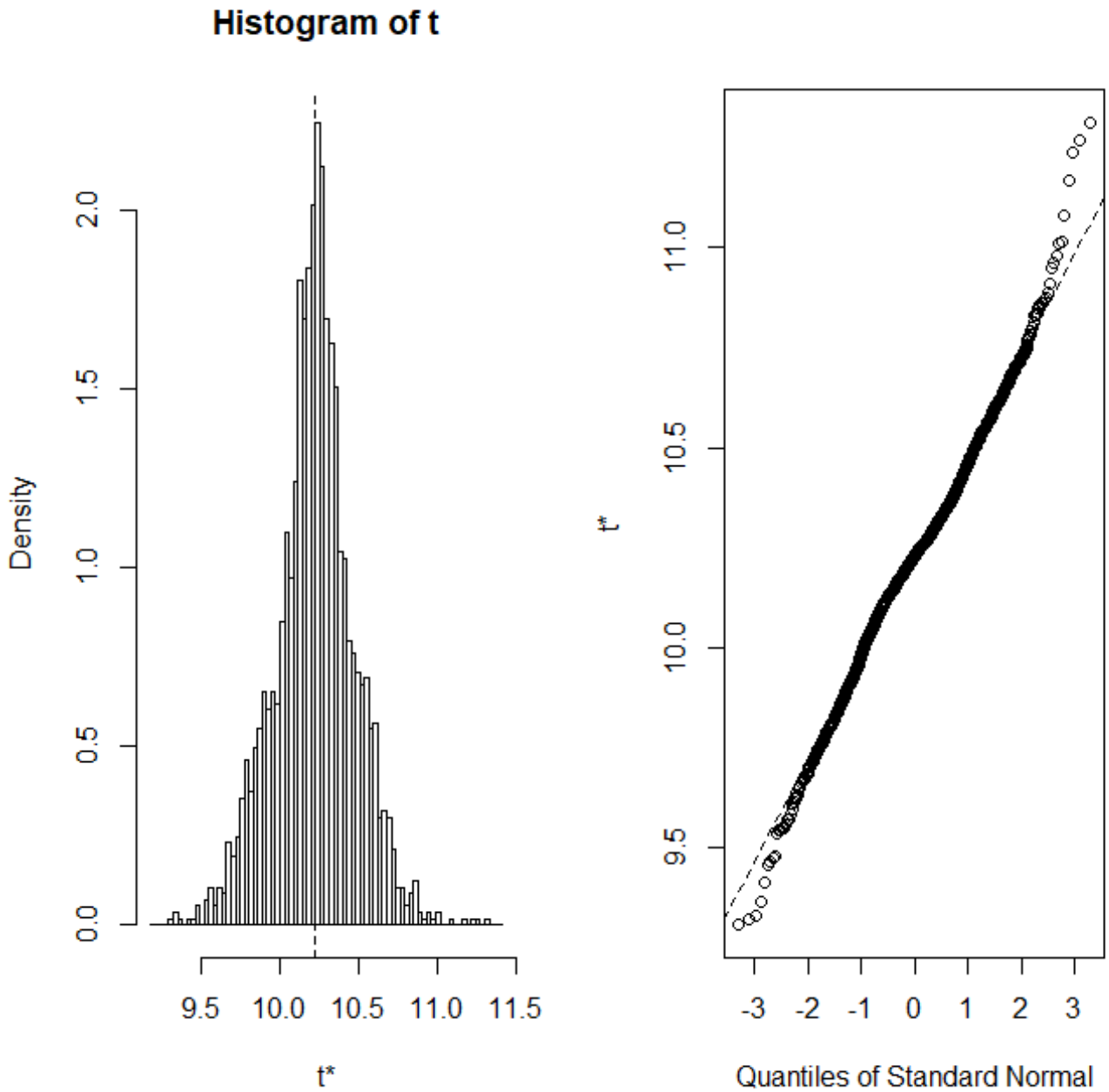
Regresión bootstrap con la librería boot

```
boot.2 = boot(data=datos, statistic=boot.reg2, R=2000)
boot.2

plot(boot.2, index=1)
plot(boot.2, index=2)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = datos, statistic = boot.reg2, R = 2000)

Bootstrap Statistics :
      original        bias    std. error
t1*  0.5333692 0.0037492018   0.2129779
t2* 10.0623031 0.0007129706   0.1578086
```


Histogram of t
Density
0.0
0.5
1.0
1.5
2.0
2.5
0.2
0.6
1.0
1.4
t*


t*
0.2
0.4
0.6
0.8
1.0
1.2
-3
-2
-1
0
1
2
3
Quantiles of Standard Normal


Histogram of t
Density
0.0
0.5
1.0
1.5
2.0
9.5
10.0
10.5
11.0
11.5
t*
t*
9.5
10.0
10.5
11.0
-3
-2
-1
0
1
2
3
Quantiles of Standard Normal

