Primeros pasos con R y RStudio

Tipos de variables en R

En estadistica tratamos con variables cuantitativas continuas y discretas y variables cualitativas nominales y ordinales veamos como codificar un ejemplo en R.

Datos numéricos.

Suponer que los datos 64, 79, 69, 77, 91 representan el peso en kg de 5 personas y queremos acumularlos en una variable numérica que se llame peso. Escribiremos lo siguiente:

```
peso <- c(64, 79, 69, 77, 91)
peso
## [1] 64 79 69 77 91
```

Vamos ahora a introducir la altura de cinco personas en metros

```
altura <- c( 1.55, 1.78, 1.67, 1.78, 1.93 )
```

Datos cualitativos.

En general para trabajar con datos cualitativos en R, tendremos que definir las variables de tipo factor que contienen además de los datos, las distintas modalidades o niveles que estas pueden tomar.

Ahora queremos introducir el sexo de estas cinco personas. Tenemos los datos *mujer, hombre, mujer, hombre, hombre*. Para que el valor de una variable sea texto (cadena de caracteres) hay que escribirlo entre comillas. Así podríamos introducir los datos anteriores como:

```
sexo <- c( "mujer", "hombre", "mujer", "hombre", "hombre" )
sexo
## [1] "mujer"  "hombre" "mujer"  "hombre" "hombre"
```

podemos transformar esta variable en una variable de tipo factor con:

```
sexo <- as.factor(sexo)
sexo
## [1] mujer  hombre mujer  hombre hombre
## Levels: hombre mujer
```

Notar como ademas de los valores de la variable me vienen especificados las modalidades o niveles `levels`.

Otra manera mas segura de obtener los resultados es especificar nosotros los niveles. Esto nos asegura que estén todos los niveles que interesan y en el orden adecuado (caso de que el orden importe)

```
sexo <- factor(sexo, levels = c("hombre", "mujer"))
sexo
## [1] mujer  hombre mujer  hombre hombre
## Levels: hombre mujer
```

En ocasiones es conveniente utilizar un "codigo numérico" para los distintos niveles (modalidades) de las variables cualitativas `levels` y asignar etiquetas `labels` a los niveles.

El grupo sanguíneo es un factor con 4 niveles. Por ejemplo queremos codificar el grupo sanguíneo con números de 1 a 4 donde 1 es A, 2 es B, 3 es AB y 4 es O. Ahora si mis cinco datos tienen grupo sanguíneo *A, B, A ,A, O*, puedo introducirlos usando los niveles antes descritos con las siguientes ordenes.

```r
gruposang <- c(1, 2, 1, 1, 4)
gruposang <- factor(gruposang,
                    levels = 1:4,
                    labels = c("A","B","AB","O"))
gruposang
## [1] A  B  A  A  O
## Levels: A B AB O
```

En resumen, para trabajar con variables cualitativas en R, es conveniente declararla como un factor que tendrá:

- *datos* numéricos o caracteres. En el ejemplo anterior los que había en gruposang
- *niveles* (`levels`) pueden ser numeros o cadenas de caracteres en este caso: 1,2,3 y 4
- *etiquetas* (`labels`) siempre cadenas de caracteres, en este ejemplo *A*, *B*, *AB*, *O*.

Cuando los niveles sean ya cadenas de caracteres no hará falta especificar las etiquetas.

Variables Cualitativas Ordinales I

Podemos especificar que los niveles están ordenados en un factor que represente una variable ordinal. Imaginar que tenemos una variable que nos dice como se encuentran fisicamente las personas entre bien regular y mal.

```
estfis <- c("bien","bien","mal","regular", "regular")
```

Podemos definir un factor ordenado mediante:

```
estfis <- factor(estfis,
                 levels=c("mal", "regular", "bien"),
                 ordered= TRUE)
estfis
## [1] bien    bien    mal     regular regular
## Levels: mal < regular < bien
```

Vemos como ahora los niveles aparecen ordenados.

Data frames

Los data frames nos permiten codificar datos agrupados por individuos, por ejemplo supongamos que las variables peso, altura, sexo y grupo sanguíneo que hemos introducido antes corresponden a la mismas personas. Es decir el la primera persona pesa 64.6 kg mide 155cm, es mujer y tiene grupo sanguíneo A y así con las demás posiciones. Podemos agrupar todos esos datos en R usando un data frame.

```
misdatos <- data.frame(peso, altura, sexo, gruposang, estfis)
misdatos
##   peso altura   sexo gruposang  estfis
## 1   64   1.55  mujer         A    bien
## 2   79   1.78 hombre         B    bien
## 3   69   1.67  mujer         A     mal
## 4   77   1.78 hombre         A regular
## 5   91   1.93 hombre         O regular
```

Observar que en la primera columna aparece un número que simplemente va contando el número de individuos que tiene mi muestra.

La orden `str(objeto)` nos da información sobre un objeto de R. Es especialmente útil para saber lo que hay dentro de un data.frame

```
str(misdatos)
## 'data.frame':    5 obs. of  5 variables:
##  $ peso      : num  64 79 69 77 91
##  $ altura    : num  1.55 1.78 1.67 1.78 1.93
##  $ sexo      : Factor w/ 2 levels "hombre","mujer": 2 1 2 1
##  $ gruposang: Factor w/ 4 levels "A","B","AB","O": 1 2 1 1
##  $ estfis    : Ord.factor w/ 3 levels "mal"<"regular"<..: 3
```

Si tenemos un data frame y queremos usar una de las variables que contiene escribiremos nombredataframe$nombre variable. Por ejemplo:

```
misdatos$altura
## [1] 1.55 1.78 1.67 1.78 1.93
```

Para guardar datos almacenados en un data frame como un fichero cvs (para leerlos con otro progama despues) podemos hacerlo con la orden:

```
write.table(misdatos, file= "misdatos.csv",
            sep=";", dec=",")
```

Podemos usar `read.table` para leer un data frame que hayamos guardado con el comando `write.table`

```
misdatosbis  <-  read.table("misdatos.csv",
                           header=TRUE, sep=";", dec=",")
misdatosbis
##   peso altura   sexo gruposang  estfis
## 1   64   1.55  mujer         A    bien
## 2   79   1.78 hombre         B    bien
## 3   69   1.67  mujer         A     mal
## 4   77   1.78 hombre         A regular
## 5   91   1.93 hombre         O regular
str(misdatosbis)
## 'data.frame':    5 obs. of  5 variables:
##  $ peso     : int  64 79 69 77 91
##  $ altura   : num  1.55 1.78 1.67 1.78 1.93
##  $ sexo     : Factor w/ 2 levels "hombre","mujer": 2 1 2 1
##  $ gruposang: Factor w/ 3 levels "A","B","O": 1 2 1 1 3
##  $ estfis   : Factor w/ 3 levels "bien","mal","regular": 1
```

Por defecto nos ha tomado las columnas de caracteres como factores pero hemos perdido el orden. ¿Qué peligro tiene que R declare automáticamente cadenas de caracteres como factores?

Importar datos.

Vamos a ver como importar datos guardados desde una hoja de Excel y mas en general desde un archivo .csv (*comma separated values*)

Para importar un data frame desde Excel nuestra hoja de cálculo debe a ser posible:

- tener los datos de cada variable en una columna,
- los nombres de la variable en la primera fila
- los datos en las filas siguentes.

Hemos de guardar el fichero Excel como un fichero csv (primero usar guardar como...
y despues seleccionar en guardar como tipo CSV(delimitado por comas) (*.csv)

y leerlo con el formato adecuado entre los descritos arriba

(en el caso de Excel en castellano, tenemos los decimales separados por comas y por tanto necesitamos las opciones `sep=";", dec=","`

En general podemos importar datos separados por comas (extensión .csv) guardados desde Excel en español con:

```
misdatos <- read.table("mifichero.csv",
                       header=TRUE, sep=";", dec=",")
```

Que corresponde a datos separados por punto y coma con decimales separados por comas (que es como exporta excel los ficheros de datos "en español")

Una manera rápida de importar datos es seleccionar las casillas que nos interesen, dar a copiar y después en R usar.

```
datos <- read.delim("clipboard",
                    header = TRUE, dec = ",", check.names=T)
```

Pero solo lo recomendamos para hacer alguna prueba rápida ya que no es reproducible (no queda constancia de lo que hemos seleccionado).

En el archivo excel `gimnasio.xlsx` encontras los datos de peso altura y ejercicio de 20 alumnos. En la columna ejercicio hemos usado los valores 1 para Nunca, 2 para Ocasional y 3 para Frecuente.

Una vez guardado como .csv desde excel lo leemos con:

```
gimnasio <- read.csv("gimnasio.csv",
                     header=TRUE, sep=";",dec=",")
```

Dos ordenes muy útiles para hacernos una idea de lo que hay en un dataframe que hemos creado son `head(nombredataframe)` y `tail(nombredataframe)` que nos da los primeros y últimos valores del dataframe respectivamente. Por ejemplo

```
head(gimnasio)
##   peso altura ejercicio
## 1   64  1,82          1
## 2   78   1,7          2
## 3   83  1,87          3
## 4   66  1.56          1
## 5   98   1,8          1
## 6   63  1,73          3
```

Con cualquier objeto de R podemos usar `str(objeto)` que nos dice que tipo de objetos hay almacenados

```
str(gimnasio)
## 'data.frame':    20 obs. of  3 variables:
##  $ peso     : num  64 78 83 66 98 63 88 78 76 67 ...
##  $ altura   : Factor w/ 15 levels " 1,43 "," 1,56 ",..: 10
##  $ ejercicio: num  1 2 3 1 1 3 2 1 1 1 ...
```

Vemos que hay que transformar la última columna en un factor ordenado.

```r
gimnasio$ejercicio <- factor(gimnasio$ejercicio,
            levels=1:3,
            labels=c("Nunca", "Ocasional", "Frecuente"),
            ordered= TRUE)

str(gimnasio)
## 'data.frame':    20 obs. of  3 variables:
##  $ peso     : num  64 78 83 66 98 63 88 78 76 67 ...
##  $ altura   : Factor w/ 15 levels " 1,43 "," 1,56 ",..: 10
##  $ ejercicio: Ord.factor w/ 3 levels "Nunca"<"Ocasional"<.
```

Notar que no puedo acceder directamente a las variables de dentro del data frame. Por ejemplo para calcular la media de una variable se usa `mean(nombrevariable)`

```
mean(peso)
## [1] 76
```

Nos da la media de peso... **del conjunto de 5 datos anterior.**

En general usaremos `nombredataframe$nombrevariable` o `nombredataframe[ ,indice de la variable]` para acceder a una variable dentro de un dataframe.

```
mean(gimnasio$peso)
## [1] 76.85
mean(gimnasio[ , 1])
## [1] 76.85
```

Otra manera de acceder dentro de un data frame es con la orden `with(nombredataframe, operacion)`

```
with(gimnasio, mean(peso))
## [1] 76.85
```

Estadística básica

Veamos como

- obtener los distintos estadísticos descriptivos usando R.
- hacer unos gráficos básicos
- inferencia univariante.

Variables cuantitativas

Por ejemplo para los datos de peso introducidos anteriormente tenemos:

```
mean(gimnasio$peso)
## [1] 76.85
median(gimnasio$peso)
## [1] 77
sd(gimnasio$peso)
## [1] 13.49181
var(gimnasio$peso)
## [1] 182.0289
```

Nos dan la media, mediana, desviación típica y varianza respectivamente.

Para obtener los cuartiles, o en general cualquier percentil podemos usar la orden `quantile(variable, probs=c(percentiles))`

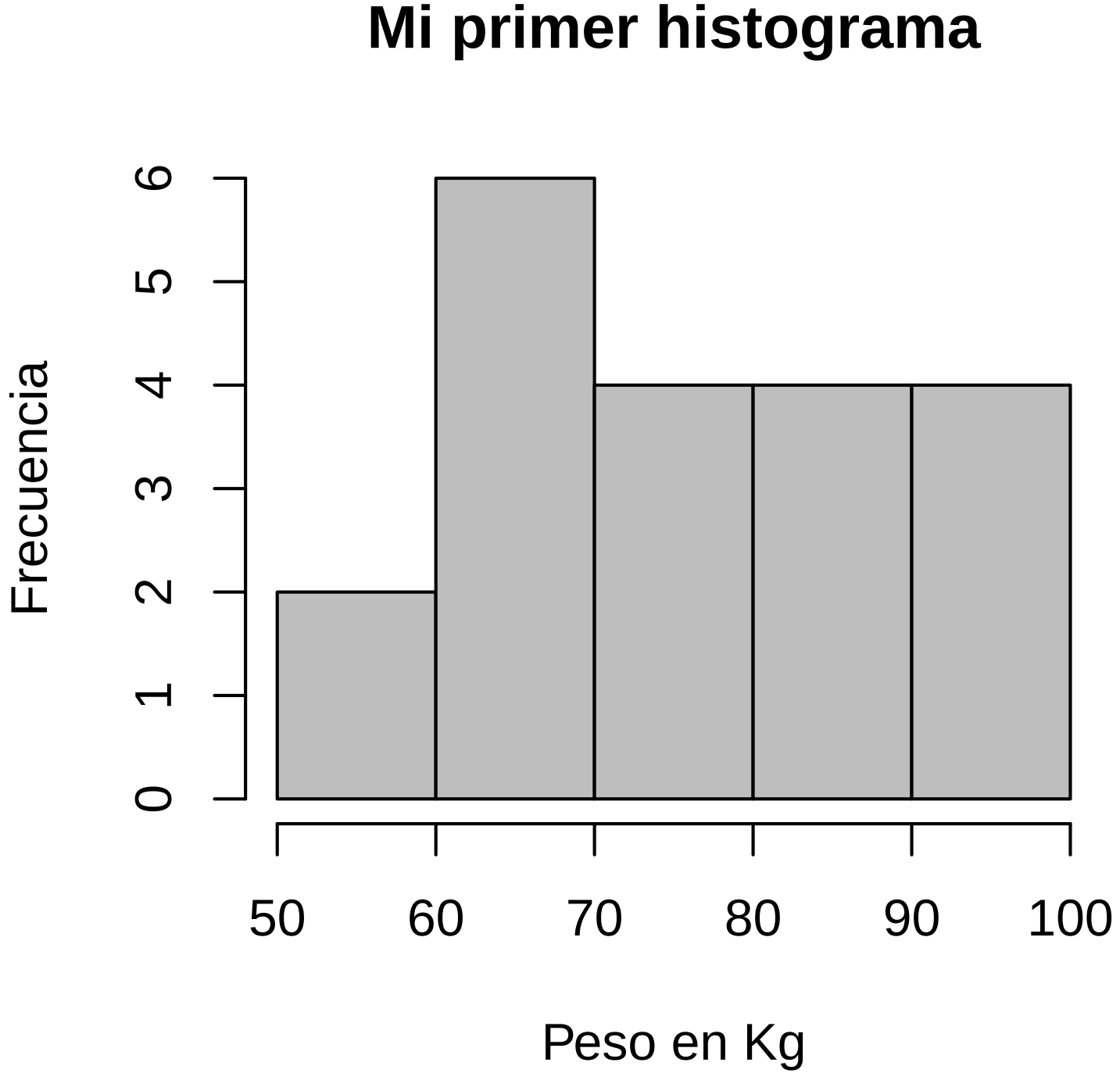
```
quantile(gimnasio$peso)
##    0%   25%   50%   75%  100%
## 53.00 66.75 77.00 86.50 99.00
quantile(gimnasio$peso,probs=c(0.05, 0.95))
##    5%   95%
## 56.80 98.05
```

Podemos obtener un resumen (datos mayor y menor, cuartiles mediana y media para variables cuantitativas o una tabla de frecuencias parar variables cualitativas) con

```
summary(gimnasio$peso)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   53.00   66.75   77.00   76.85   86.50   99.00
```

Veamos algunos gráficos útiles para variables cuantitativas:

```
hist (gimnasio$peso,
      main="Mi primer histograma",
      xlab="Peso en Kg", ylab="Frecuencia",
      col="grey")
```


Mi primer histograma
Frecuencia
0
1
2
3
4
5
6
50
60
70
80
90
100
Peso en Kg

```r
boxplot (gimnasio$peso,
         main="Diagrama de cajas",
         ylab="Peso en Kg")
```

