Tema 1. Introducción a los Métodos de Remuestreo

Bootstrap: Ejemplo sobre efectos de la aspirina

Se toman los datos originales

```
n1 = 11037 + 104      # tamaño muestra 1
s1 = 104              # numero de éxitos

n2 = 11034 + 189      # tamaño muestra 2
s2 = 189              # numero de éxitos
```

Sin embargo, no tenemos el fichero de datos *originales*, solo los **recuentos**.

Recreamos entonces el fichero original:

```
p1pre = c(rep(1,s1), rep(0,n1-s1))
p2pre = c(rep(1,s2), rep(0,n2-s2))

# Tomamos permutaciones de los datos anteriores
p1 = sample(p1pre, n1)   # muestra 1
p2 = sample(p2pre ,n2)   # muestra 2
```

Es decir, suponemos que $p1$ y $p2$ son los datos observados originales.

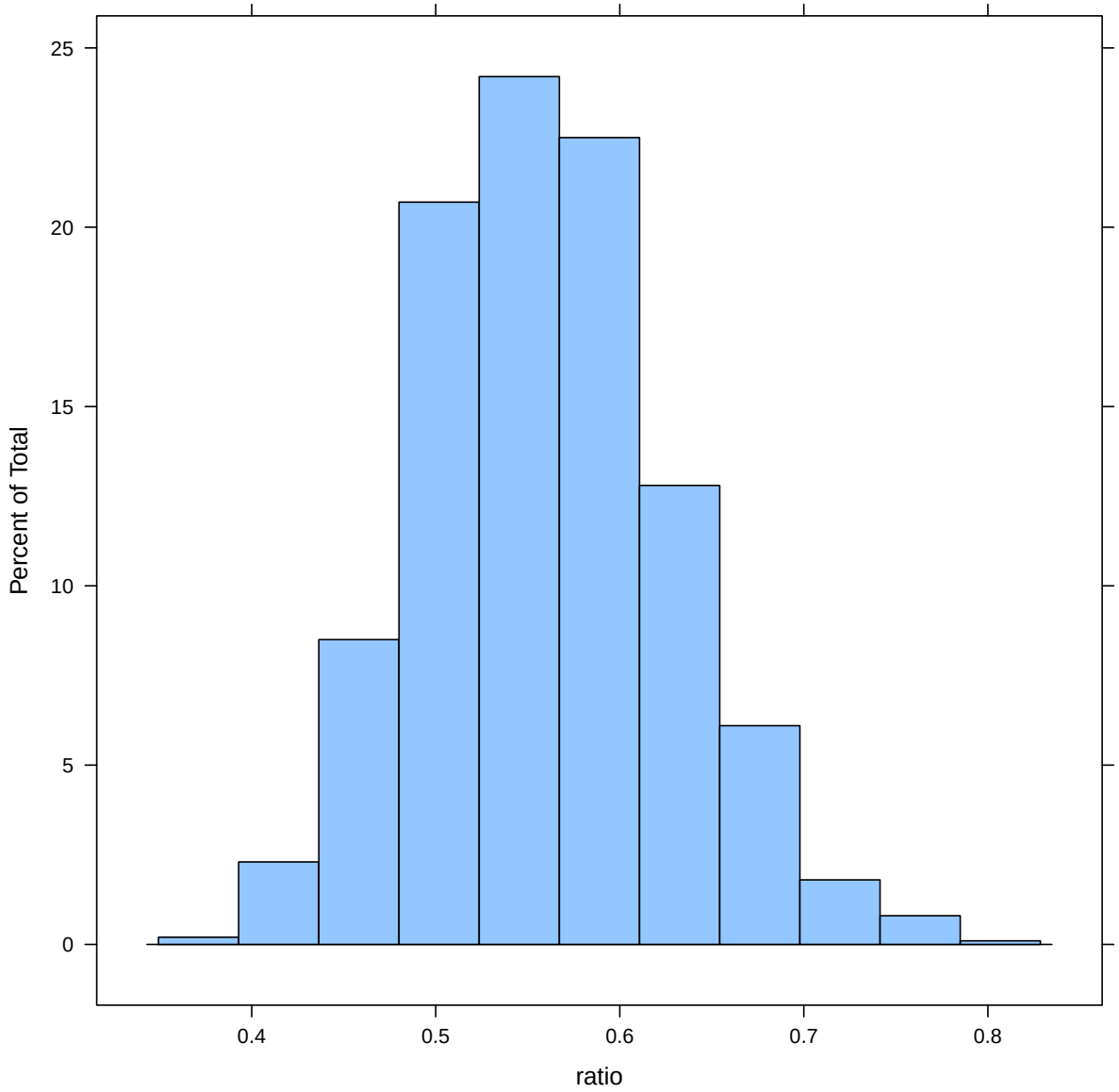
Se aplica un método de remuestreo bootstrap.

```
n.bs = 1000      # tomo n.bs muestras bootstrap

# Reservo dos vectores de ceros
bs1 = rep(0, n.bs)
bs2 = rep(0, n.bs)

for (i in 1:n.bs) {
   # Proporción de éxitos en muestras bootstrap 1 y 2
   bs1[i] = sum(sample(p1, n1, replace=TRUE))/n1
   bs2[i] = sum(sample(p2, n2, replace=TRUE))/n2
```

```
}

# Réplicas de la estimación bootstrap del ratio
ratio = bs1/bs2
```

Histograma de las estimaciones del ratio:

```
lattice::histogram(ratio)
```



```r
mean(ratio)
```

```
[1] 0.5600096
```

```r
median(ratio)
```

```
[1] 0.5572631
```

El intervalo de confianza bootstrap corresponde a los cuantiles del 0.025 y 0.975 de la muestra ordenada.

```r
quantile(ratio, probs=c(0.025, 0.975))
```

```
     2.5%     97.5%
0.4371584 0.6997495
```

O bien se ordenan las estimas del ratio de manera **manual** para obtener los intervalos de confianza bootstrap.

De esta manera, los extremos de los intervalos de confianza bootstrap son los cuantiles del 0.025 y 0.975 de la muestra ordenada.

```r
rats = sort(ratio)

CI.bs = c(rats[round(0.025*n.bs)],
rats[round(0.975*n.bs)])

CI.bs
```

```
[1] 0.4331649 0.6997233
```

Uso de `Rcpp`

Consulta en la web de `Github` para una referencia completa de `Rcpp`.

Unofficial Rcpp API Documentation:

https://github.com/coatless-api-docs/rcpp-api

O el tutorial:

Rcpp for everyone:

https://teuder.github.io/rcpp4everyone_en

Rewriting R code in C++

https://adv-r.hadley.nz/rcpp.html

Instalación previa del compilador de `Rtools`:

Para usar `Rcpp` es necesario instalar un compilador de C++.

En Windows, hay que ir a Rtools:

https://cran.r-project.org/bin/windows/Rtools

Una vez instalado todo, se escribe un programa en C++ y se puede grabar, por ejemplo, en el fichero denominado `boot_ratio2prop.cpp`

Alternativamente se puede *incrustar* el programa de `Rcpp` en el propio programa de R, aunque para programas largos esto no es muy conveniente.

