

2015

Análisis de datos selvícolas con R



Felipe Bravo (Coord)



Instituto Universitario de
Investigación en Gestión Forestal
Sostenible
ETS DE INGENIERÍAS AGRARIAS
UNIVERSIDAD DE VALLADOLID

Autores:

Coordinación: Felipe Bravo

Textos: Felipe Bravo, Celia Herrero e Irene Ruano

Programas: Felipe Bravo, Andrés Bravo-Núñez, Celia Herrero, Wilson Lara y José G. Riofrío

Cita Recomendada

Bravo, F., Herrero, C., Ruano, I., Bravo-Núñez, A., Lara, W., Riofrío, J.G. 2015. Análisis de datos selvícolas con R. Universidad de Valladolid

Este documento es resultado de diversos proyectos de innovación educativa (PID) de la Universidad de Valladolid: Un bosque de números (cursos 2013-2014 y 2014-2015)

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual Internacional 4.0



Índice

Introducción.....	1
Fundamentos para comprender e instalar R.....	1
Aspectos básicos para utilizar los programas de R.	2
Casos Prácticos: Esquema de trabajo	13
Caso Práctico 1: Importación de datos.....	14
Caso Práctico 2: Análisis descriptivo de datos.....	16
Caso Práctico 3: Análisis gráfico de datos	19
Caso Práctico 4: Análisis exploratorio de datos.....	22
Caso Práctico 5: Estudio de la influencia sobre la producción en volumen maderable de la calidad de estación y la localidad de plantaciones de chopo.....	26
Caso Práctico 6: Cálculo de variables dasométricas a partir de datos dendrométricos.....	29
Caso Práctico 7: Gráfico de evolución de altura dominante frente a la edad	33
Caso Práctico 8: Gráfico de crecimientos corrientes de <i>Pinus halepensis</i> en España.....	35
Caso Práctico 9: Cálculo del Turno de máxima renta en especie	37
Caso Práctico 10: Gráfico de evolución de un rodal a partir de una tabla de producción	39
Caso práctico 11. Ajuste y dibujo de curvas de calidad (modelos de Hossfeld-I y de Bertalanffy-Richards)	42
Caso Práctico 12: Síntesis de información edáfica mediante análisis de componentes principales (PCA).....	54
Caso Práctico 13: Clasificación de parcelas por índice de sitio a partir de datos edáficos (<i>soil-site method</i>) mediante análisis discriminante	57
Caso Práctico 14: Ajuste de ecuaciones de biomasa	60
Para saber más	68

Introducción

La cuantificación de la selvicultura permite medir y comparar los tratamientos selvícolas y sus efectos sobre los bosques. Así, dos conceptos a menudo distantes para la mayoría de las personas como son los bosques y los números pueden hacernos reflexionar sobre diversos aspectos de interés para la gestión de los bosques. Entre otras preguntas que podemos hacernos destacan dos:

1. ¿Tiene sentido cuantificar los procesos que ocurren en los bosques?
2. ¿Qué aporta la cuantificación de los procesos forestales a la toma de decisiones?

Si la respuesta a la primera pregunta es sí y a la segunda podemos responder diciendo que lo que aporta la cuantificación es relevante para la selvicultura, entonces debemos disponer de herramientas avanzadas para el análisis forestal. Una de estas herramientas es el programa informático de análisis estadístico R.

Con este manual se pretende dotar de una herramienta de apoyo a las clases de grado y máster relacionadas con los aspectos cuantitativos de la selvicultura.

Fundamentos para comprender e instalar R

R es un lenguaje adecuado para análisis estadístico y diseño avanzado de gráficos. R se puede descargar desde la página <http://www.r-project.org/> Lo primero será seleccionar el ‘mirror’ más adecuado para que la instalación sea rápida. En nuestro caso es <http://cran.es.r-project.org/>



La principal característica de R es que permite programar todos los pasos asociados a los procesos estadísticos y a la elaboración de gráficos lo que facilita un control total al usuario. Esto hace que el sistema sea muy potente pero a la vez hace que el usuario deba ser muy cuidadoso al elaborar sus ‘scripts’ ya que cualquier cosa que se no se indique expresamente no será ejecutada por el programa. El programa R puede instalarse desde estén enlace: <http://www.r-project.org/> Para acceder a manuales de R con las últimas novedades es conveniente visitar este enlace: <http://cran.r-project.org/manuals.html> Además hay muchos enlaces con ayudas y ejemplos de programas (en algunos casos es necesario tener una base sólida antes de comprender los programas que se muestran). A continuación se expone una lista de enlace de ayudas y ejemplos:

- Quick R: <http://www.statmethods.net/>
- R Programming: <http://manuals.bioinformatics.ucr.edu/home/programming-in-r>

Además debemos instalar el programa RStudio (figura 1) que es un IDE (*Integrated Development Environment*) que permite programar en R. RStudio es un programa *open source* que se puede descargar gratis de la siguiente dirección: <http://www.rstudio.com/> Hay otros IDE (como Eclipse: <http://www.eclipse.org/> RkWard: <http://rkward.sourceforge.net/> o RCommander: <http://www.rcommander.com/>) que pueden resultar de interés pero nosotros por su sencillez nos centraremos en RStudio.

Además de instalar R y RStudio, conviene instalar un editor de textos avanzado como Notepad ++ que permite generar códigos para programas informáticos de forma sencilla. Notepad ++ puede descargarse de forma gratuita de <http://notepad-plus-plus.org/>

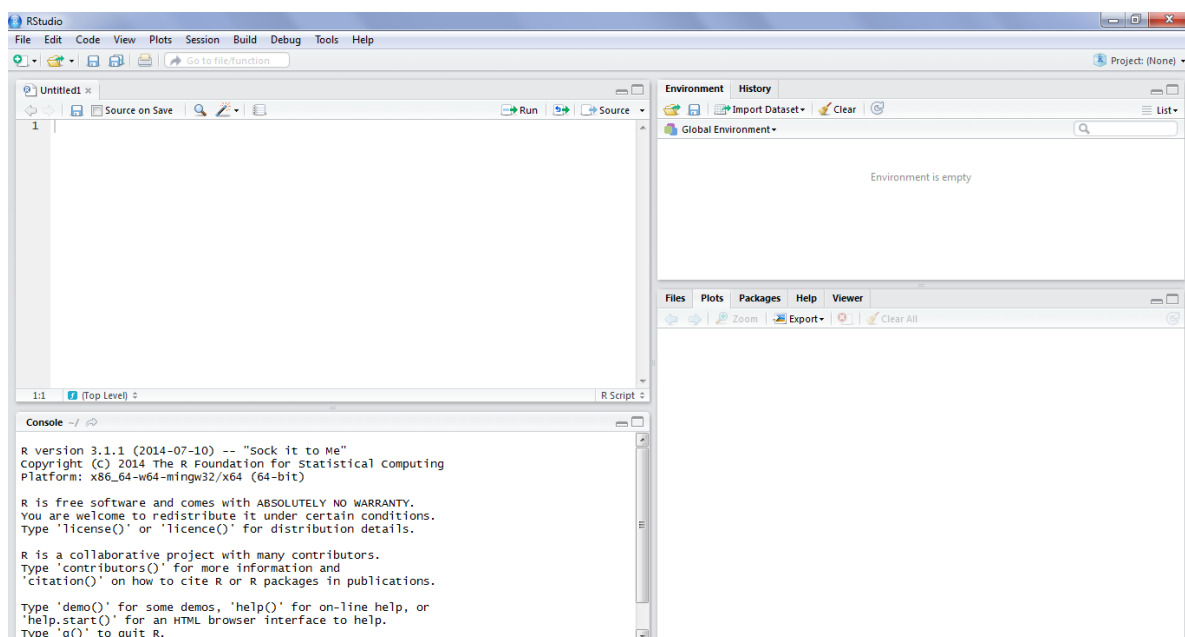


Figura 1. Imagen que muestra las pantallas de desarrollo de RStudio

Aspectos básicos para utilizar los programas de R.

R es un sistema de programación orientada a objetos por tanto siempre debemos definir los objetos. R solo nos devuelve lo que le pedimos por tanto hay que ser cuidadoso para solicitar todo lo que precisemos. Algunas instrucciones básicas que nos ayudarán a programar de forma eficaz

```
# significa que lo que sigue es un comentario  
# y que el programa no lo considera comando  
# es muy útil para comentar el programa y entender qué se  
# está haciendo
```

```
?*** Nos conecta a internet y nos busca la ayuda sobre el  
comando (***) que hayamos escrito
```

```
Al describir las rutas se debe poner / en lugar de \ y  
escribirlas siempre entre " "
```

```
<- significa =
```

```
R es 'case sensitive' por lo que debemos estar atentos a las  
mayúsculas y minúsculas así 'DATA' no es lo mismo que  
'data' ni que 'Data'
```

1. Documentos a abrir para trabajar en R.

Dado que vamos a trabajar con el IDE Rstudio, lo primero que debemos hacer es abrir este programa para comenzar la sesión y una vez abierto, abrir un Nuevo script (Archivo-Nuevo Script o File – New File - R Script), que será el documento de trabajo donde programaremos las tareas que deseemos realizar (fig 2)

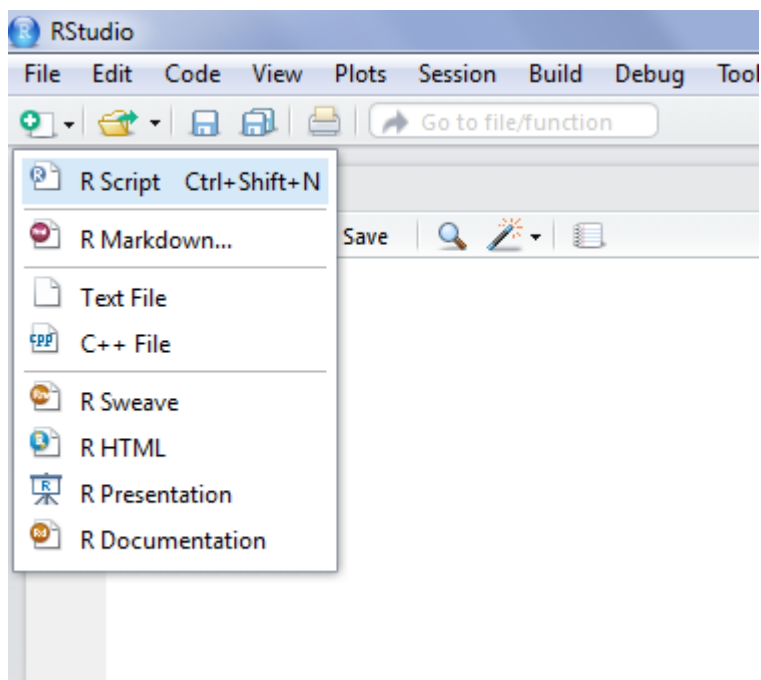



Fig. 2. Pantalla mostrando la ruta para abrir un nuevo script

Todas las órdenes que se escriban en el Nuevo Script se correrán en R, tecleando Control-Intro o el icono 'Run' [] (solo se correrán las líneas o secciones seleccionadas). Ahora ya estamos listos para comenzar a programar en R. Veamos algunas instrucciones básicas que nos serán de utilidad.

2. Instalar y requerir librerías.

Algunos de los programas necesitan instalar alguna librería. En cada programa se informará de las librerías a instalar (Vease: Esquema de trabajo). En estos casos, el programa contará con una orden denominada INSTALL (para instalar la librería) y dentro del programa REQUIRE (para llamarla).

Así, para instalar la librería 'plyr':

```
install.packages('plyr')
```

para llamarla:

```
require('plyr')
```

o

```
library('plyr')
```

El script para instalar y utilizar una librería será entonces:

```
install.packages('plyr')
require('plyr')
```

En muchos casos hay que instalar y llamar (requerir) librerías para realizar determinados procedimientos en R. Solo hay que instalar las librerías la primera vez (aunque cuando se actualiza la versión de R hay que volver a instalar las librerías) pero hay que llamarlas cada vez que se ejecute el programa. Todas las librerías disponibles en R pueden consultarse en <http://cran.r-project.org/web/packages/>

3. Manejo de datos en R.

El manejo de los datos es una de las tareas más arduas de aprender pero a la vez permite que una vez se conoce el trabajo con R resulte mucho más fácil. En este manual: <http://cran.es.r-project.org/doc/manuals/R-data.html> se pueden consultar los procedimientos para manejar bases de datos en R

3.1 Importar datos a R.

Lo normal es tener una base de datos asociada (BD) que puede contener por ejemplo: datos de inventario, pesadas de una balanza,... y que puede estar en Excel, en formato delimitados por coma,.... Para ello, lo primero que

tenemos que hacer es importar los datos a R. Es aconsejable disponer de un directorio de trabajo que nos permita referir todos los archivos a abrir o guardar a una misma ruta. Por ejemplo, creemos una carpeta en C:\ que se llame datos (en este paso cada usuario puede crear la carpeta donde crea más oportuno). En esta carpeta tendremos todas las bases de datos asociadas a los ejercicios de este manual. Los nombres de las variables no deben contener espacios y es aconsejable que solo tengan caracteres alfanuméricos (letras y números)

Todas las bases de datos que se usan este manual están disponibles en http://sostenible.palencia.uva.es/sites/default/files/manuales/datos_0.zip

El primer paso será crear el espacio de trabajo asociado a esa carpeta. Es decir si no se especifica otra cosa, R entenderá que los archivos que se importen o exporten estarán referidos a la carpeta de trabajo. La instrucción que utilizaremos es `setwd` (de “set working directory” o establecer directorio de trabajo):

```
setwd('C:/datosR')
```

Como ya se comentó antes hay ser estricto con la nomenclatura (R es ‘case sensitive’) y utilizar comillas y slash hacia la derecha (/) así como llamar a la carpeta (y la ruta) exactamente igual que en el ordenador.