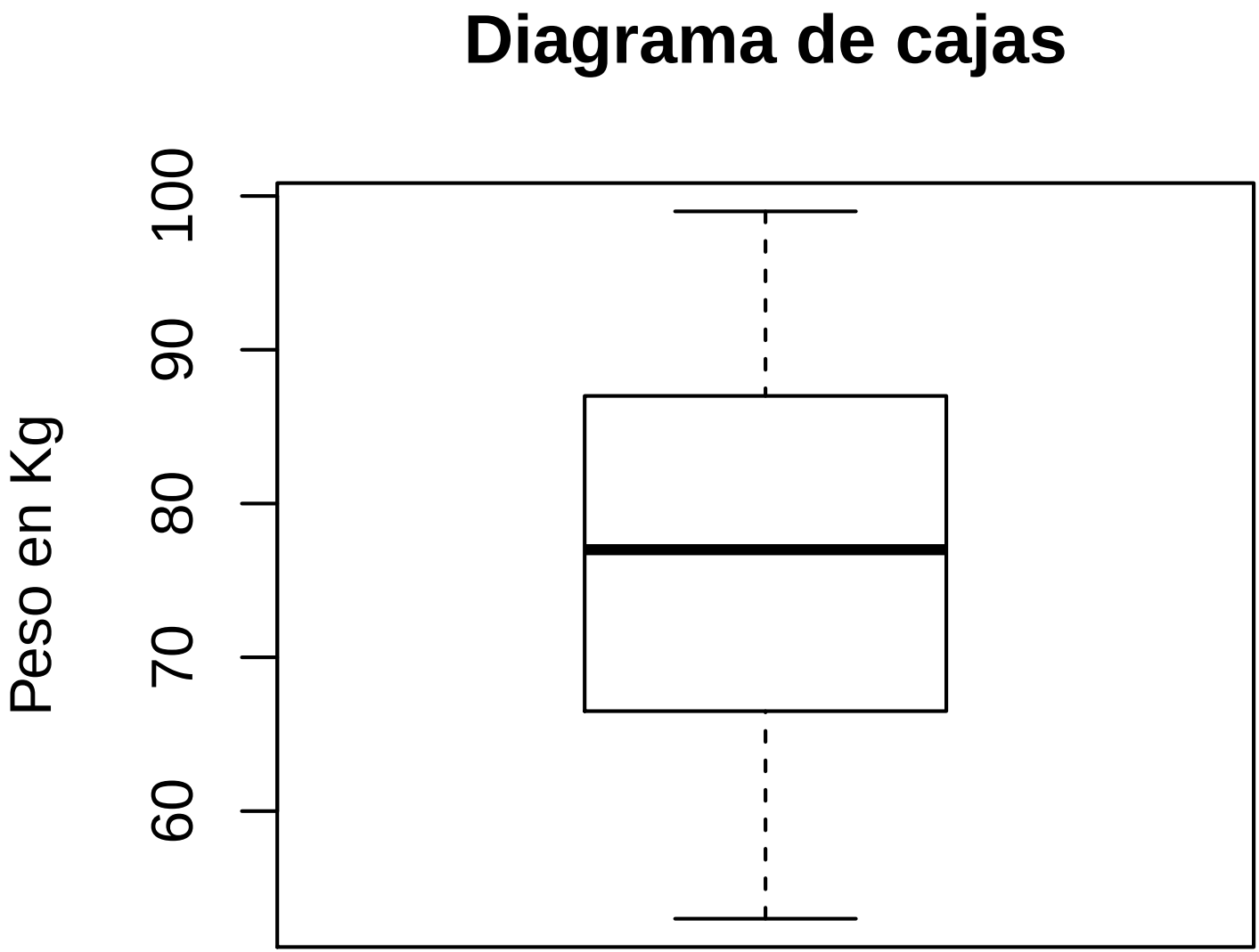
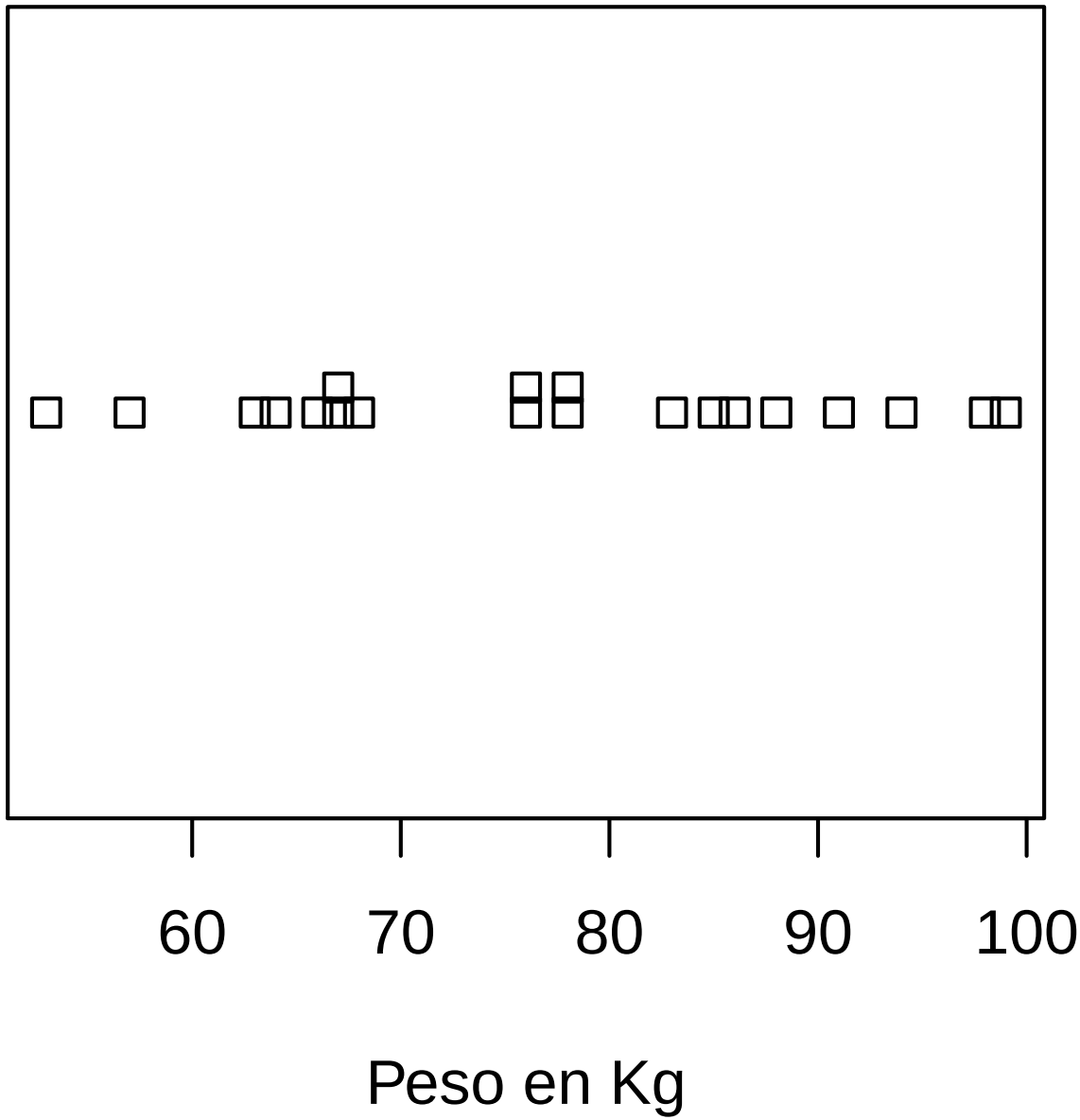


Diagrama de cajas
Peso en Kg
60
70
80
90
100

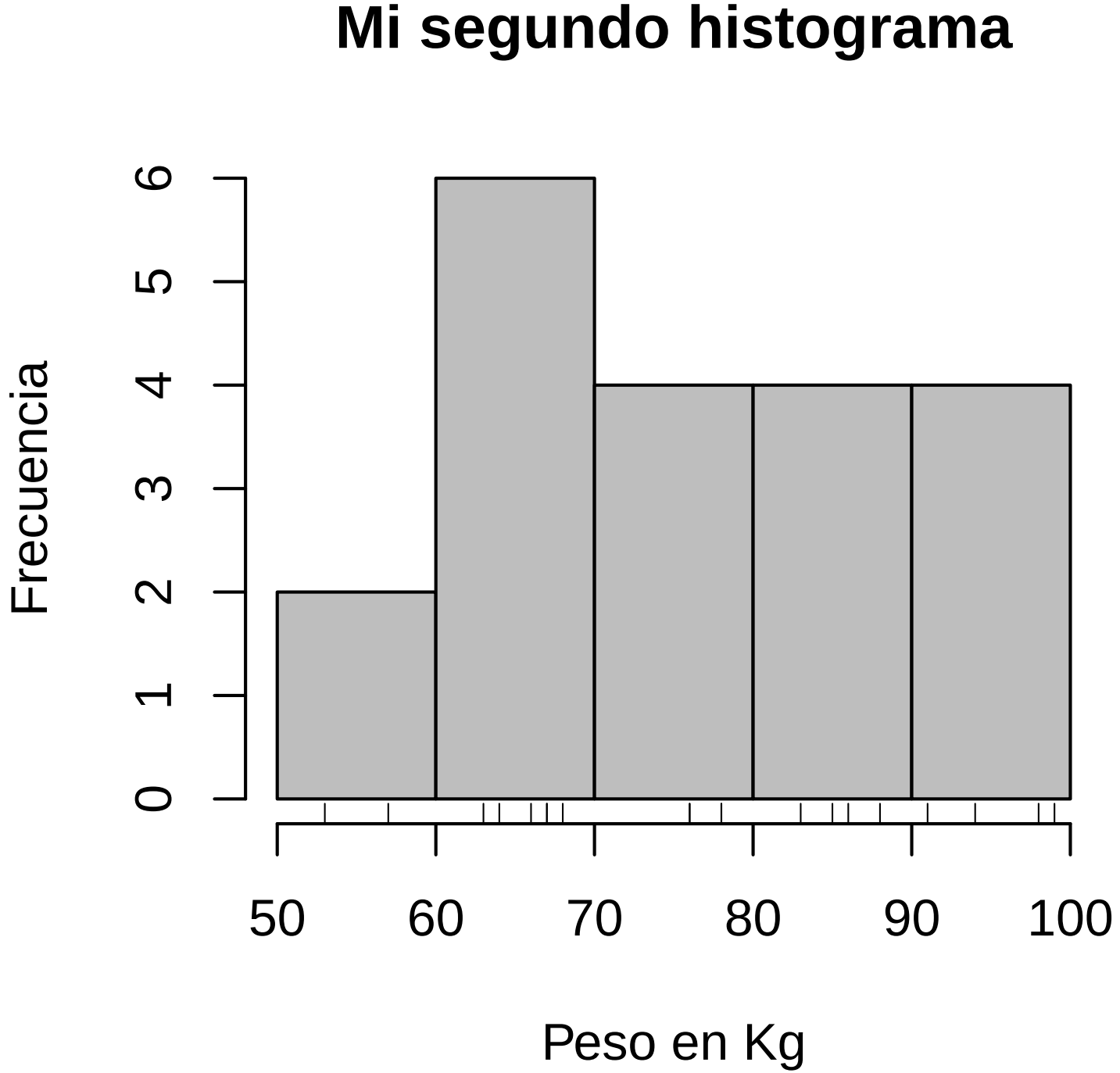
```r
stripchart(gimnasio$peso,
           main="Stripchart",
           method="stack",
           xlab="Peso en Kg")
```

Stripchart



Podemos añadir los datos a un histograma con:

```
hist (gimnasio$peso,
      main="Mi segundo histograma",
      xlab="Peso en Kg", ylab="Frecuencia",
      col="grey")
rug(gimnasio$peso)
```


Mi segundo histograma
Frecuencia
0
1
2
3
4
5
6
50
60
70
80
90
100
Peso en Kg

y a un boxplot con:

```
boxplot (gimnasio$peso,
         main="Diagrama de cajas",
         ylab="Peso en Kg",
         outline=FALSE   #Esta opción quita los outlayers de
         )

stripchart(gimnasio$peso,
           method="jitter",
           vertical=TRUE,
           pch=20,
           col="grey",
           add=TRUE)
```

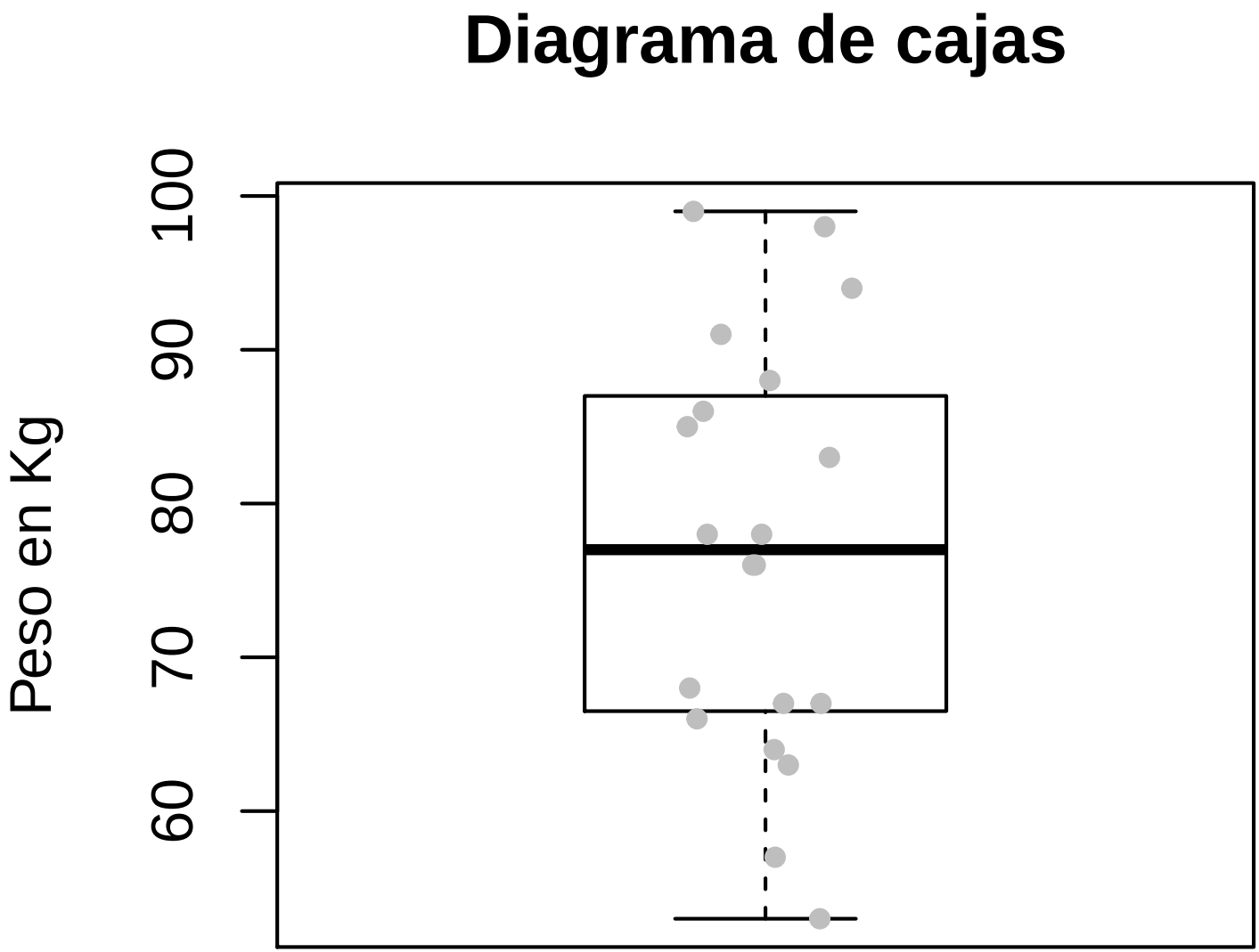


Diagrama de cajas
Peso en Kg
60
70
80
90
100

Variables Cualitativas

Para variables cualitativas no tiene sentido hacer los estadísticos anteriores ni los gráficos.

