Tema 2. Introducción al Jackknife y a los tests de permutaciones

Jackknife

Se cargan los datos de tipos de parches de hormonas *patch* desde la librería `bootstrap` (que contiene los ficheros de datos del libro de Efron y Tibshirani (1993)).

```
    data(patch, package="bootstrap")

    n = nrow(patch)
    y = patch$y
    z = patch$z
    theta.hat = mean(y) / mean(z)

    y
```

```
[1] -1200  2601 -2705  1982 -1290   351  -638 -2719
```

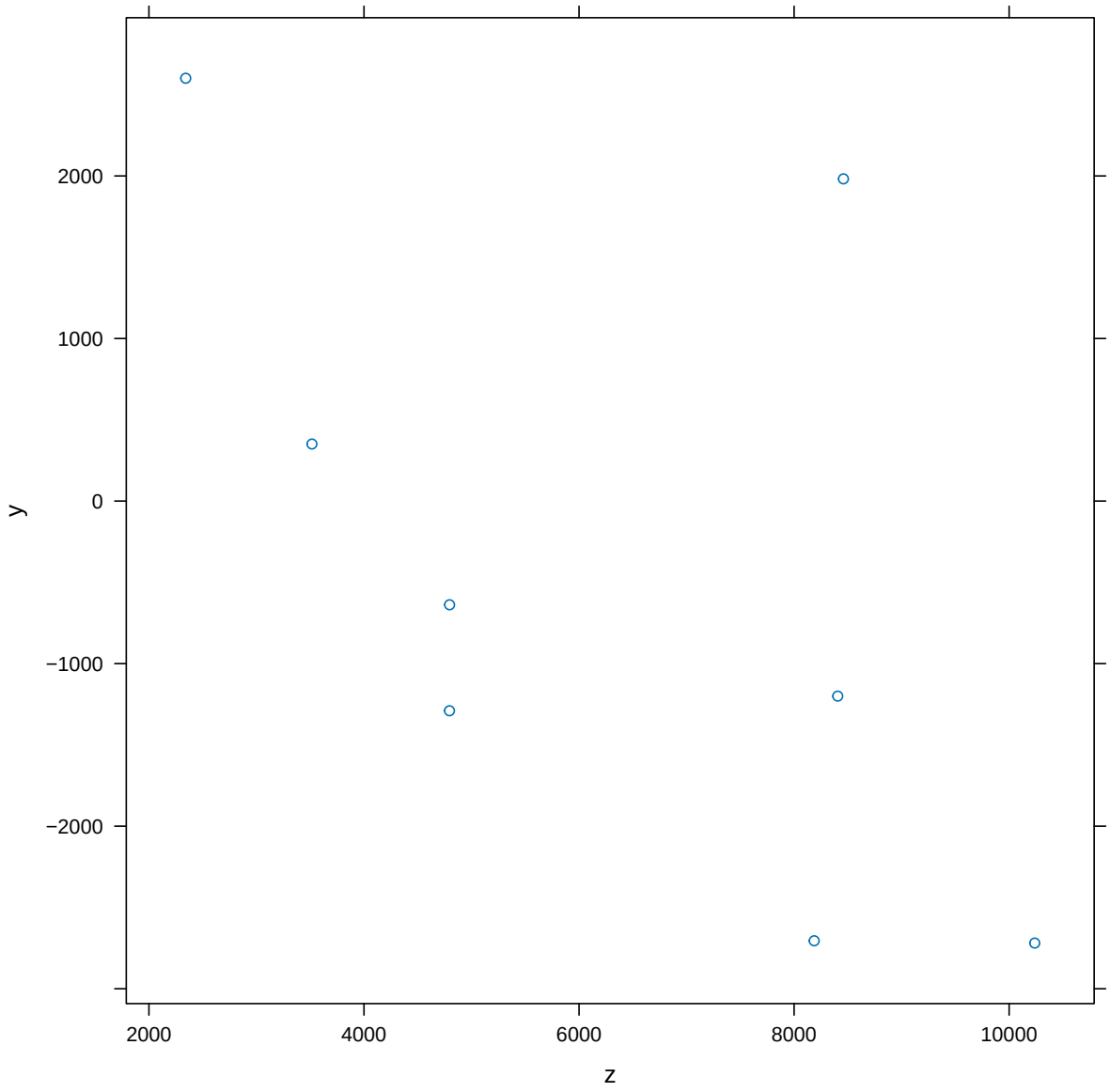
```
    z
```

```
[1]  8406  2342  8187  8459  4795  3516  4796 10238
```

```
    theta.hat
```

```
[1] -0.0713061
```

```
library(lattice)
xyplot(y ~ z, data=patch)
```




```
jacks = sapply(1:n, function(i) with(patch[-i,], mean(y)/mean(z)))
jacks
```

```
[1] -0.05711856 -0.12849970 -0.02145610 -0.13245033 -0.05067038 -0.08404803
[7] -0.06486298 -0.02219698
```

Según la fórmula teórica:

```
(biasJack = (n-1)*(mean(jacks) - theta.hat))
```

```
[1] 0.008002488
```

Alternativamente

```
theta.jack = numeric(n)

for (i in 1:n){
   theta.jack[i] = mean(y[-i]) / mean(z[-i])
}

(sesgo = (n-1)*(mean(theta.jack) - theta.hat))
```

```
[1] 0.008002488
```

Según la fórmula teórica:

```
se = sqrt((n-1) * mean((theta.jack - mean(theta.jack))^2))
se
```

```
[1] 0.1055278
```

Se puede usar `Rcpp` que es más eficiente cuando el tamaño muestral es alto.