ANOVA unifactorial con Bootstrap

- Una manera cómoda de aplicar bootstrap en técnicas ANOVA es mediante la librería `WRS2`.
- Esta librería permite trabajar también con *medias recortadas* (trimming means) y funciona bien en el caso de heterocedasticidad y falta de normalidad.

```
library(WRS2)
help(viagra)

# Se aplica un ANOVA unifactorial asumiendo normalidad
summary(aov(libido ~ dose, data=viagra))
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
dose          2  20.13  10.067   5.119 0.0247 *
Residuals    12  23.60   1.967
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA unifactorial con Bootstrap

- Se aplica un remuestreo bootstrap

```
t1waybt(libido ~ dose, tr=0, nboot=1000, data=viagra)
```

```
Test statistic: 4.3205
p-value: 0.06886
Variance explained 0.645
Effect size 0.803
```

- Se puede aplicar también un análisis *post-hoc* por parejas de categorías.

```
mcppb20(libido ~ dose, tr=0, nboot=1000,
data=viagra, crit=0.05)
```

```
                psihat ci.lower ci.upper p-value
placebo vs. low   -1.0     -2.2       0.4   0.222
placebo vs. high  -2.8     -4.2      -1.4   0.000
low vs. high      -1.8     -3.2      -0.6   0.035
```

ANOVA bifactorial con Bootstrap

- Se puede considerar también un ANOVA bifactorial

```
help(goggles)
t2way(attractiveness ~ gender*alcohol,
data=goggles, tr=0)
```

```
                  value p.value
gender           2.0323   0.164
alcohol         40.0983   0.001
gender:alcohol  24.4083   0.001
```

ANOVA bifactorial con Bootstrap

- Se puede usar también un análisis *post-hoc*

```
mcp2atm(attractiveness ~ gender*alcohol,
data=goggles, tr=0)
```

```
                   psihat  ci.lower  ci.upper p-value
gender1            11.250  -4.82883  27.32883 0.16374
alcohol1           -1.875 -18.53329  14.78329 0.77361
alcohol2           34.375  18.65382  50.09618 0.00001
alcohol3           36.250  18.82376  53.67624 0.00002
gender1:alcohol1   -1.875 -18.53329  14.78329 0.77361
gender1:alcohol2 -28.125 -43.84618 -12.40382 0.00014
gender1:alcohol3 -26.250 -43.67624  -8.82376 0.00081
```

ANOVA unifactorial con Bootstrap basado en residuos

- Otra alternativa es aplicar el bootstrap basado en modelos con la librería `boot`.

```
# Se generan unos datos artificiales
Nj = c(41, 37, 42, 40)
Ntot = sum(Nj)
muJ = rep(c(-1, 0, 1, 2), Nj)
MisDatos = data.frame(IV=factor(rep(LETTERS[1:4],Nj)),
DV = rnorm(Ntot, muJ, 6))
head(MisDatos)
```

```
  IV          DV
1  A  5.6002327
2  A  6.6621069
3  A -3.6699250
4  A  1.1622087
5  A -0.9911518
6  A  3.9720186
```

ANOVA unifactorial con Bootstrap basado en residuos

```
with(MisDatos, tapply(DV, IV, mean))
```

```
         A          B          C          D
-0.993078  1.242881  1.175913  1.120702
```

```
with(MisDatos, tapply(DV, IV, var))
```

```
       A         B         C         D
40.75370 30.33736 44.86708 42.91103
```

```
with(MisDatos,tapply(DV,IV,length))
```

```
 A  B  C  D
41 37 42 40
```

ANOVA unifactorial con Bootstrap basado en residuos

- Un ANOVA clásico obtiene

```
(anoriginal = anova(lm(DV ~ IV, data=MisDatos)))
```

```
Analysis of Variance Table

Response: DV
           Df Sum Sq Mean Sq F value Pr(>F)
IV          3  235.0  78.342  2.0802 0.1051
Residuals 156 5875.2  37.662
```

```
(Fbase = anoriginal["IV", "F value"])
```

```
[1] 2.0802
```

```
(pBase = anoriginal["IV", "Pr(>F)"])
```

```
[1] 0.1050664
```

ANOVA unifactorial con Bootstrap basado en residuos

- Aplicando la librería boot:

```
mediaglobal = mean(MisDatos$DV)
E = MisDatos$DV - mediaglobal     ## residuos

Boot.Anova = function(dat, i) {
    media.star = mediaglobal + E[i]
    anBS = anova(lm(media.star ~ IV, data=dat))
    return(anBS["IV", "F value"])
}

library(boot)
booAnova = boot(MisDatos, statistic=Boot.Anova,
R=1000)
booAnova
```

ANOVA unifactorial con Bootstrap basado en residuos

- Se obtiene

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = MisDatos, statistic = Boot.Anova, R = 1000)

Bootstrap Statistics :
    original     bias    std. error
t1* 2.080173 -1.075454   0.8386613
```

```
Fstar = booAnova$t
Fmayor = (Fstar > Fbase)
# P-valor remuestreado
(pValBS = (sum(Fmayor) / length(Fmayor)))
```

```
[1] 0.099
```