Una vez que ya tenemos definido el directorio de trabajo podemos comenzar a importar los datos. R es muy flexible y permite importar datos de diferentes formatos. Sin embargo, para trabajar con los casos de este manual se aconseja que los datos se guarden formato csv delimitado por comas.

Los **bases de datos en formato csv** se importan mediante la siguiente instrucción:

```
data0<-read.csv('dendro.csv', sep=';', dec=',', header=TRUE)
```

x

En esta orden le estamos indicando que lea los datos llamados **dendro**, que nos separe los datos mediante (;), que los decimales los tiene con separación de comas (,) y que la primera fila contiene el encabezado de las variables (`header=TRUE`). Para comprobar que los datos importados lo han sido de forma correcta, utilizamos la orden

```
head(data0)
```

La instrucción `head` nos muestra los 6 primeros registros de la base de datos. Si queremos ver más tendremos que escribir `head(data0, n)` donde `n` es el número de registros que queramos visualizar.

El script o programa de lectura de datos (en formato csv) será entonces:


```
setwd('C:/datosR')
data0<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
head(data0)
```

Si los **datos** estuvieran en **formato txt**, separado por tabulaciones, las instrucciones anteriores serían:

```
data0<-read.table('dendro.txt', sep='\t',
dec=',',header=TRUE)
head(data0)
```

Se puede indicar otro formato de datos, otra separación de los datos y otra separación de los decimales pueden tener otras opciones, como

```
data<-read.table("DATOS1.txt", sep="\t", dec=".", header=TRUE)
```

Finalmente, se pueden leer datos directamente desde la web indicando el formato adecuado. Por ejemplo en el caso de datos en formato txt que estén en un servidor web la instrucción sería la siguiente:

```
data0 <-
read.table('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt', sep="\t", dec=".", header=TRUE)
```

y si están en otro formato bastaría con adaptar la extensión del archivo.

Las instrucciones anteriores pueden abreviarse creando un objeto que corresponda con la dirección web donde se encuentren los datos. Así tendremos:

```
Web <-
('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt')
data0 <- read.table('Web', sep='\t', dec='.', header=TRUE)
```

Se pueden importar archivos de excel (.xlsx o .xls) con las siguientes instrucciones:

```
library(xlsx)
data0 <- read.xlsx("c:/dendro.xlsx", sheetName = "hoja1")
```

3.2. Manejo de datos

Uno de los aspectos más interesantes en R es la facilidad para crear, modificar, agrupar,... variables y bases de datos. Conocer bien los métodos para hacer estas acciones facilita el trabajo de manipulación y análisis de datos.

Crear nuevas variables

En la base de datos dendro.txt (que debemos importar previamente) podemos generar nuevas variables mediante las siguientes instrucciones que combinan datos que están en el archivo Libro1a.csv (que también debemos importar previamente):

```
dendro$suma <- Libro1a$dato2 + Libro1a$dato3
dendro$media <- (Libro1a$dato2 + Libro1a$dato3)/2
```

O mediante la siguiente expresión para generar una nueva variable en Libro1a:

```
attach(Libro1a)
Libro1a$suma <- dato2 + dato3
Libro1a$media <- (dato2 + dato3)/2
detach(Libro1a)
```

Para crear nuevas variables (o para programar funciones) en R se pueden utilizar, entre otros los siguientes operadores:

Operador	Descripción
Aritméticos	
+	Suma
-	Resta
*	Multiplicación
/	División
^ o **	Exponente
Lógicos	
<	Menor que
<=	Menor o igual a
>	Mayor que
>=	Mayor o igual a
==	Exactamente igual a
!=	No igual a
x y	x o y (alternativamente)
x & y	x e y (adicionalmente)
isTRUE(x)	Comprueba si x es verdadero

Mezclar bases de datos

A partir de dos bases de datos que contienen al menos una variable en común se puede generar una nueva base de datos que contenga los datos de las dos anteriores. Con las bases de datos Libro1a.csv y Libro1b.csv (que debemos importar previamente) que tienen en común la variable 'dato1'

podemos generar una nueva base de datos que contendrá todos los datos de las dos bases de datos anteriores. Para ello utilizaremos la siguiente rutina:

```
setwd('C:/datosR')
datos1 <- read.table("Libro1a.csv", sep = ";",header = TRUE)
datos2 <- read.table("Libro1b.csv", sep = ";",header = TRUE)
datafinal <- merge(datos1, datos2)
head(datafinal, 10)
```

Si la variable por la que se quiere mezclar las bases de datos tuviese un nombre diferente en cada una de ellas habría que emplear la siguiente estructura:

```
datafinal <-
merge(datos1, datos2, by.datos1='nombre1',
by.datos2='nombre2')
```

4. Gráficos en R.

Cuando se hace un gráfico trabajando con el IDE RStudio, éste sale una nueva consola. Cuando haces más gráficos sólo aparece el último. Hay dos opciones, o guardar el gráfico con el formato (pdf, png, tiff, jpg,.) preferido, copiar en el portapapeles o ir al histórico, buscar el gráfico y guardarlo. Más adelante al mostrar los diferentes ejercicios se desarrollarán los programas para realizar distintos tipos de gráficos (puede obtenerse una muy buena introducción en la *R Graph Galery* (<http://rgraphgallery.blogspot.com.es/>))

5. Rutinas en R.

En algunos casos hay que hacer rutinas en R como por ejemplo bucles, condiciones o ambos a la vez, para generar funciones.

Bucle

Los bucles en R son lentos y deben evitarse pero en algunos casos no podemos evitar utilizarlos. Por ejemplo para un valor determinado de una variable se debe realizar una acción determinada. Esto se hace con la instrucción FOR. La estructura de un bucle es la siguiente:

```
for (variable in secuencia) {hacer cosas que dependan del valor de variable}
```

Por ejemplo, generemos una variable *i* con valores de 1 a 15 y luego una nueva variable *a* que es igual a la suma de 12 a cada uno de los valores de *i*. Finalmente mostramos en pantalla los valores de *a*.

```
for (i in 1:15)
{ a <- 12 + i
  print(a)}
```

Condición

En este caso esperamos que si se cumple un requisito hacer una acción y si no se cumple hacer otra. Si tenemos dos variables (a y b) que tienen valores 4 y 5 respectivamente y queremos que el programa muestre a si es igual o mayor que b y lo contrario si es menor. Esto se hace mediante la instrucción IF... ELSE con la siguiente estructura general:

```
if (condicion) comando1 else comando2
```

Por ejemplo definamos dos números a y b iguales a 4 y 5 respectivamente y finalmente hagamos que programa nos muestre a si es mayor o menor que b o b en caso contrario

```
a <- 4
b <- 5
if (a>=b){print(a)} else {print(b)}
```

Al ejecutar esta rutina el programa nos devolverá 5.

Bucle condicionado

Con un bucle condicionado lo que tratamos es que el programa ejecute una instrucción mientras se verifique una condición que le hemos fijado. Para esto utilizamos la instrucción WHILE con la siguiente estructura general:

```
while(condición) {hacer cosas siempre que se siga verificando la condición}
```

Vamos a calcular el primer número entero cuyo valor al cubo es superior a 1000. Para ello debemos definir primero dos variables (cubo y n) que vamos a hacer igual a cero y después iremos calculando el cubo de n hasta que el valor de cubo sea superior a mil. Finalmente el programa nos mostrará el primer valor de n cuando su valor al cubo es superior a mil. Esto lo podemos hacer con este programa en R:

```
cubo = 0
n = 0
while(cubo<1000)
{
  n<-n+1
  cubo<-n^3
}
n
cubo
```

Generar funciones

En muchas ocasiones nos interesa generar una nueva variable y así no tener que escribirla de forma repetida a lo largo del programa. Esto se puede hacer mediante la siguiente estructura:

```
nombre.funcion <- function(argumentos) {  
  valor <- expresión para definir valor función dependiendo de  
  argumentos return(valor)  
}
```

Como ejemplos podemos calcular las siguientes funciones:

Calcular factorial de un número

```
facto <- function(x){  
  z <- (1:x)  
  f <- exp(sum(log(z)))  
  return(f)  
}
```

Si ahora, queremos saber el factorial de 4, escribiríamos esta instrucción

```
facto(4)
```

y nos devolverá el valor 24

Calcular combinaciones

```
combi <- function(n,m){  
  c <- facto(n)/(facto(m)*facto(n-m))  
  return(c)  
}
```

Para saber las combinaciones de 4 y 2 escribiremos:

```
combi(4,2)
```

y nos devolverá el valor 6

Generar la función de Gompertz

Si vamos a utilizar un modelo crecimiento de forma reiterada nos puede interesar programarlo como una función para no tener que escribirla de forma repetida. A continuación está programada la función Gompertz en su forma integral:

$$f(x) = e^{(Lny_0)} * e^{-r*x}$$

luego para programarla escribiremos

```
Gompertz <- function(y0,r){
  exp(log(y0)*exp(-r*x))
}
```

Si ahora queremos saber el valor de la función de Gompertz para un valor conocido de x (2) , y0 (0.0017) y de r (0.02922) escribiremos:

```
x <- 2
Gompertz(0.0017,0.02922)
```

y R nos devolverá: 0.002441535

Generar la función de Weibull

Es posible que en algunos casos precisemos definir la distribución de una variable x (por ejemplo el diámetro) utilizando la función de densidad de Weibull (con parámetros de localización, a , de escala, b y de forma, c)

$$f(x) = \frac{c}{b} \left(\frac{x-a}{b} \right)^{c-1} e^{-((x-a)/b)^c}$$

Cuando x es mayor o igual a cero. En el caso de que x sea menor que cero, f(x) es igual a cero.

Para generar esta función (asumiendo que x no tomará valores negativos y que el parámetro de localización es 0) en R escribiremos:

```
Weibull <- function(b,c){
  (c/b)*((x/b)^(c-1))*exp(-(x/b)^c)
}
```

Esta función nos devolverá en tanto por uno la proporción de la distribución que corresponde a cada valor de x. Supongamos que x representa el diámetro y que queremos saber, en un rodal cuyo diámetro máximo es 15 cm, cuántos individuos hay por clase diamétrica de 1 cm si sabemos que el parámetro de forma es 3.6, el de escala 9 y el de localización 0. Para ello escribiremos el siguiente código:

```
x <- c(1:15)
Weibull(3.6,9)
```

y obtendremos los siguientes valores:

```
[1] 0.001320900 0.007975805 0.022554036 0.046020050 0.076915726  
[6] 0.110492027 0.138850344 0.153058034 0.147151776 0.122014961  
[11] 0.085953694 0.050525891 0.024284564 0.009332920 0.002799780
```

Lo que quiere decir que de la clase 6 cm hay un 11,05 % del total de árboles del rodal.

6. Invocar scripts en R.

En muchos casos precisaremos repetir un mismo programa dentro de otro que estemos ejecutando. Para poder hacer esto se utiliza la instrucción `source()` indicando entre paréntesis el nombre del programa y, si es preciso, la ruta donde se encuentra.

```
setwd('C:/scriptsR')  
source('programa.R')
```

Casos Prácticos: Esquema de trabajo

En cada caso práctico se van a indicar los siguientes puntos:

1. Objetivo del caso práctico y de su programa asociado
2. Base de datos asociada.
Se indicará si tiene una base de datos asociada, en qué formato se encuentra y se describirán los campos de la misma, indicando el nombre de las variables y las unidades de medida.
3. Instalación de Librerías.
Se indicarán qué librerías deben instalarse para el programa:
4. Pasos a seguir en el programa.
Se indicarán los pasos a seguir en el programa, donde aparecerán las órdenes de R y donde podrán aparecer otras órdenes de interés.
5. Programa de R (script).
En este apartado se incluirá el programa tal y como debe ser ejecutado en R (sin comentarios didácticos que no se anoten con #)
El texto de este punto se puede aplicar directamente en R pues las frases que no son órdenes de R y que son meramente descriptivas vienen precedidas de una almohadilla, por lo que R las ignora. Es decir que todo lo que va en este apartado se puede copiar y pegar directamente en el nuevo script que abrimos al comenzar la sesión.

No se incluyen los detalles de los procedimientos estadísticos que se aplican en cada caso ya que se asume un conocimiento adecuado de los mismos

Caso Práctico 1: Importación de datos

1. Objetivo del programa.

El objetivo de este programa es conocer cómo se importa una base de datos en formato csv a R.

2. Base de datos asociada.

Sí. Se trata de un archivo csv denominado “dendro” en el que aparecen los campos siguientes:

P2: Parcela muestreada.

num: número del árbol muestreado.

d1 y d2: diámetros en cruz a 1,30 m del suelo medidos en mm.

Htotal y Hcopa: Altura total y de la copa del árbol medidas en m.