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.

```
library(Rcpp)

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

List jackknifeEstimas(NumericVector y, NumericVector z) {

    int n = y.size();
    NumericVector theta_jack(n);

    for (int i = 0; i < n; i++) {
        double mean_y = (sum(y) - y[i]) / (n - 1);
        double mean_z = (sum(z) - z[i]) / (n - 1);
        theta_jack[i] = mean_y / mean_z;
    }

    double theta_hat = mean(y) / mean(z);
    double sesgo = (n - 1) * (mean(theta_jack) - theta_hat);
    double se = sqrt((n-1) * mean(pow(theta_jack - mean(theta_jack), 2)));

    return List::create(Named("Theta Original") = theta_hat,
    Named("sesgo") = sesgo, Named("se") = se);
}
')

# Datos originales
z = c(8406, 2342, 8187, 8459, 4795, 3516, 4796, 10238)
y = c(-1200,  2601, -2705,  1982, -1290, 351, -638, -2719)

resulta = jackknifeEstimas(y, z)
resulta
```

```
$`Theta Original`
[1] -0.0713061

$sesgo
```

```
[1] 0.008002488

$se
[1] 0.1055278
```

Alternativamente, se puede calcular los estimadores jackknife usando la librería `bootstrap`:

```
library(bootstrap)

    n = nrow(patch)
    y = patch$y
    z = patch$z

datos = as.matrix(cbind(y, z))
```

```
sampbioeq = mean(y)/mean(z)
sampbioeq
```

```
[1] -0.0713061
```

```
# Defines la función de bioequivalencia
bioeq = function(ind, datos) { mean(datos[ind,1])/mean(datos[ind,2]) }

# Aplicas jackknife
jackbioeq = jackknife(1:n, bioeq, datos)
jackbioeq
```

```
$jack.se
[1] 0.1055278

$jack.bias
[1] 0.008002488

$jack.values
[1] -0.05711856 -0.12849970 -0.02145610 -0.13245033 -0.05067038 -0.08404803
[7] -0.06486298 -0.02219698

$call
jackknife(x = 1:n, theta = bioeq, datos)
```

```
# Estimador jackknife con el sesgo corregido
bioeqjack = sampbioeq - jackbioeq$jack.bias
bioeqjack
```

```
[1] -0.07930858
```

Ejemplo simulado sobre correlaciones

Se puede considerar un ejemplo simulado para calcular correlaciones. Se usa el comando `jackknife` de la librería `bootstrap`.

```
library(bootstrap)

datos = matrix(rnorm(30), ncol=2)
n = 15
teta = function(ind, datos){
   cor(datos[ind,1], datos[ind,2])
}

results = jackknife(1:n, teta, datos)
results
```

```
$jack.se
[1] 0.3021225

$jack.bias
[1] -0.06039187

$jack.values
 [1] -0.3837586 -0.3223967 -0.3930229 -0.6423902 -0.3618817 -0.3145269
 [7] -0.3682634 -0.4004404 -0.3977604 -0.3231626 -0.3959184 -0.3914949
[13] -0.4114455 -0.4069705 -0.5299695

$call
jackknife(x = 1:n, theta = teta, datos)
```

Tests de permutaciones

Si usamos la metodología clásica del test de la t-Student:

```
library(bootstrap)

t.test(mouse.t, mouse.c, alternative="greater",
var.equal=TRUE)
```

```
    Two Sample t-test

data:  mouse.t and mouse.c
t = 1.1214, df = 14, p-value = 0.1405
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -17.48178       Inf
sample estimates:
mean of x mean of y
 86.85714  56.22222
```

Mediante un método de remuestreo por permutaciones.

Se toman muestras **SIN** reemplazamiento.

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

thetaHat = mean(mouse.t) - mean(mouse.c)

xy = c(mouse.c,mouse.t)
n = length(mouse.c)
N = length(xy)

thetaStar = replicate(5000,
{ind = sample(1:N, n, replace=FALSE);
mean(xy[ind])-mean(xy[-ind])})
```

Se puede calcular el p-valor, sabiendo que