ANOVA unifactorial con Bootstrap no paramétrico

Alternativamente se puede remuestrear de manera directa:

```
meanstar = with(MisDatos, tapply(DV,IV,mean))
cuentas = with(MisDatos, tapply(DV,IV,length))

grpA = MisDatos$DV[MisDatos$IV=="A"] - meanstar[1]
grpB = MisDatos$DV[MisDatos$IV=="B"] - meanstar[2]
grpC = MisDatos$DV[MisDatos$IV=="C"] - meanstar[3]
grpD = MisDatos$DV[MisDatos$IV=="D"] - meanstar[4]

simIV = MisDatos$IV
R = 1000
```

ANOVA unifactorial con Bootstrap no paramétrico

```
Fstar = numeric(R)

# Tenemos una distribucion F bootstrapeada en "Fstar"
# basada en medias de grupos iguales (la hipotesis nula),
# pero no se asumen normalidad ni homogeneidad

for (i in 1:R) {
  groupA = sample(grpA, size=cuentas[1], replace=T)
  groupB = sample(grpB, size=cuentas[2], replace=T)
  groupC = sample(grpC, size=cuentas[3], replace=T)
  groupD = sample(grpD, size=cuentas[4], replace=T)

  simDV = c(groupA,groupB,groupC,groupD)
  simdata = data.frame(simDV,simIV)
  Fstar[i] = oneway.test(simDV~simIV,
             data=simdata)$statistic
}
```

ANOVA unifactorial con Bootstrap no paramétrico

```
quantile(Fstar,.95)
```

```
     95%
2.328775
```

```
Fbase = anoriginal["IV", "F value"]     # anoriginal[1,5]
Fmayor = (Fstar > Fbase)

# P-valor remuestreado
pValBS = (sum(Fmayor) / length(Fmayor))
pValBS
```

```
[1] 0.08
```

Aplicación del bootstrap a series temporales

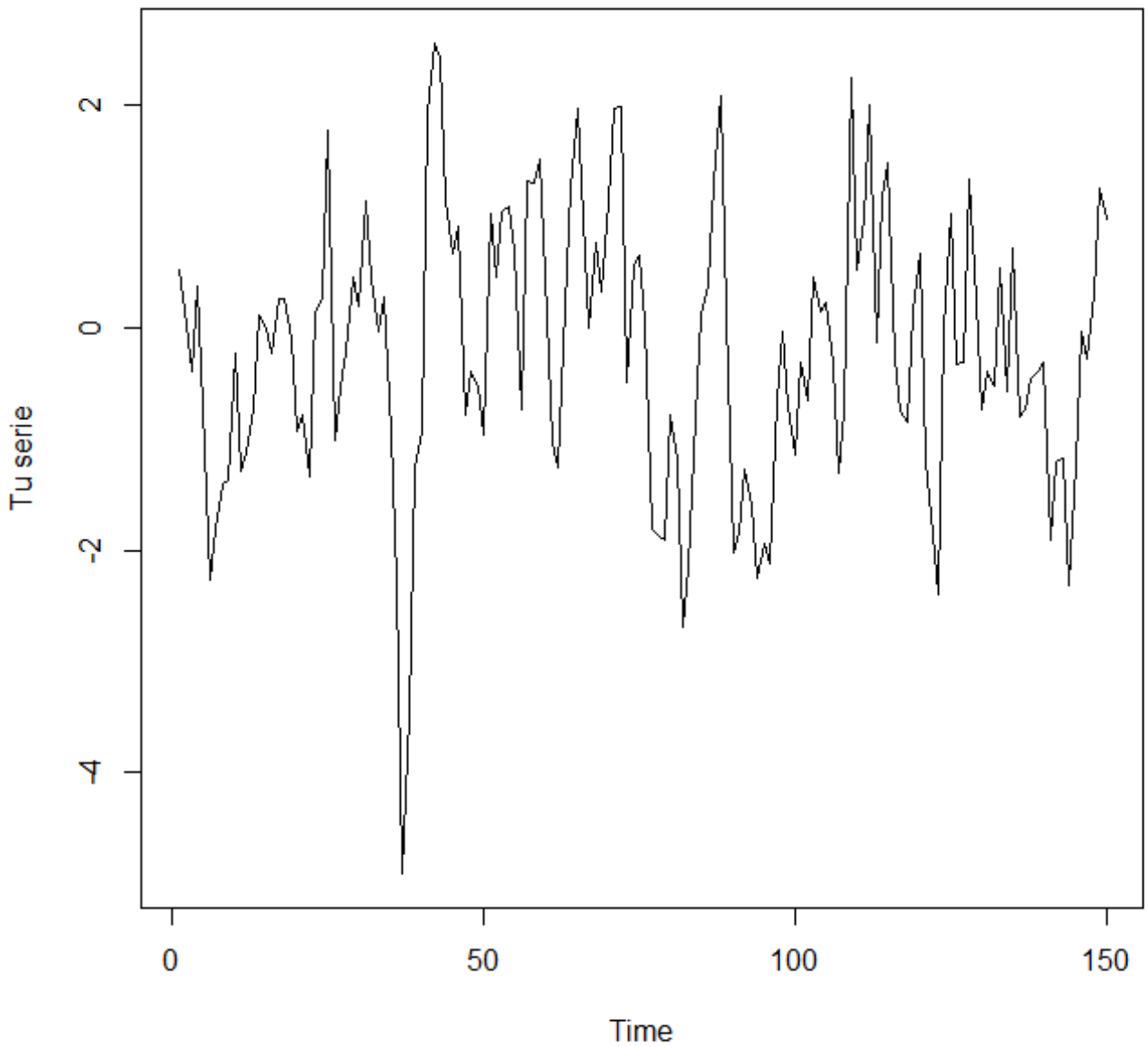
- Se considera un modelo de serie AR(1) simple