El archivo se debe copiar en la ruta donde se establezca el ‘*working directory*’ (en nuestro caso utilizaremos C:\datosR)

3. Instalación de Librerías.

No hace falta instalar ninguna librería

4. Pasos a seguir en el programa.

PASO 1. Creamos el espacio de trabajo

```
setwd('C:/datosR')  
#Se pueden visualizar los archivos de dicho fichero  
#con la función dir()
```

PASO 2. Se selecciona el archivo en dicho fichero y mandamos a R leerlo

```
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
```

sep nos indica con que queremos que separe los datos

en este caso (sep=';') es un punto y coma.

dec indica la forma de separación de decimales

header indica si los datos tienen en el encabezado el nombre de las variables. si no los tuviera lo indicaríamos como header=FALSE

PASO 3. Visualizar datos.

```
# Visualiza en la consola el contenido de los encabezados y  
#los diez primeros registros del fichero importado  
head(data, 10)
```

```
# Visualiza en la consola el contenido del fichero importado,  
#con forma de tabla  
View(data)
```

5. Programa de R (script)

```
setwd('C:/datosR')  
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)  
head(data, 10)  
View(data)
```

Caso Práctico 2: Análisis descriptivo de datos

1. Objetivo del programa.

El objetivo de este programa es realizar un análisis descriptivo de un conjunto de datos.

2. Base de datos asociada.

Sí. Se trata de un archivo csv denominado “dendro” en el que aparecen los campos siguientes:

P2: Parcela muestreada.

num: número del árbol muestreado.

d1 y d2: diámetros en cruz a 1.30 m del suelo medidos en mm.

Htotal y Hcopa: Altura total y de la copa del árbol medidas en m.

3. Instalación de Librerías.

No

4. Pasos a seguir en el programa.

PASO 1. Leemos los datos

```
setwd('C:/datosR')  
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
```

PASO 2. Se calculan las principales medidas descriptivas de un conjunto de datos.

```
#media de cada columna  
colMeans(data, na.rm=TRUE)  
#para medias por filas  
rowMeans(data, na.rm=TRUE)
```

```
#si queremos sumar los datos de las columnas o las filas  
rowSums(data, na.rm=TRUE)  
colSums(data, na.rm=TRUE)
```

```
#el comando dim te da el número de filas y de columnas  
dim(data)  
#el comando str te da información sobre el tipo de caracteres  
#que tiene cada columna y la tabla como en “diagrama de  
#hojas”.  
str(data)
```

```
#la mediana se puede conseguir así, donde tienes que
#especificar la variable y se aplica para cada una de las
#columnas
median(data$d1, na.rm=TRUE)
```

```
#o con el comando sapply , que es para aplicar en una tabla
#una función determinada
sapply(data, na.rm=TRUE, median)
#otras variables descriptivas se pueden calcular también con
#sapply como la desviación típica, la varianza y el rango
#Desviación típica
sapply(data, na.rm=TRUE, sd)
#Varianza
sapply (data, na.rm=TRUE, var)
#Rango
sapply (data, na.rm=TRUE, range)
```

```
#summary() calcula el mínimo, el máximo, la media,
#la mediana, el primer cuartil y el tercer cuartil.
summary(data, na.rm=TRUE)
```

5. Programa de R (script)

```
setwd('C:/datosR')
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)

#media de cada columna
colMeans(data, na.rm=TRUE)
#para medias por filas
rowMeans(data, na.rm=TRUE)

#si queremos sumar los datos de las columnas o las filas
rowSums(data, na.rm=TRUE)
colSums(data, na.rm=TRUE)

#el comando dim te da el número de filas y de columnas
dim(data)
#el comando str te da información sobre el tipo de caracteres
#que tiene cada columna y la tabla como en “diagrama de
#hojas”.

str(data)

#la mediana se puede conseguir así, donde tienes que
#especificar la variable y se aplica para cada una de las
#columnas

median(data$d1, na.rm=TRUE)
```

```
#o con el comando sapply, que es para aplicar en una tabla
#una función determinada

sapply(data, na.rm=TRUE,median)

#otras variables descriptivas se pueden calcular también con
#sapply como la desviación típica, la varianza y el rango

#Desviación típica
sapply(data, na.rm=TRUE,sd)

#Varianza
sapply (data, na.rm=TRUE, var)

#Rango
sapply (data, na.rm=TRUE, range)

#summary() calcula el mínimo, el máxim, la media,
#la mediana, el primer cuartil y el tercer cuartil.
summary(data, na.rm=TRUE)
```

Caso Práctico 3: Análisis gráfico de datos

1. Objetivo del programa.

El objetivo de este programa es conocer los comandos de R para poder realizar un análisis gráfico de un conjunto de datos.

2. Base de datos asociada.

Sí. Se trata de un archivo csv denominado “dendro” en el que aparecen los campos siguientes:

P2: Parcela muestreada.

num: número del árbol muestreado.

d1 y d2: diámetros en cruz a 1.30 m del suelo medidos en mm.

Htotal y Hcopa: Altura total y de la copa del árbol medidas en m.

3. Instalación de Librerías.

Sí. Debemos instalar la librería ‘lattice’

4. Pasos a seguir en el programa.

PASO 1. Instalamos la librería lattice (en R Console)

```
#Este paso requiere instalar la librería ‘lattice’:
install.packages('lattice')
```

```
#La llamamos para que se pueda ejecutar
require('lattice')
```

PASO 2. Leemos los datos

```
setwd('C:/datosR')
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)
```

PASO 3. En este paso haremos varios gráficos (nube de puntos, un boxplot, un scatter.smooth, un histograma y un contraste de pares entre variables).

Tipo de gráfico	Fución en R
Nube de puntos, líneas	plot()
Boxplot	boxplot()
Scatter somooth	scatter.smooth()
Histograma	histogram()
Contraste entre pares de variables	pairs()

Las funciones están definidas por distintos argumentos. Por ejemplo, en el caso de plot() algunos argumentos pueden ser col,main,type,sub,xlab,ylab

el argumento col define el Color de la gráfica
el argumento main define el título de la gráfica
el argumento type define el tipo de grafica a hacer
"p" para una nube de puntos
"l" para un gráfico de líneas
"b" para una gráfica de puntos y líneas
"c" para obtener una grafica de tipo "b" pero sin puntos
"h" para líneas verticales tipo histograma
"s" para gráficos de barras
"n" te indica el sistema de referencia del gráfico, pero sin gráfico
el argumento sub define un subtítulo debajo del título de los ejes x o y.
los argumentos xlab e ylab definen los títulos de cada uno de los ejes

Por ejemplo, si queremos hacer un gráfico exploratorio que compare los dos diámetros recogidos en el muestreo

data\$d1 indica a R que debe usar la variable d1 del conjunto de datos data

```
plot(data$d1,data$d2,col="red",main="Gráfico de los dos  
diámetros recogidos en el muestreo",  
ylab="d2(mm)", xlab="d1(mm)",type="p")
```

```
boxplot(d1~Htotal,data, ylab = "Htotal (m)", xlab= "d1")
```

Visualiza un diagrama de cajas de la variable Htotal por clases de d1

(d1~Htotal,data,...) es otra forma de indicar a R que variables de qué variables queremos usar

```
scatter.smooth(data$d2, data$Htotal, ylab = "Htotal (m)",  
xlab= "d2 (mm)", col="red")
```

Visualiza los datos en un gráfico exploratorio añadiendo una línea de tendencia suavizada con los puntos en rojo

```
require(lattice)  
histogram(~Htotal | d2, data)
```

Visualizar los datos (Htotal) en un histograma por clases de d2

```
pairs(~d1+d2+Htotal+Hcopa,data, main="Matriz de  
Scatterplots")
```

Visualiza los datos por pares de variables en una matriz de gráficos

5. Programa de R (script)

```
install.packages('lattice')
```

```
require('lattice')

setwd('C:/datosR')
data<-read.csv('dendro.csv',sep=';',dec=',',header=TRUE)

plot(data$d1,data$d2,col="red",main="Gráfico de los dos
diámetros recogidos en el muestreo",
      ylab="d2(mm)", xlab="d1(mm)",type="p")

boxplot(d1~Htotal,data, ylab = "Htotal (m)", xlab= "d1")

scatter.smooth(data$d2, data$Htotal, ylab = "Htotal (m)",
xlab= "d2 (mm)", col="red")

require(lattice)
histogram(~Htotal | d2, data)

pairs(~d1+d2+Htotal+Hcopa,data, main="Matriz de
Scatterplots")
```


Caso Práctico 4: Análisis exploratorio de datos

1. Objetivo del programa

En muchas ocasiones un análisis exploratorio de los datos nos puede dar ideas de cómo analizarlos o las relaciones que hay entre las variables. En este caso haremos un análisis de correlación y un gráfico de barras representando el error estándar de la media.

2. Base de datos asociada

Si. En este caso es un archivo excel denominado “biomasa.csv” en el que aparecen los siguientes campos:

Arbol: número identificativo del árbol

ht_m: altura total en metros

dbh: diámetro normal en cm

r_gruesas: peso seco de la parte considerada ramas gruesas (diámetro de 2 – 7 cm)

r_delgadas: peso seco de la parte considerada ramas delgadas (diámetro < 2 cm)

hojas: peso seco de la parte considerada hojas

fuste: peso seco de la parte considerada fuste (diámetro > 7 cm)

biomasatotal: peso seco de todas las partes

3. Instalación de librerías

Si. Debemos instalar las librerías “plyr”, “lattice”, “reshape2”, “ggplot2”.

1. Pasos a seguir en el programa

PASO 1. Instalar y cargar las librerías requeridas

Instalamos las librerías

```
install.packages('ggplot2')
install.packages('reshape2')
install.packages('lattice')
install.packages('plyr')
```

Las llamamos para que se puedan ejecutar

```
require('ggplot2')
require('reshape2')
require('lattice')
require('plyr')
```

PASO 2. Leemos los datos

```
datos<-read.csv2(C/'biomasa.csv',
sep=';',dec='.',header=T,na.string='NA')
```

PASO 3. Análisis de correlación de variables explicativas

```
xyplot(biomasatotal~dbh,data=datos,xlab="dbh(cm)",
ylab="Biomasa total(kg)",pch=20,cex=1,col="black")
```

Matriz de correlación de los datos

```
cor(datos)
```

Matriz de scatterplots

```
pairs(~ht_m+dbh+biomasatotal+fuste+r_gruesas+r_delgadas+ho
jas,datos, main="Matriz de Scatterplots")
```

PASO 4. Gráfico de barras representando el error estándar de la media. hay que modificar el formato del data frame a "long format", función "melt"

```
head(datos)
P_biomasa<-melt(datos,id.vars=c("Arbol"),
measure.vars=c("fuste","r_gruesas","r_delgadas","hojas","biomas
atotal"),
variable.name="component",value.name="biomasa")
head(P_biomasa)
```

Función q calcula la media,SD y error estandar SE y CI intervalo de confianza 95%

```
biom <- ddply(P_biomasa, c("component"), summarise,
N = sum(!is.na(biomasa)),mean = mean(biomasa,
na.rm=TRUE),
sd= sd(biomasa, na.rm=TRUE), se = sd / sqrt(N) )
head(biom,30)
```

Gráfico

```
ggplot(biom, aes(x=component, y=mean , fill=component))+
geom_bar()+
geom_bar(position=position_dodge(),colour="black",
show_guide=FALSE,
stat="identity")+ #posicion de barra de error
geom_errorbar(aes(ymin=mean-se, ymax=mean+se),
#barra de error
width=.5,position=position_dodge(.9))+
scale_fill_brewer(palette="Greys")+
ylab("Biomass(kg)")+ xlab("Biomass component")+
scale_x_discrete(breaks=c("fuste",
"r_gruesas","r_delgadas","hojas","biomasatotal"),
labels=c("Stem", "TB", "tb","Leaves","BT"))+
theme_bw()+ theme(legend.position="none",
panel.grid.major=element_blank(),#lineas de fondo
```

```
axis.text.y = element_text(size=14),
axis.title.y = element_text(size=14))
```

5. Programa de R (script)

```
# PASO 1. Instalar y cargar las librerías requeridas
# Instalar librerías necesarias
install.packages('ggplot2')
install.packages('reshape2')
install.packages('lattice')
install.packages('plyr')

# las llamamos para que se puedan ejecutar
require('ggplot2')
require('reshape2')
require('lattice')
require('plyr')

# PASO 2. Leemos los datos
datos<-read.csv2(C/'biomasa.csv',
sep='; ',dec='.',header=T,na.string='NA')

# PASO 3. Análisis de correlación de variables explicativas
xyplot(biomasatotal~dbh,data=datos,xlab="dbh(cm)",
ylab="Biomasa total(kg)",pch=20,cex=1,col="black")

# Matriz de correlación de los datos
cor(datos)

# Matriz de scatterplots
pairs(~ht_m+dbh+biomasatotal+fuste+r_gruesas+r_delgadas+hojas,datos, main="Matriz de Scatterplots")

# PASO 4.
#Gráfico de barras representando el error estándar de la media.
# modificar el formato del data frame a "long format"
# función "melt"
head(datos)
P_biomasa<-melt(datos,id.vars=c("Arbol"),
measure.vars=c("fuste","r_gruesas","r_delgadas","hojas","biomasa
total"),
variable.name="component",value.name="biomasa")
head(P_biomasa)

#funcion q calcula la media,SD y error estandar SE y
#CI intervalo de confianza 95%
biom <- ddply(P_biomasa, c("component"), summarise,
```

```
      N = sum(!is.na(biomasa)), mean = mean(biomasa,
na.rm=TRUE),
      sd= sd(biomasa, na.rm=TRUE), se = sd / sqrt(N) )
head(biom,30)

#Gráfico
ggplot(biom, aes(x=component, y=mean , fill=component))+
  geom_bar()+
  geom_bar(position=position_dodge(),colour="black",
show_guide=FALSE,
  stat="identity")+ #posicion de barra de error
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se),
#barra de error
  width=.5,position=position_dodge(.9))+
  scale_fill_brewer(palette="Greys")+
  ylab("Biomass(kg)")+ xlab("Biomass component")+
  scale_x_discrete(breaks=c("fuste",
"r_gruesas","r_delgadas","hojas","biomasatotal"),
  labels=c("Stem", "TB", "tb","Leaves","BT"))+
  theme_bw()+ theme(legend.position="none",
  panel.grid.major=element_blank(),#lineas de fondo
  axis.text.y = element_text(size=14),
  axis.title.y = element_text(size=14))
```

Caso Práctico 5: Estudio de la influencia sobre la producción en volumen maderable de la calidad de estación y la localidad de plantaciones de chopo

1. Objetivo del programa.

El objetivo de este programa es conocer si la calidad de estación y la localidad de diferentes plantaciones sobre la producción en volumen maderable. Para cumplir este objetivo haremos un estudio exploratorio de los datos mediante diagramas de cajas (*boxplot*) y análisis de la varianza sin y con interacciones entre los factores.

