USO DE HERRAMIENTAS INFORMÁTICAS PARA LA RECOPILACIÓN, ANÁLISIS E INTERPRETACIÓN DE DATOS DE INTERÉS EN LAS CIENCIAS BIOMÉDICAS

Módulo 2
Gráficas con R

Alfredo Rigalli
Maela Lupo
María Eugenia Chulibert
Mercedes Lombarte
Patricia Lupión

Centro Universitario de Estudios Medioambientales
Facultad de Ciencias Médicas
Universidad Nacional de Rosario

ISBN 978-987-86-0199-1

USO DE HERRAMIENTAS INFORMÁTICAS PARA LA RECOPILACIÓN, ANÁLISIS E INTERPRETACIÓN DE DATOS DE INTERÉS EN LAS CIENCIAS BIOMÉDICAS

Gráficas con R

Alfredo Rigalli
Maela Lupo
María Eugenia Chulibert
Mercedes Lombarte
Patricia Lupión

Centro Universitario de Estudios Medioambientales
Facultad de Ciencias Médicas
Universidad Nacional de Rosario

Uso de herramientas informáticas para la
recopilación, análisis e interpretación de
datos de interés en las ciencias biomédicas :
gráficas con R / Alfredo Rigalli ... [et al.]. - 1a
edición para el alumno - Rosario : Alfredo
Rigalli, 2019.
Libro digital, PDF

Archivo Digital: descarga
ISBN 978-987-86-0199-1

1. Software Didáctico. I. Rigalli, Alfredo.
CDD 610.28

AUTORES

Chulibert, María Eugenia: Licenciada en nutrición. Estudiante del doctorado en Ciencias Biomédicas de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario. Becaria doctoral del CONICET.

Lombarte, Mercedes: Licenciada en biotecnología y Doctora en Ciencias Biomédicas. Investigadora del Laboratorio de Biología Ósea y del Centro Universitario de Estudios Medioambientales y docente de la cátedra de Química Biológica de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario.

Lupión, Patricia: Licenciada en biotecnología. Estudiante del doctorado en Ciencias Biomédicas de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario. Becaria doctoral del CONICET.

Lupo, Maela: Licenciada en biotecnología y Doctora en Ciencias Biomédicas. Investigadora del Laboratorio de Biología Ósea y del Centro Universitario de Estudios Medioambientales y docente de la cátedra de Química Biológica de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario.

Rigalli, Alfredo: Bioquímico y Doctor en bioquímica. Investigadora del Laboratorio de Biología Ósea y del Centro Universitario de Estudios Medioambientales y docente de la cátedra de Química Biológica de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario. Investigador independiente del Consejo de Investigaciones de la UNR y del CONICET

Tabla de contenidos

ORGANIZACIÓN DE LA OBRA

Esta obra está dividida en módulos y clases. Cada módulo agrupa temas diferentes. Brevemente

Módulo 1: introducción al manejo de objetos y funciones en R.

Módulo 2: introducción al uso de bibliotecas gráficas.

Módulo 3: introducción a la estadística básica.

Módulo 4: análisis multivariado de datos numéricos y análisis especiales de datos.

Módulo 5: desarrollo de scripts y programación en R.

Cada módulo se divide en 9 clases, las cuales constan de tablas específicas para cada clase, así como de un vídeo y una ejercitación. Al final de las 9 clases existe un examen final del módulo.

Las clases llevarán el nombre Clase1- seguido de un número de 1-9 si son clases del módulo 1, por ejemplo. Así tendrá clases Clase2-3, Clase4-1, etc según sean la clase 3 del módulo 2 o la clase 1 del módulo 4.

Las planillas de cálculo en formatos ods o xls llevarán la denominación tablaR1-3.ods por ejemplo si es la planilla de cálculo para la clase 3 del módulo 1. En el interior de la planilla hallará tablas con los nombre tablaR131, tablaR132, tablaR133, etc. Todas tablas para el módulo 1 (primer número), de la clase 3 (segundo número) y el tercer número indica el número de tabla. Con estos nombres serán introducidos como objetos en el espacio de trabajo.

Al principio de cada clase hallará un link al vídeo sobre la clase y tendrá un link a la planilla de cálculo con las tablas para el desarrollo de la clase.

1. Clase 2.1

Vídeo: https://youtu.be/WJlVyi8YtTw

Tabla de datos: http://hdl.handle.net/2133/10502

1.1. Revisión ingreso de datos y auditoría

A continuación veremos las principales estadísticas descriptivas de una muestra, de uso común y algunas de menos uso. Para ellos introduciremos datos y realizaremos su auditoría previa.

Supongamos que tenemos una muestra de datos en que se han medido diferentes variables. utilizaremos la tablaR211 de la planilla de cálculo tablaR2-1.xls/ods. Los datos de esta planilla surgen de la medición en diferentes semillas de diferentes variables: el tratamiento aplicado (tratados (t), controles (c)), longitud (medida del largo de la raíz a los 7 días, la que se halla expresada en mm) y si se produjo la germinación (si-no).

En primer lugar introducimos los datos. Tratándose se una tabla pequeña podemos hacerlo a través del portapapeles

```
> tablaR211<-read.table("clipboard",header=TRUE,dec=",",sep="\t")
```

```
> tablaR211
   numero germinacion tratamiento longitud
1       1         si           c     23.1
2       2         si           c     23.0
3       3         no           c       NA
4       4         si           c     22.9
5       5         si           c     21.0
6       6         si           c     21.8
7       7         si           c     20.0
8       8         no           c       NA
9       9         si           c     19.0
10     10         si           c     21.0
11     11         no           c       NA
12     12         si           c     22.0
13     13         si           c     22.0
14     14         si           c     21.9
15     15         si           c     23.2
16     16         no           c       NA
17     17         si           c     21.4
18     18         si           c     20.2
19     19         no           c       NA
20     20         si           c     20.8
21     21         si           c     20.0
22     22         si           c     20.0
23     23         si           c     20.0
24     24         no           c       NA
25     25         si           t     19.0
26     26         si           t     18.7
27     27         no           t       NA
28     28         si           t     13.0
29     29         si           t     13.0
30     30         no           t       NA
```

31	31	si	t	25.0
32	32	si	t	15.8
33	33	si	t	15.0
34	34	si	t	15.0
35	35	no	t	NA
36	36	no	t	NA
37	37	si	t	12.9
38	38	si	t	15.0
39	39	no	t	NA
40	40	si	t	15.0
41	41	si	t	15.0
42	42	si	t	12.8
43	43	si	t	13.0
44	44	no	t	NA
45	45	no	t	NA
46	46	si	t	14.0
47	47	si	t	14.0
48	48	no	t	NA
49	49	si	t	14.8
50	50	si	t	13.0

Auditoría de datos. Existen diversas funciones que se recomiendan utilizar para asegurarse del correcto ingreso de los datos.