```
thetaHat
```

```
[1] 30.63492
```

Así

```
mean(thetaStar>thetaHat)
```

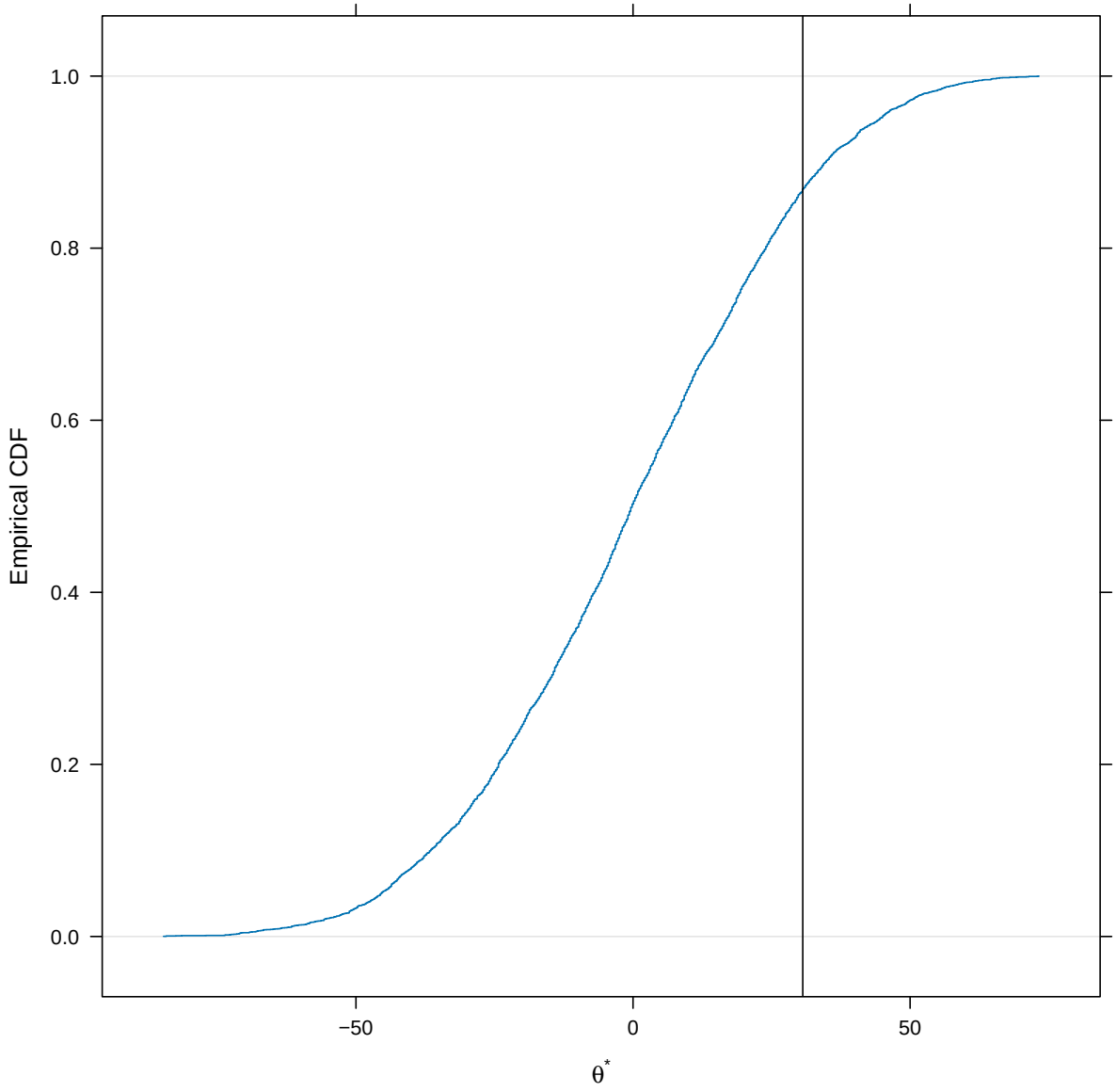
```
[1] 0.1312
```

Gráficamente se observa que

```
library(latticeExtra)

tetalabel = expression(theta^{"*"})

ecdfplot(thetaStar, xlab = tetalabel) + layer(panel.abline(v=thetaHat))
```


1.0
0.8
0.6
0.4
0.2
0.0
Empirical CDF
−50
0
50
θ*

Alternativamente, se puede usar `Rcpp` que es más eficiente cuando el tamaño muestral es alto.

```
library(Rcpp)

sourceCpp(code = '
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]

double permu_diff(NumericVector mouse_t, NumericVector mouse_c, int nReps) {
  int n_t = mouse_t.size();
  int n_c = mouse_c.size();

  NumericVector xy(n_t + n_c);
  std::copy(mouse_t.begin(), mouse_t.end(), xy.begin());
  std::copy(mouse_c.begin(), mouse_c.end(), xy.begin() + n_t);

  double thetaHat = mean(mouse_t) - mean(mouse_c);

  int N = xy.size();

  NumericVector thetaStar(nReps);

  for (int i = 0; i < nReps; i++) {
    IntegerVector ind = sample(N, n_c, false);    // false indica sin reemplazamiento

    NumericVector subSample1(n_c);
    NumericVector subSample2(n_t);

    for(int j = 0, k1 = 0, k2 = 0; j < N; j++) {

    // Mira en
    // https://www.educative.io/answers/how-to-use-the-find-function-in-cpp

      auto kk = std::find(ind.begin(), ind.end(), j + 1);
      // j+1 porque ind contiene valores que empiezan en 1

      if (kk != ind.end()) {
      // kk != final del rango?
      // true -> encontrado

      // De manera compacta, alternativamente
      // if(std::find(ind.begin(), ind.end(), j + 1) != ind.end()) {

        subSample1[k1++] = xy[j];
      } else {
        subSample2[k2++] = xy[j];
      }

    }
```

```
    double mean1 = mean(subSample1);
    double mean2 = mean(subSample2);
    thetaStar[i] = mean1 - mean2;
  }

  return mean(thetaStar > thetaHat);
}
')
```

```
# sourceCpp("tufichero.cpp")

mouse.t = c(94, 197, 16, 38, 99, 141, 23)
mouse.c = c(52, 104, 146, 10, 50, 31, 40, 27, 46)

thetaHat = mean(mouse.t) - mean(mouse.c)
thetaHat
```

```
[1] 30.63492
```

```
sale = permu_diff(mouse.t, mouse.c, 5000)
sale
```

```
[1] 0.1362
```

Ejemplo sobre actividades extraescolares

Se trata de determinar si la participación en actividades extraescolares aumenta la capacidad de empatía de los estudiantes.

Para ello el colegio ofrece un programa voluntario en el que cada participante se designa de forma aleatoria a un grupo *control* que no recibe clases extraescolares o a un grupo *tratamiento* que sí las recibe.

Al final del año todas las personas del estudio realizan un examen que determina su capacidad empática. A la vista de los resultados ¿Se puede considerar que las clases extraescolares tienen un impacto en el promedio de cómo se relacionan socialmente los estudiantes?.