2. Base de datos asociada.

Sí. Se trata de un archivo de en formato txt denominado "DATOS3.txt en el que aparecen los campos siguientes:

LOC: Localidad de plantación

CALIDAD: Calidad de Estación

VOL: Volumen maderable en m³/ha

[Nota: no dejar espacios en la variable, que todas sus letras vayan juntas]

3. Instalación de Librerías.

No

4. Pasos a seguir en el programa.

PASO 1. Leemos los datos

```
setwd('C:/datosR')
```

```
chopos <- read.table("DATOS3.txt", header=TRUE, sep="\t",  
dec=".")
```

PASO 2. Visualizamos lo datos

```
head(chopos)
```

visualiza el encabezamiento y los primeros 6 registros de los datos

```
chopos
```

muestra toda la base de datos

PASO 3. Realizamos un diagrama de cajas (*boxplot*) a partir de los datos

Diagrama de cajas del volumen para calidades y localidades

```
boxplot(VOL ~ LOC*CALIDAD,data=chopos)
```

PASO4. Realizamos los análisis de la varianza sin y con interacciones entre los factores (localidad y calidad)

```
anova1 <- aov(VOL ~ LOC + CALIDAD, data =chopos)
summary(anova1)
```

Realiza y muestra el análisis de la varianza con modelo aditivo sin interacciones

```
anova2 <- aov(VOL ~ LOC + CALIDAD + LOC*CALIDAD,
data =chopos)
summary(anova2)
```

Realiza y muestra el análisis de la varianza con modelo aditivo con interacciones

```
anova3 <- aov(VOL ~ LOC*CALIDAD, data =chopos)
summary(anova3)
```

Realiza y muestra el análisis de la varianza con modelo aditivo con todas las interacciones posibles

5. Programa de R (script)

```
#PASO 1. Leemos los datos

setwd('C:/datosR')

chopos <- read.table("DATOS3.txt", header=TRUE, sep="\t",
dec=".")
# lee los datos

#PASO 2. Visualizamos lo datos

head(chopos)
# visualiza el encabezamiento y los primeros 6 registros de los
#datos

chopos
#muestra toda la base de datos

#PASO 3. Realizamos un diagrama de cajas (boxplot)

boxplot(VOL ~ LOC*CALIDAD,data=chopos)
# Diagrama de cajas del volumen para calidades y localidades

#PASO4. Realizamos los análisis de la varianza sin y con
#interacciones entre los factores (localidad y calidad)

anova1 <- aov(VOL ~ LOC + CALIDAD, data =chopos)
summary(anova1)
```

```
# Realiza y muestra el análisis de la varianza con modelo aditivo
#sin interacciones

anova2 <- aov(VOL ~ LOC + CALIDAD + LOC*CALIDAD, data
=chopos)
summary(anova2)
# Realiza y muestra el analisis de la varianza con modelo aditivo
#con interacciones

anova3 <- aov(VOL ~ LOC*CALIDAD, data =chopos)
summary(anova3)
# Realiza y muestra el análisis de la varianza con modelo aditivo
#con todas las interacciones posibles
```

Caso Práctico 6: Cálculo de variables dasométricas a partir de datos dendrométricos

1. Objetivo del programa.

El objetivo de este programa es calcular las variables dasométricas de un rodal a partir de datos dendrométricos de un inventario forestal.

2. Base de datos asociada.

Sí. Se trata de un archivo de Excel denominado “dendro” en el que aparecen los campos siguientes:

P2: Parcela muestreada.

num: número del árbol muestreado.

d1 y d2: diámetros en cruz a 1.30 m del suelo medidos en mm.

Htotal y Hcopa: Altura total y de la copa del árbol medidas en m.

[Nota: no dejar espacios en la variable, que todas sus letras vayan juntas]

3. Instalación de Librerías.

Sí. (en *R Console*)

```
install.packages('plyr')
```

4. Pasos a seguir en el programa.

PASO 1. Instalamos la librería plyr

Este paso requiere correr la orden:

```
install.packages('plyr')
```

La llamamos

```
require('plyr')
```

PASO 2. Leemos los datos

```
setwd('C:/datosR')
data<-read.csv('dendro2.csv',sep=';',dec=',',header=TRUE)
```

PASO 3. Calculamos las principales variables dasométricas

Pasos para el cálculo de G (m²); N (pies/ha) y dg (m)

```
area<-900#[m2]
data['dm_m']<- with(data,
(1/1000)*(d1+d2)/2,na.rm=TRUE)#[m]
data<-data[data$dm_m>=0.07,]
data['dm2_m2']<- with(data,dm_m^2)#[m^2]
data['Abi_m2']<- with(data,pi*(dm_m^2)/4)#[m^2]
data['G_m2']<- with(data,Abi_m2*10000/area)# [m^2]
means<- ddply(data,.(P2),summarise, dm_m =
mean(dm_m,na.rm=TRUE),dm2_m2 =
```



```
mean(dm2_m2,na.rm=TRUE),G_m2 =
mean(G_m2,na.rm=TRUE),Hmean_m=
mean(Htotal,na.rm=TRUE))
```

```
freq<- data.frame(table(data$P2))
d_summ<- merge(means,freq,by.x='P2',by.y='Var1')
# merge(freq,means,by.x='Var1',by.y=P2)
d_summ['pies_ha']<- with(d_summ,floor(Freq*10^4/area))
#ver ceiling
d_summ['dg_m']<-
with(d_summ,100*(dm2_m2/Freq)*0.5,na.rm=TRUE)
```

Pasos para el cálculo de Hm (m) y Hdom (m)

```
assman<-area*100/10^4
# número de individuos dominantes por parcela según
#Assman
hd_data<- data[,c('P2','dm_m','Htotal')]
hd_data<-
hd_data[with(hd_data,order(P2,dm_m,decreasing=TRUE)),]
hd_fun<-function(hd_data){
mean(hd_data[1:assman,'Htotal']) }
hd_summ<-
ddply(hd_data,.(hd_data$P2),function(data)hd_fun(data))
colnames(hd_summ)<-c('P2','Hdom_m')
```

Pasos para el cálculo de Ddom (m)

```
dA_data<- data[,c('P2','dm_m','Abi_m2')]
dA_data<-
dA_data[with(dA_data,order(P2,Abi_m2,decreasing=TRUE)),
]
```

head(dA_data)

```
dA_fun<-function(dA_data){
mean(dA_data[1:assman,'dm_m'],na.omit=TRUE)
}
dA_summ<-
ddply(dA_data,.(dA_data$P2),function(data)dA_fun(data))
colnames(dA_summ)<-c('P2','Ddom_m')
```

Obtención de los resultados y exportación de los mismos

```
dAsumm<-merge(hd_summ,dA_summ)
summary<-merge(d_summ,dAsumm)
setwd('C:/datosR')
write.table(summary,'summary.txt',col.names=TRUE,row.names=FALSE)
summary
```

5. Programa de R (script)

```

#PASO 1. Instalamos la librería plyr
#Este paso requiere correr la orden:
install.packages('plyr')
#Hay que seleccionar en el CRAN MIRROR, Spain.

#La llamamos
require('plyr')

#PASO 2. Leemos los datos
setwd('C:/datosR')
data<-read.csv('dendro2.csv',sep=';',dec=',',header=TRUE)

#PASO 3. Calculamos las principales variables dasométricas

#Pasos para el cálculo de G (m2); N (pies/ha) y dg (m)
area<-900#[m2]
data['dm_m']<- with(data, (1/1000)*(d1+d2)/2,na.rm=TRUE)#[m]
data<-data[data$dm_m>=0.07,]
data['dm2_m2']<- with(data,dm_m^2)#[m^2]
data['Abi_m2']<- with(data,pi*(dm_m^2)/4)#[m^2]
data['G_m2']<- with(data,Abi_m2*10000/area)# [m^2]
means<- ddply(data,.(P2),summarise, dm_m =
mean(dm_m,na.rm=TRUE),dm2_m2 =
mean(dm2_m2,na.rm=TRUE),G_m2 =
mean(G_m2,na.rm=TRUE),Hmean_m=
mean(Htotal,na.rm=TRUE))

freq<- data.frame(table(data$P2))
d_summ<- merge(means,freq,by.x='P2',by.y='Var1')
# merge(freq,means,by.x='Var1',by.y=P2)
d_summ['pies_ha']<- with(d_summ,floor(Freq*10^4/area)) #ver
ceiling
d_summ['dg_m']<-
with(d_summ,100*(dm2_m2/Freq)*0.5,na.rm=TRUE)

#Pasos para el cálculo de Hm (m) y Hdom (m)

assman<-area*100/10^4
# número de individuos dominantes por parcela según Assman
hd_data<- data[,c('P2','dm_m','Htotal')]
hd_data<-
hd_data[with(hd_data,order(P2,dm_m,decreasing=TRUE)),]
hd_fun<-function(hd_data){
mean(hd_data[1:assman,'Htotal']) }
hd_summ<-
ddply(hd_data,.(hd_data$P2),function(data)hd_fun(data))
colnames(hd_summ)<-c('P2','Hdom_m')

```

```
#Pasos para el cálculo de Ddom (m)
dA_data<- data[,c('P2','dm_m','Abi_m2')]
dA_data<-
dA_data[with(dA_data,order(P2,Abi_m2,decreasing=TRUE)),]
# head(dA_data)
dA_fun<-function(dA_data){
mean(dA_data[1:assman,'dm_m'],na.omit=TRUE)
}
dA_summ<-
ddply(dA_data,.(dA_data$P2),function(data)dA_fun(data))
colnames(dA_summ)<-c('P2','Ddom_m')

#Obtención de los resultados y exportación de los mismos
dAsumm<-merge(hd_summ,dA_summ)
summary<-merge(d_summ,dAsumm)
setwd('C:/datosR')
write.table(summary,'summary.txt',col.names=TRUE,row.names=
FALSE)
summary
```

Caso Práctico 7: Gráfico de evolución de altura dominante frente a la edad

1. Objetivo del programa.

El objetivo de este programa es realizar un gráfico con una línea que represente la altura dominante frente a la edad.

2. Base de datos asociada.

No.

3. Instalación de Librerías. No.

4. Pasos a seguir en el programa.

PASO 1. Creamos los datos

Se crea una secuencia regular con la función seq
t indicará el tiempo en años

```
t<-seq(from=20,to=70,by=5)
```

los argumentos de la función seq son

from que indica desde que número empezará la secuencia

to que indica el número en el que finalizara la secuencia

by que indica la distancia entre números consecutivos de la secuencia

de esta forma `t<-seq(from=20,to=70,by=5)` significa que a t se le asignan los valores de una secuencia que va desde el 20 al 70 de 5 en 5.

Hdom indicará la altura dominante

```
Hdom<-c(7,8.7,10.1,11.4,12.6,13.6,14.4,15.2,15.9,16.5,17.1)
```

PASO 2. Realizamos la gráfica

Realizamos una gráfica de la altura dominante frente al tiempo

```
plot(t,Hdom,type="l",col="black",main="altura dominante  
frente a la edad",ylab="Hdom(m)"  
,xlab="Edad(años)")
```

5. Programa de R (script)

```
#Se crea una secuencia regular con la función seq  
#t indicará el tiempo en años  
t<-seq(from=20,to=70,by=5)  
  
#Hdom indicará la altura dominante  
Hdom<-c(7,8.7,10.1,11.4,12.6,13.6,14.4,15.2,15.9,16.5,17.1)
```

```
# Realizamos la gráfica  
  
plot(t,Hdom,type="l",col="black",main="altura dominante frente a  
la edad",ylab="Hdom(m)"  
      ,xlab="Edad(años)")
```

Caso Práctico 8: Gráfico de crecimientos corrientes de *Pinus halepensis* en España

1. Objetivo del programa.

El objetivo de este programa es realizar un gráfico con las curvas de crecimiento corriente correspondiente a las cuatro calidades de estación representadas por las tablas de producción para *Pinus halepensis* desarrolladas por Montero et al (2001)

2. Base de datos asociada.

No.

3. Instalación de Librerías.

No.