1- tipo de objeto

La función str(), nos indica el tipo de objetos, la cantidad de observaciones y el número de variables. Para nuestro caso son 4 columnas o variables y tenemos 50 líneas u observaciones. Luego nos da detalles de cada columna. Por ejemplo, la columna número son valores enteros, los que figuran como int. En cambio germinación (si-no) y tratamiento (t o c) con factores cada uno de ellos con dos niveles, que veremos aparecen en la descripción

```
> str(tablaR211)
'data.frame'        : 50 obs. of  4 variables:
 $ numero         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ germinacion  : Factor w/ 2 levels "no","si": 2 2 1 2 2 2 2 1 2 2 ...
 $ tratamiento    : Factor w/ 2 levels "c","t": 1 1 1 1 1 1 1 1 1 1 ...
 $ longitud       : num  23.1 23 NA 22.9 21 21.8 20 NA 19 21 ...
```

La función is.data.frame(), cuya respuesta es TRUE o FALSE, nos indica con TRUE si los datos ingresaron como un data.frame y FALSE si es cualquier otro tipo de objeto. Es normal que las tablas ingresen como data.frame de manera predeterminada.

```
> is.data.frame(tablaR211)
[1] TRUE
```

2- tipo de variables: continuas, factores, caracter, numérica

La función summary() nos da información sobre las variables. Como se puede ver en nuestro caso, la variable numero es una variable numérica que fue tomada como continua y esto queda detallado en que nos indica el mínimo, 1er cuartilo, mediana, 3er cuartilo y máximo. En cambio para las variables que son factores nos indica el nombre del factor y la cantidad de elementos con el valor de cada factor. En nuestro caso por ejemplo, la variable germinacion tiene 14 unidades experimentales que no germinaron (no) y 36 que sí lo han hecho (si). Para las otras variables la interpretación es similar. Puede hacerse una aclaración en la variable longitud. Como se verá es una variable continua y nos indica las mismas estadísticas que para número. Sin embargo, abajo

vemos NA's: 14. Esto significa que 14 líneas no tienen valor de longitud. En este caso en particular corresponde a aquellas semillas que no germinaron y por lo tanto no se midió la longitud de la raiz.

```
> summary(tablaR211)
    numero         germinacion  tratamiento    longitud    
 Min.   : 1.00   no:14       c:24        Min.   :12.80  
 1st Qu.:13.25   si:36       t:26        1st Qu.:14.95  
 Median :25.50                           Median :19.50  
 Mean   :25.50                           Mean   :18.26  
 3rd Qu.:37.75                           3rd Qu.:21.50  
 Max.   :50.00                           Max.   :25.00  
                                         NA's   :14
```

Número de filas y columnas. El número de filas y columnas debe coincidir con las de la tabla original. Las funcion nrow() y ncol() son útiles para obtener esta información.

```
> nrow(tablaR211)

[1] 50

> ncol(tablaR211)

[1] 4
```

Una vez realizadas las comprobaciones, podemos pasar a realizar el cálculo de estadísticas descriptivas y a comenzar la formulación de gráficas con el fin de tener una idea de los datos que disponemos

1.2. Estadísticas descriptivas más comunes

1.2.1. media

La media de una muestra de datos se calcula con la función mean()

```
> mean(tablaR211$longitud,na.rm=TRUE)

[1] 18.25833
```

recuerde que cuando en una columna tiene valores NA, debe excluirlos en el cálculo para ello el argumento na.rm=TRUE, permite obtener el valor de la estadística excluyendo aquellos valores NA. Si no excluye los valores, el valor de la media dará como resultado NA.

1.2.2. desvío estándar

El desvío estándar de una muestra puede calcularlo con la función sd()

```
> sd(tablaR211$longitud,na.rm=TRUE)

[1] 3.840266
```

1.2.3. variancia

La variancia se calcula con la función var()

```
> var(tablaR211$longitud,na.rm=TRUE)

[1] 14.74764
```

otra forma de cálculo

puede ser elevando al cuadrado el desvío estándar

```
> (sd(tablaR211$longitud,na.rm=TRUE))^2

[1] 14.74764
```

1.2.4. mediana

La mediana de una muestra de datos se calcula con la función median().

```
> median(tablaR211$longitud,na.rm=TRUE)

[1] 19.5
```

1.2.5. rango

El rango o valor mínimo y máximo de una muestra se calcula con la función range()

```
> range(tablaR211$longitud,na.rm=TRUE)

[1] 12.8 25.0
```

1.2.6. percentilos

```
> quantile(tablaR211$longitud,na.rm=TRUE)
   0%    25%    50%    75%  100%
12.80  14.95  19.50  21.50  25.00
```

si deseara un percentilo en particular, por ejemplo aquel valor de la variable que acumula el 30 % de valores menores a él (percentilo 30)

```
> quantile(tablaR211$longitud,probs=0.30,na.rm=TRUE)

30%

 15
```

1.2.7. modo

Recuerde que R no tiene función que calcule el modo o moda, es decir el valor que más veces se presenta. No confunda la función mode() con el cálculo del modo o moda.

Para calcular el modo, puede transformar los datos a caracter y factor y luego pedir un summary. Allí le indica la cantidad de veces que se presenta cada valor.

Supongamos que trabajamos con la columna longitud. Creamos un data.frame al que llamamos modo, ordenamos la variable longitud (con la función sort()) y la transformamos en caracter con las función as.character()

```
> modo<-data.frame(as.factor(as.character(sort(tablaR211$longitud))))
```

luego pedimos un summary() del data.frame modo y nos mostrará cuantas veces está repetido cada valor. En neustro caso vemos que el valor 15 aparece 5 veces, entonces 15 es la moda de nuestra muestra

```
> summary(modo)
 as.factor.as.character.sort.tablaR211.longitud...
15     : 5
 13     : 4
 20     : 4
 14     : 2
 19     : 2
 21     : 2
 (Other):17
```

1.2.8. max

El máximo valor de una muestra numérica se halla con la función max()

> max(tablaR211$longitud,na.rm=T)

[1] 25

1.2.9. min

El mínimo valor de un conjunto de números se calcula con la función min()

> min(tablaR211$longitud,na.rm=T)

[1] 12.8

1.2.10. intervalo intercuartilos

El intervalo intercuartilo es una medida de dispersión de los datos y es la diferencia entre el cuartilo 25 y el 75%

> IQR(tablaR211$longitud,na.rm=T)

[1] 6.55

podemos comprobar qué es esta estadística calculando los percentilo.

```
> quantile(tablaR211$longitud,na.rm=TRUE)
   0%    25%    50%    75%  100%
12.80  14.95  19.50  21.50  25.00 ------> 21,5-14,95= 6,55
```

1.2.11. mediana de las desviaciones absolutas

Es una medida robusta de la dispersión de datos. Calcula la mediana de las desviaciones absolutas de cada valor respecto de un valor (center) No es tan influenciada como el desvío estándar por valores extremos.