Se compila el programa mediante la orden `sourceCpp`.

Notas

`#include <Rcpp.h>`

Esta línea incluye el archivo de encabezado Rcpp, que proporciona funcionalidades para integrar fácilmente código C++ con R.

`using namespace Rcpp;`

Esta línea importa el espacio de nombres Rcpp, lo que significa que podemos usar objetos y funciones Rcpp sin especificar el espacio de nombres explícitamente.

`// [[Rcpp::export]]`

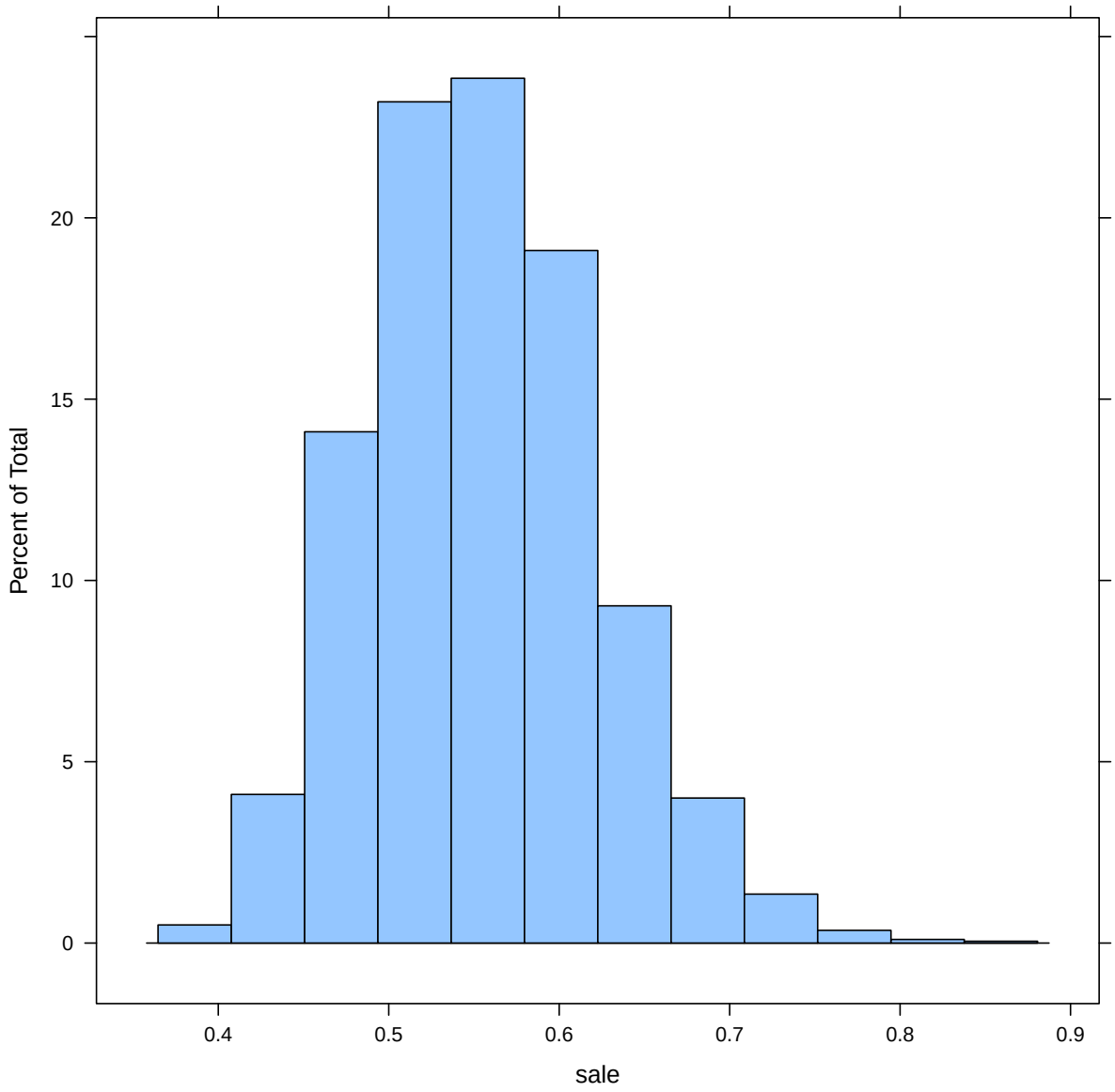
Este es un atributo proporcionado por Rcpp. Indica que la siguiente función debe exportarse y hacerse accesible desde R.

```
n1 = 11037 + 104      # tamaño muestra 1
s1 = 104              # numero de éxitos

n2 = 11034 + 189      # tamaño muestra 2
s2 = 189              # numero de éxitos

p1pre = c(rep(1,s1), rep(0,n1-s1))
p2pre = c(rep(1,s2), rep(0,n2-s2))

# Tomamos permutaciones de los datos anteriores
p1 = sample(p1pre, n1)   # muestra 1
p2 = sample(p2pre ,n2)   # muestra 2

# install.packages("Rcpp")
if (!require(Rcpp)) install.packages("Rcpp")

library(Rcpp)

sourceCpp(code='
#include <Rcpp.h>
using namespace Rcpp ;
// [[Rcpp::export]]

NumericVector boot_ratio2prop(NumericVector p1,
NumericVector p2, int replicas=1000) {

    int n1 = p1.size();
    int n2 = p2.size();

    NumericVector bs1(replicas);
    NumericVector bs2(replicas);
    NumericVector ratio(replicas);
    bool replace = true;

    for(int i=0; i<replicas; i++) {
       bs1[i] = sum(sample(p1, n1, replace))/n1;
       bs2[i] = sum(sample(p2, n2, replace))/n2;
       ratio[i] = bs1[i]/bs2[i];
```

```
    }

return ratio;
}
'
)
```

Se ejecuta el programa desde R

```
replica = 2000

sale = boot_ratio2prop(p1, p2, replica)

lattice::histogram(sale)
```


Percent of Total
0
5
10
15
20
0.4
0.5
0.6
0.7
0.8
0.9
sale

Alternativa usando `dplyr`

Ejemplo de efecto de aspirina

```
library(dplyr)
library(purrr)

ensayo = data_frame(paciente = 1:22071,
grupo = ifelse(paciente <= 11037, "aspirina", "control"),
ataqueCorz = c(rep(TRUE, 104), rep(FALSE, 10933), rep(TRUE, 189), rep(FALSE, 10845)))
ensayo
```

```
# A tibble: 22,071 x 3
   paciente grupo    ataqueCorz
      <int> <chr>    <lgl>
 1        1 aspirina TRUE
 2        2 aspirina TRUE
 3        3 aspirina TRUE
 4        4 aspirina TRUE
 5        5 aspirina TRUE
 6        6 aspirina TRUE
 7        7 aspirina TRUE
 8        8 aspirina TRUE
 9        9 aspirina TRUE
10       10 aspirina TRUE
# i 22,061 more rows
```

```
sum_stats = ensayo %>%
   group_by(grupo) %>%
   summarise(
      n_ataques = sum(ataqueCorz),
      n_gente = n(),
      rate_ataques = (n_ataques / n_gente) * 100
   )

sum_stats
```

```
# A tibble: 2 x 4
  grupo    n_ataques n_gente rate_ataques
  <chr>        <int>   <int>        <dbl>
1 aspirina       104   11037        0.942
2 control        189   11034        1.71
```

