Introducción al entorno R y Rstudio

Análisis de datos y modelos lineales en R

Enrique Andivia[1]

Master Universitario en Restauración de Ecosistemas
Curso 2019-2020

1 Departamento de Biodiversidad, Ecología y Evolución. Universidad Complutense de Madrid

email: e.andivia@gmail.com

web: http://enriqueandivia.wixsite.com/forests

INTRODUCCIÓN AL ENTORNO R Y RSTUDIO

1- R, algo más que un lenguaje de programación

En esencia R es un lenguaje de programación y un entorno para la manipulación de datos y la realización de cálculos, análisis estadísticos y gráficos. R nació en 1997 en el Departamento de Estadística de la Universidad de Auckland (Nueva Zelanda). Desde entonces se han desarrollo multitud de aplicaciones en R y su uso se ha generalizado en el análisis estadístico de datos en distintas áreas de investigación como la ecología. Además R supone un esfuerzo colaborativo y cuenta con una gran comunidad online no solo de desarrolladores sino también de usuarios experimentados dispuestos a resolver cualquier duda a otros usuarios.

Como hemos comentado, R es un lenguaje de programación (basado en el lenguaje S) y como tal permite la implementación de cualquier tipo de funciones que el usuario pueda imaginar. Por ejemplo, en R hay funciones para realizar un ANOVA a un set de datos pero también existe una función que permite pedir una pizza desde tu ordenador si vives en Australia.

El entorno de R ofrece una serie de ventajas sobre los programas de análisis estadísticos convencionales:

- Es un software libre y gratuito. Que sea un software libre significa que el código del mismo es accesible y modificable por cualquier usuario. Dicho código está bajo las condiciones de la licencia GNU-GPL. Aunque dicho código pueda ser modificado por cualquier usuario de R la mayor parte de las funciones implementadas han sido meticulosamente supervisadas por el “R Core team”, un grupo de estadísticos de gran prestigio internacional.
- Como hemos comentado anteriormente, R es una herramienta muy flexible con multitud de librerías (llamadas paquetes) que incluyen una serie de funciones programadas por otros usuarios y que se utilizan para llevar a cabo un procedimiento o un análisis concreto.
- Existe una gran comunidad online de usuarios donde es difícil no encontrar una respuesta a cualquier duda o problema que pudiéramos tener. Además muchos de los desarrolladores de las funciones de R se encuentran activos en esta comunidad por lo que la mayoría de los problemas son resueltos. Además existe mucha documentación online y gratuita sobre cómo usar R o cualquier paquete o función específica.
- Al ser un lenguaje de programación nos permite crear nuestras propias funciones y procedimientos, de modo que podemos adaptar todo a nuestras necesidades concretas.
- Favorece la reproducibilidad de la ciencia mediante la elaboración de scripts donde se recogen todos los análisis estadísticos aplicados a un set de datos y que puede ser utilizado por cualquier otra persona o modificado según la naturaleza concreta de los datos o estudios.

2- Instalación de R y Rstudio

R está disponible para las plataformas Windows, Linux y Mac Os. Para su descarga hay que ir a la página principal del proyecto (CRAN):

http://cran.r-project.org

Aquí encontraremos desde manuales gratuitos hasta el propio código fuente de R. Para la instalación de R seleccionaremos la descarga para el sistema operativo de nuestro portátil u ordenador y posteriormente pincharemos en el enlace "*install R for the first time*" y en *"Download R…"*. Una vez descargado el archivo de instalación simplemente hay que abrirlo y seguir las instrucciones del asistente de instalación.

Junto a R vamos a instalar también Rstudio. Rstudio es un entorno de desarrollo integrado para R, de código abierto y libre que permite escribir y correr scripts de R. Además confiere un entorno de Windows para R lo que facilita la realización de algunas tareas para usuarios principiantes. Rstudio se descarga en la siguiente dirección:

https://rstudio.com/products/rstudio/

Seleccionaremos Rstudio Desktop y a continuación la versión Open Source Edition en su versión gratuita. Al igual que para R, seleccionaremos el instalador para nuestro sistema operativo.

Antes de comenzar es importante que comprobemos qué separador estamos utilizando para los decimales en nuestros ordenadores. Para evitar problemas debemos utilizar el separador decimal y de millares por defecto de la mayoría de ordenadores del mundo, es decir "." para los decimales y "," para los millares. En los ordenadores españoles estos separadores vienen por defecto al contrario. Para comprobar esto debemos abrir la aplicación "Hora e idioma" de Windows, seleccionar "región" en el menú de la izquierda, luego opciones adicionales en el menú de la derecha y "cambias formatos de fecha hora y número" en la ventana que aparece. Finalmente pincharemos en "configuración adicional" aquí se podrá cambiar el separador utilizado. En caso de que fuera necesario cambiarlo no olvidéis pinchar en "aplicar".

3- Paquetes en R

Como hemos comentado anteriormente existe multitud de funciones en R para realizar diversos procedimientos y estas funciones suelen estar agrupadas en paquetes "*packages*". La primera vez que instalamos R, no solo instalamos el entorno de trabajo sino también una serie de paquetes básicos que van desde la manipulación de datos hasta el ajuste de modelos pasando por la representación gráfica. Tanto estos paquetes básicos como el código fuente solo pueden ser modificados por el denominado "R core team" aunque cualquier usuario puede sugerir cambios en este código.

Gran parte del éxito de R reside en la cantidad de paquetes que se han ido desarrollando y que atiende a las necesidades de la comunidad científica en cuanto a la implementación de las distintas técnicas estadísticas en el análisis de datos. Así podremos encontrar paquetes para la realización de sencillos test de contraste de hipótesis hasta paquetes con las más modernas técnicas de análisis de estadística bayesiana. Una vez que instalamos un paquete en R, todas las funciones del mismo quedan a nuestra disponibilidad así como un documento de ayuda donde se explica cómo usar todas y cada una de las funciones contenidas en el paquete. El autor o autores de los paquetes son los últimos responsables de su mantenimiento aunque aquellos paquetes más populares son monitorizados constantemente por un grupo de expertos.

Todos estos paquetes se encuentran en el CRAN (Comprehensive R Archive Network) que en la actualidad alberga casi 14,000 paquetes distintos.

La instalación de paquetes se hace a través de la página web del CRAN y de alguno de sus repositorios (mirrors). En España existen dos repositorios de R. Pero lo más fácil sin duda es instalarlo desde R mediante una simple línea de comando. Por ejemplo instalemos el paquete *vegan*:

```
> install.packages("vegan", dep=T)
```

En este ejemplo estamos usando una función ya pre-establecida en R "install.package" la cual usa, en este caso dos argumentos "vegan" (nombre del paquete a instalar) y "dep=T" un argumento por el cual le decimos a R que instale todos los paquetes que necesite "vegan" y que no tengamos actualmente en nuestra librería. Para poder trabajar en R con este paquete debemos cargarlo una vez descargado. Esto lo haremos en cada sesión y se hace con la función "library".

```
> library("vegan")
```

4. El entorno de trabajo de Rstudio

Rstudio está organizado en cuatro secciones distintas. La primera vez que entremos a Rstudio veremos solo 3 secciones: la consola de R (izquierda), el "environment" o ambiente de trabajo (arriba a la derecha) y abajo a la izquierda tenemos una serie de pestañas para visualizar nuestros archivos, la ayuda o los paquetes que tenemos instalado. Como se puede observar Rstudio es más amigable desde el punto de vista visual que R y además tiene un sistema basado en ventanas y pestañas con la que podemos ejecutar distintas funciones, como por ejemplo importar datos externos. Siempre que trabajemos en R es recomendable abrir un script, para ello seleccionaremos File – New File – R script. Al hacer esto aparecerá una nueva sección arriba a la izquierda, justo encima de la consola. El script nos permite organizar todo nuestro código dentro de un entorno en el que podemos corregirlo, modificarlo o retocarlo según nuestra necesidad y que además nos permite guardar dicho código para ser usado en otro momento. La consola es donde ejecutamos nuestro código, para ello simplemente escribimos nuestra función y pulsamos "enter", de este modo R nos devolverá el resultado de dicha función. Sin embargo la forma más recomendable de trabajar en R es ejecutar nuestra función desde el script, para ello utilizaremos "run" o bien nos pondremos sobre la línea de código que queremos ejecutar y pulsaremos Ctrl + r o Ctrl + enter en función de la versión o sistema operativo que utilicemos.

5. La esencia del lenguaje R

R es un lenguaje que opera basándose en objetos y funciones. Generalmente en R operamos con cinco tipos distintos de objetos: vectores, matrices, dataframes, listas y funciones. Cada tipo de objetos tiene una serie de características y atributos distintos que son reconocidos por cada uno de los comandos de R. Esto significa que una misma función aplicada a distintos objetos de R puede dar lugar a resultados distintos. Hay que ser cuidadoso con el tipo de objeto con el que trabajamos ya que algunos comandos solo son aplicables a algunos tipos de objetos. Para asignar un objeto en R utilizamos el comando asignación "<-", también se puede utilizar "=" pero pueda dar lugar a confusión con el operador aritmético. Cada vez que se cree un objeto en R

este aparecerá en Rstudio en la sección de arriba a la derecha. Si queremos saber qué tipo de objeto es el que tenemos podemos usar la función “mode” o “class”. Por ejemplo creemos dos objetos x e y al que le asignaremos el número 4.5 y la palabra “objeto”, respectivamente. Antes de esto debemos saber que siempre que asignemos algún valor que sea un texto es imprescindible utilizar comillas.

```
> x <- 4.5
> y <- "objeto"
> mode(x)
[1] "numeric"
> mode(y)
[1] "character"
```

Podemos ver cómo una vez aplicada la función “mode” R nos dice que en el caso de “x” estamos ante un vector numérico y en el caso de “y” ante un vector de texto. Cuando escribimos en un script en R podemos insertar comentarios para que nos sirva de guía sobre qué estamos haciendo y por qué. Para insertar comentarios y que R no los interprete como comandos pondremos el símbolo # antes de escribir nada. Una vez que hemos asignado un valor o valores a un objeto estos quedan guardados en la memoria y podemos consultar el valor de los mismos simplemente llamándolos por su nombre.

```
> x
[1] 4.5
> y
[1] "objeto"
```

En R además existe toda una serie de operaciones aritméticas pre-establecidas y que podemos utilizar como si de una calculadora se tratase. Veamos unos ejemplos:

```
> 5+6+8+9+10
[1] 38
> 5*6
[1] 30
> 60/4
[1] 15
> 5+4+8+20-2
[1] 35
> 5+4*5
[1] 25
> (5+4)*5
[1] 45
> log(35)
[1] 3.555348
> 3^2
[1] 9
> 3^2/3
[1] 3
> exp(1)
[1] 2.718282
> log10(35)
[1] 1.544068
> sqrt(16)
[1] 4
> pi
[1] 3.141593
> sin(pi/2)
[1] 1
> cos(pi/2)
```

```
[1] 6.123032e-17
> tan(pi/2)
[1] 1.633124e+16
> asin(1)
[1] 1.570796
> asin(1)*2
[1] 3.141593
> asin(1)*2-pi
[1] 0
```

5.1. Vectores

Los vectores son los objetos más sencillos y se definen como una entidad simple conformada por una serie de valores ordenados. Por ejemplo podemos crear un vector llamado v1 que consista en 5 valores numéricos, por ejemplo 2.4, 4,6, 3.2, 6.9, 5.2.

```
> v1<- c(2.4, 4,6, 3.2, 6.9, 5.2)
> v1
[1] 2.4 4.0 6.0 3.2 6.9 5.2
```

Cuando se quiere asignar más de un solo valor (o argumento) se usa el comando “c” que significa concatenar. Con los vectores también es posible operar aritméticamente y aplicar funciones a los mismos.

```
> v1
[1] 2.4 4.0 6.0 3.2 6.9 5.2
> 1/v1
[1] 0.4166667 0.2500000 0.1666667 0.3125000 0.1449275 0.1923077
> v1+100
[1] 102.4 104.0 106.0 103.2 106.9 105.2
> log(v1)
[1] 0.8754687 1.3862944 1.7917595 1.1631508 1.9315214 1.6486586
> v2 <- c(1,3,4,8,15,20)
> v1+v2
[1]  3.4  7.0 10.0 11.2 21.9 25.2
> v3 <- v1-v2
> v3
[1]   1.4   1.0   2.0  -4.8  -8.1 -14.8
> mean(v1)
[1] 4.616667
> sd(v1)
[1] 1.718623
> length(v1)
[1] 6
> max(v1)
[1] 6.9
> min(v1)
[1] 2.4
> range(v1)
[1] 2.4 6.9
> summary(v1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.400   3.400   4.600   4.617   5.800   6.900
```

Existen una serie de comandos en R que facilitan la creación de vectores, por ejemplo “:”, “rep” y “seq”.

```
> v3 <- 1:10
> v3
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> v4 <- seq(1,10,2)
> v4
 [1]  1  3  5  7  9
> v5 <- rep(1,10)
> v5
 [1] 1 1 1 1 1 1 1 1 1 1
> v6 <- c(rep(1,3),seq(4,8,0.5))
> v6
 [1] 1.0 1.0 1.0 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
> v7 <- c(vector6,"separar",vector4)
> v7
 [1] "1"       "1"       "1"       "4"       "4.5"     "5"       "5.5"
"6"       "6.5"
[10] "7"       "7.5"     "8"       "separar" "1"       "2"       "3"
"4"       "5"
[19] "6"       "7"       "8"       "9"       "10"
> mode(v7)
[1] "character"
> v7 <- c(v6,rep(0,3),v4)
> v7
 [1]  1.0  1.0  1.0  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  0.0
0.0  0.0  1.0  2.0  3.0
[19]  4.0  5.0  6.0  7.0  8.0  9.0 10.0
> mode(v7)
[1] "numeric"
> v8 <- rep(1:4,2)
> v8
[1] 1 2 3 4 1 2 3 4
> v9 <- rep(1:4,each=2,2)
> v9
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
> v10 <- rep(1:4,1:4)
> v10
 [1] 1 2 2 3 3 3 4 4 4 4
> v11 <- rep(1:4,c(4,1,4,2))
> v11
 [1] 1 1 1 1 2 3 3 3 4 4
> v12 <- rep(c("gato","perro","canario"),c(5,3,1))
> v12
[1] "gato"    "gato"    "gato"    "gato"    "gato"    "perro"   "perro
"   "perro"   "canario"
```

5.2. Matrices y listas

Una matriz en R no es más que un vector con dos atributos más: el número de filas y el número de columnas. Por ejemplo vamos a crear un vector que contengan 1000 datos al azar entre 1 y 10, esto lo haremos con la función “sample”. Posteriormente organizaremos los 1000 datos en una matriz de 100 filas y 10 columnas. Posteriormente utilizaremos la función “str” para conocer la estructura de los datos:

```
> x <- sample(1:10,1000,replace=T)
> y <- matrix(x,nrow=100,ncol=10)
> str(x)
 int [1:1000] 7 1 10 8 6 7 9 10 8 1 ...
> str(y)
 int [1:100, 1:10] 7 1 10 8 6 7 9 10 8 1 ...
```

Como vemos el vector “x” está formado por 1 sola fila y 1000 valores, por el contrario la matriz “y” está conformada por 100 filas con 10 columnas cada una. Ahora es importante saber cómo

seleccionar o extraer datos de estos objetos y para ello utilizaremos los corchetes. Para los vectores nos vale con poner entre corchetes el orden del valor que queremos conocer o extraer mientras que para las matrices tendremos que especificar, en este caso con dos dimensiones, la fila y la columna en donde esté el valor que nos interesa. La fila se selecciona con el primer valor dentro del corchete, mientras que el segundo hace referencia a la columna.

```
> x[5]
[1] 6
> y[1,7]
[1] 8
> y[1,]
 [1] 7 4 1 5 4 4 8 3 4 1
> y[,3]
  [1]  1  4  2  4  7  2  9  1  6 10  2  9  4  4  4  2  1  1  2  2  8 1
0  5  1  3  1  6  8  2 10
 [31]  1 10  6  9  9  4  3  3  7  6  2  1  5  9  3  5  7  7  1  7  6
9  5 10  6  5  3  5 10  4
 [61]  3  6  8  5 10  2  7  3  3  9  1 10  6  1  1 10  5  9  8  2 10
9  7  7  1  1  6  7  4  2
 [91]  3  7  9  8  7  7  5  5  6  5
> y[c(1,5),]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    7    4    1    5    4    4    8    3    4     1
[2,]    6    2    7    9    6    2    7    2    6    10
y[-c(2:4,6:100),]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    7    4    1    5    4    4    8    3    4     1
[2,]    6    2    7    9    6    2    7    2    6    10
```

Cuando ponemos la coma y no especificamos nada antes o después de la misma estaremos seleccionando toda la columna o fila, respectivamente. Igualmente también podemos seleccionar todo menos una fila o columna completa utilizando "-" o seleccionar varias con "c" o usando combinaciones con ":". Si por ejemplo quisiéramos crear un nuevo vector sin los valores 1 a partir de la matriz:

```
> y1 <- y[y>1]
> str(y1)
 int [1:894] 7 10 8 6 7 9 10 8 9 9 ...
> y2 <- y[y!=1]
> str(y2)
 int [1:894] 7 10 8 6 7 9 10 8 9 9 ...
```

Vamos a crear 3 vectores (A, B y C) cada uno perteneciente a una fila de la matriz y, por ejemplo las filas 1, 5 y 7. Posteriormente vamos a crear una matriz (A) con esos tres vectores y una base de datos (datA) con los vectores 5 y 7. Finalmente crearemos un objeto "list" con el vector A, la matriz A y la base de datos A.

```
> vA <- y[1,]
> vB <- y[5,]
> vC <- y[7,]
> matA <- matrix(c(vA,vB,vC),nrow=10, ncol=3)
> datA <- data.frame(vA,vB,vC)
> listA <- list(vC,matA,datA)
> listA
[[1]]
 [1]  9  1  9  3  8 10  9  8  2  4
```

```
[[2]]
      [,1] [,2] [,3]
 [1,]    7    6    9
 [2,]    4    2    1
 [3,]    1    7    9
 [4,]    5    9    3
 [5,]    4    6    8
 [6,]    4    2   10
 [7,]    8    7    9
 [8,]    3    2    8
 [9,]    4    6    2
[10,]    1   10    4

[[3]]
        vA      vB      vC
1        7       6       9
2        4       2       1
3        1       7       9
4        5       9       3
5        4       6       8
6        4       2      10
7        8       7       9
8        3       2       8
9        4       6       2
10       1      10       4
> str(listA)
List of 3
 $ : int [1:10] 9 1 9 3 8 10 9 8 2 4
 $ : int [1:10, 1:3] 7 4 1 5 4 4 8 3 4 1 ...
 $ :'data.frame':      10 obs. of  3 variables:
  ..$ vectorA: int [1:10] 7 4 1 5 4 4 8 3 4 1
  ..$ vectorB: int [1:10] 6 2 7 9 6 2 7 2 6 10
  ..$ vectorC: int [1:10] 9 1 9 3 8 10 9 8 2 4
```

La selección de objetos de la lista se realiza mediante un doble encorchetado en el que el primer valor hace referencia al número del objeto de interés dentro de la lista y el segundo corchete a los valores a extraer:

```
> listA[[1]][4]
[1] 3
> listA[[2]][2,]
[1] 4 2 1
```

5.3. Dataframes

Vamos a crear una base de datos en la que vamos a incluir una variable continua que vamos a denominar “Crecimiento”, otra denominada “ConcentracionN” y otra variable categórica denominada “Fertilización” que va a tener tres niveles “Control”, “Bajo” y “Alto”. Cada una de estas variables será una columna y habrá un total de 60 filas. Si pensamos en un diseño de experimento podríamos estar ante un ensayo de fertilización en campo donde 60 plantas fueron sometidas a 3 tratamientos de fertilización distintos (20 plantas por tratamiento) y donde se midió el crecimiento de las plantas y la concentración en nitrógeno de las hojas. Vamos a crear los datos de forma aleatoria pero que sigan una distribución normal y en el que el crecimiento de las plantas y su concentración en N aumenten con el tratamiento de fertilización. Vamos a utilizar la

función “rnorm” para crear datos aleatorios siguiendo una distribución normal y la función “gl” para crear una variable categórica.

```
> crec1 <- rnorm(20,mean=3.5, sd=0.5)
> crec2 <- rnorm(20,mean=5.0, sd=0.75)
> crec3 <- rnorm(20,mean=7.5, sd=1.25)
> Crecimiento<-(c(crec1,crec2,crec3))
> Crecimiento
 [1] 2.882152 3.685515 3.519634 2.747603 2.915675 3.643760 4.315669 3.
682431 3.330840 3.686191
[11] 3.165776 3.956044 3.462526 2.949562 2.897646 3.063326 3.083253 2.
858005 3.415494 4.065842
[21] 4.570263 5.410939 5.806463 5.696051 4.384500 5.389989 4.671619 4.
388911 4.559401 4.843910
[31] 3.537893 5.692475 5.511125 4.991143 5.549929 5.849718 4.801862 4.
979774 6.043304 4.725729
[41] 6.188517 8.980999 8.788615 8.628569 9.060791 9.016299 7.614057 9.
198583 9.200969 6.656486
[51] 6.168390 5.590017 9.125558 7.866038 7.209036 9.541430 7.949096 6.
113807 7.062101 6.521802
> conc1 <- rnorm(20,mean=1,sd=0.10)
> conc2 <- rnorm(20,mean=1.2,sd=0.15)
> conc3 <- rnorm(20,mean=1.5,sd=0.22)
> ConcentracionN <- c(conc1,conc2,conc3)
> Fertilización <- gl(3,20,labels=c("Control","Bajo","Alto"))
> datos <- data.frame(Crecimiento,ConcentracionN,Fertilizacion)
> str(datos)
'data.frame':  60 obs. of  3 variables:
 $ Crecimiento   : num  2.88 3.69 3.52 2.75 2.92 ...
 $ ConcentracionN: num  0.915 1.002 0.939 0.869 0.92 ...
 $ Fertilizacion : Factor w/ 3 levels "Control","Bajo",..: 1 1 1 1 1 1
1 1 1 1 ...
```

Si ahora quisiéramos añadir una columna con, por ejemplo un número de identificación para cada planta utilizaríamos la función “cbind”.

```
> ID <- 1:60
> datos <- cbind(datos,ID)
> str(datos)
```

Para los objetos dataframe los valores se seleccionan al igual que para los vectores y matrices, o bien utilizamos el comando “$” con el que seleccionaremos una columna del dataframe.

```
> datos[c(1,2),]
  Crecimiento ConcentracionN Fertilizacion ID
1    2.882152      0.9147335       Control  1
2    3.685515      1.0018516       Control  2
> datos[,1]
 [1] 2.882152 3.685515 3.519634 2.747603 2.915675 3.643760 4.315669 3.
682431 3.330840 3.686191
[11] 3.165776 3.956044 3.462526 2.949562 2.897646 3.063326 3.083253 2.
858005 3.415494 4.065842
[21] 4.570263 5.410939 5.806463 5.696051 4.384500 5.389989 4.671619 4.
388911 4.559401 4.843910
[31] 3.537893 5.692475 5.511125 4.991143 5.549929 5.849718 4.801862 4.
979774 6.043304 4.725729
[41] 6.188517 8.980999 8.788615 8.628569 9.060791 9.016299 7.614057 9.
198583 9.200969 6.656486
```

```
[51] 6.168390 5.590017 9.125558 7.866038 7.209036 9.541430 7.949096 6.
113807 7.062101 6.521802
> datos$Crecimiento
 [1] 2.882152 3.685515 3.519634 2.747603 2.915675 3.643760 4.315669 3.
682431 3.330840 3.686191
[11] 3.165776 3.956044 3.462526 2.949562 2.897646 3.063326 3.083253 2.
858005 3.415494 4.065842
[21] 4.570263 5.410939 5.806463 5.696051 4.384500 5.389989 4.671619 4.
388911 4.559401 4.843910
[31] 3.537893 5.692475 5.511125 4.991143 5.549929 5.849718 4.801862 4.
979774 6.043304 4.725729
[41] 6.188517 8.980999 8.788615 8.628569 9.060791 9.016299 7.614057 9.
198583 9.200969 6.656486
[51] 6.168390 5.590017 9.125558 7.866038 7.209036 9.541430 7.949096 6.
113807 7.062101 6.521802
> datos[datos$Fertilizacion=="Control",]
   Crecimiento ConcentracionN Fertilizacion ID
1     2.882152      0.9147335       Control  1
2     3.685515      1.0018516       Control  2
3     3.519634      0.9387375       Control  3
4     2.747603      0.8685142       Control  4
5     2.915675      0.9199732       Control  5
6     3.643760      0.9549626       Control  6
7     4.315669      0.9453476       Control  7
8     3.682431      0.9560430       Control  8
9     3.330840      1.1856105       Control  9
10    3.686191      0.8442612       Control 10
11    3.165776      0.9926739       Control 11
12    3.956044      1.2193011       Control 12
13    3.462526      1.1206020       Control 13
14    2.949562      1.0079288       Control 14
15    2.897646      1.0639835       Control 15
16    3.063326      1.0305831       Control 16
17    3.083253      0.9095490       Control 17
18    2.858005      0.9947515       Control 18
19    3.415494      0.9498117       Control 19
20    4.065842      0.8193273       Control 20
```

También podemos operar con los valores del dataframe:

```
> mean(datos$Crecimiento)
[1] 5.420218
```

5.4. Funciones

A lo largo de los ejemplos anteriores hemos utilizado distintas funciones, por ejemplo “mean”. Estas funciones nos permiten realizar una gran variedad de cálculos que vienen implementados en las mismas. Las funciones trabajan con una serie de argumentos, algunos obligatorios y otros opcionales, que nosotros como usuarios debemos especificar. Como R es de código abierto podemos ver el código detrás de cada función. Si queremos saber qué argumentos tiene y cuáles son los valores de los mismos podemos acudir a los documentos de ayuda (se abrirán en la sección inferior derecha en Rstudio):

```
> ?mean
> help(mean)
```

En caso de que queramos usar una función y esta no haya sido creada anteriormente, podemos crearla nosotros mismos. Por ejemplo vamos a crear una función llamada “horasFinde” con la que se calculará las horas que faltan para el próximo fin de semana. Para eso necesitaremos aportar dos argumentos el día de la semana (numérico) en el que estamos “dia” y la hora del día que es “hora”. ¿Cómo haríamos los cálculos? pues si estuviéramos a las 0 horas del lunes (día = 1) quedarían 120 horas para el fin de semana (24 * 5), entonces primero calcularemos cuantas horas han pasado ya desde el lunes y posteriormente a 120 horas le restaremos las transcurridas. Para hacer una función en R se utiliza la función “function” seguida de los argumentos necesarios (entre paréntesis) y posteriormente los cálculos necesarios entre llaves. Para usar la función lo haremos como en el resto de los ejemplos vistos hasta ahora, es decir con el nombre de la función seguido de los argumentos de la misma:

```
> horasFinde<- function (dia, hora)
+ {
+     h1 <- dia * 24
+     h2 <- h1 - hora
+     result<- 120 - h2
+     return (result)
+ }
> horasFinde(2,16)
[1] 88
```

Otro ejemplo, hagamos una función para calcular la hipotenusa según el teorema de Pitágoras a partir del valor de los catetos (a y b):

```
> pitagoras <- function(a,b)
+ {
+   c1 <- a^2+b^2
+   c <- sqrt(c1)
+   return(c)
+   }
> pitagoras(2,2)
[1] 2.828427
```

6. Comenzando una sesión en R

Es importante saber que R trabaja en un directorio local “working directory” de donde leerá o a donde guardará los datos que generemos. Para saber en qué directorio estamos trabajando utilizaremos la siguiente función:

```
> getwd()
[1] "C:/Users/Enrique/Documents"
```

Para cambiar el directorio de trabajo se usa la función “setwd” seguida del directorio entre comillas, es importante que utilicemos la barra en diagonal hacia la derecha “/” o bien dos hacia la izquierda “\\” para señalar la ruta, si no nos dará error. Es también importante no usar rutas muy largas ya que pueden dar errores.

```
> setwd("C:/Users/Enrique/Desktop/Data analysis/CursoR")
> getwd()
[1] "C:/Users/Enrique/Desktop/Data analysis/CursoR"
```

Para trabajar en R existen una serie de recomendaciones o buenas prácticas. Trabajando desde Rstudio podemos crear un nuevo proyecto lo que nos permitirá tener todos nuestros scripts y objetos guardados y que se cargarían siempre que cargáramos dicho proyecto. Al trabajar con scripts es importante comentar las líneas de código mediante el uso de "#", de este modo cuando se lea el script dentro de un tiempo o sea leído por parte de otro colega pueda entender por qué utilizamos un procedimiento u otro. Trabajando con R es más importante guardar el script que los resultados. Además si guardamos el script con nuestros análisis o gráficos luego podremos aplicarlos fácilmente a otros datos. Para guardar los objetos que hemos creados utilizamos dos funciones: "save" para guardar unos objetos específicos o "save.image" para salvar todos los objetos. Posteriormente para cargarlos utilizaremos la función "load".

```
> save(vector1,vector2,file="objetos.RData")
> save.image(file="objetos.RData")
> load("objetos.RData")
```

6.1. Importar datos externos

En R podemos importar datos de diversa naturaleza y procedentes de distintas plataformas. Existen distintas funciones para importar datos de diversa índole como datos de Excel, de archivos de textos, de otros softwares como SPSS, rasters, shapefiles y un largo etc. También podemos descargar datos que se encuentren online o en cualquier repositorio web. En la mayoría de los casos solemos trabajar con datos que hemos recogido y digitalizado en una hoja de cálculo tipo Excel. Lo más recomendable una vez que tenemos nuestra hoja de cálculo es guardarla no como un archivo Excel sino como un archivo de texto delimitado por tabulaciones (.txt) o bien como un archivo CSV delimitado por comas (.csv). De este modo evitaremos cualquier conflicto entre distintas versiones del software utilizado para la generación de la base de datos. La función utilizada en R para importar datos es "*read.table*" la cual viene acompañada de una serie de argumentos. Generalmente usamos dos argumentos claves "*header*" para decir si la primera fila contiene encabezado (=T) o no (=F) y el argumento "*sep*" para decir que tipo de separación tienen nuestros datos, por ejemplo *sep* = "\t" si los datos están separado por tabulaciones o *sep* = "," si están separados por comas. En caso de que queramos exportar unos datos o los resultados de un análisis utilizaremos la función *"write.table"*, por ejemplo para exportar a un archivo los datos que hemos creado de crecimiento:

```
> write.table(datos, file="crecimiento.csv",sep=",")
> taludes <- read.table(file="taludes.csv", header=T, sep=",")
```

6.2. Funciones básicas para comprobar que los datos son correctos

Lo primero que debemos hacer una vez que hemos importado unos datos es comprobar que se ha hecho correctamente y que no existe ningún tipo de error. Entre estas funciones están: la ya conocida "str" que nos muestra la estructura de los datos, "dim" que nos proporciona el número de columnas y filas, "head" que nos muestra las primeras filas y "names" con el que podemos ver los nombres de las variables. Cuanto más grande sea nuestro set de datos más importante será comprobar que todo está correcto.

```
 > str(taludes)
'data.frame':  91 obs. of  3 variables:
```

```
 $ Biomasa: num  3.04 3.57 3.67 4.59 4.34 ...
 $ Luz    : Factor w/ 3 levels "Nivel 1","Nivel 2",..: 1 1 1 1 1 1 1 1
1 1 ...
 $ Agua   : Factor w/ 2 levels "Lluvia estival",..: 2 2 2 2 2 2 2 2
2 ...
> dim(taludes)
[1] 91  3
> head(taludes)
   Biomasa     Luz           Agua
1 3.039767 Nivel 1 Sequia estival
2 3.573515 Nivel 1 Sequia estival
3 3.669172 Nivel 1 Sequia estival
4 4.592334 Nivel 1 Sequia estival
5 4.337278 Nivel 1 Sequia estival
6 2.254230 Nivel 1 Sequia estival
> names(taludes)
[1] "Biomasa" "Luz"     "Agua"
```

Uno de los errores más comunes es que R no reconozca correctamente la naturaleza de alguna variable. Para cambiar la naturaleza de las variables utilizamos una serie de funciones dependiendo de la transformación que queramos conseguir, por ejemplo “as.numeric” o “as.factor”. Por ejemplo, vamos a añadir una variable a nuestro set de datos de crecimiento a la que llamaremos “Fecha” que será un factor con dos niveles “1” y “2”.

```
Fecha <- rep(c(rep(1,10),rep(2,10)),3)
datos <- cbind(datos,Fecha)
str(datos)
'data.frame':  60 obs. of  5 variables:
 $ Crecimiento   : num  3.19 2.91 2.73 3.46 1.61 ...
 $ ConcentracionN: num  0.806 0.968 0.968 0.85 1.123 ...
 $ Fertilizacion : Factor w/ 3 levels "Control","Bajo",..: 1 1 1 1 1 1
1 1 1 1 ...
 $ ID            : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Fecha         : num  1 1 1 1 1 1 1 1 1 1 ...
```

Como vemos R ha reconocido la Fecha como una variable numérica cuando debería ser un factor.

```
> datos$Fecha <- as.factor(datos$Fecha)
> str(datos)
'data.frame':  60 obs. of  5 variables:
 $ Crecimiento   : num  3.19 2.91 2.73 3.46 1.61 ...
 $ ConcentracionN: num  0.806 0.968 0.968 0.85 1.123 ...
 $ Fertilizacion : Factor w/ 3 levels "Control","Bajo",..: 1 1 1 1 1 1
1 1 1 1 ...
 $ ID            : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Fecha         : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

Si quisiera cambiar el nombre de alguna de las columnas, por ejemplo “date” en vez de “Fecha”

```
> colnames(datos)[5] <- "date"
> str(datos)
'data.frame':  60 obs. of  5 variables:
 $ Crecimiento   : num  3.19 2.91 2.73 3.46 1.61 ...
 $ ConcentracionN: num  0.806 0.968 0.968 0.85 1.123 ...
 $ Fertilizacion : Factor w/ 3 levels "Control","Bajo",..: 1 1 1 1 1 1
1 1 1 1 ...
 $ ID            : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
 $ date          : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

6.3. Funciones para la manipulación de datos

Cuando trabajamos con un set de datos existen una serie de funciones muy útiles para extraer o resumir información. Por ejemplo las funciones “tapply” y “aggregate” agrupan los valores de un determinado vector o conjunto de datos, respectivamente, de acuerdo a una variable y les aplica una determinada función. La principal diferencia entre ambas es que “tapply” se aplica a una variable simple y “aggregate” puede usarse con varias. Veamos un ejemplo:

```
> tapply(datos$Crecimiento,datos$Fertilizacion,mean)
 Control     Bajo     Alto
3.326713 4.946403 7.507952
> aggregate(datos[,1:2],by=list(datos$Fertilizacion),mean)
  Group.1 Crecimiento ConcentracionN
1 Control    3.326713      0.9971316
2    Bajo    4.946403      1.2089181
3    Alto    7.507952      1.4896771

> aggregate(datos[,1:2],by=list(datos$Fertilizacion,datos$date),mean)
  Group.1 Group.2 Crecimiento ConcentracionN
1 Control       1    3.234808      0.9917488
2    Bajo       1    4.961571      1.1923134
3    Alto       1    7.510661      1.5251207
4 Control       2    3.418618      1.0025144
5    Bajo       2    4.931235      1.2255228
6    Alto       2    7.505243      1.4542334
```

Una función muy útil, no solo para aplicar sobre datos pero también sobre otros tipos de objetos es “summary”. Esta función proporciona un resumen del objeto en cuestión siendo distinta la información que muestra en función del objeto al cual se aplique:

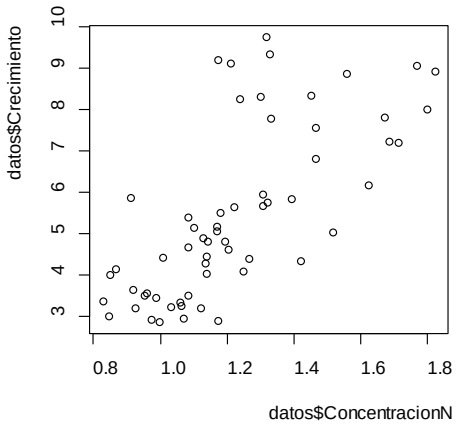
```
> summary(datos)
  Crecimiento     ConcentracionN   Fertilizacion       ID        Fecha
 Min.   : 2.297   Min.   :0.8405   Control:20    Min.   : 1.00   1:30
 1st Qu.: 3.780   1st Qu.:1.0043   Bajo   :20    1st Qu.:15.75   2:30
 Median : 5.015   Median :1.1850   Alto   :20    Median :30.50
 Mean   : 5.200   Mean   :1.2052                 Mean   :30.50
 3rd Qu.: 6.057   3rd Qu.:1.4092                 3rd Qu.:45.25
 Max.   :10.246   Max.   :1.8329                 Max.   :60.00
```

7. Gráficos simples

Como ya hemos comentado, R permite realizar una gran variedad de gráficos. Además cada función usada para la realización de gráficos contiene una enorme variedad de opciones que permiten la completa personalización de los mismos. Cuando creemos un gráfico en Rstudio este aparecerá en la sección de abajo a la derecha. A este gráfico podremos añadirle distintos elementos y exportarlos a una amplia variedad de formatos.