4. Pasos a seguir en el programa.

PASO 1. Creamos los datos

Se crea una secuencia regular con la función seq
t indicará el tiempo en años

```
t<-seq(from=20,to=120,by=10)
```

generamos los datos de crecimiento corriente para las calidades 11, 14, 17 y 20 de acuerdo con las tablas de producción de Montero et al (2001)

```
CC20 <- c(0,5.4,5.3,4.3,3.7,3.0,2.6,2.2,1.8,1.4,1.2)
CC17 <- c(0,4.3,4.2,3.4,3.1,2.7,2.4,2,1.6,1.3,1.1)
CC14 <- c(0,2.9,3.3,3.0,2.9,2.4,1.9,1.6,1.3,1.1,0.9)
CC11 <- c(0,1.7,2.1,1.9,1.8,1.5,1.3,1.0,0.9,0.7,0.6)
```

PASO 2. Realizamos la gráfica

Creamos un gráfico con dos curvas (CM y CC respecto al tiempo)

```
plot(t,CC20,type="l",col="black",main="crecimiento corriente
frente a la edad/claras moderadas",ylab="CC(m3/ha)"
,xlab="Edad(años)")
```

añadimos las líneas correspondientes al resto de calidades

```
lines(t,CC17, type="l",col="red")
lines(t,CC14, type="l",col="blue")
lines(t,CC11, type="l",col="green")
```

5. Programa de R (script)

```
#PASO 1. Creamos los datos

#Se crea una secuencia regular con la función seq
#t indicará el tiempo en años

t<-seq(from=20,to=120,by=10)

# generamos los datos de crecimiento corriente para las calidades
# 11, 14, 17 y 20 de acuerdo con las tablas de producción de
#Montero et al (2001)

CC20 <- c(0,5.4,5.3,4.3,3.7,3.0,2.6,2.2,1.8,1.4,1.2)
CC17 <- c(0,4.3,4.2,3.4,3.1,2.7,2.4,2.1,1.6,1.3,1.1)
CC14 <- c(0,2.9,3.3,3.0,2.9,2.4,1.9,1.6,1.3,1.1,0.9)
CC11 <- c(0,1.7,2.1,1.9,1.8,1.5,1.3,1.0,0.9,0.7,0.6)

#PASO 2. Realizamos la gráfica

# Creamos un gráfico con dos curvas (CM y CC respecto al tiempo)

plot(t,CC20,type="l",col="black",main="crecimiento corriente
frente a la edad/claras moderadas",ylab="CC(m3/ha)"
      ,xlab="Edad(años)")
#añadimos las líneas correspondientes al resto de calidades

lines(t,CC17, type="l",col="red")
lines(t,CC14, type="l",col="blue")
lines(t,CC11, type="l",col="green")
```

Caso Práctico 9: Cálculo del Turno de máxima renta en especie

1. Objetivo del programa.

El objetivo de este programa es realizar un gráfico con dos curvas superpuestas. Las dos curvas representarán el crecimiento medio y corriente de una masa forestal frente a la edad.

2. Base de datos asociada.

No.

3. Instalación de Librerías. No.

4. Pasos a seguir en el programa.

PASO 1. Generamos los datos

Se crea una secuencia regular con la función seq
t indicará el tiempo en años

```
t<-seq(from=20,to=70,by=5)
```

los argumentos de la función seq son
from que indica desde que número empezará la secuencia
to que indica el número en el que finalizara la secuencia
by que indica la distancia entre números consecutivos de la secuencia

de esta forma t<-seq(from=20,to=70,by=5) significa
que a t se le asignan los valores de una secuencia que va
desde el 20 al 70 de 5 en 5.

Generamos los datos correspondientes al crecimiento medio (CM) y al
crecimiento corriente (CC)

```
CM<-c(1.7,2,2.3,2.5,2.6,2.7,2.7,2.7,2.7,2.6,NA)  
CC<-c(NA,3.6,3.7,3.6,3.4,3.1,2.9,2.7,2.4,2.2,NA)
```

NA quiere decir que ese dato no está disponible

PASO 2. Realizamos la gráfica

Creamos un gráfico con dos curvas (CM y CC respecto al tiempo)

```
plot(t,CC,type="l",col="black",main="Turno de máxima renta  
en especie"  
  ,sub="Betula alba/Galicia",xlab="Edad(años)"  
  ,ylab="Crec.(m^3/ha y años)")  
lines(t,CM, type="l",col="red")
```


5. Programa de R (script)

```
#PASO 1. Generamos los datos

t<-seq(from=20,to=70,by=5)
CM<-c(1.7,2,2.3,2.5,2.6,2.7,2.7,2.7,2.7,2.6,NA)
CC<-c(NA,3.6,3.7,3.6,3.4,3.1,2.9,2.7,2.4,2.2,NA)

#NA quiere decir que ese dato no está disponible

#PASO 2. Realizamos la gráfica

# Creamos un gráfico con dos curvas (CM y CC respecto al tiempo)

plot(t,CC,type="l",col="black",main="Turno de máxima renta en
especie"
      ,sub="Betula alba/Galicia",xlab="Edad(años)"
      ,ylab="Crec.(m^3/ha y años)")
lines(t,CM, type="l",col="red")
```

Caso Práctico 10: Gráfico de evolución de un rodal a partir de una tabla de producción

1. Objetivo del programa.

El objetivo de este programa es generar una tabla de producción y sus gráficos asociados.

2. Base de datos asociada.

No.

3. Instalación de Librerías.

No.

4. Pasos a seguir en el programa.

PASO 1. Generamos los datos

Edad

```
t<-
c(20,20,25,25,30,30,35,35,40,40,45,45,50,50,55,55,60,60,65,65,
,70)
```

Número de árboles

```
Y<-
c(3486,2482,2482,1931,1931,1593,1593,1368,1368,1210,1210,
1094,1094,1006, 1006,938,938,883,883,883,883)
```

Diámetro medio cuadrático

```
Z<-
c(6.4,7,8.2,8.7,9.8,10.3,11.4,11.8,12.6,13.1,13.8,14.2,14.9,15.2,
15.8,16.1,16.7,17,17.4,17.7,18.1)
```

Área basimétrica

```
K<-
c(11.1,9.5,13,11.6,14.6,13.4,16,14.9,17.2,16.2,18.2,17.4,19.1,1
8.3,19.8,19.2,20.5,19.9,21.1,20.6,21.6)
```

Volumen

```
S<-
c(33.1,28.8,46.9,42.3,60.6,56,73.8,69.3,86.2,82,97.7,93.7,108.3
,104.6,118,114.6,126.8,123.7,134.9,132,142.2)
```

PASO 2. Dibujamos los gráficos

```
plot(t,Y,type="lines",xlab="Edad(años)",ylab="arb/ha")
plot(t,Z,type="lines",xlab="Edad(años)",ylab="cm")
plot(t,K,type="lines",xlab="Edad(años)",ylab="m^2/ha")
plot(t,S,type="lines",xlab="Edad(años)",ylab="m^3/ha")
```

Dividimos el eje de ordenadas por 10 para que las marcas de división sean más pequeñas.

```
Y<-Y/10
plot(t,Y,type="lines",xlab="Edad(años)",ylab="arb/ha/10")
lines(t,K)
```

5. Programa de R (script)

```
#PASO 1. Generamos los datos

#Edad
t<-
c(20,20,25,25,30,30,35,35,40,40,45,45,50,50,55,55,60,60,65,65,70)

#Número de árboles
Y<-
c(3486,2482,2482,1931,1931,1593,1593,1368,1368,1210,1210,1094
,1094,1006, 1006,938,938,883,883,883,883)

#Diámetro medio cuadrático
Z<-
c(6.4,7,8.2,8.7,9.8,10.3,11.4,11.8,12.6,13.1,13.8,14.2,14.9,15.2,15.8,
16.1,16.7,17,17.4,17.7,18.1)

#Área basimétrica
K<-
c(11.1,9.5,13,11.6,14.6,13.4,16,14.9,17.2,16.2,18.2,17.4,19.1,18.3,1
9.8,19.2,20.5,19.9,21.1,20.6,21.6)

#Volumen
S<-
c(33.1,28.8,46.9,42.3,60.6,56,73.8,69.3,86.2,82,97.7,93.7,108.3,104.
6,118,114.6,126.8,123.7,134.9,132,142.2)

#PASO 2. Dibujamos los gráficos

plot(t,Y,type="lines",xlab="Edad(años)",ylab="arb/ha")
plot(t,Z,type="lines",xlab="Edad(años)",ylab="cm")
plot(t,K,type="lines",xlab="Edad(años)",ylab="m^2/ha")
plot(t,S,type="lines",xlab="Edad(años)",ylab="m^3/ha")

#Dividimos el eje de ordenadas por 10 para que las marcas de
#división sean más pequeñas.

Y<-Y/10
```

```
plot(t,Y,type="lines",xlab="Edad(años)",ylab="arb/ha/10")  
lines(t,K)
```

Caso práctico 11. Ajuste y dibujo de curvas de calidad (modelos de Hossfeld-I y de Bertalanffy-Richards)

Las propiedades básicas de las curvas de calidad de estación son:

1. tener una asíntota horizontal y un punto de inflexión
2. tener un comportamiento lógico
3. la familia de curvas debe ser, salvo contadas ocasiones, polimórfica, es decir, las curvas no deben ser proporcionales.

En este caso práctico en primer lugar elaboraremos unas curvas de calidad anamórficas mediante el método de la curva guía y posteriormente elaboraremos unas curvas de calidad polimórficas mediante regresión no lineal.

1. Objetivo del programa.

El objetivo de este programa es ajustar y dibujar las curvas de calidad con el modelo de HOSSFELD I y BERTALANFFY-RICHARDS.

El modelo de Hossfeld-I tiene la siguiente forma:

$$H_0 = \frac{t^2}{(a + b * t)^2} \text{ [Ec 1.1]}$$

El modelo de Bertalanffy-Richards tiene la siguiente estructura:

$$H_0 = a * (1 - e^{-b*t})^c \text{ [Ec 1.2]}$$

2. Base de datos asociada.

Sí. Se trata de un archivo de txt denominado “DATOS4-CurvasCalidad” en el que aparecen los campos siguientes:

Edad: Se refiere a la edad de la parcela o de árbol.

Altura es la altura dominante de la parcela.

Parcela es la parcela de medición.

Tipo se refiere al origen de los datos (parcela o tronco) Parcela, si el dato procede de inventario y Tronco, si el dato procede de análisis de tronco.

SI es el índice de sitio.

Random es un número aleatorio que se ha generado por si se quiere partir la base de datos y reservar parte de ella para la validación del modelo.

3. Instalación de Librerías.

Sí.

```
install.packages(lattice)
```

4. Pasos a seguir en el programa.

Modelo de Hossfeld-I

PASO 1. Instalamos la librería lattice

Este paso requiere correr la orden:

```
install.packages('lattice')
```

Hay que seleccionar en el CRAN MIRROR, Spain.

La llamamos

```
require('lattice')
```

PASO 2. Leemos los datos

```
setwd('C:/datosR')
datos<-read.delim('DATOS4-CurvasCalidad.csv',sep=";",
dec=".",header=TRUE)
```

Dejamos solo los datos correspondientes a análisis de tronco (eliminamos los de parcela)

```
datos1<-subset(datos,tipo !="parcela")
```

Visualizamos los datos.

```
head(datos1)
```

PASO 3. Dibujamos los datos

Dibujar la trayectoria Ho~edad para los datos de análisis de tronco. Si queremos abrir el gráfico en una pantalla nueva, ponemos `win.graph()`. En el caso de que se trabaje en RStudio no hace falta esta instrucción.

```
xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
col="grey",
main="Arboles tipo (Ps),Alto Valle del Ebro",
ylab="Altura (m)", xlab="Edad (años)")
```

la librería lattice ha permitido tener acceso al argumento groups

PASO 4. Ajuste de las curvas de calidad (anamórficas)

Ajuste del modelo de Hossfeld I (curvas anamórficas) mediante el **método de la curva guía**

a. Preparación de los datos

Borramos los datos con Edad igual a 0 años

```
datos1 <- subset(datos1, Edad !=0)
```

Generamos una variable dependiente y otra independiente a partir de la ecuación 1.1

```
datos1$vardep <- datos1$Edad/datos1$altura^0.5
```

b. Ajuste de un modelo lineal

Se ajusta un modelo lineal, para obtener los parámetros semilla necesarios para el ajuste no lineal

```
Hossfeld.lineal <- lm(datos1$vardep ~ (datos1$Edad))  
summary(Hossfeld.lineal)
```

muestra el resultado del ajuste lineal, a y b significativas y con valores $a = 5.731933$ y $b = 0.176904$. R^2 igual 0.7666

c. Gráfico de la curva guía

```
X<-c(1:140)  
Y<-c(X^2/(5.731933+0.176904*X)^2)  
plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,  
main="Curvas Guía, Hossfeld I", ylab="Altura (m)", xlab =  
"Edad (años)")
```

d. Gráfico de las curvas de calidad (anamórficas)

Para esta especie se han definido 5 índices de sitios diferentes (14, 17, 20, 23 y 26 m de Ho a la edad típica = 100 años) Se sustituyen la altura y la edad por los valores de índice de sitio y de edad típica para obtener los parámetros a ó b, manteniendo uno constante.

```
X<-c(1:140)
```

d.1. Gráfico del modelo de Hossfeld I-b.