Podemos calcularla centrada en la mediana

> mad(tablaR211$longitud,center=median(tablaR211$longitud,na.rm=TRUE),na.rm=T)

[1] 5.26323

o en la media

> mad(tablaR211$longitud,center=mean(tablaR211$longitud,na.rm=TRUE),na.rm=T)

[1] 4.979065

medias con peso

Supongamos que los valores de las mediciones no son afectados por el mismo error y cuanto más pequeña la medición, mayor es el error. En virtud de esto deseamos que no todos los valores tengan el mismo peso sobre el cálculo de la media.

Supongamos que el peso del error sobre la medición ha sido modelizado y que dicho estudio indicó que cuanto más pequeño el valor menor es el peso que debería tener.

Se obtuvieron pesos (no importa por que se eligió dicho modelo) para los valores medidos que figuran en la tabla tablaR212. Introducimos en nuestro espacio de trabajo la tablaR212

> tablaR212<-read.table("clipboard",header=TRUE,dec=",",sep="\t")

> tablaR212

	medicion	peso
1	0.1	0.004347826
2	0.2	0.008695652
3	0.2	0.008695652
4	0.2	0.008695652
5	0.4	0.017391304
6	0.5	0.021739130
7	0.5	0.021739130
8	23.0	1.000000000
9	21.0	0.913043478
10	20.0	0.869565217
11	20.0	0.869565217
12	19.0	0.826086956
13	15.0	0.652173913
14	12.0	0.521739130
15	9.0	0.391304348
16	5.0	0.217391304

Calculemos primero la media de los datos de la columna medición. Recuerde que para seleccionar una columna de un data.frame debe escribir el nombre del data.frame separado del nombre de la columna por $

```
> mean(tablaR212$medicion)
```

[1] 9.13125

Si calculamos la media dando importancia al peso

```
> weighted.mean(tablaR212$medicion,tablaR212$peso)
```

[1] 17.84251

vemos que la media es más alta, ya que los pesos asigna valores menores de peso a los valores menores de la medición. También podrían ser valores de peso al revés o cualquier otro modelo que se debiera aplicar.

1.2.12. frecuencias absolutas y relativas

1.2.12.1 frecuencias absoluta

Supongamos que queremos conocer cuantas unidades experimentales de la tablaR211 tienen tratamiento (t) y cuantos son controles (c)

```
> table(tablaR211$tratamiento)
 c  t
24 26
```

el código siguiente nos permite saber cuantos son c y cuantos t, pero sin incluir aquellos que no han tenido medición de longitud. En los casos que no se midió la longitud, en esta columna figurará NA. Si quisiéramos conocer cuantos son c y cuanto t, pero dentro de aquellos en que se pudo medir la longitud, podemos utilizar el siguiente código. Agregamos la condición de que la columna longitud sea diferente de NA. "Diferente de" se escribe en R como !=.

```
table(tablaR211$tratamiento[tablaR211$longitud!="NA"])
 c  t
18 18
```

1.2.12.2 frecuencia relativa

La frecuencia relativa no es otra cosa que la razón entre el número de unidades de un factor dividido el número total de unidades. Dicho de otra manera el tanto por uno.

```
> prop.table(table(tablaR211$tratamiento))
   c    t
0.48 0.52
```

1.2.12.3 porcentaje

Si deseáramos los porcentajes de unidades para cada nivel del factor, solo debemos multiplicar por 100 la frecuencia relativa.

```
> prop.table(table(tablaR211$tratamiento))*100
c  t
48 52
```

48 % de las unidades experimentales han sido controles (c) y 52 % tratadas (t)

1.3. tapply()

Esta función le permite calcular estadísticas por grupos. Veamos en la tablaR211.

Supongamos que deseamos conocer la media de la longitud pero agrupados por tratamiento (c o t). Es decir deseamos conocer la media de las longitudes de los controles y de los tratados.

```
> tapply(tablaR211$longitud,factor(tablaR211$tratamiento),mean,na.rm=TRUE)
       c        t
21.29444 15.22222
```

si deseáramos el desvío estándar: sd, aplicamos la misma función salvo que en lugar de mean utilizamos sd.

```
> tapply(tablaR211$longitud,factor(tablaR211$tratamiento),sd,na.rm=TRUE)
       c        t
1.273934 3.036359
```

De igual manera para el rango

```
> tapply(tablaR211$longitud,factor(tablaR211$tratamiento),range,na.rm=TRUE)
$c
[1] 19.0 23.2

$t
[1] 12.8 25.0
```

1.4. round()

La función round() permite redondear con diferente número de decimales (argumento digits=)

Combinemos round con tapply. Primero obtenemos las medias de las longitudes de los grupos c y t

```
> tapply(tablaR211$longitud,factor(tablaR211$tratamiento),mean,na.rm=TRUE)
       c        t
21.29444 15.22222
```

ahora hacemos lo mismo pero lo redondeamos

```
> round(tapply(tablaR211$longitud,factor(tablaR211$tratamiento),mean,na.rm=TRUE),digits=1)
   c    t
21.3 15.2
```

1.5. subset()

La función subset() es una poderosa herramienta que permite seleccionar valores según diversos criterios. El uso adecuado y consciente de esta función permite observaciones casi imposibles a simple vista . Primero veamos como funciona en grado creciente de complejidad

Deseamos seleccionar las unidades con su tratamiento y longitud, pero solo de aquellas que tienen tratamiento c.

```
> subset(tablaR211,tratamiento=="c",select=c(tratamiento,longitud))
   tratamiento longitud
1            c     23.1
2            c     23.0
3            c       NA
4            c     22.9
5            c     21.0
6            c     21.8
7            c     20.0
8            c       NA
9            c     19.0
10           c     21.0
11           c       NA
12           c     22.0
13           c     22.0
14           c     21.9
15           c     23.2
16           c       NA
17           c     21.4
18           c     20.2
19           c       NA
20           c     20.8
21           c     20.0
22           c     20.0
23           c     20.0
24           c       NA
```

Ahora la selección pretende quedarse con los tratamientos y longitudes pero solo de aquellas con tratamiento c pero que su longitud ha podido ser medida.

```
> subset(tablaR211,tratamiento=="c" & longitud !="NA",select=c(tratamiento,longitud))
   tratamiento longitud
1            c     23.1
2            c     23.0
4            c     22.9
5            c     21.0
6            c     21.8
7            c     20.0
9            c     19.0
10           c     21.0
12           c     22.0
13           c     22.0
14           c     21.9
15           c     23.2
17           c     21.4
18           c     20.2
20           c     20.8
21           c     20.0
22           c     20.0
23           c     20.0
```