$$y_t = \beta y_{t-1} + \varepsilon_t$$

donde $\varepsilon_t \sim N(0,1)$.

- Se simulan unos datos

```
N = 150
epsilon = rnorm(N)
y = epsilon
for(i in 2: N){
      y[i] = y[i-1]*0.7 + epsilon[i]
   }

plot.ts(ts(y), t="l", ylab="Tu serie")
```


Tu serie
2
0
-2
-4
0
50
100
150
Time

Aplicación del bootstrap a series temporales

O bien usando el comando de R

```
arima.sim(n=N, list(ar=0.7), innov=epsilon, n.start=1,
start.innov=0)
```

```
Time Series:
Start = 1
End = 150
Frequency = 1
  [1] -1.65475863 -0.38318004  0.63597925  0.81702209
  0.99204425  2.30593597  ........
```

Aplicación del bootstrap a series temporales

Se puede estimar β por máxima verosimilitud, usando el comando `arima`

```
(est.arima = arima(y, order=c(1,0,0), include.mean=FALSE))
```

```
Call:
arima(x = y, order = c(1, 0, 0), include.mean = FALSE)

Coefficients:
         ar1
      0.6910
s.e.  0.0588

sigma^2 estimated as 0.8743:  log likelihood = -203.09,
aic = 410.19
```

Aplicación del bootstrap a series temporales

¿Cuáles son los métodos que se podrían aplicar en este caso?

- **Bootstrap de parejas de puntos**: Aquí **NO** se puede hacer porque se rompe la estructura de la serie temporal.
- **Bootstrap de residuos**: se preserva la estructura original de la serie cuando se asume la estrutura de dependencia entre los residuos.
- **Bootstrap de mediante bloques móviles (*moving blocks*)**: se preserva la estructura original de la serie.

Bootstrap en series temporales con residuos

- Para construir la muestra bootstrap se usan los residuos estimados.

```
kk = residuals(est.arima)
betaB = coef(est.arima)

betaBoot = replicate(5000, {
 epsilon = sample(kk, size=N, replace=TRUE)
 eso = arima.sim(n=N, list(ar=betaB), innov=epsilon,
 n.start=1, start.innov=0)
 coef(arima(eso, order=c(1,0,0), include.mean=FALSE))
}
)
```