Hossfeld I-b tiene el parámetro b común, es decir que se asume una única asíntota para todas las curvas. Por lo que b se mantiene constante e igual 0.176904 y así, despejamos a

SI= 14 a = 9.03572419

SI= 17 a = 6.5631625
 SI= 20 a = 4.67027977
 SI= 23 a = 3.16104141
 SI= 26 a = 1.92121351

Se va a dibujar un conjunto de curvas de calidad anamórficas o proporcionales

```
Y<-c(X^2/(9.03572419+0.176904*X)^2)
K<-c(X^2/(6.5631625+0.176904*X)^2)
Z<-c(X^2/(4.67027977+0.176904*X)^2)
G<-c(X^2/(3.16104141+0.176904*X)^2)
H<-c(X^2/(1.92121351+0.176904*X)^2)
```

```
plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,
      main="Curvas de Calidad, Hossfeld I, b común",
      ylab="Altura (m)", xlab = "Edad (años)")
lines(X,K,lwd=1, col = "blue")
lines(X,Z,lwd=1, col = "green")
lines(X,G,lwd=1, col = "black")
lines(X,H,lwd=1, col = "yellow")
```

en el gráfico, los siguientes parámetros definen características:

Lty, define el tipo de línea: (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) o uno de sus caracteres: "blank", "solid", "dashed", "dotted", "dotdash", "longdash", o "twodash". la opción "blank" utiliza líneas invisibles, no dibujadas.

ahora juntamos el gráfico de las curvas de calidad con las líneas de los análisis de troncos

```
xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
       col="grey",
       main="Arboles tipo (Ps),Alto Valle del Ebro/Hossfeld I b
común", ylab="Altura (m)", xlab="Edad
(años)",panel=function(x, y, ...){
  panel.xyplot(x, y, ...)
  panel.lines(X, Y, col=2, lwd=2)
  panel.lines(X, K, col=4, lwd=2)
  panel.lines(X, Z, col=3, lwd=2)
  panel.lines(X, G, col=1, lwd=2)
  panel.lines(X, H, col=7, lwd=2)})
```

d.2.Gráfico del modelo de Hossfeld I-a.

Hossfeld I-a tiene el parámetro a común, es decir que se asume no existe una única asíntota para todas las curvas

Análogamente, se deja a constante e igual 5.731933 y se despeja b

SI= 14 b = 0.20994191

SI= 17 b = 0.1852163

SI= 20 b = 0.16628747

SI= 23 b = 0.15119508

SI= 26 b = 0.13879680

```
Q<-c(X^2/(5.731933+0.20994191 *X)^2)
W<-c(X^2/(5.731933+0.1852163 *X)^2)
E<-c(X^2/(5.731933+0.16628747 *X)^2)
R<-c(X^2/(5.731933+0.15119508 *X)^2)
U<-c(X^2/(5.731933+0.13879680 *X)^2)
plot(X,Q,lwd=1, type="l", col = "red", lty=1, ljoin=10,
      main="Curvas de Calidad, Hossfeld I, a común",
      ylab="Altura (m)", xlab = "Edad (años)")
lines(X,W,lwd=1, col = "blue")
lines(X,E,lwd=1, col = "green")
lines(X,R,lwd=1, col = "black")
lines(X,U,lwd=1, col = "yellow")
```

Ahora con los datos de los análisis de troncos por detrás

```
xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
       col="grey",
       main="Arboles tipo (Ps),Alto Valle del Ebro/Hossfeld I a
       común", ylab="Altura (m)", xlab="Edad
       (años)",panel=function(x, y, ...){
         panel.xyplot(x, y, ...)
         panel.lines(X, Q, col=2, lwd=2)
         panel.lines(X, W, col=4, lwd=2)
         panel.lines(X, E, col=3, lwd=2)
         panel.lines(X, R, col=1, lwd=2)
         panel.lines(X, U, col=7, lwd=2)})
```

PASO 5. Ajuste de las curvas de calidad (anamórficas) mediante regresión no lineal

Ajuste del modelo de Hossfeld I (mediante regresión no lineal)

Los apartados a y b se realizan igual que en el PASO 4.

c. Ajuste no lineal

Definimos el modelo de Hossfeld I para su posterior ajuste no lineal

```
Hossfeld <-
datos1$altura~datos1$Edad^2/(a+b*datos1$Edad)^2
```

Crea en el objeto Hossfeld el texto del modelo que se quiere ajustar por regresión no lineal

```
nlmod <- nls(Hossfeld, data= datos1, start=list(a=5.7319,  
b=0.1769))
```

Ajusta un modelo de regresión no lineal utilizando como parámetros semillas de a y b los valores obtenidos en el ajuste lineal

```
summary(nlmod)  
datos1$predhnl <- predict(nlmod)
```

Calcula el predicho de H0 utilizando las estimaciones de los parámetros del modelo no lineal

```
head(datos1)
```

El apartado d (gráficos) se realiza de forma análoga a lo descrito en el PASO 4

Modelo de Bertalanffy-Richards

PASO 1. Instalamos la librería lattice

Este paso requiere correr la orden:

```
install.packages('lattice')
```

Hay que seleccionar en el CRAN MIRROR, Spain.

La llamamos

```
require('lattice')
```

PASO 2. Leemos los datos

```
setwd('C:/datosR')  
datos<-read.table('DATOS4-CurvasCalidad.txt',sep='\t',  
dec=',',header=TRUE)
```

Dejamos solo los datos correspondientes a análisis de tronco (eliminamos los de parcela)

```
datos1<-subset(datos,tipo !="parcela")
```

Visualizamos los datos.

```
head(datos1)
```

PASO 3. Dibujamos los datos

Dibujar la trayectoria Ho~edad para los datos de análisis de tronco si queremos abrir el gráfico en una pantalla nueva, ponemos `win.graph()`. si se corre en RStudio no hace falta.

```
xyplot(altura~Edad,data=datos1,groups=parcela, type="l",  
col="grey", main="Arboles tipo (Ps),Alto Valle del Ebro",  
ylab="Altura (m)", xlab="Edad (años)")
```

la librería `lattice` ha permitido tener acceso al argumento `groups`

PASO 4. Ajuste de las curvas de calidad

Ajuste del modelo de BERTALANFFY-RICHARDS I

Ajuste mediante regresión de diferentes modelos candidatos para la curva de Richards. Por ello, haremos diferentes iteraciones con diferentes valores para b en la ecuación 1.2. Tras diversas iteraciones, hemos llegado a la conclusión de que el modelo adecuado se obtiene cuando $b = 0.015$

a. Preparación de los datos

Borramos los datos con Edad igual a 0 años

```
datos1 <- subset(datos1, Edad !=0)
```

Generamos una variable dependiente y otra independiente a partir de la ecuación 1.2

```
datos1$vardep <- log (datos1$altura)  
datos1$varindep <- log (1-exp(-0.015*datos1$Edad))
```

b. Ajuste de un modelo lineal

Se ajusta un modelo lineal, para obtener los parámetros semilla necesarios para el ajuste no lineal

```
Richards.lineal <- lm(datos1$vardep ~ (datos1$varindep))  
summary(Richards.lineal)
```

muestra el resultado del ajuste línea, a y c significativas y con valores $a = 3.38019$ y $c = 1.39210$ (b ya lo habíamos fijado en 0.015). R^2 igual 0.8442

Gráfico de la curva guía y de las curvas anamórficas con el modelo de Richards

c. Gráfico de la curva guía

```
X<-c(1:140)
Y<-c(exp(3.38019)*(1-exp(-0.015*X))^1.39210)
plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,
main="Curvas Guía, Richards", ylab="Altura (m)", xlab =
"Edad (años)")
```

d. Gráfico de las curvas de calidad (anamórficas)

Para esta especie se han definido 4 índice de sitios diferentes (14, 17, 20 y 23 m de Ho a la edad típica = 100 años)

Se sustituyen la altura y la edad por los valores de índice de sitio y de edad típica para obtener los parámetros a ó c, manteniendo uno constante.

```
X<-c(1:140)
```

d.1. Gráfico del modelo de Richards-c.

Richards I-c tiene el parámetro c común, es decir que se asume una única asíntota para todas las curvas. Por eso c se mantiene constante e igual 1.39210 y b ya lo habíamos fijado en 0.015 y así, despejamos a

SI= 14 a = 2.88572706

SI= 17 a = 3.11972512

SI= 20 a = 3.33011979

SI= 23 a = 3.52236642

SI= 26 a = 3.70012628

Se va a dibujar un conjunto de curvas de calidad anamórficas o proporcionales

```
Y<-c(exp(2.88572706)*(1-exp(-0.015*X))^1.39210)
K<-c(exp(3.11972512)*(1-exp(-0.015*X))^1.39210)
Z<-c(exp(3.33011979)*(1-exp(-0.015*X))^1.39210)
G<-c(exp(3.52236642)*(1-exp(-0.015*X))^1.39210)
H<-c(exp(3.70012628)*(1-exp(-0.015*X))^1.39210)
```

```
plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,
main="Curvas de Calidad, Richards, b y c común",
ylab="Altura (m)", xlab = "Edad (años)")
lines(X,K,lwd=1, col = "blue")
lines(X,Z,lwd=1, col = "green")
lines(X,G,lwd=1, col = "black")
lines(X,H,lwd=1, col = "yellow")
```

ahora juntamos el gráfico de las curvas de calidad con las líneas de los análisis de troncos

```
xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
col="grey",
main="Arboles tipo (Ps),Alto Valle del Ebro/Richards b y
c común", ylab="Altura (m)", xlab="Edad
(años)",panel=function(x, y, ...){
panel.xyplot(x, y, ...)
panel.lines(X, Y, col=2, lwd=2)
panel.lines(X, K, col=4, lwd=2)
panel.lines(X, Z, col=3, lwd=2)
panel.lines(X, G, col=1, lwd=2)
panel.lines(X, H, col=7, lwd=2)}})
```

PASO 5. Ajuste de las curvas de calidad (anamórficas) mediante regresión no lineal

Se deben repetir los apartados a y b descritos en el PASO 4 anterior

a. Preparación de los datos

```
datos1 <- subset(datos1, Edad !=0)
datos1$Y<-log(datos1$altura)
b <- 0.009
```

Asignamos diferentes valores a "b" y repetimos la ejecución del siguiente código de R (hasta el gráfico); finalizar cuando los perfiles transformados tengan apariencia de rectas

```
datos1$X<-log(1-exp(-b*datos1$Edad))
```

b. Visualizamos los datos

```
xyplot(Y~X, data=datos1, groups=parcela, type="l",
col="grey", main="Gráfico de y frente a
x", ylab="y", xlab="x")
```

c. Ajuste de un modelo no lineal

Se ajusta un modelo lineal, para obtener los parámetros semilla necesarios para el ajuste no lineal

```
Bertalanffy.lineal<-lm(Y~X,data=datos1)
summary(Bertalanffy.lineal)
```

Creamos el objeto richards el texto del modelo que se quiere ajustar por regresión no lineal

```
Richards<-datos1$altura~a*(1-exp(-b*datos1$Edad))^c
```

ajuste de un modelo no lineal

```
nlmod<-
nls(datos1,Richards,start=list(a=3.81,b=0.009,c=1.3))
summary(nlmod)
```

4. Programa de R (script)

Solo se presenta el ajuste para el modelo Hossfeld-I

```
# PASO 1. Instalar la librería lattice

install.packages('lattice')
require('lattice')

#PASO 2. Leer los datos
setwd('C:/datosR')
datos<-read.delim('DATOS4-CurvasCalidad.csv',sep=";", dec="."
,header=TRUE)

#Eliminar los datos que son de parcela y no de análisis de tronco

datos1<-subset(datos,tipo !="parcela")
head(datos1)

#PASO 3. Dibujar los datos

xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
col="grey",
      main="Arboles tipo (Ps),Alto Valle del Ebro", ylab="Altura
(m)", xlab="Edad (años)")

#PASO 4. Ajuste de las curvas de calidad (anamórficas)

#a. Preparación de los datos

#Borramos los datos con Edad igual a 0 años
datos1 <- subset(datos1, Edad !=0)

#Generamos una variable dependiente y otra independiente a
#partir de la ecuación 1.1

datos1$vardep <- datos1$Edad/datos1$altura^0.5

#b. Ajuste de un modelo lineal

Hossfeld.lineal <- lm(datos1$vardep ~ (datos1$Edad))
```

```

summary(Hossfeld.lineal)

#c. Gráfico de la curva guía

X<-c(1:140)
Y<-c(X^2/(5.731933+0.176904*X)^2)
plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,
main="Curvas Guía, Hossfeld I", ylab="Altura (m)", xlab = "Edad
(años)")

#d. Gráfico de las curvas de calidad (anamórficas)

X<-c(1:140)

#d.1. Gráfico del modelo de Hossfeld I-b.

Y<-c(X^2/(9.03572419+0.176904*X)^2)
K<-c(X^2/(6.5631625+0.176904*X)^2)
Z<-c(X^2/(4.67027977+0.176904*X)^2)
G<-c(X^2/(3.16104141+0.176904*X)^2)
H<-c(X^2/(1.92121351+0.176904*X)^2)

plot(X,Y,lwd=1, type="l", col = "red", lty=1, ljoin=10,
main="Curvas de Calidad, Hossfeld I, b común",
      ylab="Altura (m)", xlab = "Edad (años)")
lines(X,K,lwd=1, col = "blue")
lines(X,Z,lwd=1, col = "green")
lines(X,G,lwd=1, col = "black")
lines(X,H,lwd=1, col = "yellow")

xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
col="grey",
      main="Arboles tipo (Ps),Alto Valle del Ebro/Hossfeld I b
común", ylab="Altura (m)", xlab="Edad (años)",panel=function(x,
y, ...){
  panel.xyplot(x, y, ...)
  panel.lines(X, Y, col=2, lwd=2)
  panel.lines(X, K, col=4, lwd=2)
  panel.lines(X, Z, col=3, lwd=2)
  panel.lines(X, G, col=1, lwd=2)
  panel.lines(X, H, col=7, lwd=2);})

#d.2.Gráfico del modelo de Hossfeld I-a.

Q<-c(X^2/(5.731933+0.20994191 *X)^2)
W<-c(X^2/(5.731933+0.1852163 *X)^2)
E<-c(X^2/(5.731933+0.16628747 *X)^2)
R<-c(X^2/(5.731933+0.15119508 *X)^2)