Ahora a la selección le exigimos no solo que haya sido medida y pertenezcan al tratamiento "c", sino que también ese valor sea mayor a 22 mm.

```
> subset(tablaR211,tratamiento=="c" & longitud !="NA" & longitud >22,select=c(tratamiento,longitud))
  tratamiento longitud
1           c     23.1
2           c     23.0
4           c     22.9
15          c     23.2
```

Si quisiéramos conocer la media de los valores de longitud pero solo de los controles "c" a los que se ha medido la longitud y la misma fue mayor que 22

```
> mean(subset(tablaR211,tratamiento=="c" & longitud !="NA" & longitud >22,select=longitud)$longitud)
[1] 23.05
```

1.6. Calculo de estadísticas por filas y columnas

```
> mean(tablaR211[,4],na.rm=TRUE)

[1] 18.25833

> mean(tablaR211[tablaR211$tratamiento=="c",4],na.rm=TRUE)

[1] 21.29444
```

2. Clase 2

Vídeo: https://youtu.be/3wE367uo3CE

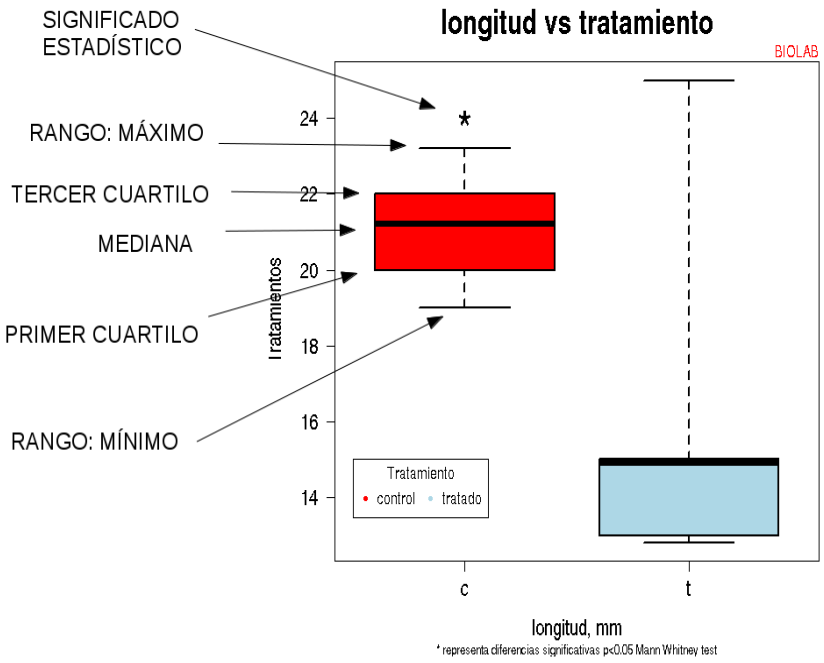
Tabla de datos: http://hdl.handle.net/2133/10514

En este módulo veremos la construcción de gráficos utilizando funciones o bibliotecas disponibles en R. En el módulo 5 veremos la construcción de gráficos con el uso de scripts, estos últimos permiten construir el gráfico con los requerimientos personales, sin necesidad de adaptarse a nada prearmado. Iremos construyendo de a pasos un gráfico y conociendo argumentos, que asimilaremos de observar la gráfica obtenida. La mayoría de los argumentos utilizados son comunes a todos los tipos de gráficos.

2.1. Gráficos boxplot

El gráfico boxplot o box and whisker muestra básicamente la mediana, rango, primer y tercer cuartilo (percentilo 25 y 75%) de una muestra. Puede presentar varias muestras a la vez y permitir su comparación a través del ensayo estadístico correspondiente.

La figura siguiente muestra sus partes principales



Para este tipo de gráfica utilizaremos la planilla de cálculo tablaR2-2.ods/xls, hoja tablaR221.

Introduzca los datos de dicha planilla con el nombre tablaR221 y haga las auditorías correspondientes (ejercite las buenas prácticas de computación)

```
tablaR221<-read.table("clipboard",header=TRUE,dec=",",sep="\t")
```

vemos la tabla

```
> tablaR221
   numero germinacion tratamiento longitud
1       1          si          c     23.1
2       2          si          c     23.0
3       3          no          c       NA
4       4          si          c     22.9
5       5          si          c     21.0
6       6          si          c     21.8
7       7          si          c     20.0
8       8          no          c       NA
9       9          si          c     19.0
10     10          si          c     21.0
11     11          no          c       NA
12     12          si          c     22.0
13     13          si          c     22.0
14     14          si          c     21.9
15     15          si          c     23.2
16     16          no          c       NA
17     17          si          c     21.4
18     18          si          c     20.2
19     19          no          c       NA
20     20          si          c     20.8
21     21          si          c     20.0
22     22          si          c     20.0
23     23          si          c     20.0
24     24          no          c       NA
25     25          si          t     19.0
26     26          si          t     18.7
27     27          no          t       NA
28     28          si          t     13.0
29     29          si          t     13.0
30     30          no          t       NA
31     31          si          t     25.0
32     32          si          t     15.8
33     33          si          t     15.0
34     34          si          t     15.0
35     35          no          t       NA
36     36          no          t       NA
37     37          si          t     12.9
38     38          si          t     15.0
39     39          no          t       NA
40     40          si          t     15.0
41     41          si          t     15.0
42     42          si          t     12.8
43     43          si          t     13.0
44     44          no          t       NA
45     45          no          t       NA
46     46          si          t     14.0
47     47          si          t     14.0
```

```
48    48          no          t      NA
49    49          si          t    14.8
50    50          si          t    13.0
```

pedimos un resumen de la misma para ver columnas y tipo de datos

```
> summary(tablaR221)
     numero       germinacion tratamiento    longitud
 Min.   : 1.00       no:14      c:24       Min.   :12.80
 1st Qu.:13.25       si:36      t:26       1st Qu.:14.95
 Median :25.50                             Median :19.50
 Mean   :25.50                             Mean   :18.26
 3rd Qu.:37.75                             3rd Qu.:21.50
 Max.   :50.00                             Max.   :25.00
                                           NA's   :14
```

comprobamos que los datos de la columna numero y longitud son numéricos, mientras que germinación y tratamiento son categóricos. Lo mismo podemos comprobar con la función str()

```
> str(tablaR221)
'data.frame':        50 obs. of  4 variables:
 $ numero     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ germinacion: Factor w/ 2 levels "no","si": 2 2 1 2 2 2 2 1 2 2 ...
 $ tratamiento: Factor w/ 2 levels "c","t": 1 1 1 1 1 1 1 1 1 1 ...
 $ longitud   : num  23.1 23 NA 22.9 21 21.8 20 NA 19 21 ...
```

str() nos da más información como qué tipo de objeto tenemos (data.frame), cuantos datos en total y cuantas variables y luego las columnas (identificadas con $). Es conveniente siempre comprobar si el número de filas y columnas es coincidente con la planilla de cálculo utilizada

```
> nrow(tablaR221)
[1] 50
> ncol(tablaR221)
[1] 4
```