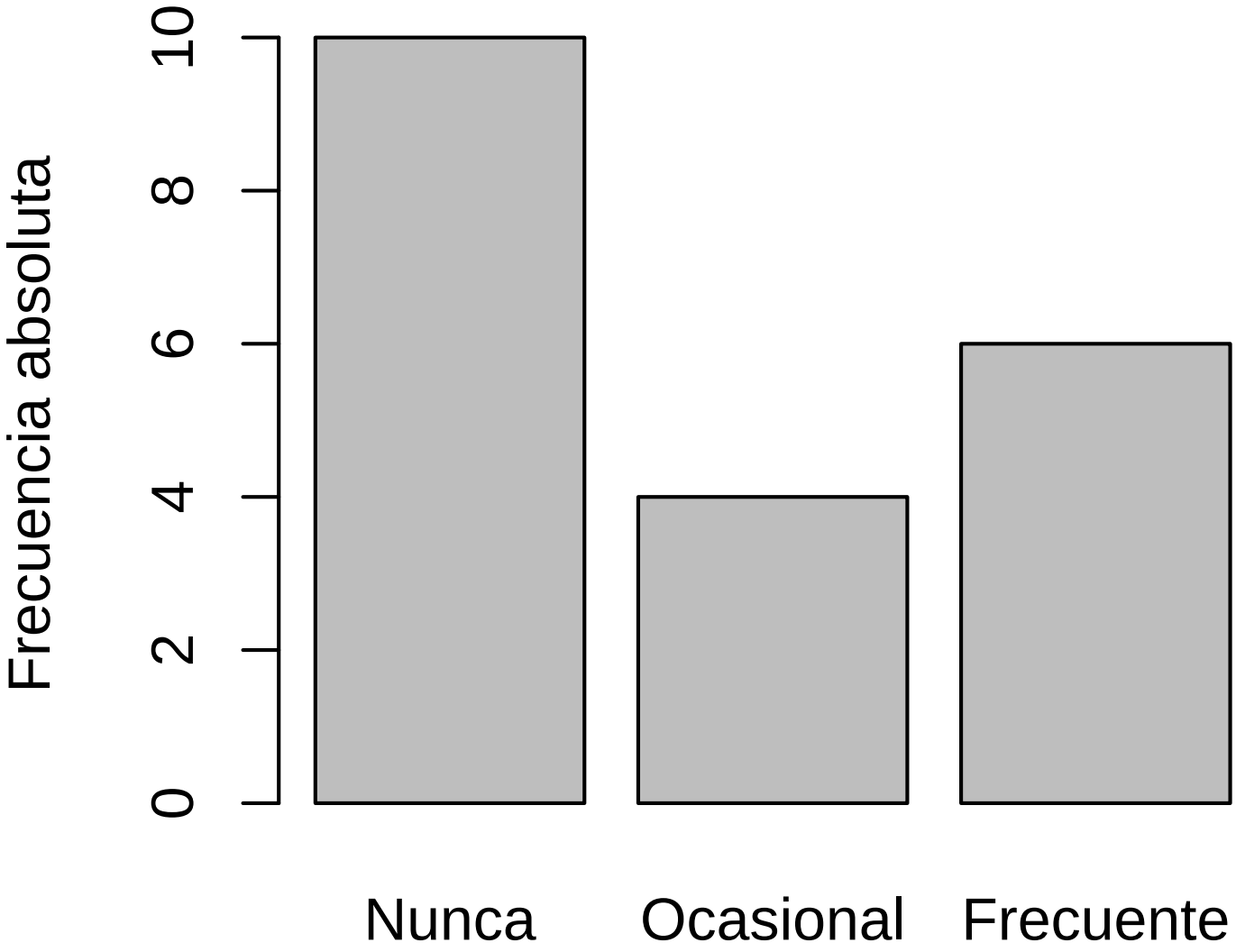
Podemos hacer una tabla de frecuencias con la orden table

```
table(gimnasio$ejercicio)
##
##     Nunca Ocasional Frecuente
##        10         4         6
# para obtener frecuencias relativas  dividimos por el
# número de datos, que obtendremos con la orden
# nrow(dataframe)  o length(vector)

table(gimnasio$ejercicio)/nrow(gimnasio)
##
##     Nunca Ocasional Frecuente
##       0.5       0.2       0.3
```

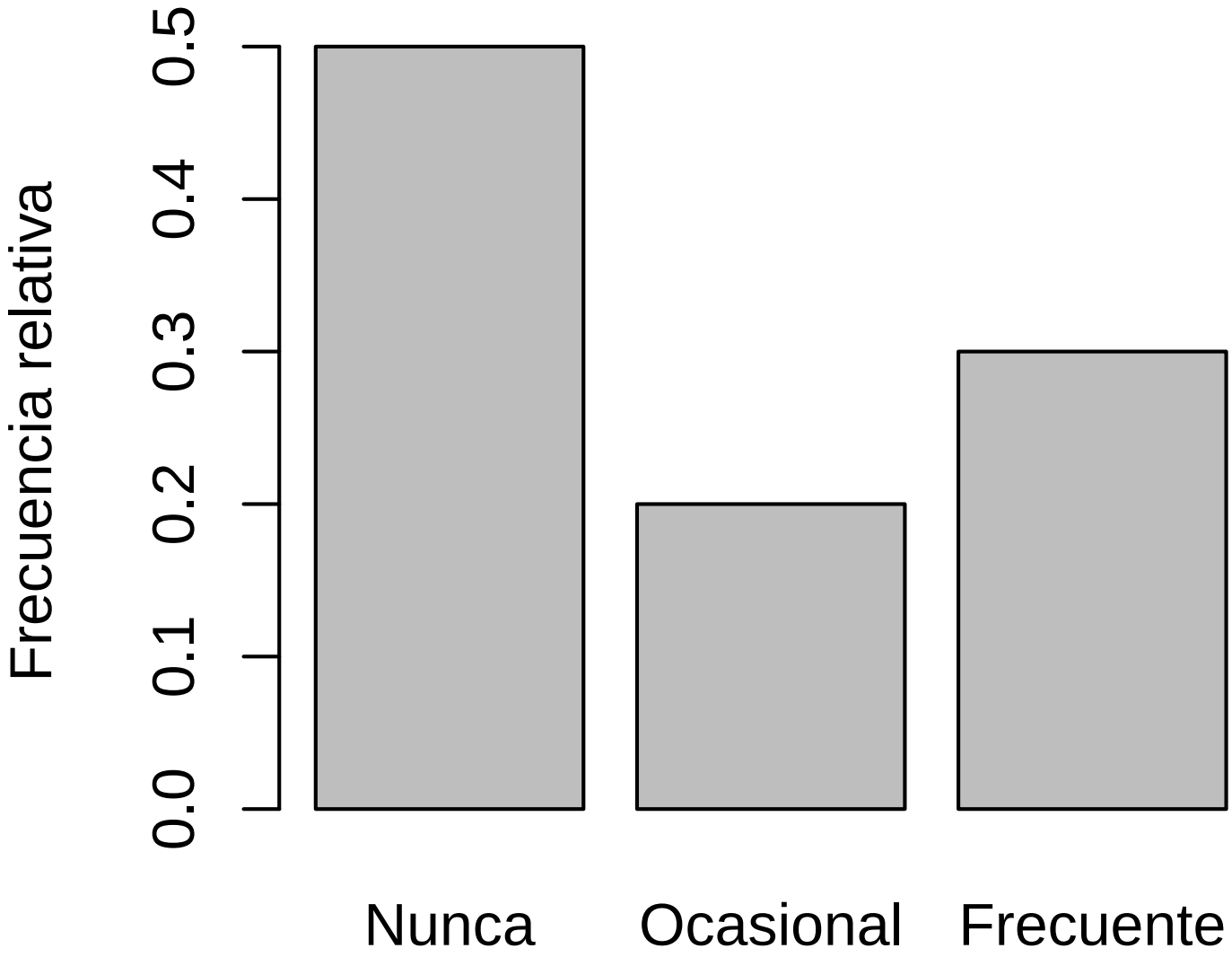
```r
summary(gimnasio$ejercicio)
##     Nunca Ocasional Frecuente
##        10         4         6
```

Ahora podemos hacer gráficos usando esa tabla de frecuencias. Por ejemplo:

```
barplot(table(gimnasio$ejercicio),
        ylab="Frecuencia absoluta")
```


Frecuencia absoluta
0
2
4
6
8
10
Nunca
Ocasional
Frecuente

```r
barplot(table(gimnasio$ejercicio)/nrow(gimnasio),
        ylab="Frecuencia relativa")
```



Nos da un diagrama de barras.

Intervalos de confianza y tests de hipótesis. (Una y dos medias)

Veremos los procedimientos para calcular intervalos de confianza para una y dos medias usando r. En ambos casos usaremos la orden `t.test`

Estimación de una media.

R nos permite estimar mediante intervalo de confianza, la media de una población a partir de los datos de una muestra.

La estimación de la media la realizamos con:

```
t.test(nombredatos, mu=mediacontraste, conf.level=nivelconf)
```

Ejemplo: Queremos comprobar si es cierto que la ingesta diaria energética en KJ para 11 mujeres con una determinada dieta se desvía significativamente de la cantidad recomendada de 7725 KJ.

La variable aleatoria que estudiamos es *ingesta diaria energética en KJ para mujeres con una determinada dieta*.

```
ingesta <- c(5260,5470,5640,6180,6390,
             6515,6805,7515,7515,8230,8770)
t.test(ingesta, mu=7725,conf.level=0.95)
##
##  One Sample t-test
##
## data:  ingesta
## t = -2.8208, df = 10, p-value = 0.01814
## alternative hypothesis: true mean is not equal to 7725
## 95 percent confidence interval:
##  5986.348 7520.925
## sample estimates:
## mean of x
##  6753.636
```

Con un 95 % de confianza podemos afirmar que la media de la ingesta diaria energética en KJ para mujeres con esta dieta está entre 5986.348 kJ y 7520.925 KJ, y por tanto es inferior a la cantidad recomendada de 7725 KJ en al menos 204.075 KJ.

Si queremos extraer únicamente los datos del intervalo de confianza podemos hacer:

```
t.test(ingesta, mu=7725,conf.level=0.95)$conf.int
## [1] 5986.348 7520.925
## attr(,"conf.level")
## [1] 0.95
```

`t.test` hace por defecto, además del intervalo de confianza, un test de dos colas. El *p-valor* aparece en `p.value`. Para especificar el número de colas podemos usar la opción `alternative=` que puede tomar los valores:

1. `"two.sided"` para un test de dos colas
2. `"less"` para un test de una cola a la izda
3. `"greater"`

Así si en el ejemplo anterior quisieramos hacer un test de una cola a la izquierda escribiríamos:

```
t.test(ingesta, mu=7725,conf.level=0.95, alternative="less")
##
##  One Sample t-test
##
## data:  ingesta
## t = -2.8208, df = 10, p-value = 0.009069
## alternative hypothesis: true mean is less than 7725
## 95 percent confidence interval:
##      -Inf 7377.781
## sample estimates:
## mean of x
##  6753.636
```

Comparar el *p-valor* con el del ejemplo anterior.

Método no paramétrico. Contraste de wilcoxon. I

Podemos realizar el contraste de los rangos con signo de Wilcoxon utilizando `wilcox.test()`

```
wilcox.test(ingesta,mu=7725)
## Warning in wilcox.test.default(ingesta, mu = 7725): cannot
## with ties
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  ingesta
## V = 8, p-value = 0.0293
## alternative hypothesis: true location is not equal to 7725
```

Comparación de medias:

Ha habido una queja de que los medicos de cierta aseguradora pasa menos tiempo en media con los pacientes obesos. Tenemos los datos de una muestra en *tiempopeso.txt* donde la variable peso está codificada como *1=“no_obeso”*, *2 = “obeso”*. En primer lugar leemos los datos.

```
datosAseguradora <- read.table("tiempopeso.txt",
                               header=TRUE)
```

La variable Peso no es un factor pero querríamos que lo fuese. Definimos Peso como un factor.

```
datosAseguradora$Peso <- factor(datosAseguradora$Peso,
                                levels=1:2,
                                labels=c("no_obeso", "obeso"))
str(datosAseguradora)
## 'data.frame':    71 obs. of  2 variables:
##  $ Peso  : Factor w/ 2 levels "no_obeso","obeso": 1 1 1 1 1
##  $ Tiempo: int  15 15 45 40 45 20 40 30 40 30 ...
```

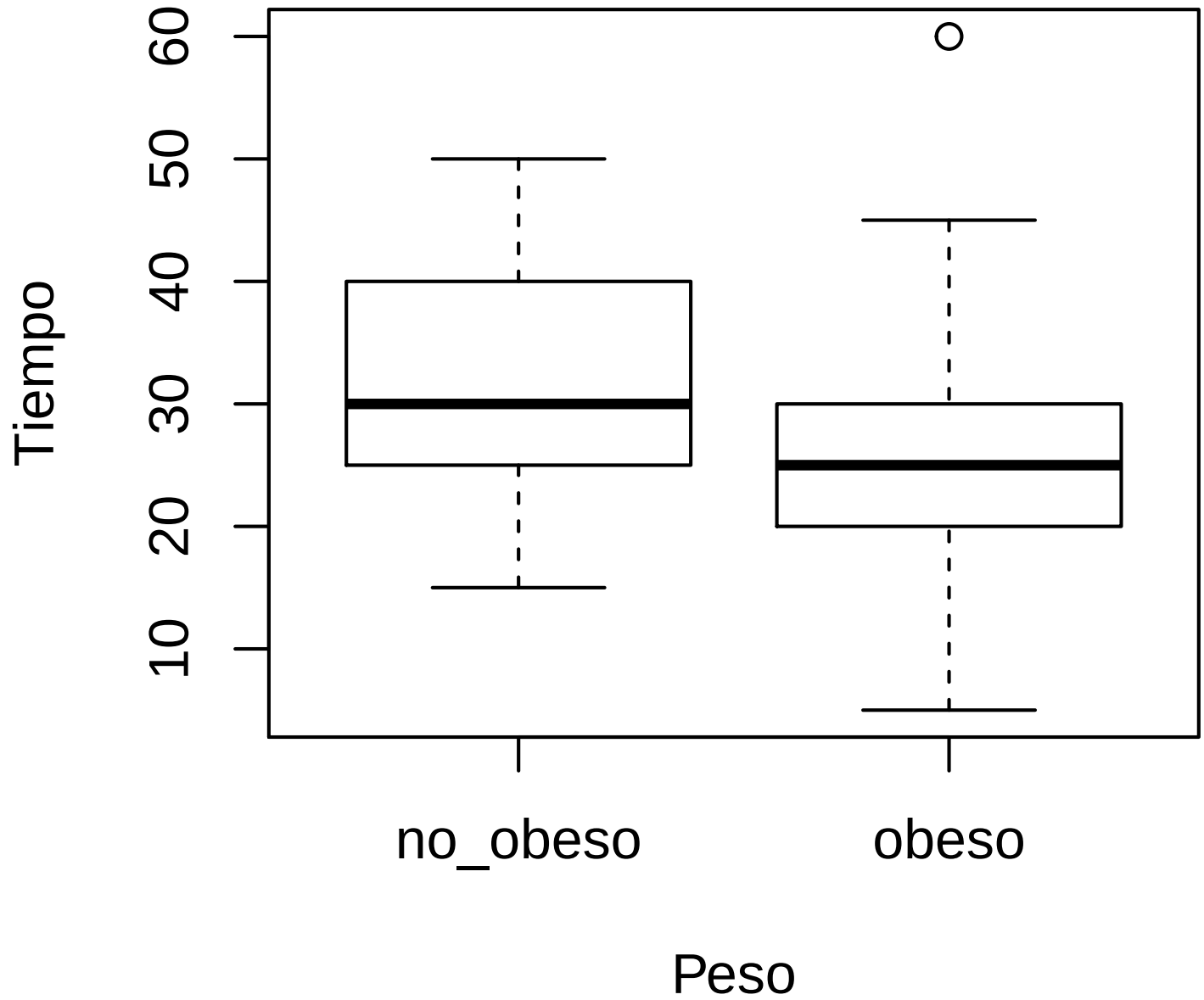
Comparamos los datos gráficamente con diagramas de cajas.

Nota: para conseguir la virgulilla ~ en Windows has de:

1. presionar la tecla `Alt`.
2. sin soltar `Alt` introducir el numero 126 en el teclado numérico.
3. soltar la tecla `Alt`.

para conseguir la virgulilla ~ en OSX has de presionar la tecla `alt` y la tecla `ñ` a la vez

```
boxplot(Tiempo~Peso, data=datosAseguradora)
```



En R la virgulilla `~`, se puede traducir como *según los valores de* por ejemplo, haz un boxplot de la variable Tiempo según los valores de la variable Peso.

Notar que no hemos necesitado escribir `datosAseguradora$Tiempo` y `datosaseguradora$Peso`. En las expresiones que usen la virgulilla, se puede especificar el data frame incluyendo la opción `data = nombredeldataframe`

¿Son las varianzas iguales?

```
var.test(Tiempo~Peso, data=datosAseguradora)
##
##  F test to compare two variances
##
## data:  Tiempo by Peso
## F = 1.0443, num df = 32, denom df = 37, p-value = 0.8931
## alternative hypothesis: true ratio of variances is not equa
## 95 percent confidence interval:
##  0.5333405 2.0797269
## sample estimates:
## ratio of variances
##           1.044316
```

El intervalo de confianza para el cociente de las varianzas nos muestra que este cociente puede valer uno y por tanto podemos considerar las variables iguales.