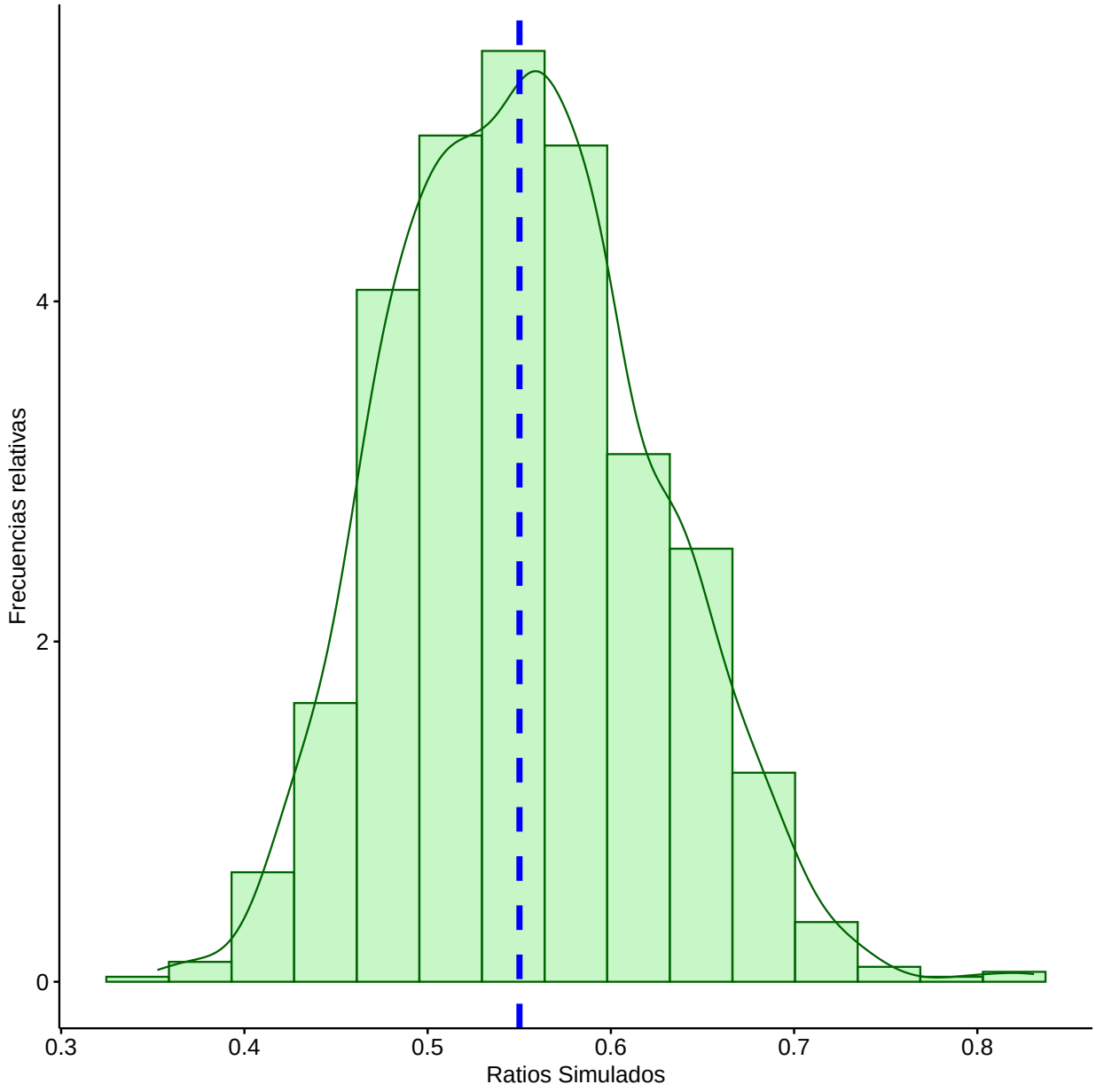
```
ratio_rates = sum_stats$rate_ataques[1] / sum_stats$rate_ataques[2]
ratio_rates
```

```
[1] 0.550115
```

```
boot_ratio_tasas = function(){
   boot_muestra = ensayo %>%
     group_by(grupo) %>%
       sample_frac(replace = TRUE)

    tasas = boot_muestra %>%
      summarise(rate_ataques = sum(ataqueCorz) / n()) %>%
      pull(rate_ataques)    # convierte los elementos del dataframe en vectores

    tasas[1] / tasas[2]
}

# ratioBoot = rerun(1000, boot_ratio_tasas()) %>%
# map_dbl(~.x)   # convierte los elementos de la lista en números reales

ratioBoot = unlist(rerun(1000, boot_ratio_tasas()))

se = sd(ratioBoot)
cat("se: ", se, "\n")
```

```
se:  0.07045184
```

```
library(ggpubr)
gghistogram(ratioBoot, y="..density..",
ylab="Frecuencias relativas", xlab="Ratios Simulados",
bins = 15, fill = "lightgreen", color="darkgreen", add_density = TRUE) +
geom_vline(xintercept=ratio_rates, colour="blue", linetype = "dashed", size=1.5)
```




```
# hist(boot_ratio_tasas)
```

Ejemplo del efecto de un tratamiento quirúrgico sobre ratones

```
Trata = c(94, 197, 16, 38, 99, 141, 23)
Cont = c(52, 104, 146, 10, 51, 30, 40, 27, 46)

# Grafico stem-and-leaf de grupo tratamiento
stem(Trata,scale=2)
```

```
  The decimal point is 1 digit(s) to the right of the |

   0 | 6
   2 | 38
   4 |
   6 |
   8 | 49
  10 |
  12 |
  14 | 1
  16 |
  18 | 7
```

```
# Grafico stem-and-leaf de grupo control
stem(Cont,scale=2)
```

```
  The decimal point is 1 digit(s) to the right of the |

   0 | 0
   2 | 70
   4 | 0612
   6 |
   8 |
  10 | 4
  12 |
  14 | 6
```

```
ecdf1 = ecdf(Trata)
ecdf2 = ecdf(Cont)

plot(ecdf2, verticals=TRUE, do.points=FALSE, col='blue', xlab="dias",
ylab="Distribucion Empirica",
main="Cont(azul)/Trata(naranja)" )

plot(ecdf1, verticals=TRUE, do.points=FALSE, add=TRUE, col='orange')
```