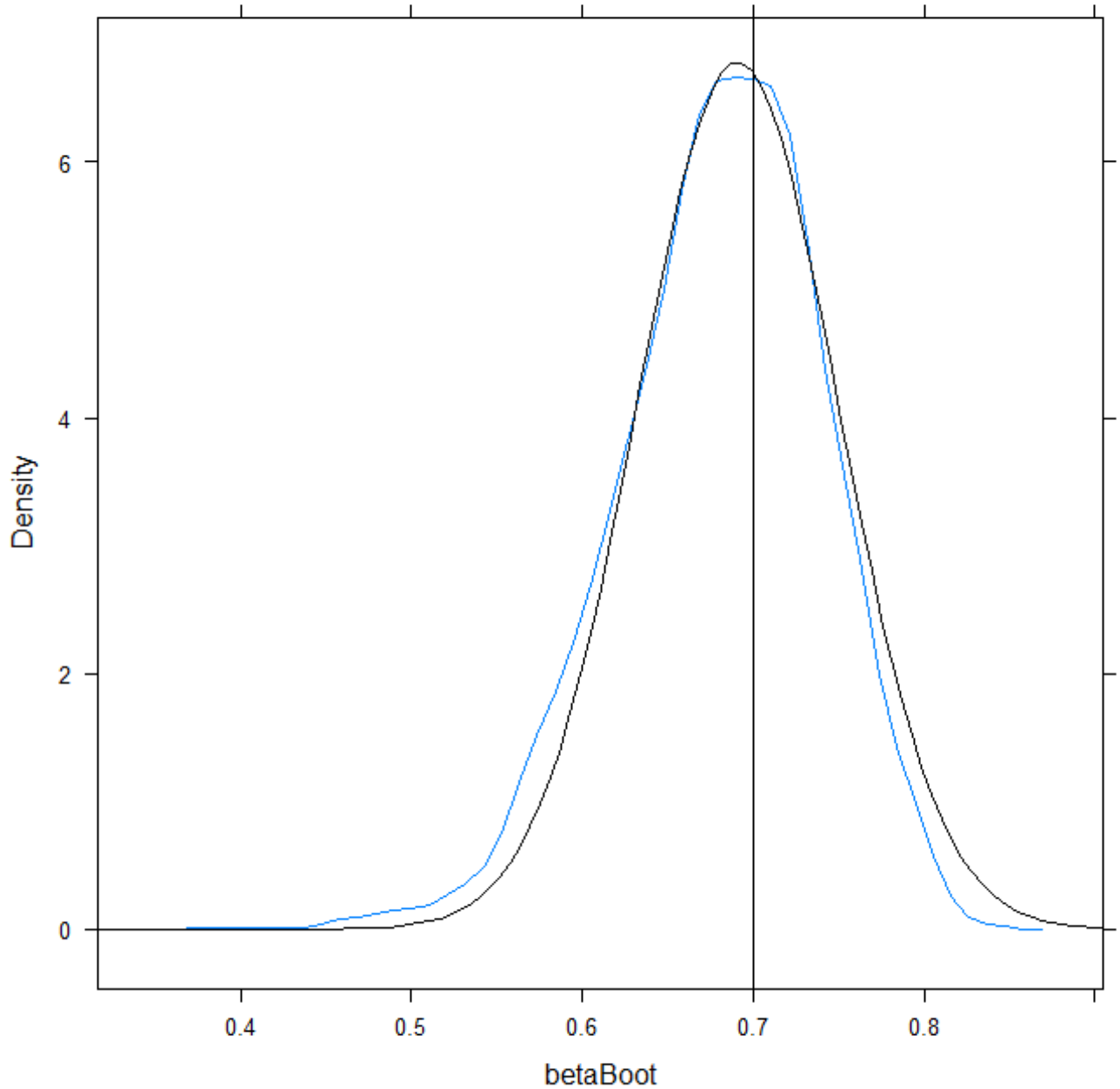
Bootstrap en series temporales con residuos

- Se compara el estimador bootstrap del error estándar, con el obtenido de la sere original mediante *EMV*:

```
c(sd(betaBoot), sqrt(vcov(est.arima)))
```

```
[1] 0.05965961 0.05882693
```

```
library(latticeExtra)

sdBeta = sqrt(vcov(est.arima))
densityplot(~betaBoot, plot.points=FALSE) +
layer(panel.abline(v=0.7)) +
layer(panel.mathdensity(args=list(mean=betaB,
sd=sdBeta), col="black", n=100))
```


Density
0
2
4
6
0.4
0.5
0.6
0.7
0.8
betaBoot

Bootstrap en series temporales con residuos

- Supongamos ahora que se supone de manera errónea que el proceso es un AR(2)

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \varepsilon_t$$

- Se estiman entonces los parámetros, suponiendo que es un AR(2).

```
est2 = arima(y, order=c(2,0,0), include.mean=FALSE)
est2
```

```
Coefficients:
          ar1      ar2
      0.5622  0.0984
s.e.  0.0809  0.0823

sigma^2 estimated as 0.9247: log likelihood=-207.23,
aic = 420.45
```

Bootstrap en series temporales con residuos

- Se puede estudiar la precisión de β_2

```
kk = residuals(est2)
(betaB = coef(est2))
```

```
       ar1        ar2
0.56215491 0.09843461
```

```
betaBoot2 = replicate(5000,{
epsilon = sample(kk ,N, replace=TRUE)
eso = arima.sim(n=N, list(ar=betaB), innov=epsilon)
coef(arima(eso, order=c(2,0,0), include.mean=FALSE))}
)
```