```

```

U<-c(X^2/(5.731933+0.13879680 *X)^2)
plot (X,Q,lwd=1, type="l", col = "red", lty=1, ljoin=10,
      main="Curvas de Calidad, Hossfeld I, a común",
      ylab="Altura (m)", xlab = "Edad (años)")
lines(X,W,lwd=1, col = "blue")
lines(X,E,lwd=1, col = "green")
lines(X,R,lwd=1, col = "black")
lines(X,U,lwd=1, col = "yellow")

xyplot(altura~Edad,data=datos1,groups=parcela,type="l",
       col="grey",
       main="Arboles tipo (Ps),Alto Valle del Ebro/Hossfeld I a
común", ylab="Altura (m)", xlab="Edad (años)",panel=function(x,
y, ...){
  panel.xyplot(x, y, ...)
  panel.lines(X, Q, col=2, lwd=2)
  panel.lines(X, W, col=4, lwd=2)
  panel.lines(X, E, col=3, lwd=2)
  panel.lines(X, R, col=1, lwd=2)
  panel.lines(X, U, col=7, lwd=2)})

#PASO 5. Ajuste de las curvas de calidad (anamórficas):
#Regresión no lineal

#a. Preparación de los datos

datos1 <- subset(datos1, Edad !=0)

datos1$vardep <- datos1$Edad/datos1$altura^0.5

#b. Ajuste de un modelo lineal

Hossfeld.lineal <- lm(datos1$vardep ~ (datos1$Edad))
summary(Hossfeld.lineal)

#c. Ajuste no lineal

Hossfeld <- datos1$altura~datos1$Edad^2/(a+b*datos1$Edad)^2
nlmod <- nls(Hossfeld, data= datos1, start=list(a=5.7319,
b=0.1769))
summary(nlmod)
datos1$predhnl <- predict(nlmod)

head(datos1)

```


Caso Práctico 12: Síntesis de información edáfica mediante análisis de componentes principales (PCA)

1. Objetivo del programa.

Mediante el análisis de componentes principales (PCA) lo que pretendemos es sintetizar la información mediante la generación de un número reducido de nuevas variables que son combinación lineal de las variables originales de la base de datos y que son independientes entre sí. Al interpretar los componentes principales se debe considerar tanto el signo como la magnitud de los parámetros asociados a las variables originales. Esto no es fácil y, por tanto, el conocimiento que se tenga del objeto de estudio es clave en el adecuado uso de este método. En este ejemplo vamos a reducir un conjunto de variables edáficas de diversas parcelas a una serie de componentes principales.

2. Base de datos asociada.

Sí.

Se trata de un archivo de Excel denominado “DATOS2.txt” en el que aparecen los campos siguientes:

SI4: Índice de sitio de la parcela (14, 17, 20 o 23)

ARENA: % de arena

LIMO: % de limo

ARCILLA: % de arcilla

P: Fosforo

K: Potasio

Ca: Calcio

Mg: Magnesio

Na: Sodio

CARBONAT: Carbonatos

CALIZA: Caliza

MO: Materia orgánica

CCC: Capacidad de intercambio catiónico

pH: pH

CONDUCT: Conductividad

[Nota: no dejar espacios en la variable, que todas sus letras vayan juntas]

3. Instalación de Librerías.

No

4. Pasos a seguir en el programa.

PASO 1. Creamos el espacio de trabajo

Se pueden visualizar los archivos de dicho fichero con la función `dir()`

```
setwd('C:/datosR')
```

```
dir()
```

PASO 2. Se selecciona el archivo en dicho fichero y mandamos a R leerlo

Leer y visualizar los datos

```
suelo <- read.table("E:/Practicas_MFA/DATOS2.txt",  
header=TRUE, sep="\t", dec=".")
```

PASO 3. Visualizar datos.

```
head(suelo)
```

visualiza el encabezamiento y los primeros 6 registros de los datos

```
suelo
```

muestra toda la base de datos

PASO 4. Realiza el análisis de componentes principales

```
pc <- princomp(suelo[, -1], cor=TRUE)
```

realiza el análisis de componente principales

```
plot (pc, type="l")
```

Dibuja el gráfico con la varianza que recoge cada componente

```
pc$loadings  
pc$scores
```

Muestra los pesos de los componentes principales

```
biplot(pc)
```

Muestra el gráfico de componentes principales con los pesos de cada variable

```
plot(pc$scores[,1], pc$scores[,2])  
text(pc$scores[,1], pc$scores[,2], suelo[,1], pos=4)
```

Dibuja las observaciones sobre los dos primeros ejes mostrando el Índice de sitio

5. Programa de R (script)

```
#PASO 1. Creamos el espacio de trabajo  
setwd('C:/datosR')  
dir()  
  
#PASO 2. Se selecciona el archivo en dicho fichero  
suelo <- read.table("E:/Practicas_MFA/DATOS2.txt",  
header=TRUE, sep="\t", dec=".")  
  
#PASO 3. Visualizar datos.
```

```
head(suelo)
suelo

#PASO 4. Realiza el análisis de componentes principales
pc <- princomp(suelo[,-1],cor=TRUE)
plot (pc, type="l")
pc$loadings
pc$scores
biplot(pc)
plot(pc$scores[,1],pc$scores[,2])
text(pc$scores[,1],pc$scores[,2],suelo[,1],pos=4)
```

Caso Práctico 13: Clasificación de parcelas por índice de sitio a partir de datos edáficos (*soil-site method*) mediante análisis discriminante

1. Objetivo del programa.

El análisis discriminante es un método de clasificación que se puede utilizar cuando los datos se pueden agrupar en dos o más clases conocidas a priori. El método se basa en la idea de que los datos se agrupan en el espacio de forma que es posible construir un conjunto de funciones lineales que sirvan para discriminar los grupos y clasificar nuevas observaciones. En este ejemplo vamos a utilizar los mismos datos que en el ejemplo anterior con el objeto de generar un método que permita clasificar la calidad de estación en función de variables edáficas.

2. Base de datos asociada.

Sí. Se trata de un archivo de Excel denominado "DATOS2.txt" en el que aparecen los campos siguientes:

SI4: Índice de sitio de la parcela (14, 17, 20 o 23)

ARENA: % de arena

LIMO: % de limo

ARCILLA: % de arcilla

P: Fosforo

K: Potasio

Ca: Calcio

Mg: Magnesio

Na: Sodio

CARBONAT: Carbonatos

CALIZA: Caliza

MO: Materia orgánica

CCC: Capacidad de intercambio catiónico

pH: pH

CONDUCT: Conductividad

[Nota: no dejar espacios en la variable, que todas sus letras vayan juntas]

3. Instalación de Librerías.

Sí. Hace falta la librería MASS.

4. Pasos a seguir en el programa.

Los pasos 1, 2 y 3 son idénticos a los realizados en el ejemplo anterior

PASO 1. Creamos el espacio de trabajo

Se pueden visualizar los archivos de dicho fichero con la función `dir()`

```
setwd('C:/datosR')
```

```
dir()
```

PASO 2. Se selecciona el archivo en dicho fichero y mandamos a R leerlo

Leer y visualizar los datos

```
suelo <- read.table("E:/Practicas_MFA/DATOS2.txt",  
header=TRUE, sep="\t", dec=".")
```

PASO 3. Visualizar datos.

```
head(suelo)
```

visualiza el encabezamiento y los primeros 6 registros de los datos

```
suelo
```

muestra toda la base de datos

PASO 4. Realizar el análisis discriminante

```
library(MASS)
```

```
disc <- lda(SI4 ~ ., data=suelo)
```

Análisis discriminante lineal

```
Disc
```

muestra el resultado

```
plot(disc)
```

dibuja la dispersión de datos frente a las funciones lineales definidas (en este caso 3)

```
clas <- predict(disc,suelo[,-1])  
plot(clas$x,pch=as.character(suelo[,1]),col=as.numeric(clas$class)+1)
```

```
table(suelo[,1],clas$class)
```

muestra una tabla de filas (aquí es suelo[,1]) frente a columnas (aquí clas\$class)

5. Programa de R (script)

```
#PASO 1. Creamos el espacio de trabajo  
setwd('C:/datosR')  
dir()  
  
#PASO 2. Se selecciona el archivo en dicho fichero  
suelo <- read.table("E:/Practicas_MFA/DATOS2.txt",  
header=TRUE, sep="\t", dec=".")
```

```
#PASO 3. Visualizar datos.
head(suelo)
suelo

#PASO 4. Realizar el análisis discriminante

library(MASS)

disc <- lda(SI4 ~ ., data=suelo)
# Análisis discriminante lineal

Disc
# muestra el resultado

plot(disc)
# dibuja la dispersión de datos frente a las funciones lineales
#definidas (en este caso 3)

clas <- predict(disc,suelo[,-1])
plot(clas$x,pch=as.character(suelo[,1]),col=as.numeric(clas$class)
+1)

table(suelo[,1],clas$class)
```

Caso Práctico 14: Ajuste de ecuaciones de biomasa

1. Objetivo del programa

La cuantificación de carbono en un sistema forestal o para una especie concreta se realiza a través de la estimación de la biomasa. Para conocer la biomasa se pueden ajustar unas ecuaciones para cada componente. En este caso se ajustan ecuaciones para estimar la biomasa para cada parte considerada (fuste, hojas, ramas gruesas y ramas delgadas) pero se usa la metodología SUR (Seemingly Unrelated Regressions) o de mínimos cuadrados conjuntos para conferir la propiedad aditiva a las ecuaciones generadas. Hay que tener en cuenta que con este programa no se corrige la heterocedasticidad.

2. Base de datos asociada

Si. En este caso es un archivo excel denominado “biomasa.csv” en el que aparecen los siguientes campos:

Arbol: número identificativo del árbol

ht_m: altura total en metros

dbh: diámetro normal en cm

r_gruesas: peso seco de la parte considerada ramas gruesas (diámetro de 2 – 7 cm)

r_delgadas: peso seco de la parte considerada ramas delgadas (diámetro < 2 cm)

hojas: peso seco de la parte considerada hojas

fuste: peso seco de la parte considerada fuste (diámetro > 7 cm)

biomasatotal: peso seco de todas las partes

3. Instalación de librerías

Si. Debemos instalar las librerías “plyr”, “lattice” y “systemfit”.

4. Pasos a seguir en el programa

PASO 1. Instalar y cargar las librerías requeridas

```
# Instalar librerías necesarias
install.packages('plyr')
install.packages('systemfit')
```

```
# las llamamos para que se puedan ejecutar
require('plyr')
require('lattice')
require('systemfit')
```

PASO 2. Leemos los datos

```
datos<-read.csv2(C/'biomasa.csv',
sep='; ',dec='.',header=T,na.string='NA')
```

PASO 3. Ajuste de los modelos para cada parte de biomasa considerada en el trabajo de campo. En este apartado solo se presenta el ajuste para el fuste a modo de ejemplo pero habría que repetirlo para cada parte (ramas, hojas...)

Se definen los modelos generales

```

modelo1<-fuste~ b1*dbh*ht_m
modelo2<-fuste~ b1*dbh^2*ht_m
modelo3<-fuste~ (b1*dbh)+(b2*dbh^2)
modelo4<-fuste~ (b1*dbh)+(b2*dbh^2)+(b3*dbh^2*ht_m)
modelo5<-fuste~ (b1*dbh)+(b2*ht_m)
modelo6<-fuste~ (b1*dbh^2)+(b2*dbh^2*ht_m)
modelo7<-fuste~ (b1*dbh^2)+(b2*ht_m)
modelo8<-fuste~ (b1*dbh^2)+(b2*ht_m)+(b3*dbh^2*ht_m)
modelo9<-fuste~ (b1*dbh^2)+(b2*dbh*ht_m)
modelo10<-fuste~(b1*dbh^2*ht_m)+(b2*dbh*ht_m)
modelo11<-fuste~(b1*dbh^b2*ht_m^b3)
modelo12<-fuste~(b1*dbh^b2)
modelo13<-fuste~(b1*(dbh*ht_m)^b2)

```

Se ajustan los modelos con la función nls (nonlinear regresion)
start= son los valores de arranque de la interacción del modelo
msMaxIter= el número máximo de interacciones permitidas

```

MF.1<- nls (modelo1, datos, start=list(b1=1))
MF.2<- nls (modelo2, datos, start=list(b1=1))
MF.3<- nls (modelo3, datos, start=list(b1=1,b2=1))
MF.4<- nls (modelo4, datos, start=list(b1=1,b2=1,b3=1))
MF.5<- nls (modelo5, datos, start=list(b1=1,b2=1))
MF.6<- nls (modelo6, datos, start=list(b1=1,b2=1))
MF.7<- nls (modelo7, datos, start=list(b1=1,b2=1))
MF.8<- nls (modelo8, datos, start=list(b1=1,b2=1,b3=1))
MF.9<- nls (modelo9, datos, start=list(b1=1,b2=1))
MF.10<- nls(modelo10,datos, start=list(b1=0.06,b2=0.3),
            control=list(msMaxIter = 200))
MF.11<- nls (modelo11, datos, start=list(b1=0.07,b2=2.8,b3=-
0.84),
            control=list(msMaxIter = 200))
MF.12<- nls (modelo12, datos, start=list(b1=0.07,b2=2.45),
            control=list(msMaxIter = 200))
MF.13<- nls (modelo13, datos, start=list(b1=0.007 ,b2=1.88 ),
            control=list(msMaxIter = 200))