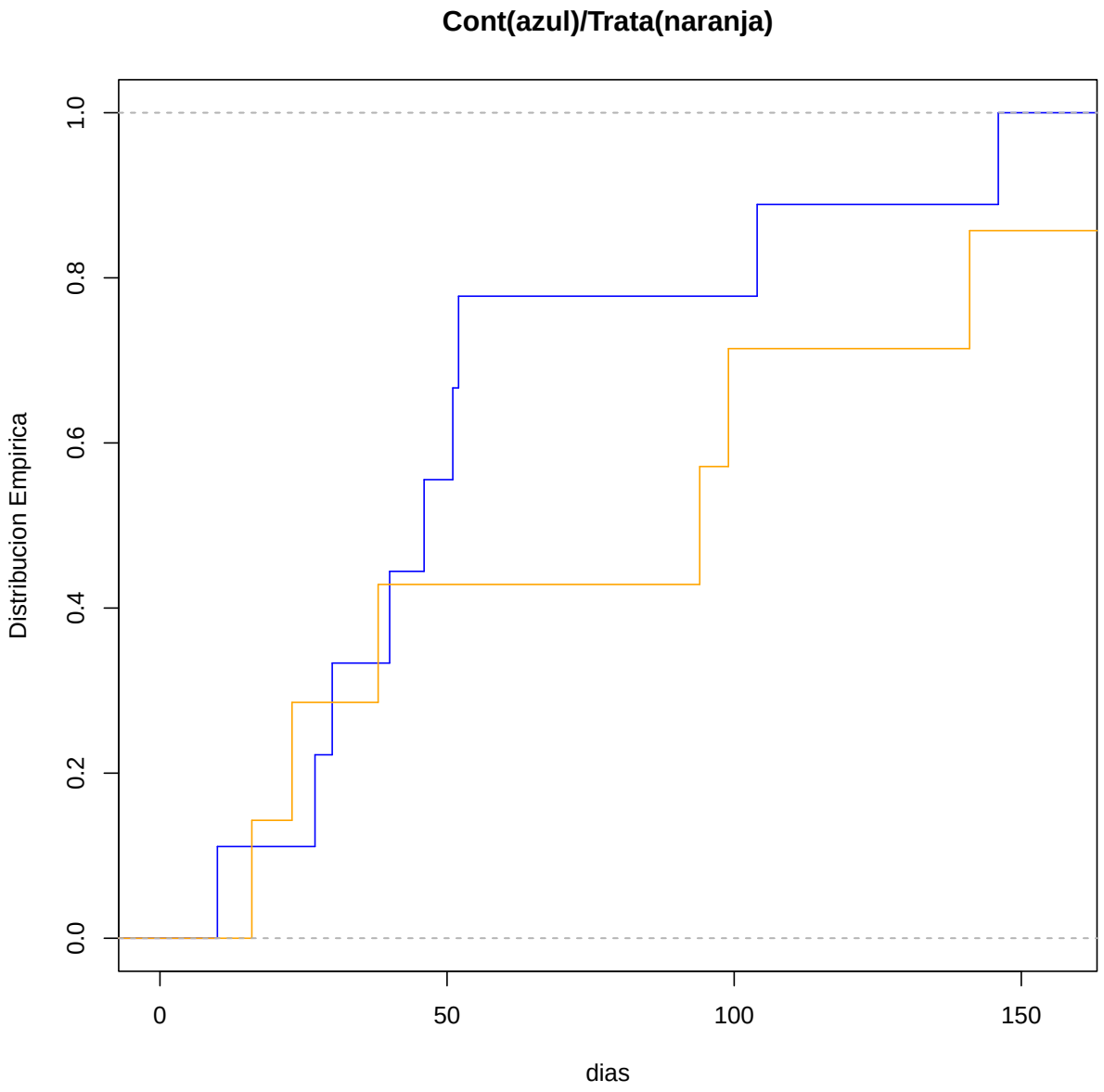
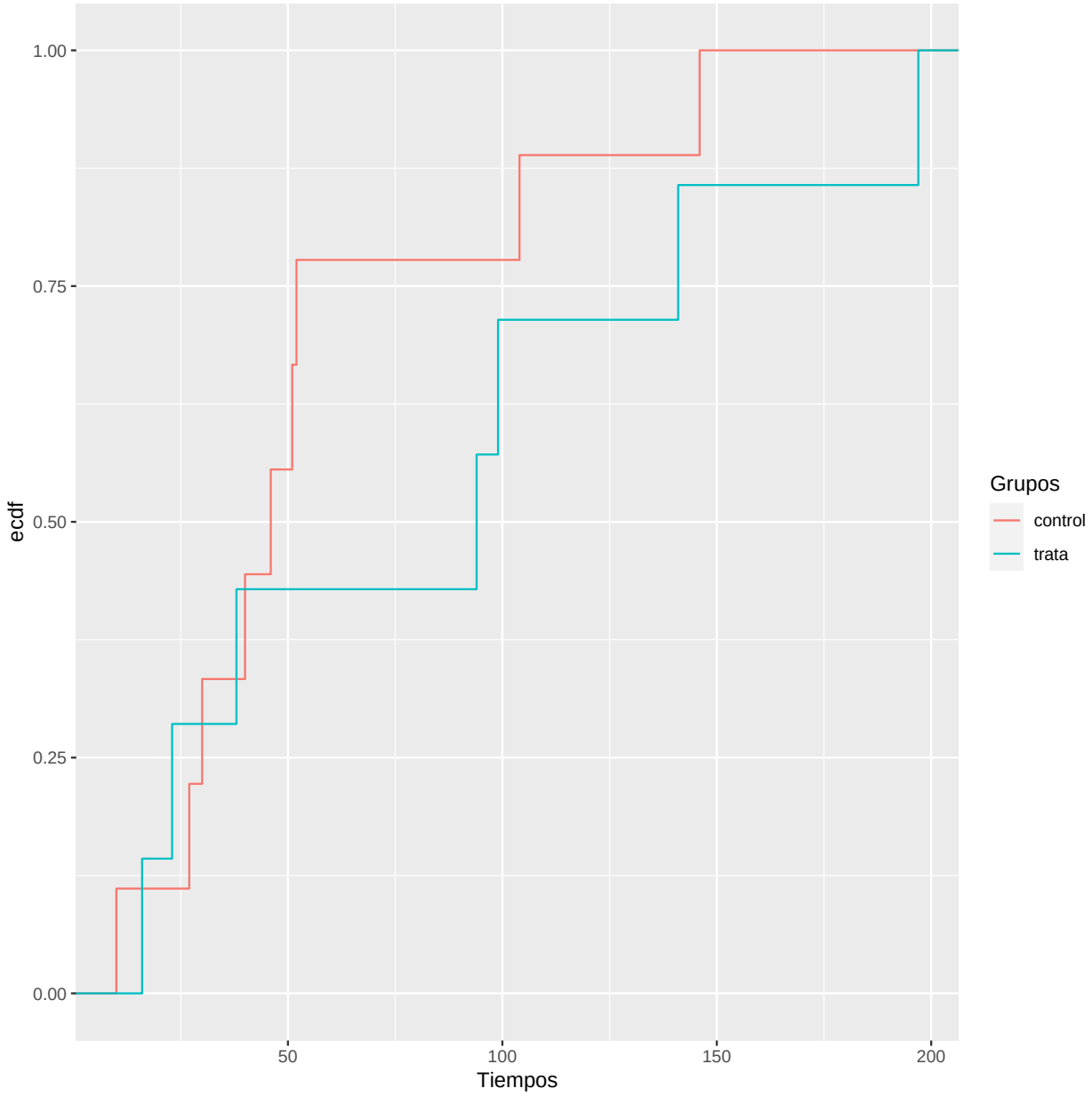