Ahora podemos calcular el intervalo de confianza para comparar las medias con:

```r
t.test(Tiempo~Peso, var.equal=TRUE, data=datosAseguradora)
##
##  Two Sample t-test
##
## data:  Tiempo by Peso
## t = 2.856, df = 69, p-value = 0.005663
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##   1.997955 11.255633
## sample estimates:
## mean in group no_obeso    mean in group obeso
##               31.36364                24.73684
```

Dos observaciones:

- por defecto `mu=0`, `conf.level=0.95`
- por defecto R hace el test de Welch para varianzas distintas

(si no ponemos nada va a tomar `var.equal=FALSE`).

La alternativa no paramétrica a un test de hipótesis para muestras equivalentes es el contraste de la suma de rangos de wilcoxon que es equivalente al test U de Mann-Whitney.

```
wilcox.test(Tiempo~Peso, data = datosAseguradora)
## Warning in wilcox.test.default(x = c(15L, 15L, 45L, 40L, 45
## compute exact p-value with ties
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Tiempo by Peso
## W = 866, p-value = 0.003985
## alternative hypothesis: true location shift is not equal to
```

Veamos un ejemplo con muestras emparejadas. Estudiamos el numero de accidentes de tráfico en 9 puntos negros antes y después de señalarlos como tales.

```
Antes <- c(3,3,2,2,4,3,2,2,1)
Despues <- c(3,2,1,1,2,1,1,1,2)
accidentes=data.frame(Antes, Despues)
rm(Antes,Despues)
```

Para hacer un test de muestras emparejadas, puedo o bien crear una nueva variable y aplicar un test de una sola variable.

```
Diferencia=accidentes$Antes-accidentes$Despues
t.test(Diferencia, mu=0, conf.level=0.95)
```

O directamente con la opción paired=TRUE.

```
with(accidentes, t.test(Antes,Despues, paired=TRUE,
                        conf.level=0.95))
##
##  Paired t-test
##
## data:  Antes and Despues
## t = 2.8737, df = 8, p-value = 0.02071
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  0.1755951 1.6021826
## sample estimates:
## mean of the differences
##                0.8888889
```

El método análogo no paramétrico se puede hacer con:

```
with(accidentes,
     wilcox.test(Antes,Despues, paired=TRUE))
## Warning in wilcox.test.default(Antes, Despues, paired = TRU
## exact p-value with ties
## Warning in wilcox.test.default(Antes, Despues, paired = TRU
## exact p-value with zeroes
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  Antes and Despues
## V = 32.5, p-value = 0.04007
## alternative hypothesis: true location shift is not equal to
```

Regresión y Correlación.

Regresión lineal simple

El ejemplo que vamos a ver estudia como varía la tasa de crecimiento de unas plantas (en mm/semana) con la humedad del suelo. Los datos están en un archivo llamado `plant.growth.rate.csv`

Ejercicio: Leer el archivo `plant.growth.rate.csv` en un data frame que llamaremos Estudio_Crecimiento

En primer lugar vamos a cambiar los nombres de la variable a `humedad_suelo` y `tasa_crecimiento`

```
names(Estudio_Crecimiento) <- c("humedad_suelo",
                                "tasa_crecimiento")
```

Ahora dibujaremos una nube de puntos para ver si podemos apreciar una tendencia lineal.

Es importante el orden de las variables:

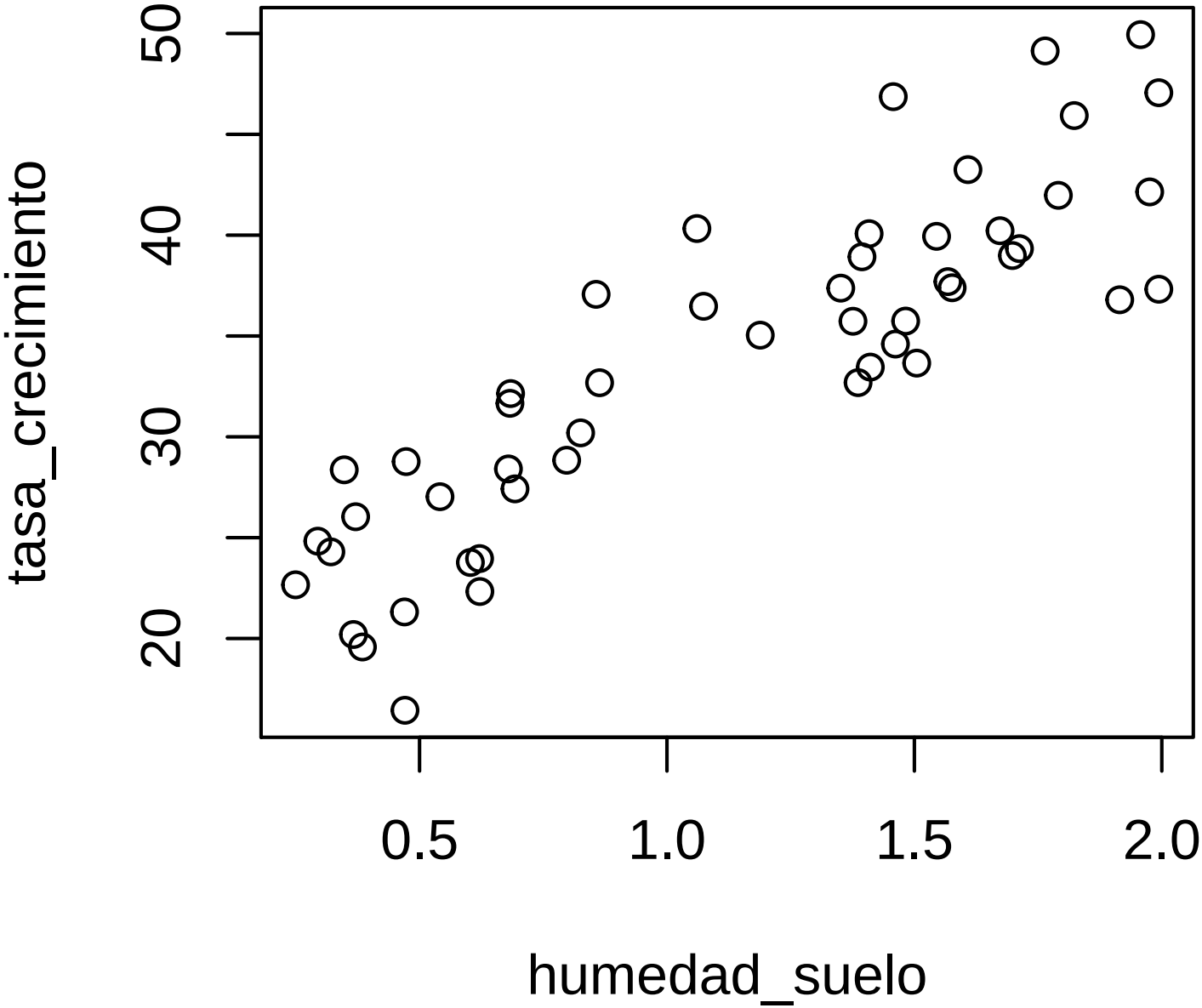
`tasa_crecimiento ~ humedad_suelo`

se entiende como *tasa de crecimiento en funcion de la humedad del suelo* donde

- `humedad_suelo` aparece en el eje X (variable independiente)
- `tasa_crecimiento` en el eje Y (variable dependiente)

En este caso:

```
plot(tasa_crecimiento ~ humedad_suelo,
     data = Estudio_Crecimiento)
```


tasa_crecimiento
20
30
40
50
0.5
1.0
1.5
2.0
humedad_suelo

Para calcular el coeficiente de correlación de Pearson de dos variables `x` e `y` usaremos `cor(x,y)`.

Como estamos trabajando con un data frame utilizaremos la orden with

```
with(Estudio_Crecimiento,
     cor(humedad_suelo, tasa_crecimiento) )
## [1] 0.8745262
```

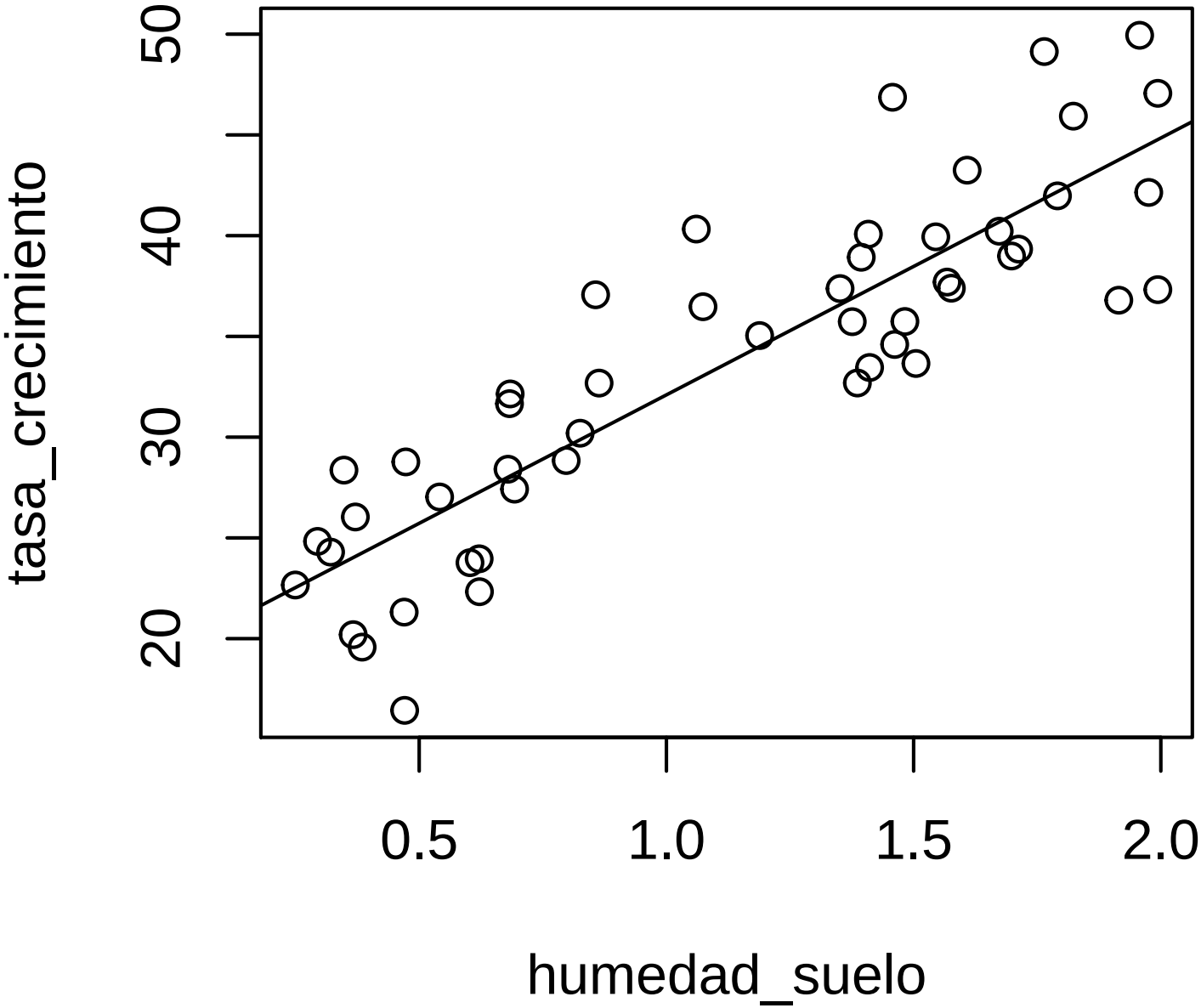
Obteniendo una buena correlación.

Para calcular los coeficientes de la recta de regresión de la tasa de crecimiento en funcion de la humedad del suelo (cuidado con el orden de las variables) basta con hacer:

```
recta = lm(tasa_crecimiento ~ humedad_suelo,
           data = Estudio_Crecimiento)
```

Podemos dibujar el gráfico anterior con la recta incluida.

```
plot(tasa_crecimiento ~ humedad_suelo,
     data = Estudio_Crecimiento)
recta = lm(tasa_crecimiento ~ humedad_suelo,
          data = Estudio_Crecimiento)
abline(recta)
```


tasa_crecimiento
20
30
40
50
0.5
1.0
1.5
2.0
humedad_suelo

Ahora exploramos que hay en recta:

```
recta
##
## Call:
## lm(formula = tasa_crecimiento ~ humedad_suelo, data = Estudio_Crecimiento)
##
## Coefficients:
##   (Intercept)  humedad_suelo
##         19.35          12.75
```

La recta de regresión obtenida (redondeando) es

$\mu_{Y|X} = 19.348 + 12.75 \cdot x,$

donde X es la variable predictora (humedad del suelo) e Y es la variable dependiente (tasa de crecimiento) que queremos predecir.

Con ella podemos estimar valores desconocidos que estén en el rango de los datos obtenidos (en nuestro caso valores de humedad del suelo entre 0.25 y 2).

Por ejemplo para un valor de humedad del suelo de 1 obtenemos una estimación para la tasa de crecimiento de $19.348 + 12.75 \cdot 1 = 32.098$ mm/semana.

Podemos obtener valores de predicción con la orden `predict` (cuidado con la sintaxis)

```
predict(recta, newdata = data.frame(humedad_suelo = 1))
##      1
## 32.098
```

Notar que no podemos extrapolar los datos más allá del rango de la muestra.

Podemos tener más información sobre los coeficientes con:

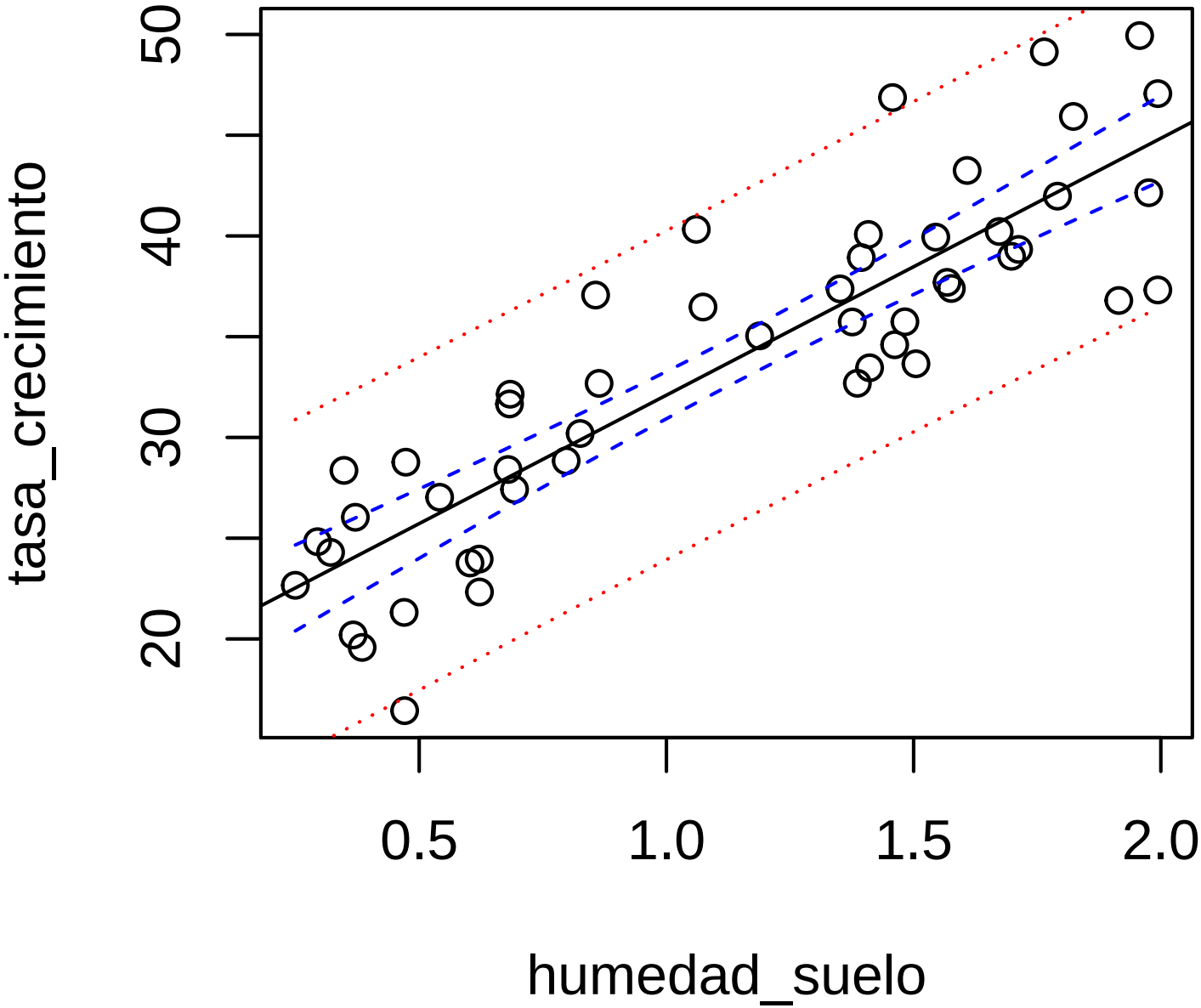
```
summary(recta)
##
## Call:
## lm(formula = tasa_crecimiento ~ humedad_suelo, data = Estudio_Cre
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9089 -3.0747  0.2261  2.6567  8.9406
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    19.348      1.283   15.08   <2e-16 ***
## humedad_suelo  12.750      1.021   12.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.019 on 48 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7599
## F-statistic: 156.1 on 1 and 48 DF,  p-value: < 2.2e-16
```