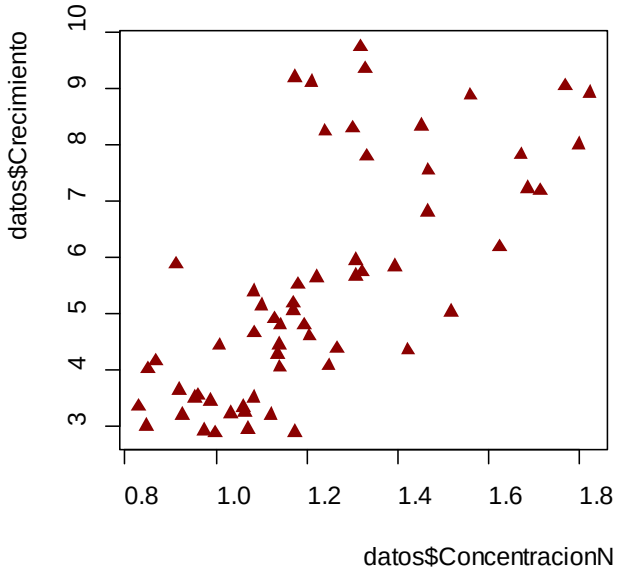
El gráfico más sencillo que podemos hacer es representar dos variables numéricas continuas, para ello utilizamos la función “plot”. Por ejemplo representemos para nuestros datos creados la variable crecimiento respecto a la concentración de N.

```
> plot(datos$ConcentracionN,datos$Crecimiento)
```



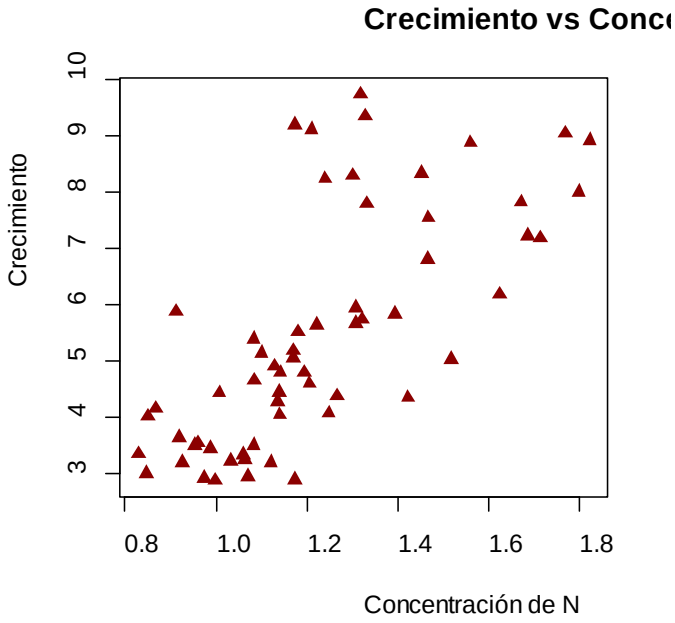
Podemos por ejemplo cambiar la forma y el color de nuestros puntos con los argumentos “pch” y “col”:

```
> plot(datos$ConcentracionN,datos$Crecimiento, pch=17, col="darkred")
```



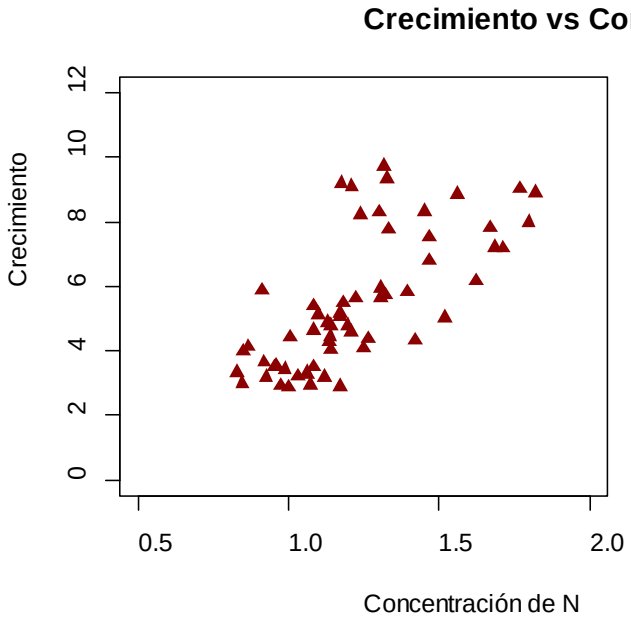
También podemos cambiar el nombre a los ejes y añadir un título:

```
> plot(datos$ConcentracionN,datos$Crecimiento, pch=17, col="darkred",
+      main="Crecimiento vs Concentración de N",
+      xlab="Concentración de N", ylab="Crecimiento")
```



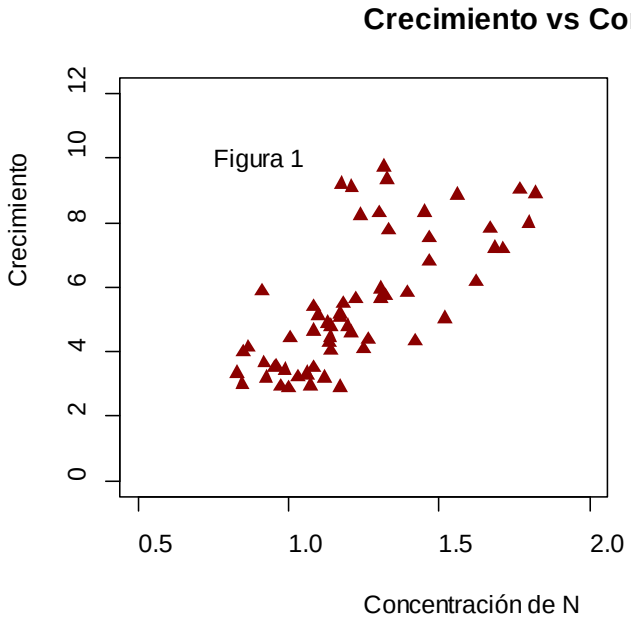
Y también podemos extender los límites de nuestros ejes:

```
> plot(datos$ConcentracionN,datos$Crecimiento, pch=17, col="darkred",
+      main="Crecimiento vs Concentración de N",
+      xlab="Concentración de N", ylab="Crecimiento",
+      ylim=c(0,12), xlim=c(0.5,2))
```



Incluso podríamos añadir un texto, donde quisiéramos, con la función “text”. Este texto se añadirá en el último gráfico creado.

```
> text(0.75,10,"Figura 1")
```

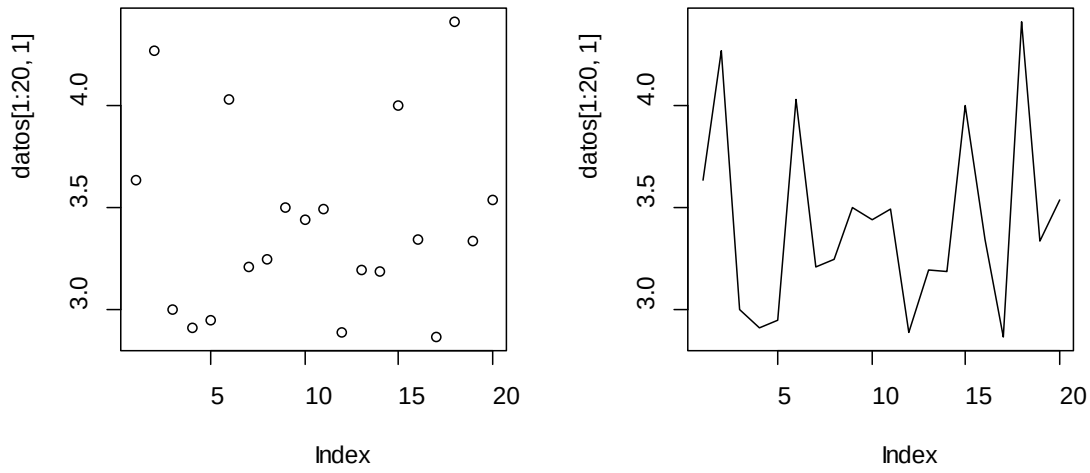


Con Rstudio podemos exportar fácilmente los gráficos realizados, simplemente haciendo uso del botón “export”. Pero también podemos exportar los gráficos a partir de la línea de comando. Para ello hay que seguir una determinada lógica, primero crear el archivo con el formato en el que queremos la exportación (jpeg, pdf, bitmap, bmp, etc), luego realizar el gráfico y finalmente cerrar la edición del gráfico con “dev.off()”. Vamos a exportar uno de nuestros gráficos a un archivo png. Cada tipo de archivo tiene una función asociada para su exportación con una serie de argumentos, algunos generales y otros específicos, recuerda usar la ayuda. Recordad que el archivo se os guardará en vuestro directorio de trabajo.

```
> png(file="figura1.png")
> plot(datos$ConcentracionN,datos$Crecimiento, pch=17, col="darkred",
+      main="Crecimiento vs Concentración de N",
+      xlab="Concentración de N", ylab="Crecimiento",
+      ylim=c(0,12), xlim=c(0.5,2))
> text(0.75,10,"Figura 1")
> dev.off()
```

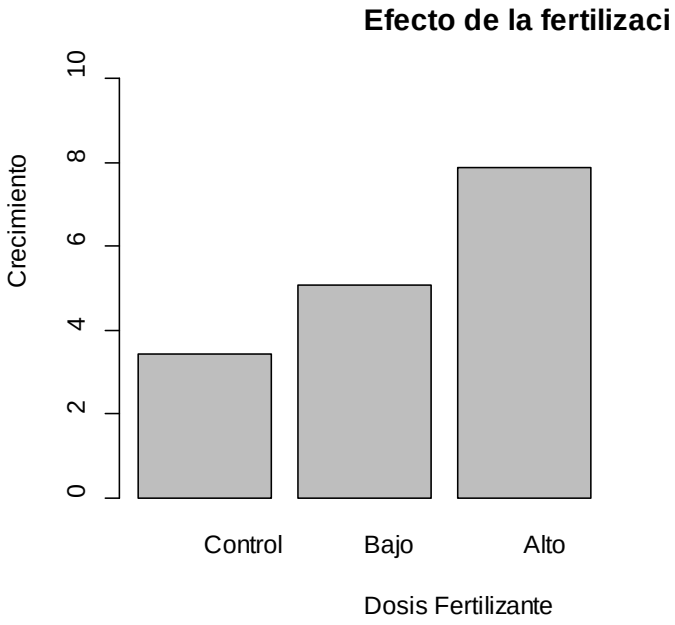
Como hemos visto, la función “plot” nos representa un gráfico de dispersión, en el caso anterior entre dos variables. Pero también podemos utilizar “plot” para dibujar una serie de valores y unirlos mediante líneas por ejemplo. También podemos dibujar un gráfico compuesto, es decir incluir distintos gráficos en una misma ventana. Vamos a dibujar dos gráficos uno en el que se representarán los primeros 20 valores de crecimiento y otro que será lo mismo pero uniremos los puntos mediante una línea, esto lo haremos con el argumento “type”. Para dividir la ventana de gráficos en varias usaremos “par(mfrow=c(x,y))” donde x son las filas e y las columnas.

```
> par(mfrow=c(1,2))
> plot(datos[1:20,1],type="p")
> plot(datos[1:20,1],type="l")
```



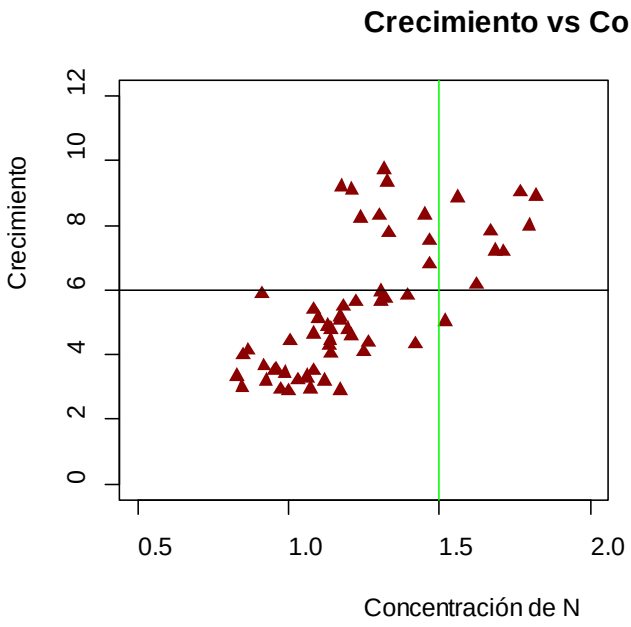
También pueden crearse gráficos de barras con la función “barplot”. Por ejemplo representemos el valor medio del crecimiento para cada tratamiento de fertilización:

```
> medias <- as.vector(tapply(datos$Crecimiento,datos$Fertilizacion,mea
n))
> barplot(medias, ylim=c(0,10), ylab="Crecimiento",
+         xlab=c("Dosis Fertilizante"),
+         names.arg=levels(datos$Fertilizacion),
+         main="Efecto de la fertilización en el crecimiento")
```



Además existe toda una serie de funciones que nos permiten añadir elementos al gráfico con solo especificar las cordenadas x e y, por ejemplo puntos “points(x,y)”, líneas “lines(x,y)”, texto “text(x,y,texto)”, polígonos “polygon(x,y)”, leyenda “legend(x,y,leyenda)”, rectas “rect(x1,y1,x2,y2) o segmentos “segments(x0,y0,x1,y1). Además podemos añadir líneas verticales u horizontales:

```
> plot(datos$ConcentracionN,datos$Crecimiento, pch=17, col="darkred",
+      main="Crecimiento vs Concentración de N",
+      xlab="Concentración de N", ylab="Crecimiento",
+      ylim=c(0,12), xlim=c(0.5,2))
> abline(h=6)
> abline(v=1.5,col="green")
```



8. Un paso más allá: repetir procedimientos y establecer condicionantes

Algo muy común en programación es querer repetir un procedimiento *n* veces, para ello se utiliza lo que se conoce como *loop*. En R, un *loop* se consigue mediante el comando "for". Lo que hacemos es pedir a R que algo (*i*) tomado en una secuencia de valores (*in* value:value) se ejecute dentro de un comando o función determinada tantas veces como valores diferentes de *i* hayamos determinado. Por ejemplo si quisiéramos repetir una operación un número determinado de veces:

```
> for (i in 1:5)
+   {
+   print(i^2)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25

> x <- 2
> for(i in 1:3)
+ {
+   x <- x + 8
+   }
> x
[1] 26
```

```
> j <- 0
> k <- 0
> for (i in 1:5) {
+   j <- j+1
+   k <- k+i*j
+   print(i+j+k) }
[1] 3
[1] 9
[1] 20
[1] 38
[1] 65

> h <- c("Pinus nigra","Pinus pinea","Pinus pinaster","Pinus sylvestri
s","Pinus canariensis", "Pinus radiata")
> for (i in 1:6)
+ { print(h[i])
+   }
[1] "Pinus nigra"
[1] "Pinus pinea"
[1] "Pinus pinaster"
[1] "Pinus sylvestris"
[1] "Pinus canariensis"
[1] "Pinus radiata"
```

También podemos utilizarlo para remplazar valores

```
> m <- c(1:10)
> for (i in 5:10)
+ { m[i]<-10*m[i]
+ }
> m
 [1]   1   2   3   4  50  60  70  80  90 100
```

También podemos hacer bucles anidados, muy útiles cuando trabajemos con matrices:

```
> mat <- matrix(c(1:100), nrow=10, ncol=10)
> mat
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100
> for (i in 1:10)
+ { for (j in i:10)
+ { mat[i,j]<-mat[i,j]-mat[i,j]
+ }
+ }
> mat
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    2    0    0    0    0    0    0    0    0     0
 [3,]    3   13    0    0    0    0    0    0    0     0
 [4,]    4   14   24    0    0    0    0    0    0     0
 [5,]    5   15   25   35    0    0    0    0    0     0
 [6,]    6   16   26   36   46    0    0    0    0     0
 [7,]    7   17   27   37   47   57    0    0    0     0
```

```
 [8,]    8   18   28   38   48   58   68    0    0     0
 [9,]    9   19   29   39   49   59   69   79    0     0
[10,]   10   20   30   40   50   60   70   80   90     0
```

También se pueden añadir condicionales “if” y “else”:

```
> v <- c(1,4,5,6,2)
> k <- NULL
> for (i in 1:5)
+ {if (v[i]>=5)
+   {k<-"Yes"}
+ print(k)}
NULL
NULL
[1] "Yes"
[1] "Yes"
[1] "Yes"
```

```
> v <- c(1,4,5,6,2)
> k <- NULL
> for (i in 1:5)
+ {if (v[i]>=5)
+ {k<-"Yes"}
+   else {
+     k <- "No"}
+   print(k)}
[1] "No"
[1] "No"
[1] "Yes"
[1] "Yes"
[1] "No"
```

Sin embargo, los *loops* cuando se aplican a grandes bases de datos suelen ser lentos por lo que se recomienda su uso únicamente cuando sea estrictamente necesario. Existen otras funciones como “apply” y “tapply” que permiten extraer información de datos sin la necesidad de aplicar buc les. La función “tapply” ya la hemos usado, “apply” permite aplicar una determinada función a las filas o columnas de una matriz. Para que la función se aplique a filas o columnas es necesario us ar el argumento “MARGIN”, “MARGIN=1” para filas y “MARGIN=2” para columnas. Por ejemplo, vamos a calcular el sumatorio de las filas y columnas de la matriz “mat”, anteriormente creada.

```
> mat
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    2    0    0    0    0    0    0    0    0     0
 [3,]    3   13    0    0    0    0    0    0    0     0
 [4,]    4   14   24    0    0    0    0    0    0     0
 [5,]    5   15   25   35    0    0    0    0    0     0
 [6,]    6   16   26   36   46    0    0    0    0     0
 [7,]    7   17   27   37   47   57    0    0    0     0
 [8,]    8   18   28   38   48   58   68    0    0     0
 [9,]    9   19   29   39   49   59   69   79    0     0
[10,]   10   20   30   40   50   60   70   80   90     0
> apply(mat,1,sum)
 [1]   0   2  16  42  80 130 192 266 352 450
> apply(mat,2,sum)
 [1]  54 132 189 225 240 234 207 159  90   0
```

9. Lecturas recomendadas

i. Adler J (2012) R in a nutshell. O'Reilly.

ii. Cayuela L (2014) Una introducción a R (versión 1.5).

iii. Crawley MJ (2007) The R Book. Wiley.

iv. Crawley MJ (2013) The R Book. Second edition. Wiley.

v. Venables WN, Smith DM, R Core Team (2013) An introduction to R.

vi. Zuur AF, Ieno EN, Meesters EHWG (2009) A beginner's guide to R. Springer

Ejercicios adicionales

1- Crea una matriz de datos que siga una distribución normal estándar, con 10 columnas y 20 fila s en cada columna. Una vez creada esta matriz, seleccionar primero el elemento que ocupa la fil a 5 de la tercera columna. Después selecciona todos los elementos de la fila 5. Luego todos los elementos de la columna 2. Finalmente selecciona todos los elementos desde la columna 1 a la 5.

2- Crea una nueva matriz de datos que contenga las cinco últimas columnas de la matriz anterior y una nueva variable que contenga números del 1 al 20.

3- Para el archivo taludes, calcula los valores medios de biomasa por nivel de luz, por nivel de ag ua y por ambos niveles juntos. Elabora un gráfico de barras para mostrar el efecto de la luz sobre la biomasa y otro para mostrar el efecto del agua.

Análisis exploratorio de datos

Análisis de datos y modelos lineales en R

Enrique Andivia[1]

Master Universitario en Restauración de Ecosistemas
Curso 2019-2020

1 Departamento de Biodiversidad, Ecología y Evolución. Universidad Complutense de Madrid

email: e.andivia@gmail.com

web: http://enriqueandivia.wixsite.com/forests

ANÁLISIS EXPLORATORIO DE DATOS

1. Conceptos básicos

1.1. Tipos de variables

Siguiendo el método científico una vez que formulamos una hipótesis, en la mayoría de los casos, debemos diseñar un estudio para comprobar si nuestra hipótesis se cumple o no. Los primeros pasos en este diseño están relacionado con la elección de las variables del estudio. En un sentido amplio hablamos de dos tipos de variables: variables respuesta (o dependientes) y variables explicativas (también llamadas independientes o covariables). La variable respuesta es aquella que cuantifica o describe el fenómeno que queremos estudiar y en un sentido estadístico será aquella que modelaremos en función de las distintas variables independientes. Por lo tanto las variables independientes o explicativas son aquellas responsables, a priori del fenómeno estudiado. En función de si se evalúa una o varias variables respuestas al mismo tiempo distinguimos entre estadística univariante o multivariante. En este curso nos centraremos en estadística univariante, es decir en modelos donde solo una variable respuesta está implicada, independientemente del número de variables independientes o explicativas que consideremos.

Las variables, sean respuestas o explicativas, también se clasifican en función de su naturaleza en variables cuantitativas y cualitativas. Es importante asegurarnos de que R ha identificado la naturaleza de nuestras variables correctamente ya que esto tiene implicaciones y restricciones para el análisis estadístico. Las variables cuantitativas pueden a su vez distinguirse entre variables continuas y discretas. Ejemplos de este tipo de variables son: peso, altura, temperatura, crecimiento, concentración, número de individuos, etc. Ejemplos de variables cualitativas o categóricas son: género, país, mes, días, etc. Generalmente en estadística llamamos factores a las variables independientes categóricas y niveles a cada una de las categorías. Por ejemplo en el ejemplo del fertilizante, el tratamiento sería una variable categórica y “Control”, “Bajo” y “Alto” serían los niveles del factor. Es importante que ajustemos el tipo de análisis estadístico a los objetivos que perseguimos y al tipo de datos de los que dispongamos. Además cuando diseñemos nuestro estudio o nuestro muestreo deberíamos tener en cuenta qué análisis estadístico vamos a utilizar e incluso modificar el muestreo acorde al posterior análisis de los datos. Debemos tener siempre en cuenta que no hay necesidad de utilizar un análisis estadístico más complicado del que necesitamos para responder a nuestras preguntas.

Otro aspecto importante a tener en cuenta es la presencia de lo que conocemos como valores perdidos “missing values”. Estos valores perdidos pueden ser debido a errores de muestreo y no deben confundirse con ceros. En R la denominación para este tipo de datos es NA. Es importante saber que en algunas funciones uno de los argumentos está relacionado con la consideración o no de estos valores (na.rm=T). Por ejemplo:

```
> a <- c(2,5,NA,5,4,NA)
> mean(a)
[1] NA
> mean(a,na.rm=T)
[1] 4
> a[is.na(a)]<-0
> a
[1] 2 5 0 5 4 0
```

1.2. Estadística descriptiva

La estadística descriptiva permite la organización, resumen y descripción de un conjunto de datos. Así podemos conocer cómo se encuentra distribuida una variable, algunas medidas de tendencia central o cuál es la variabilidad y dispersión de nuestros datos. Es muy importante que para cada una de nuestras variables conozcamos una serie de parámetros que nos proporcionan información sobre las mismas.

Entre las medidas de tendencia central más populares se encuentran la media y la mediana y entre las de dispersión la varianza, la desviación estándar o la desviación típica.

Vamos a crear unos datos y calcular estos parámetros, primero realizando nosotros mismos los cálculos y posteriormente usando las funciones ya establecidas en R.

```
> a <- rnorm(100,5,0.8)
> media <- sum(a)/length(a)
> media
[1] 5.077931
> mean(a)
[1] 5.077931
> median(a)
[1] 5.132992
> max(a)
[1] 6.76081
> min(a)
[1] 3.596144
> range(a)
[1] 3.596144 6.760810
> max(a)-min(a)
[1] 3.164665
> summary(a)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.596   4.485   5.133   5.078   5.617   6.761
> b <- NULL
> varianza <- function(datos)
+ { for (i in 1:length(datos))
+ { b[i]<-(datos[i]-mean(datos))^2
+   }
+   print(sum(b)/(length(datos)-1))}
> varianza(a)
[1] 0.5740517
> var(a)
[1] 0.5740517
> sqrt(var(a))
[1] 0.757662
> sd(a)
[1] 0.757662
> sd(a)/sqrt(length(a))
[1] 0.0757662
```

Tanto la media como la varianza son estimaciones de la población hechas a partir de una determinada muestra. Estos estimadores pueden llevar un sesgo asociado, por ejemplo en el caso de la media se dice que es un estimador insesgado de la media poblacional cuando las observaciones son seleccionadas de manera aleatoria, cuando dichas observaciones son independientes entre sí y cuando las observaciones provienen de una población mayor en la que dicha variable sigue una distribución aleatoria normal. En el caso del estimador de la varianza

poblacional (i.e. varianza muestral) nuestro denominador es *n-1*, en vez de *n*. La razón es que si utilizáramos *n* dicho estimador estaría sesgado ya que ya hemos utilizado toda la muestra para la estimación del parámetro de la media. Vamos a ilustrar esto con un ejemplo, si tomáramos una sola observación de una población, la media sería igual a dicha observación y la varianza sería 0, ya que la estimación de la media poblacional es igual a la propia observación. El problema reside en que ya hemos utilizado dicha observación para estimar la media y no tenemos ninguna otra observación independiente para estimar la varianza. Es por ello que se necesita al menos dos observaciones para estimar la varianza y por ello el estimador de la misma utiliza *n-1* como denominador. Esto nos lleva a la definición de grados de libertad. Los grados de libertad representa el número de observaciones independientes en una determinada muestra para estimar parámetros estadísticos.

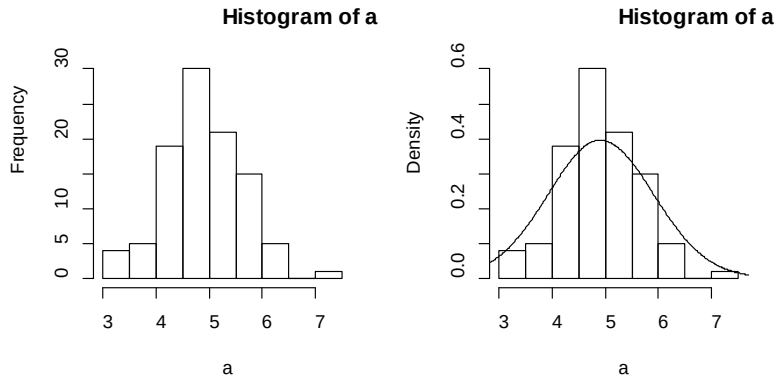
Los parámetros de dispersión de datos dependen del valor medio al que van asociado por lo que no aportan una gran información por sí solo. Una forma independiente de expresar la variabilidad de los datos es a partir del coeficiente de variación (CV) que se define como la desviación estándar dividido por el valor medio (puede expresarse en %). Este coeficiente es útil si queremos comparar la variabilidad de dos o más variables.

A partir de la desviación estándar se suele definir el intervalo de confianza alrededor de la media. Para una variable con una distribución normal, aproximadamente el 95% de las observaciones se dan en el rango $\bar{x}$-1.96×sd y $\bar{x}$+1.96×sd. Esto significa que hay un 95% de probabilidades de que al muestrear una determinada población el valor medio de nuestra variable de interés esté dentro de dicho intervalo.

1.3. La distribución normal

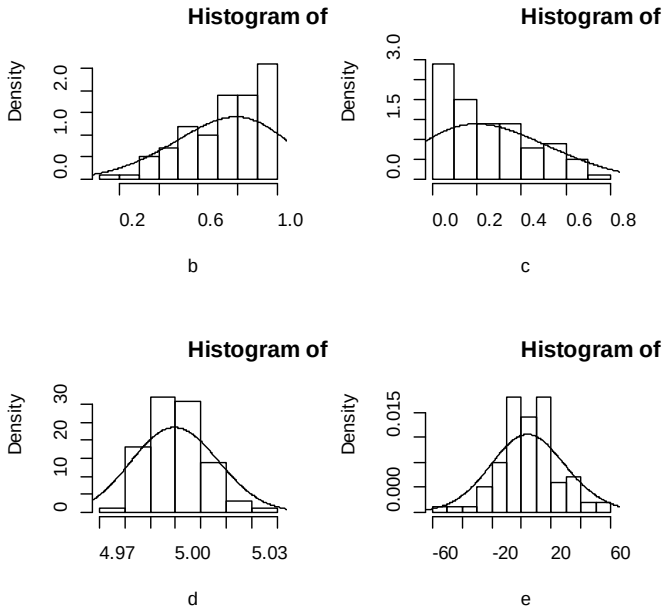
La distribución normal, gaussiana o de campana de Gauss es la base teórica sobre la que se asienta técnicas como el análisis de la varianza. El histograma de una variable con una distribución normal presenta una serie de características que facilita la generalización de algunas teorías estadísticas que son la base de muchas técnicas de análisis de datos.

```
> par(mfrow=c(1,2))
> hist(a)
> hist(a,probability=TRUE)
> lines(density(a,adjust=3))
```



Como observamos, la mayoría de las observaciones se encuentran agrupadas en torno a los valores centrales, aunque las observaciones se extiendan hacia ambos lados. Esto hace que tengamos una distribución aproximadamente simétrica. Existen una serie de parámetros asociado a la forma de la distribución de nuestra variable de interés. Así, la asimetría (skewness) describe cómo difiere la distribución de nuestra variable de una distribución simétrica. De este modo una distribución normal presenta un valor de asimetría igual a 0, mientras que si es mayor o menor de 0 diremos que la distribución está desplazada con respecto al centro hacia la derecha o la izquierda, respectivamente. Por otro lado, la kurtosis nos informa de cómo de agrupados están los datos en el centro de la distribución, con valores negativos indicando una mayor probabilidad en el centro de la distribución que en los extremos, y valores positivos indicando lo contrario. Para calcular la asimetría y la kurtosis utilizaremos el paquete “moments”.

```
> b <- rbeta(100,3,1)
> c <- rbeta(100,1,3)
> d <- rnorm(100,5,0.01)
> e <- rnorm(100,5,20)
> skewness(b)
[1] -0.4806235
> skewness(c)
[1] 0.6947095
> kurtosis(d)
[1] 3.018295
> kurtosis(e)
[1] 2.848715
> par(mfrow=c(2,2))
> hist(b,probability=TRUE)
> lines(density(b,adjust=3))
> hist(c,probability=TRUE)
> lines(density(c,adjust=3))
> hist(d,probability=TRUE)
> lines(density(d,adjust=3))
> hist(e,probability=TRUE)
> lines(density(e,adjust=3))
```



2. Outliers

Un outlier es una observación que presenta un valor muy grande o muy pequeño en comparación con la mayor parte de las observaciones realizadas. El problema que presentan estos outliers desde el punto de vista estadístico es que pueden influenciar los resultados de nuestros análisis. De todos modos la presencia de outliers no tiene porqué ser siempre un error de muestreo o un problema, de hecho en función de nuestra pregunta podríamos estar interesados en esos outliers. A lo largo de esta sección vamos a mostrar una serie de herramientas para detectar posibles outliers en nuestros datos y cómo lidiar con ellos.

2.1. Identificación de outliers potenciales en una dimensión

Antes de discutir si una observación es o no realmente un outliers, debemos identificar todas aquellas observaciones que son considerablemente mayores o menores a la mayoría y que por lo tanto pueden ser potencialmente consideradas como outliers. Lo primero que vamos a hacer es identificar estos potenciales outliers en cada una de nuestras variables numéricas (dependientes o independientes) por separado.

La primera herramienta para detectar estos outliers son los boxplots (diagrama de cajas y bigotes). Estos gráficos están basados en cuantiles (cuartiles y percentiles). Por ejemplo el cuartil Q2 es aquel que deja a un lado y a otro la mitad de nuestras observaciones, en otras palabras la mediana. En un boxplot (Figura 1) se representa normalmente la mediana (mediante una línea), los cuartiles Q1 y Q3 (límites de la caja) y unas líneas que representan 1.5 veces a cada lado de la caja la distancia entre el Q1 y el Q3.

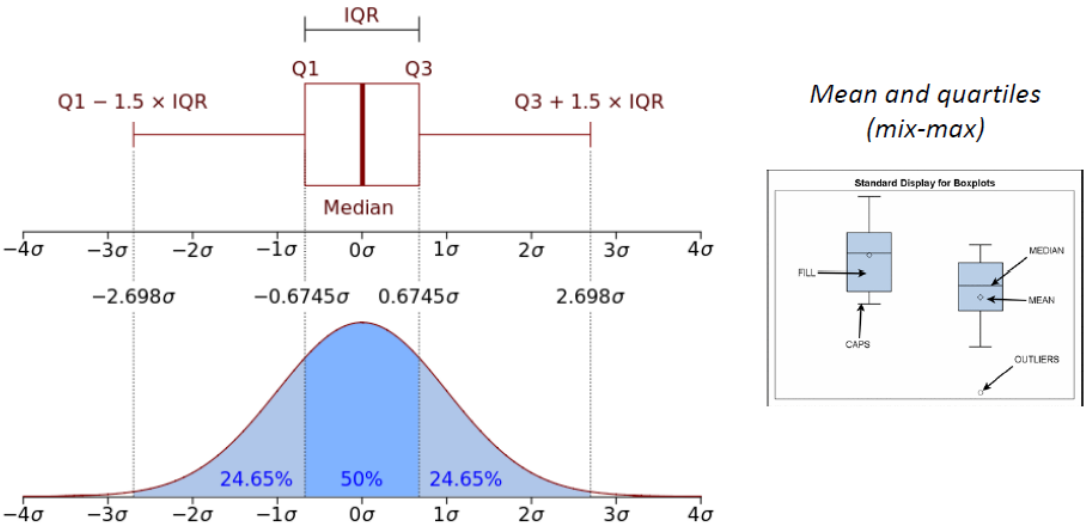
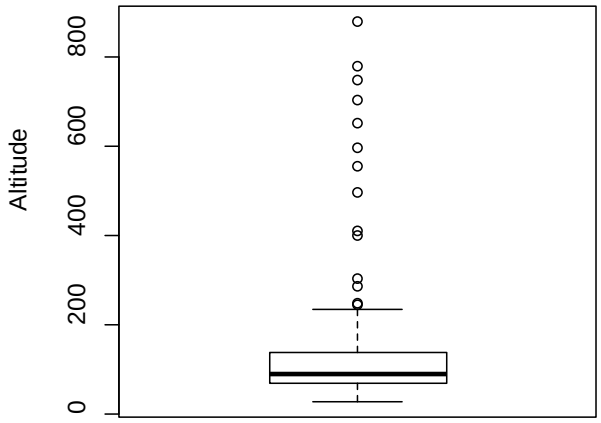


Figura 1: Representación gráfica de un boxplot (modificado de Ruiz-Benito (2016), Universidad de Alcalá)

Vamos a realizar un boxplot para analizar si existen outliers potenciales en el set de datos IrishPh (Zuur et al. 2009).

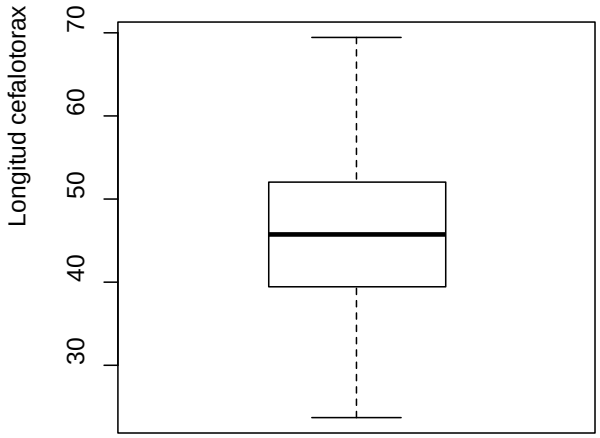
```
> setwd("C:/Users/Enrique/Desktop/Data analysis/CursoR")
> IrishPh <- read.table (file="IrishPh.txt", header=T)
> str(IrishPh)
'data.frame':  210 obs. of  7 variables:
```

```
 $ ID       : int  27 28 29 30 31 32 33 34 35 36 ...
 $ Easting  : int  117294 114989 115931 120904 147715 159527 157561 157
351 158982 167071 ...
 $ Northing: int  161990 165877 182021 181242 184192 189332 176617 172
758 165623 170498 ...
 $ Altitude: int  45 73 75 72 89 60 68 48 49 65 ...
 $ Forested: int  2 1 2 2 2 2 1 2 2 2 ...
 $ pH       : num  7.24 6.45 6.92 7.09 6.98 7.36 7.24 7.37 7.34 7.65 ..
.
 $ SDI      : num  48.8 73.5 67.1 53.2 57.9 ...
> boxplot (IrishPh$Altitude, ylab="Altitude")
```



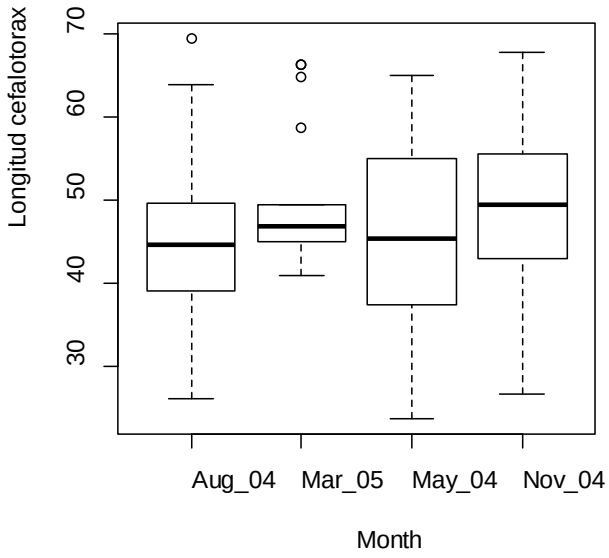
Como vemos nuestro boxplot nos identifica 10 observaciones que podrían ser potencialmente outliers. Algunas personas prefieren borrar estos puntos directamente, la recomendación general es apuntar que son potenciales outliers y seguir viendo qué ocurre. Veamos otro ejemplo, está vez con la base de datos Procamburus.txt (Ieno y Zuur 2015) que muestra el efecto del mes y el sexo en la longitud del cefalotórax del cangrejo de río (*Procambarus clarkii*).

```
> Crayfish <- read.table(file="Procambarus.txt", header=T)
> str(Crayfish)
'data.frame':  746 obs. of  10 variables:
 $ Month   : Factor w/ 4 levels "Aug_04","Mar_05",..: 3 3 3 3 3 3 3 3
3 3 ...
 $ CTL     : num  47.6 45.1 59.7 51.2 59.6 44.5 45.8 42.2 54.9 65.1 ..
.
 $ CW      : num  19.7 18.6 26.4 21.4 24.4 18.7 20.5 17.7 25.8 28.6 ..
.
 $ PLdx    : num  23.8 NA 39.5 39.1 42.8 24.5 27.2 24.6 47.4 65.7 ...
 $ PWdx    : num  17.8 NA 15 13.6 14.9 9.2 11 7.4 17.3 21.5 ...
 $ PLsx    : num  NA 26 40.7 39.5 41.2 26.9 25.4 16.3 51.5 NA ...
 $ PWsx    : num  NA 7.8 14.3 13.8 15.4 9.5 10.5 3.8 19.5 NA ...
 $ Weight  : num  23.5 16.5 50 33 50 20 23 16 52 63 ...
 $ Sex     : Factor w/ 2 levels "f","m": 1 2 1 2 1 1 1 2 2 2 ...
 $ Maturity: int  4 2 4 1 4 3 3 2 1 1 ...
> boxplot(Crayfish$CTL, ylab= "Longitud cefalotorax")
```



En este caso no se ha identificado ningún punto como potencialmente sospechoso de ser un outlier. Una extensión muy útil de los boxplots son los boxplots condicionales. Estos gráficos sirven, además de para identificar outliers, para examinar diferencias entre la mediana de diferentes grupos o como un diagnóstico para identificar heterogeneidad de varianzas entre los distintos niveles de un factor, lo cual veremos más adelante. En este caso vamos a crear un boxplot de la longitud del cefalotórax en función del mes.

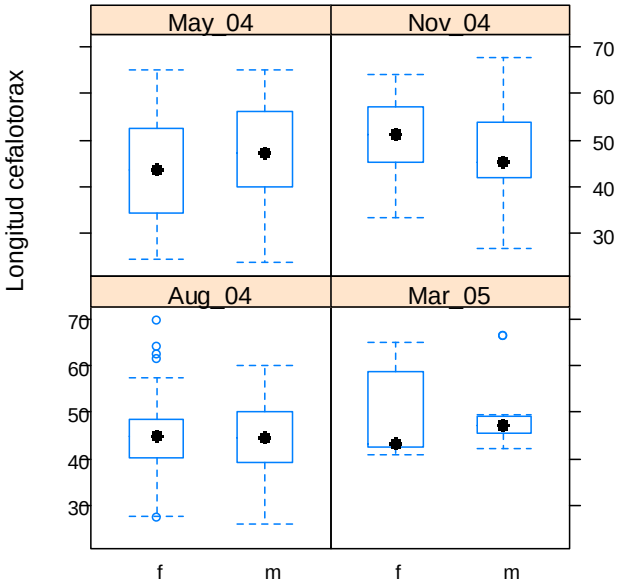
```
> boxplot(Crayfish$CTL ~ Crayfish$Month, ylab="Longitud cefalotorax",
xlab="Month")
```



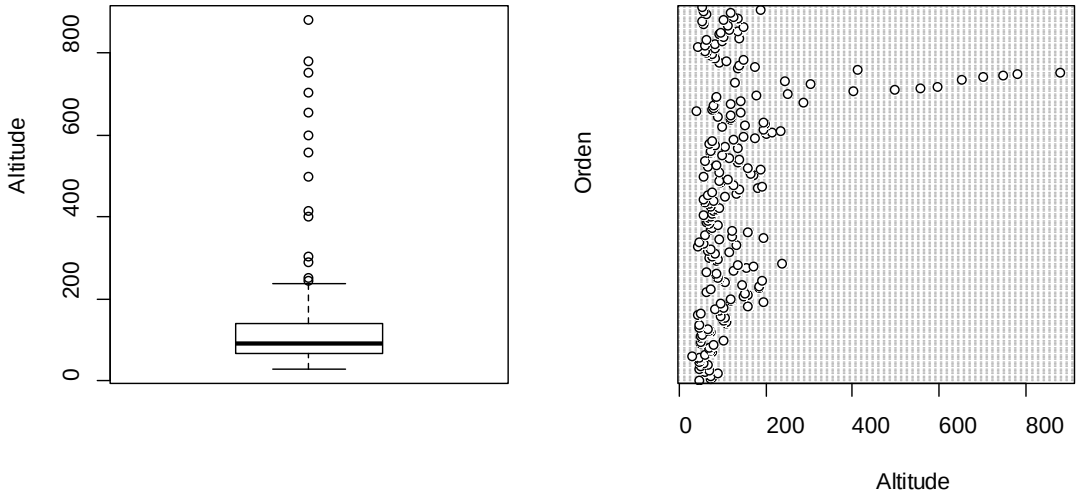
Sorprendentemente donde antes no había problemas, ahora se observa que para la medición realizada en marzo de 2005 hay tres observaciones sospechosas de ser outliers. Por ahora vamos a obviar esto ya que puede ser que dichas observaciones se deban a otra variable subyacente y estos outliers podrían tenerse en cuenta mediante la inclusión de más variables en el modelo (por ejemplo mediante una interacción entre el mes y otra variable). La estrategia a seguir sería identificar cuáles son estas observaciones (función "*identify*") y comprobar los residuos del modelo que ajustemos para estas observaciones.