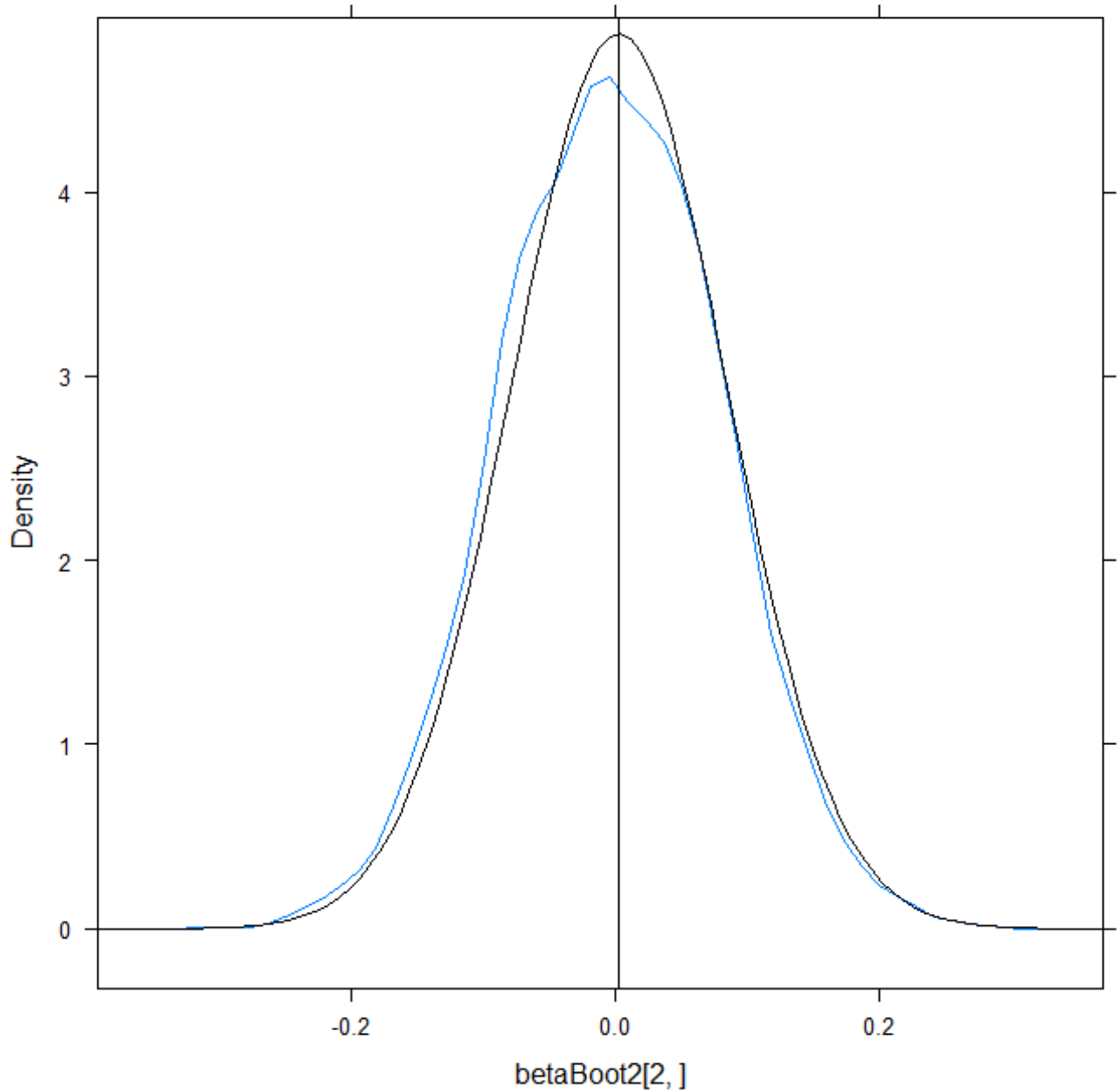
Bootstrap en series temporales con residuos

- Se puede estudiar y comparar también la desviación estandar de β_2

```
c(sd(betaBoot2[2,]), sqrt(vcov(est2.arima)[2,2]))
```

```
[1] 0.08139569 0.08225318
```