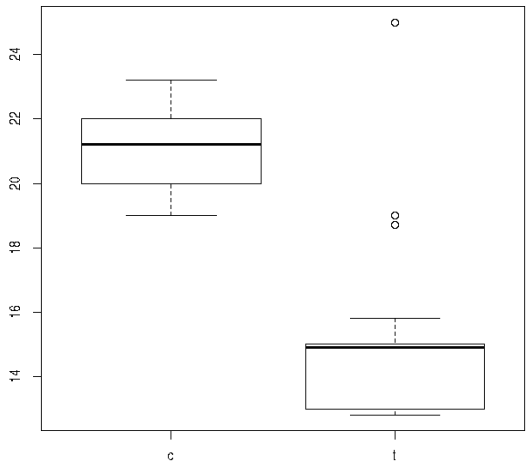
Hagamos entonces nuestro primer gráfico con mínimos argumentos. Solo indicamos el tipo de gráfico, las variables involucradas y la tabla.

En primer lugar graficaremos la variable longitud, que es una variable continua para los dos niveles (c y t) de la variable tratamiento. A las variables que pueden adquirir dos o más valores categóricos, las llamaremos también "factores" y cada valor de esta variable lo llamaremos "nivel". En este caso el factor tratamiento tiene dos niveles: c y t.

Para realizar un gráfico del tipo boxplot utilizamos la función de R boxplot().

```
> boxplot(longitud~tratamiento,data=tablaR221)
```

El signo ~ significa "en función de". Es decir que estamos graficando la variable longitud en función de los niveles del factor tratamiento.



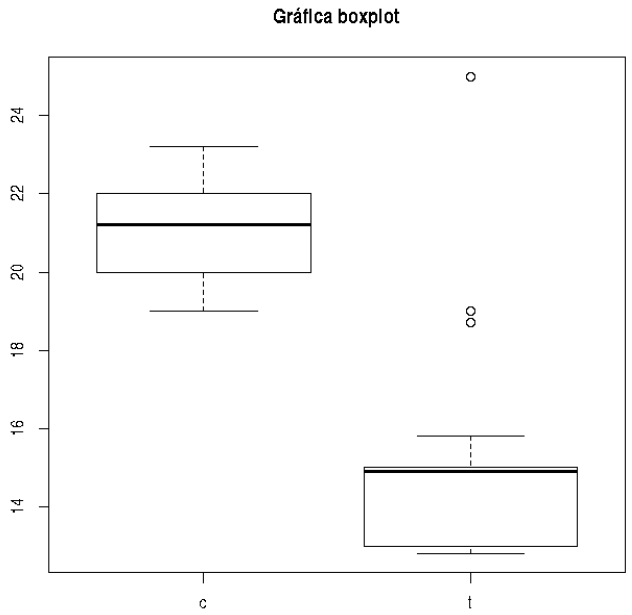
Esta gráfica muestra la mediana (linea horizontal oscura en la caja), percentilo 25 y 50 (bordes inferior y superior de la caja, respectivamente), rango de datos (borde inferior y superior del bigote).

En esta figura hay puntos outliers (ya veremos como incluirlos en los bigotes).

La gráfica obtenida es rudimentaria, puede mejorarse indefinidamente y comenzaremos con ello paso a paso. Es importante recordar que los argumentos que utilizaremos para mejorar estas gráficas son aplicables luego a otras gráficas. Llamamos argumentos a datos que colocamos dentro de la función para hacer la gráfica. En el caso siguiente main, es un argumento.

Veamos nuevos argumentos agregados sobre el gráfico realizado. En cada caso se muestran resaltado en amarillo los nuevos argumentos respecto del gráfico anterior (compare siempre con el gráfico anterior)

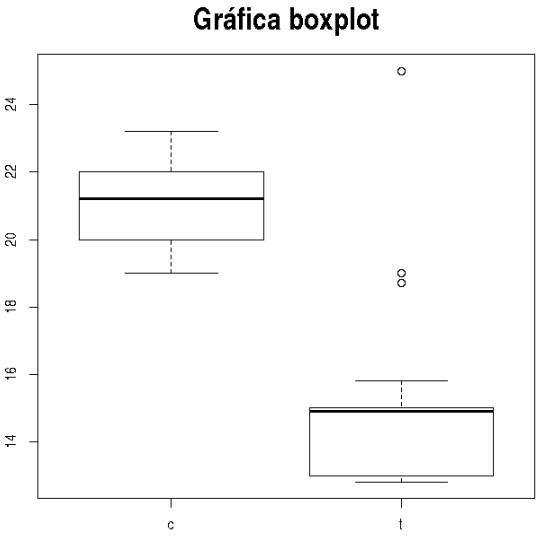
> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot")



El argumento main permite colocar un título al gráfico

introducimos ahora el argumento cex.main, que permite cambiar el tamaño de la letra del título.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2)

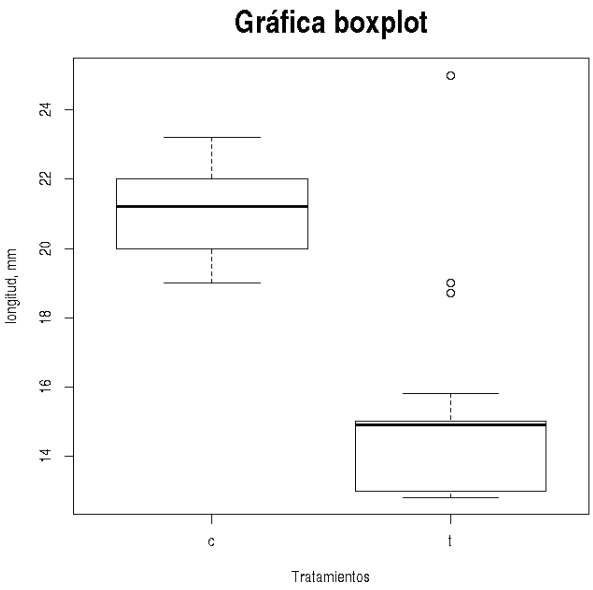


El argumento cex.main permite modificar el tamaño del título colocado con el argumento main.