Gráfico alternativo con ggplot

```
library(ggplot2)
Tiempos = c(Trata, Cont)
Grupos = as.factor(c(rep("trata",length(Trata)),
rep("control",length(Cont))))

Losdatos = data.frame(Tiempos, Grupos)
```

```
ggplot(Losdatos, aes(Tiempos, colour = Grupos)) +
stat_ecdf()
```



Siguiendo a *Efron y Tibshirani*:

```
mean(Trata)
```

```
[1] 86.85714
```

```
mean(Cont)
```

```
[1] 56.22222
```

```
(sdDiff = sqrt(var(Trata)/length(Trata)+var(Cont)/length(Cont)))
```

```
[1] 28.93607
```

Aplicas el **Teorema Central del Límite**.

```
(t = (mean(Trata) - mean(Cont)) / sdDiff)
```

```
[1] 1.058711
```

Pones los valores en un solo vector y defines otro vector de 1's y 2's según su grupo de pertenencia:

```
x = matrix(c(Trata, Cont, rep(1,length(Trata)), rep(2, length(Cont))), ncol=2)

# t-test de Student
t.test(x[,1] ~ x[,2])
```

```
    Welch Two Sample t-test

data:  x[, 1] by x[, 2]
t = 1.0587, df = 9.6545, p-value = 0.3155
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 -34.15279  95.42263
sample estimates:
mean in group 1 mean in group 2
       86.85714        56.22222
```

Uso de `Rcpp`

Se escribe un programa en C++ y se graba e.j. en el fichero denominado `MediaRatones.cpp`

Alternativamente, se puede *incrustar* el programa de Rcpp en el propio programa de R.

Se compila el programa mediante la orden `sourceCpp`.

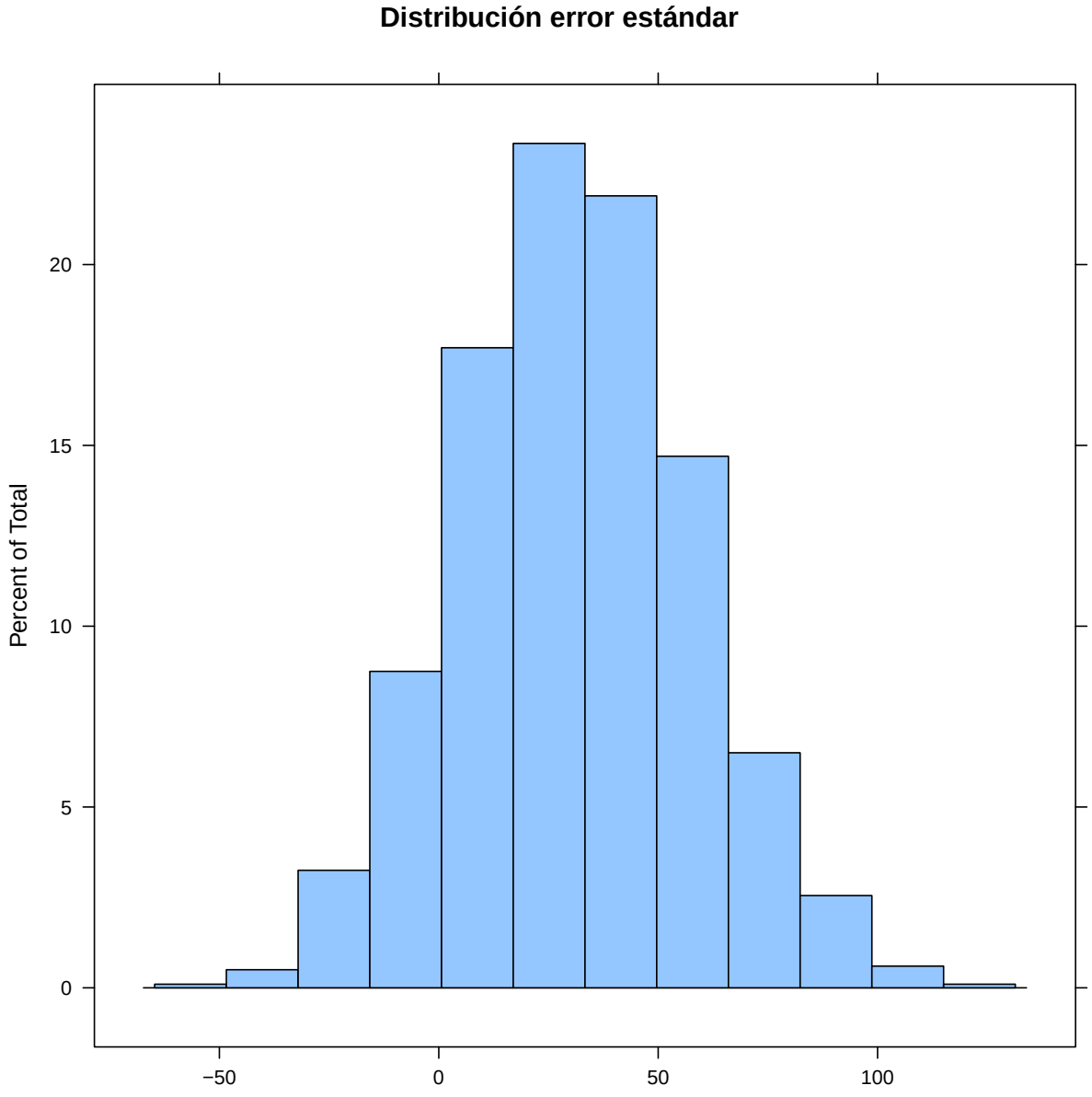
```
# install.packages("Rcpp")
if (!require(Rcpp)) install.packages("Rcpp")

library(Rcpp)

sourceCpp(code='
#include <Rcpp.h>
using namespace Rcpp ;
// [[Rcpp::export]]

List boot_MediaRatones(NumericVector trata, NumericVector control, int replicas=1000) {

    int n1 = trata.size();
    int n2 = control.size();
    double sale;

    NumericVector bs1(replicas);
    NumericVector bs2(replicas);
    NumericVector diff(replicas);

    bool replace = true;

    for(int i=0; i<replicas; i++) {

       bs1[i] =
       mean(sample(trata, n1, replace));

       bs2[i] =
       mean(sample(control, n2, replace));

       diff[i] = bs1[i]-bs2[i];
    }

    sale = sd(diff);

List saletodo;
    saletodo["sd"] = sale;
    saletodo["vector"] = diff;
    return saletodo;
}
```

```
'
)
```

```
Trata = c(94, 197, 16, 38, 99, 141, 23)
Cont = c(52, 104, 146, 10, 50, 31, 40, 27, 46)

sale = boot_MediaRatones(Trata, Cont, replica)

sale["sd"]
```

```
$sd
[1] 26.78162
```

```
lattice::histogram(unlist(sale["vector"]), main="Distribución error estándar", xlab="")
```


Distribución error estándar
Percent of Total
0
5
10
15
20
−50
0
50
100

Cálculo de medianas

Se calcula el error estándar del estadístico *mediana muestral*.