```
datosOri = read.table("AfterSchool.csv", header = TRUE, sep = ",", row.names = 1)

datos = datosOri[, c("Treatment", "Delinq")]
colnames(datos) = c("grupo", "puntuacion")

datos$grupo = as.factor(datos$grupo)
levels(datos$grupo) = c("control", "tratamiento")
```

El diseño experimental del estudio emplea una asignación aleatoria de las personas a los dos grupos (tratamiento y control) para posteriormente llevar a cabo el experimento.

Esta aleatorización en la asignación implica que, en promedio, los dos grupos son iguales para todas las características, de tal forma que la única diferencia entre ellos será si reciben o no el tratamiento.

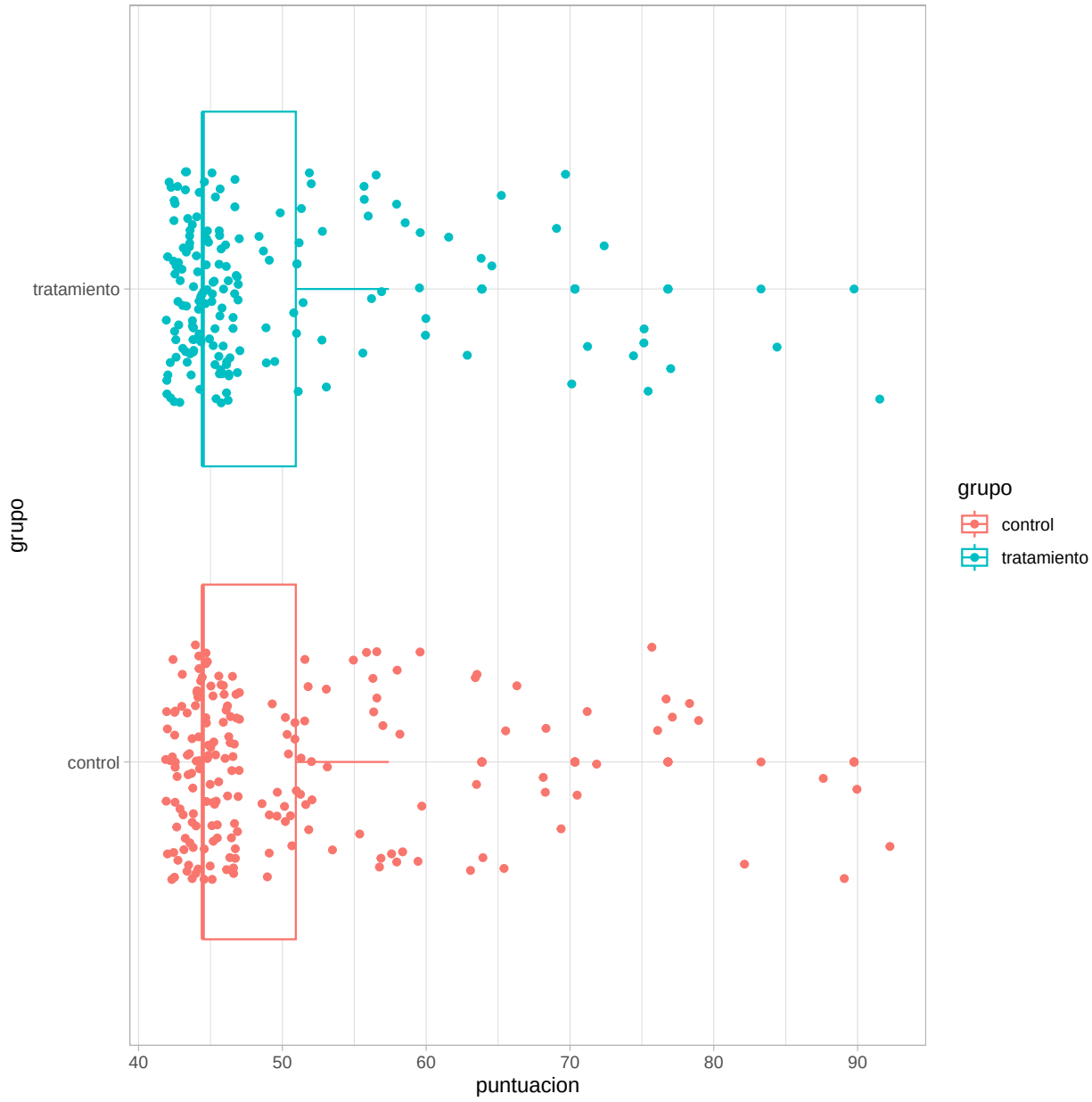
Determinar si la diferencia observada es significativa equivale a preguntarse cómo de probable es obtener esta diferencia si el tratamiento no tiene efecto y los estudiantes se han asignado de forma aleatoria en cada grupo, es decir, determinar si la diferencia observada es mayor de lo que cabría esperar debido únicamente a la variabilidad producida por la formación aleatoria de los grupos.

El diseño experimental de asignación aleatoria a los grupos permite obtener conclusiones de tipo causa/efecto. Sin embargo, dado que la selección de los sujetos no ha sido aleatoria sino mediante voluntarios, los resultados no son extrapolables a toda la población.

Usando gráficos descriptivos de diagrama de cajas y diagrama de densidad, no parece que exista una diferencia marcada en la distribución de los dos grupos.

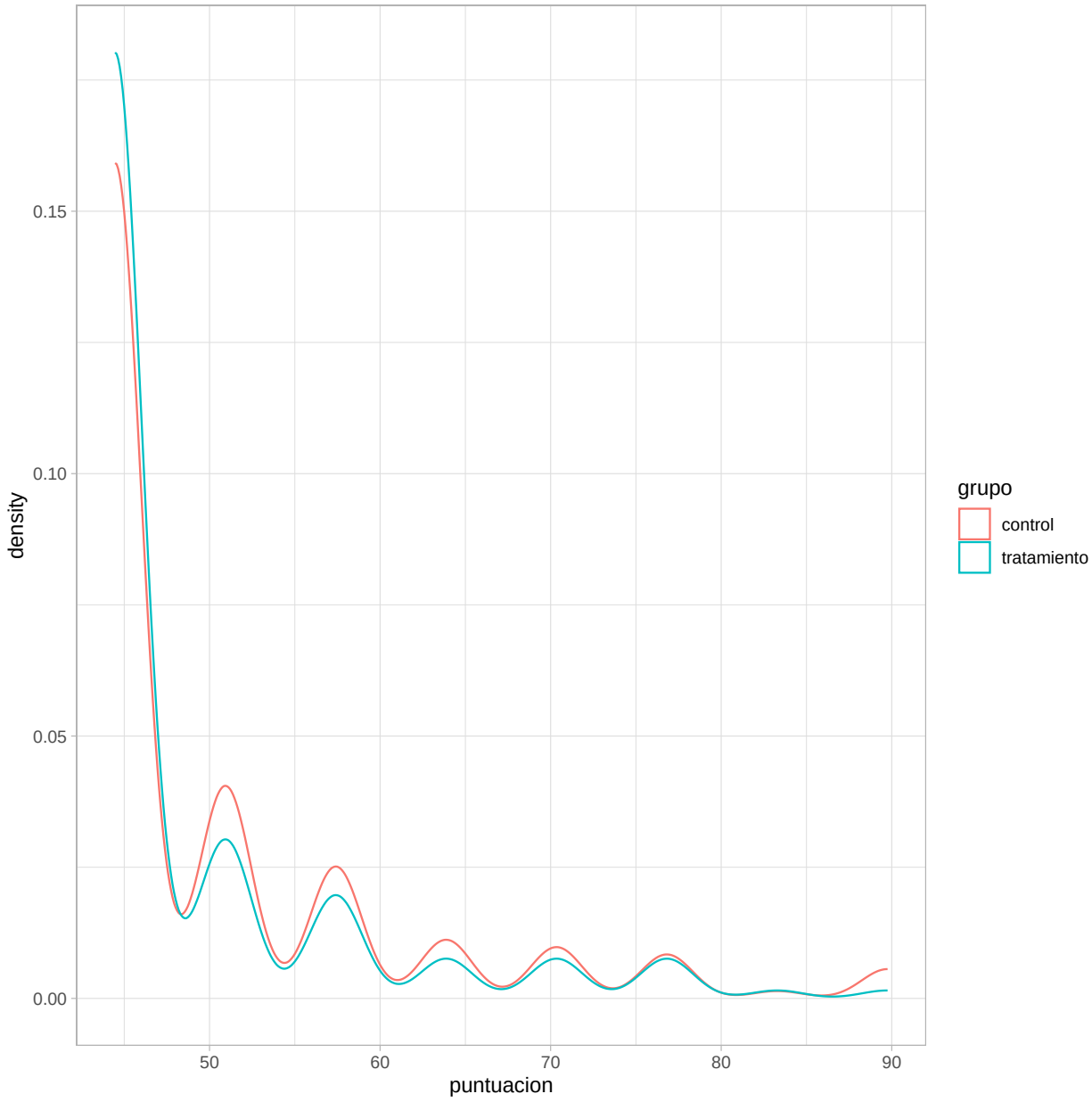
```
library(ggplot2)