A continuación introducimos los argumentos xlab e ylab.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm")

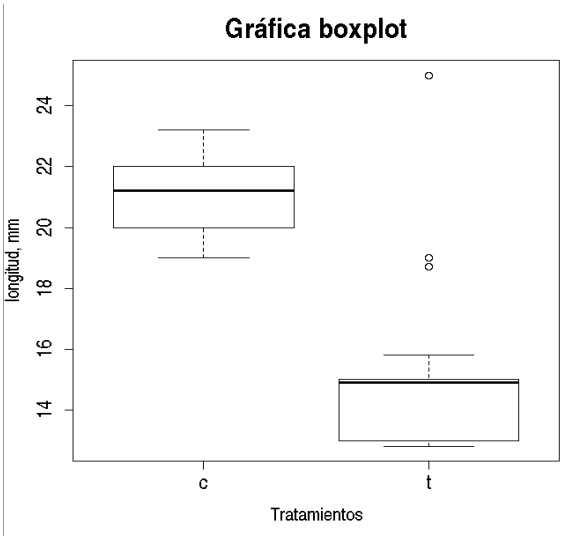
El argumento xlab permite colocar un rótulo al eje x y el argumento ylab, colocar el rótulo al eje y.



Introducimos argumentos cex.lab y cex.axis.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5)

El argumento cex.lab permite cambiar el tamaño de los rótulos de los ejes y el argumento cex.axis el tamaño de los números o letras utilizados para los ticks en los ejes.

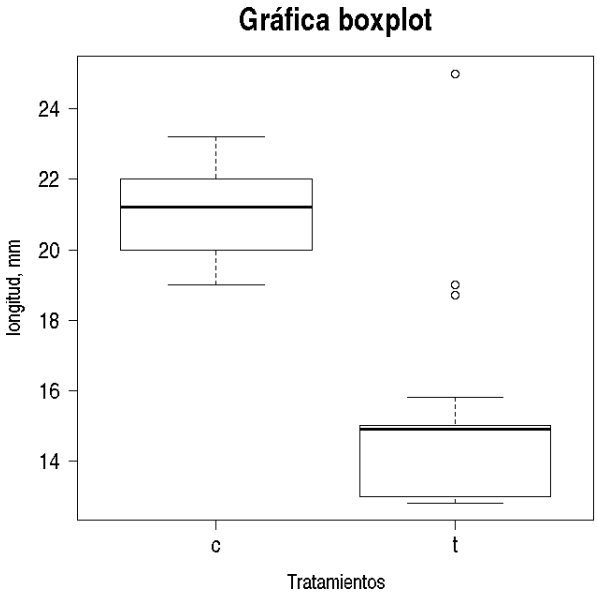


El argumento las.

boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica

boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1)

El argumento las permite rotar los números en los ejes colocándolos paralelos o perpendiculares a ellos.



Según el valor del argumento las se obtienen diferentes resultados

las= 0 rótulos de escala de los ejes paralelos a ambos ejes

las= 1 rótulos de eje y perpendicular al eje, rótulos de x paralelos al eje

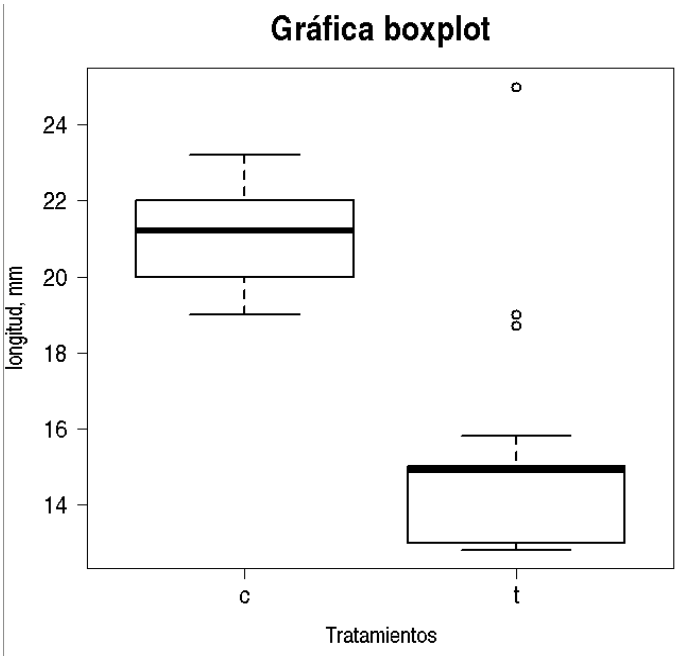
las= 2 rótulos de los eje perpendiculares a ambos ejes.

las =3 rótulos de eje y paralelo al eje, rótulos de x perpendicular al eje

Introducción del argumento lwd (line width)

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2)

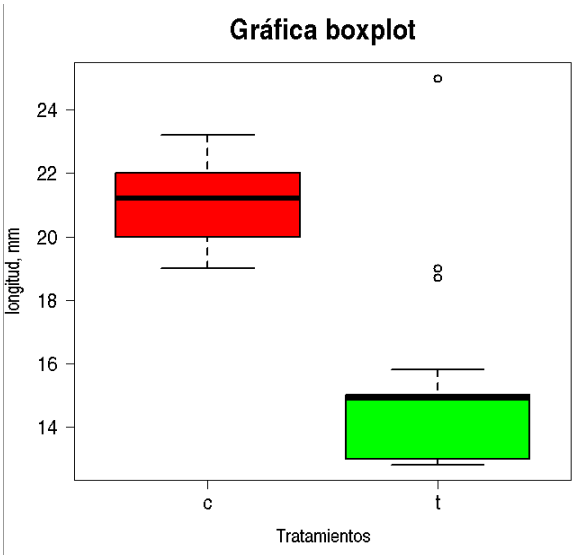
El argumento lwd cambia el grosor de las líneas de las cajas y bigotes



Introducción del argumento col (color).

>boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"))

El argumento col permite cambiar el color de los datos. Si se coloca col="red", todos los datos serán rojos, pero si utilizamos un vector como en el caso mostrado, asignaremos "red" a un nivel y "green" al otro.



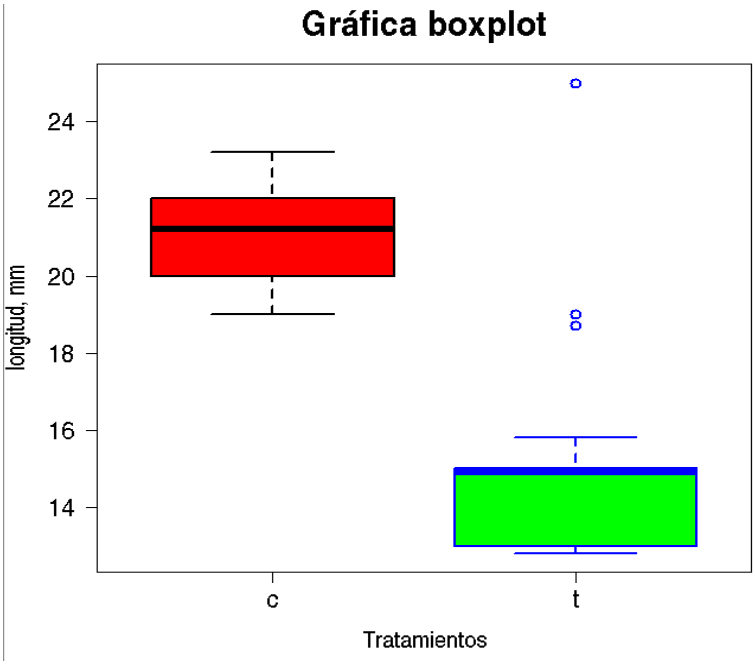
Para conocer los colores disponibles escribir en linea de comandos colours(). En los gráficos se puede especificar el color con números. Para el ejemplo mostrado podría escribirse

boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("2","3"))