Veamos como calcular un intervalo de confianza para la media de kilos dado un peso (en azul) llamada en inglés *confidence band*, y para la predicción individual de kilos según peso (en rojo) llamada *prediction band*. Para calcularlas necesitaremos el siguiente código.

```
recta=lm(tasa_crecimiento ~ humedad_suelo,
         data = Estudio_Crecimiento)

datosaux= with(Estudio_Crecimiento,
               seq(min(humedad_suelo),max(humedad_suelo),len=100))

conf_interval <- predict(recta,
                newdata=data.frame(humedad_suelo = datosaux),
                interval="confidence",level= 0.95)


pred_interval <- predict(recta,
                newdata=data.frame(humedad_suelo = datosaux),
                interval="prediction",level = 0.95)
```

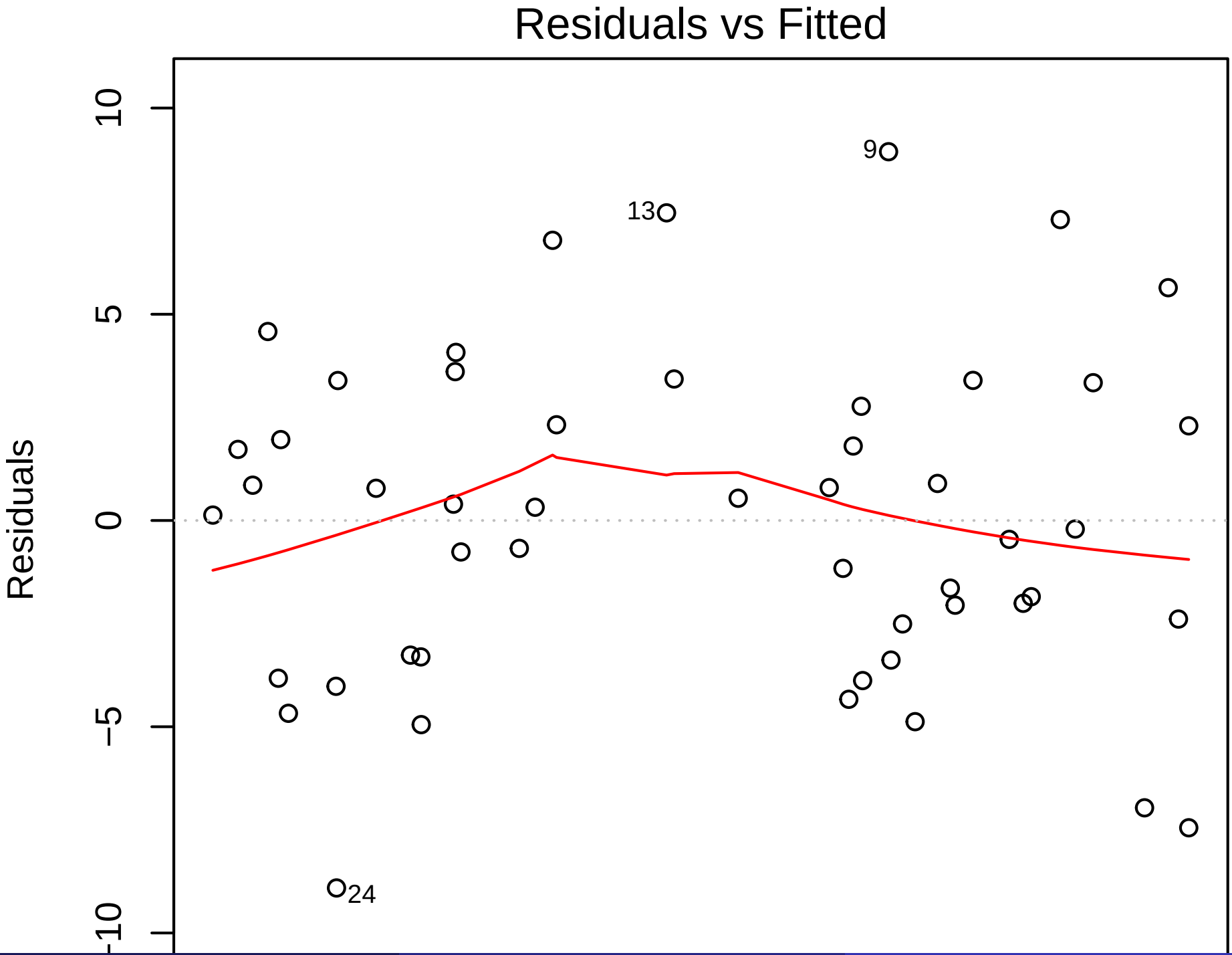
Ahora dibujamos la nube de puntos, la recta de regresión, las bandas de confianza e intervalos de predicción.

```
plot(tasa_crecimiento ~ humedad_suelo,
     data = Estudio_Crecimiento)
abline(recta)

lines(datosaux, conf_interval[,2], col="blue", lty=2)
lines(datosaux, conf_interval[,3], col="blue", lty=2)

lines(datosaux, pred_interval[,2], col="red", lty=3)
lines(datosaux, pred_interval[,3], col="red", lty=3)
```


tasa_crecimiento
20
30
40
50
0.5
1.0
1.5
2.0
humedad_suelo

Se puede (y se debe) hacer mucho más para estudiar la bondad de nuestra recta para explicar los datos de la muestra: intervalos de confianza de los coeficientes, estudiar los residuos, por ejemplo podemos obtener unos diagnósticos para la regresión con:

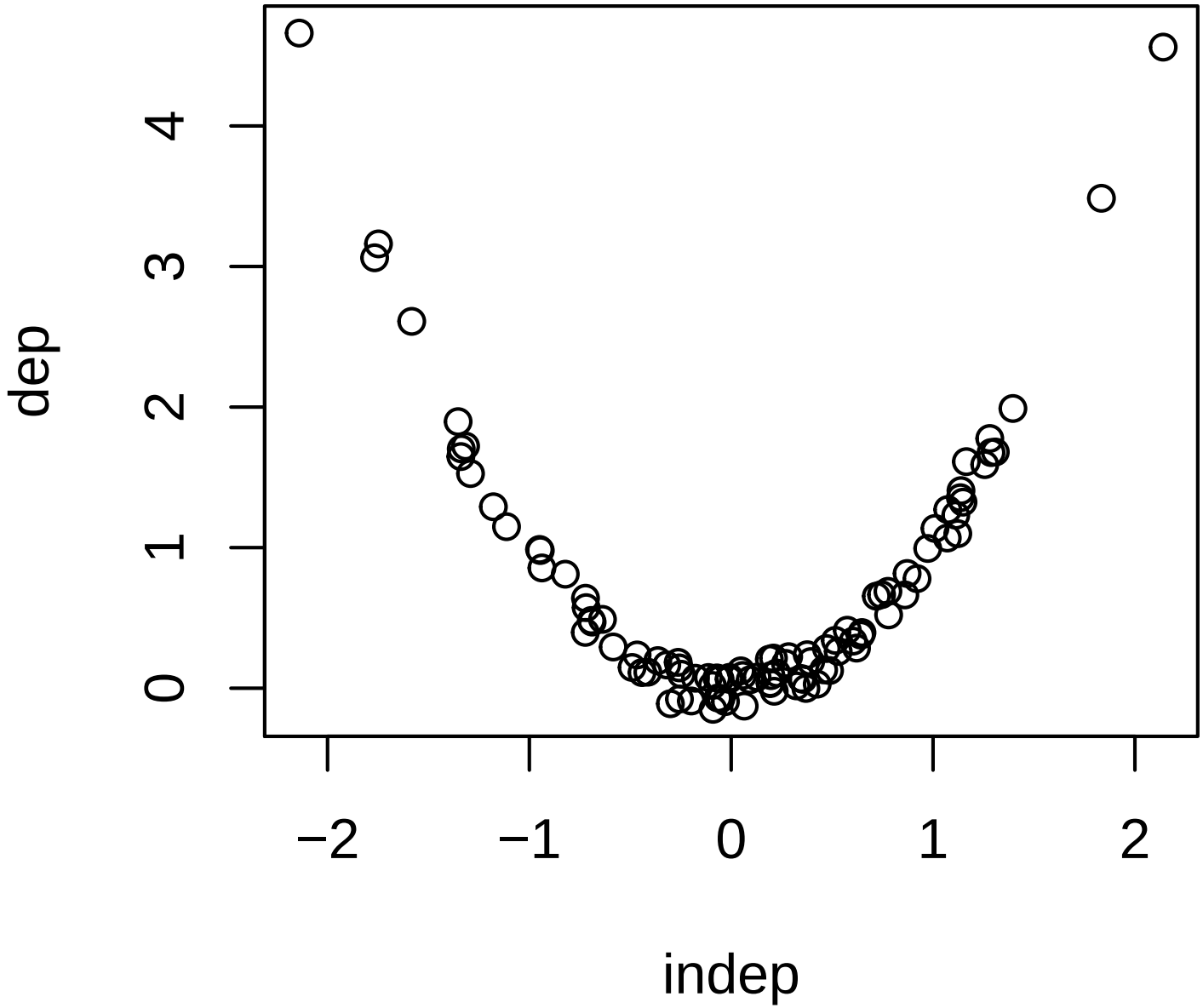
```
par(mfrow=c(2,2))
plot(recta)
```


Residuals vs Fitted
Residuals
10
5
0
−5
−10
9
13
24

Regresión Cuadrática.

En algunas ocasiones se apreciará una tendencia entre nuestros datos que no será una recta: (Descargar de ADI *datcuad.txt*) Vemos una tendencia cuadrática clara en este ejemplo.

```
datoscuadraticos=read.table("datcuad.txt")

plot(dep~indep,data=datoscuadraticos)
```


dep
0
1
2
3
4
−2
−1
0
1
2
indep

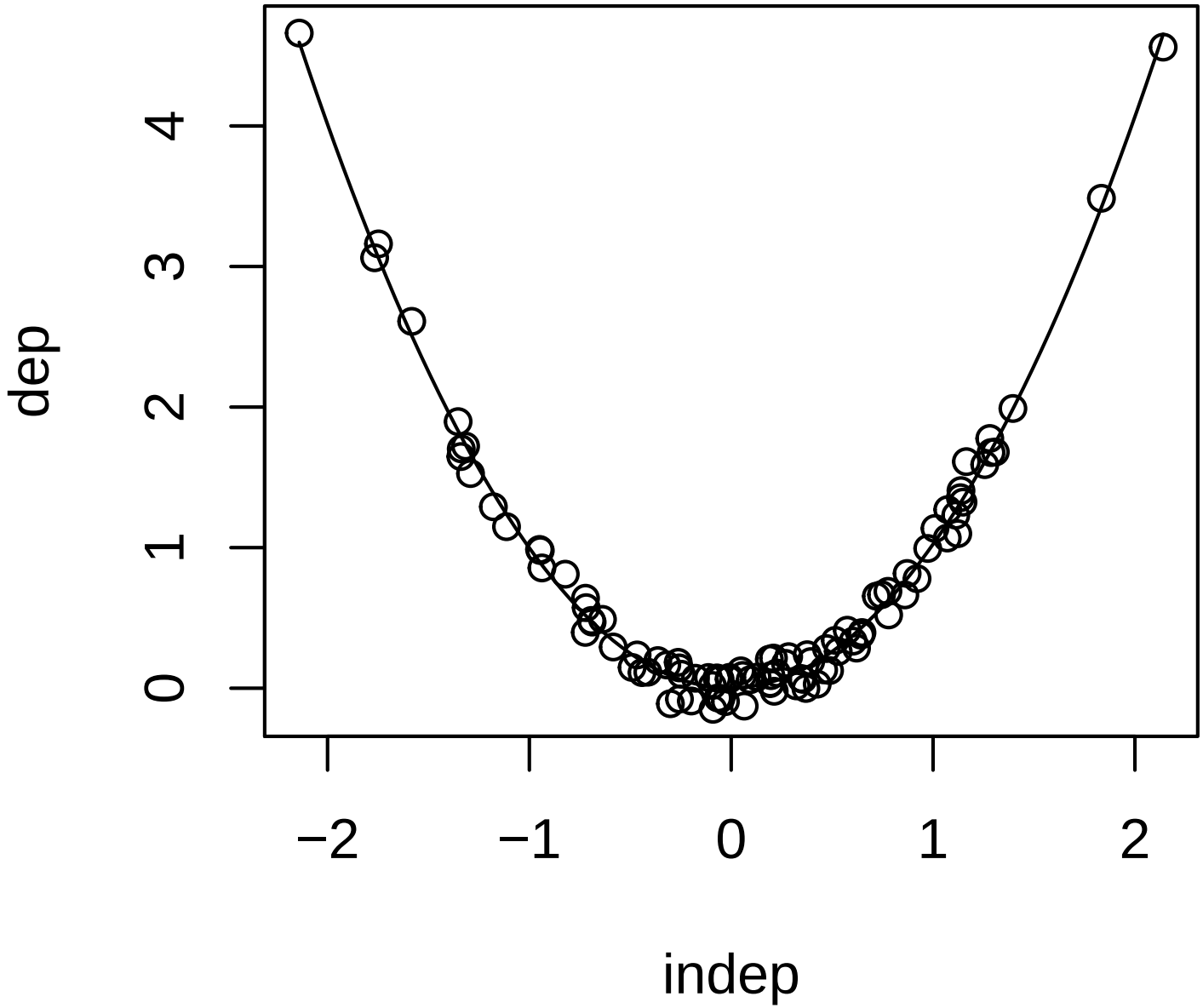
```
parabola <- lm(dep~indep + I(indep^2),
               data=datoscuadraticos)
parabola
##
## Call:
## lm(formula = dep ~ indep + I(indep^2), data = datoscuadrati
##
## Coefficients:
## (Intercept)        indep   I(indep^2)
##    0.001562     0.013416     1.009149
```

De donde obtenemos:

$$\mu_{dep|indep} = 0.002 + 0.013 \cdot indep + 1.009 \cdot indep^2$$

Podemos dibujar la nube de puntos y el ajuste con:

```r
plot(dep~indep, data=datoscuadraticos)
datosaux <-  seq(min(datoscuadraticos$indep),
                 max(datoscuadraticos$indep),
                 len=100)
lines(datosaux,predict(parabola,data.frame(indep=datosaux)))
```


dep
0
1
2
3
4
−2
−1
0
1
2
indep

Ahora podemos obtener el coeficiente de determinación R^2 con:

```
summary(parabola)$r.squared
## [1] 0.990405
```

Obtenemos $R^2 = 0.990405$. El 99% de la variabilidad de la variable dependiente viene explicada por la variabilidad en la variable independiente.

Ejercicio: Realizar una regresión cuadrática para el ejemplo anterior.

Estimación de proporciones

Veremos como estimar proporciones en datos categóricos con R.

Estimación de una proporción.

Suponemos que tiramos una moneda y obtenemos 900 caras de 1000 tiradas. Todo parece indicar que la moneda está trucada. Podemos confirmar estadísticamente esto, realizando un intervalo de confianza para la proporcion, y el correspondiente test de hipótesis (por defecto de dos colas.)

```
prop.test(900,1000, conf.level=0.95)
##
##  1-sample proportions test with continuity correction
##
## data:  900 out of 1000, null probability 0.5
## X-squared = 638.4, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.8793091 0.9175476
## sample estimates:
##   p
## 0.9
```

Recordamos que este test se basa en aproximar una binomial por una normal. R puede calcular el test binomial exacto.

```
binom.test(900,1000, conf.level=0.95)
##
##  Exact binomial test
##
## data:  900 and 1000
## number of successes = 900, number of trials = 1000, p-value
## alternative hypothesis: true probability of success is not
## 95 percent confidence interval:
##  0.8797121 0.9178947
## sample estimates:
## probability of success
##                    0.9
```

Si tenemos una variable que sea un factor con dos modalidades podemos hacer directamente el test de proporciones sobre una tabla. Veamos un ejemplo: Entre los directivos de 145 empresas biotecnológicas hay 88 hombres y 57 mujeres. ¿Puede ser esta diferencia debida al azar? Supongamos que nos dan los datos en un vector llamado sexo, con 1 para hombre y 2 para mujer.

```r
sexo <- factor(c(rep(1,88),rep(2,57)),levels=1:2, labels=c("ho
table(sexo)
## sexo
## hombre  mujer
##     88     57
prop.test(table(sexo),conf.level=0.95)
##
##  1-sample proportions test with continuity correction
##
## data:  table(sexo), null probability 0.5
## X-squared = 6.2069, df = 1, p-value = 0.01273
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.5221451 0.6858942
## sample estimates:
##         p
## 0.6068966
```

Por defecto, si tenemos dos modalidades, R calcula un intervalo de confianza para la proporción de la primera modalidad, en este caso hombre.