ggplot(data = datos, aes(x = grupo, y = puntuacion, colour = grupo)) +
geom_boxplot() +
geom_jitter(width=0.25) +
coord_flip() +
theme_light()
```



La correspondientes gráficas de densidad son

```
ggplot(data = datos, aes(x = puntuacion, colour = grupo)) +
geom_density() +
theme_light()
```



Las medias de ambos grupos y su variabilidad son similares.

```
tapply(X = datos$puntuacion, INDEX = datos$grupo, FUN = mean)
```

```
    control tratamiento
   50.72559    49.01896
```

```r
tapply(X = datos$puntuacion, INDEX = datos$grupo, FUN = sd)
```

```
    control tratamiento
   10.52089     8.97423
```

Dado que las observaciones no se distribuyen de forma normal, los tests paramétricos de la t-Student no son adecuados, aunque dado que el tamaño muestral es grande, el resultado aproximado sería relativamente bueno.

Existen diferentes tipos de test no paramétricos que se pueden aplicar cuando no se cumplen las condiciones del teorema del límite central del límite, por ejemplo el test de *Mann-Witney* que compara medianas.

Otra opción son los métodos de resampling que permiten trabajar con cualquier estadístico (media, mediana, varianza...).

Un hecho a tener en cuenta a la hora de elegir entre tests de permutaciones o bootstrap es que el muestreo no ha sido aleatorio, ya que se ha basado en voluntarios: Una vez obtenidos los sujetos del estudio, sí se han asignado aleatoriamente a los diferentes grupos.

```r
dif_obs = mean(datos[datos$grupo == "control", "puntuacion"]) -
          mean(datos[datos$grupo == "tratamiento", "puntuacion"])