Mediante el paquete "*lattice*" podemos crear gráficos multipaneles. Por ejemplo vamos a crear un gráfico que sea un boxplot condicional para cada fecha de medición y sexo.

```
> library(lattice)
> bwplot(CTL ~Sex|Month, data=Crayfish, ylab="Longitud cefalotorax")
```



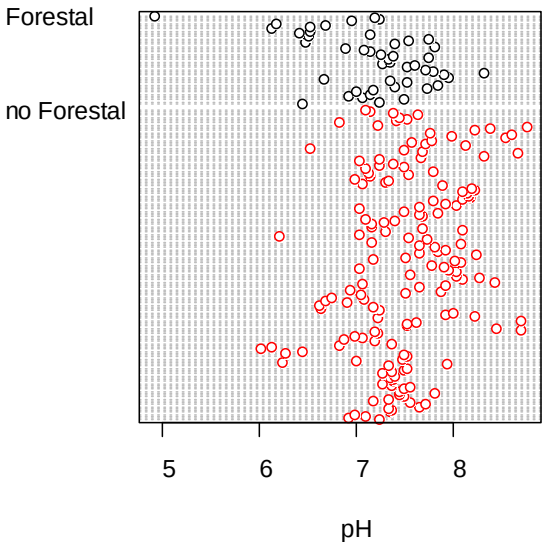
Otro tipo de gráfico usado en la detención de outliers es el Cleveland dotplot. En este tipo de gráfico se suele representar el número de la observación en el eje de la *y,* y el valor de la observación en el eje de la x. Mediante este gráfico podemos comprobar si los potenciales outliers están agrupados, es decir son observaciones "próximas" (en el espacio, tiempo, mismo individuo…) o si no muestran ningún patrón dentro del grupo de observaciones que nosotros tenemos. Veamos un ejemplo con los datos de IrishPh:



Como podemos observar ambos gráficos nos identifican una serie de observaciones como posibles outliers y además en el dotplot podemos observar que todas estas observaciones están agrupadas. Al igual que para el boxplot podemos elaborar un dotplot condicional en función de

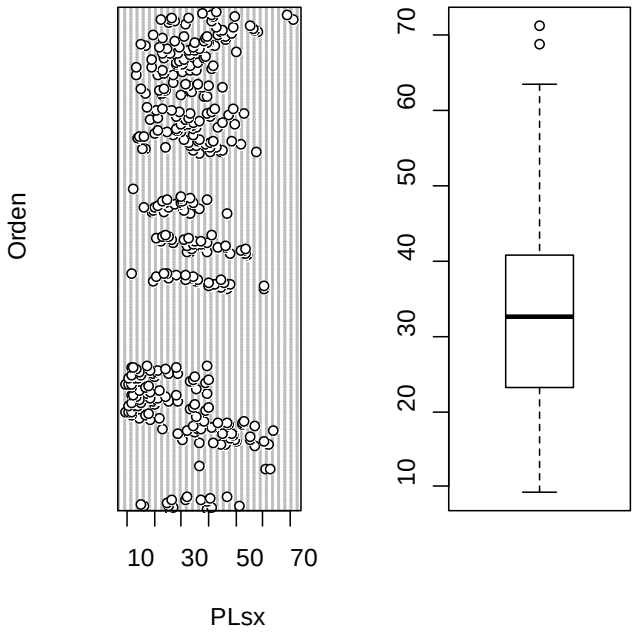
una variable categórica. Por ejemplo hagamos dicho gráfico para los datos de pH en función de si el área era forestal o no (variable “Forested”). Como hemos visto anteriormente en la estructura de los datos la variable “Forested” toma valores de 1 y 2 y su clase es “integer”, para poderla usar en un gráfico condicional debemos transformarla a variable categórica (factor) y cambiaremos el 1 por “Forestal” y el 2 por “no Forestal”. Como vemos en el gráfico la variación en pH es similar entre ambas clases de áreas y no se ven claros outliers.

```
> IrishPh$Forested <- factor(IrishPh$Forested, levels=c(1,2),
+                            labels=c("Forestal","no Forestal"))
> dev.off()
null device
          1
> dotchart(IrishPh$pH,groups=IrishPh$Forested, xlab="pH",
+          col=IrishPh$Forested)
```



La recomendación aquí es utilizar ambos gráficos para identificar potenciales outliers, pero teniendo muy en cuenta que aunque los boxplots identifiquen un punto como outlier, no significa que lo sea. Por ejemplo hagamos un boxplot y un dotplot para la variable “PLsx” (longitud del propodus) de la base de datos “Procambarus”. En este caso podemos observar como los dos puntos identificados como outliers por el boxplot no quedan excesivamente separados del resto de observaciones cuando estas son representadas como una nube de punto.

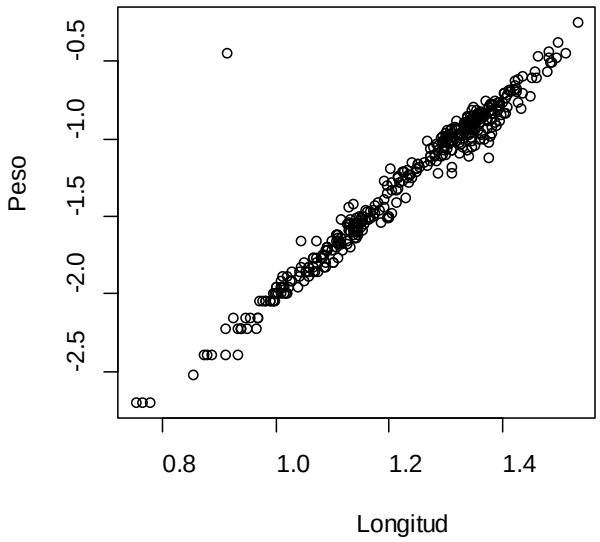
```
> par(mfrow=c(1,2), mar=(c(5,4,1,1)))
> dotchart(Crayfish$PLsx, xlab="PLsx", ylab="Orden")
> boxplot(Crayfish$PLsx)
```



2.2. Identificación de outliers potenciales en dos dimensiones

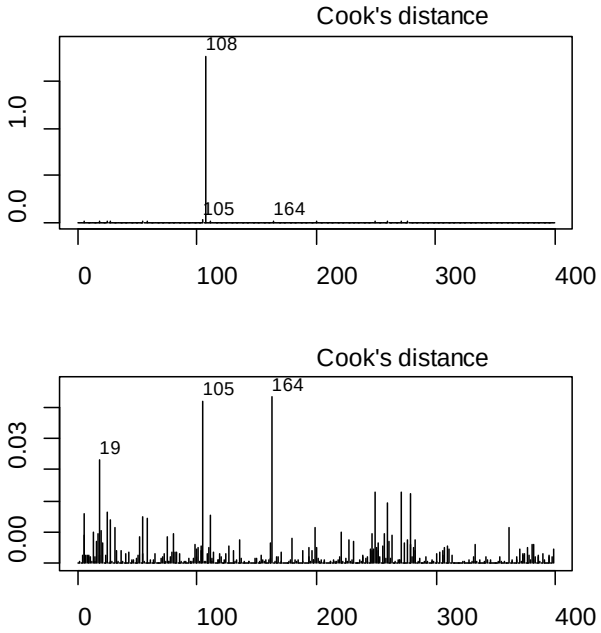
Cuando estamos estudiando la relación entre dos variables, es posible que no identifiquemos ningún outlier potencial cuando analizamos dichas variables de forma individual pero que al representar una frente a la otra veamos que algunas observaciones no cumplen con la relación predominante entre ambas variables, en este caso estamos hablando de outliers en dos dimensiones (x-y). Vamos a ver cómo identificar estos outliers potenciales mediante un ejemplo. En este caso usaremos la base de datos “wedgeclamII” (Ieno y Zuur 2015) concretamente la relación entre el peso seco libre de ceniza (AFD) y la longitud del berberecho *Donax hanleyanus*. Observaremos en el gráfico que existe una observación (ID = 108) que se desmarca claramente del resto de observaciones.

```
> Wclam <- read.table(file="wedgeclamII.txt", header=T)
> str(Wclam)
'data.frame':  399 obs. of  4 variables:
 $ ID    : num  1 2 3 4 5 6 7 8 9 10 ...
 $ MONTH : num  11 11 11 11 11 11 11 11 11 11 ...
 $ LENGTH: num  28.4 16.6 13.7 17.4 11.8 ...
 $ AFD   : num  0.248 0.052 0.028 0.07 0.022 0.187 0.361 0.05 0.087 0.
128 ...
> plot(log10(Wclam$LENGTH),log10(Wclam$AFD), xlab="Longitud", ylab="Pe
so")
```



En este caso solo tenemos dos variables y es sencillo identificar mediante un gráfico bidimensional aquellos puntos susceptibles de ser muy influyentes en nuestro análisis. Sin embargo este hecho se vuelve más complicado cuando tenemos más de una variable independiente o covariable. En estos casos podemos ayudarnos del parámetro denominado distancia de Cook. Básicamente esta distancia se calcula ajustando el modelo con todas las observaciones y luego dejando fuera una de ellas y comparando ambos ajustes. Algunos autores establecen un valor de distancia de Cook igual a 1 como un umbral a partir del cual debemos preocuparnos ya que nuestro análisis estará fuertemente influenciado por dicha observación. Esta distancia se calcula para el diagnóstico de un modelo y por tanto la identificación de una observación influyente no significa que debamos obviarla inmediatamente. Para ver estos gráficos hay que ajustar un modelo lineal de regresión, lo que veremos más adelante aunque adelantaremos el código, posteriormente mostraremos el gráfico resultante considerando la observación 108 (gráfico superior) y sin considerarla (gráfico inferior). Como podemos ver la observación 108 presenta valores de distancia de Cook superiores a 1 por lo que está influenciando bastante nuestros datos. Una vez que no lo consideramos en el análisis todas las observaciones presentan valores de la distancia de Cook bastante pequeños.

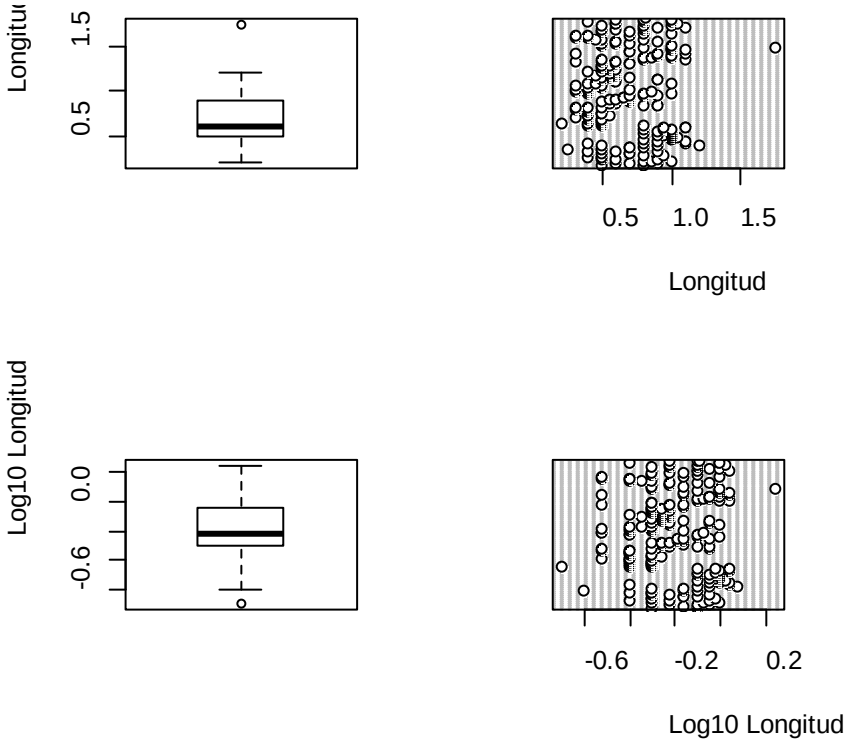
```
> M1 <- lm(log10(AFD)~log10(LENGTH), data=Wclam)
> M2 <- lm(log10(AFD)~log10(LENGTH), data=Wclam, subset=-108)
> par(mfrow=c(2,1), mar=c(3,3,2,2))
> plot(M1,which=4)
> plot(M2,which=4)
```



2.3. ¿Qué hacer una vez identificados posibles outliers?

Una vez que hemos identificados potenciales outliers en nuestros datos tenemos una serie de opciones: eliminarlos, hacer como que no los hemos visto (puede acabar en desastre), mostrar los resultados estadísticos con y sin estos outliers, o bien aplicar alguna transformación a los datos. En el mejor de los casos estaremos seguro de que nuestro outlier es fruto de un error de muestreo, medición, análisis o de toma de datos. Por ejemplo si estamos trabajando con una base de datos de altura de árboles y tenemos una observación realizada en encina que nos dice que dicho árbol mide 300 m, estaremos sin duda ante un error. Una buena solución es comprobar si este dato está dentro de los límites de dicha variable según la bibliografía. Otra opción es transformar los datos, usando generalmente la raíz cuadrada o el logaritmo, mediante estas transformaciones comprimiremos las observaciones que están alejadas de la mayoría del grupo. Veamos un ejemplo con la base de datos “pellets” (Ieno y Zuur 2015) que recoge la longitud y peso de cagarrutas de un gusano marino.

```
> pellets <- read.table(file="pellets.txt", header=T)
> str(pellets)
'data.frame':  215 obs. of  3 variables:
 $ Length: num  0.8 0.5 0.8 0.7 0.6 0.9 0.8 1 0.85 0.5 ...
 $ Width : num  0.4 0.3 0.4 0.4 0.3 0.5 0.4 0.5 0.4 0.25 ...
 $ month : int  8 8 8 8 8 8 8 8 8 ...
> par(mfrow=c(2,2))
> boxplot(pellets$Length, ylab="Longitud")
> dotchart(pellets$Length, xlab="Longitud")
> boxplot(log10(pellets$Length), ylab="Log10 Longitud")
> dotchart(log10(pellets$Length), xlab="Log10 Longitud")
```



Sin embargo la transformación de variables tiene una serie de pros y contras que hemos de tener en consideración antes de tomar la decisión de aplicarlas o no. Este tipo de transformación es útil cuando queremos deshacernos de un outlier (como hemos visto), cuando queremos linealizar una relación entre dos variables (p. ej. pasar de una relación exponencial a una lineal) o bien cuando muestreamos en gradientes ambientales y no podemos llenar todos los huecos existentes en los valores de nuestra variable ambiental. Sin embargo, la transformación de nuestras variables tiene una serie de inconvenientes. Por ejemplo, si después de una transformación modelizamos una variable respuesta en función de una o más covariables tendremos que volver a transformar de vuelta los parámetros estimados por el modelo. También puede ser problemático a la hora de interpretar ciertas variables que ya están de por sí transformadas mediante logaritmos o que incluyen el logaritmo en su fórmula original (p. ej. pH, o el índice de diversidad de Shannon). Un problema mayor es cuando existen interacciones entre variables ya que la transformación de una de ellas puede tener influencia en la significación de la interacción con la otra variable. Además cuando las transformaciones son en nuestra variable respuesta es mejor seleccionar la familia correcta de distribución, esto lo veremos más adelante en los modelos lineales generalizados.

3. Normalidad

Como hemos comentado anteriormente la distribución normal presenta una serie de características deseables para la aplicación de muchas técnicas estadística, lo cual no significa que no se puedan realizar análisis cuando no se cumpla la norma de normalidad. Sin embargo existe una confusión muy generalizada sobre qué significa el supuesto de normalidad. Una cosa es que nuestra variable respuesta siga una distribución normal y otra cosa bien diferente es el

supuesto de normalidad en un análisis estadístico. Por ejemplo en una regresión lineal entre una variable respuesta continua y una covariable, el supuesto de normalidad significa que nuestra variable respuesta se distribuye de forma normal a *cada* valor de la covariable. Dicho de otro modo, si nosotros hiciéramos un muestreo aleatorio de nuestra variable respuesta en cada valor de nuestra covariable, las observaciones derivadas de dicho muestreo se distribuirán de forma normal, coincidiendo el valor de la línea de regresión ajustada con el centro de dicha distribución. Mejor si lo vemos gráficamente:

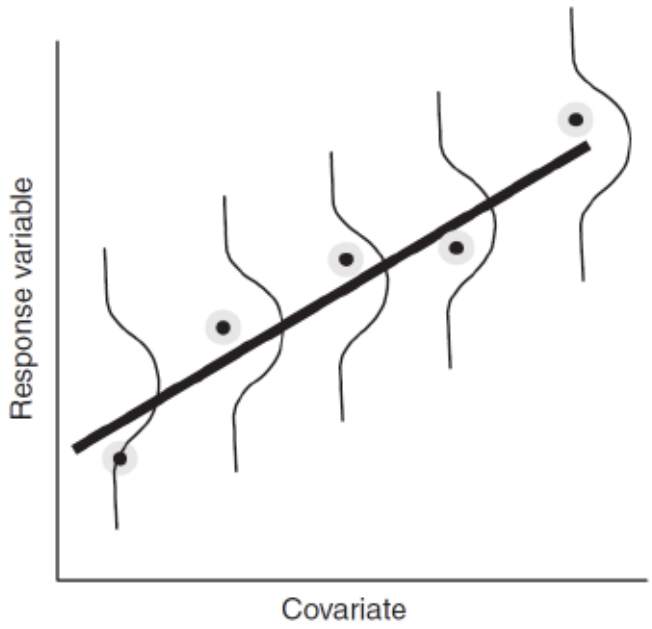


Figura 2: Representación gráfica del supuesto de normalidad en un modelo de regresión lineal simple (modificado de Ruiz-Benito (2016), Universidad de Alcalá)

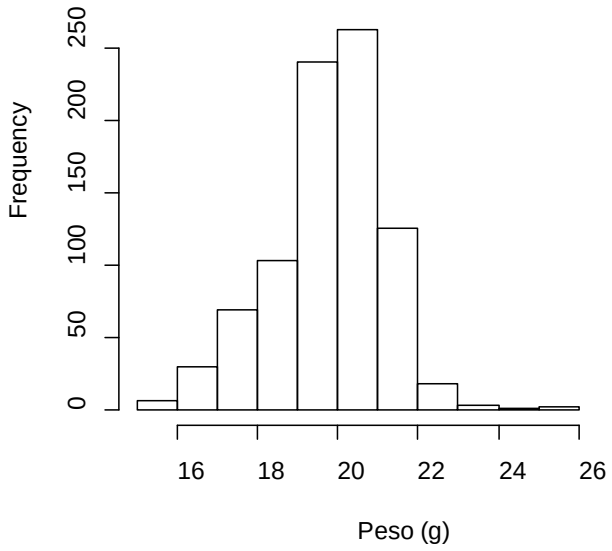
La normalidad en un modelo de regresión lineal no es tan importante como por ejemplo el supuesto de homogeneidad que veremos más adelante. Es decir tenemos que ser razonablemente rigurosos a la hora de comprobar el supuesto de normalidad, lo que no significa que debamos obviarlo. En vez de comprobar la normalidad de nuestra variable respuesta y realizar una transformación, deberíamos realizar una adecuada validación del modelo de regresión y comprobar que los residuos de dicho modelo sí se distribuyen de forma normal. Vamos a analizar a continuación una serie de técnicas para comprobar la normalidad, en este caso de nuestra variable respuesta. En el capítulo dedicado a los modelos lineales veremos cómo comprobar que los residuos de nuestro modelo siguen una distribución normal.

3.1. Histogramas

El histograma es la herramienta gráfica clásica para comprobar la normalidad de una determinada variable. El histograma se crea a partir de una tabla de frecuencia y el objetivo es mostrar el centro y la distribución de nuestros datos. Veamos un ejemplo basado en el peso corporal de individuos de una especie de gorrión.

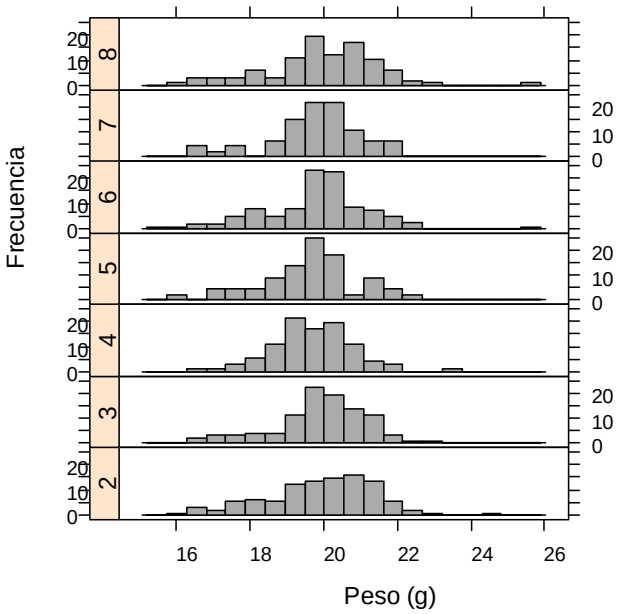
```
> setwd("C:/Users/Enrique/Desktop/Data analysis/CursoR")
> Sparrows <- read.table(file="Sparrows.txt", header=T)
> str(Sparrows)
'data.frame':  979 obs. of  10 variables:
```

```
 $ Species : Factor w/ 2 levels "SESP","SSTS": 2 2 2 2 2 2 2 2 2 2 ...
 $ Sex     : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 1 2 2 1 2 .
..
 $ Wingcrd : num  58 56.5 59 59 57 57 57 57 53.5 56.5 ...
 $ Tarsus  : num  21.7 21.1 21 21.3 21 20.7 22 20.8 20.1 22.2 ...
 $ Head    : num  32.7 31.4 33.3 32.5 32.5 32.5 33.1 32.1 32.1 32.2 ..
.
 $ Culmen  : num  13.9 12.2 13.8 13.2 13.8 13.3 13.8 13.5 13.7 13.6 ..
.
 $ Nalospi : num  10.2 10.1 10 9.9 9.9 9.9 9.9 9.8 9.8 9.8 ...
 $ Wt      : num  20.3 17.4 21 21 19.8 17.5 19.6 21.2 18.5 20.5 ...
 $ Observer: int  2 2 2 2 2 2 2 2 2 2 ...
 $ Age     : int  0 0 0 0 0 0 0 0 0 0 ...
> Sparrows2 <- Sparrows[Sparrows$Species=="SSTS",]
> hist(Sparrows2$Wt, xlab="Peso (g)", main="")
```



Al igual que anteriormente podemos elaborar histogramas condicionales. En este caso observamos que existe una serie de covariables como el sexo, la edad o el observador. Utilizando el paquete *lattice* vamos a crear un panel con un histograma para cada observador. En este ejemplo vamos a introducir un poco más de código en el texto para elegir cómo se distribuirán los paneles (*layout*), el color (*col*), dónde aparecerán las etiquetes de los niveles del factor (*strip* y *strip.left*) o el número de barras del histograma (*nint*).

```
> histogram(~Wt|factor(Observer), data=Sparrows2,
+           layout=c(1,7), col="darkgrey",
+           strip=FALSE, strip.left=TRUE,
+           nint=20,
+           xlab="Peso (g)", ylab="Frecuencia")
```

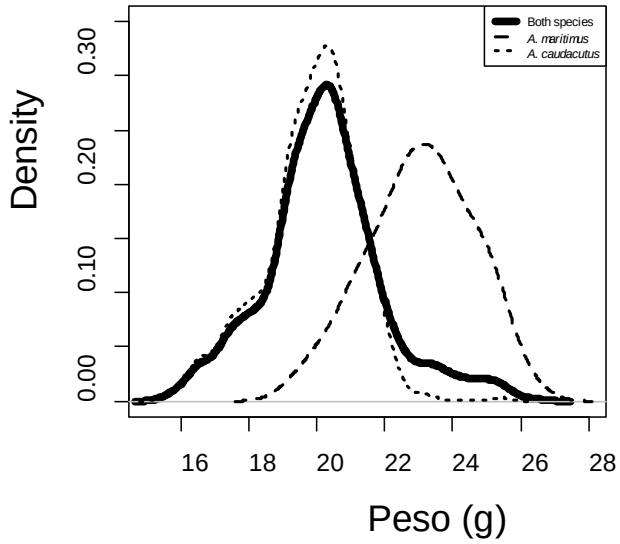


¿Debemos preocuparnos si observamos que nuestra variable respuesta no sigue una distribución normal según los histogramas? No, lo que debemos hacer es continuar con nuestro análisis, elaborar un modelo que tenga en cuenta nuestras covariables de interés, extraer los residuos de dicho modelo y comprobar que esos residuos se distribuyen normalmente. En caso contrario los resultados de nuestro análisis se invalidarían y habría que tomar cartas en el asunto, por ejemplo elegir otro tipo de modelo. Un caso contrario sería la aplicación de un test de contraste de hipótesis como el *t-test*, en este caso solo tenemos nuestra variable respuesta dividida en dos grupos que compararemos entre sí. En este caso nuestra variable respuesta si tiene que estar normalmente distribuida dentro de cada grupo.

3.2. Gráficos de densidad de Kernel

La forma de los histogramas está fuertemente influenciada por el número de intervalos que consideremos. Por ello es también recomendable utilizar las curvas de densidad de Kernel que nos muestran gráficamente una curva que se ajusta a la función de densidad de probabilidad de nuestra variable. Es decir, si integramos el área debajo de dicha curva el resultado es 1 que coincide con la probabilidad de que una observación de nuestra variable esté dentro de dicho rango. Vamos a realizar un gráfico de densidad de Kernel, utilizando los datos de los gorriones pero esta vez tendremos en cuenta las dos especies a la vez y luego por separado. Vamos a representar las tres curvas en el mismo gráfico:

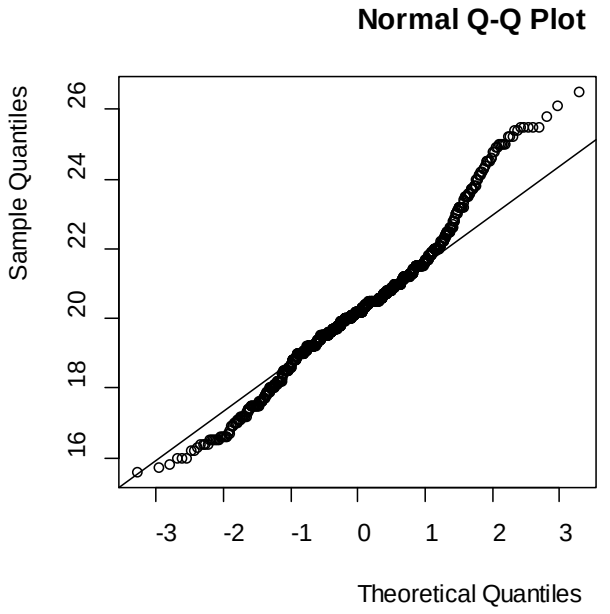
```
> d1 <- density(Sparrows$Wt[Sparrows$Species=="A. maritimus"])
> d2 <- density(Sparrows$Wt[Sparrows$Species=="A. caudacutus"])
> lines(d1, lty=2, lwd=2)
> lines(d2, lty=3, lwd=2)
> legend("topright",legend=expression("Both species"
+                                     ,italic("A. maritimus"),
+                                     italic("A. caudacutus")),
+                                     lty=c(1,2,3),
+                                     lwd=c(5,2,2), cex=0.5)
```



3.3. Q-Q plots

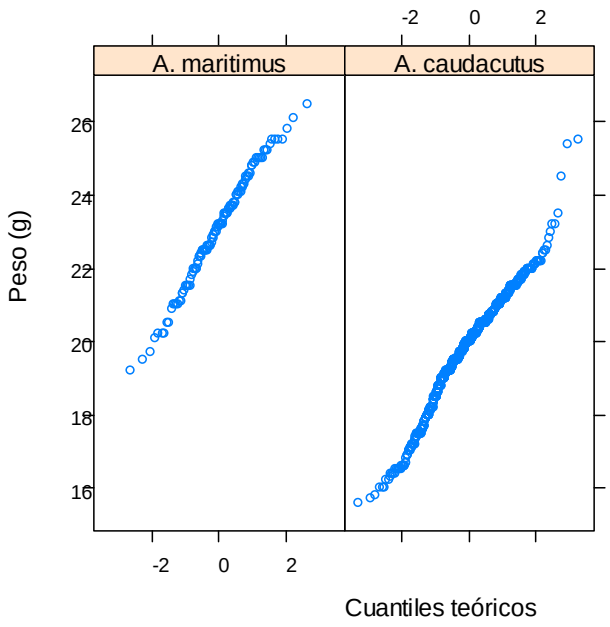
Los gráficos cuantil-cuantil (Q-Q) representan y enfrenta los cuantiles de dos distribuciones. Cuando los puntos se distribuyen a lo largo de la línea del gráfico significa que ambas distribuciones son iguales. Aprovechando este concepto podemos representar los cuantiles de la distribución de nuestra variable respuesta (o de los residuos del modelo) frente a los cuantiles de una distribución normal. De modo que si estos datos quedan distribuido “más o menos” a lo largo de la recta y=x del gráfico podemos concluir que nuestra variable o nuestros residuos se distribuyen de forma normal. Si aplicamos un gráfico Q-Q al peso de los gorriones veremos cómo nuestra variable se desvía de la normalidad en la parte derecha de la distribución.

```
> qqnorm(Sparrows$Wt)
> qqline(Sparrows$Wt)
```



Al igual que con el resto de los gráficos, el paquete *lattice* nos permite producir múltiples Q-Q plots. En este caso hagamos uno por especie.

```
> qqmath(~Wt|Species, data=Sparrows,
+         xlab="Cuantiles teóricos", ylab="Peso (g)")
```



3.4. Test de normalidad

Existen algunos test paramétricos que permiten testar si una determinada distribución se aleja significativamente de una distribución normal. Entre los test más usados se encuentran el test de Shapiro-Wilk "*shapiro.test(variable)*" y el de Kolmogorov-Smirnov *"ks.test(variable, "pnorm")"*. Ambas pruebas producen un valor de probabilidad (p valor) que en caso de ser menor al umbral 0.05 nos indicaría que se rechaza la hipótesis nula de que nuestra distribución es igual a una distribución normal. Muchos investigadores prefieren este tipo de pruebas porque se basan en un umbral a partir del cual pueden decidir si una distribución se asemeja a una normal o no. Sin embargo, no se recomienda tomar una decisión sobre la normalidad de una variable o de unos residuos basándonos solo y exclusivamente en los resultados de este tipo de pruebas. Por ejemplo, cuando tenemos un número de muestras pequeño, es difícil que el histograma de nuestra variable se asemeje al de una distribución normal. Por el contrario cuanto más observaciones tengamos de una variable mayor probabilidad de que muchas de ellas se desvíen de la supuesta normalidad.

Como hemos comentado anteriormente desviaciones de la normalidad no suponen un problema en modelos lineales como la regresión. Por otro lado es importante que dicha normalidad se cumpla en los residuos del modelo ajustado, aunque también se permite cierta variación con respecto a la distribución normal. Por último, no es necesario que nuestras covariables se distribuyan normalmente.

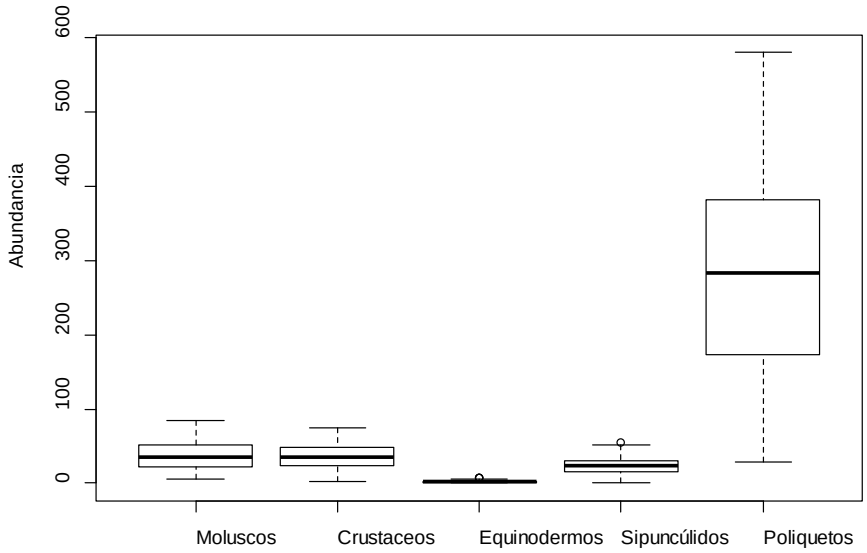
4. Homogeneidad de varianza

A la homogeneidad de varianzas también se le llama homocedasticidad y supone una suposición importante en técnicas estadísticas como los modelos lineales o las pruebas t. La homogeneidad de varianzas sucede cuando la dispersión de los valores de la variable dependiente es similar para cada valor de la variable independiente. Cuando dicha homogeneidad no se cumple, lo que es común en datos ecológicos, podemos invalidar la significación estadística de los parámetros que estimemos ya que incrementamos el denominado error de tipo I, es decir aumentamos la probabilidad de rechazar la hipótesis nula lo que nos puede hacer caer en falsos positivos. Al igual que para la normalidad existen distintas herramientas gráficas y también algunos test para evaluar dicha homocedasticidad.

4.1. Boxplot condicionales

Anteriormente hemos vista la utilidad de los boxplots para visualizar la variabilidad de una determinada variable. Los boxplots condicionados son muy útiles para evaluar la homocedasticidad de una variable dependiente continua para los distintos niveles de una variable independiente categórica. Vamos a visualizar un ejemplo donde se viola la homogeneidad de varianzas usando una base de datos sobre la abundancia de distintos grupos taxonómicos de macroinvertebrados marinos (Macrobenthos.txt (Ieno y Zuur 2009)). Como vemos el grupo de los poliquetos es el más abundante y a su vez presenta una variación muchísimo mayor que el resto de grupos taxonómicos. Esto es un claro ejemplo de heterogeneidad de varianzas, por lo que tendremos que ser especialmente cuidadosos a la hora de aplicar un modelo lineal. Al igual que la normalidad debemos procurar que la homogeneidad de varianzas se cumpla en los residuos del modelo.

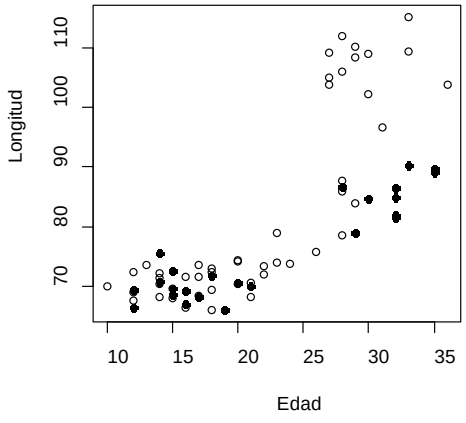
```
> Benthos <- read.table(file="Macrobenthos.txt", header=T)
> str(Benthos)
'data.frame':  400 obs. of  12 variables:
 $ time       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ period     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ CT         : int  0 0 0 0 0 1 1 1 1 1 ...
 $ effort     : num  0 0 0 0 0 1.71 1.71 1.71 1.71 1.71 ...
 $ o_matter   : num  1.16 1.38 1.84 1.62 1.62 2.13 1.78 1.87 1.56 1.15
...
 $ mud        : num  35.3 34.5 38.4 30.7 32.6 ...
 $ silt       : num  25.1 21.9 23.5 17.7 23.3 ...
 $ clay       : num  10.16 12.52 14.94 12.99 9.27 ...
 $ turbidity  : num  5.71 9.55 8.01 8.01 8.78 9.55 9.55 7.25 8.01 8.01
...
 $ temperature: num  10.5 10.6 10.6 10.6 10.6 ...
 $ Abundance  : int  11 18 15 39 24 19 17 27 5 18 ...
 $ TaxonID    : int  1 1 1 1 1 1 1 1 1 1 ...
> Benthos$TaxonID <- factor(Benthos$TaxonID,
+                           levels=c(1:5),
+                           labels=c("Moluscos","Crustaceos",
+                                    "Equinodermos","Sipuncúlidos",
+                                    "Poliquetos"))
> boxplot(Abundance~TaxonID, data=Benthos, ylab="Abundancia")
```



4.2. Scatterplots para variables continuas

En la mayoría de los ejemplos vistos hasta ahora hemos comparado los valores o la dispersión de la variable dependiente para cada uno de los niveles de una variable categórica. Cuando nuestra variable o variables independientes son también cuantitativas y continuas la forma de visualizar dicha heterogeneidad es mediante un gráfico de dispersión en el que se enfrente la variable dependiente y alguna de las variables independientes. Vamos a ver un ejemplo en el que no solo comprobaremos la homogeneidad de varianza para variables continuas sino que haremos un gráfico condicional en función de una variable categórica. Usaremos para ello la base de datos de Fur_seal.txt (Ieno and Zuur 2015) y nos centraremos en las variables: longitud (VAR26), edad (IS) y sexo (Sex).

```
> Fseal <- read.table(file="Fur_Seal.txt", header=T)
> Fseal$Sex2 <- Fseal$Sex
> Fseal$Sex2[Fseal$Sex==2] <- 16
> plot(Fseal$IS, Fseal$VAR26,
+      pch=Fseal$Sex2,ylab="Longitud", xlab="Edad")
```



Como vemos, la dispersión de los valores de la longitud aumenta con la edad y además dicha dispersión también cambia con el sexo. Este sería un ejemplo donde se viola la suposición de homogeneidad de varianza.

4.3. Test para evaluar la homogeneidad de varianza

Al igual que para la normalidad existen una serie de test paramétricos que permiten evaluar la homogeneidad de varianza. Sin embargo, la objetividad asociada a obtener un valor de probabilidad en estos test no quiere decir que su capacidad para discernir cuando hay o no homogeneidad de varianza sea superior a las comprobaciones gráficas. Los test más conocidos son el de Bertlett y el de Levene, en ambos test la hipótesis nula es que ambas varianzas son iguales por lo que un valor de *p* por debajo de 0.05 indicaría que las varianzas no son homogéneas. De nuevo vamos a poner un ejemplo en el que dicho test se aplica a una variable dependiente pero la recomendación es hacerlo sobre los residuales del modelo.

```
 > library("lawstat")
> bartlett.test(Abundance~TaxonID, data=Benthos)

        Bartlett test of homogeneity of variances

data:  Abundance by TaxonID
Bartlett's K-squared = 1064.5, df = 4, p-value < 2.2e-16

> levene.test(Benthos$Abundance, Benthos$TaxonID)

        modified robust Brown-Forsythe Levene-type test based on the a
bsolute deviations from the median

data:  Benthos$Abundance
Test Statistic = 146.73, p-value < 2.2e-16
```

Ambos test indican que no existe homogeneidad de varianza. Cabe recordar de nuevo que la homogeneidad de varianza debe comprobarse sobre los residuos de un modelo estadístico. Cuando tenemos distintas variables independientes, ya sean continuas, categóricas o una mezcla de ellas, la manera recomendada de comprobar si se cumple el supuesto de homogeneidad es realizar un gráfico de los residuos del modelo frente a cada una de las variables independientes por separado.

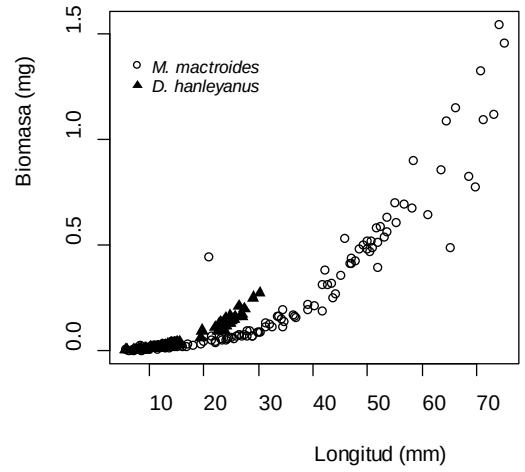
5. Explorando relaciones entre variables

Siempre que nos enfrentemos a una base de datos, especialmente si es de gran tamaño, es importante que conozcamos qué variables están relacionadas y cómo lo están (linealmente, exponencialmente, etc). El tipo de patrón que observemos en nuestro análisis exploratorio determinará, en parte, que modelo matemático elegimos para nuestro análisis.

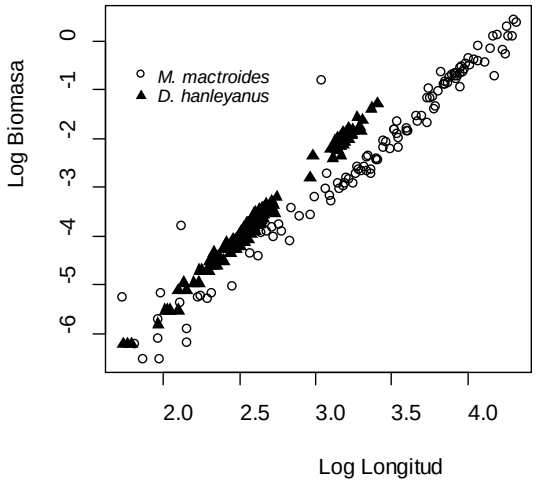
5.1. Scatterplots

Para analizar las relaciones entre variables, los scatterplots o gráficos de dispersión son de gran ayuda. Trabajaremos con la base de datos “Mussels.txt” (Ieno y Zuur 2015) con la que realizaremos un gráfico que nos relacione la biomasa de los individuos con su longitud y que nos diferencie entre las dos especies que se analizan.

```
> Mussels <- read.table(file="Mussels.txt", header=T)
> str(Mussels)
'data.frame':  154 obs. of  4 variables:
 $ LengthM : num  13.76 9.4 8.2 9.26 8.6 ...
 $ BiomassM: num  0.0124 0.0054 0.0047 0.0053 0.0028 0.0052 0.0015 0.0
021 0.0023 0.0034 ...
 $ LengthD : num  30.2 23.1 26.9 19.4 23.8 ...
 $ BiomassD: num  0.274 0.11 0.171 0.06 0.094 0.117 0.127 0.198 0.25 0
.209 ...
> Biomass <- c(Mussels$BiomassM,Mussels$BiomassD)
> Length <- c(Mussels$LengthM,Mussels$LengthD)
> Species <- rep(c(1,17),each=154)
> plot(x=Length, y=Biomass,
+      xlab="Longitud (mm)", ylab="Biomasa (mg)",
+      pch=Species)
> legend(x=5,y=1.45,
+        legend=expression(italic("M. mactroides"),
+                          italic("D. hanleyanus")),
+        pch=c(1,17), cex=0.8, bty="n")
```

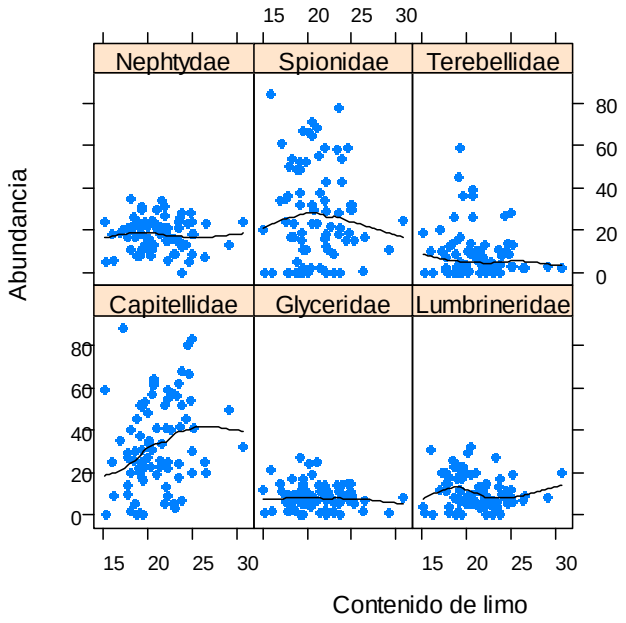


Como vemos, la relación entre la biomasa y la longitud es exponencial y los parámetros de dicha exponencial serían distintos para cada especie. Alternativamente al uso de un modelo no lineal podemos intentar transformar ambas variables para que la relación entre ambas sea lineal, aunque corremos el riesgo de que el efecto que vemos de la especie desaparezca, aunque parece no ser el caso.