Comparación de dos proporciones.

La asignatura de bioestadística se divide en dos grupos, uno en inglés y otro en castellano. En el grupo de inglés 14 alumnos de un total de 38 obtienen un sobresaliente, mientras que en el grupo en castellano, 10 alumnos de un total de 40 obtienen un sobresaliente. Vamos a realizar un test de hipótesis para la igualdad de proporciones. De nuevo tenemos una variable cualitativa con dos modalidades (sacar un sobresaliente o no.)

```r
exitos <- c(14,10)
nalumnos <- c(38,40)
prop.test(exitos,nalumnos, conf.level=0.95)
##
##  2-sample test for equality of proportions with continuity
##
## data:  exitos out of nalumnos
## X-squared = 0.7872, df = 1, p-value = 0.3749
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.1110245  0.3478666
## sample estimates:
##    prop 1    prop 2
## 0.3684211 0.2500000
```

Vemos que la diferencia observada en esta muestra no es significativa.

Tablas de Contingencia.

Prueba de independencia. I

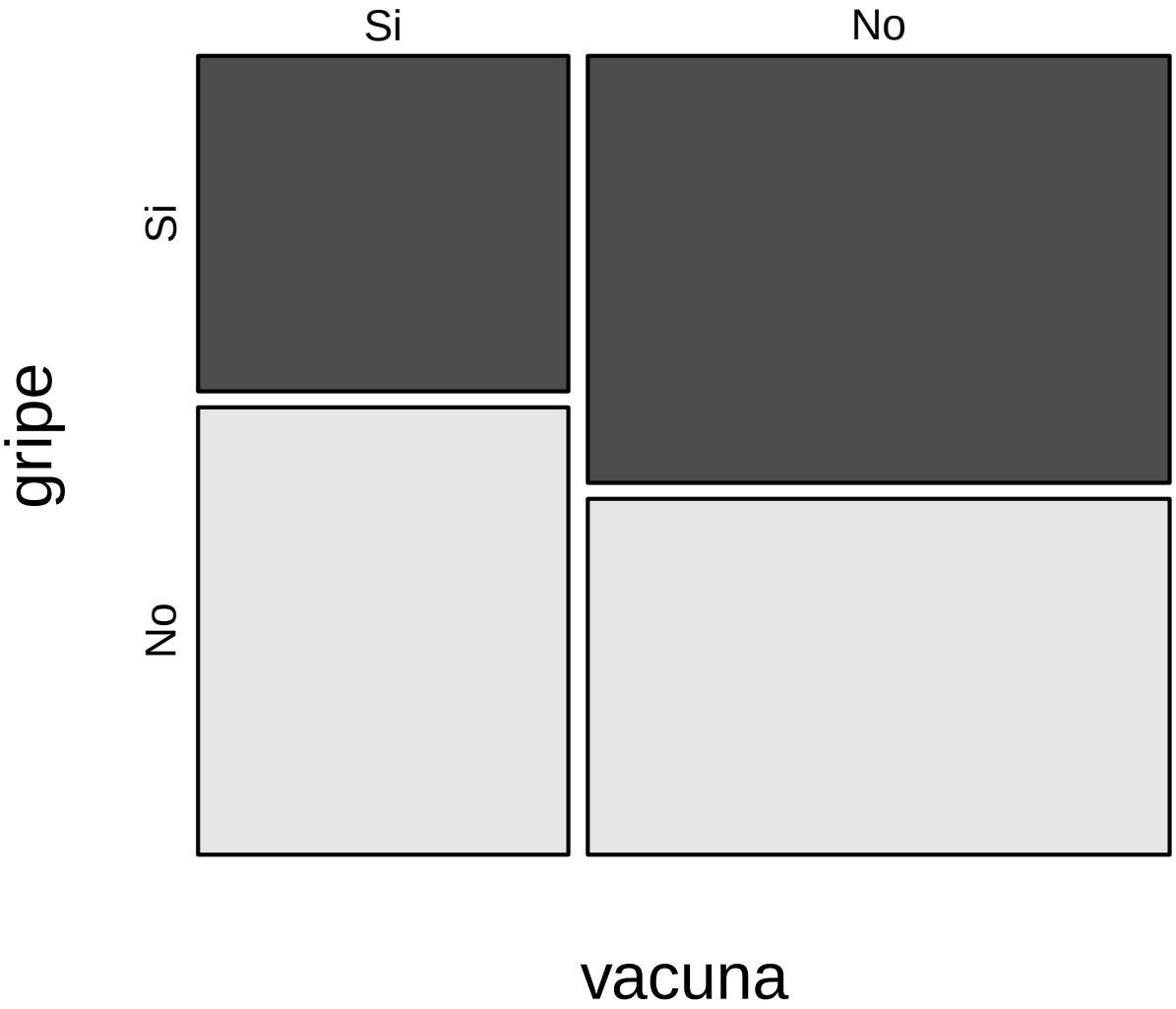
Realizamos una investigación sobre una nueva vacuna contra la gripe. Se elige una muestra aleatoria de 900 individuos y se clasifica a cada uno de ellos según haya contraído la gripe durante el último año o no, y según haya sido o no vacunado. Se recogen los datos descritos a continuación.

```
gripe <- c(rep(1,450),rep(2,450))
gripe <- factor(gripe, levels=1:2,labels=c("Si","No"))
vacuna <- (c(rep(1,150),rep(2,300),rep(1,200),rep(2,250)))
vacuna <- factor(vacuna,levels=1:2,labels=c("Si","No"))
```

Realizamos ahora la estadística descriptiva para estos datos mediante una tabla de contingencia y un diagrama de mosaico.

```
table(vacuna, gripe)
##        gripe
## vacuna  Si  No
##     Si 150 200
##     No 300 250
mosaicplot(table(vacuna,gripe), color = TRUE)
```

table(vacuna, gripe)



La orden summary ejecuta ahora un test chi cuadrado de independencia.

```
summary(table(vacuna, gripe))
## Number of cases in table: 900
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 11.688, df = 1, p-value = 0.0006289
```

También lo hace la orden `chisq.test(table(), correct=FALSE)`

```
chisq.test(table(vacuna,gripe),correct=FALSE)
##
##  Pearson's Chi-squared test
##
## data:  table(vacuna, gripe)
## X-squared = 11.688, df = 1, p-value = 0.0006289
```

La opción incluye la corrección de continuidad de Yates para tablas 2x2. es un poco más conservadora

```
chisq.test(table(vacuna,gripe),correct=TRUE)
##
##  Pearson's Chi-squared test with Yates' continuity correcti
##
## data:  table(vacuna, gripe)
## X-squared = 11.225, df = 1, p-value = 0.0008068
```

Por último con R podemos hacer el test exacto de Fischer para tablas 2x2. Esto será especialmente útil cuando alguna de las casillas tenga menos de 5 individuos.

```r
fisher.test(table(vacuna,gripe))
##
##  Fisher's Exact Test for Count Data
##
## data:  table(vacuna, gripe)
## p-value = 0.0007957
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.4725751 0.8263125
## sample estimates:
## odds ratio
##  0.6253451
```

{.allowframebreaks} Prueba de homogeneidad.

Veamos ahora un test de homogeneidad. En un estudio sobre la inclusión del huevo en la dieta se vió que la variable sexo jugaba un papel importante. El investigador pensó que el tipo de cocción preferido podría ser distinto en hombres y mujeres. Tomo una muestra de 25 mujeres a las que se preguntó como preferían ingerir el huevo: a) frito, b) tortilla, c) cocido, y se hizo lo mismo con una muestra de 17 hombres, obteniendo los siguientes resultados.

```
mitabla <- matrix(c(5,9,12,3,7,5),ncol=3)
colnames(mitabla)=c("frito","tortilla","cocido")
rownames(mitabla)=c("Mujer","Hombre")
mitabla
##         frito tortilla cocido
## Mujer       5       12      7
## Hombre      9        3      5
mosaicplot(mitabla, color=TRUE)
```

mitabla

Realizamos ahora el correspondiente test de homogeneidad

```
chisq.test(mitabla)
## Warning in chisq.test(mitabla): Chi-squared approximation m
##
##  Pearson's Chi-squared test
##
## data:  mitabla
## X-squared = 5.8516, df = 2, p-value = 0.05362
```

Nos damos cuenta que aparece una advertencia de que el test puede no ser adecuado. Esto es debido a que tenemos tres casos con 5 individuos o menos de un total de 6 (más del 20%). R nos permite utilizar un método de montecarlo para encontrar un p-value.

```
chisq.test(mitabla,simulate.p.value = TRUE, B = 10000)
##
##  Pearson's Chi-squared test with simulated p-value (based o
##  replicates)
##
## data:  mitabla
## X-squared = 5.8516, df = NA, p-value = 0.05809
```

Análisis de la varianza

Anova de una vía.

Vamos a usar unos datos presentes en R. Se trata de un estudio sobre 6 insecticidas. Se dividió un área geográfica homogénea en 72 zonas del mismo tamaño. Se seleccionaron 12 zonas al azar para el insecticida A, de las restantes 12 para el insecticida B, de las restantes 12 para el insecticida C, etc hasta tener una asignación aleatoria de 12 zonas para cada insecticida. Los datos están guardados en InsectSrays.

```r
data(InsectSprays)
str(InsectSprays)
## 'data.frame':    72 obs. of  2 variables:
##  $ count: num  10 7 20 14 14 12 10 23 17 20 ...
##  $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1
head(InsectSprays)
##   count spray
## 1    10     A
## 2     7     A
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A
```

Vamos a realizar algunos estadísticos descriptivos por insecticida. Vamos a ver dos maneras de trabajar muy útiles en R.
`with(dataframe,funcion(variables))` permite útilizar variables de un data frame sin necesidad de poner `dataframe$variable`, o usar `attach(dataframe)`.

Por otra parte utilizaremos `tapply(variable, grupo, funcion)` que aplica una funcion a una variable según los valores de grupo, una variable cualitativa.

Por ejemplo vamos a calcular la media, desviación estardar, y tamaño muestral de la variable `count` según el tipo de insecticida codificado en la variable `spray`:

```r
with(InsectSprays,tapply(count,spray,mean))
##         A         B         C         D         E         F
## 14.500000 15.333333  2.083333  4.916667  3.500000 16.666667
with(InsectSprays,tapply(count,spray,sd))
##         A         B         C         D         E         F
## 4.719399 4.271115 1.975225 2.503028 1.732051 6.213378
with(InsectSprays,tapply(count,spray,length))
##  A  B  C  D  E  F
## 12 12 12 12 12 12
```

Hay otra función en R aggregate que hace el mismo cálculo.

```
aggregate(count~spray, data=InsectSprays, FUN= mean)
##   spray      count
## 1     A 14.500000
## 2     B 15.333333
## 3     C  2.083333
## 4     D  4.916667
## 5     E  3.500000
## 6     F 16.666667
```

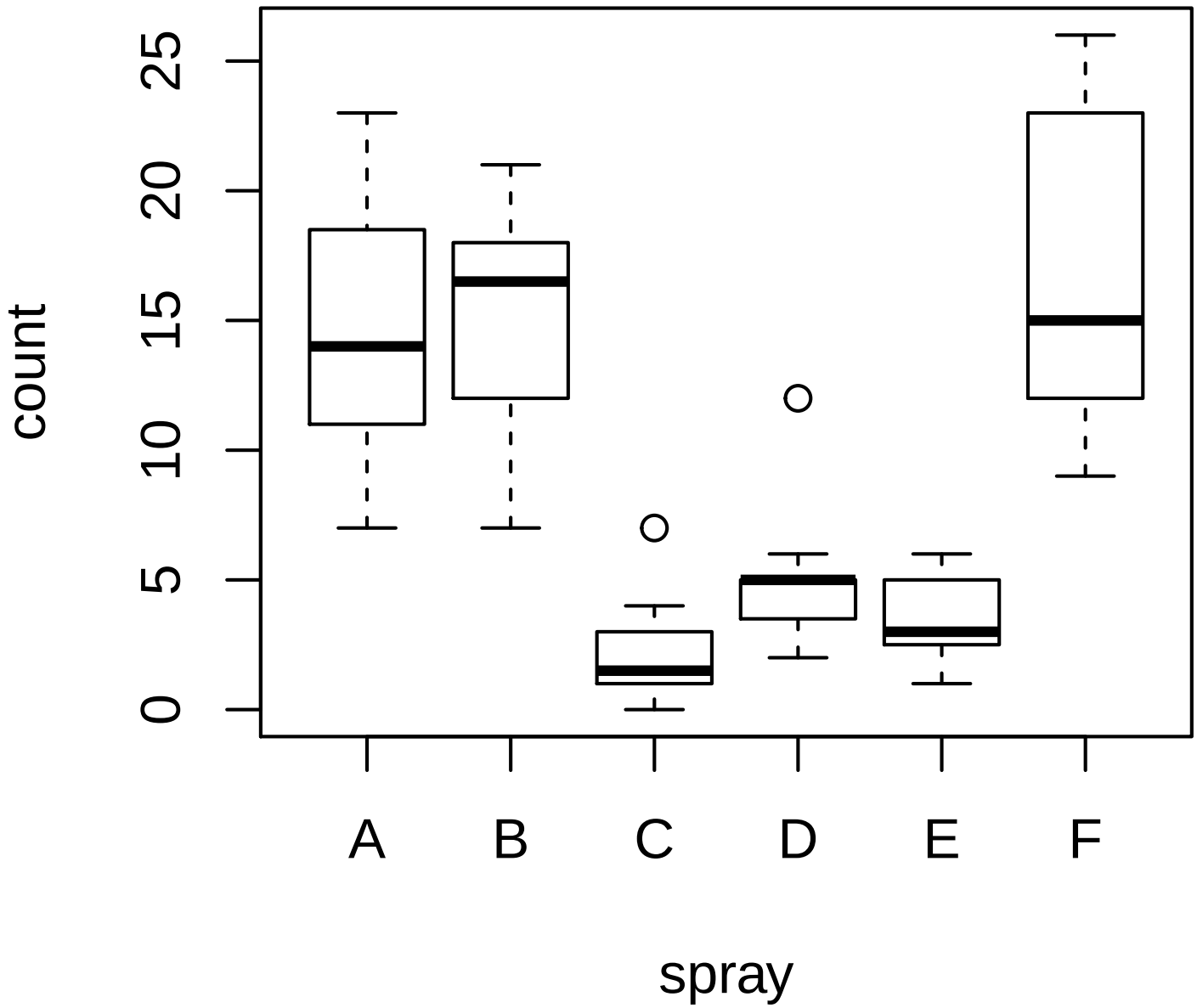
Podemos conseguir una tabla con todos los objetos calculados del siguiente modo:

```
do.call(data.frame,
        aggregate(count~spray,
                  data=InsectSprays,
                  FUN = function(x) c(m = mean(x),
                                      s = sd(x),
                                      n=length(x))))
##   spray    count.m  count.s count.n
## 1     A 14.500000 4.719399      12
## 2     B 15.333333 4.271115      12
## 3     C  2.083333 1.975225      12
## 4     D  4.916667 2.503028      12
## 5     E  3.500000 1.732051      12
## 6     F 16.666667 6.213378      12
```

Pero la orden se hace más y más ininteligible. Veremos como resolver este problema con dplyr.

Ahora dibujamos los correspondientes diagramas de caja. Vemos como boxplot nos permite una alternativa a `with(...)` usando `boxplot(...,data=InsectSprays)`. Hay muchas funciones en R que permiten especificar el dataframe como opción con `data=nombredataframe`. El resultado es el mismo que el que obtendríamos usando `with(InsectSprays,boxplot(count~spray))`

```
boxplot(count ~ spray, data=InsectSprays)
```


count
0
5
10
15
20
25
A
B
C
D
E
F
spray

En este gráfico se aprecia que no se cumplen los requisitos para una ANOVA. Algunas distribuciones no son simétricas, hay bastantes outlayers, y además las varianzas parecen bastante distintas.