Esto sería semejante a los programas anteriores sustituyendo el comando `mean` por `median`.

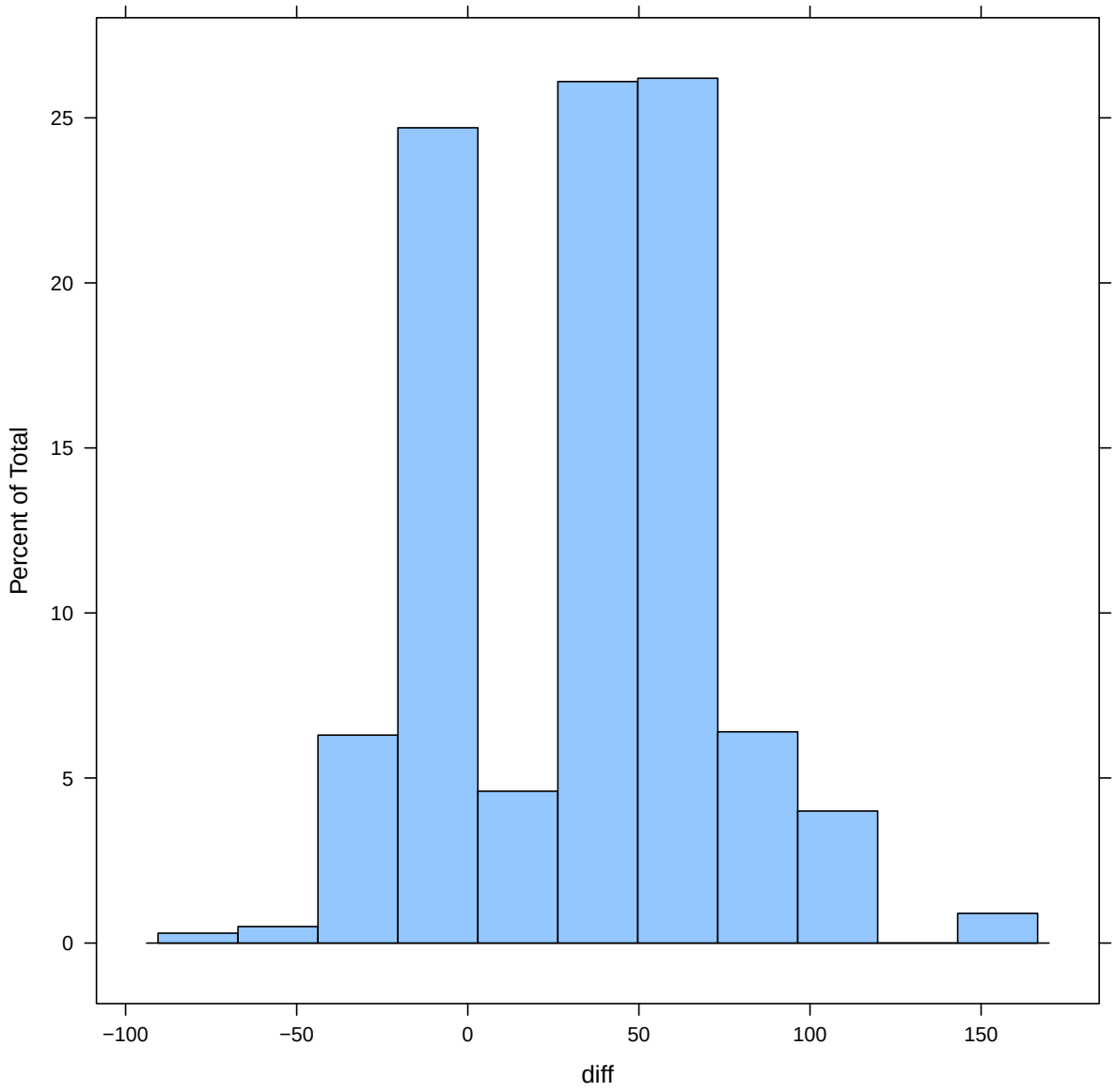
```
# Tamaños muestrales
n1 = length(Trata)
n2 = length(Cont)

n.bs = 1000        # número de muestras bootstrap

bs1 = rep(0, n.bs)
bs2 = rep(0, n.bs)

for (i in 1:n.bs) {
   bs1[i] = median(sample(Trata, n1, replace=TRUE))
   bs2[i] = median(sample(Cont, n2, replace=TRUE))
}
```

```
# Réplicas bootstrap de estimadores de las diferencias
diff = bs1-bs2
sd(diff)       # estima del error estándar
```

```
[1] 39.58423
```

Histograma de las estimas de las diferencias:

```
lattice::histogram(diff)
```



R proporciona otras funciones para bucles implícitos como `apply`, `tapply` y `lapply`.

El comando `apply` ejecuta una función a través de una matriz, un dataframe o un array de datos.

La sintaxis de la función se resume en:

```
apply(#Array, matrix o dataframe, Margin= #1 se aplica sobre filas, #2 se aplica sobre
columnas #c(1,2) se aplica sobre ambas filas y columnas, FUN= #Funcion)
```

```r
# Función ejemplo

subtractmean = function(x) {
return(x-mean(x))
}

# Definimos una matriz 2 por 3
X = matrix(rnorm(2*3), nrow=2, ncol= 3)
X
```

```
           [,1]       [,2]       [,3]
[1,] -0.5059951  0.6355219 -0.5834676
[2,]  0.2343510 -2.1316442  0.1043022
```

```r
Y = apply(X,2,subtractmean)
Y
```

```
           [,1]      [,2]       [,3]
[1,] -0.3701731  1.383583 -0.3438849
[2,]  0.3701731 -1.383583  0.3438849
```

Del mismo modo se pueden definir comandos relacionados: `lapply` para listas y `sapply` para vectores.

Mira en

https://www.datacamp.com/community/tutorials/r-tutorial-apply-family

Ejemplo de ratones: Programas alternativos

Caso de la media.

```r
B = 2000

mean(Trata) - mean(Cont)
```

```
[1] 30.63492
```

```r
sd(replicate(B, mean(sample(Trata,replace=TRUE)) -
mean(sample(Cont, replace=TRUE))))
```

```
[1] 26.75313
```

```r
# Caso de la mediana
median(Trata) - median(Cont)
```

```
[1] 48
```

```r
sd(replicate(B, median(sample(Trata,replace=TRUE)) -
median(sample(Cont, replace=TRUE))))
```