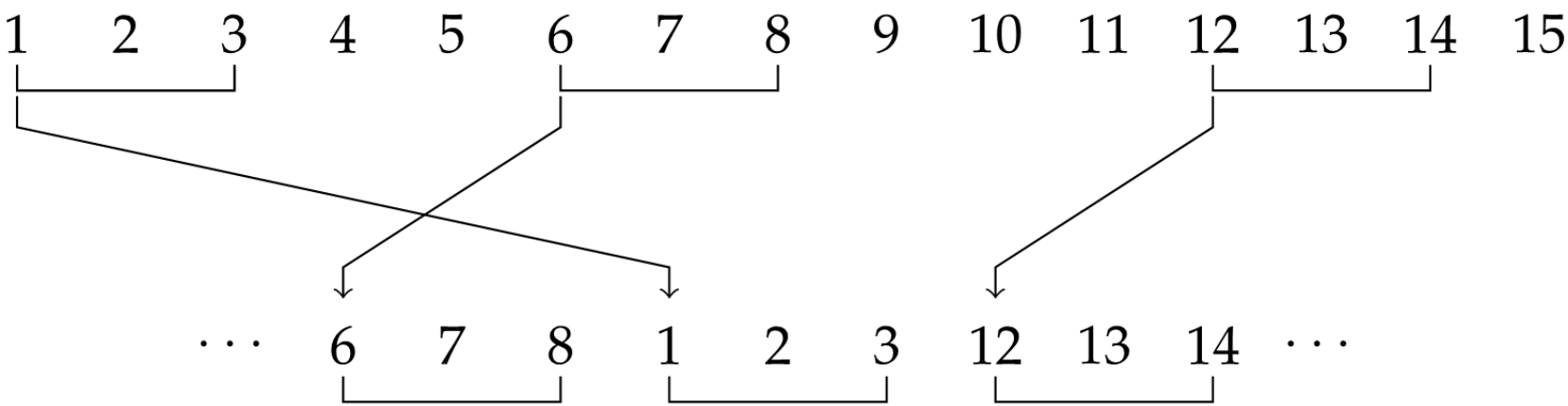
```
seB = sqrt(diag(vcov(est2.arima)))

library(latticeExtra)

densityplot(betaBoot2[2,],plot.points=FALSE)+
layer(panel.abline(v=betaB[2]))+
layer(panel.mathdensity(args=list(mean=betaB[2],
sd=seB[2]),col="black",n=100))
```


Density
0
1
2
3
4
-0.2
0.0
0.2
betaBoot2[2,]

Bloques móviles (*moving blocks*)

- En el esquema del bootstrap mediante análisis de residuos se asume que se ***sabe*** cuál es el proceso que genera los datos.
- Pero, en el esquema de bloques móviles se asume solo que un bloque de datos corto tiene un patrón de comportamiento semejante.
- Por ejemplo

```
N = 150
blockLen = 5
blockNum = N/blockLen
```


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
· · · 6 7 8 1 2 3 12 13 14 · · ·

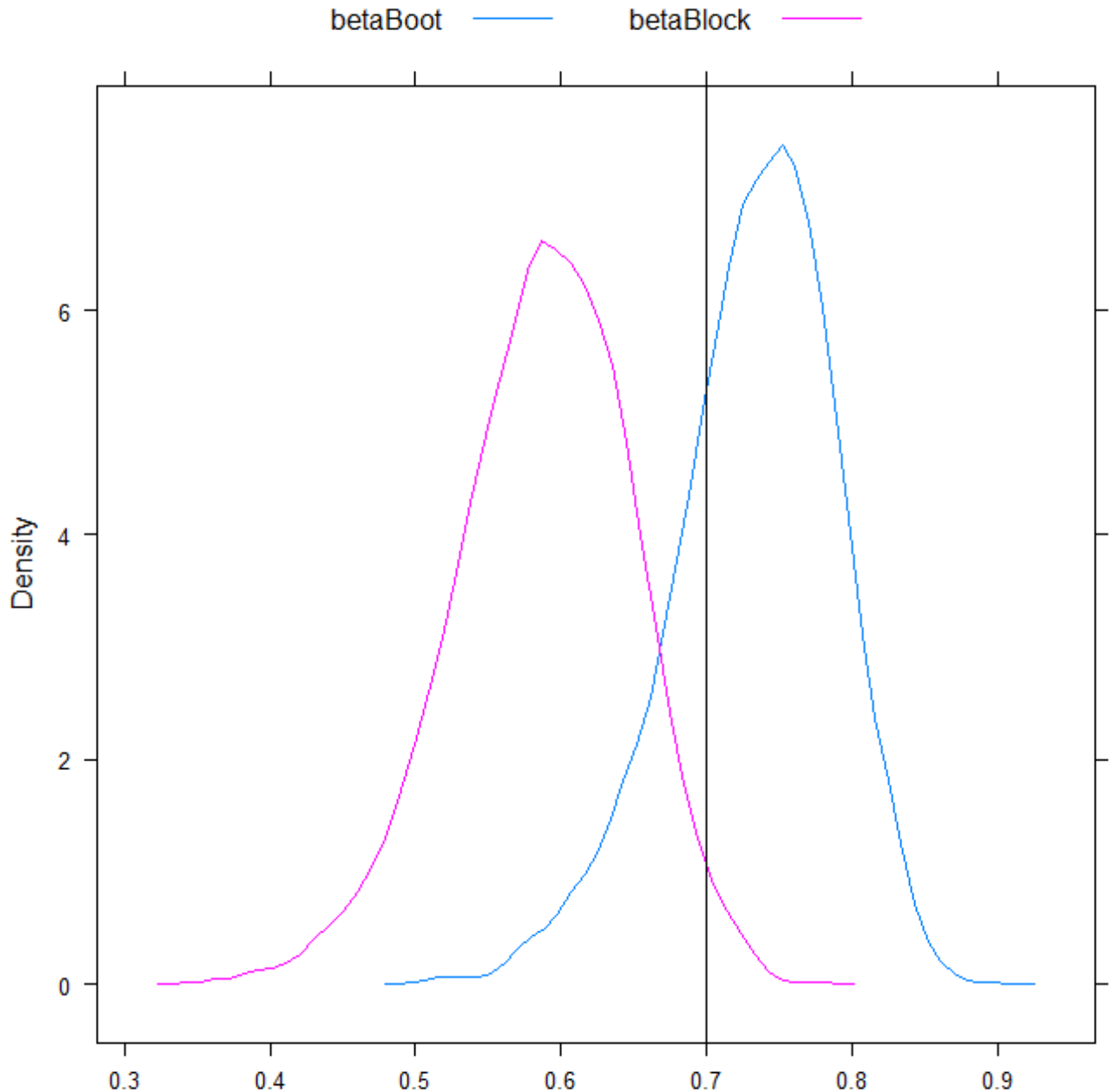
Bloques móviles (*moving blocks*)

```
betaBlock = replicate(5000,{
start = sample(1:(N-blockLen+1),
size=blockNum, replace=TRUE);
blockedIndices =
c(sapply(start, function(x) seq(x,x+blockLen-1)))
eso = y[blockedIndices]
coef(arima(eso, order=c(1,0,0), include.mean=FALSE))
})
```

```
c(sd(betaBlock), sqrt(vcov(est.arima)))
```

```
[1] 0.06217682 0.05449991
```

```
densityplot(~ betaBoot + betaBlock, xlab="",
plot.points=FALSE, auto.key=list(columns=2)) +
layer(panel.abline(v=0.7))
```