Aún así realizaremos una ANOVA para ver cual sería el procedimiento. Hay varias maneras de realizar una ANOVA con R.

```
salida.aov = aov(count ~ spray, data=InsectSprays)
summary(salida.aov)
##              Df Sum Sq Mean Sq F value Pr(>F)
## spray         5   2669   533.8    34.7 <2e-16 ***
## Residuals    66   1015    15.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

Otra manera es utilizar la orden `oneway.test()` que hace la corrección de Welch para varianzas distintas (pero requiere normalidad)

```
oneway.test(count~spray, data=InsectSprays)
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  count and spray
## F = 36.065, num df = 5.000, denom df = 30.043, p-value = 7.
```

El resultado difiere del anterior por la corrección de Welch. Si ponemos la opción `var.equal=TRUE` obtendremos los mismos resultados que con `aov`

```
oneway.test(count~spray, data=InsectSprays, var.equal=TRUE)
##
##  One-way analysis of means
##
## data:  count and spray
## F = 34.702, num df = 5, denom df = 66, p-value < 2.2e-16
```

Una tercera manera es utilizar un modelo lineal con `lm` y después utilizar la orden `anova()` sobre la salida del modelo. En realidad esto es lo que hace la orden aov internamente. En una anova de una vía con una variable *tratamiento* y una variable *respuesta* haremos `lm(respuesta~tratamiento)`. Estamos haciendo una regresión donde la variable respuesta es cuantitativa y la variable tratamiento es cuantitativa.

```
modelo=lm(count~spray, data=InsectSprays)
anova(modelo)
## Analysis of Variance Table
##
## Response: count
##            Df Sum Sq Mean Sq F value    Pr(>F)
## spray       5 2668.8  533.77  34.702 < 2.2e-16 ***
## Residuals 66 1015.2   15.38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

Si queremos ahora ver que grupos son significativamente distintos, podemos hacer un test Post Hoc. Se pueden hacer estos tests con el comando `pairwise.t.test(variable,grupo)`. Este método nos permite hacer distintas correcciones con `p.adjust=` Por ejemplo para realizar la corrección de Bonferroni realizaríamos:

```
with(InsectSprays,
     pairwise.t.test(count,spray,p.adj="bonferroni"))
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  count and spray
##
##   A        B        C        D        E
## B 1        -        -        -        -
## C 1.1e-09  1.3e-10  -        -        -
## D 1.5e-06  1.8e-07  1        -        -
## E 4.1e-08  4.9e-09  1        1        -
## F 1        1        4.2e-12  6.1e-09  1.6e-10
##
## P value adjustment method: bonferroni
```

En R el test HSD de Tuckey (Honest Significant Difference) que nos da intervalos de confianza corregidos par las diferencias de medias para cada par de grupos.

```
TukeyHSD(salida.aov)
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = count ~ spray, data = InsectSprays)
##
## $spray
##             diff        lwr       upr     p adj
## B-A    0.8333333  -3.866075  5.532742 0.9951810
## C-A -12.4166667 -17.116075 -7.717258 0.0000000
## D-A  -9.5833333 -14.282742 -4.883925 0.0000014
## E-A -11.0000000 -15.699409 -6.300591 0.0000000
## F-A    2.1666667  -2.532742  6.866075 0.7542147
## C-B -13.2500000 -17.949409 -8.550591 0.0000000
## D-B -10.4166667 -15.116075 -5.717258 0.0000002
## E-B -11.8333333 -16.532742 -7.133925 0.0000000
## F-B    1.3333333  -3.366075  6.032742 0.9603075
## D-C    2.8333333  -1.866075  7.532742 0.4920707
## E-C    1.4166667  -3.282742  6.116075 0.9488669
## F-C   14.5833333   9.883925 19.282742 0.0000000
```

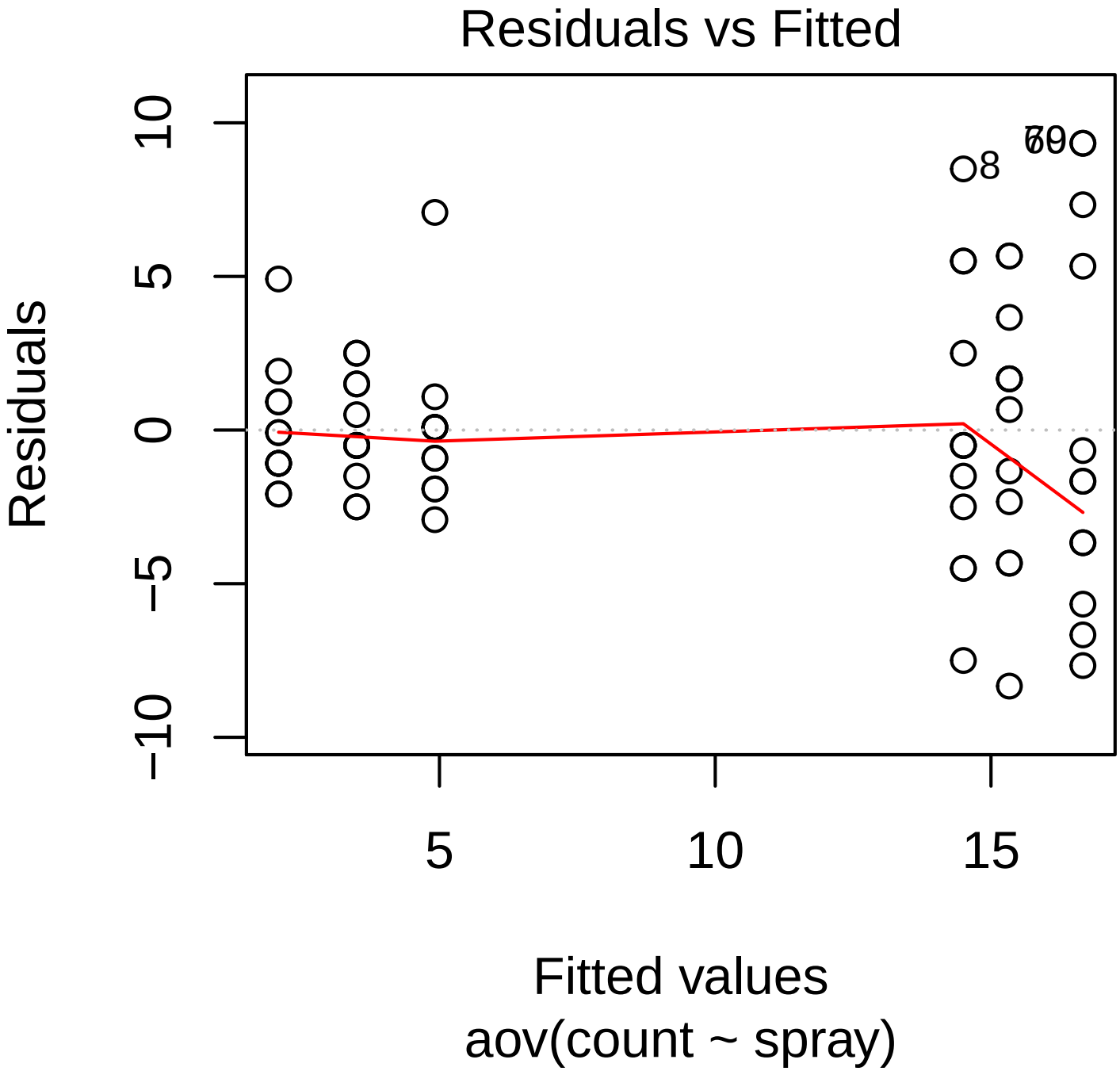
Una manera formal de probar si las varianzas son iguales es el test de Bartlett.

```
bartlett.test(count ~ spray, data=InsectSprays)
##
##  Bartlett test of homogeneity of variances
##
## data:  count by spray
## Bartlett's K-squared = 25.96, df = 5, p-value = 9.085e-05
```

No podemos asegurar que las varianzas son iguales.

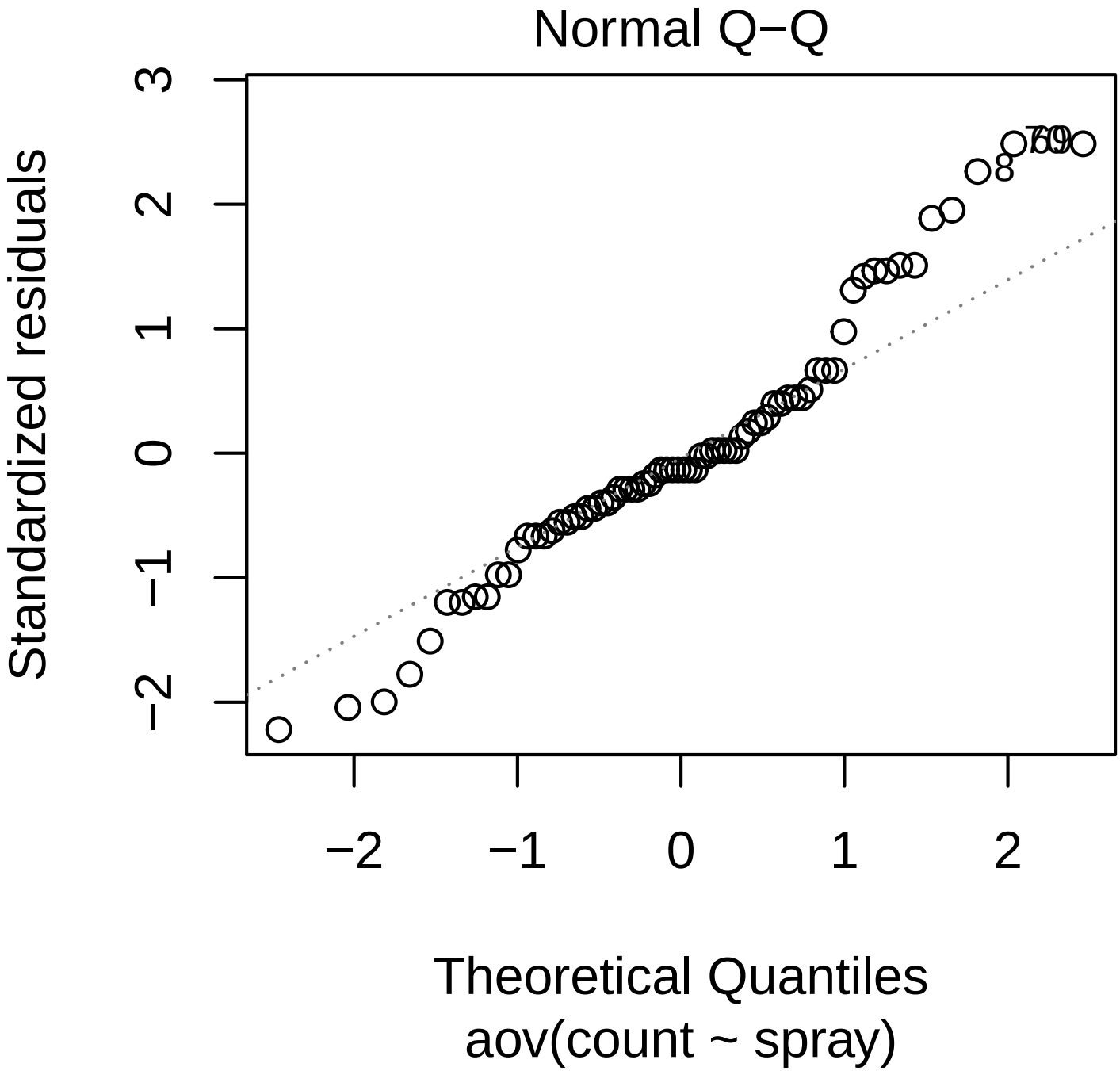
Podemos obtener una serie de diagnósticos de un test de anova con `plot(salida.aov)`

```
plot(salida.aov, 1)
```


Residuals vs Fitted
Residuals
10
5
0
−5
−10
5
10
15
Fitted values
aov(count ~ spray)

Nos da un gráfico de la distribución de los residuos. Idealmente ha de mostrar la misma dispersión en cada valor. Aquí se aprecia un preocupante aumento en la dispersión de los residuos para valores grandes.

```
plot(salida.aov, 2)
```


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−2
−1
0
1
2
Theoretical Quantiles
aov(count ~ spray)

Nos da un gráfico cuantil-cuantil que compara la distribución de los residuos con una normal. Vemos que en las colas se aleja de una normal.

La alternativa no paramétrica es el test de Kruskall-Wallis. Este test necesita que todas las distribuciones sean similares y que haya homogeneidad de varianzas.

```
kruskal.test(count ~ spray, data=InsectSprays)
##
##  Kruskal-Wallis rank sum test
##
## data:  count by spray
## Kruskal-Wallis chi-squared = 54.691, df = 5, p-value = 1.51
```

Cuando nuestros datos no cumplen las condiciones para un test (normalidad, igualdad de varianzas), podemos aplicar transformaciones a nuestros datos $Y' = f(Y)$ para que si las cumplan y realizar los tests sobre los datos transformados. Habrá que tener especial cuidado a la hora de interpretar los resultados pues nos interesan resultados sobre los datos sin transformar.

Las transformaciones más habituales son:

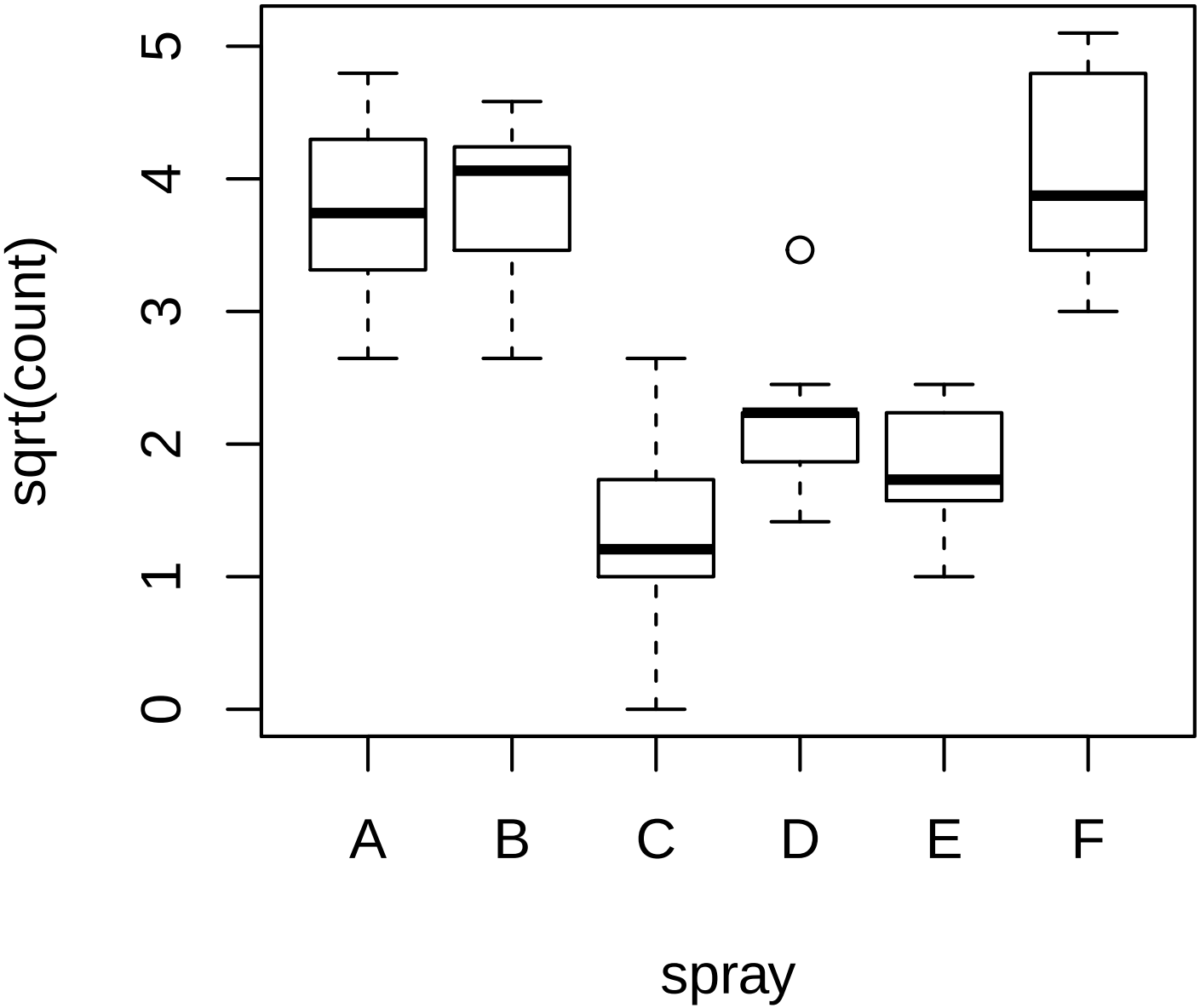
- $Y' = \log(Y)$ Transformación válida para valores positivos o $Y' = \log(Y + 1)$ si los datos incluyen al 0. La media aritmética queda transformada en media geométrica. Algunos casos en los que esta transformación es útil comprenden cuando las medidas son cocientes o productos de variables o cuando la distribución de los datos tiene una cola a la derecha o cuando el grupo que tiene la media mayor también tiene la desviación estandar mayor o cuando los datos comprenden varios ordenes de magnitud.
- $Y' = \mathrm{logit}(Y)$ Usada fundamentalmente para proporciones.