Una alternativa a realizar simples scatterplots es realizar múltiples scatterplots en los que podamos observar como nuestra variable dependiente se relaciona con múltiples covariables. Vamos a realizar un ejemplo con la función “xyplot” del paquete “lattice”. Para ello utilizaremos los datos “Zoobenthos.txt” (Ieno y Zuur 2015) donde se analiza la abundancia de distintas familias de poliquetos en relación al contenido de limo en los sedimentos. Vamos a trabajar con 6 de estas familias. Como veremos la relación entre la abundancia de estas familias y el contenido en limo de los sedimentos no es muy clara.

```
> Poly <- read.table(file="Zoobenthos.txt", header=T)
> MyVar <- c("Glyceridae", "Spionidae",
+            "Nephtydae", "Lumbrineridae",
+            "Terebellidae", "Capitellidae")
> SpAbund <- as.vector(as.matrix(Poly[,MyVar]))
> ID <- rep(MyVar, each=80)
> Silt <- rep(Poly$silt,6)
> library(lattice)
> xyplot(SpAbund~Silt|factor(ID),
+        xlab="Contenido de limo",
+        ylab="Abundancia",
+        panel=function(x,y){
+          panel.points(x,y, pch=16)
+          panel.loess(x, y, col=1)})
```

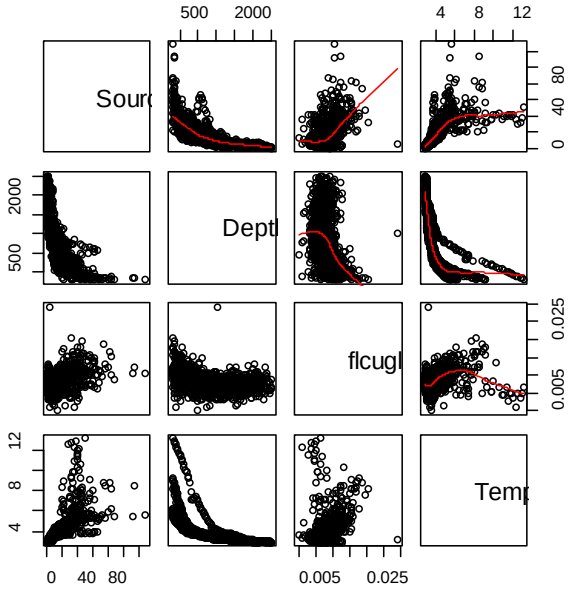


5.2. Pairplots

Los pairplots o gráficos pareados nos permiten explorar relaciones múltiples entre variables, por lo que son muy útiles cuando queremos ver como se relacionan entre sí una serie de variables continuas. Vamos a ver en un ejemplo como se relacionan entre sí las variables bioluminiscencia (Source), profundidad (Depth), fluorescencia (flcugl) y temperatura (Temp) pertenecientes a la base de datos “HegerPriede.txt” (Ieno y Zuur 2015). Cada panel resultante muestra la relación entre dos de las variables. Como vemos no solo se muestra la relación entre la variable

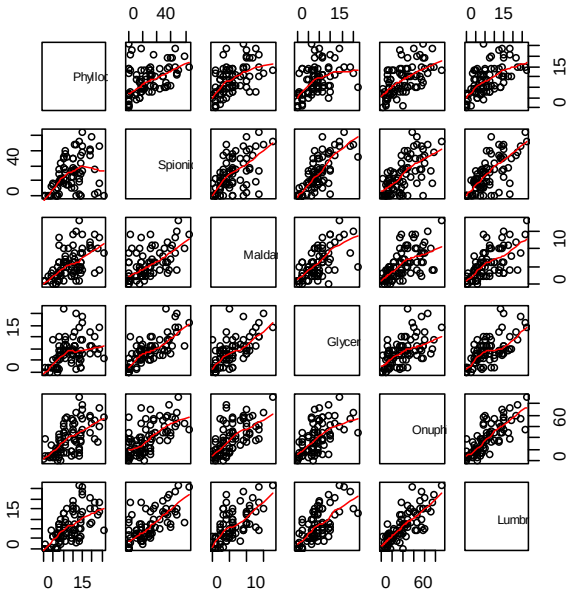
dependiente (Source) y el resto de covariables sino también la relación entre las variables independientes.

```
> BioLum <- read.table (file="HegerPriede.txt", header=T)
> MyVar <- c("Sources","Depth","flcugl","Temp")
> pairs(BioLum[,MyVar],upper.panel=panel.smooth)
```



Este tipo de gráfico no solo se puede aplicar para ver la relación entre nuestra variable dependiente y nuestras variables independientes, sino que también podemos utilizarlo para ver posibles relaciones entre variables independientes o bien para explorar si múltiples variables respuesta se encuentran correlacionadas entre sí. Veamos un ejemplo de este último caso para la abundancia de algunas familias de poliquetos.

```
> Poly <- read.table(file="Zoobenthos.txt", header=T)
> MyVar <- c("Phyllodocidae", "Spionidae",
+            "Maldanidae", "Glyceridae",
+            "Onuphidae", "Lumbrineridae")
> pairs(Poly[,MyVar],panel=panel.smooth)
```



6. Colinealidad

La colinealidad sucede cuando existe correlación elevada entre dos variables independientes o explicativas, cuando esto ocurre entre más de dos covariables se denomina multicolinealidad. Uno de los malentendidos más comunes en el análisis de datos es incluir en el modelo todas y cada una de las variables que se han medido, sin comprobar previamente las posibles relaciones entre estas variables. Si la colinealidad no es tratada adecuadamente podrían darse confusiones en la interpretación de los resultados derivados del análisis de los datos. Por ejemplo, la combinación de una serie de covariables puede dar lugar a una alta significación del efecto de las mismas en el modelo, pero una simple modificación de las mismas puede provocar que ninguna de ellas resulte significativa.

6.1. El coeficiente de correlación

Cuando dos variables muestran la tendencia a aumentar o disminuir a la vez, se dice de ellas que están correlacionadas. Los valores de los distintos coeficientes de correlación están entre -1 y 1, siendo 0 cuando dos variables no están correlacionadas y 1 ó -1 cuando están completamente correlacionadas. El signo de la correlación nos indica si la misma es positiva o negativa. Existen distintos coeficientes de correlación, como Spearman o Kendall, aunque el más usado de todos es el coeficiente de correlación de Pearson (*r*) que se define como:

$$r = \frac{1}{N-1} \times \sum_{i=1}^{N} \left(\frac{x_i - \bar{x}}{s_x}\right) \times \left(\frac{y_i - \bar{y}}{s_y}\right)$$

Donde N es la muestra total, x_i y y_i son los valores de la variable, $\bar{x}$ y $\bar{y}$ la media de dichas variables, y S_x y S_y sus desviación estándar. En R podemos calcular el coeficiente de correlación de Pearson para distintas variables de forma muy sencilla. Vamos a utilizar una base de datos sobre comunidades de arañas donde se evalúan una serie de variables ambientales y de estructura del hábitat de estas especies en plantaciones forestales de conífera y especies de hoja ancha (Ieno y Zuur 2015). Las variables ambientales son: GVC (cobertura vegetal), HLC (cobertura herbácea), SLC (cobertura matorral), NLC (cobertura acículas), LC (cobertura hojarasca), FWDC (cobertura maderas finas), CWDC (cobertura maderas gruesas), LLC (cobertura hojas), MLD (profundidad hojarasca), OM (materia orgánica suelo).

```
> Spiders <- read.table (file="Spiders.txt", header=T)
> str(Spiders)
'data.frame':  173 obs. of  14 variables:
 $ S_plot: int  1 1 1 1 1 1 1 2 2 2 ...
 $ GVC   : num  0.01 0.01 0.01 0.01 0.03 0.27 0.15 0.15 0.59 0.35 ...
 $ HLC   : num  0.27 0.45 0.63 0.49 0.32 0.73 0.68 0.2 0.39 0.38 ...
 $ SLC   : num  0.64 0.39 0.17 0.36 0.63 0.08 0.32 0.38 0.03 0.38 ...
 $ NLC   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ LC    : num  0 0 0 0 0 0 0 0.63 0.06 0.49 ...
 $ FWDC  : num  0 0 0 0 0 0 0 0.06 0.2 0.1 ...
 $ CWDC  : num  0 0 0 0 0 0 0 0.21 0.18 0.05 ...
 $ LLC   : num  0 0 0 0 0 0 0 0.63 0.06 0.49 ...
 $ MLD   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ OM    : num  0.18 0.24 0.27 0.17 0.38 0.9 0.91 0.34 0.17 0.23 ...
 $ X     : int  120123 120051 120006 119927 119855 119788 119742 14623
2 146188 146158 ...
```

```
 $ Y     : int  183465 183558 183512 183563 183559 183546 183501 10277
7 102818 102891 ...
 $ Hlog10: num  1.05 1.13 1 0.87 0.94 1.01 1.2 0.38 0.71 0.82 ...
> MyVar <- c("LC","LLC","HLC","NLC")
> cor(Spiders[,MyVar])
             LC         LLC         HLC        NLC
LC   1.00000000  1.00000000  0.02090388 -0.2909683
LLC  1.00000000  1.00000000  0.02090388 -0.2909683
HLC  0.02090388  0.02090388  1.00000000 -0.7018856
NLC -0.29096826 -0.29096826 -0.70188559  1.0000000
```

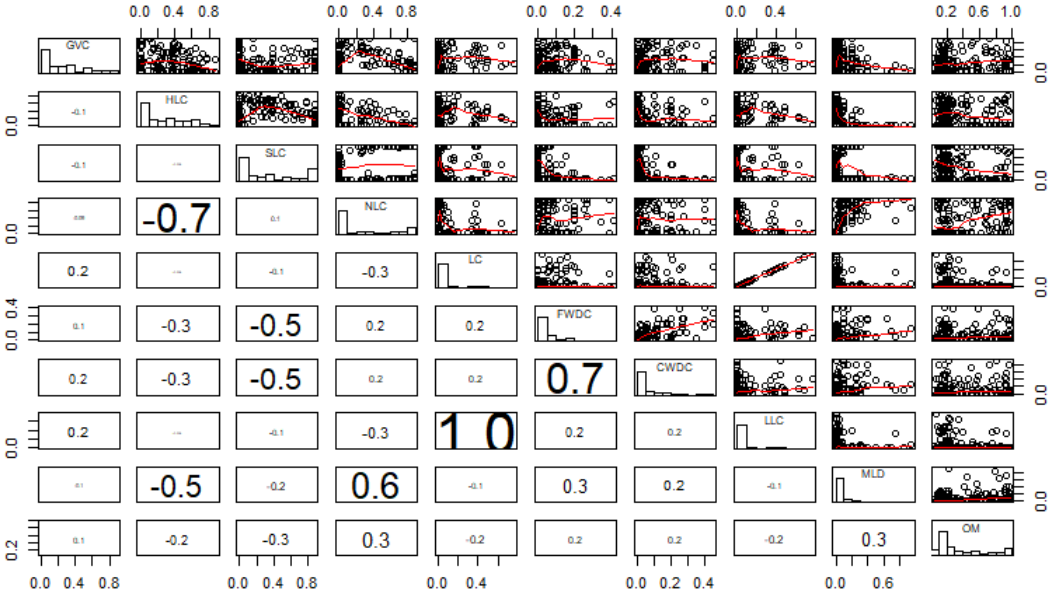
Lo que obtenemos es una matriz de correlación donde se muestra el coeficiente de correlación entre las variables tomadas de dos en dos. Como se puede observar la diagonal es igual a 1 ya que muestra la correlación de cada variable con sí misma. Como regla general un coeficiente de correlación superior a 0.8 (ó -0.8) es crítico y la inclusión en el modelo de esas dos variables podría suponer un problema desde el punto de vista de la validación de los resultados. Por otro lado valores de entre 0.50 y 0.70 (en valor absoluto) también requieren una consideración especial, sobre todo cuando la relación entre la variable respuesta y las covariables es débil. Es estos casos habría que ser más conservadores a la hora de introducir en el modelo variables que estuvieran correlacionadas.

Un aspecto importante a tener en cuenta son los outliers ya que estos pueden ser los causantes de una correlación entre variables incluso de afectar al signo de la correlación. Por todo ello es recomendable identificar primero los outliers y calcular la correlación entre variables una vez que se haya decidido qué hacer con dichas observaciones, es decir una vez se han eliminado o bien se ha decidido mantenerlas.

6.2. Evaluación gráfica de correlación entre variables

Como hemos visto anteriormente, los gráficos de dispersión son los adecuados para evaluar de forma gráfica las relaciones existentes entre variables. Vamos a ir un paso más allá y vamos a crear un gráfico de dispersión múltiple al que añadiremos los coeficientes de correlación. Para ello vamos a instalar primero una librería local que contiene una modificación de la función *pairs* para poder añadir el coeficiente de correlación a nuestro gráfico. Una de las cosas que debemos tener en cuenta es que el coeficiente de correlación de Pearson se utiliza para correlaciones lineales, sin embargo la existencia de relaciones no lineales entre variables también implica colinealidad. Por ello es recomendable no solo basarse en los coeficientes de correlación sino también echar un vistazo a los gráficos de dispersión entre variables. Echando un vistazo al ejemplo vemos como hay un 100% de colinealidad entre LLC y LC, el primer paso sería eliminar una de ellas. También vemos como otros pares de covariables como NLC-HCL y FWDC-CWDC presentan valores de correlación elevados. Decidir cuál de las covariables excluir del análisis depende del criterio del investigador respecto a cuál considera mejor indicador, en este caso, de las variables ambientales que pueden influir a las comunidades forestales de arañas. En cualquier caso, en la discusión no podremos concluir si el efecto sobre las comunidades de arañas es debido a LLC, a LC o a ambas, independientemente de cuál de ellas hayamos introducido en el modelo.

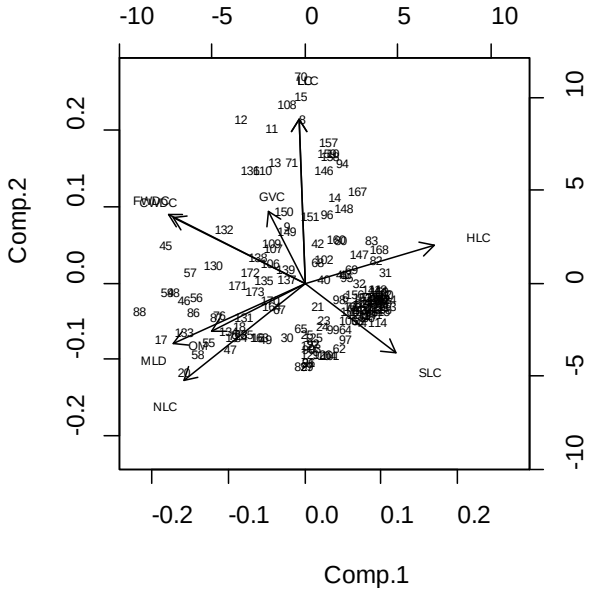
```
> setwd("C:/Users/Enrique/Desktop/Data analysis/CursoR")
> Spiders <- read.table (file="Spiders.txt", header=T)
> Spiders2 <- na.exclude(Spiders)
> MyVar <- c("GVC","HLC","SLC","NLC","LC",
+            "FWDC","CWDC","LLC","MLD","OM")
> source("HighstatLibV8.R")
> pairs(Spiders2[,MyVar], upper.panel = panel.smooth,
+       diag.panel = panel.hist, lower.panel=panel.cor, cex.cor=4)
```



6.3. Principal component analysis (PCA) para la identificación de colinealidad entre covariables

El análisis de componentes principales es una técnica de análisis multivariante que permite reducir la dimensionalidad de los datos. A partir de una serie de observaciones pertenecientes a variables posiblemente correlacionadas entre sí, esta técnica ordena estas observaciones en un espacio multidimensional en el que dichas dimensiones (componentes principales) no están correlacionadas. El gráfico asociado al PCA nos proporciona pues información de qué variables están correlacionadas entre sí, donde cada variable se representa con una línea y el ángulo entre estas líneas muestra la correlación entre esas variables. Si las líneas van en la misma dirección es que la correlación es positiva mientras que si van en dirección contraria, el coeficiente de correlación será negativo.

```
> cor_PCA <- princomp(Spiders2[,MyVar],cor=TRUE)
> biplot(cor_PCA, cex=c(0.5,0.5), col=1)
```



7. Lecturas recomendadas

i. Bolker MB (2008) Ecological models and data in R. Princeton University.

ii. Burnham KP, Anderson DR (2002) Model selection and multimodel inference. Springer

iii. Crawley MJ (2013) The R Book. Second edition. Wiley.

iv. Graham MH (2003) Confronting multicollinearity in ecological multiple regression. Ecology 84: 2809-2815.

v. Ieno EN, Zuur AF (2015) A beginner's guide to data exploration and visualization with R. Highland Statisitcs Ltd.

vi. Quinn GP, Keough MJ (2002) Experimental design and data analysis for biologists. Cambridge University Press.

vii. Zuur AF, Ieno EN, Meesters EHWG (2009) A beginner's guide to R. Springer

viii. Zuur AF, Ieno EN, Elphick CS (2010) A protocol for data exploration to avoid common statistical problems. Methods in Ecology and Evolution 1: 3-14.

Modelos de regresión lineal

Análisis de datos y modelos lineales en R

Enrique Andivia[1]

Master Universitario en Restauración de Ecosistemas
Curso 2019-2020

1 Departamento de Biodiversidad, Ecología y Evolución. Universidad Complutense de Madrid

email: e.andivia@gmail.com

web: http://enriqueandivia.wixsite.com/forests

MODELOS DE REGRESIÓN LINEAL

1. ¿Qué tipo de análisis debo aplicar?

Uno de los aspectos más complicado de la estadística, especialmente cuando damos nuestros primeros pasos en ella, es saber qué tipo de análisis es el más adecuado para cada caso. La elección del tipo de análisis viene determinada por la pregunta que quiere ser respondida, pero también por la propia naturaleza de las variables y efectos estudiados. Ya sabemos que tenemos dos tipos de variable, la variable respuesta que es en la que estamos interesados y la/s variable/s independientes o explicativas de las que nos interesa su relación con nuestra variable respuesta. En principio, si nuestras variables son todas continuas estamos ante un análisis de regresión, si por el contrario nuestras variables explicativas son todas variables categóricas (factores) entonces debemos aplicar un ANOVA (análisis de varianza), mientras que la combinación de variables explicativas continuas y categóricas implica la realización de un análisis de covarianza o modelo ANCOVA (tabla 1).

Variable Respuesta	Variables explicativas	Tipo de análisis
Continua	Continua(s)	Regresión lineal
Continua	Categórica(s)	ANOVA
Continua	Ambas	ANCOVA

Tabla 1: Tipo de modelo lineal en función de la naturaleza de las variables

El objetivo de nuestro análisis estadístico es estimar los parámetros del modelo que conducen al mejor ajuste del mismo a nuestros datos. El mejor modelo es aquel que explica la mayor variabilidad posible, siempre que los parámetros del modelo sean estadísticamente significativos. Es importante comprender que no existe un único modelo sino una gran cantidad de modelos que se ajustan a nuestros datos y es esencial seleccionar aquel que sea el más simple posible (principio de parsimonia). Cuando trabajamos con modelos estadísticos hay una serie de suposiciones que tenemos que tener muy en cuenta, la primera de ellas es que todos los modelos están mal, es decir son simplificaciones de la realidad y por tanto no podemos tomarlos como dogmas de fe, sin embargo algunos de ellos son útiles sobre todo cuando queremos deducir procesos a partir de patrones. Análogamente nunca podremos saber con certeza cuál es el mejor modelo ya que todos llevan asociado una incertidumbre, eso sí entre dos modelos que explique la misma variabilidad siempre debemos quedarnos con el más simple de todos (navaja de Occam). Por todo ellos es muy importante que antes de enredarnos en el análisis de los datos tengamos un profundo conocimiento del problema que vamos a estudiar así como tener bien claro los objetivos de nuestro estudio. Esto nos lleva a plantear nuestro modelo en base a nuestras hipótesis previas, hay que evitar el “fishing” es decir buscar y buscar hasta encontrar alguna relación entre variables. Finalmente, no hay que olvidar que antes de realizar nuestros análisis debemos explorar nuestros datos.

1.2. Supuestos de los modelos lineales

Anteriormente hemos visto cómo elegir el modelo lineal a aplicar en función de la naturaleza de nuestras variables independientes (tabla 1), sin embargo la aplicación de un modelo lineal queda supeditado al cumplimiento de una serie de supuestos.

- *Supuesto de normalidad:* Los modelos lineales asumen una distribución normal de los residuos del modelo. En el capítulo dedicado al análisis exploratorio de datos vimos cómo evaluar la normalidad sobre las variables y explicamos que el criterio de normalidad se refiere a los residuos del modelo y no a nuestra variable dependiente. Es cierto que, en parte, la distribución de los residuos del modelo viene definida por la distribución de nuestra variable dependiente. Por ejemplo, si nuestra variable dependiente son unos y ceros (presencia/ausencia o vivo/muerto), o bien es un conteo (nº individuos) los residuos de nuestro modelo lineal no van a ser normales. En este caso específico utilizaremos modelos lineales generalizados (ver capítulo 5). Cabe decir que los modelos lineales permiten cierto margen en cuanto a la desviación de la normalidad de los residuos pero una violación manifiesta de este principio puede invalidar las inferencias que hagamos a partir de nuestro análisis.
- *Supuesto de homocedasticidad:* Para evitar estimar la varianza de nuestra variable respuesta para cada valor de la variable independiente, se asume que la varianza es constante en todo el rango de la variable independiente. Esto implica que la varianza residual de nuestro modelo debe ser constante. La violación de este supuesto puede hacernos caer en errores de tipo I (falsos positivos). Para evaluar este aspecto se representan los residuos frente a los valores ajustados por el modelo. Uno de los principales problemas que suele haber es que los residuos aumenten conforme los valores ajustados por el modelo también lo hacen. Esto puede deberse al efecto de alguna variable que no se ha tenido en cuenta o bien a la presencia de outliers en nuestros datos. Como vimos en el capítulo anterior una forma de lidiar con este aspecto es transformar nuestra variable respuesta.
- *Supuesto de linealidad:* Lo cual no significa que la relación entre la variable dependiente y la independiente sea lineal si no que dicha relación puede expresarse como una combinación lineal de parámetros.
- *Supuesto de independencia:* Este supuesto se basa en que las observaciones que realizamos deben ser independientes entre sí, y por lo tanto los residuos del modelo también lo serán. Cuando estas observaciones no son independientes entre sí, no pueden considerarse verdaderas réplicas y en este caso a dichas observaciones no independientes se le denominan pseudorréplicas. Este supuesto es esencial para la aplicación de un modelo lineal, ya que la violación de este supuesto implica un error en el cálculo de los grados de libertad. Por ejemplo, si muestreamos 10 individuos de encinas para medir la concentración de nitrógeno de sus hojas y en cada individuo medimos 5 hojas distintas, tendremos 50 observaciones pero realmente el número de réplicas independientes será de 10 y por lo tanto los grados de libertad deberían ser 10 y

no 50. Basta pensar que la concentración de nitrógeno será más parecida entre las 5 hojas de un mismo individuo que entre hojas de distintos individuos, a este hecho se le denomina autocorrelación. Las formas más comunes de violar el supuesto de independencia son la autocorrelación espacial, cuando distintas observaciones están más próximas entre sí que otras, y la autocorrelación temporal cuando se realizan distintas mediciones de nuestra variable dependiente en los mismos individuos. El supuesto de independencia está muy relacionado con el diseño de nuestro estudio o muestreo por lo que se puede evitar siguiendo una serie de pautas. Por otro lado podemos evitarlo promediando el valor de nuestra variable dependiente por cada réplica. Por ejemplo en el caso de las encinas teniendo en cuenta solo el valor medio de la concentración de N de las 5 hojas pertenecientes a cada individuo. Por último existen herramientas estadísticas para tener en cuenta esta no independencia sin perder la variabilidad asociada a ella, son los modelos lineales mixtos (capítulo 6).

2. Regresión lineal simple

2.1. Fundamento matemático de la regresión lineal

El ejemplo más simple de un modelo lineal es cuando queremos evaluar el efecto de una variable independiente sobre nuestra variable respuesta. Si queremos escribir esta relación de forma matemática, la manera más simple es que nuestra variable respuesta (*Y*) sea una función lineal de nuestra variable explicativa (*X*):

$$Y = a + bx$$

Es decir si queremos saber cuánto vale *Y* para un determinado valor de *x* basta con tomar dicho valor multiplicarlo por *b* y sumarle *a*. Basta pues estimar estos dos parámetros para conocer el valor de nuestra variable respuesta para cada valor de la variable independiente. Al parámetro *a* se le denomina intercepto y es el valor de nuestra variable respuesta cuando la variable independiente es igual a cero. Al parámetro *b* se le denomina pendiente y es cuánto aumenta nuestra variable respuesta por unidad de la variable independiente.

El siguiente paso es estimar dichos parámetros a partir de la muestra que nosotros hemos analizado. Para entender cómo se estiman tenemos que recordar que los datos de una regresión son tomados de manera pareada, es decir que en una misma observación tenemos un dato para nuestra variable respuesta (Y_i) y para nuestra variable independiente (x_i). Sin embargo los parámetros *a* y *b* son los mismos para todos los pares de datos por lo que al menos que nuestro modelo prediga con exactitud cada una de nuestras observaciones tendremos un error asociado al modelo. Dicho error o residuo (ε_i) es la diferencia entre el valor predicho de la variable respuesta ($\hat{Y}_i$) para un determinado valor de la variable independiente y el valor real observado de nuestra variable respuesta (Y_i) en dicha observación (Fig 1). Este error, es una variable aleatoria que sigue una distribución normal. Para cada observación tendremos:

$$Y_i = a + bx_i + \varepsilon_i$$

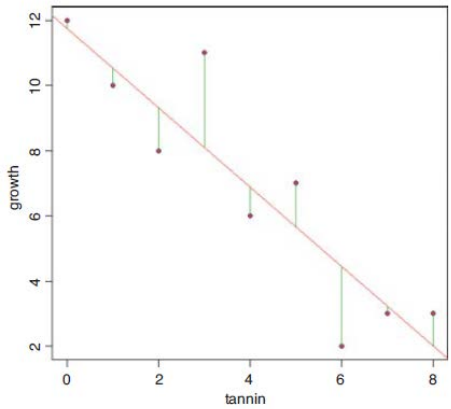


Figura 1: Representación gráfica del concepto de residuo en un modelo de regresión lineal entre una variable respuesta (growth) y una variable independiente (tannin). Modificado de Crawley 2013.

Los "mejores" parámetros para este modelo se estiman mediante máxima verosimilitud, es decir dados los datos el valor de los parámetros será aquel que haga los datos más verosímiles o lo que es lo mismo, que la probabilidad de que para un determinado valor de la variable independiente el modelo prediga el valor observado para la variable respuesta sea máxima. Si los supuestos de los modelos lineales se cumplen dicha máxima verosimilitud viene dada por el método de los mínimos cuadrados. El término mínimos cuadrados se refiere a la suma de los residuos al cuadrado. Se eleva los residuos al cuadrado para quedarnos exclusivamente con su magnitud. De modo que los mejores parámetros del modelo serán aquellos que minimicen la suma de los residuos cuadrados del modelo. La suma de cuadrados de los residuos (*RSS*) se define matemáticamente como:

$$RSS = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Para el cálculo de los parámetros del modelo debemos definir previamente la suma de cuadrados de cada variable (*SSY* y *SSX*), así como la suma del producto de ambas (*SSXY)*. Las sumas de cuadrados se definen como el sumatorio cuadrado de la diferencia de cada observación con respecto a la media.

$$SSY = \sum_{i=1}^{n} (Y_i - \bar{Y})^2$$

$$SSX = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$SSXY = \sum_{i=1}^{n} (x_i - \bar{x})(Y_i - \bar{Y})$$

Cabe destacar que si dividimos el SSY y el SSXY por (n-1) obtendremos la varianza de Y (S^2_y) y lo que denominamos covarianza (S_{xy}) que es un indicador de cómo varía una variable con respecto a otra. La varianza al ser una suma de cuadrados es siempre positiva, sin embargo esto no sucede para la covarianza, por lo que pueden darse para una determinada observación

valores positivos y negativos, en estos casos la suma total estará en torno a cero indicando que no hay relación entre ambas variables. A partir de estas sumas podemos estimar primero la pendiente (*b*) y posteriormente el intercepto (*a*):

$$b = \frac{SSXY}{SSX}$$

$$a = \bar{Y} - b\bar{x}$$

Aparte de estimar los parámetros de la ecuación de la recta nos quedaría por estimar un parámetro asociado al error de nuestro modelo. El error de nuestro modelo es una variable aleatoria que sigue una distribución normal, de ahí el supuesto de normalidad, con media igual a cero y varianza σ^2. En el caso que nuestras predicciones sean muy similares a nuestras observaciones, menos variabilidad tendremos en los residuos y por tanto la varianza del error estará más cercana a cero. Por tanto el error de nuestro modelo tiene un cierto paralelismo con la definición de la suma cuadrada de residuos, por lo que para estimar la varianza asociada al error basta calcular el valor medio de esta suma de cuadrados:

$$\sigma^2 = \frac{RSS}{n-2} = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-2}$$

Como vemos la suma cuadrada de residuos se divide por n-2, esto es debido a que ya hemos usado dos observaciones para calcular los dos parámetros del modelo (*a* y *b*), por lo que el número de observaciones independientes (esto es los grados de libertad) para calcular un nuevo parámetro es igual al número total de observaciones menos dos.

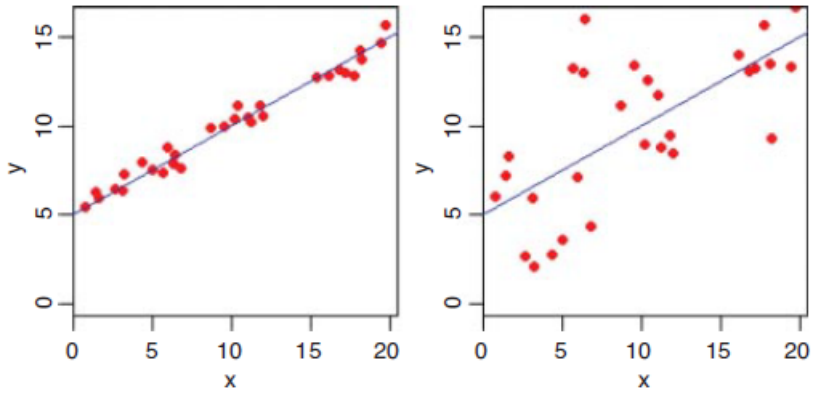


Figura 2: Representación gráfica de dos modelos lineales con similares parámetros pero distinto grado de ajuste al set de datos. Modificado de Crawley 2013.

Asociado a los errores del modelo existe otro aspecto importante que debemos tener en cuenta, el grado de ajuste del mismo. En la figura 2 vemos como dos modelos con el mismo intercepto y pendiente pueden ajustarse de forma distinta a los datos, por lo que necesitamos una forma de evaluar dicho ajuste. Para ello se utiliza el coeficiente de determinación (R^2) que se define como la fracción de la variabilidad de nuestra variable respuesta explicada por la variación de la variable independiente, o lo que es lo mismo por nuestro modelo. Si la varianza total de nuestra variable respuesta es *SSY* y la parte no explicada por el modelo es la suma cuadrado de residuos (*RSS*), definimos la parte de la variabilidad explicada por el modelo (*SSR*) como:

$$SSR = SSY - RSS$$

Como el coeficiente de determinación es la proporción de la varianza explicada por el modelo:

$$R^2 = \frac{SSR}{SSY} = \frac{SSY - RSS}{SSY}$$

Al ser una proporción, el coeficiente de determinación se define entre 0 y 1, cuanto más se acerque a 1 mayor será la variabilidad explicada y mejor se ajustará nuestro modelo a los datos. Es importante recordar que aunque el coeficiente de determinación sea 1, esto no significa que haya una relación causa-efecto entre nuestra variable respuesta y las variables independientes.

Es importante entender que los parámetros del modelo son realmente una estimación de los verdaderos valores de los parámetros poblacionales. Por ello, estos parámetros llevan asociados una incertidumbre. El siguiente paso es testar estadísticamente si los parámetros estimados son o no estadísticamente distintos de cero, esto es especialmente importante para la pendiente ya que su significación nos indicaría que hay un efecto de la variable independiente sobre nuestra variable respuesta. Como en cualquier contraste de hipótesis, nuestra hipótesis nula es que la estimación de nuestro parámetro es igual a cero, en caso de que esta hipótesis nula no se rechace concluiremos que no tenemos indicios para establecer una relación causa-efecto entre nuestras variables. La significación de estos parámetros se comprueba mediante un t-test. Por otro lado podemos realizar un análisis de la varianza para nuestro modelo de regresión lineal a partir de la variabilidad explicada por nuestro modelo (*SSR*), la no explicada (*RSS*) y la total de nuestra variable dependiente (*SSY*). Para un modelo de regresión lineal la tabla de análisis de varianza tendrá este aspecto:

Fuente	Suma cuadrados	Grados de libertad	Cuadrados medios	F-ratio
Modelo	*SSR*	1	*SSR/1*	*(SSR/1)/(RSS/n-1-1)*
Error	*RSS*	n-2	*RSS/n-1-1*	
Total	*SSY*	n-1		

Tabla 2: Tabla de análisis de la varianza para un modelo de regresión lineal simple

La hipótesis nula en un modelo de regresión lineal es que la pendiente no es significativamente distinta de cero. Para testar si se cumple o no, tenemos que comparar el valor del estadístico *F* con el valor crítico de *F* (basado en la distribución del estadístico) con un grado de libertad en el numerador (número de parámetros estimado en el modelo menos 1) y n-2 grados de libertad en el denominador.

2.2. Regresión lineal simple en R

Los modelos lineales en R, independientemente del número y naturaleza de las variables independientes se ajustan mediante la función *lm*. Vamos a trabajar en este ejemplo con unos datos sobre abundancia de larvas de ranas en 35 charcas independientes. En cada una de estas charcas se midieron una serie de característica de las mismas, una de ella fue un índice de degradación del hábitat, donde valores más cercanos a 0 indican que no existe perturbación. Se recomienda antes de empezar realizar un apropiado análisis exploratorio de los datos.

```
> setwd("C:/Users/Enrique/Desktop/Data analysis/CursoR")
> ranas <- read.table (file="datos_ranas", header=T,sep=";")
> str(ranas)
'data.frame':  35 obs. of  6 variables:
```

```
 $ Pond  : Factor w/ 35 levels "ani","ara","arr",..: 27 31 19 10 24 20
1 25 11 28 ...
 $ TH2   : num  2 1.9 2.2 1.3 1.2 3.5 2.9 2.4 2 2 ...
 $ LandsH: num  0.819 0.325 0.673 0.611 0.325 ...
 $ Area  : int  286 741 858 2730 1800 2320 728 650 1075 350 ...
 $ Prf   : num  0.8 0.5 0.7 1 0.7 0.9 1.3 0.7 0.4 0.5 ...
 $ Abun  : num  11.7 18.3 22.5 18.7 16.5 ...

> mod <- lm(Abun~TH2,data=ranas)
> summary(mod)

Call:
lm(formula = Abun ~ TH2, data = ranas)

Residuals:
    Min      1Q  Median      3Q     Max
-15.331  -3.513   0.788   3.320  15.590

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   21.265      2.954   7.199 2.98e-08 ***
TH2           -3.204      1.014  -3.161  0.00336 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.743 on 33 degrees of freedom
Multiple R-squared:  0.2324,   Adjusted R-squared:  0.2092
F-statistic: 9.993 on 1 and 33 DF,  p-value: 0.00336
> anova(mod)
Analysis of Variance Table

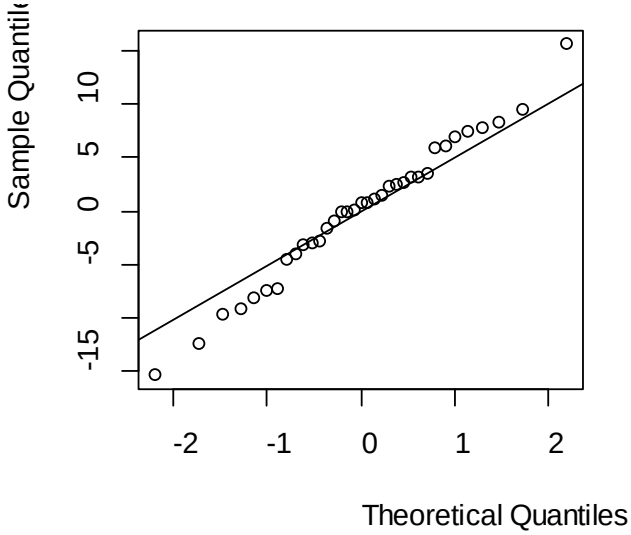
Response: Abun
          Df Sum Sq Mean Sq F value  Pr(>F)
TH2        1  454.4  454.40  9.9927 0.00336 **
Residuals 33 1500.6   45.47
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Con la función *summary* aplicada sobre el modelo accedemos a la información relativa a los parámetros estimados (estimate) del modelo (intercepto y pendiente) además del contraste de hipótesis (t-test) aplicado sobre los mismos (valor de t y p) y el error asociado a la estimación de dichos parámetros. También tenemos información sobre el R^2. En la tabla ANOVA, por su parte tenemos información relativa a la suma de cuadrados (modelo y residual), el valor del estadístico F y la significación del mismo para los grados de libertad del numerador y denominador específico de nuestro análisis. Podemos concluir que existe un efecto de la degradación del hábitat sobre la abundancia de larvas de ranas y que dicho efecto es negativo (pendiente negativa). El siguiente paso es comprobar que se cumplen los supuestos de los modelos lineales a partir de los residuales del mismo. Vamos a extraer primero los residuales del modelo, realizar un histograma y un gráfico Q-Q y comprobar la normalidad de los residuos con un test de Shapiro-Wilk.

```
> resi <- residuals(mod, type="pearson") # residuales de Pearson
> hist(resi, breaks = 5, xlab = "", main = "")
> qqnorm(resi, main="")
> qqline(resi)
> shapiro.test (resi)
```

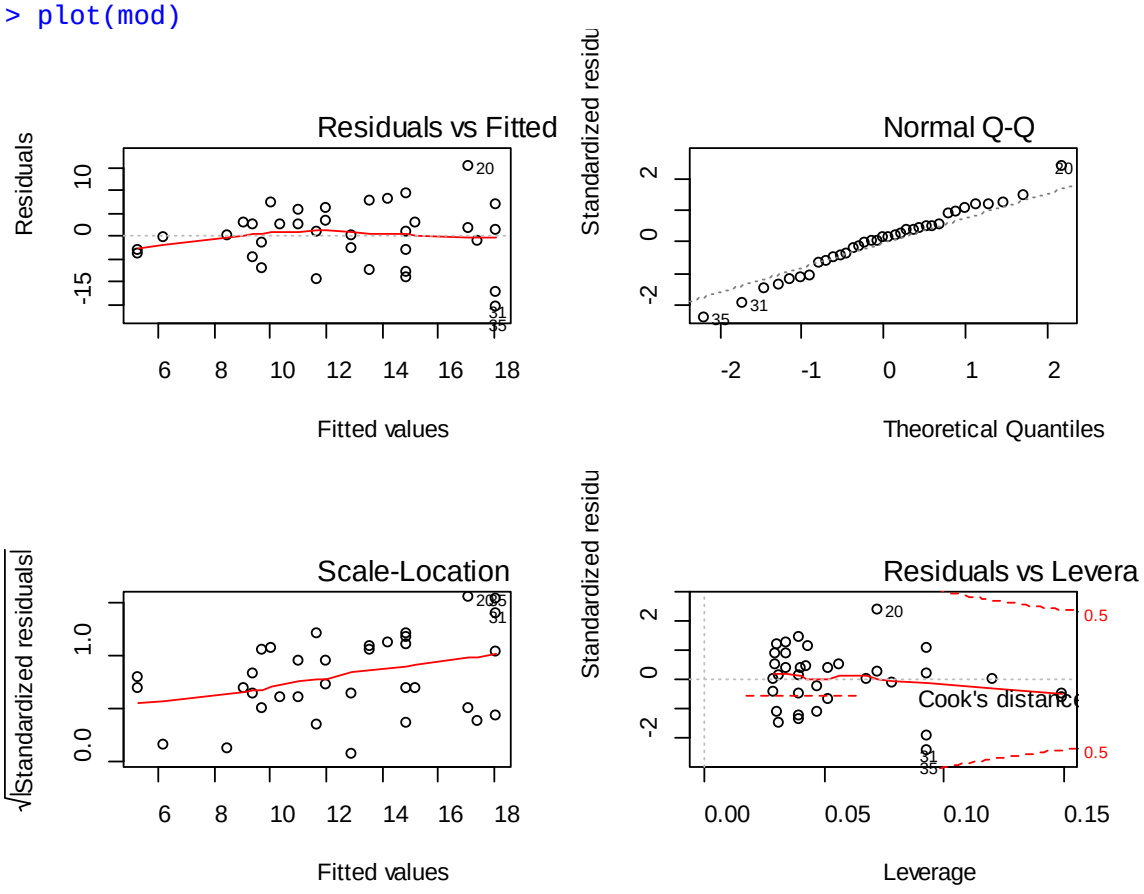
```
        Shapiro-Wilk normality test

data:  resi
W = 0.98681, p-value = 0.9421
```



Si aplicamos la función *plot* sobre nuestro modelo obtendremos 4 gráficos distintos para comprobar la idoneidad del modelo ajustado.