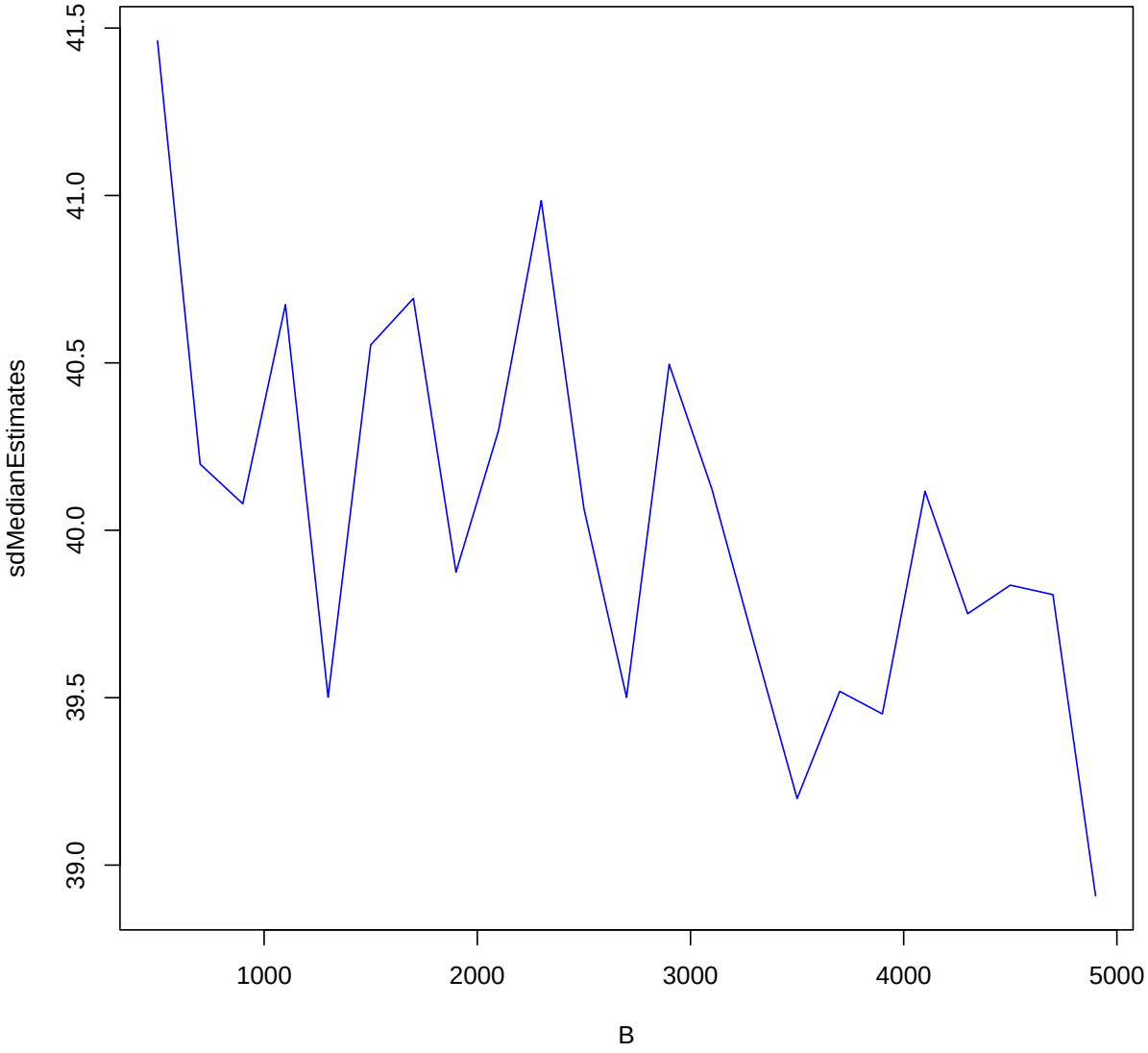
```
[1] 40.35879
```

En el caso de usar `Rcpp`, el programa para la mediana sería semejante al de la media, escribiendo `median` en lugar de `mean` en el programa.

Programa para las medianas con distintos tamaños de muestras bootstrap

```r
sdMedian = function(B){
    sd(replicate(B,
    median(sample(Trata, replace=TRUE)) -
    median(sample(Cont, replace=TRUE))))
    }

# Se prueban diferentes numeros de replicas
B = seq(500, 5000, 200)
sdMedianEstimates = sapply(B, sdMedian)
```

```r
plot(sdMedianEstimates ~ B, type="l", col="blue")
```


sdMedianEstimates
41.5
41.0
40.5
40.0
39.5
39.0
1000
2000
3000
4000
5000
B

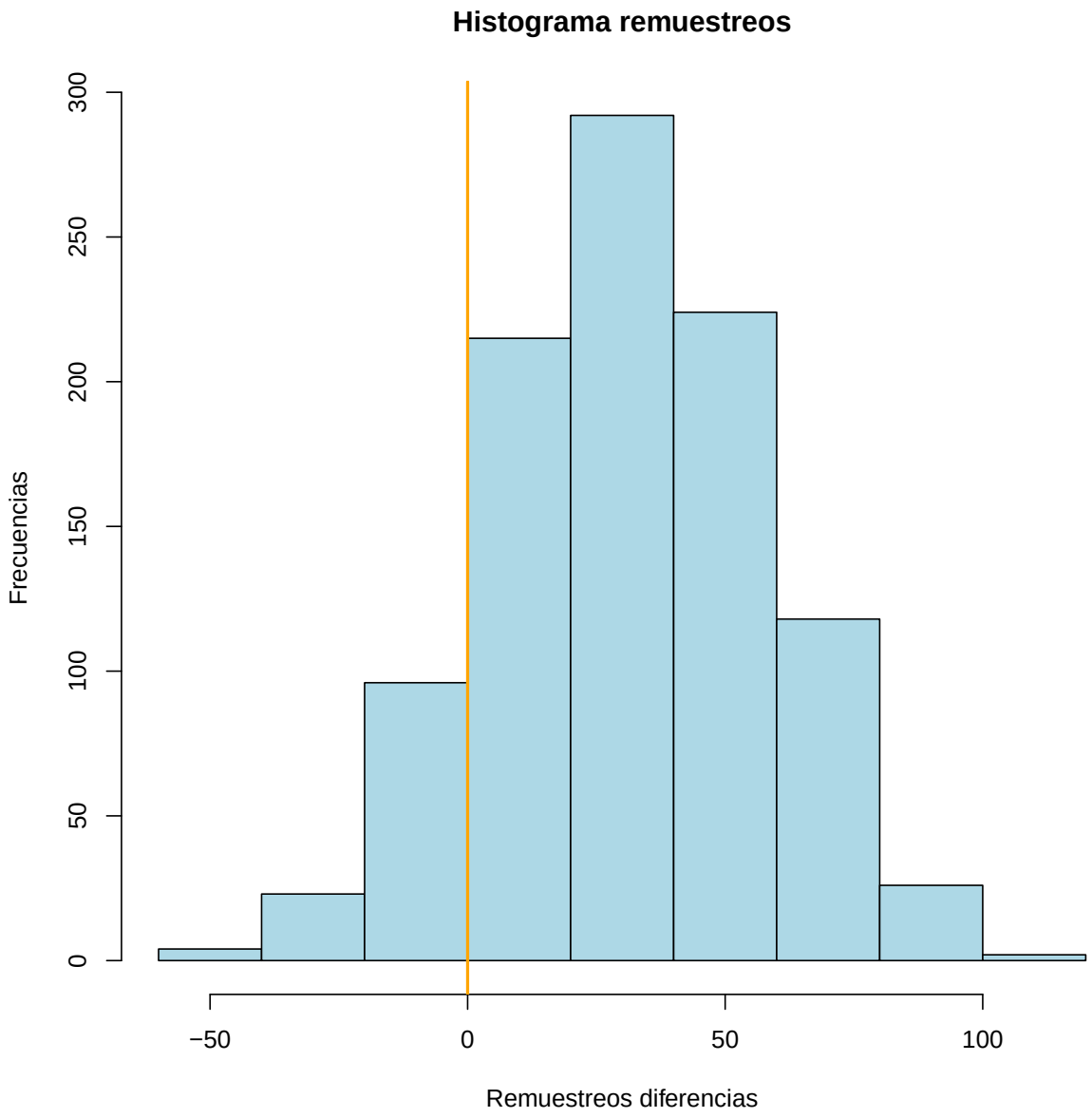
Uso de las librerías `boot` y `bootstrap`.