El argumento border.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"))

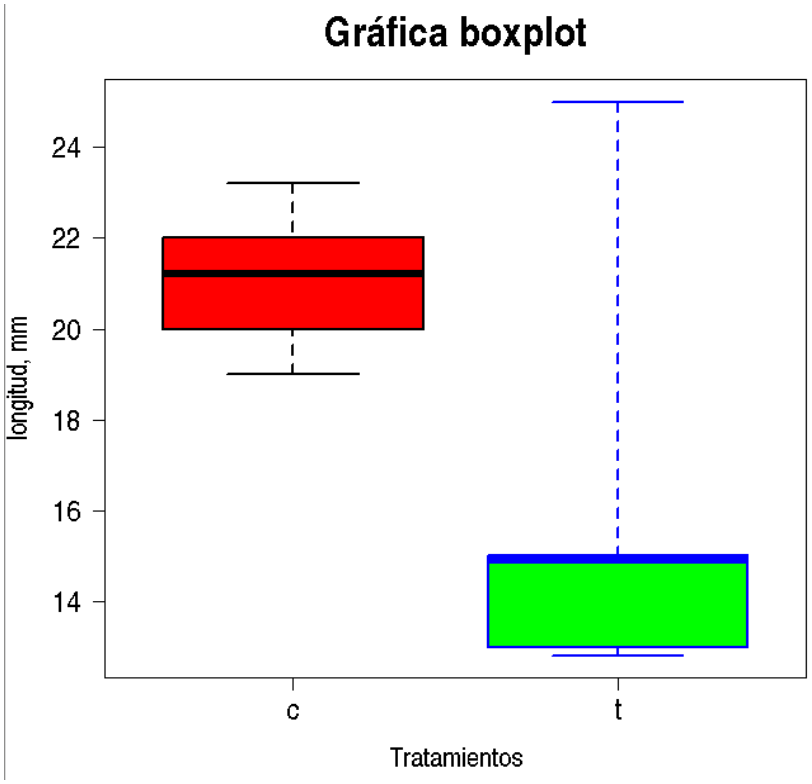
El argumento border permite asignarle colores a los bordes de las cajas



El argumento range

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0)

El argumento range es de difícil manejo. Un uso sencillo es: si se coloca range=0, no muestra datos outliers sino que muestra los bigotes hasta el máximo y mínimo. Si no se coloca el argumento range, mostrará outliers, pero es complejo aun comprender como fijarlo.

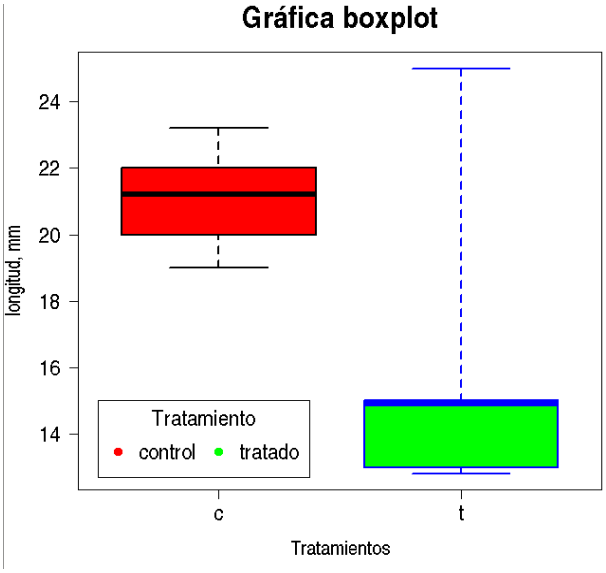


Ahora agregaremos una leyenda al gráfico. En primer lugar se realiza el gráfico para lo cual ejecutamos el comando anterior y a continuación la función legend()

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="Tratamientos",ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0)

> legend(0.5,15,pch=c(20,20),legend=c("control","tratado"),title="Tratamiento",horiz=T,col=c("red","green"),cex=1.5)

Al ejecutar la función legend() luego de un gráfico permitirá colocar una leyenda con las series en la gráfica. Ver explicaciones luego del gráfico



Argumentos de legend:

0.5 es la posición respecto del eje x,

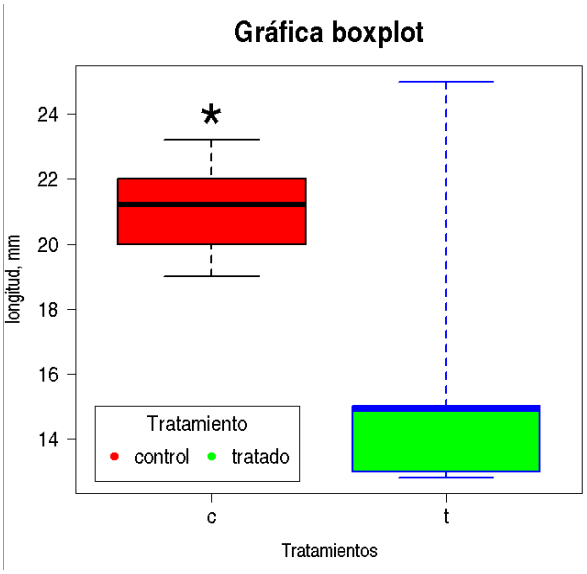
15 la posición respecto del eje y, en ambos casos las medidas corresponden al vértice superior derecho de la leyenda.

pch=20, es el tipo de símbolo para colocar el color de la serie.

horiz: T, ubica las leyendas horizontales. F, las coloca una debajo de la otra

Si agregamos la línea de comando que se indica a continuación, podremos colocar un asterisco (o lo que desee) en este caso sobre los datos del nivel c.

```
> text(1,24,"*",cex=5)
```



1 y 24 son las posiciones de x e y respectivamente donde aparece el texto, en este caso el *.