```
> par(mfrow=c(2,2))
> plot(mod)
```



El primer gráfico muestra los residuos frente a los valores ajustados y sirve para evaluar la homocedasticidad. Lo que no queremos encontrarnos es un patrón detectable en este gráfico,

idealmente el gráfico debería asemejarse a una nube aleatoria de puntos. En nuestro caso parece que los residuos aumentan conforme aumentan los valores ajustados lo que podría indicar que no se cumple el supuesto de homocedasticidad. Sin embargo esta tendencia es debida al efecto de un par de puntos aislados. El segundo gráfico ya lo conocemos y sirve para evaluar la normalidad de los residuos. El tercer gráfico es análogo al primero pero en otra escala que permite visualizar si hay alguna tendencia, como podemos observar el patrón identificado en el primer gráfico se hace algo más patente. En el último gráfico podemos evaluar los puntos más influyentes en la estimación de los parámetros del modelo. En este gráfico la distancia de Cook se expresa como un contorno al igual que las curvas de nivel en un mapa geológico. En este caso específico no parece haber puntos de los que preocuparnos en cuanto a su influencia en la estimación de los parámetros.

3. Regresión lineal múltiple

La regresión lineal múltiple es un modelo lineal con dos o más variables independientes continuas. Los modelos múltiples son muy útiles para evaluar el efecto conjunto de diversas variables en nuestra variable de interés, sin embargo presentan una serie de características a tener en cuenta y que pueden suponer un problema de cara a la robustez de nuestros análisis. Uno de los principales problemas es el *over-fitting* que incluso puede llevar a tener más variables independientes que observaciones. La inclusión de variables en el modelo dispara el número de parámetros a estimar, de forma general se suele recomendar tener al menos 10 observaciones independientes por parámetro a estimar, aunque como siempre esto no debe tomarse a raja tabla y dependerá en parte de la naturaleza de nuestro estudio. Otro problema que ya vimos en el análisis exploratorio de datos es la colinealidad entre variables explicativas. Finalmente la inclusión de diversas variables explicativa dará lugar a una batería de modelos de entre los cuales tendremos que seleccionar el mejor de ellos. Para ello tendremos que seguir el principio de parsimonia, es decir seleccionar el modelo más simple que explique razonablemente bien nuestro fenómeno de interés.

3.1. Fundamento matemático de la regresión lineal múltiple

Tanto la formulación matemática como los supuestos de estos modelos son análogos a los de los modelos de regresión simple. Es decir, el error está normalmente distribuido y la varianza es constante. La formulación para el modelo de regresión múltiple más sencillo considera un intercepto y un valor de la pendiente para cada variable explicativa:

$$Y_i = a + b_1 x_{1i} + b_2 x_{2i} + \varepsilon_i$$

Sin embargo el efecto de nuestras variables explicativas sobre la variable respuesta no ha de ser exclusivamente independiente, pudiera ser que la relación entre nuestra variable respuesta y una variable explicativa estuviera modulada por otra variable explicativa. Aunque este concepto es abstracto, basta con echar un ojo a un gráfico en tres dimensiones para entender este efecto (Fig 1):

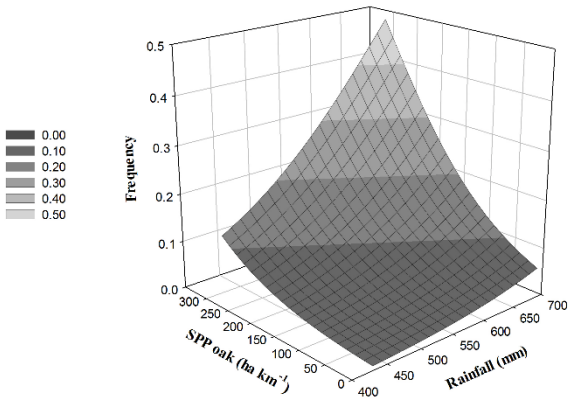


Figura 1: Ejemplo de gráfico tridimensional mostrando una interacción entre la disponibilidad de semilla (SPP oak) y la precipitación (Rainfall) en la regeneración de especies forestales (Frequency) leñosas en matorrales mediterráneos (Andivia et al. 2017).

Como vemos, ambas variables explicativas tienen un efecto positivo en la regeneración, sin embargo este efecto positivo es mucho más patente cuando ambos aumentan a la vez. En caso de que no hubiera interacción entre factores la representación gráfica de estas variables tendría como resultado un plano sin curvatura. La formulación más sencilla de una regresión con interacción entre factores implica estimar un nuevo parámetro que multiplique al producto de las dos variables independientes.

$$Y_i = a + b_1 x_{1i} + b_2 x_{2i} + b_3 x_{1i} x_{2i} + \varepsilon_i$$

Cuando incorporamos una interacción al modelo y esta resulta significativa los parámetros de las pendientes de dichas variables (b_1 y b_2) dejan de ser informativos. Además hay que tener presente que si una interacción es significativa, debemos mantener los efectos principales (pendientes) de estas variables por separado aunque dichos parámetros no resulten significativos. Otro aspecto importante de las regresiones múltiples es poder comparar el efecto de las variables independientes en función del parámetro de la pendiente. Podemos pensar que aquel que presente una mayor pendiente tendrá un efecto mayor sobre la variable respuesta, sin embargo el valor de la pendiente depende de la magnitud de las variables explicativas por lo que es necesario estandarizar dichas variables para que los parámetros sean comparables entre sí. Dicha estandarización se realiza a sustrayendo el valor medio a cada observación y dividiéndolo por la desviación estándar.

3.2. Simplificación y selección de modelos

El objetivo de la selección de modelos es elegir de entre todos los modelos potenciales un modelo con el mínimo número de parámetros posible pero con suficiente poder explicativo para los datos observados. El principio de parsimonia establece que ante el mismo poder explicativo seleccionaríamos un modelo con *k-1* parámetros antes que uno con *k* parámetros, un modelo con *k-1* variables explicativas antes que uno con *k* variables, un modelo lineal a uno no lineal o un modelo sin interacciones a uno con interacciones. El método más usado en la selección de modelos es el denominado *stepwise selection*. En este método partimos de un modelo máximo (full o maximal model) en el que consideramos todas nuestras variables de interés así como las interacciones en las que estemos interesados. Es importante que este modelo se base en nuestras hipótesis para así evitar el denominado *fishing*. Siguiendo el principio de parsimonia iremos eliminando términos no significativos empezando por las interacciones y continuando por

las variables explicativas. La comparación de modelos se ha venido realizando a partir de los *p* valores de los parámetros del modelo o bien mediante un test de la *F* o de *Chi-cuadrado* en los que se compara el ajuste de dos modelos distintos. En los últimos años, dicha comparación se viene realizando a partir de los criterios de información y más concretamente mediante el criterio de información de Akaike (AIC).

3.3. Criterio de información de Akaike (AIC)

El criterio de información de Akaike está basado en el logaritmo de verosimilitud, el cual es inversamente proporcional al ajuste del modelo, penalizado por el número de parámetros del modelo (*k*). Por lo tanto resulta muy útil para encontrar el modelo más simple pero manteniendo un ajuste razonable.

$$AIC = -2 \times log - likelihood + 2(k+1)$$

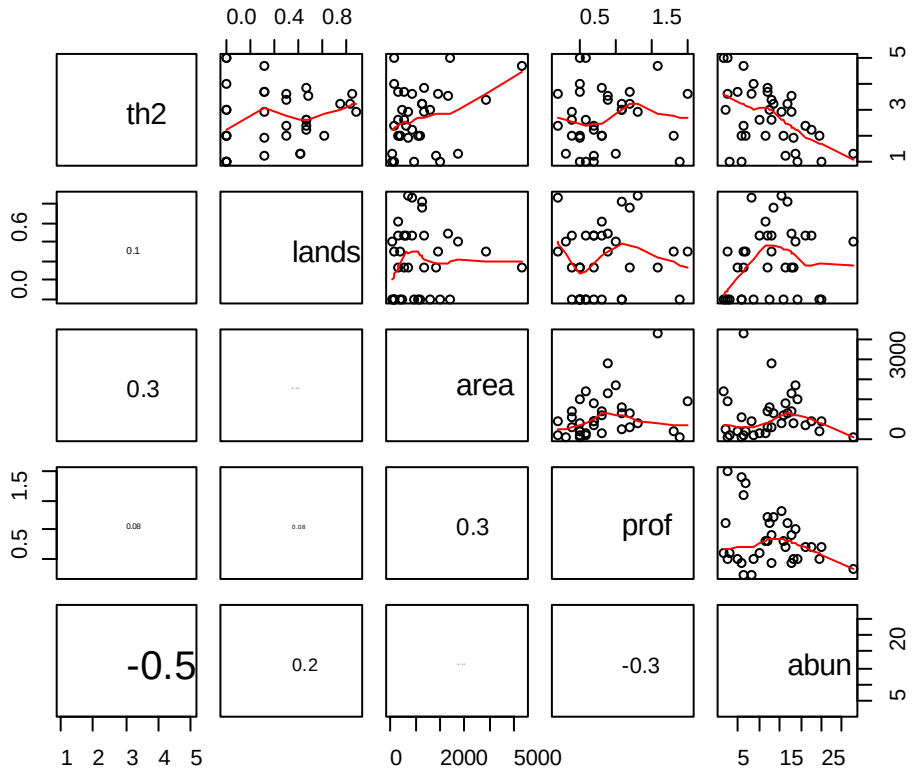
Cuanto mayor sea el valor de AIC peor será el modelo. Como regla general se establece una diferencia de dos unidades de AIC para justificar la selección de un modelo más complejo. Es decir, si comparamos un modelo con otro más simple y el AIC del modelo más simple es mayor en más de dos unidades que el AIC del modelo más complejo, seleccionaremos este último, en caso contrario seleccionaremos el modelo más simple. Algunos autores incluso aumentan este umbral a cinco o diez unidades de AIC cuando es una interacción el término evaluado. Existen otros criterios de información, como el criterio de información de Akaike corregido (AICc), el cual se recomienda para un número de observaciones pequeño, y el criterio de información bayesiano (BIC) recomendado para modelos con muchas observaciones.

3.4. Regresión lineal múltiple y selección de modelos en R

Anteriormente hemos visto la formulación para un modelo de regresión simple en el que la variable respuesta se establecía en primer lugar frente a la variable independiente (*y~x*). En el caso de regresiones múltiples (p.ej con dos variables independientes *x* y z) existen distintas formas de considerarlas en un modelo de regresión. Si solo estamos interesados en el efecto principal de cada uno de ellos, la formulación será *x+z*, mientras que si estamos interesados en la interacción entre ambas escribiremos *x:z*. También podemos estar interesados en los efectos principales y la interacción en cuyo caso la formulación será *x*z* (análogo a *x+z+x:z*). Vamos a ajustar un modelo de regresión múltiple y seleccionar el mejor modelo usando los datos sobre abundancia de ranas. Primero veremos las relaciones entre la variable respuesta (abundancia) y las variables explicativas.

```
> str(ranas)
'data.frame':  35 obs. of  6 variables:
 $ Pond  : Factor w/ 35 levels "ani","ara","arr",..: 27 31 19 10 24 20
1 25 11 28 ...
 $ TH2   : num  2 1.9 2.2 1.3 1.2 3.5 2.9 2.4 2 2 ...
 $ LandsH: num  0.819 0.325 0.673 0.611 0.325 ...
 $ Area  : int  286 741 858 2730 1800 2320 728 650 1075 350 ...
 $ Prf   : num  0.8 0.5 0.7 1 0.7 0.9 1.3 0.7 0.4 0.5 ...
 $ Abun  : num  11.7 18.3 22.5 18.7 16.5 ...
> th2 <- as.numeric(ranas$TH2)
> lands <- as.numeric(ranas$LandsH)
> area <- as.numeric(ranas$Area)
> prof <- as.numeric(ranas$Prf)
> abun <- as.numeric(ranas$Abun)
> dat <- data.frame(th2,lands,area,prof,abun)
```

```
> cor(dat) # obtener matriz de correlaciones
              th2      lands        area        prof        abun
th2    1.00000000 0.11394523  0.29864090  0.08293367 -0.48210821
lands  0.11394523 1.00000000  0.03410289  0.07709486  0.23464175
area   0.29864090 0.03410289  1.00000000  0.28961838 -0.02147081
prof   0.08293367 0.07709486  0.28961838  1.00000000 -0.26804196
abun  -0.48210821 0.23464175 -0.02147081 -0.26804196  1.00000000
> source("HighstatLibV8.R")
> pairs(dat,upper.panel=panel.smooth,lower.panel=panel.cor, cex.cor=4)
```



Antes de ajustar nuestro modelo vamos a estandarizar nuestras variables. El modelo que ajustaremos tendrá en cuenta el efecto de cada una de las variables explicativas y la interacción entre el índice de perturbación (th2) y la superficie arbolada alrededor (lands), y la interacción entre el área (area) y la profundidad de la charca (prof).

```
> dat$th2 <- (dat$th2-mean(dat$th2))/sd(dat$th2)
> dat$lands <- (dat$lands-mean(dat$lands))/sd(dat$lands)
> dat$area <- (dat$area-mean(dat$area))/sd(dat$area)
> dat$prof <- (dat$prof-mean(dat$prof))/sd(dat$prof)
> mod1 <- lm(abun~th2+lands+area+prof+th2:lands+area:prof, dat)
> summary(mod1)

Call:
lm(formula = abun ~ th2 + lands + area + prof + th2:lands + area:prof,
    data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-12.084  -3.867  -0.146   3.958  12.002
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.79041    1.12306  11.389 5.06e-12 ***
th2         -4.81204    1.42012  -3.388   0.0021 **
lands        2.51317    1.10335   2.278   0.0306 *
area         1.86536    1.39174   1.340   0.1909
prof        -2.26443    1.15747  -1.956   0.0605 .
th2:lands   -1.02449    1.40704  -0.728   0.4726
area:prof   -0.09441    1.11625  -0.085   0.9332
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.293 on 28 degrees of freedom
Multiple R-squared:  0.4328,   Adjusted R-squared:  0.3113
F-statistic: 3.561 on 6 and 28 DF,  p-value: 0.009532

> step <- stepAIC(mod1, direction="back")
Start:  AIC=134.95
abun ~ th2 + lands + area + prof + th2:lands + area:prof

            Df Sum of Sq    RSS    AIC
- area:prof  1    0.2833 1109.1 132.96
- th2:lands  1   20.9954 1129.8 133.61
<none>                   1108.8 134.95

Step:  AIC=132.96
abun ~ th2 + lands + area + prof + th2:lands

            Df Sum of Sq    RSS    AIC
- th2:lands  1    20.756 1129.9 131.61
<none>                   1109.1 132.96
- area       1    91.684 1200.8 133.74
- prof       1   153.767 1262.9 135.50

Step:  AIC=131.61
abun ~ th2 + lands + area + prof

        Df Sum of Sq    RSS    AIC
<none>               1129.9 131.61
- area   1     83.07 1213.0 132.09
- prof   1    172.41 1302.3 134.58
- lands  1    190.12 1320.0 135.05
- th2    1    550.43 1680.3 143.50
```

Análogamente podemos hacer nosotros toda la batería de modelos paso a paso:

```
> modfull <- mod1
> mod2 <- update(modfull,~.-area:prof)
> mod3 <- update(modfull,~.-th2:lands)
> AIC(modfull,mod2,mod3)
        df      AIC
modfull  8 236.2764
mod2     7 234.2853
mod3     7 234.9329
> AICc(modfull)
[1] 241.8148
> AICc(mod2)
[1] 238.4335
> AICc(mod3)
[1] 239.081
```

```
> mod4 <- update(mod2,~.-th2:lands)
> AIC(mod2,mod4)
     df      AIC
mod2  7 234.2853
mod4  6 232.9342
> AICc(mod4)
[1] 235.9342
> mod5 <- update(mod4,~.-th2)
> mod6 <- update(mod4,~.-lands)
> mod7 <- update(mod4,~.-area)
> mod8 <- update(mod4,~.-prof)
> AIC(mod4,mod5,mod6,mod7,mod8)
     df      AIC
mod4  6 232.9342
mod5  5 244.8244
mod6  5 236.3773
mod7  5 233.4172
mod8  5 235.9047
> AICc(mod5)
[1] 246.8933
> AICc(mod6)
[1] 238.4463
> AICc(mod7)
[1] 235.4862
> AICc(mod8)
[1] 237.9737
> mod9 <- update(mod7,~.-th2)
> mod10 <- update(mod7,~.-lands)
> mod11 <- update(mod7,~.-prof)
> AIC(mod7,mod9,mod10,mod11)
      df      AIC
mod7   5 233.4172
mod9   4 242.9497
mod10  4 236.3918
mod11  4 234.7596
> AICc(mod9)
[1] 244.283
> AICc(mod10)
[1] 237.7251
> AICc(mod11)
[1] 236.093
> mod12 <- update(mod11,~.-th2)
> mod13 <- update(mod11,~.-lands)
> AIC(mod11,mod12,mod13)
      df      AIC
mod11  4 234.7596
mod12  3 244.1415
mod13  3 236.8652
> AICc(mod12)
[1] 244.9157
> AICc(mod13)
[1] 237.6394
> modfinal <- mod13
> summary(modfinal)

Call:
lm(formula = abun ~ th2, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-15.331  -3.513   0.788   3.320  15.590
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   12.650      1.140  11.099 1.12e-12 ***
th2           -3.656      1.156  -3.161  0.00336 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.743 on 33 degrees of freedom
Multiple R-squared:  0.2324,   Adjusted R-squared:  0.2092
F-statistic: 9.993 on 1 and 33 DF,  p-value: 0.00336
```

Hemos realizado la selección de modelos en base a AIC y AICc, como hemos visto la selección difiere entre ambos criterios de información. Según AIC el mejor modelo incluye el efecto principal de th2 y lands, sin embargo usando AICc solo quedaría retenido en el modelo final el th2. Como hemos comentado para muestras pequeñas se recomienda el AICc, además si hacemos un summary del modelo que contiene las dos variables independientes observamos que la pendiente para la variable lands no es significativa para un $p < 0.05$, aunque está muy próximo. Que no hayamos seleccionado lands no significa que en la discusión de nuestros resultados no hagamos referencia a esta variable indicando que podría ser una variable a tener en cuenta.

```
> summary(mod11)

Call:
lm(formula = abun ~ th2 + lands, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1786  -4.9440   0.0518   3.3673  13.9965

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   12.650      1.092  11.589 5.46e-13 ***
th2           -3.909      1.115  -3.507  0.00137 **
lands          2.225      1.115   1.996  0.05454 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.458 on 32 degrees of freedom
Multiple R-squared:  0.3174,   Adjusted R-squared:  0.2747
F-statistic: 7.439 on 2 and 32 DF,  p-value: 0.002222
```

En caso de seleccionar distintas variables podemos evaluar la importancia relativa de cada una de ellas comparando el incremente en AIC cuando dicha variable es eliminada del modelo, es decir cuánto empeora el modelo cuando no consideramos dicha variable. Otra forma de evaluar esta importancia es mediante la comparación de las pendientes para cada variable, siempre que estas hayan sido estandarizadas.

Vamos a ver otro ejemplo. En este caso cargaremos la base de datos Ozone.txt (Ieno y Zuur 2015) sobre mediciones de ozono en función de la radiación, la temperatura y el viento. Nuestro modelo partirá de la interacción triple entre tres variables independientes.

```
> Ozone <- read.table(file="Ozone.txt",header=T)
> str(Ozone)
'data.frame':  110 obs. of  4 variables:
```

```
 $ rad  : int  190 118 149 313 299 99 19 256 290 274 ...
 $ temp : int  67 72 74 62 65 59 61 69 66 68 ...
 $ wind : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
 $ ozone: int  41 36 12 18 23 19 8 16 11 14 ...
> head(Ozone)
  rad temp wind ozone
1 190   67  7.4    41
2 118   72  8.0    36
3 149   74 12.6    12
4 313   62 11.5    18
5 299   65  8.6    23
6  99   59 13.8    19
> for (i in 1:length(Ozone[,])) {
+   Ozone[,i] <- as.numeric(Ozone[,i])
+ }
> mean1 <- c(mean(Ozone[,1]),mean(Ozone[,2]),mean(Ozone[,3]))
> sd1 <- c(sd(Ozone[,1]),sd(Ozone[,2]),sd(Ozone[,3]))
> ms <- c(mean1,sd1)
> ms <- matrix(ms,ncol=3,nrow=2,byrow=TRUE)
> for (i in 1:3) {
+   {for (k in 1:length(Ozone[,i]))
+     Ozone[k,i] <- (Ozone[k,i]-ms[1,i])/ms[2,i]
+   }
+ }
> mfull <- lm(ozone~rad*temp*wind,data=Ozone)
> summary(mfull)

Call:
lm(formula = ozone ~ rad * temp * wind, data = Ozone)

Residuals:
    Min      1Q  Median      3Q     Max
-36.273 -10.384  -1.230   7.682  59.969

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    36.329      1.846  19.680  < 2e-16 ***
rad             7.620      1.949   3.910 0.000166 ***
temp           16.243      1.965   8.268 5.34e-13 ***
wind           -8.988      2.046  -4.393 2.74e-05 ***
rad:temp        4.829      1.907   2.532 0.012880 *
rad:wind       -1.985      2.094  -0.948 0.345391
temp:wind      -6.288      1.832  -3.432 0.000865 ***
rad:temp:wind  -1.814      1.896  -0.957 0.340845
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.88 on 102 degrees of freedom
Multiple R-squared:  0.7254,   Adjusted R-squared:  0.7065
F-statistic: 38.49 on 7 and 102 DF,  p-value: < 2.2e-16

> m1 <- lm(ozone~rad*temp+rad*wind+temp*wind,data=Ozone)
> AIC(mfull,m1)
      df      AIC
mfull  9 943.5799
m1     8 942.5631
> m2 <- update(m1,~.-rad:temp)
> m3 <- update(m1,~.-rad:wind)
> m4 <- update(m1,~.-temp:wind)
> AIC(m1,m2,m3,m4)
   df      AIC
m1  8 942.5631
```

```
m2  7 946.6064
m3  7 940.9556
m4  7 951.6077
> m5 <- update(m3,~.-rad:temp)
> m6 <- update(m3,~.-temp:wind)
> AIC(m3,m5,m6)
   df      AIC
m3  7 940.9556
m5  6 946.1578
m6  6 953.6951
> mfinal <- m3
> summary(mfinal)

Call:
lm(formula = ozone ~ rad + temp + wind + rad:temp + temp:wind,
    data = Ozone)

Residuals:
    Min      1Q  Median      3Q     Max
-36.749  -9.885  -1.977   8.566  59.868

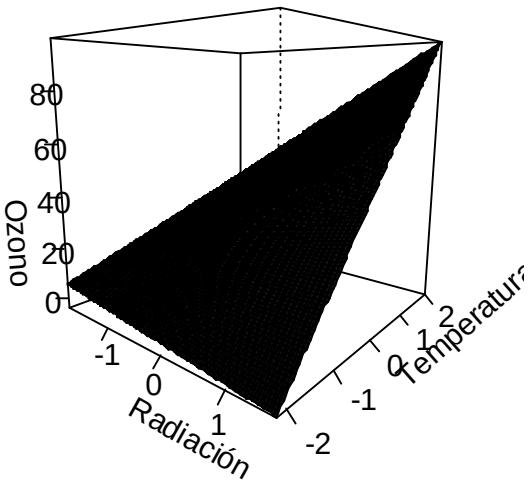
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   36.492      1.792  20.358  < 2e-16 ***
rad            7.756      1.892   4.101 8.20e-05 ***
temp          16.552      1.932   8.565 1.06e-13 ***
wind          -9.810      1.861  -5.272 7.36e-07 ***
rad:temp       4.824      1.818   2.653 0.009234 **
temp:wind     -6.187      1.602  -3.862 0.000196 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

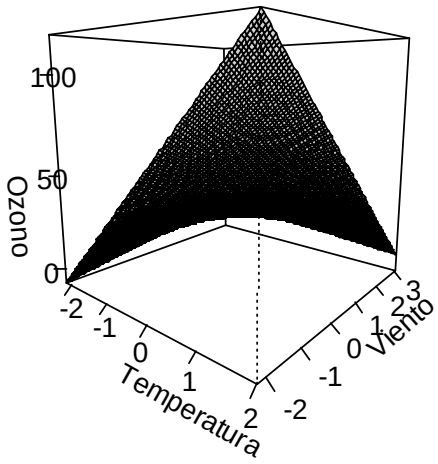
Residual standard error: 16.82 on 104 degrees of freedom
Multiple R-squared:  0.7219,   Adjusted R-squared:  0.7086
F-statistic:    54 on 5 and 104 DF,  p-value: < 2.2e-16
```

El modelo final selecciona las interacciones rad:temp y temp:wind, además claro de su efectos principales. Ahora vamos a realizar un gráfico en tres dimensiones de cada interacción a partir de las predicciones del modelo. Primero creamos los valores de las variables independientes donde vamos a predecir y posteriormente a partir de los parámetros (coeficientes) del modelo calculamos las predicciones para esos valores. Cuando predecimos para una de las interacciones, usamos el valor medio de la otra variable independiente como valor fijo. Como hemos estandarizado las variables el valor medio de una variable estandarizada es 0 por lo que no utilizaremos los parámetros de esa variable independiente en la predicción.

```
> rad <- seq(min(Ozone$rad), max(Ozone$rad), length=100)
> temp <- seq(min(Ozone$temp), max(Ozone$temp), length=100)
> wind <- seq(min(Ozone$wind), max(Ozone$wind), length=100)
>
> coef<-coef(mfinal)
> coef
(Intercept)         rad        temp        wind    rad:temp   temp:win
d
  36.492166    7.756377   16.551852   -9.810005    4.823752   -6.18682
1
> #Primero interacción rad*temp
> pred<-function(rad,temp)
+ {coef[1]+coef[2]*rad+
```

```
+     coef[3]*temp+coef[5]*rad*temp}
> z <- outer(rad,temp,pred)
> persp(x=rad,y=temp,
+       xlab="Radiación",ylab="Temperatura",zlab="Ozono",
+       z=z,theta=40, phi=10,   # theta y phi=rotación
+       shade=1.75, border = NULL,
+       axes=TRUE, nticks = 5,ticktype="detailed")
>
> #Segundo interacción temp*wind
> pred1<-function(temp,wind) # Como estandarizo la media de wind es 0
por lo que se van los coeficientes
+ {coef[1]+coef[3]*temp+coef[4]*wind+
+     coef[6]*temp*wind}
> z1 <- outer(temp,wind,pred1)
> persp(x=temp,y=wind,
+       xlab="Temperatura",ylab="Viento",zlab="Ozono",
+       z=z1,theta=40, phi=10,   # theta y phi=rotación
+       shade=1.75, border = NULL,
+       axes=TRUE, nticks = 5,ticktype="detailed")
```





3.5. Colinealidad en modelos lineales: Variance Inflation Factors (VIF)

El factor de inflación de la varianza proporciona una estimación cuantitativa de la multicolinealidad entre covariables en un análisis de regresión. Este índice mide hasta qué punto la varianza de un coeficiente de regresión estimado se incrementa a causa de la colinealidad. Si echamos un vistazo a la fórmula mediante la que se calcula la varianza asociada a la estimación de un parámetro mediante mínimos cuadrados veremos que a mayor R^2 (más cercano a 1) mayor varianza asociada al parámetros. Hay que aclarar que este R^2 es el cuadrado del coeficiente de correlación múltiple entre las variables independientes del modelo y no el coeficiente de determinación del mismo. En términos prácticos, si introducimos una variable en el modelo que aumenta la correlación entre las mismas significa que la información proporcionada por esta variable ya está incluida en el resto y además aumentamos la varianza asociada a la estimación de nuestros parámetros por lo que podemos estar influyendo la significación de los mismos.

$$Variance(b_j) = \frac{1}{1 - R_j^2} \times \frac{\sigma^2}{(n-1)S_j^2}$$

$$VIF_j = \frac{1}{1 - R_j^2}$$

En cuanto a los valores de VIF a partir de los cuales hay que tomar en serio la colinealidad entre nuestras variables no existe un consenso claro. El límite más aceptado es el de 5, aunque algunos autores establecen este límite en 10. Sin embargo cuando nuestras variables no tienen un efecto muy marcada sobre nuestra variable respuesta, tener valores de VIF mayores de 3 podría suponer un problema ya que muchos de los parámetros podrían dejar de ser significativos. El procedimiento a seguir es calcular el VIF sobre las variables independientes presentes en el modelo y en caso de alta colinealidad eliminar aquella con un VIF superior y volver a calcular el VIF para cada covariable.

```
> library(car)
> vif(mfinal)
      rad      temp      wind  rad:temp temp:wind
 1.378533  1.438884  1.333946  1.420112  1.117484
```

4. Lecturas recomendadas

i. Bolker MB (2008) Ecological models and data in R. Princeton University.

ii. Burnham KP, Anderson DR (2002) Model selection and multimodel inference. Springer

iii. Cayuela L (2015) Modelos lineales: regresión, ANOVA y ANCOVA.

iv. Crawley MJ (2013) The R Book. Second edition. Wiley.

v. Graham MH (2003) Confronting multicollinearity in ecological multiple regression. Ecology 84: 2809-2815.

vi. Ieno EN, Zuur AF (2015) A beginner's guide to data exploration and visualization with R. Highland Statistics Ltd.

vii. Quinn GP, Keough MJ (2002) Experimental design and data analysis for biologists. Cambridge University Press.

Modelos lineales con factores: ANOVA y ANCOVA

Análisis de datos y modelos lineales en R

Enrique Andivia[1]

Master Universitario en Restauración de Ecosistemas
Curso 2019-2020

1 Departamento de Biodiversidad, Ecología y Evolución. Universidad Complutense de Madrid

email: e.andivia@gmail.com

web: http://enriqueandivia.wixsite.com/forests

MODELOS LINEALES CON FACTORES: ANOVA y ANCOVA

1. Uso de factores como variables predictoras

En el módulo anterior hemos visto como ajustar modelos lineales cuando nuestras variables independientes son cuantitativas y continuas (i.e. regresión). Sin embargo, muchos diseños experimentales resultan en la exposición de nuestro material experimental a distintos *niveles* de un determinado *factor*. Este factor no es más que una variable categórica y sus niveles son los distintos valores que toma dicho factor. Pongamos un ejemplo. Queremos saber cuál es el efecto de la fertilización con nitrógeno sobre la producción agrícola de un campo de maíz. Para ello sometemos a las plantas a distintas dosis de N, concretamente a 5 dosis (0, 10, 20, 40 y 80 mgN/planta). Estamos pues ante un diseño experimental en el que se evaluará el efecto de 5 niveles distintos (dosis de N) de un determinado factor (fertilización nitrogenada) sobre la producción de maíz. Siguiendo con la terminología es importante decir que el término factor es muchas veces sustituido por *tratamiento*, utilizándose ambos de forma indistinta. Como hemos visto en el ejemplo, el término factor o tratamiento hace referencia a cada una de las manipulaciones que llevamos a cabo en un estudio experimental (dosis de nitrógeno o niveles de riego). En estudios observacionales, por su parte, el término factor o tratamiento se utiliza para referirse a cada uno de los grupos que queremos comparar. Por ejemplo, si queremos comparar cual es el estado hídrico de una determinada planta en función de si crece bajo árbol, matorral o sin vegetación colindante, el tipo de hábitat, en este caso con tres niveles distintos, será el factor o tratamiento cuyo efecto en nuestra variable respuesta queremos comprobar.

1.1. ANOVA: un caso particular de modelo lineal

A este tipo de diseño o análisis estadístico en el cual nuestra variable respuesta es continua y nuestra variable independiente categórica se le denomina ANOVA (ANalysis Of VAriance). Es importante que entendamos que una regresión y un ANOVA son análogas, en ambos casos estamos ante modelos lineales, excepto por la naturaleza de la variable independiente. De hecho, si pensamos en el ejemplo de las distintas dosis de nitrógeno aplicadas a las plantas de maíz podemos comprobar que existe un pequeño paso entre realizar un ANOVA teniendo en cuenta el efecto de los 5 niveles de fertilización sobre la producción de maíz y ajustar una regresión lineal entre dicha producción y la dosis de nitrógenos aplicada (figura 1).En el apartado 2 veremos de forma más concreta como un ANOVA se ajusta a un modelo lineal de forma análoga a como se hace en una regresión lineal.

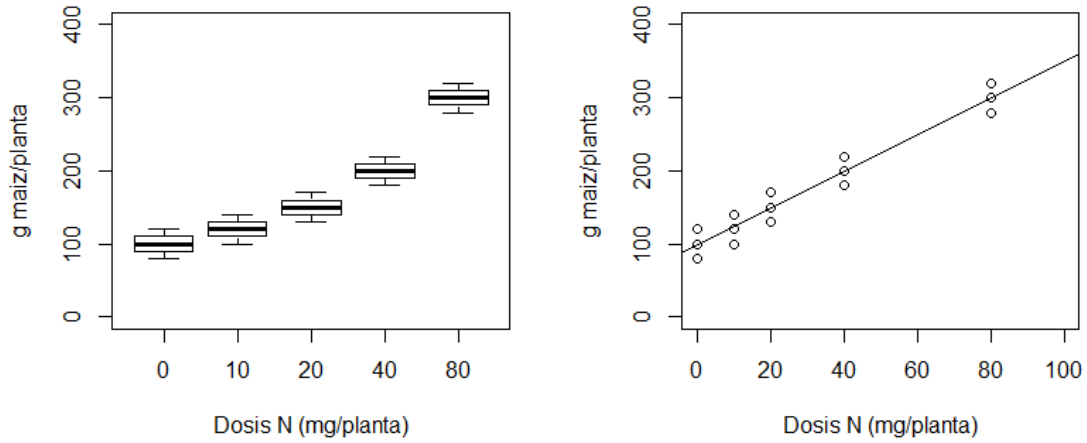


Figura 1: Representación gráfica de la producción de maíz por planta en función de la dosis de nitrógeno aportada (izquierda); y regresión lineal ajustada al mismo set de datos (derecha).

1.2. ¿Cómo funciona un ANOVA?

Generalmente, el objetivo detrás de un ANOVA es comprobar si se cumple o no una determinada hipótesis mediante la comparación de los valores medios de nuestra variable respuesta en cada uno de los niveles que conforman un factor. Sin embargo la comparación de dichos valores medios se realiza mediante la comparación de sus varianzas. Para entender esto vamos a partir del ejemplo más sencillo, un experimento con solo dos niveles del mismo factor. En primer lugar debemos calcular la variación total de nuestra variable respuesta (*y*), para ello calcularemos la *suma total de cuadrados* (*SST*, en sus siglas en inglés) que es la suma de la diferencia al cuadrado entre cada uno de los valores medidos de nuestra variable respuesta (y_i) y el valor medio de la misma ($\bar{y}$):

$$SST = \sum(y_i - \bar{y})^2$$

Sin embargo sabemos que nuestros datos fueron tomados en dos niveles distintos (*a* y *b*) de un tratamiento o factor determinado (*x*), el cual hipotetizamos que tendrá un efecto sobre nuestra variable respuesta. Es por ello que debemos conocer cuánto varían nuestros datos con respecto al valor medio de su nivel de referencia y no respecto al valor medio total. Para ello calculamos la *suma de errores cuadrados* (*SSE*, en sus siglas en inglés):

$$SSE = \sum(y_{i,A} - \bar{y}_A)^2 + \sum(y_{i,B} - \bar{y}_B)^2$$

A partir de estas definiciones resulta inmediato darse cuenta que si el valor medio de la variable respuesta es el mismo para los dos niveles del factor de estudio, entonces *SST* y *SSE* son iguales. Sin embargo, ¿qué pasará conforme dichos valores medios sean más diferentes? Basta pensar un poco para darse cuenta que si la diferencia entre los valores medios de los niveles del factor es muy elevada la distancia entre el valor de la variable respuesta y el valor medio total será mucho mayor que la distancia entre el valor de la variable respuesta y el valor medio del nivel de su factor, y por tanto *SSE* será menor que *SST*. Es por ello que definimos la *suma de errores cuadrados del tratamiento* (*SSA*) como la diferencia entre *SST* y *SSE*. El valor de esta

suma nos dirá cómo es la diferencia entre los valores medios para ambos niveles del factor de modo que a mayor *SSA* mayor será esta diferencia y viceversa. A partir de esto podemos interpretar cada una de las celdas de la tabla ANOVA (tabla 1) resultante de este tipo de análisis estadístico.

	Suma cuadrados (*SS*)	*g.l*	Error cuadrático medio (*MS*)	*F*	*Sig*
Factor	*SSA*	$k-1$	$MSA = \frac{SSA}{k-1}$	$\frac{MSA}{s^2}$	*p*
Error	*SSE*	$k(n-1)$	$s^2 = \frac{SSE}{k(n-1)}$		
Total	*SST*	$kn-1$			

Tabla 1: Tabla resultante de un análisis ANOVA genérico con un solo factor

Las sumas de cuadrados (*SS*) son divididas por sus grados de libertad (*g.l*) para calcular la varianza del error cuadrático medio (*MS*), donde *k* es igual al número de niveles del tratamiento menos 1 y *n* es igual al número de réplicas para cada nivel de un factor. La significación de la diferencias entre medias se evalúa mediante un test *F*, cuyo estadístico es calculado como el cociente entre la varianza del tratamiento o factor y la varianza del error. El valor de este estadístico se compara finalmente con el valor crítico de *F* para un determinado nivel de significación (normalmente 0.95) y grados de libertad del numerador (*k – 1*) y del denominador (*k(n – 1)*).

1.3. Suposiciones del ANOVA

Como hemos comentados con anterioridad un análisis ANOVA no es más que un caso específico de un modelo lineal y por tanto ha de cumplir los mismos supuestos que en el caso de las regresiones lineales, especialmente aquellos relacionados con la normalidad en la distribución de los residuos del modelo y con la homogeneidad de varianzas. Otro supuesto muy importante del ANOVA es la independencia de sus muestras y por tanto de sus errores. En los diseños ANOVA estándar cada réplica debe ser independiente del resto de réplicas de dicho tratamiento y del conjunto total de réplicas. Es decir que cada réplica no puede influir en otra. La violación de este supuesto da lugar a autorreplicación y por tanto hay que tener en cuenta otros diseños y modelos específicos. El resto de supuestos son la aleatorización del muestreo, es decir que nuestras observaciones son tomadas al azar, y el efecto aditivo de los tratamientos que no es más que decir que la respuesta de una determinada observación a un tratamiento puede ser separada en la respuesta de la observación *per se* y la respuesta por haber recibido dicho tratamiento.

2. El ANOVA más simple: one-way ANOVA

Para ver los pasos necesarios en el análisis ANOVA vamos a trabajar con un ejemplo concreto. Vamos a cargar los datos “RxF_growth.csv” el cual está delimitado por comas. Este experimento tiene varios factores pero para este ejemplo nos centraremos en el efecto del riego sobre el crecimiento. En un campo agrícola abandonado se plantaron 144 plantas de pino piñonero que fueron sometidas a distintos tratamientos de riego: “R0”, “R1”, “R3” y “R4”, correspondientes a 0,

150, 300 y 600 mm anuales. El diámetro de cada planta se evaluó al principio y al final del experimento (dos años más tarde). Nuestra pregunta es si el aporte de agua aumenta el crecimiento de las plántulas de pino.

Primero vamos a explorar nuestros datos, comprobando el valor medio y la varianza para cada tratamiento y evaluando la homogeneidad de varianzas mediante el test de Fligner y el de Levene (este último está en la librería car):

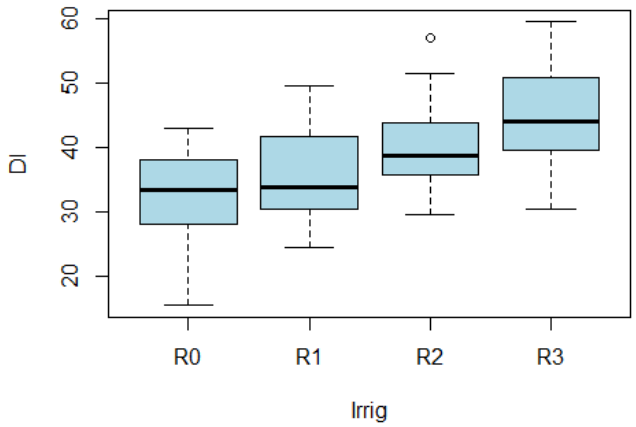
```
> tapply(DI,Irrig,mean)
      R0       R1       R2       R3
32.24457 35.40298 39.67361 44.92013
> tapply(DI,Irrig,var)
      R0       R1       R2       R3
49.99066 49.73167 39.45638 53.52958
> fligner.test(DI~Irrig)

        Fligner-Killeen test of homogeneity of variances

data:  DI by Irrig
Fligner-Killeen:med chi-squared = 1.8127, df = 3, p-value = 0.6122
> leveneTest(DI~Irrig)
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   3   0.647 0.5861
      140
```

Veamos gráficamente el valor de la mediana y la varianza para cada uno de nuestros tratamientos:

```
> plot(DI~Irrig,col="light blue")
```



Como vemos, parece que existe un efecto del riego sobre el crecimiento en diámetro de los pinos. Como comentamos anteriormente el ANOVA es un caso especial de modelo lineal, de

hecho la función que utilizaremos en R es la misma que para ajustar un modelo de regresión (lm).