dif_obs
```

```
[1] 1.706636
```

Determinar si la diferencia observada es significativa equivale a preguntarse cómo es de probable obtener esta diferencia si el tratamiento no tiene efecto y los estudiantes se han asignado de forma aleatoria en cada grupo.

Para poder obtener la probabilidad exacta, se requiere calcular todas las posibles permutaciones en las que 356 personas pueden repartirse en dos grupos y calcular la diferencia de medias para cada una.

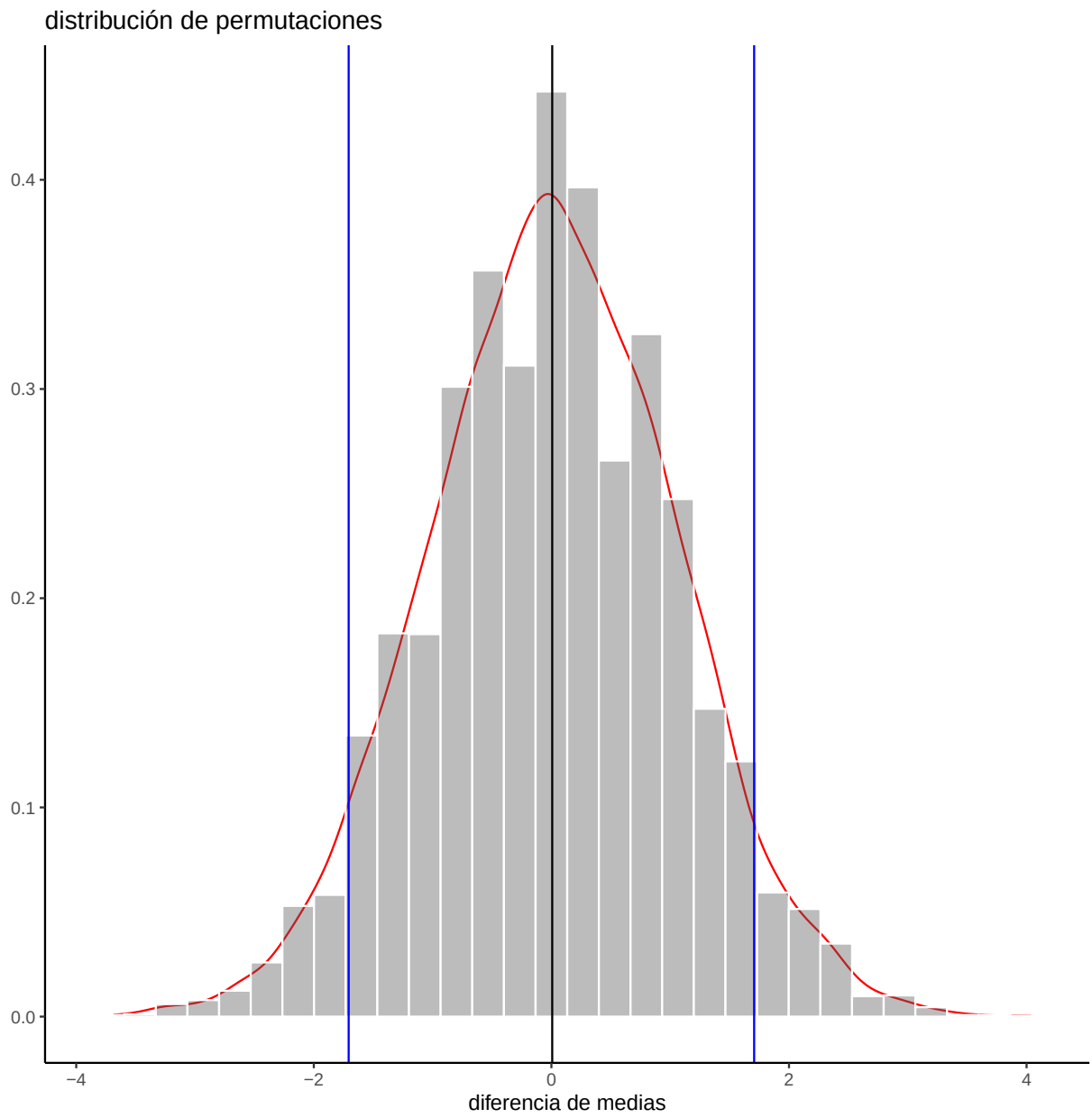
El número de permutaciones posibles es muy elevado, (3.93×10^{105}), por lo que se recurre a una simulación de MonteCarlo.

En cada iteración se asignan aleatoriamente las personas a cada uno de los grupos manteniendo el tamaño original de cada uno, se calcula la diferencia de medias y se almacena el valor.

```r
distribucion_permutaciones = rep(NA, 9999)
n_control = length(datos$grupo[datos$grupo == "control"])
n_tratamiento = length(datos$grupo[datos$grupo == "tratamiento"])
for (i in 1:9999) {
# mezcla aleatoria de las observaciones
datos_aleatorizados = sample(datos$puntuacion)
distribucion_permutaciones[i] = mean(datos_aleatorizados[1:n_control]) -
mean(datos_aleatorizados[n_control + 1:n_tratamiento])
}
```

Los datos simulados forman lo que se conoce como distribución de MonteCarlo y representa la variación esperada en la diferencia de medias debida únicamente a la asignación aleatoria de grupos.

```
library(ggplot2)
qplot(distribucion_permutaciones, geom = "blank") +
geom_line(aes(y = ..density..), stat = "density", colour = "red") +
geom_histogram(aes(y = ..density..), alpha = 0.4, colour = "white") +
geom_vline(xintercept = mean(distribucion_permutaciones)) +
geom_vline(xintercept = dif_obs, colour = "blue") +
geom_vline(xintercept = -dif_obs, colour = "blue") +
labs(title = "distribución de permutaciones", x = "diferencia de medias") +
theme_classic()
```




```
summary(distribucion_permutaciones)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.68679 -0.69854  0.03030  0.00688  0.68626  4.03893
```

```
sd(distribucion_permutaciones)
```

```
[1] 1.037534
```

La diferencia media entre grupos si el tratamiento no es efectivo es muy próxima a cero (línea vertical negra).

La desviación típica de la distribución de permutaciones indica que la variabilidad debida a la asignación aleatoria de los sujetos a los diferentes grupos es muy pequeña.

Finalmente, se calcula la probabilidad (p-valor) de obtener diferencias igual o más extremas que la observada (líneas verticales azules).

```
p_valor = (sum(abs(distribucion_permutaciones) > abs(dif_obs)))/9999
p_valor
```

```
[1] 0.1018102
```

El p-valor obtenido muestra una evidencia muy débil en contra de la hipótesis nula de que el tratamiento no tiene efecto, sugiriendo que asistir a clases extra escolares no mejora la capacidad de empatía para los estudiantes que formaron parte del experimento.

Aunque siendo estrictos, no se podría extrapolar a la población de estudiantes ya que la selección de sujetos no fue aleatoria.

A modo de comprobación, siendo los tamaños muestrales mayores que 30 observaciones, el test de la t-Student debería dar un resultado similar.