```

Selección de los modelos a partir del resumen del modelo. Se seleccionan a partir de los parámetros significativos.

```
summary(MF.1)
```



```
summary(MF.2)
summary(MF.3)
summary(MF.4)
summary(MF.5)
summary(MF.6)
summary(MF.7)
summary(MF.8)
summary(MF.9)
summary(MF.10)
summary(MF.11)
summary(MF.12)
summary(MF.13)
```

Para elegir los modelos hay que ver la significatividad de bi (resaltado en negrita en el siguiente ejemplo). Todos los parámetros deben ser significativos para ser considerado como posible modelo. Ejemplo:

```
Formula: fuste ~ b1 * dbh * ht_m
```

```
Parameters:
```

	Estimate	Std. Error	t value	Pr(> t)
b1	0.26771	0.01041	25.7	<2e-16 ***

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 8.407 on 23 degrees of freedom
```

```
Number of iterations to convergence: 1
```

```
Achieved convergence tolerance: 3.897e-08
```

Una vez elegidos los modelos significativos hay que seleccionar el mejor modelo, evaluando sólo los modelos que hemos seleccionado en el paso anterior.

Hacemos una función anónima para determinar r^2 , sse, res, AICs Y BICs

```
stats<-function(model){
  var.<-all.vars(as.formula(model))[1]
  sse<-sum(residuals(model)^2)
  set<-eval(model$call$data)
  fr2<-function(model){#funcion de r^2
    datos<-eval(model$call$data)
    datos$MF<-datos[,var.]-mean(datos[,var.])
    datos$MF2<-datos$MF^2
    SST<-sum(datos$MF2)
    1-(sse/SST)}
  rse<-summary(model)$sigma
  aic<-AIC(model)
  bic<-BIC(model)
  c(r2=fr2(model), sse=sse,rse=rse,AIC=aic,BIC=bic)
```

}

Calculamos los parámetros r^2 , sse, res, AICs Y BICs de los modelos seleccionados

```
parameters<-rbind(
  stats(MF.1),stats(MF.2),stats(MF.5), stats(MF.12))
parameters
```

Se seleccionan los modelos en sentido jerárquico: menor valor de criterio de Akaike (AIC), menor valor de raíz media del cuadrado del error (rse) y el mayor coeficiente de determinación (r^2).

	r^2	sse	rse	AIC	BIC
[1,]	0.8239879	1625.4499	8.406648	173.2807	175.6368
[2,]	0.9285911	659.4523	5.354609	151.6296	153.9857
[3,]	0.8221455	1642.4639	8.640454	175.5306	179.0648
[4,]	0.9428526	527.7495	4.897817	148.2827	151.8168

Una vez seleccionado el mejor modelo para cada parte es aconsejable anotar los valores de los coeficientes ajustados para usarlos en el paso 4

PASO 4. Ajuste simultáneo de ecuaciones – SUR

Se definen los modelos seleccionados para cada fracción

```
MF<-fuste~ (b1*dbh^b2)
MRG<-r_gruesas~ (b3*dbh^2*ht_m)
MRD<-r_delgadas~ (b4*dbh^2*ht_m)
MH<-hojas~ (b5*dbh)+(b6*dbh^2)
```

Se hace el modelo aditivo de la biomasa total como la suma de las fracciones de los modelos seleccionados

```
MBT<-
biomasatotal~((b1*dbh^b2)+(b3*dbh^2*ht_m)+(b4*dbh^2*ht_m)
+(b5*dbh)+(b6*dbh^2))
```

Se hace el ajuste simultáneo introduciendo como start.values los coeficientes de los modelos seleccionados en el paso 3.

```
labels<- list("Fuste","R.gruesas","R.delgadas","Hojas","BiomT")
#etiquetas de cada modelo

start.values <-
c(b1=0.11908,b2=2.16974,b3=0.0045503,b4=0.0054363,
  b5=-0.178823,b6=0.034375)
```

```

system.biomasa<-list(MF,MRG, MRD, MH,MBT)
#sistema de ecuaciones- modelos

modelBIOM<-nlssystemfit("SUR", system.biomasa, start.values
,data=datos,
                        eqnlabels=labels, maxiter=1000)
#nlssystemfit= sistema de ecuaciones no lineales
#SUR=metodo, system.biomasa=sistema de ecuaciones, valores
#de coeficientes,

```

Obtenemos la información del modelo ajustado

```

# Resumendel modelo completo
print(modelBIOM)
# Ajuste de la ecuación [[x]], en este caso 2
modelBIOM$eq[[2]]

# Mostrar los coeficientes
modelBIOM$b

# Mostrar los p-valor de coeff
modelBIOM$p

#R2 de los modelos ajustados
sapply(modelBIOM$eq,function(x){(x)$adjr2})#adjr2
## [1] 0.9447131 0.7999883 0.8464466 0.8861826 0.9338434

#extraer fitted y residuals del SUR model
fit_SUR<-
data.frame(sapply(modelBIOM$eq,function(x){(x)$predicted}))
names(fit_SUR)<-c('fit_FF','fit_RG','fit_RD','fit_H','fit_BT')
res_SUR<-
data.frame(sapply(modelBIOM$eq,function(x){(x)$res}))
names(res_SUR)<-c('res_FF','res_RG','res_RD','res_H','res_BT')
SUR_R<-cbind(fit_SUR,res_SUR

```

5. Programa de R (script)

```

# PASO 1. Instalar y cargar las librerías requeridas
# Instalar librerías necesarias
install.packages('plyr')
install.packages('systemfit')
# las llamamos para que se puedan ejecutar
require('plyr')
require('lattice')
require('systemfit')

# PASO 2. Leemos los datos

```

```

datos<-read.csv2(C/'biomasa.csv',
sep='; ',dec='.',header=T,na.string='NA')

# PASO 3 Ajuste del modelo para el fuste
#Se definen los modelos generales
modelo1<-fuste~ b1*dbh*ht_m
modelo2<-fuste~ b1*dbh^2*ht_m
modelo3<-fuste~ (b1*dbh)+ (b2*dbh^2)
modelo4<-fuste~ (b1*dbh)+ (b2*dbh^2)+( b3*dbh^2*ht_m)
modelo5<-fuste~ (b1*dbh)+ (b2*ht_m)
modelo6<-fuste~ (b1*dbh^2)+(b2*dbh^2*ht_m)
modelo7<-fuste~ (b1*dbh^2)+(b2*ht_m)
modelo8<-fuste~ (b1*dbh^2)+(b2*ht_m)+(b3*dbh^2*ht_m)
modelo9<-fuste~ (b1*dbh^2)+(b2*dbh*ht_m)
modelo10<-fuste~(b1*dbh^2*ht_m)+(b2*dbh*ht_m)
modelo11<-fuste~(b1*dbh^b2*ht_m^b3)
modelo12<-fuste~(b1*dbh^b2)
modelo13<-fuste~(b1*(dbh*ht_m)^b2)

#Se ajustan los modelos con la funcion nls (nonlinear regresion)
MF.1<- nls (modelo1, datos, start=list(b1=1))
MF.2<- nls (modelo2, datos, start=list(b1=1))
MF.3<- nls (modelo3, datos, start=list(b1=1,b2=1))
MF.4<- nls (modelo4, datos, start=list(b1=1,b2=1,b3=1))
MF.5<- nls (modelo5, datos, start=list(b1=1,b2=1))
MF.6<- nls (modelo6, datos, start=list(b1=1,b2=1))
MF.7<- nls (modelo7, datos, start=list(b1=1,b2=1))
MF.8<- nls (modelo8, datos, start=list(b1=1,b2=1,b3=1))
MF.9<- nls (modelo9, datos, start=list(b1=1,b2=1))
MF.10<- nls(modelo10,datos, start=list(b1=0.06,b2=0.3),
            control=list(msMaxIter = 200))
MF.11<- nls (modelo11, datos, start=list(b1=0.07,b2=2.8,b3=-
0.84),
            control=list(msMaxIter = 200))
MF.12<- nls (modelo12, datos, start=list(b1=0.07,b2=2.45),
            control=list(msMaxIter = 200))
MF.13<- nls (modelo13, datos, start=list(b1=0.007 ,b2=1.88 ),
            control=list(msMaxIter = 200))

# Selección de los modelos a partir de los parámetros
#significativos
summary(MF.1)
summary(MF.2)
summary(MF.3)
summary(MF.4)
summary(MF.5)
summary(MF.6)
summary(MF.7)

```

```

summary(MF.8)
summary(MF.9)
summary(MF.10)
summary(MF.11)
summary(MF.12)
summary(MF.13)

#Selección del mejor modelo de los modelos significativos
# función anónima para determinar r^2, sse, res, AICs Y BICs
stats<-function(model){
  var.<-all.vars(as.formula(model))[1]
  sse<-sum(residuals(model)^2)
  set<-eval(model$call$data)
  fr2<-function(model){#funcion de r^2
    datos<-eval(model$call$data)
    datos$MF<-datos[,var.]-mean(datos[,var.])
    datos$MF2<-datos$MF^2
    SST<-sum(datos$MF2)
    1-(sse/SST)}
  rse<-summary(model)$sigma
  aic<-AIC(model)
  bic<-BIC(model)
  c(r2=fr2(model), sse=sse,rse=rse,AIC=aic,BIC=bic)
}

# Cálculo de los parámetros
parameters<-rbind(
  stats(MF.1),stats(MF.2),stats(MF.5), stats(MF.12))
parameters

# El paso 3 hay que repetirlo para cada componente de biomasa
#que se quiera analizar.

# PASO 4. Ajuste simultáneo de ecuaciones – SUR

# Definición de los modelos seleccionados para cada fracción
MF<-fuste~ (b1*dbh^b2)
MRG<-r_gruesas~ (b3*dbh^2*ht_m)
MRD<-r_delgadas~ (b4*dbh^2*ht_m)
MH<-hojas~ (b5*dbh)+(b6*dbh^2)
# modelo aditivo de la biomasa total
MBT<-
biomasatotal~((b1*dbh^b2)+(b3*dbh^2*ht_m)+(b4*dbh^2*ht_m)
  +(b5*dbh)+(b6*dbh^2))

# ajuste simultáneo

```

```
labels<-
list("Fuste","R.gruesas","R.delgadas","Hojas","BiomT")#etiquetas
de cada modelo
start.values <-
c(b1=0.11908,b2=2.16974,b3=0.0045503,b4=0.0054363,
  b5=-0.178823,b6=0.034375)
system.biomasa<-list(MF,MRG, MRD, MH,MBT)#sistema de
ecuaciones- modelos
modelBIOM<-nlsystemfit("SUR", system.biomasa, start.values
,data=datos,
  eqnlabels=labels, maxiter=1000)
#nlsystemfit= sistema de ecuaciones no lineales
#SUR=metodo, system.biomasa=sistema de ecuaciones, valores de
coeficientes,

# Obtenemos la información del modelo ajustado
# Resumen
print(modelBIOM)
# Ajuste de la ecuación [[x]], en este caso 2
modelBIOM$eq[[2]]

# Mostrar los coeficientes
modelBIOM$b
# Mostrar los p-valor de coeff
modelBIOM$p
#R2 de los modelos ajustados
sapply(modelBIOM$eq,function(x){(x)$adjr2})#adjr2
## [1] 0.9447131 0.7999883 0.8464466 0.8861826 0.9338434
#extraer fitted y residuals del SUR model
fit_SUR<-
data.frame(sapply(modelBIOM$eq,function(x){(x)$predicted}))
names(fit_SUR)<-c('fit_FF','fit_RG','fit_RD','fit_H','fit_BT')
res_SUR<-
data.frame(sapply(modelBIOM$eq,function(x){(x)$res}))
names(res_SUR)<-c('res_FF','res_RG','res_RD','res_H','res_BT')
SUR_R<-cbind(fit_SUR,res_SUR
```

Para saber más

Referencias selvícolas

- Bravo, F., Álvarez, J.G., Río, M. del, et al 2012. *Growth and Yield Models in Spain: Historical Overview, contemporary Examples and Perspectives*. Instituto Universitario de Investigación en Gestión Forestal Sostenible. Universidad de Valladolid-INIA
- Burkhardt, H., & Tomé, M. 2012. *Modeling Forest Trees and Stands*. Springer.
- Diéguez-Aranda, U., Arias-Rodil, M., Álvarez González, J.G. 2012 *Prácticas de ampliación de Dasometría con R*. Departamento de Ingeniería Agroforestal, Universidad de Santiago de Compostela, Escuela Politécnica Superior de Lugo 42 páginas
- Pretzsch H. 2010 *Forest dynamics, growth and yield. From measurement to model*. Springer
- Weiskittel, A.R., Hann, D.W., Kershaw, J.A., Vanclay, J.K. 2011. *Forest Growth and Yield Modeling* Wiley

Análisis de datos con R

- Adler, J. 2009. *R in a nutshell*. O'Reilly
- Robinson, A.P., Hamann, J.D. 2011. *Forest Analytics with R*. Springer

Enlaces a páginas web

- <http://www.r-project.org/>
- <http://cran.r-project.org/manuals.html>
- <http://www.statmethods.net/>
- <http://manuals.bioinformatics.ucr.edu/home/programming-in-r>
- <http://www.rstudio.com/>
- <http://www.cookbook-r.com/>
- <http://rgraphgallery.blogspot.com.es>
- <http://stats.stackexchange.com/questions/138/resources-for-learning-r>
- <http://gastonsanchez.com/teaching/>