```
> modelo1 <- lm(DI~Irrig, data=pinos)
> summary(modelo1)

Call:
lm(formula = DI ~ Irrig, data = pinos)

Residuals:
     Min       1Q   Median       3Q      Max
-16.8187  -4.3861  -0.7638   5.5471  17.3473

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   32.245      1.157  27.873  < 2e-16 ***
IrrigR1        3.158      1.636   1.931   0.0556 .
IrrigR2        7.429      1.636   4.541 1.20e-05 ***
IrrigR3       12.676      1.636   7.748 1.72e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.941 on 140 degrees of freedom
Multiple R-squared:  0.3258,   Adjusted R-squared:  0.3114
F-statistic: 22.55 on 3 and 140 DF,  p-value: 5.682e-12
```

Sabemos de la unidad anterior, que en un modelo de regresión lineal existen dos parámetros a estimar, el intercepto (β_0) y la pendiente que multiplica al valor de la variable independiente (β_1) tal que:

$$Y = \beta_0 + \beta_1 X$$

En este ejemplo tenemos una sola variable independiente (el riego), sin embargo el output del modelo ajustado nos da el valor del intercepto y de tres parámetros más. ¿Cómo interpreto esto? El porqué de este ajuste reside en la naturaleza categórica de la variable independiente. En estos casos el modelo ajusta tantas pendientes como *n – 1* niveles del factor existan, en nuestro ejemplo tenemos 4 niveles de riego por lo que el modelo ajusta 3 pendientes más un intercepto. Este intercepto (β_0) toma el valor de la variable respuesta para el nivel de referencia del factor y que R lo identifica por orden alfabético. Es decir, para nuestro ejemplo el valor del intercepto es igual al valor medio del crecimiento de las plantas para el tratamiento R0. De este modo, el valor de los parámetros estimado para los niveles R1, R2 y R3 (β_1, β_2 y β_3, respectivamente) nos indica la diferencia que existe entre el valor medio de la variable respuesta para dichos niveles y el del nivel de referencia, tal que:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$

De manera que para ajustar el modelo, el factor es dividido en tantas variables como niveles haya para dicho factor, de forma que dichas variables tomaran valores de 0 ó 1 dependiendo de si el individuo en cuestión pertenece o no a ese nivel. Llevándolo a nuestro ejemplo

$$Crecimiento = \beta_0 + \beta_1 dR1 + \beta_2 dR2 + \beta_3 dR3$$

Donde *dR1*, *dR2* y *dR3* son variables *dummies* cuyo valor depende de si el pino en cuestión pertenece a ese nivel o no. Por ejemplo, un pino perteneciente al tratamiento R0 tendrá unos valores para las variables *dummies* de 0,0,0 (*dR1*, *dR2* y *dR3*) y su crecimiento será igual a β_0 (32.245 mm/year), mientras que un pino perteneciente al tratamiento R2 tendrá un valor de dichas variables *dummies* de 0,1,0 y su crecimiento será igual a $\beta_0 + \beta_2$ (32.245 + 7.429 mm/year). Es posible que nos interese que nuestro nivel de referencia sea otro distinto al que R selecciona automáticamente, por ejemplo para visualizar mejor la explicación anterior vamos a ver qué pasa si seleccionamos al nivel de mayor riego (R3) como nivel de referencia:

```
> levels(Irrig)
[1] "R0" "R1" "R2" "R3"
> pinos$Irrig2 <- relevel(Irrig, ref="R3")
> attach(pinos)
> levels(Irrig2)
[1] "R3" "R0" "R1" "R2"
> levels(Irrig)
[1] "R0" "R1" "R2" "R3"
> modelo2 <- lm(DI~Irrig2, data=pinos)
> summary(modelo2)

Call:
lm(formula = DI ~ Irrig2, data = pinos)

Residuals:
     Min       1Q   Median       3Q      Max
-16.8187  -4.3861  -0.7638   5.5471  17.3473

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   44.920      1.157  38.830  < 2e-16 ***
Irrig2R0     -12.676      1.636  -7.748 1.72e-12 ***
Irrig2R1      -9.517      1.636  -5.817 3.89e-08 ***
Irrig2R2      -5.247      1.636  -3.207  0.00166 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.941 on 140 degrees of freedom
Multiple R-squared:  0.3258,   Adjusted R-squared:  0.3114
F-statistic: 22.55 on 3 and 140 DF,  p-value: 5.682e-12
```

Vemos cómo ahora los parámetros estimados son todos negativos ya que el nivel de referencia seleccionado es el que mayor crecimiento presenta por lo que para estimar el valor del resto de tratamientos habrá que restar una determinada cantidad al valor del nivel de referencia.

Sin embargo, desde el punto de vista de nuestra pregunta inicial estaríamos más interesados en conocer la significación de nuestro tratamiento riego en su conjunto, para ello necesitaríamos la tabla ANOVA que vimos en el primer apartado de esta unidad. A esta tabla se accede en R con la función *anova*:

```
> anova(modelo1)
Analysis of Variance Table

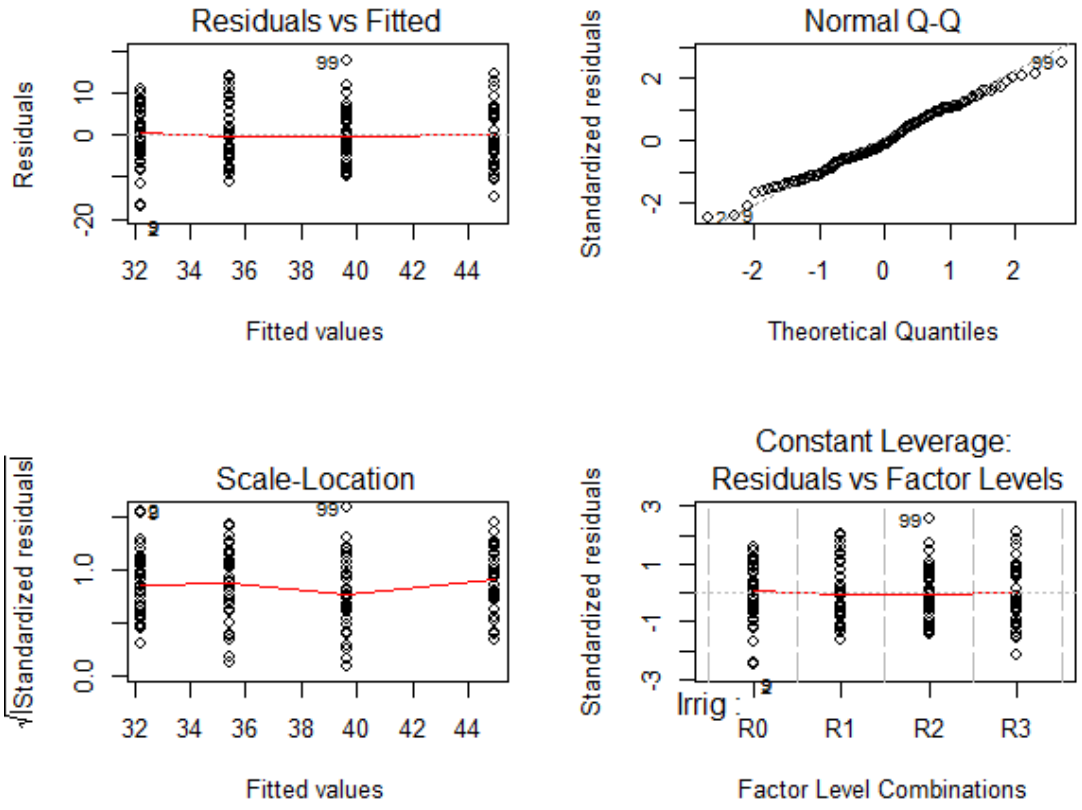
Response: DI
           Df Sum Sq Mean Sq F value    Pr(>F)
Irrig       3 3259.6 1086.53  22.553 5.682e-12 ***
```

```
Residuals 140 6744.8   48.18
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Esta tabla nos dice que efectivamente existe un efecto significativo del riego en el crecimiento en diámetro de los pinos, además podemos calcular cuanta variabilidad está siendo explicada por el riego simplemente calculando el porcentaje de la suma de cuadrados que representa nuestro factor con respecto al total y que en este caso es del 32.6 %, la misma cifra que el R^2 obtenido en el ajuste del modelo lineal.

El siguiente paso, al tratarse de un modelo lineal, es comprobar que se cumplen los supuestos necesarios para aplicar este tipo de modelos:

```
> x11()
> par(mfrow=c(2,2))
> plot(modelo1)
```



En este caso los dos primero gráficos son los más importantes, el primero relacionado con el supuesto de homogeneidad de varianza y el segundo con la normalidad en la distribución de los residuos. En el primero no encontramos ningún patrón en los residuos respecto a los valores ajustados y en el segundo vemos que nuestros datos se ajustan bien a una línea recta, aunque podemos comprobarlo con el test de Shapiro-Wilk:

```
> shapiro.test(residuals(modelo1))

        Shapiro-Wilk normality test
```

```
data:  residuals(modelo1)
W = 0.9907, p-value = 0.4572
```

Hemos visto que el factor riego tiene un efecto significativo en el crecimiento de nuestros pinos, pero ¿son significativamente distintos los diferentes niveles de riego entre sí? Si echamos un vistazo a los coeficientes que hemos ajustado sabemos que el crecimiento aumenta conforme aumenta el riego (todos los coeficientes son positivos), ahora vamos a comprobar si este aumento de un tratamiento a otro es significativo. Para ello podemos utilizar la significación de los parámetros, ya que estos expresan si la diferencia entre el nivel de referencia (intercepto) y un determinado nivel del factor es significativamente distinta de cero. También podemos utilizar un análisis Post-hoc habiendo muchos tests que pueden aplicarse. En este caso vamos a utilizar el t-test con el ajuste de Bonferroni. Otra opción es instalar el paquete de R "agricolae" que cuenta con procedimientos para implementar distintos test como el test de Scheffe, test LSD, Tukey HSD…

```
> pairwise.t.test(DI, Irrig, data=pinos, p.adjust.method="bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  DI and Irrig

   R0      R1      R2
R1 0.33    -       -
R2 7.2e-05 0.06    -
R3 1.0e-11 2.3e-07 0.01

P value adjustment method: bonferroni
```

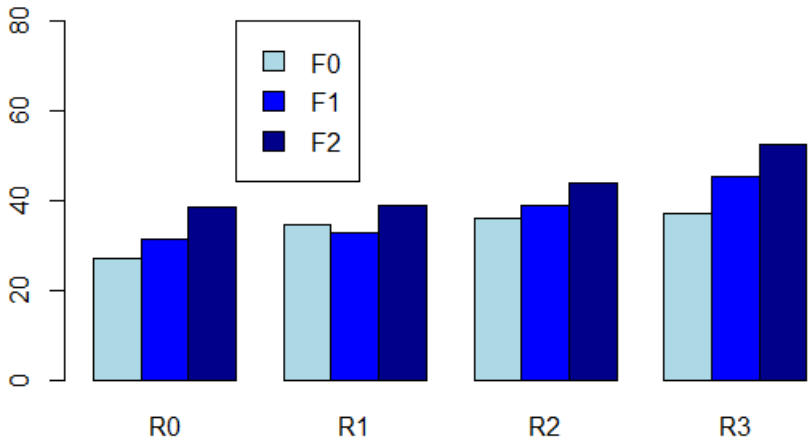
Como vemos aunque el factor riego es significativo no todas las comparaciones entre tratamientos lo son. En este caso el crecimiento de los pinos que recibieron la dosis mayor de riego fue significativamente superior al resto de tratamientos, pero en estos solo encontramos diferencias significativas ($p < 0.05$) entre el tratamiento R2 y el R0.

3. Diseños factoriales

Los experimentos factoriales son aquello en los que existen dos o más factores cada uno con dos o más niveles para dicho factor y en los que además los niveles de ambos factores se combinan. Es decir, podemos estudiar la interacción entre ambos factores o como la respuesta de nuestra variable dependiente está relacionada con la combinación de los factores de estudio. Al igual que en el apartado anterior utilizaremos un ejemplo para ilustrar este caso. Siguiendo con el caso de estudio de los pinos, ¿qué pasaría si combinara el riego con distintos tratamientos de fertilización? En dicho estudio las 144 plántulas de pino serían sometida a 4 niveles de riego (R0, R1, R2 y R3) y 3 niveles de fertilización nitrogenada (F0, F1 y F2) correspondientes a 0, 150 y 300 kg N/ha, respectivamente. Al igual que anteriormente vamos a inspeccionar los datos:

```
> barplot(tapply(DI,list(Fert,Irrig),mean),
+         beside=T,ylim=c(0,80),col=c("light blue","blue","dark blue")
)
> labs <- c("F0","F1","F2")
```

```
> legend(4,80,labs,fill= c("light blue","blue","dark blue"))
> tapply(DI,list(Fert,Irrig),mean)
         R0       R1       R2       R3
F0 27.00350 34.44215 36.19111 36.92910
F1 31.19367 32.85778 38.78382 45.44403
F2 38.53654 38.90902 44.04590 52.38725
```



Ahora, al igual que anteriormente ajustamos el modelo que satisfaga nuestra pregunta, en nuestro caso queremos conocer cómo influye la combinación de distintos tratamientos de riego y fertilización en el crecimiento en diámetro de las plántulas de pino, por tanto estaremos interesados no solo en el efecto de cada factor por separado sino también en el de su interacción.

```
> modelo3 <- lm(DI~Irrig*Fert, data=pinos)
> summary(modelo3)

Call:
lm(formula = DI ~ Irrig * Fert, data = pinos)

Residuals:
     Min       1Q   Median       3Q      Max
-12.1875  -3.7370   0.0644   3.5732  12.9750

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       27.004      1.563  17.282  < 2e-16 ***
IrrigR1            7.439      2.210   3.366 0.000998 ***
IrrigR2            9.188      2.210   4.158 5.75e-05 ***
IrrigR3            9.926      2.210   4.492 1.53e-05 ***
FertF1             4.190      2.210   1.896 0.060120 .
FertF2            11.533      2.210   5.219 6.80e-07 ***
IrrigR1:FertF1    -5.775      3.125  -1.848 0.066870 .
IrrigR2:FertF1    -1.597      3.125  -0.511 0.610086
IrrigR3:FertF1     4.325      3.125   1.384 0.168730
IrrigR1:FertF2    -7.066      3.125  -2.261 0.025389 *
IrrigR2:FertF2    -3.678      3.125  -1.177 0.241313
IrrigR3:FertF2     3.925      3.125   1.256 0.211335
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 5.413 on 132 degrees of freedom
Multiple R-squared:  0.6134,   Adjusted R-squared:  0.5812
F-statistic: 19.04 on 11 and 132 DF,  p-value: < 2.2e-16
```

Como vemos, este modelo es más complicado que el anterior. En este caso tenemos 12 parámetros, 1 correspondiente al intercepto, 3 a los niveles del factor riego, 2 a los niveles del factor fertilización y 6 correspondientes a la interacción de ambas factores. El valor del intercepto corresponde al crecimiento de aquellos pinos sometidos a la combinación de los tratamientos R0 y F0 (niveles de referencia). Por lo que si queremos conocer el crecimiento estimado de los pinos sometidos a la combinación R0-F2 tendríamos *DI = 27.004 + 11.533*, es decir al valor del intercepto (nivel de referencia) le sumaríamos el modificador del tratamiento F2. Por otro lado si queremos saber el valor estimado de crecimiento para la combinación R3-F2 entonces *DI = 27.004 + 9.926 + 11.533 + 3.925*, ya que al valor del nivel de referencia le sumaríamos el modificador del tratamiento R3 más el del F2 más el de la combinación R3-F2. Veamos ahora la significación de los factores considerando todos sus niveles:

```
> anova(modelo3)
Analysis of Variance Table

Response: DI
            Df Sum Sq Mean Sq F value    Pr(>F)
Irrig        3 3259.6 1086.53 37.0846 < 2.2e-16 ***
Fert         2 2388.9 1194.44 40.7679 1.632e-14 ***
Irrig:Fert   6  488.5   81.41  2.7788   0.01411 *
Residuals  132 3867.4   29.30
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Como vemos, ambos factores (riego y fertilización) además de su interacción han resultado significativos. Tenemos que tener cuidado con la interpretación del efecto de los factores cuando la interacción de ambos es significativa ya que puede llevarnos a conclusiones erróneas. Por ejemplo imaginemos que el efecto positivo sobre el crecimiento de someter a las plantas al mayor riego solo es significativos cuando las plantas son fertilizadas, en este caso tanto ambos factores como su interacción resultaran significativos en el modelo pero el efecto sobre el crecimiento no es debido al mayor riego sino a la combinación de este con la fertilización. Para ver esto realizaremos un análisis post-hoc pero nos centraremos en la comparación entre las distintas combinaciones de tratamientos utilizando el test de Tukey:

```
> TukeyHSD(aov(DI~Irrig*Fert, data=pinos))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = DI ~ Irrig * Fert, data = pinos)

$Irrig
           diff        lwr       upr     p adj
R1-R0  3.158414 -0.1613311  6.478159 0.0684849
R2-R0  7.429039  4.1092939 10.748784 0.0000002
R3-R0 12.675558  9.3558133 15.995303 0.0000000
R2-R1  4.270625  0.9508800  7.590370 0.0057939
```

```
R3-R1  9.517144  6.1973995 12.836889 0.0000000
R3-R2  5.246519  1.9267745  8.566264 0.0003948

$Fert
          diff       lwr       upr     p adj
F1-F0 3.428354 0.8092789  6.047429 0.0065878
F2-F0 9.828213 7.2091372 12.447288 0.0000000
F2-F1 6.399858 3.7807831  9.018934 0.0000001

$`Irrig:Fert`
                    diff         lwr        upr     p adj
R1:F0-R0:F0  7.4386458   0.0854419 14.7918498 0.0446996
R2:F0-R0:F0  9.1876042   1.8344002 16.5408081 0.0032267
R3:F0-R0:F0  9.9256000   2.5723961 17.2788039 0.0009021
R0:F1-R0:F0  4.1901625  -3.1630414 11.5433664 0.7594458
R1:F1-R0:F0  5.8542708  -1.4989331 13.2074748 0.2639112
R2:F1-R0:F0 11.7803125   4.4271086 19.1335164 0.0000260
R3:F1-R0:F0 18.4405208  11.0873169 25.7937248 0.0000000
R0:F2-R0:F0 11.5330333   4.1798294 18.8862373 0.0000428
…….
```

También podemos aplicar un t-test con el ajuste de Bonferroni para evaluar la significancia de las comparaciones múltiples entre todas las combinaciones posibles de riego y fertilización:

```
> pairwise.t.test(DI,Irrig:Fert,p.adjust.method="bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  DI and Irrig:Fert

        R0:F0   R0:F1   R0:F2   R1:F0   R1:F1   R1:F2   R2:F0   R2:F1   R2:F2   R3:F0   R3:F1
R0:F1 1.00000 -       -       -       -       -       -       -       -       -       -
R0:F2 4.5e-05 0.07609 -       -       -       -       -       -       -       -       -
R1:F0 0.06589 1.00000 1.00000 -       -       -       -       -       -       -       -
R1:F1 0.59741 1.00000 0.74480 1.00000 -       -       -       -       -       -       -
R1:F2 2.1e-05 0.04315 1.00000 1.00000 0.46376 -       -       -       -       -       -
R2:F0 0.00379 1.00000 1.00000 1.00000 1.00000 1.00000 -       -       -       -       -
R2:F1 2.7e-05 0.05233 1.00000 1.00000 0.54514 1.00000 1.00000 -       -       -       -
R2:F2 1.8e-10 2.8e-06 0.91722 0.00181 9.0e-05 1.00000 0.03472 1.00000 -       -       -
R3:F0 0.00101 0.69397 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 0.10631 -       -
R3:F1 5.5e-12 1.3e-07 0.14392 0.00013 5.0e-06 0.24275 0.00338 0.20410 1.00000 0.01195 -
R3:F2 < 2e-16 4.9e-15 3.2e-07 1.9e-11 3.5e-13 7.3e-07 1.4e-09 5.5e-07 0.01591 7.9e-09 0.13673

P value adjustment method: bonferroni
```

4. ANCOVA: Mezclando variables independientes categóricas y continuas

El modelo ANCOVA (ANalysis of COVAriance) es un modelo lineal en el que la variable respuesta es continua y existe, al menos, una variable independiente categórica y una continua. El procedimiento que se sigue consta de varios pasos, en primer lugar se ajusta un modelo lineal entre las dos variables continuas para cada uno de los niveles del factor (variable categórica), posteriormente se estima el valor del intercepto y la pendiente para cada nivel del factor y finalmente se simplifica el modelo eliminando aquellos parámetros innecesarios. Una vez hemos comprendido cómo funcionan las regresiones lineales y los modelos ANOVA entender cómo se ajusta un modelo ANCOVA es inmediato. Para ilustrar este procedimiento vamos a trabajar

sobre un ejemplo práctico. Los datos de este experimento fueron tomados en una dehesa de encinas donde 20 individuos al azar fueron tomados como control y otros 20 fueron fertilizados con una solución completa de macro y micronutrientes. Al cabo de 3 años de experimento se evaluó la producción de bellota de cada uno de los individuos al final del tercer año, así como el incremento en diámetro de dichos individuos durante el periodo estudiado.

```
> encinas <- read.table("bellotas.csv", header=T, sep=",")
> str(encinas)
'data.frame':  40 obs. of  4 variables:
 $ DI         : num  6.22 6.49 4.92 5.13 5.42 ...
 $ Acorn      : num  59.8 61 14.7 19.3 34.2 ...
 $ Fert       : Factor w/ 2 levels "Control","Fertilized": 1 1 1 1 1 1
1 1 1 1 ...
 $ Competition: Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 1 ...
> attach(encinas)
```

Vemos que existe otro factor, “competition” el cual ignoraremos por el momento. Lo que queremos investigar con este experimento es si la producción de bellota depende del crecimiento, y además si la fertilización influye en la producción de bellota. Por lo tanto, nuestro modelo teórico sería:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

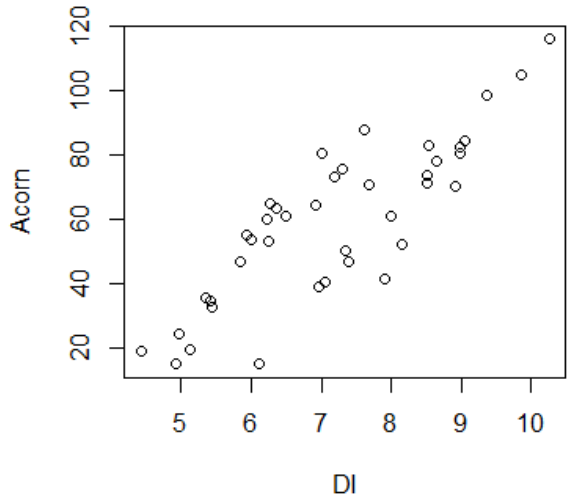
Donde X_1 y X_2 serían las variables independientes (crecimiento y fertilización). Aplicando lo que ya sabemos de los apartados anteriores y sabiendo que el factor fertilización solo tiene dos niveles (control y fertilized), nuestro modelo teórico pasaría a ser del tipo:

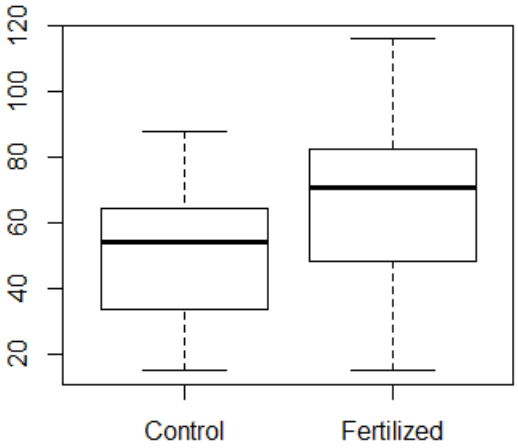
$$Produccion = \beta_0 + \beta_1 Crecimiento + \beta_2 dFertilized$$

Es decir, tenemos tres parámetros a estimar aunque en este caso hay ciertas diferencias con los parámetros estimados para el ANOVA. Primero tenemos que considerar que la pendiente para la relación de la producción de bellota con el crecimiento (β_1) se ajusta considerando los datos como un todo, es decir sin distinguir por niveles del factor, por lo tanto si queremos calcular el valor de producción de bellota estimado para un nivel determinado habrá que multiplicar el parámetro pendiente por el valor medio de la variable cuantitativa independiente para ese tratamiento (el crecimiento en este caso). Por lo tanto el valor del intercepto (β_0) habrá que sumarlo al resultado de esta multiplicación para estimar el valor medio de producción de bellota de las plantas control (i.e. no fertilizadas) y en el caso que estemos interesados en calcular que pasa con el otro nivel tendremos pues, que sumar también el valor de (β_1). Veamos gráficamente como varia nuestra variable respuesta en función de las variables independientes consideradas y ajustemos el modelo ANCOVA:

```
> plot(Acorn ~ DI, data = encinas)
```

```
> boxplot(Acorn ~ Fert, data = encinas)
```






```
> tapply(Acorn, Fert, mean)
   Control Fertilized
   50.8805    67.9405
> tapply(DI, Fert, mean)
   Control Fertilized
    6.0529     8.3094
```

Parece ser que existe una relación positiva entre producción de bellotas y crecimiento y además la producción aumenta con la fertilización.

```
> ancova1 <- lm(Acorn ~ Fert + DI, data=encinas)
> summary(ancova1)

Call:
lm(formula = Acorn ~ Fert + DI, data = encinas)

Residuals:
     Min       1Q   Median       3Q      Max
-17.1920  -2.8224   0.3223   3.9144  17.3290

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     -91.726      7.115  -12.89 2.96e-15 ***
FertFertilized  -36.103      3.357  -10.75 6.11e-13 ***
DI               23.560      1.149   20.51  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.747 on 37 degrees of freedom
Multiple R-squared:  0.9291,   Adjusted R-squared:  0.9252
F-statistic: 242.3 on 2 and 37 DF,  p-value: < 2.2e-16
```

Así pues el valor estimado de producción de bellota de una encina fertilizada sería *-91.726 + (23.560 x 8.3094) + (-36.103 x 1) = 67.94*. En el caso de una encina no fertilizada sería *-91.726 + (23.560 x 6.0529) + (-36.103 x 0) = 50.88*.

Podríamos plantearnos si existe interacción entre nuestra variable independiente cuantitativa y la variable independiente categórica. Para nuestro caso particular esto significaría que la relación entre producción de bellota y crecimiento variaría en función de la fertilización recibida. Siguiendo con nuestro ejemplo el modelo con la interacción de ambos factores sería el modelo más complejo, de hecho para ser estrictos deberíamos haber empezado por ajustar este modelo e ir simplificando.

```
> ancova2 <- lm(Acorn ~ Fert * DI, data=encinas)
> summary(ancova2)

Call:
lm(formula = Acorn ~ Fert * DI, data = encinas)

Residuals:
     Min       1Q   Median       3Q      Max
-17.3177  -2.8320   0.1247   3.8511  17.1313

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        -94.367     10.933  -8.632 2.71e-10 ***
FertFertilized     -30.806     16.842  -1.829   0.0757 .
DI                  23.996      1.788  13.417 1.41e-15 ***
FertFertilized:DI   -0.756      2.354  -0.321   0.7500
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.831 on 36 degrees of freedom
Multiple R-squared:  0.9293,   Adjusted R-squared:  0.9234
F-statistic: 157.6 on 3 and 36 DF,  p-value: < 2.2e-16

> anova(ancova2)
Analysis of Variance Table

Response: Acorn
          Df  Sum Sq Mean Sq  F value    Pr(>F)
Fert       1  2910.4  2910.4  62.3795 2.262e-09 ***
DI         1 19148.9 19148.9 410.4201 < 2.2e-16 ***
Fert:DI    1     4.8     4.8   0.1031      0.75
Residuals 36  1679.6    46.7
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

La significación de los parámetros del modelo así como de los distintos factores en la tabla ANOVA parece indicar que el modelo con la interacción no mejora al modelo con solo los efectos principales. Para comprobarlo podemos realizar un ANOVA de ambos modelos y comprobar si la eliminación de la interacción empeora de manera significativa las estimaciones del modelo:

```
> anova(ancova2, ancova1)
Analysis of Variance Table

Model 1: Acorn ~ Fert * DI
```

```
Model 2: Acorn ~ Fert + DI
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     36 1679.7
2     37 1684.5 -1   -4.8122 0.1031   0.75
```

Vemos que la eliminación de la interacción no empeora significativamente (p = 0.75) el potencial explicativo del modelo y además ganamos un grado de libertad. Podemos plantearnos simplificar aún más el modelo, por ejemplo eliminando el factor (fertilización)

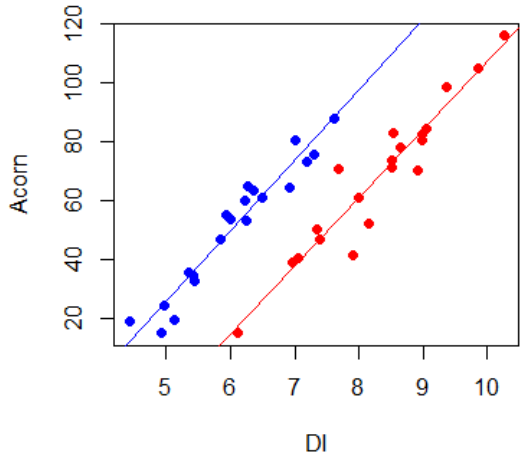
```
 > anova(ancova1, ancova3)
Analysis of Variance Table

Model 1: Acorn ~ Fert + DI
Model 2: Acorn ~ DI
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     37 1684.5
2     38 6948.8 -1   -5264.4 115.63 6.107e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

En este caso si vemos que el modelo empeora significativamente, por tanto nos quedamos con el modelo con solo los factores principales, siendo ambos significativos. Como veremos gráficamente la relación entre producción de bellota y crecimiento no cambia con la dosis de fertilizante recibida.

```
> anova(ancova1)
Analysis of Variance Table

Response: Acorn
          Df  Sum Sq Mean Sq F value    Pr(>F)
Fert       1  2910.4  2910.4  63.929 1.397e-09 ***
DI         1 19148.9 19148.9 420.616 < 2.2e-16 ***
Residuals 37  1684.5    45.5
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> plot(DI,Acorn,pch=16,col=c("blue","red")[as.numeric(Fert)])
> abline(lm(Acorn[Fert=="Control"]~DI[Fert=="Control"]),col="blue")
> abline(lm(Acorn[Fert=="Fertilized"]~DI[Fert=="Fertilized"]),col="red
")
```



Una forma de automatizar este proceso es mediante la función *step*, la cual se aplica sobre el modelo más complejo y utiliza el criterio de información de Akaike (AIC) para seleccionar el mejor modelo:

```
> step(ancova2)
Start:  AIC=157.5
Acorn ~ Fert * DI

          Df Sum of Sq    RSS    AIC
- Fert:DI  1    4.8122 1684.5 155.61
<none>                 1679.7 157.50

Step:  AIC=155.61
Acorn ~ Fert + DI

       Df Sum of Sq     RSS    AIC
<none>               1684.5 155.61
- Fert  1    5264.4  6948.8 210.30
- DI    1   19148.9 20833.4 254.22

Call:
lm(formula = Acorn ~ Fert + DI, data = encinas)

Coefficients:
   (Intercept)  FertFertilized             DI
         -91.73          -36.10          23.56
```

Vamos a ver un ejemplo en el que la interacción entre la variable independiente categórica y continua es significativa. En nuestro estudio con las encinas introducimos un nuevo factor o tratamiento, la competencia. Estamos ante el mismo estudio que el anterior pero para cada uno de los niveles de fertilización (asignado a 20 encinas), en 10 de ellas se eliminó la competencia de herbáceas y matorral mientras que en el resto (las otras 10) no se llevó a cabo ninguna acción. Vamos pues a explorar gráficamente nuestros datos, ajustar el modelo más complejo, seleccionar el mejor modelo y ver algunos resultados interesantes:

```
> plot(Acorn ~ DI, data = encinas)
```

```
> boxplot(Acorn ~ Competition, data = encinas)
> tapply(Acorn, Competition, mean)
      no      yes
58.12111 60.46545
> tapply(DI, Competition, mean)
      no      yes
7.110444 7.239000
> ancova4 <- lm(Acorn ~ Fert * Competition * DI, data=encinas)
> summary(ancova4)

Call:
lm(formula = Acorn ~ Fert * Competition * DI, data = encinas)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1976  -3.3404   0.0833   2.5343  18.9454

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      -57.177     29.776  -1.920 0.063781
.
FertFertilized                     9.017     52.506   0.172 0.864724
Competitionyes                   -52.313     38.206  -1.369 0.180460
DI                                18.563      4.393   4.225 0.000185
***
FertFertilized:Competitionyes     -31.326     60.374  -0.519 0.607422
FertFertilized:DI                  -5.571      7.203  -0.774 0.444882
Competitionyes:DI                   8.178      6.261   1.306 0.200813
FertFertilized:Competitionyes:DI    2.864      8.706   0.329 0.744312
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.696 on 32 degrees of freedom
Multiple R-squared:  0.9396,   Adjusted R-squared:  0.9264
F-statistic: 71.09 on 7 and 32 DF,  p-value: < 2.2e-16

> anova(ancova4)
Analysis of Variance Table

Response: Acorn
                   Df  Sum Sq Mean Sq  F value    Pr(>F)
Fert                1  2910.4  2910.4  64.9187 3.363e-09 ***
Competition         1   103.3   103.3   2.3047    0.1388
DI                  1 19048.5 19048.5 424.8857 < 2.2e-16 ***
Fert:Competition    1    20.9    20.9   0.4665    0.4995
Fert:DI             1     0.2     0.2   0.0040    0.9499
Competition:DI      1   221.0   221.0   4.9298    0.0336 *
Fert:Competition:DI 1     4.9     4.9   0.1082    0.7443
Residuals          32  1434.6    44.8
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> step(ancova4)
Start:  AIC=159.19
Acorn ~ Fert * Competition * DI

                     Df Sum of Sq    RSS    AIC
- Fert:Competition:DI  1    4.8522 1439.5 157.33
<none>                             1434.6 159.19

Step:  AIC=157.33
Acorn ~ Fert + Competition + DI + Fert:Competition + Fert:DI +
    Competition:DI
```

```
                   Df Sum of Sq    RSS    AIC
- Fert:DI           1    35.714 1475.2 156.31
- Fert:Competition  1    64.519 1504.0 157.08
<none>                          1439.5 157.33
- Competition:DI    1   221.015 1660.5 161.04

Step:  AIC=156.31
Acorn ~ Fert + Competition + DI + Fert:Competition + Competition:DI

                   Df Sum of Sq    RSS    AIC
- Fert:Competition  1    32.152 1507.3 155.17
<none>                          1475.2 156.31
- Competition:DI    1   185.481 1660.7 159.04

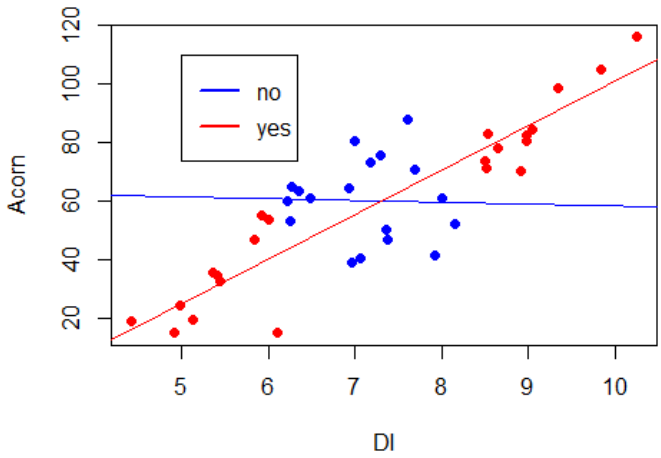
Step:  AIC=155.17
Acorn ~ Fert + Competition + DI + Competition:DI

                 Df Sum of Sq    RSS    AIC
<none>                        1507.3 155.17
- Competition:DI  1     174.2 1681.6 157.54
- Fert            1    3808.5 5315.8 203.58

Call:
lm(formula = Acorn ~ Fert + Competition + DI + Competition:DI,
    data = encinas)

Coefficients:
      (Intercept)     FertFertilized     Competitionyes
DI  Competitionyes:DI
          -49.566            -33.334            -41.915             17
.506              5.778
> ancova5 <- lm(Acorn ~ Fert + Competition + DI + Competition:DI, data
=encinas)
> anova(ancova5)
Analysis of Variance Table