```
t.test(puntuacion ~ grupo, data=datos, var.equal=TRUE)
```

```
    Two Sample t-test

data:  puntuacion by grupo
t = 1.6379, df = 354, p-value = 0.1023
alternative hypothesis: true difference in means between group control and group tratamiento is not equa
95 percent confidence interval:
 -0.3425852  3.7558562
sample estimates:
    mean in group control mean in group tratamiento
                 50.72559                  49.01896
```

Y se obtiene que los p-valores son semejantes.

Tablas de contingencia

```
tabla = as.table(rbind(c(178, 138, 108), c(570, 648, 442),
        c(138,252, 252)))

(res = chisq.test(tabla))
```

```
    Pearson's Chi-squared test

data:  tabla
X-squared = 69.157, df = 4, p-value = 3.42e-14
```

Recuentos esperados

```
res$expected
```

```
          A        B        C
A 137.8078 161.4497 124.7425
B 539.5304 632.0910 488.3786
C 208.6618 244.4593 188.8789
```

Con la función `chisq.test` también se pueden hacer contrastes por simulación Montecarlo, es decir, calculando el estadístico de la chi cuadrado para todas las posibles tablas con las mismas sumas marginales por filas y columnas de la tabla original.

```
chisq.test(tabla, sim=T, B=2000)
```

```
    Pearson's Chi-squared test with simulated p-value (based on 2000
    replicates)

data:  tabla
X-squared = 69.157, df = NA, p-value = 0.0004998
```

Uso de la librería `coin`

Ejemplo simulado test permutaciones usando `coin`

```
library(coin)

# Simulas unos datos
niveles = c(40, 57, 45, 55, 58, 57, 64, 55, 62, 65)
trata = factor(c(rep("A",5), rep("B",5)))
eso = data.frame(trata, niveles)
```

Test clásico t Student

```
t.test(niveles ~ trata, data=eso, var.equal=TRUE)
```

```
    Two Sample t-test

data:  niveles by trata
t = -2.345, df = 8, p-value = 0.04705
alternative hypothesis: true difference in means between group A and group B is not equal to 0
95 percent confidence interval:
 -19.0405455  -0.1594545
sample estimates:
mean in group A mean in group B
           51.0            60.6
```

Número posible de permutaciones

```
choose(10,5)
```

```
[1] 252
```

```
oneway_test(niveles~trata, data=eso, distribution="exact")
```

```
    Exact Two-Sample Fisher-Pitman Permutation Test

data:  niveles by trata (A, B)
Z = -1.9147, p-value = 0.07143
alternative hypothesis: true mu is not equal to 0
```

Ejemplo sobre difusión de agua vía membrana fetal usando `coin`

```
agua_entra = data.frame(
pd = c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46, 1.15,
0.88, 0.90, 0.74, 1.21),
edad=factor(c(rep("A termino", 10),
rep("12-26 Semanas", 5)))
)

choose(15,5)
```

```
[1] 3003
```

```
# Test de permutaciones
eso = oneway_test(pd ~ edad, data=agua_entra,
distribution="exact")

print(eso)
```

```
    Exact Two-Sample Fisher-Pitman Permutation Test

data:  pd by edad (12-26 Semanas, A termino)
Z = -1.5225, p-value = 0.1319
alternative hypothesis: true mu is not equal to 0
```

```
# Test de permutaciones aproximado
eso2 = oneway_test(pd ~ edad, data=agua_entra,
distribution=approximate(nresample=1000))

print(eso2)
```

```
    Approximative Two-Sample Fisher-Pitman Permutation Test

data:  pd by edad (12-26 Semanas, A termino)
Z = -1.5225, p-value = 0.128
alternative hypothesis: true mu is not equal to 0
```

```
pvalue(eso)
```

```
[1] 0.1318681
```

```
pvalue(eso2)
```

```
[1] 0.128
99 percent confidence interval:
 0.1021182 0.1575007
```

Ejemplo ANOVA usando la librería `coin`

ANOVA unifactorial no paramétrico

Ejemplo sobre la longitud de sardinas en función de ciertos factores ambientales.

```
library(coin)

peces = data.frame(
longi=c(46, 28, 46, 37, 32, 41, 42, 45, 38, 44, 42, 60,
32, 42, 45, 58, 27, 51, 42, 52, 38, 33, 26, 25, 28, 28,
26, 27, 27, 27,31, 30, 27, 29, 30, 25, 25, 24, 27, 30),
sitio = factor(c(rep("I", 10), rep("II", 10),
rep("III", 10), rep("IV", 10))))

# test de Kruskal-Wallis
kruskal.test(longi ~ sitio, data=peces)
```

```
    Kruskal-Wallis rank sum test

data:  longi by sitio
Kruskal-Wallis chi-squared = 22.852, df = 3, p-value = 4.335e-05
```

```
# test de Kruskal-Wallis por permutaciones
kwa = kruskal_test(longi ~ sitio,
data=peces, distribution=approximate(nresample=1000))

print(kwa)
```

```
    Approximative Kruskal-Wallis Test

data:  longi by sitio (I, II, III, IV)
chi-squared = 22.852, p-value < 0.001
```

```
pvalue(kwa)
```

```
[1] <0.001
99 percent confidence interval:
 0.000000000 0.005284306
```

```
pairwise.wilcox.test(peces$longi, peces$sitio,
                 p.adjust.method = "BH")
```