Se puede programar fácilmente el ejemplo de los ratones con las librerías `bootstrap` y `boot`.

```
library(bootstrap)

mouse.boot.c = bootstrap(mouse.c, 1000, mean)
mouse.boot.t = bootstrap(mouse.t, 1000, mean)

mouse.boot.diff = mouse.boot.t$thetastar -
mouse.boot.c$thetastar

sd(mouse.boot.diff)
```

```
[1] 26.66637
```

```
hist(mouse.boot.diff, main='Histograma remuestreos',
col='lightblue', xlab='Remuestreos diferencias',
ylab='Frecuencias')
abline(v=0, col="orange", lwd=2)
```



Con la librería `boot`:

```
library(boot)
trat = c(94, 197, 16, 38, 99, 141, 23)
control = c(52, 104, 146, 10, 51, 30, 40, 27, 46)

# Defines un dataframe que asigna valores 1 y 2 dependiendo
# de si es control o tratamiento

bichos = data.frame(sobrevive=c(control,trat),
```

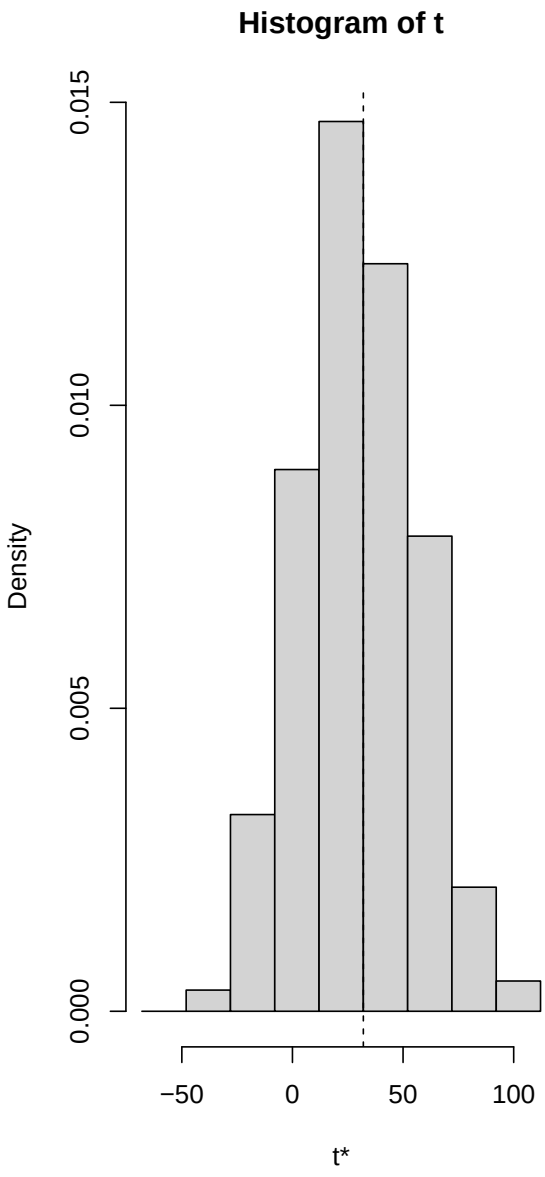
```r
grupo = c(rep(1,length(control)), rep(2,length(trat))))

bichos
```

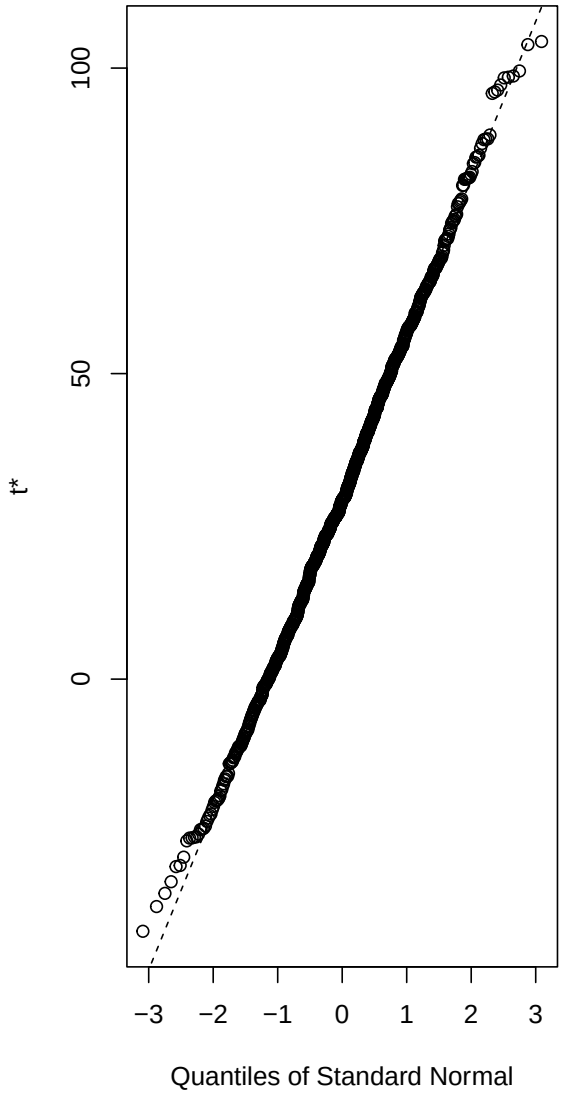
```
   sobrevive grupo
1         52     1
2        104     1
3        146     1
4         10     1
5         51     1
6         30     1
7         40     1
8         27     1
9         46     1
10        94     2
11       197     2
12        16     2
13        38     2
14        99     2
15       141     2
16        23     2
```

```r
lafuncion = function(x, i) {

# Remuestreas dentro de cada grupo las observaciones
booty = tapply(x$sobrevive, x$grupo,
FUN=function(x) {sample(x,length(x),TRUE)} )

# Calculas las diferencias de las medias de cada grupo
diff(sapply(booty, mean))
}

# Aplicas la función
bbichos = boot(data=bichos, lafuncion, R=1000)

sd(bbichos$t)
```

```
[1] 25.86556
```

```r
plot(bbichos, nclass=10)
```


Histogram of t
Density
0.000
0.005
0.010
0.015
−50
0
50
100
t*


t*
0
50
100
−3
−2
−1
0
1
2
3
Quantiles of Standard Normal