betaBoot
betaBlock
Density
0
2
4
6
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Bootstrap con tsboot

- Se puede usar el comando `tsboot` de la librería boot

```
library(boot)

N = 150
epsilon = rnorm(N)

# Simulas un AR(1)
y = arima.sim(n=N, list(ar=0.6), innov=epsilon,
n.start=1, start.innov=0)

bootf = function(miserie){
  fit = ar(miserie, order.max=1) # modelo AR(1)
  return(fit$ar)
}
```

Bootstrap con tsboot

- tsboot con bloques móviles.

```
# bootstrap por bloques cada uno con longitud 10
boot2 = tsboot(y, bootf, R=5000, l=10, sim="fixed")

teta.star = as.vector(boot2$t)
summary(teta.star)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1151  0.4533  0.5130  0.5049  0.5622  0.7179
```

```
# IC de percentil
quantile(teta.star, probs=c(0.025, 0.975))
```

```
     2.5%     97.5%
0.3242073 0.6427601
```

Bootstrap con tsboot

```
ar1 = ar(y, order.max=1)  # Ajustas un AR(1)
armodel = list(order=c(1,0,0), ar=ar1$ar)

bootf = function(miserie){
fit = ar(miserie, order.max=1) # modelo AR(1)
return(fit$ar)
}

bootsim = function(res, n.sim, argumentos){
  # generacion de series replicadas con arima.sim
  rg1 = function(n, res){ sample(res, n, replace=TRUE) }
  ts.orig = argumentos$ts
  ts.mod = argumentos$model
  return(mean(ts.orig) + ts(arima.sim(model=ts.mod,
  n=n.sim, rand.gen=rg1, res=as.vector(res))))
}
```

Bootstrap con tsboot

- tsboot con remuestreo de residuos

```
boot1= tsboot(y, bootf, R=1000, sim="model",
n.sim=length(y), orig.t=FALSE, ran.gen=bootsim,
ran.args=list(ts=y, model=armodel))

teta.star = as.vector(boot2$t)
summary(teta.star)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1151  0.4533  0.5130  0.5049  0.5622  0.7179
```

```
# IC de percentil
quantile(teta.star, probs=c(0.025,0.975))
```

```
     2.5%     97.5%
0.3242073 0.6427601
```

Modelos GLM con bootstrap

- Se pueden considerar modelos lineales generalizados

```
library(boot)
help(remission)

model.boot = function(data,indices){
    sub.data = data[indices,]
    model = glm(r ~ LI, family="binomial",
    data=sub.data)
    coef(model)
}

glm.boot = boot(remission, model.boot, R=2000)
glm.boot
```

Modelos GLM con bootstrap

- Se obtiene

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = remission, statistic = model.boot, R = 2000)

Bootstrap Statistics :
     original     bias    std. error
t1* -3.777140 -2.624717     24.69725
t2*  2.897264  2.670815     25.08782
```

Modelos GLM con bootstrap

- Los correspondientes intervalos de confianza son

```
boot.ci(glm.boot, index=1, type="bca")
```

```
Intervals :
Level       BCa
95%   (-8.601, -1.329)
Calculations and Intervals on Original Scale
```

```
boot.ci(glm.boot, index=2, type="bca")
```

```
Intervals :
Level       BCa
95%   (0.677,  8.908)
Calculations and Intervals on Original Scale
```