```
    Pairwise comparisons using Wilcoxon rank sum test with continuity correction

data:  peces$longi and peces$sitio

    I      II     III
II  0.3054 -      -
III 0.0021 0.0021 -
IV  0.0021 0.0021 1.0000

P value adjustment method: BH
```

```
# Por permutaciones
library(rcompanion)

pairwisePermutationTest(longi ~ sitio,
data=peces, teststat="quadratic", method = "fdr")
```

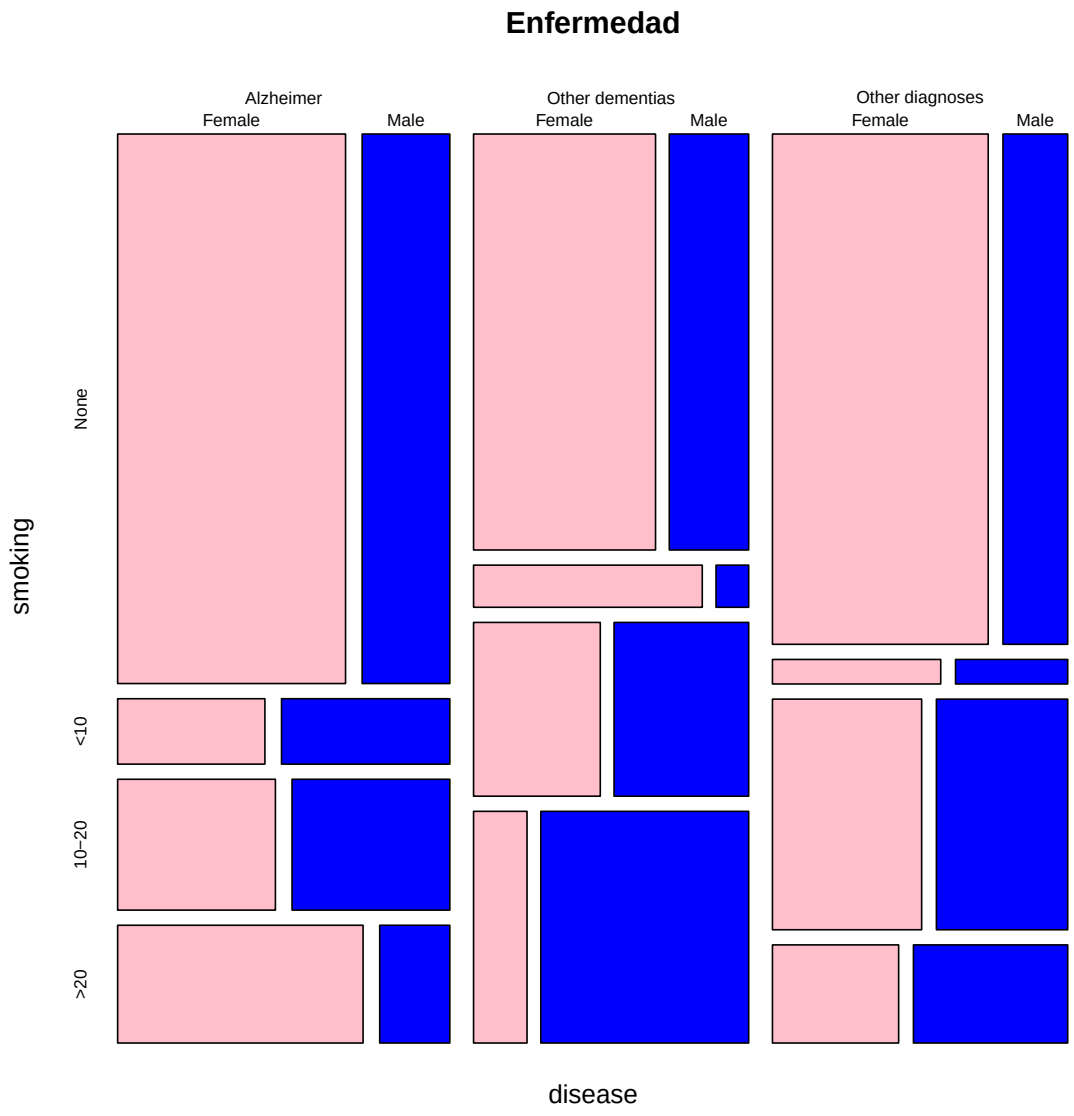
```
    Comparison   Stat   p.value p.adjust
1   I - II = 0  1.744    0.1866 0.223900
2  I - III = 0  10.91 0.0009543 0.001899
3   I - IV = 0  12.32  0.000447 0.001899
4 II - III = 0  10.39  0.001266 0.001899
5  II - IV = 0  11.14 0.0008462 0.001899
6 III - IV = 0 0.2298    0.6317 0.631700
```

Ejemplo tablas de contingencia usando la librería `coin`

```
head(alzheimer)
```

```
  smoking   disease gender
1    None Alzheimer Female
2    None Alzheimer Female
3    None Alzheimer Female
4    None Alzheimer Female
5    None Alzheimer Female
6    None Alzheimer Female
```

```
mosaicplot(~disease + smoking + gender, data=alzheimer, main="Enfermedad",
col=c("pink","blue"), off=c(5,5,5,5))
```



Contraste de independencia

```
it_alz = independence_test(disease ~ smoking | gender,
data=alzheimer, distribution="approximate")

print(it_alz)
```

```
    Approximative General Independence Test

data:  disease by
```

```
     smoking (None, <10, 10-20, >20)
     stratified by gender
maxT = 3.5106, p-value = 0.005
alternative hypothesis: two.sided
```

```
females = alzheimer$gender == "Female"
males = alzheimer$gender == "Male"

pvalue(independence_test(disease ~ smoking, data=alzheimer,
subset=females, distribution="approximate"))
```

```
[1] 0.2508
99 percent confidence interval:
 0.2397055 0.2621330
```

```
pvalue(independence_test(disease ~ smoking,
data=alzheimer, subset=males, distribution="approximate"))
```

```
[1] <1e-04
99 percent confidence interval:
 0.0000000000 0.0005296914
```