Response: Acorn
               Df  Sum Sq Mean Sq  F value    Pr(>F)
Fert            1  2910.4  2910.4  67.5794 1.094e-09 ***
Competition     1   103.3   103.3   2.3992   0.13040
DI              1 19048.5 19048.5 442.2996 < 2.2e-16 ***
Competition:DI  1   174.2   174.2   4.0459   0.05203 .
Residuals      35  1507.3    43.1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> plot(DI,Acorn,pch=16,col=c("blue","red")[as.numeric(Competition)])
> abline(lm(Acorn[Competition=="no"]~DI[Competition=="no"]),col="blue"
)
> abline(lm(Acorn[Competition=="yes"]~DI[Competition=="yes"]),col="red
")
> legend(5,110,c("no","yes"),lty=c(1,1),lwd=c(2.5,2.5),col=c("blue","r
ed"))
```



En este caso observamos que el sentido de la relación entre la producción de bellota y el crecimiento va a depender de si controlamos la competencia alrededor de las encinas.

5. Lecturas recomendadas

i. Cayuela L (2015) Modelos lineales: regresión, ANOVA y ANCOVA.

ii. Crawley MJ (2013) The R Book. Second edition. Wiley.

iii. Quinn GP, Keough MJ (2002) Experimental design and data analysis for biologists. Cambridge University Press.

Modelos lineales generalizados y mixtos

Análisis de datos y modelos lineales en R

Enrique Andivia[1]

Master Universitario en Restauración de Ecosistemas
Curso 2019-2020

1 Departamento de Biodiversidad, Ecología y Evolución. Universidad Complutense de Madrid

email: e.andivia@gmail.com

web: http://enriqueandivia.wixsite.com/forests

MODELOS LINEALES GENERALIZADOS Y MIXTOS

1. Conceptos básicos sobre los modelos lineales generalizados

1.1. Necesidad de generalizar los modelos lineales

Como hemos visto en los módulos anteriores, los modelos lineales (regresión, ANOVA o ANCOVA) tienen una serie de suposiciones cuya violación implica cometer errores estadísticos más o menos graves que pueden dar lugar a la malinterpretación de nuestros resultados. Sin embargo los datos recogidos en la naturaleza generalmente violan algunos de los supuestos de normalidad, homocedasticidad o linealidad. Problemas comunes en ecología, como el aumento de la varianza conforme aumenta la media se han solucionado normalmente mediante la transformación de la variable respuesta. Sin embargo ya conocemos los *pros* y *contras* de estas transformaciones y la dificultad de interpretar las predicciones de estos modelos. Una alternativa a la transformación de la variable respuesta y a la falta de normalidad es el uso de los modelos lineales generalizados. Estos modelos, conocidos como GLM (Generalized Linear Models) son una extensión de los modelos lineales que permiten utilizar distribuciones no normales de los errores y varianzas no constantes.

El uso de los GLM es directamente considerado cuando trabajamos con ciertos tipos de variables respuesta que violan sistemáticamente los supuestos de normalidad y homocedasticidad requeridos por los modelos lineales. Por ejemplo en aquellas variables que son un conteo (p.ej. abundancia de especies), proporciones (p.ej. porcentaje de supervivencia) o bien respuestas de tipo binaria (p.ej. presencia o ausencia, vivo o muerto). Este tipo de variable viola el supuesto de los modelos lineales sobre varianza constante (Figura 1), en el caso de los conteos la varianza tiende a aumentar linealmente con la media debido a la presencia frecuente de muchos ceros. En el caso de las proporciones esta relación suele tener forma de U invertida mientras que para otras variables la relación entre la varianza y la media aumenta de forma no lineal.

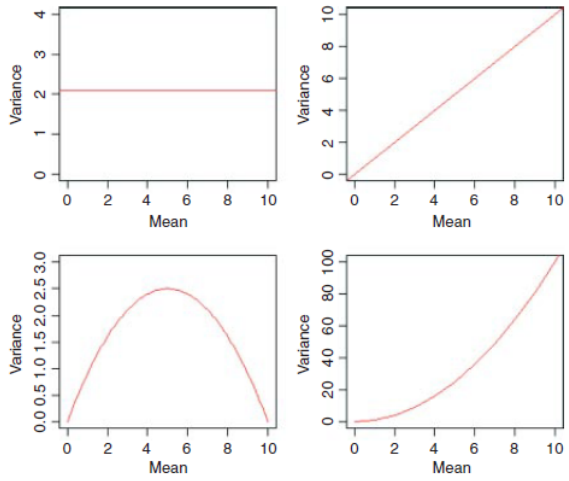


Figura 1: Distintas relaciones entre la varianza y la media de las observaciones para la variable respuesta: constante (arriba izq.), aumento lineal (arriba der.), U invertida (abajo izq.) y no lineal (abajo der.).

1.2. Las estructura de errores

Los modelos lineales generalizados permiten tener en cuenta estructuras de errores distintas de la distribución normal de los mismos. Esto es muy útil ya que la distribución normal es una distribución continua, sin embargo muchas de las variables respuestas utilizadas en ecología pueden ser categóricas (p.ej. conteo de especies), por lo que no tiene sentido utilizar una distribución continua para los residuos del modelo ajustado. Las principales estructuras de errores son: Poisson, binomial, gamma y exponencial. Cada una de estas estructuras tiene una serie de características que las hace más adecuadas para un determinado tipo de variable respuesta. La estructura de error de Poisson está especialmente indicada para su uso en el análisis estadístico de variables respuestas relacionadas con el conteo (abundancia de individuos, riqueza de especies, número de crias, etc). La estructura de error binomial se usa para el análisis de proporciones (tasa de mortalidad, porcentaje de infección, etc) o para respuestas binarias (presencia/ausencia de especie, vivo/muerto, etc), mientras que la Gamma está especialmente indicada para su uso cuando la varianza aumenta de manera constante según aumenta la media de la muestra. Finalmente la estructura exponencial suele usarse en análisis de supervivencia.

Para saber si debemos aplicar un modelo lineal clásico o un GLM es importante conocer, en primer lugar, el tipo de variable respuesta y su naturaleza, y además realizar un análisis apropiado de los residuos del modelo ajustado. Esto nos va a permitir evaluar cómo se comportan nuestros residuos respecto de la normalidad y saber cuándo es conveniente utilizar uno u otro tipo de distribución de error.

1.3. Las funciones de enlace (link function)

Hasta ahora hemos visto que los modelos lineales generalizados son adecuados cuando la estructura de errores no sigue una distribución normal o cuando se viola el supuesto de homogeneidad de varianzas. Otra utilidad de los GLM es cuando la relación entre la variable respuesta y las variables independientes no es lineal. En este contexto la función de vínculo o enlace es aquella que describe la relación entre la variable respuesta y la variable independiente encargándose de linealizar dicha relación, de modo que si la relación entre ambas es logarítmica, tendré que usar una función de vínculo de este tipo. Este hecho supone que los valores estimados por el modelo no son los valores de la variable respuesta si no de su logaritmo (siguiendo con el ejemplo de relación logarítmica), de modo que si queremos conocer los valores reales estimados por el modelo debemos utilizar la función inversa a la función de enlace para transformarlos (en este caso la exponencial). Básicamente, estamos realizando un procedimiento análogo a transformar nuestra variable respuesta, tal y como se suele hacer cuando ajustamos un modelo de regresión. Sin embargo los GLM automáticamente devuelven los valores estimados por el modelo mediante la transformación inversa de la función de enlace por lo que podemos interpretar directamente dichos valores estimados. Las funciones de enlace más comunes así como su utilidad están recogidas en la tabla 1.

Función de enlace	Fórmula	Uso
Identidad	μ	Errores normales
Logarítmica	$\log(\mu)$	Errores tipo Poisson
Logit	$\log(\frac{\mu}{n-\mu})$	Errores binomiales (proporciones)
Recíproca	$\frac{1}{\mu}$	Errores Gamma (datos continuos)
Raíz cuadrada	$\sqrt{\mu}$	Conteos
Exponencial	μ^n	Funciones de potencia

Tabla 1: Funciones de enlace más comunes y uso de las mismas

Como vemos existen unos usos muy específicos para las funciones de enlaces, es por ello que algunas de ellas se usan por defecto para determinadas distribuciones de errores. A estas funciones se las denomina funciones de enlace canónicas. Sin embargo aunque estas se aplican por defecto debemos probar distintas funciones de enlace y comprobar cuál de ellas se ajusta mejor a nuestro caso específico. El uso de estas funciones viene determinado en muchas ocasiones por los límites propios de nuestra variable dependiente o respuesta. De hecho una de las mayores utilidades de este tipo de funciones es acotar las predicciones de nuestro modelo. Así si nuestra variable respuesta es un conteo estaremos interesados en acotar nuestras predicciones a valores positivo por lo que utilizaremos una función de enlace logarítmica. Por otro lado si trabajamos con proporciones acotaremos nuestras predicciones entre 0 y 1 (o 0 y 100) mediante el uso de una función logit.

2- Ajuste, evaluación y validación de GLMs

Cuando trabajamos con modelos lineales generalizados tenemos que tener en cuenta que no existe un solo modelo válido, sino un conjunto de modelos del cual debemos seleccionar aquel más plausible y a la vez más sencillo. Es por ello que cuando utilizamos este tipo de modelos los criterios de información son fundamentales. El objetivo será seleccionar aquel modelo que explique la mayor proporción de la variabilidad y en el que todos los parámetros que contiene sean estadísticamente significativos.

Como en todo análisis de datos, comenzaremos con un análisis exploratorio de los mismos donde seamos capaces de identificar la relación existente entre nuestra variable respuesta y cada una de las variables explicativas (categóricas o continuas) consideradas. Además no debemos olvidar evaluar la colinealidad existentes entre las variables explicativas, eliminando del análisis aquellas potencialmente problemáticas por su alta correlación con otras covariables. El siguiente paso es la elección de la estructura de errores y de la función de enlace a utilizar en nuestro modelo. Como hemos comentado, la naturaleza de nuestra variable respuesta determinará la estructura de errores y la función de enlace a utilizar, sin embargo en otros casos esta elección no es tan inmediata por lo que se recomienda evaluar los residuos para comprobar

si estamos utilizando la estructura correcta para los mismos, además de utilizar distintas funciones de enlace para comprobar cuál de ellas se ajusta mejor a nuestros datos.

Una vez seleccionado el tipo de estructura de error, la función de enlace y la estructura de las variables explicativas debemos evaluar el grado de ajuste de nuestro modelo a los datos. En modelos de regresión lineal usábamos el valor de R^2 para conocer qué proporción de la varianza era explicada por nuestro modelo. En GLMs utilizamos la devianza para obtener esta información. Este cambio en la forma de evaluar el ajuste de nuestro modelo a los datos se debe al método utilizado para estimar los parámetros del modelo. En regresión lineal se utiliza el método de mínimos cuadrados que es un caso específico del método de máxima verosimilitud utilizado en GLMs. De este modo calculamos la devianza nula y la devianza residual que son los equivalentes para la máxima verosimilitud de la suma de cuadrados total y residual, respectivamente. La devianza nula es la devianza del modelo nulo, que es aquel que contiene solo el intercepto y que por lo tanto es el modelo mínimo que se puede ajustar a un set de datos. La devianza residual es aquella variabilidad que no es explicada por el modelo que se está evaluando. De este modo, la devianza explicada por el modelo (D^2) es:

$$D^2 = \frac{devianza\ nula - devianza\ residual}{devianza\ nula} x100$$

Al igual que con los modelos de regresión lineal, debemos usar los residuos del modelo ajustado en la validación del mismo. El problema es que debido a que la estructura de errores no es normal, los residuos ordinarios (valores observados – valores estimados) muestran heterogeneidad por lo que son menos informativos. Una solución es usar los residuos estandarizados de Pearson que descuentan el efecto del valor medio sobre los residuos y los cuales se calculan de forma distinta para cada distribución de error. Otro tipo de residuos son los residuos de la devianza, los cuales son usados por defecto por muchas funciones de R. Los residuos de la devianza cuantifican la contribución de cada observación a la devianza residual. Independientemente de los residuos que utilicemos lo que debemos evaluar es la ausencia de ajuste del modelo a nuestros datos, es por ello que buscaremos tendencias cuando representemos los residuos frente a los valores ajustados y frente a cada una de las covariables incluidas y no incluidas en el modelo. En el caso de los GLMs no es necesario evaluar la normalidad de los residuos ya que estamos considerando otras estructuras de errores distintas.

Un aspecto a tener en cuenta en los GLMs es la presencia de sobredispersión (overdispersion). Esto ocurre cuando la variación en los datos excede la variabilidad esperada basada en la distribución de error esperada. Esto suele ocurrir cuando trabajamos con distribuciones Poisson o binomiales. Esto podría dar lugar a que la devianza esté inflada y a los problemas derivados de este aspecto. Una de las principales razones que dan lugar a sobredispersión es que no hayamos considerado una o más covariables o interacciones importantes, o bien no hayamos seleccionamos correctamente nuestra distribución de errores o función de enlace.

2.1. Especificación en R

Para ajustar un modelo lineal generalizado en R utilizaremos la función *glm*. Esta función irá acompañada por la formulación de nuestro modelo, donde aparezca la variable dependiente y la/s independiente/s de acuerdo a la estructura que ya hemos aplicado para los modelos lineales de regresión. Además de esto, debemos incluir un argumento para especificar la estructura de errores que utilizaremos, dicho argumento es *family* e irá acompañado por la función de enlace mediante el argumento *link*. Estas son las posibilidades existentes para ambos argumentos más usadas:

- *gaussian (link="identity")*

- *binomial (link="logit")*

- *poisson (link="log")*

- *Gamma (link="inverse")*

Cabe mencionar que en caso de no especificar el argumento *link*, R utilizará la función de enlace canónica para esa distribución de error, que es la que hemos especificado en cada caso. Existen otras familias como *quasibinomial* o *quasipoisson* muy útiles en el caso de que nuestros datos tengan sobredispersión.

3. GLM con datos de conteo (Poisson)

Vamos a ajustar un modelo lineal generalizado con una estructura de error de Poisson a unos datos de ejemplo en los que se estudia el efecto de la composición forestal sobre la abundancia de ardilla roja en Escocia, estimada a partir del número de piñas roídas. Además del número de piñas (SqCones), se midieron las siguientes variables relacionadas con la estructura forestal: número de árboles (Ntrees), altura media de los árboles (TreeHeight), diámetro medio de los árboles (DBH) y cobertura de copas (CanopyCover). Este ejemplo (RedSquirrels.txt) ha sido tomado de Zuur et al. (2013). Lo primero que haremos será comprobar si existen outliers y si hay colinealidad entre nuestras variables, en este caso tenemos un outlier en el DBH y la altura del árbol y su diámetro son candidatos a presentar colinealidad.

```
> SQ <- read.table (file="RedSquirrels.txt", header=T)
> str(SQ)
'data.frame':  52 obs. of  6 variables:
 $ Id         : Factor w/ 52 levels "Abern1","Abern10",..: 1 12 23 27
28 29 30 31 32 2 ...
 $ SqCones    : int  61 4 15 9 42 4 12 27 0 4 ...
 $ Ntrees     : int  32 4 34 22 22 21 19 15 12 9 ...
 $ DBH        : num  0.23 0.27 0.17 0.23 0.18 0.23 0.22 0.26 0.23 0.12
...
 $ TreeHeight : num  20.4 15.2 16 22.4 19.4 ...
 $ CanopyCover: num  91.3 61.5 91.4 92 93.2 93.5 88.5 88 89.8 73.3 ...
> boxplot(SQ$DBH)
> SQ <- subset(SQ, SQ$DBH < 0.7)
> plot(SQ$DBH,SQ$TreeHeight,xlab="Tree Height",ylab="DBH")
> cor.test(SQ$DBH,SQ$TreeHeight)
```

```
        Pearson's product-moment correlation

data:  SQ$DBH and SQ$TreeHeight
t = 5.2536, df = 49, p-value = 3.224e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3890319 0.7515257
sample estimates:
     cor
0.600261
```

```
> corvif(SQ[,c(3:6)])


Variance inflation factors

                GVIF
Ntrees      1.977228
DBH         2.855581
TreeHeight  1.884290
CanopyCover 1.223123
```

Como vemos DBH y Altura están correlacionadas, aunque no se supere el valor de VIF de 3. Vamos a considerar solo una de ellas en el modelo. Por lo tanto el modelo a ajustar va a tener exclusivamente los efectos principales de la altura, el número de árboles y la cobertura de copas. Antes de ajustar el modelo estandarizaremos las variables para poder comparar sus coeficientes. El modelo lo ajustaremos con *family="poisson"*, como no especificamos función de enlace utilizaremos la canónica, en este caso la función logarítmica.

```
> SQ$Ntrees.std <- (SQ$Ntrees-mean(SQ$Ntrees))/sd(SQ$Ntrees)
> SQ$TreeHeight.std <- (SQ$TreeHeight-mean(SQ$TreeHeight))/sd(SQ$TreeH
eight)
> SQ$CanopyCover.std <- (SQ$CanopyCover-mean(SQ$CanopyCover))/sd(SQ$Ca
nopyCover)
> M1 <- glm(SqCones~Ntrees.std+TreeHeight.std+CanopyCover.std,family="
poisson",data=SQ)
> summary(M1)

Call:
glm(formula = SqCones ~ Ntrees.std + TreeHeight.std + CanopyCover.std,
    family = "poisson", data = SQ)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-6.581  -3.388  -1.385   1.488   7.324

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      2.62399    0.04252  61.718  < 2e-16 ***
Ntrees.std       0.27415    0.02889   9.490  < 2e-16 ***
TreeHeight.std   0.19669    0.04601   4.275 1.91e-05 ***
CanopyCover.std  0.52852    0.06602   8.006 1.19e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)
```

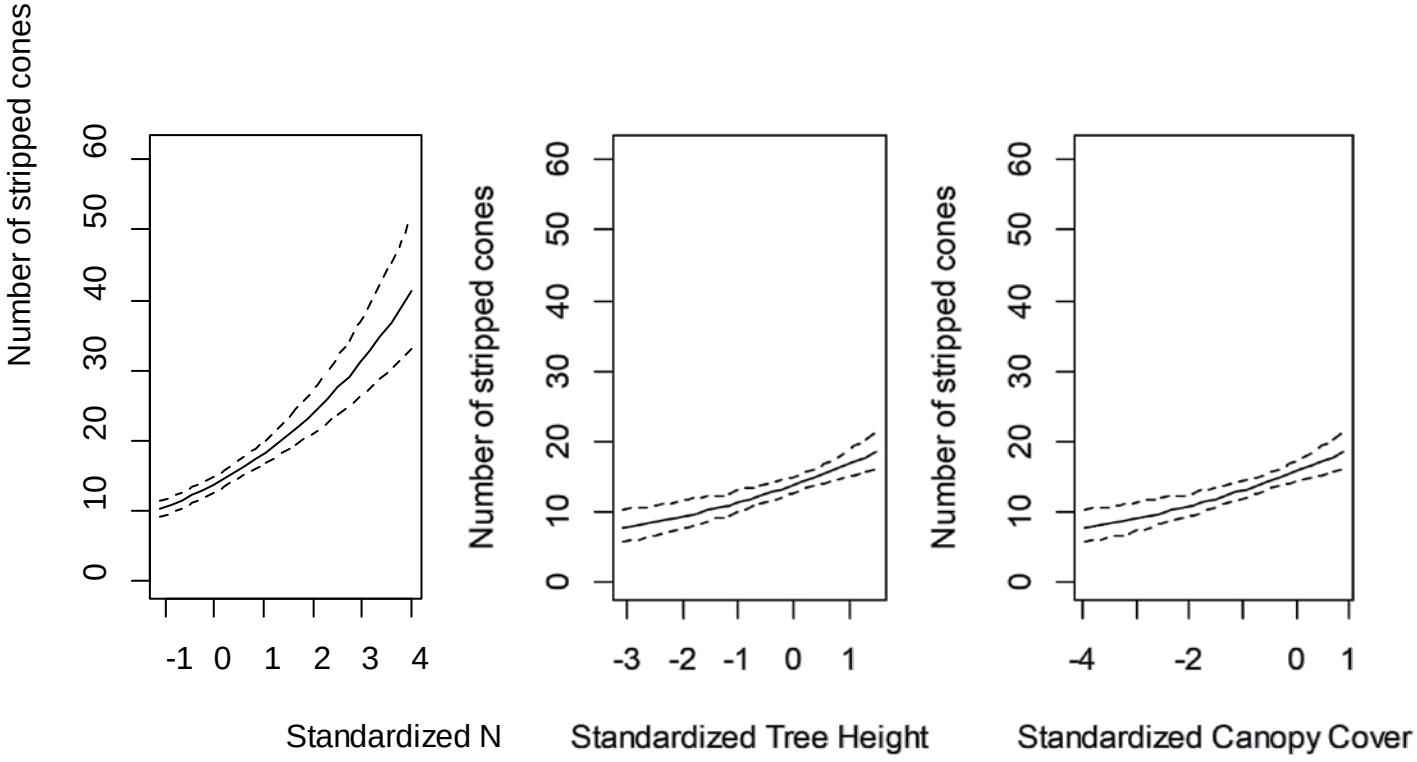
```
    Null deviance: 882.20  on 50  degrees of freedom
Residual deviance: 647.41  on 47  degrees of freedom
AIC: 850.98

Number of Fisher Scoring iterations: 5
```

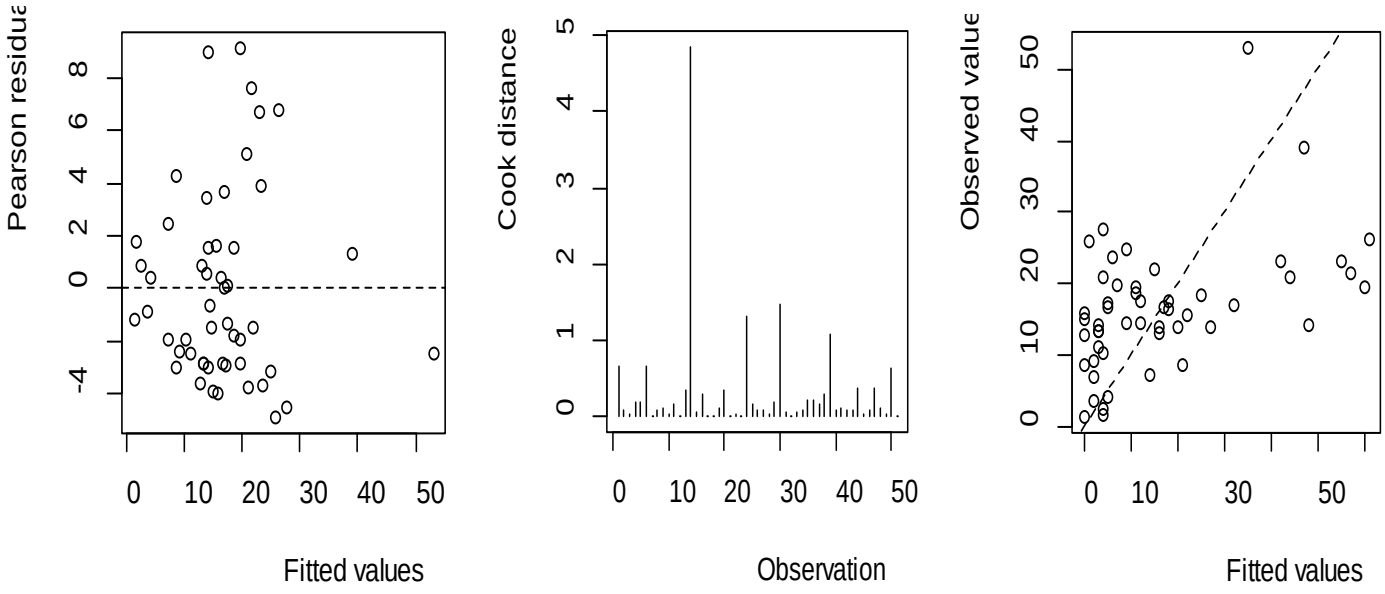
Como vemos, el modelo explica un 26.61% de la devianza ((882.20-647.41)/882.20). Como observamos todos los parámetros han resultado significativos. Vamos a realizar predicciones para las tres variables. Cuando predecimos para una determinada covariable, fijamos el valor del resto de covariables en el valor medio de la misma, como están estandarizas este valor medio es igual a 0.

```
> plot(x=SQ$Ntrees.std, y=SQ$SqCones, xlab="Standardized Number of tre
es",
+      ylab="Number of stripped cones", type="n")
> range(SQ$Ntrees.std)
[1] -1.104672  4.010439
> dataPlot <- data.frame(Ntrees.std=seq(-1.10,4.01, length=25),
+                        TreeHeight.std=0, CanopyCover.std=0)
> P1 <- predict(M1, newdata=dataPlot, type="link", se.fit=TRUE)
> lines(x=dataPlot$Ntrees.std, y=exp(P1$fit))
> lines(x=dataPlot$Ntrees.std, y=exp(P1$fit+1.96*P1$se.fit),lty=2)
> lines(x=dataPlot$Ntrees.std, y=exp(P1$fit-1.96*P1$se.fit),lty=2)
> range(SQ$TreeHeight.std)
[1] -3.055115  1.479494
> dataPlot1 <- data.frame(Ntrees.std=0,
+                        TreeHeight.std=seq(-3.05,1.47, length=25),
+                        CanopyCover.std=0)
> P2 <- predict(M1, newdata=dataPlot1, type="link", se.fit=TRUE)
> plot(x=SQ$TreeHeight.std, y=SQ$SqCones, xlab="Standardized Tree Heig
ht",
+      ylab="Number of stripped cones", type="n")
> lines(x=dataPlot1$TreeHeight.std, y=exp(P2$fit))
> lines(x=dataPlot1$TreeHeight.std, y=exp(P2$fit+1.96*P2$se.fit),lty=2
)
> lines(x=dataPlot1$TreeHeight.std, y=exp(P2$fit-1.96*P2$se.fit),lty=2
)
> range(SQ$CanopyCover.std)
[1] -3.9649004  0.8858951
> dataPlot2 <- data.frame(Ntrees.std=0,TreeHeight.std=0,
+                         CanopyCover.std=seq(-3.96,0.88, length=25))
> plot(x=SQ$CanopyCover.std, y=SQ$SqCones, xlab="Standardized Canopy C
over",
+      ylab="Number of stripped cones", type="n")
> P3 <- predict(M1, newdata=dataPlot2, type="link", se.fit=TRUE)
> lines(x=dataPlot2$CanopyCover.std, y=exp(P2$fit))
> lines(x=dataPlot2$CanopyCover.std, y=exp(P2$fit+1.96*P2$se.fit),lty=
2)
> lines(x=dataPlot2$CanopyCover.std, y=exp(P2$fit-1.96*P2$se.fit),lty=
2)
```

El resultado son tres gráficos, uno por covariable donde se muestra la predicción del modelo para el rango de dicha covariable:



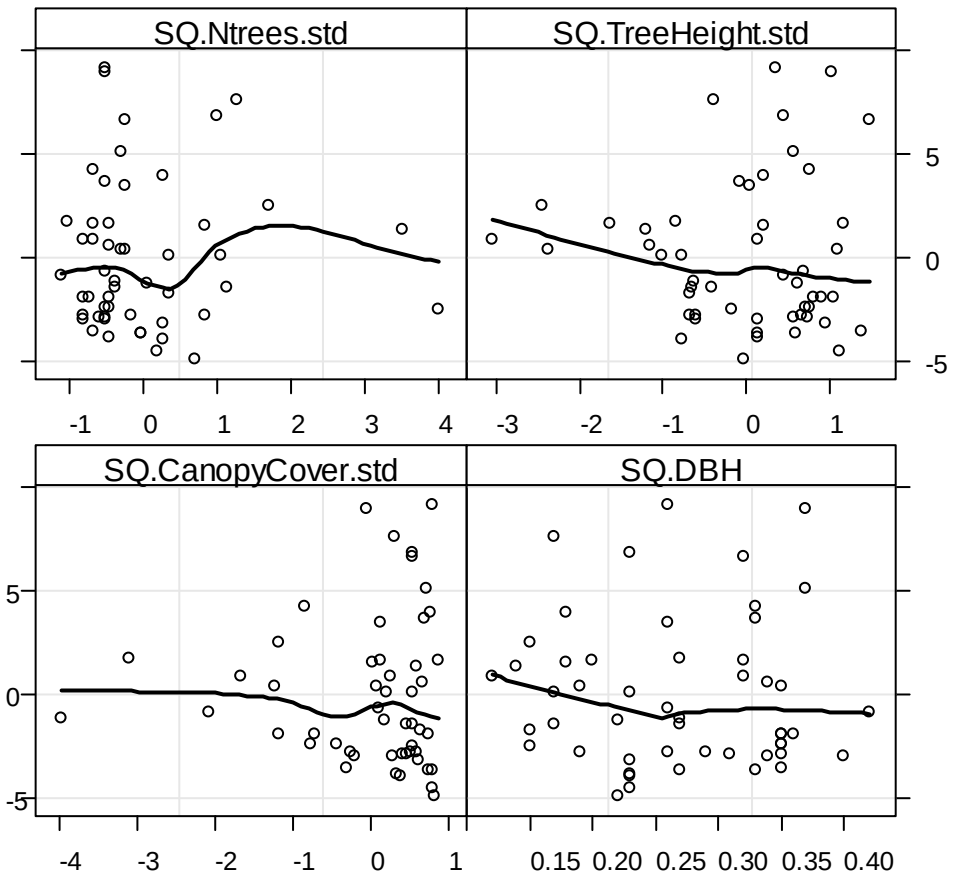
Vamos a comprobar si existe sobredispersión en nuestro modelo. En un modelo Poisson, el parámetro que indica la sobredispersión es igual a la devianza residual dividida por los grados de libertad residuales. Si dicho parámetro es mayor que 1, decimos que existe sobredispersión. En este caso el valor del parámetro es 13.77 (647.41/47 ver summary del modelo). Para intentar conocer las causas de dicha sobredispersión realizaremos una validación del modelo basado en



una serie de gráficos: a) Residuos de Pearson *vs* valores estimados; b) Distancias de Cook; c) Valores estimados *vs* valores observados.

De estos gráficos extraemos que existen una serie de observaciones con unos residuos elevados, que algunas observaciones a su vez influyen mucho en el ajuste del modelo aunque debido a su número eliminarlos no es una opción y que el ajuste del modelo no es muy bueno. Otra de las causas de sobredispersión es que las relaciones con las covariables no sean lineales o que haya alguna covariable que no hayamos tenido en cuenta en el modelo. Vamos a representar los residuos del modelo frente a cada variable incluida en el modelo y a su vez frente al DBH que ha sido inicialmente excluido del análisis.

```
> SQ2 <- data.frame(SQ$Ntrees.std,SQ$TreeHeight.std,SQ$CanopyCover.std
,SQ$DBH,E1)
> names(SQ2)
[1] "SQ.Ntrees.std"      "SQ.TreeHeight.std"  "SQ.CanopyCover.std" "SQ
.DBH"
[5] "E1"
> MyVar <- c("SQ.Ntrees.std","SQ.TreeHeight.std","SQ.CanopyCover.std",
"SQ.DBH")
> Myxyplot(SQ2,MyVar,"E1")
```



Como observamos no existe un patrón claro que nos pueda indicar una respuesta no lineal ante las covariables o bien que el DBH fuera importante y por tanto deberíamos haberla incluido en nuestro modelo. Llegados a este punto podemos deducir que la causa de la sobredispersión está

en una elección errónea de la estructura de errores del modelo. Normalmente los errores no siguen de forma perfecta una distribución Poisson o binomial, en estos casos se recomienda el utilizar una distribución (*family*=) *quasipoisson* o *quasibinomial*. En el caso de las distribuciones Poisson cuando los conteos son elevados se recomienda el uso de una distribución binomial negativa. En este caso el rango de la variable respuesta es de 0 a 80 por lo que podríamos pensar en la posibilidad de ajustar un modelo con dicha distribución.

```
> M2 <- glm.nb(SqCones~Ntrees.std+TreeHeight.std+CanopyCover.std,data=
SQ)
> summary(M2)

Call:
glm.nb(formula = SqCones ~ Ntrees.std + TreeHeight.std + CanopyCover.s
td,
    data = SQ, init.theta = 1.079102385, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4506  -1.0501  -0.4199   0.3910   1.6464

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.6180     0.1419  18.444   <2e-16 ***
Ntrees.std        0.3059     0.1543   1.982   0.0475 *
TreeHeight.std    0.1548     0.1606   0.964   0.3351
CanopyCover.std   0.5362     0.1735   3.090   0.0020 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(1.0791) family taken to be
1)

    Null deviance: 77.262  on 50  degrees of freedom
Residual deviance: 59.073  on 47  degrees of freedom
AIC: 383.81

Number of Fisher Scoring iterations: 1


              Theta:  1.079
          Std. Err.:  0.232

 2 x log-likelihood:  -373.813
```

Cabe destacar que el summary de este modelo nos estima un nuevo parámetro (Theta) asociado a la dispersión de la varianza. En este caso la sobredispersión está cercana a 1 (59.073/47 = 1.25). Observemos que la significación de los parámetros del modelo es muy distinta al modelo anterior. Una vez solucionado el problema de la sobredispersión, realizaremos la selección del modelo más parsimonioso.

```
> M2.1 <- update(M2,~.-Ntrees.std)
> M2.2 <- update(M2,~.-TreeHeight.std)
> M2.3 <- update(M2,~.-CanopyCover.std)
> AIC(M2,M2.1,M2.2,M2.3)
    df      AIC
M2   5 383.8132
```

```
M2.1  4 385.7714
M2.2  4 382.7519
M2.3  4 389.6779
> M3 <- M2.2
> M3.1 <- update(M3,~.-Ntrees.std)
> M3.2 <- update(M3,~.-CanopyCover.std)
> AIC(M3,M3.1,M3.2)
     df      AIC
M3    4 382.7519
M3.1  3 383.7719
M3.2  3 390.9637
> M4 <- M3.1
> Mnull <- glm.nb(SqCones~1,data=SQ)
> AIC(M4,Mnull)
      df      AIC
M4     3 383.7719
Mnull  2 393.5114
> summary(M4)

Call:
glm.nb(formula = SqCones ~ CanopyCover.std, data = SQ, init.theta = 0.
9882916132,
    link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4311  -1.0526  -0.2918   0.5740   1.5246

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.6583     0.1475   18.02  < 2e-16 ***
CanopyCover.std   0.6737     0.1684    4.00 6.32e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(0.9883) family taken to be
1)

    Null deviance: 71.908  on 50  degrees of freedom
Residual deviance: 58.905  on 49  degrees of freedom
AIC: 383.77

Number of Fisher Scoring iterations: 1

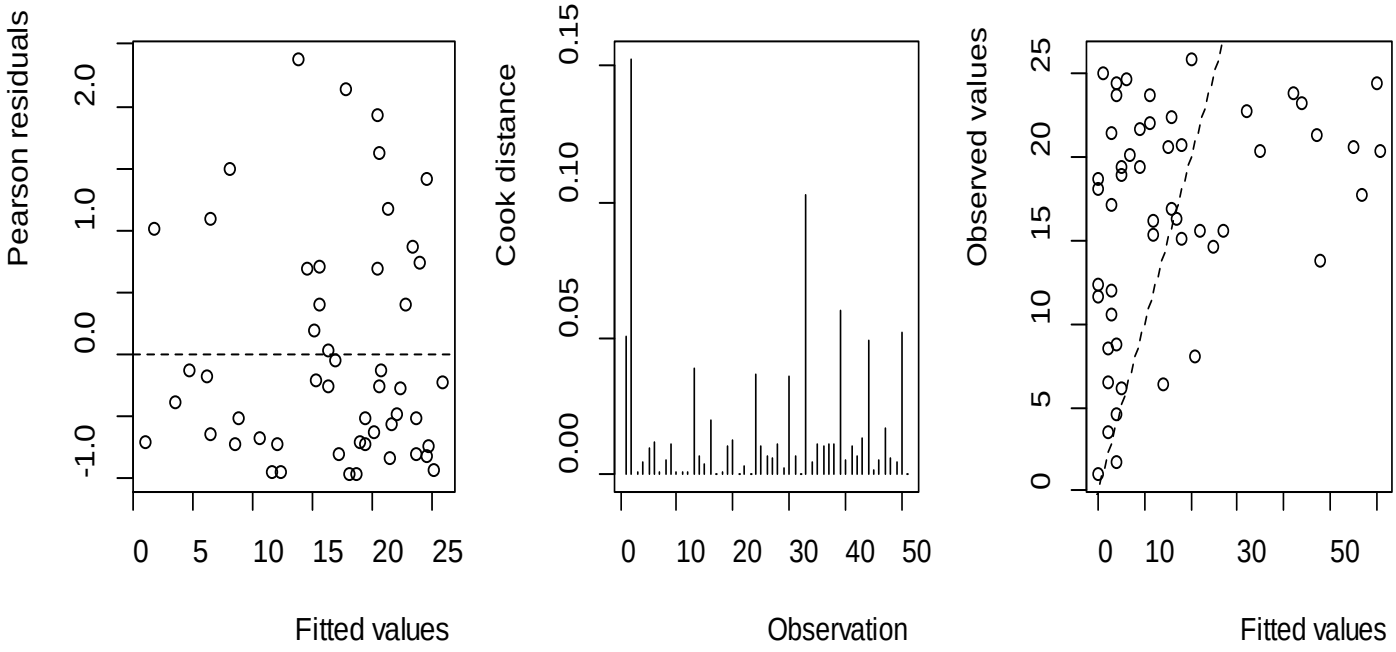

              Theta:  0.988
          Std. Err.:  0.208

 2 x log-likelihood:  -377.772
```

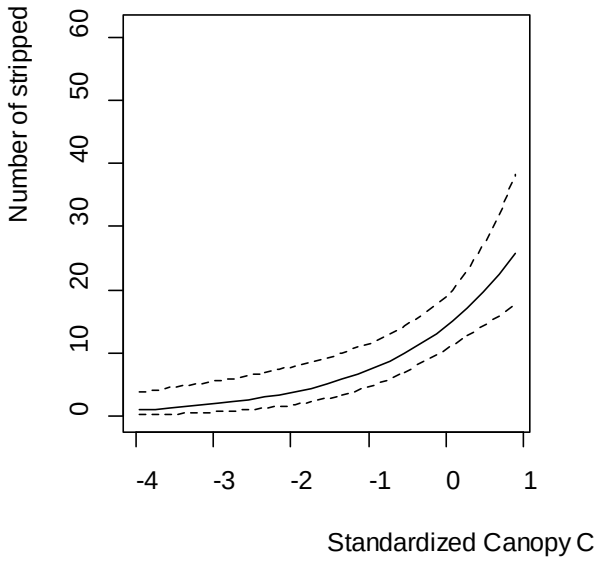
El modelo seleccionado solo considera el la cobertura de copa como covariable de interés para explicar el número de piñas roídas encontradas. Comprobaremos ahora que los gráficos de validación del modelo mejoran con respecto al modelo Poisson y mostraremos en un gráfico el efecto de la cobertura de copa en nuestra variable respuesta.

```
> F4 <- fitted(M4)
> E4 <- resid(M4, type="pearson")
> plot(F4,E4,xlab="Fitted values", ylab="Pearson residuals")
> abline(h=0, lty=2)
```

```
> plot(cooks.distance(M4),type="h",xlab="Observation", ylab="Cook dist
ance")
> plot(SQ$SqCones,F4,xlab="Fitted values", ylab="Observed values")
> abline(coef=c(0,1),lty=2)
```




```
> range(SQ$CanopyCover.std)
[1] -3.9649004  0.8858951
> dataPlot <- data.frame(CanopyCover.std=seq(-3.96,0.88, length=25))
> plot(x=SQ$CanopyCover.std, y=SQ$SqCones, xlab="Standardized Canopy C
over",
+      ylab="Number of stripped cones", type="n")
> P4 <- predict(M4, newdata=dataPlot, type="link", se.fit=TRUE)
> lines(x=dataPlot$CanopyCover.std, y=exp(P4$fit))
> lines(x=dataPlot$CanopyCover.std, y=exp(P4$fit+1.96*P4$se.fit),lty=2
)
> lines(x=dataPlot$CanopyCover.std, y=exp(P4$fit-1.96*P4$se.fit),lty=2
)
```

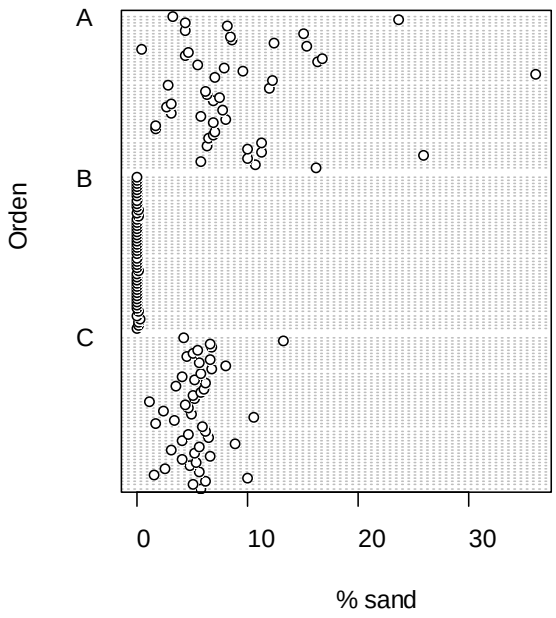


4. GLM con datos de presencia/ausencia (binomial)

Para ajustar un modelo lineal generalizado a datos de presencia/ausencia utilizaremos como ejemplo el set de datos PolichaetaV3.txt (Zuur et al. 2013). Estos datos muestran la presencia del gusano marino *Heteromastus similis* en tres localidades con diferentes condiciones ambientales (Location), en dos profundidades distinta de sustrato (Level) y en función del porcentaje de arena de dicho sustrato (MedSand). Como siempre cargaremos nuestros datos y realizaremos una exploración previa, en este caso centrándonos en la colinealidad entre variables y en el sentido de incluir interacciones triples o dobles.

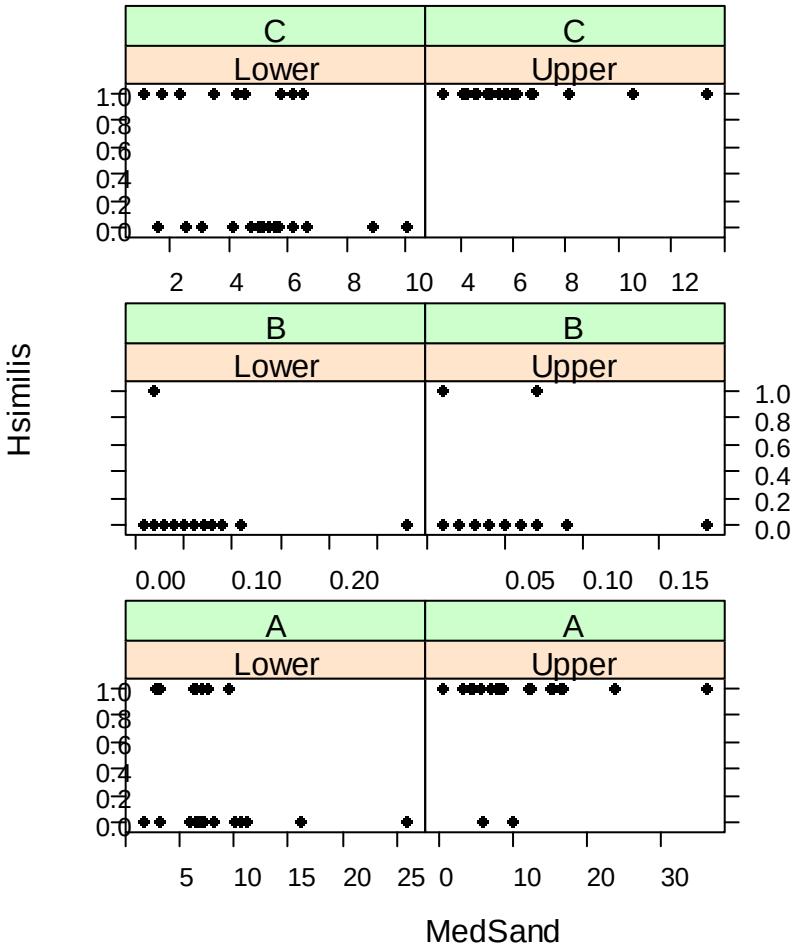
```
> PO <- read.table (file="PolychaetaV3.txt", header=T)
> str(PO)
'data.frame':  144 obs. of  4 variables:
 $ Hsimilis: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Level   : Factor w/ 2 levels "Lower","Upper": 1 1 1 1 1 1 2 1 1 1 .
..
 $ Location: Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...
 $ MedSand : num  16.24 10.73 5.84 10.03 25.95 ...
> table(PO$Hsimilis)

 0  1
78 66
> dotchart (PO$MedSand, xlab="% sand", ylab="Orden", groups=factor(PO$
Location))
> tapply(PO$MedSand,PO$Location,mean)
       A        B        C
9.070000 0.048125 5.432083
```



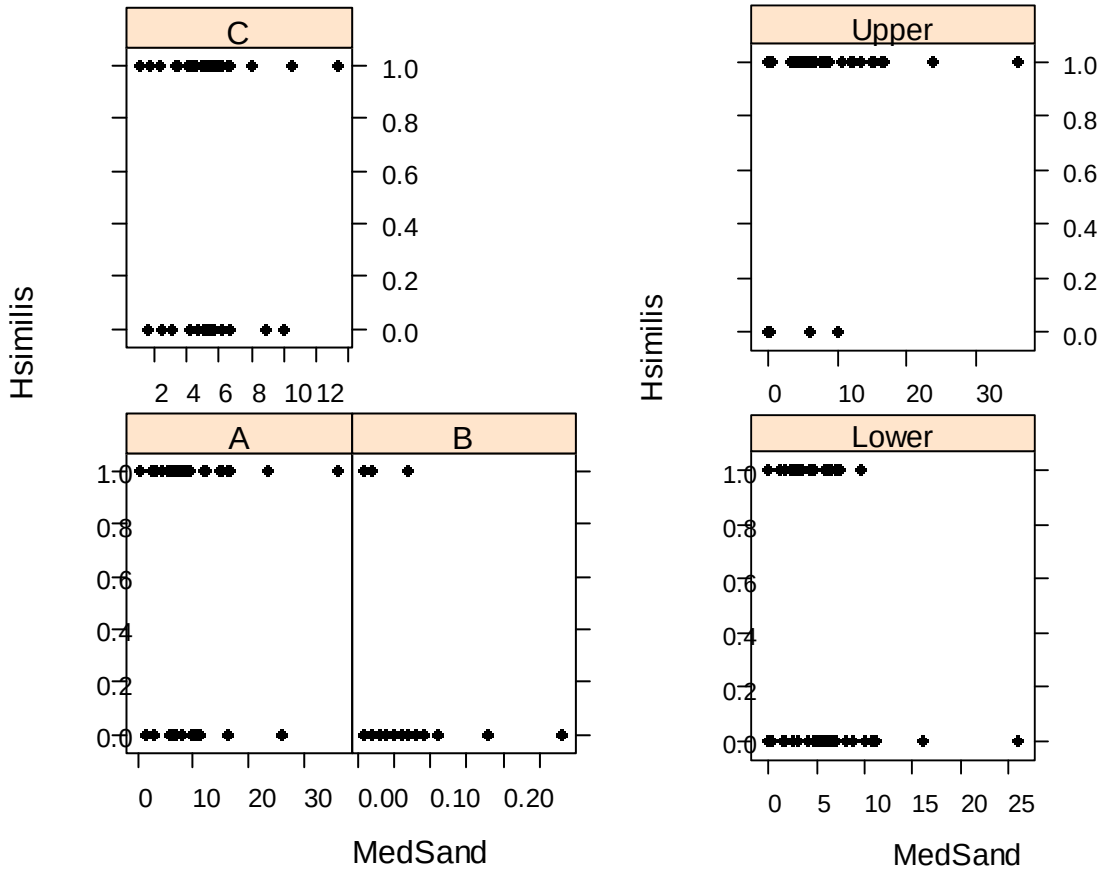
Como observamos existe una colinealidad entre el % de arena del sustrato y la localización, ya que en una de las localizaciones (B), el porcentaje de arena en el sustrato es muy cercano a 0 mientras que en el resto de localidades varía entre 0 y 30. Probamos ahora la viabilidad de incluir la triple interacción o las interacciones dobles entre las variables explicativas.