- $Y' = \sqrt{Y + a}$ donde a puede tomar los valores 0, 1/2 o 1 según los datos. Usada cuando los datos son conteos.

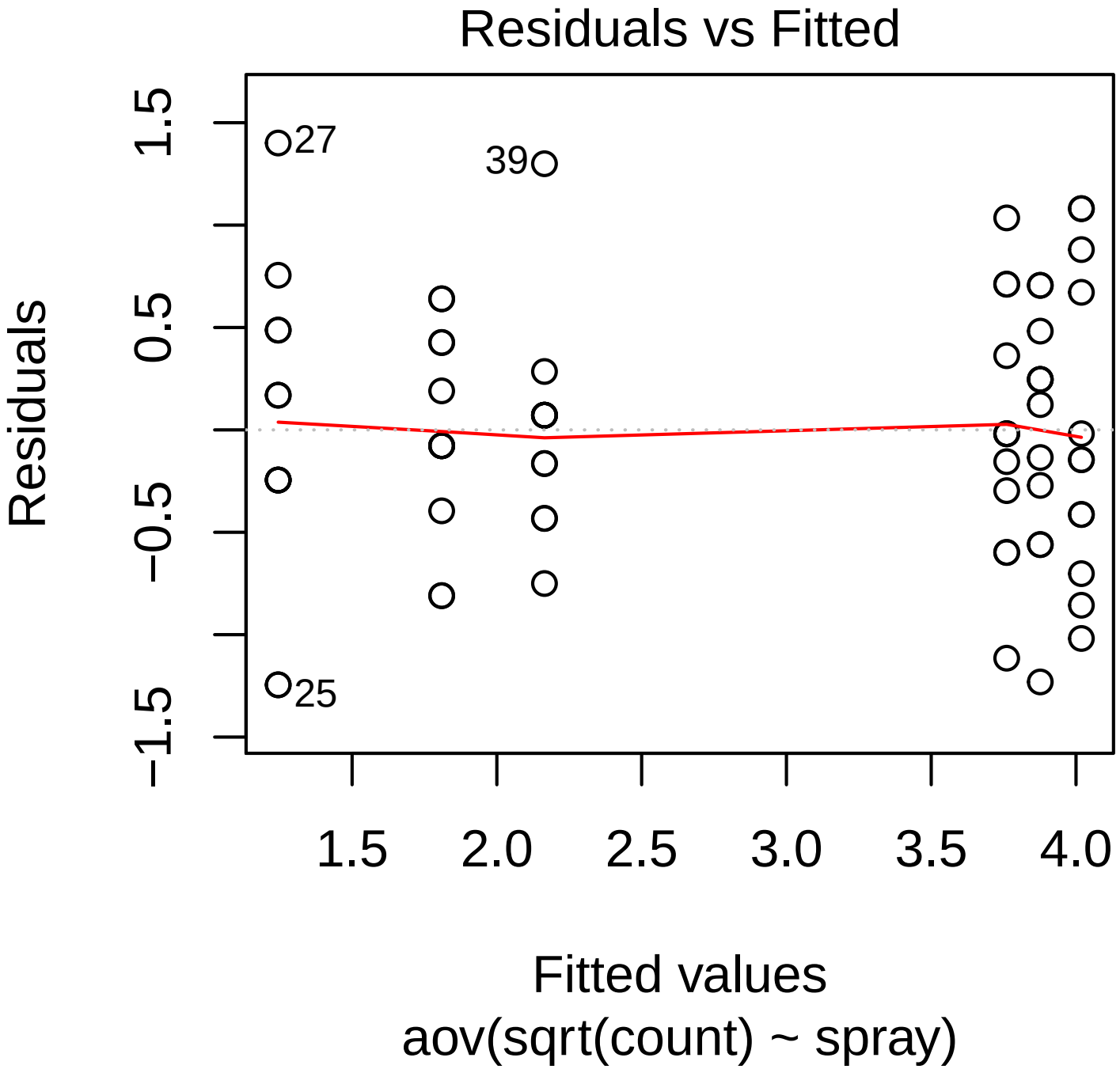
Hay muchas otras transformaciones posibles, pero hay que tener cuidado a la hora de probar distintas transformaciones porque podríamos incurrir en los mismos problemas que al hacer comparaciones múltiples (Incremento del error de tipo I)

En los datos de InsectSpray si hacemos:

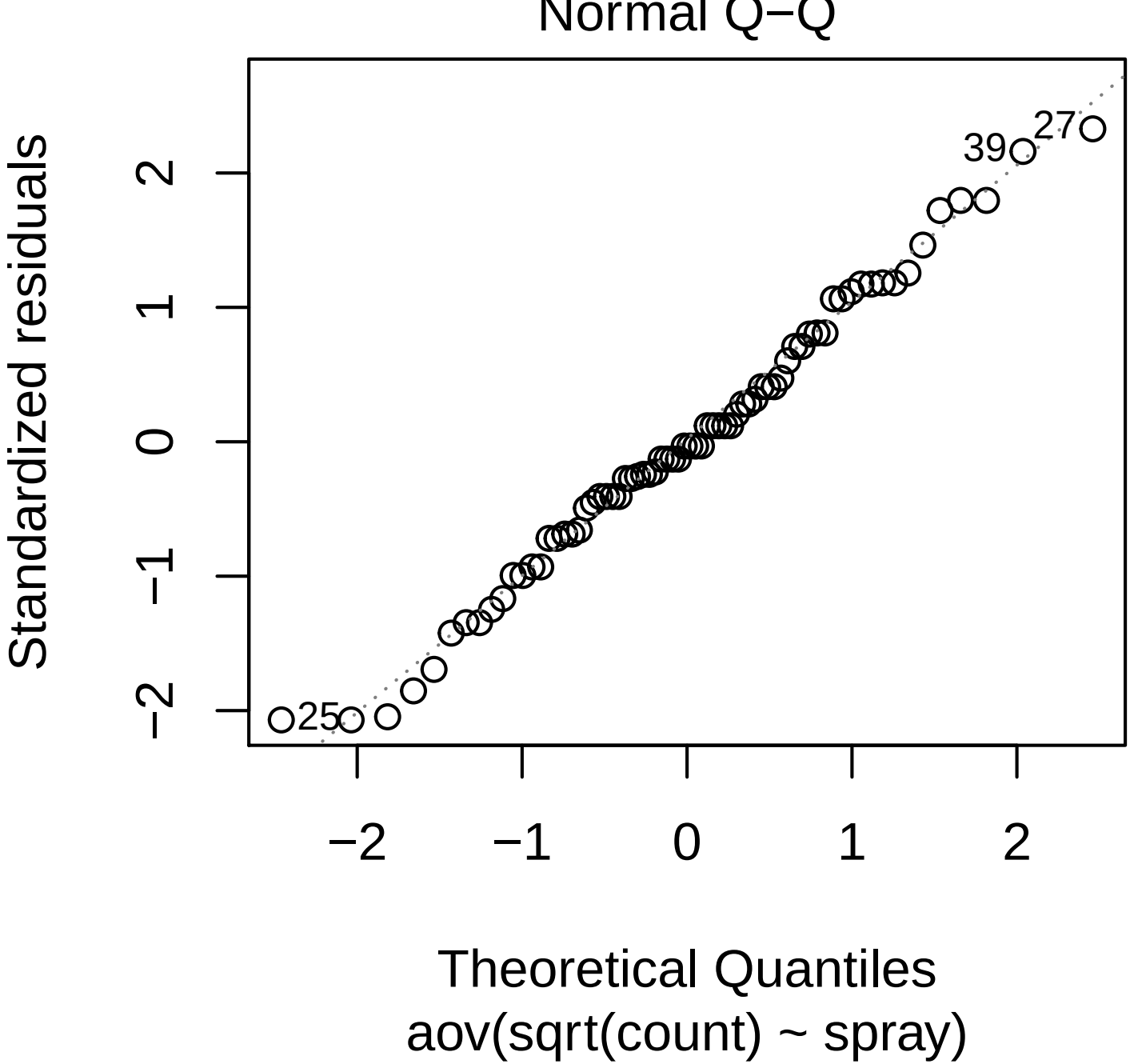
```
bartlett.test(sqrt(count) ~ spray, data=InsectSprays)
##
##  Bartlett test of homogeneity of variances
##
## data:  sqrt(count) by spray
## Bartlett's K-squared = 3.7525, df = 5, p-value = 0.5856
with(InsectSprays,boxplot(sqrt(count) ~ spray))
```


sqrt(count)
0
1
2
3
4
5
A
B
C
D
E
F
spray

```r
salida.aov2 <- aov(sqrt(count) ~ spray, data = InsectSprays)
summary(salida.aov2)
##              Df Sum Sq Mean Sq F value Pr(>F)
## spray         5  88.44  17.688    44.8 <2e-16 ***
## Residuals    66  26.06   0.395
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
plot(salida.aov2,1)
```


Residuals vs Fitted
Residuals
1.5
0.5
−0.5
−1.5
27
39
25
1.5
2.0
2.5
3.0
3.5
4.0
Fitted values
aov(sqrt(count) ~ spray)

```r
plot(salida.aov2,2)
```


Normal Q−Q
Standardized residuals
−2
−1
0
1
2
25
39
27
−2
−1
0
1
2
Theoretical Quantiles
aov(sqrt(count) ~ spray)

Anova de dos vías.

En el fichero *quinn.csv* encontramos datos sobre los efectos que tienen la estación del año y la densidad de ejemplares adultos en la producción de huevos de la especie *Sinphonaria Diemenensis*

```
quinn=read.csv("quinn.csv")
str(quinn)
## 'data.frame':    24 obs. of  3 variables:
##  $ DENSITY: int  8 8 8 8 8 8 15 15 15 15 ...
##  $ SEASON : Factor w/ 2 levels "spring","summer": 1 1 1 2 2
##  $ EGGS   : num  2.88 2.62 1.75 2.12 1.5 ...
```

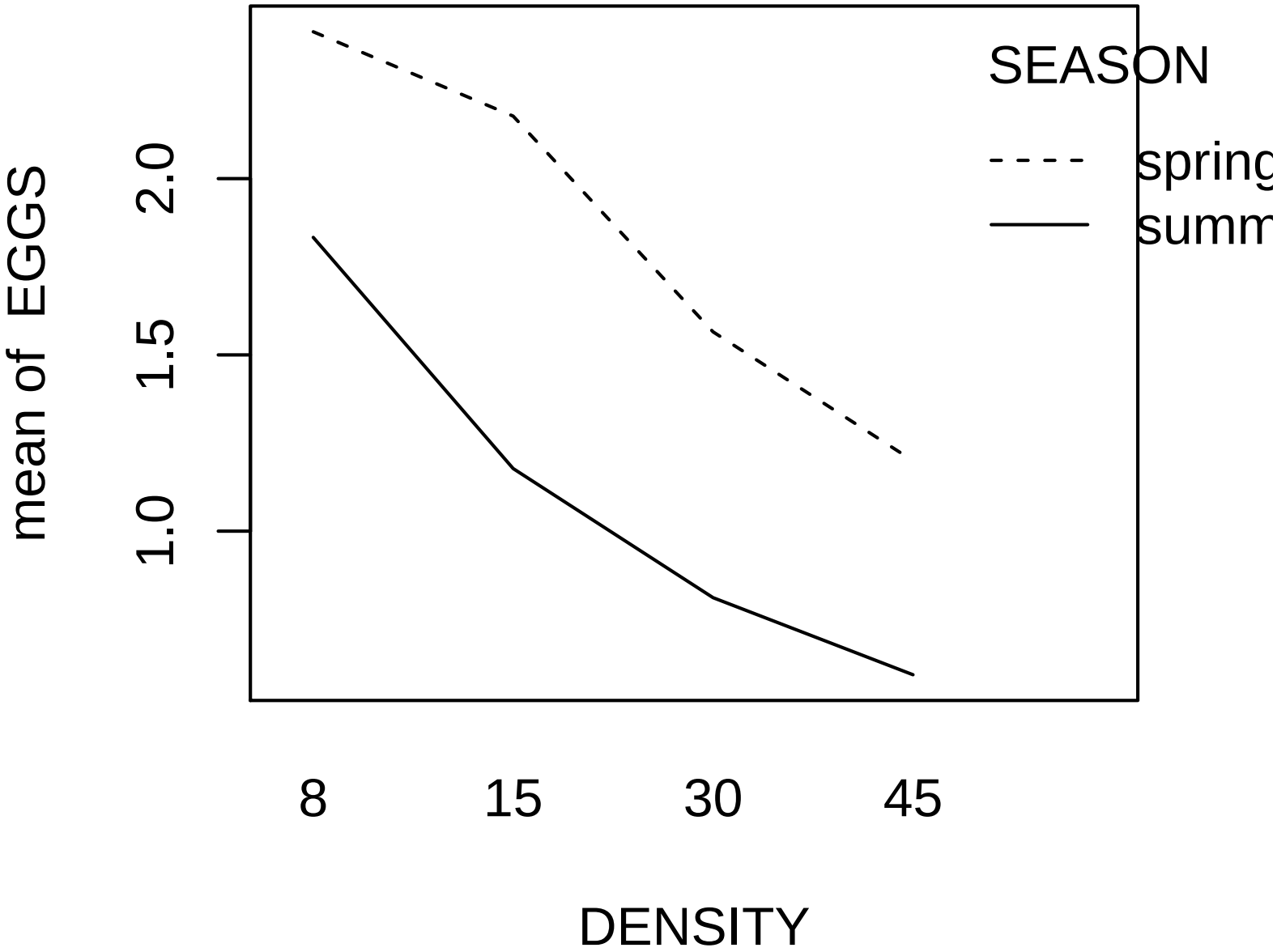
Este conjunto de datos es típico de un ANOVA con dos factores (y efectos fijos). Tenemos una variable (respuesta) cuantitativa, que es la producción de huevos (la tercera columna de los datos). Y queremos estudiar la relación de esa variable con dos variables (explicativas), que son la la densidad de adultos y la estación (primera y segunda columnas, respectivamente).

Se trata de dos variables cualitativas o factores. Esto es especialmente evidente en el caso de la variable SEASON, que tiene dos niveles (spring y summer). Vemos que la variable DENSITY no es un factor, así que la transformamos en uno.

```
quinn.aov = aov(EGGS ~ SEASON + DENSITY, data = quinn)
anova(quinn.aov)
## Analysis of Variance Table
##
## Response: EGGS
##           Df Sum Sq Mean Sq F value    Pr(>F)
## SEASON     1 3.2502  3.2502  20.439  0.000187 ***
## DENSITY    1 5.0241  5.0241  31.595 1.406e-05 ***
## Residuals 21 3.3394  0.1590
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

Podemos hacer un estudio factorial (viendo las posibles interacciones)

```
quinn.aov = aov(EGGS ~ SEASON * DENSITY, data = quinn)
anova(quinn.aov)
## Analysis of Variance Table
##
## Response: EGGS
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## SEASON           1 3.2502  3.2502 19.5350 0.0002637 ***
## DENSITY          1 5.0241  5.0241 30.1971 2.226e-05 ***
## SEASON:DENSITY   1 0.0118  0.0118  0.0711 0.7925333
## Residuals       20 3.3275  0.1664
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
with(quinn,interaction.plot(DENSITY, SEASON, EGGS))
```


SEASON
spring
summ
mean of EGGS
1.0
1.5
2.0
8
15
30
45
DENSITY

El factor DENSITY se muestra en el eje horizontal, la respuesta EGGS en el eje vertical, y cada uno de los dos factores de SEASON se muestra como una línea que conecta los correspondientes valores.

En un caso ideal, la ausencia completa de interacción correspondería a dos líneas perfectamente paralelas en este gráfico (y la existencia de interacción es evidente cuando las líneas se cruzan).

En la práctica, el paralelismo en general está lejos de ser perfecto. No obstante, la gráfica de este ejemplo sí parece indicar que no existe interacción entre ambos factores. Y en cualquier caso, tenemos los resultados de la tabla ANOVA para corroborarlo.