```
> xyplot(Hsimilis~MedSand | Level*Location,
+        data=PO, pch=16, col=1, scales=list(alternating=TRUE,
+                                            x=list(relation="free"),
+                                            y=list(relation="same")))
```



Aquí podemos observar la interacción triple entre el % de arena, el nivel de profundidad y la localización. En el caso de la localización B, casi todas las observaciones son 0 mientras que en el caso de la localización C y A, casi no hay ausencias para la profundidad superior. Si queremos evaluar de forma correcta una interacción triple deberíamos tener, a groso modo, un número parecido de observaciones de presencia y ausencia para todas las combinaciones posibles.

```
> xyplot(Hsimilis~MedSand | Location,
+        data=PO, pch=16, col=1, scales=list(alternating=TRUE,
+                                            x=list(relation="free"),
+                                            y=list(relation="same")))
> xyplot(Hsimilis~MedSand | Level,
+        data=PO, pch=16, col=1, scales=list(alternating=TRUE,
+                                            x=list(relation="free"),
+                                            y=list(relation="same")))
> plot.design(Hsimilis~Level+Location, data=PO)
```



Aquí podemos observar las interacciones dobles entre el % de arena y la localización (izquierda) y entre el % de arena y la profundidad. En el caso de la localización B observamos solo tres observaciones de presencia, además como hemos comentado antes existe una gran colinealidad entre la localización y el % de arena (ver eje de las x en el gráfico de la izquierda). En cuanto a la interacción entre el % de arena y la profundidad podemos observar como las observaciones están mejor distribuidas (ojo hay puntos superpuestos en la profundidad superior), además tenemos datos para toda la variabilidad de % de arena. Por otro lado podemos pensar en una hipótesis acerca de esta interacción ya que podemos esperar que la presencia de *H.similis* en horizontes profundos dependa de la estructura del sustrato (% arena). Por tanto el modelo completo tendrá en cuenta esta interacción y sus componentes principales. Aunque solo hay una covariable, vamos a estandarizarla ya que mejorará el ajuste del modelo.

```
> PO$MedSand.st <- (PO$MedSand-mean(PO$MedSand))/sd(PO$MedSand)
> M1 <- glm(Hsimilis~MedSand.st*Level, data=PO, family="binomial")
> summary(M1)

Call:
glm(formula = Hsimilis ~ MedSand.st * Level, family = "binomial",
    data = PO)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.4182  -0.7308  -0.6067   0.4507   1.8918

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)            -1.0875     0.2737  -3.974 7.07e-05 ***
MedSand.st              0.1104     0.3197   0.346     0.73
LevelUpper              3.0920     0.6505   4.753 2.00e-06 ***
MedSand.st:LevelUpper   3.9667     0.9385   4.227 2.37e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

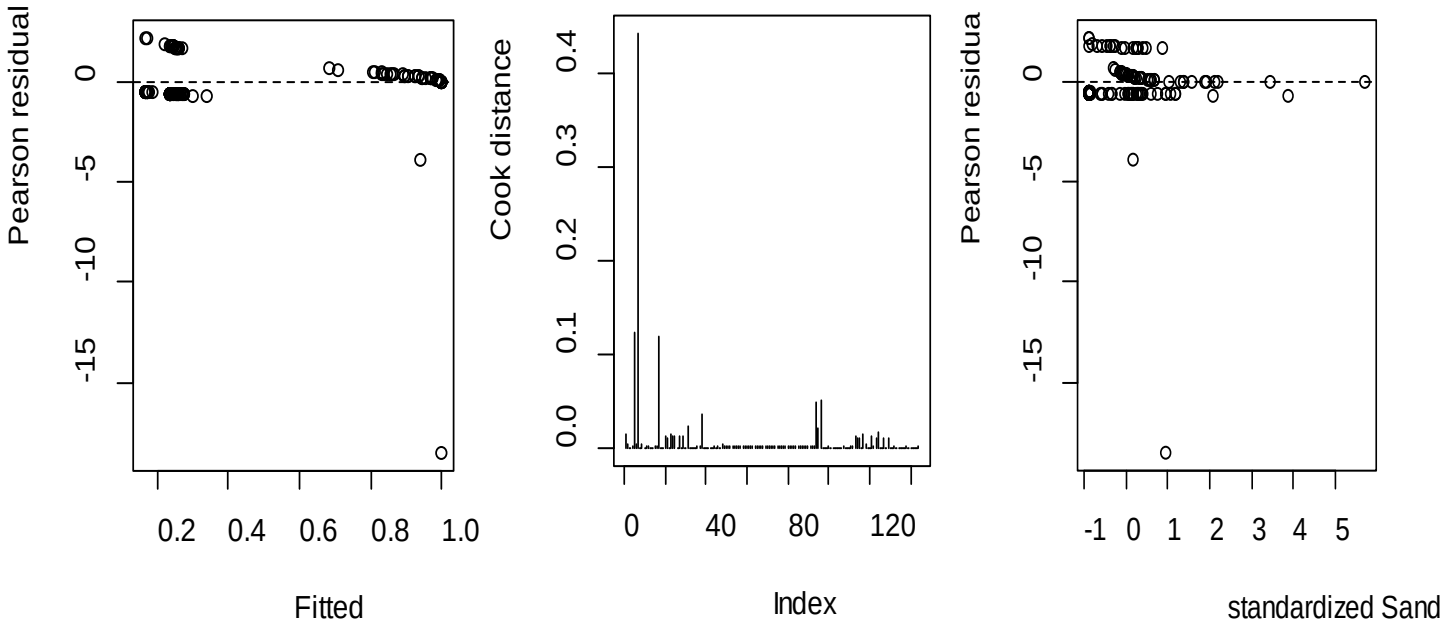
    Null deviance: 198.63  on 143  degrees of freedom
Residual deviance: 123.58  on 140  degrees of freedom
AIC: 131.58

Number of Fisher Scoring iterations: 7
```

```
> M1.1 <- glm(Hsimilis~MedSand.st+Level, data=PO, family="binomial")
> AIC(M1,M1.1)
     df      AIC
M1    4 131.5764
M1.1  3 157.8658
```

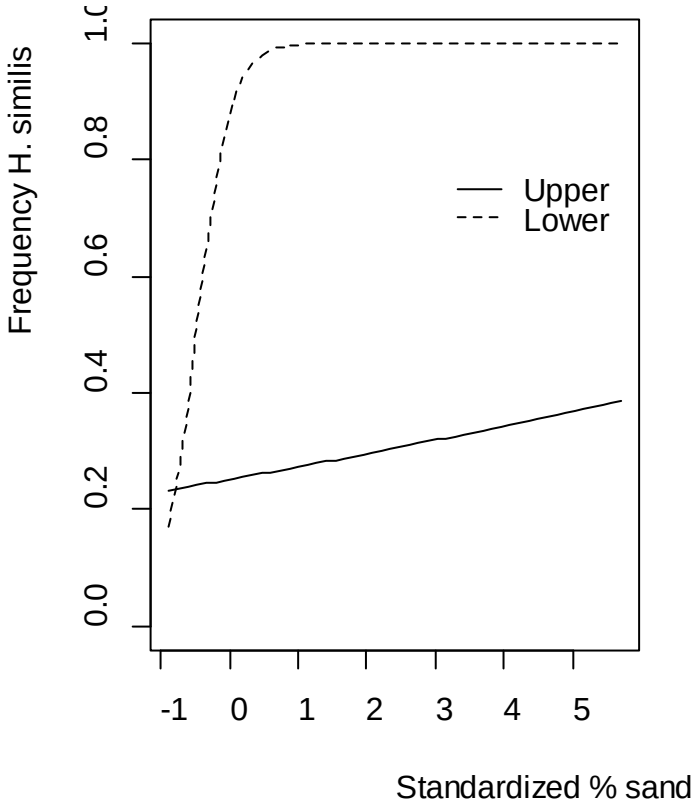
En este caso el modelo seleccionado es el mismo que el completo, al seleccionar la interacción no podemos simplificarlo más. Los resultados del modelo también descartan la presencia de sobredispersión ya que la devianza residual no es mayor a los grados de libertad del modelo. Falta realizar la validación gráfica del modelo a partir de los residuos del mismo, como nuestras observaciones toman valores de 1 ó 0, tendremos dos bandas de residuos.

```
> E1 <- resid(M1,type="pearson")
> F1 <- fitted(M1, type="response")
> plot(F1,E1,xlab="Fitted",ylab="Pearson residuals")
> abline(h=0, lty=2)
> plot(cooks.distance((M1),type="h",ylab="Cook distance"))
> plot(PO$MedSand.st,E1,xlab="standardized Sand %",ylab="Pearson resid
uals")
> abline(h=0,lty=2)
```



Finalmente nos queda visualizar los resultados de nuestro modelo. Para ello nos basaremos en predicciones realizadas a partir del modelo ajustado.

```
> range(PO$MedSand.st)
[1] -0.8856961  5.7148408
> dataPlot <- data.frame(MedSand.st=rep(seq(-0.88,5.71, length=50),2),
+                        Level=c(rep("Lower",50),rep("Upper",50)))
> plot(x=PO$MedSand.st, y=PO$Hsimilis, xlab="Standardized % sand",
+      ylab="Frequency H. similis", type="n")
> P1 <- predict(M1, newdata=dataPlot, type="link", se.fit=TRUE)
> lines(x=dataPlot$MedSand.st[dataPlot$Level=="Upper"],
+       y=exp(P1$fit[c(1:50)])/(1+exp(P1$fit[c(1:50)])))
> lines(x=dataPlot$MedSand.st[dataPlot$Level=="Lower"],
+       y=exp(P1$fit[c(51:100)])/(1+exp(P1$fit[c(51:100)])),lty=2)
> legend(x=3,y=0.8, legend=expression("Upper","Lower"), lty=c(1,2),bty
="n")
```



5. Introducción a los modelos mixtos

Los modelos mixtos son una extensión de los modelos lineales, lineales generalizados o no lineales. Estos modelos permiten incluir en ellos factores que influyen sobre la varianza de nuestras observaciones permitiendo así modelar no solo las medias sino también la varianza y la covarianza. Estos modelos sirven a su vez para incluir variabilidad que no sea constante y que esté correlacionada dentro de los niveles de un determinado factor, es por ello que son muy útiles para los experimentos y muestreos donde se incumpla el supuesto de independencia dando lugar a pseudorréplicas. El calificativo de mixto es debido a que incluyen dos tipos de factores:

- Efectos fijos: Son parámetros desconocidos que queremos estimar en nuestro modelo, por tanto estamos interesados en comprobar su efecto sobre nuestra variable respuesta. Los niveles de estos factores son informativos o bien han sido incluso establecidos a priori. Estos efectos tienen una influencia sobre la media de la variable respuesta.
- Efectos aleatorios: Afectan a la estructura de varianza-covarianza de la variable respuesta y es por ello que su efecto es sobre la varianza y no sobre la media. Aquellas observaciones realizadas dentro del mismo nivel de un factor aleatorio estarán correlacionadas entre sí. Estos niveles del factor aleatorio no son informativos y provienen de una población de niveles mayor de la cual nosotros extraemos una muestra aleatoria.

Como hemos comentado anteriormente los modelos mixtos son especialmente útiles para los casos en los que tenemos autocorrelación entre nuestras observaciones. Esta autocorrelación suele ser espacial, cuando tomamos diversas muestras en un mismo plot o muy cercanas unas de otras con respecto a otras muestras, y/o temporal cuando realizamos medidas repetidas en un mismo individuo o punto de muestreo. Esta autocorrelación da lugar a pseudorréplicas y supone una violación del principio de independencia de los modelos lineales. La principal ventaja de los modelos mixtos respecto a la presencia de pseudorréplicas es que economiza los grados de libertad. En vez de estimar un parámetro para cada uno de los niveles del factor aleatorio, el modelo mixto predice una variabilidad asociada a los parámetros estimados siendo dicha variabilidad debida a los distintos niveles de este factor aleatorio.

Para discernir si un determinado factor debe ser considerado como fijo o aleatorio debemos preguntarnos si estamos o no interesados en el efecto de dicho factor, si la respuesta es que sí entonces debemos considerarlo como parte de la estructura fija del modelo. Otra pregunta es si estamos considerando todos los niveles posibles de dicho factor o si bien estamos trabajando con una muestra de dichos niveles, si la respuesta es que es una muestra (aleatoria) entonces estamos ante un factor aleatorio. Es importante señalar que al menos son necesarios 5 niveles para considerar un factor como aleatorio, aunque preferiblemente el número de niveles debería ser superior a 10.

6. Formulación de modelos mixtos en R

Existen dos paquetes principales para ajustar modelos lineales en R, el paquete *nlme* y su predecesor *lme4*. Las funciones para ajustar modelos mixtos son *lme (nlme)* y *lmer (lme4)* respectivamente. La especificación de la estructura del modelo cambia en función del paquete que utilicemos. Es importante reseñar que el efecto aleatorio puede influenciar o bien el intercepto o bien la relación de nuestra variable respuesta con alguna de las covariables. En la mayoría de los casos el efecto es sobre el intercepto. Por ejemplo si tenemos autocorrelación espacial al tener muchos puntos de muestreos dentro de una misma parcela, la parcela se considerará como efecto aleatorio ya que las muestras dentro de la misma parcela estarán más autocorrelacionadas entre sí que con las muestras de otras parcelas. De igual manera si hemos medido el crecimiento de unos determinados árboles durante 20 años, los datos tomados en los mismos individuos están correlacionados entre sí. En este tipo de casos el efecto aleatorio suele

considerarse en el intercepto y vendría a indicar que el intercepto de cada parcela o individuo puede variar de forma aleatoria, dicho de otro modo nuestros árboles tienen un crecimiento intrínseco (i.e. intercepto) independientemente de su relación con las covariables. Si por el contrario creemos que la relación entre el crecimiento de nuestros árboles y la precipitación puede cambiar de manera aleatoria con el individuo entonces consideraríamos un efecto aleatorio del individuo en la pendiente de dicha relación. Por otro lado los efectos aleatorios pueden estar anidados, por ejemplo si muestreamos distintos puntos en distintas parcelas ubicadas en distintos bosques. En este ejemplo la parcela estaría anidada dentro del bosque.

6.1. Ajuste con lme (*nlme*)

En este caso se utilizan dos argumentos distintos, uno para la parte fija (*fixed=*) donde se incluirían las covariables (e.g. *x1*, *x2*) y otro para la aleatoria (*random=*) donde se especificaría el efecto(s) aleatorio(s) (e.g. *a* y *b*). La variable respuesta (*y*) estará especificada en la parte fija del modelo. A continuación se especifican distintos ejemplos:

Modelo mixto con dos covariables y un efecto random en el intercepto

lme (fixed = y ~ x1+x2, random = ~ 1 | a)

Modelo mixto con dos covariables y un efecto random en el intercepto y sobre la pendiente de una de las covariables

lme (fixed = y ~ x1+x2, random = ~ x1 | a)

Modelo mixto con dos covariables y un efecto random en el intercepto donde el efecto b está anidado en a

lme (fixed = y ~ x1+x2, random = ~ 1 | a / b)

6.1. Ajuste con lmer (*lme4*)

En este caso especificamos todo en un mismo argumento, efectos fijos y aleatorios. En primer lugar especificamos los efectos fijos y en después los aleatorios.

Modelo mixto con dos covariables y un efecto random en el intercepto

lmer (y ~ x1+x2+(~ 1 | a))

Modelo mixto con dos covariables y un efecto random en el intercepto y sobre la pendiente de una de las covariables

lmer (y ~ x1+x2+(~ x1 | a))

Modelo mixto con dos covariables y un efecto random en el intercepto donde el efecto b está anidado en a

lmer (y ~ x1+x2+(~ 1 | a / b))

7. Ejemplo de modelo mixto

Para este ejemplo utilizaremos los datos de Zuur et al. (2007) utilizan datos de bentos marino procedente de nueve zonas intermareales (beach) de la costa holandesa. En cada zona intermareal se tomaron cinco muestras (Sites) de la macro-fauna y variables abióticas. En este ejemplo vamos a ver si existe alguna relación entre la riqueza de especies y la altura de cada punto de muestreo con respecto al nivel medio de la marea (NAP). Los datos son una matriz de abundancia para las distintas especies, por lo que para calcular la riqueza voy a convertir dicha matriz en una de presencia/ausencia y sumarlas para cada fila (observación). Posteriormente ajustaremos primero un modelo lineal obviando la violación de la independencia de las observaciones,

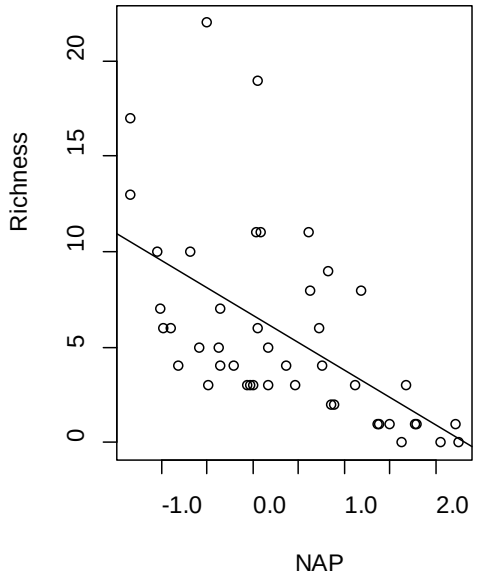
```
> RIKZ <- read.table(file="RIKZ.txt", header=T)
> RIKZ$Richness <- apply(RIKZ[, 2:76] > 0, 1, sum)
> M1 <- lm(Richness ~ NAP, data = RIKZ)
> plot(RIKZ$NAP, RIKZ$Richness, xlab="NAP", ylab="Richness")
> abline(M1)
> summary(M1)

Call:
lm(formula = Richness ~ NAP, data = RIKZ)

Residuals:
    Min      1Q  Median      3Q     Max
-5.0675 -2.7607 -0.8029  1.3534 13.8723

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.6857     0.6578  10.164 5.25e-13 ***
NAP          -2.8669     0.6307  -4.545 4.42e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

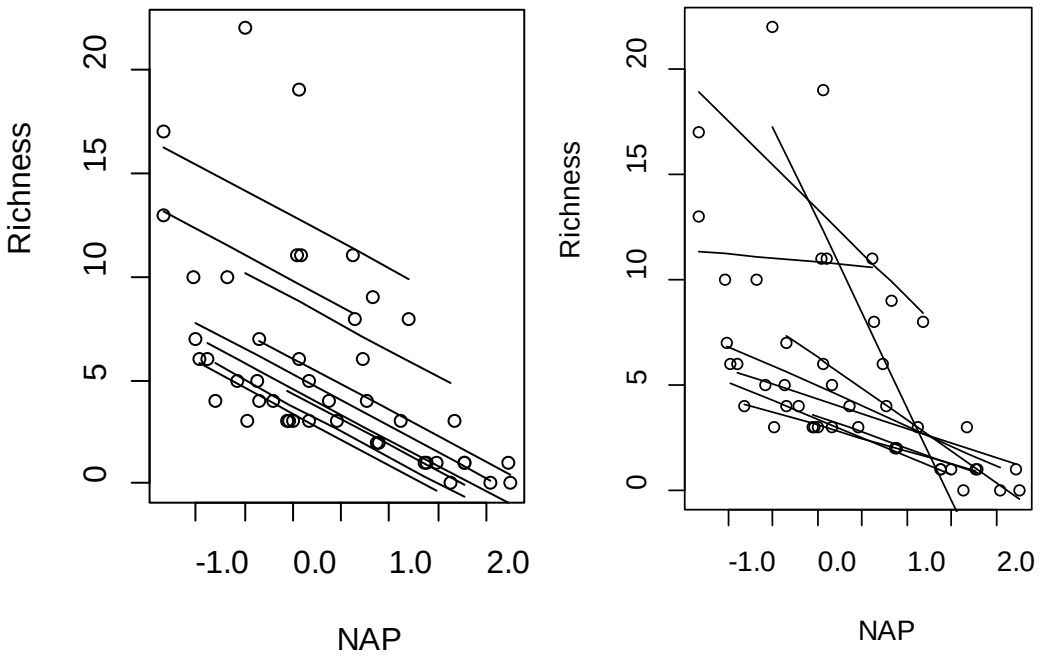
Residual standard error: 4.16 on 43 degrees of freedom
Multiple R-squared:  0.3245,   Adjusted R-squared:  0.3088
F-statistic: 20.66 on 1 and 43 DF,  p-value: 4.418e-05
```



El modelo assume una relación lineal entre riqueza y NAP sin embargo está relación podría depender de la playa, de modo que la playa influyera sobre el intercepto (izq) o bien sobre la pendiente (derecha).

```
> M1.1<-lm(Richness~NAP+factor(Beach), data = RIKZ)
> plot(RIKZ$NAP,RIKZ$Richness,xlab="NAP",ylab="Richness")
> for (i in 1:9){
+    J<-RIKZ$Beach==i
+    x1<-RIKZ$NAP[J]
+    y1<-M1.1$fitted[J]
+    Ord<-order(x1)
+    lines(x1[Ord],y1[Ord])}
```

```
> M1.2<-lm(Richness~NAP*factor(Beach), data = RIKZ)
> plot(RIKZ$NAP,RIKZ$Richness,xlab="NAP",ylab="Richness")
> for (i in 1:9){
+    J<-RIKZ$Beach==i
+    x1<-RIKZ$NAP[J]
+    y1<-M1.2$fitted[J]
+    Ord<-order(x1)
+    lines(x1[Ord],y1[Ord])}
```




```
> summary(M1.1)

Call:
lm(formula = Richness ~ NAP + factor(Beach), data = RIKZ)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8518 -1.5188 -0.1376  0.7905 11.8384

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)      9.8059     1.3895   7.057 3.22e-08 ***
NAP             -2.4928     0.5023  -4.963 1.79e-05 ***
factor(Beach)2   3.0781     1.9720   1.561  0.12755
factor(Beach)3  -6.4049     1.9503  -3.284  0.00233 **
factor(Beach)4  -6.0329     2.0033  -3.011  0.00480 **
factor(Beach)5  -0.8983     2.0105  -0.447  0.65778
factor(Beach)6  -5.2231     1.9682  -2.654  0.01189 *
factor(Beach)7  -5.4367     2.0506  -2.651  0.01196 *
factor(Beach)8  -4.5530     1.9972  -2.280  0.02883 *
factor(Beach)9  -3.7820     2.0060  -1.885  0.06770 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.06 on 35 degrees of freedom
Multiple R-squared:  0.7025,   Adjusted R-squared:  0.626
F-statistic: 9.183 on 9 and 35 DF,  p-value: 5.645e-07

> summary(M1.2)

Call:
lm(formula = Richness ~ NAP * factor(Beach), data = RIKZ)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3022 -0.6614 -0.1076  0.4072  5.9090

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         10.8219     1.3341   8.112 1.03e-08 ***
NAP                 -0.3718     1.5493  -0.240 0.812140
factor(Beach)2       2.5238     1.7685   1.427 0.165013
factor(Beach)3      -7.4212     1.7346  -4.278 0.000211 ***
factor(Beach)4      -7.7342     1.8541  -4.171 0.000281 ***
factor(Beach)5       1.9609     1.9485   1.006 0.323166
factor(Beach)6      -6.4973     1.7494  -3.714 0.000938 ***
factor(Beach)7      -7.3013     2.2611  -3.229 0.003253 **
factor(Beach)8      -5.8704     1.7981  -3.265 0.002974 **
factor(Beach)9      -4.5268     1.8631  -2.430 0.022036 *
NAP:factor(Beach)2  -3.8034     1.9941  -1.907 0.067169 .
NAP:factor(Beach)3  -1.3835     2.0405  -0.678 0.503511
NAP:factor(Beach)4  -0.8767     1.9528  -0.449 0.657044
NAP:factor(Beach)5  -8.5283     2.1340  -3.996 0.000447 ***
NAP:factor(Beach)6  -1.0167     1.8290  -0.556 0.582872
NAP:factor(Beach)7  -1.1458     2.2756  -0.504 0.618683
NAP:factor(Beach)8  -1.5212     1.8134  -0.839 0.408891
NAP:factor(Beach)9  -2.5957     1.9537  -1.329 0.195100
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.479 on 27 degrees of freedom
Multiple R-squared:  0.8494,   Adjusted R-squared:  0.7546
F-statistic: 8.957 on 17 and 27 DF,  p-value: 4.056e-07
```

Como vemos el número de parámetros a estimar aumenta conforme añadimos complejidad al modelo. Si hiciéramos una selección por AIC deberíamos quedarnos con el más complejo de ellos.

```
> AIC(M1,M1.1,M1.2)
     df      AIC
M1    3 259.9535
M1.1 11 239.0549
M1.2 19 224.4232
```

La pregunta que debemos hacernos es si estamos interesados en saber cuál es el efecto de la playa en nuestro análisis o si la playa es un factor del cual hemos tomado una muestra aleatoria donde estudiar la relación entre la riqueza de especies y nuestra variable NAP. Sin embargo si quitamos la playa del análisis nuestro R2 pasaría de 0.85 a 0.33. Por otro lado no podemos obviar la violación del principio de independencia en las observaciones ya que existe una correlación espacial entre las muestras tomadas dentro de la misma playa. No considerar este aspecto puede llevarnos a malinterpretar la significación de los efectos fijos del modelo. Además no nos olvidemos que estamos ajustando 18 parámetros con tan solo 45 observaciones. La solución a esto es aplicar un modelo mixto donde consideremos la playa como efecto aleatorio. En este ejemplo utilizaremos la función *lmer* del paquete *lme4*.

```
> M2 <- lmer(Richness ~ NAP + (1|Beach), data = RIKZ)
> summary(M2)
Linear mixed model fit by REML ['lmerMod']
Formula: Richness ~ NAP + (1 | Beach)
   Data: RIKZ

REML criterion at convergence: 239.5

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.4227 -0.4848 -0.1576  0.2519  3.9794

Random effects:
 Groups   Name        Variance Std.Dev.
 Beach    (Intercept) 8.668    2.944
 Residual             9.362    3.060
Number of obs: 45, groups:  Beach, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)   6.5819     1.0958   6.007
NAP          -2.5684     0.4947  -5.192

Correlation of Fixed Effects:
    (Intr)
NAP -0.157
> anova(M2)
Analysis of Variance Table
    Df Sum Sq Mean Sq F value
NAP  1 252.33  252.33  26.952
```

Como vemos el resultado producido no nos proporciona un valor de significación para los parámetros del modelo, este punto es de gran controversia en la comunidad científica. En este curso hemos optado por seleccionar el mejor modelo mediante criterios de información por lo que seguiremos ese camino. Como vemos el modelo empeora al quitar nuestra covariable de interés. Aquí debemos aclarar que los modelos mixtos ajustan sus parámetro por máxima

verosimilitud restringida (REML), sin embargo si queremos comparar distintos modelos que difieran solo en la estructura de la parte fija debemos utilizar en este caso el método de la máxima verosimilitud (ML).

```
> M2.1 <- lmer(Richness ~ NAP + (1|Beach), data = RIKZ, REML=F)
> M2.2 <- lmer(Richness ~ 1 + (1|Beach), data = RIKZ, REML=F)
> AIC(M2.1,M2.2)
     df      AIC
M2.1  4 249.8291
M2.2  3 269.3035
```

Como hemos visto anteriormente también podemos pensar que existe un efecto aleatorio de la playa sobre la relación existente entre la riqueza de especie y el NAP.

```
> M3 <- lmer(Richness ~ NAP + (NAP|Beach), data = RIKZ)
> summary(M3)
Linear mixed model fit by REML ['lmerMod']
Formula: Richness ~ NAP + (NAP | Beach)
   Data: RIKZ

REML criterion at convergence: 232.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.8213 -0.3411 -0.1675  0.1921  3.0397

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Beach    (Intercept) 12.596   3.549
          NAP          2.941   1.715    -0.99
 Residual              7.305   2.703
Number of obs: 45, groups:  Beach, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)   6.5887     1.2648   5.209
NAP          -2.8300     0.7229  -3.915

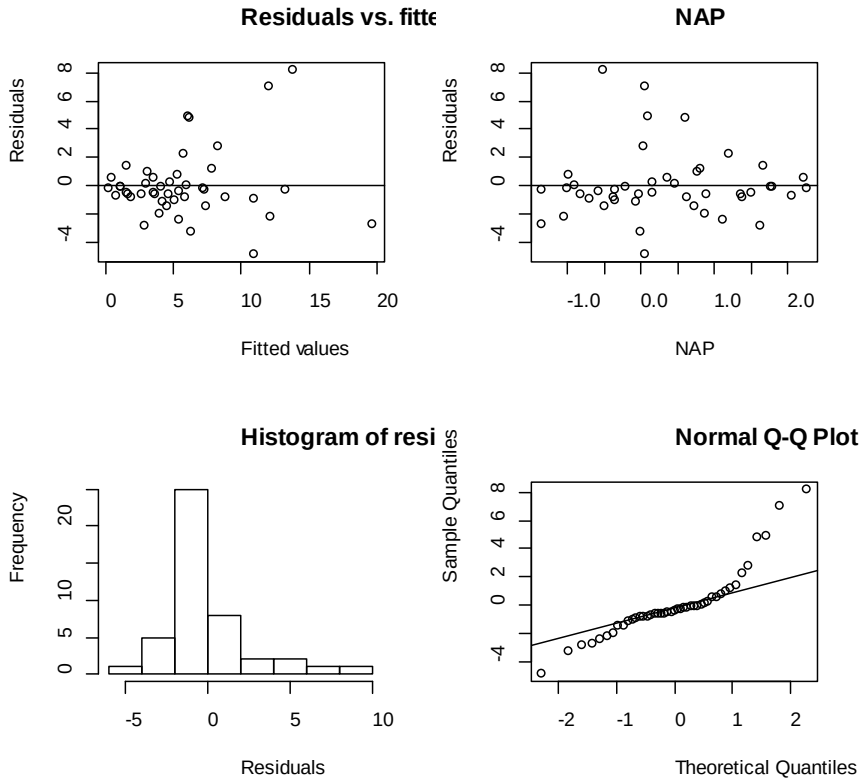
Correlation of Fixed Effects:
    (Intr)
NAP -0.819
> anova(M3)
Analysis of Variance Table
    Df Sum Sq Mean Sq F value
NAP  1 111.95  111.95  15.324
> M3.1 <- lmer(Richness ~ NAP + (NAP|Beach), data = RIKZ, REML=F)
> M3.2 <- lmer(Richness ~ 1 + (NAP|Beach), data = RIKZ, REML=F)
> AIC(M3.1,M3.2)
     df      AIC
M3.1  6 246.6561
M3.2  5 254.2402
```

Los resultados del modelo son muy parecidos, siendo el NAP importante en ambos casos. Pero, ¿cómo seleccionar la parte aleatoria de un modelo? Existen dos enfoques, el primero (Burnham and Anderson 2002) basado en el denominado “Information-Theoretic approach” establece que debemos conocer la estructura de correlación de nuestros datos por lo que somos nosotros

quienes establecemos la misma *a priori*. La segunda expone que se pueden comparar mediante criterios de información dos modelos con distinta estructura aleatoria y ver cual presenta un AIC menor. Es importante que en este caso la estimación de los parámetros se realice mediante REML. En el ejemplo anterior el modelo con el efecto aleatorio sobre el intercepto y la pendiente presenta un AIC menor.

```
> AIC(M2,M3)
   df      AIC
M2  4 247.4802
M3  6 244.3839
```

Al igual que en todos los tipos de modelos debemos evaluar el modelo seleccionado mediante sus residuos. En este caso la función *plot* no nos dibuja los gráficos por lo que debemos hacerlos a mano. Para saber si el modelo es adecuado debemos mirar los residuos normalizados del modelo estimado a partir de REML. También dibujaremos los residuos frente a los valores predichos para ver si hay homocedasticidad. Si no hay homogeneidad de varianzas entonces podemos aplicar una transformación en la variable respuesta; ver si la inclusión de otra covariable mejora el modelo; o aplicar un modelo mixto generalizado con una estructura de error distinta. También podemos dibujar los residuos frente a las covariables. Al igual que con otros tipos de modelos debemos no encontrar ningún tipo de patrón en estos gráficos.



Como podemos observar existe heterocedasticidad ya que la varianza aumenta a medida que aumenta la media de los valores estimados. Una posible causa es que los valores de riqueza son

un conteo de especie y por lo tanto su estructura de error se ajustaría mejor a una distribución Poisson. Es posible ajustar un modelo mixto generalizado mediante el uso de la función *glmer*.

```
> M4 <- glmer(Richness ~ NAP + (NAP|Beach), data = RIKZ, family=poisso
n)
> summary(M4)
Generalized linear mixed model fit by maximum likelihood (Laplace Appr
oximation) ['glmerMod']
 Family: poisson  ( log )
Formula: Richness ~ NAP + (NAP | Beach)
   Data: RIKZ

     AIC      BIC   logLik deviance df.resid
   218.7    227.8   -104.4    208.7       40

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.35846 -0.51129 -0.21846  0.09802  2.45384

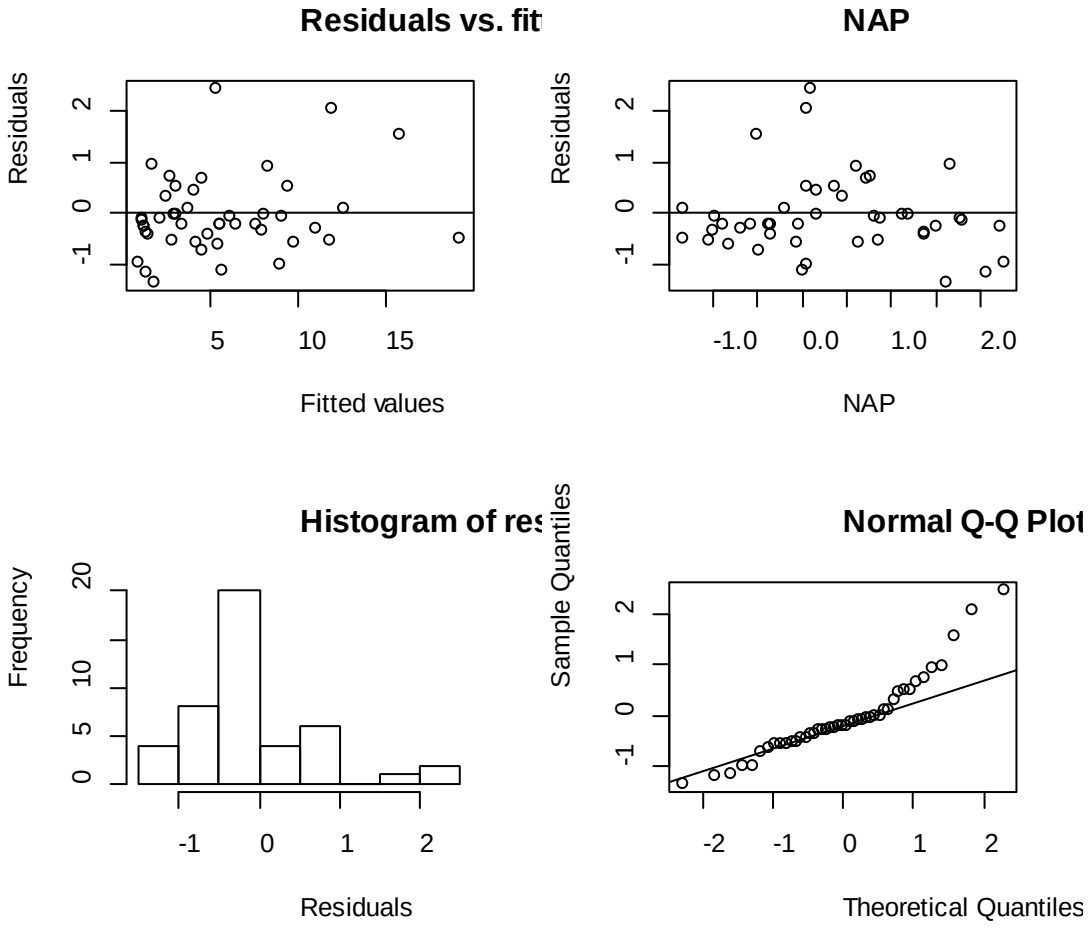
Random effects:
 Groups Name        Variance Std.Dev. Corr
 Beach  (Intercept) 0.2630   0.5128
        NAP         0.0891   0.2985   0.18
Number of obs: 45, groups:  Beach, 9

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.6942     0.1868   9.071  < 2e-16 ***
NAP          -0.6074     0.1374  -4.421 9.81e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
    (Intr)
NAP 0.121
```

En este caso vemos como la pendiente para el NAP cambia, debido a la función de vínculo logarítmica usada en la distribución Poisson. Si observamos ahora los residuos de este modelo veremos cómo su comportamiento ha mejorado.

```
> Res <- residuals(M4, type="pearson")
> Fit <- fitted(M4)
> par(mfrow=c(2,2))
> plot(Res ~ Fit, xlab="Fitted values", ylab="Residuals", main="Residu
als vs. fitted")
> abline(h=0)
> plot(Res ~ RIKZ$NAP, xlab="NAP", ylab="Residuals", main = "NAP")
> abline(h=0)
> hist(Res, main="Histogram of residuals", xlab="Residuals")
> qqnorm(Res)
> qqline(Res)
```



8. Lecturas recomendadas

i. Bolker MB (2008) Ecological models and data in R. Princeton University.

ii. Cayuela L (2011) Introducción al diseño de experimentos

iii. Cayuela L (2014) Modelos lineales mixtos (LMM) y modelos lineales generalizados mixtos (GLMM) en R

iv. Cayuela L (2015) Modelos lineales generalizados.

v. Crawley MJ (2013) The R Book. Second edition. Wiley.

vi. Ieno EN, Zuur AF (2015) A beginner's guide to data exploration and visualization with R. Highland Statisitcs Ltd

vii. Quinn GP, Keough MJ (2002) Experimental design and data analysis for biologists. Cambridge University Press.

viii. Zuur AF, Ieno EN, Smith GM (2007) Analysing ecological data. Springer, New York

ix. Zuur AF, Ieno EN, Walker NJ, Saveliev AA, Smith GM (2009) Mixed effects models and extensions in ecology with R. Springer New York

x. Zuur AF, Hilbe JM, Ieno EN (2013) A beginner's guide to GLM and GLMM with R. Highland Statisitcs Ltd