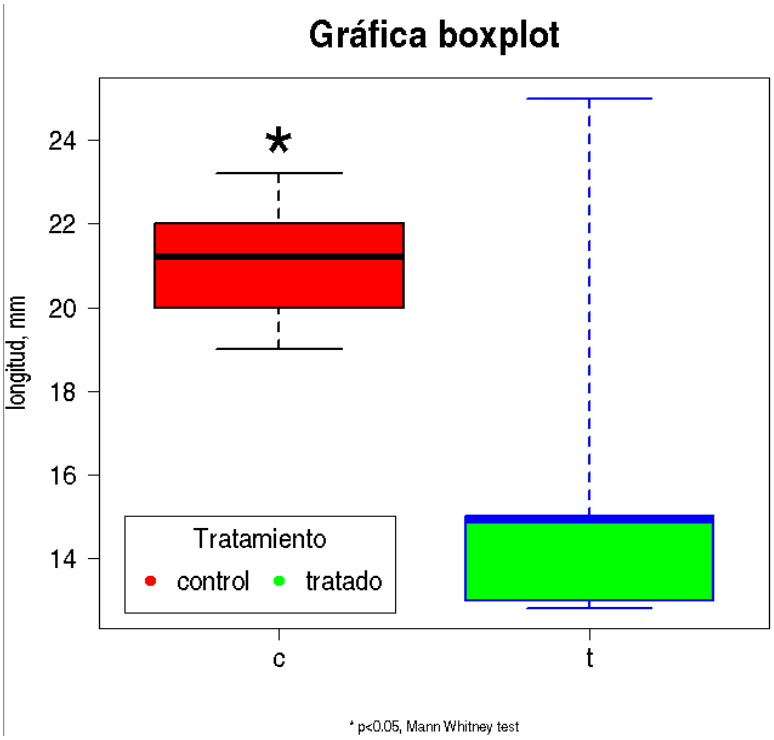
Veamos otras modificaciones. Eliminamos argumento xlab y agregamos el argumento sub, que cumplirían mas o menos la misma función.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,~~xlab="Tratamientos"~~,ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,sub="* p<0.05, Mann Whitney test", cex.sub=0.8)

>legend(0.5,15,pch=c(20,20),legend=c("control","tratado"),title="Tratamiento",horiz=T,col=c("red","green"),cex=1.5)

> text(1,24,"*",cex=5)

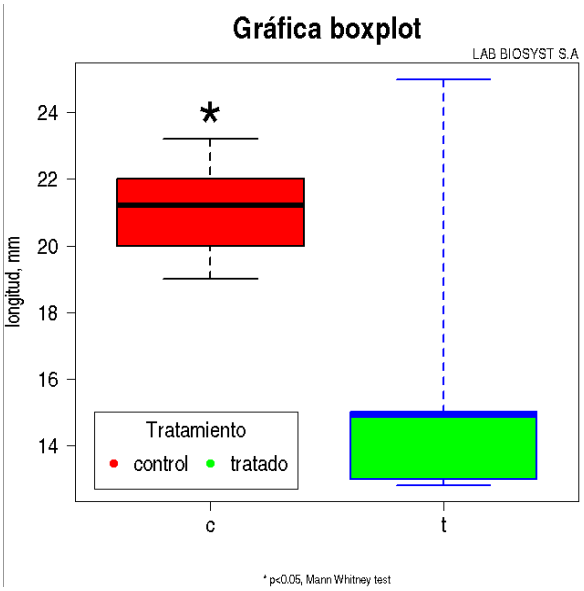
sub: coloca un texto como pie de la figura. cex.sub: como todo argumento cex. fija tamaño



Utilicemos ahora la función mtext(), como se muestra en la siguiente línea de comando

```
> mtext("LAB BIOSYST S.A", adj=1,col="black",cex=1)
```

coloca un texto sobre la línea superior del gráfico. El argumento adj=0 permite colocar el texto especificado a la izquierda. Si el argumento adj=0.5, el texto quedará en el medio y si el valor es adj=1, el texto quedará a la derecha. Note lo lógico de los valores del argumento: 0, 0.5, 1. Así es R. Cuando comience a interiorizarse verá que todo fluye naturalmente.



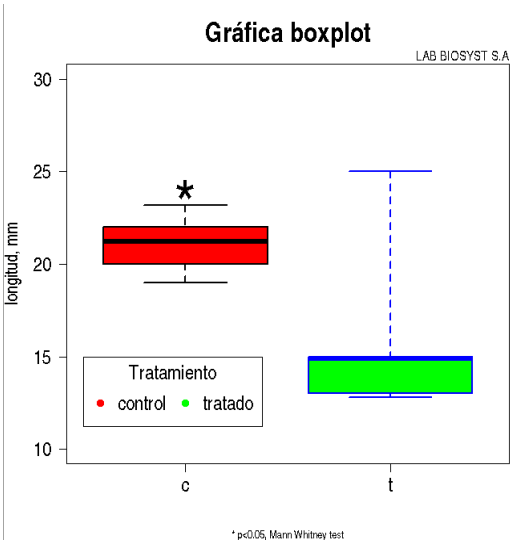
Con el argumento ylim podemos fijar los límites del eje y, en el ejemplo siguiente fijaremos el eje y en valores comprendidos entre 10 y 30.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,ylab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,sub="* p<0.05, Mann Whitney test", cex.sub=0.8,ylim=c(10,30))

> legend(0.5,15,pch=c(20,20),legend=c("control","tratado"),title="Tratamiento",horiz=T,col=c("red","green"),cex=1.5)

> text(1,24,"*",cex=5)> mtext("LAB BIOSYST S.A", adj=1,col="black",cex=1)

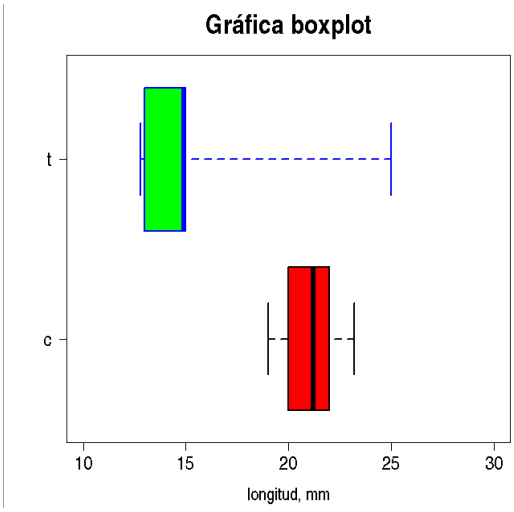
La forma general del argumento es: ylim=c(valor inferior, valor superior). Es un vector con que se indica el límite superior e inferior del eje y. Note que en nuestro ejemplo fue c(10,30) y el eje y solo muestra valores en ese rango.



lo mismo puede hacerse con xlim=c(menor valor de eje x, mayor valor de eje x). En este tipo de gráfica no tiene sentido.

Veamos ahora el argumento horizontal. Este argumento permite hacer los box verticales u horizontales, según sea FALSE O TRUE.

> boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,ylim=c(10,30),horizontal=TRUE)

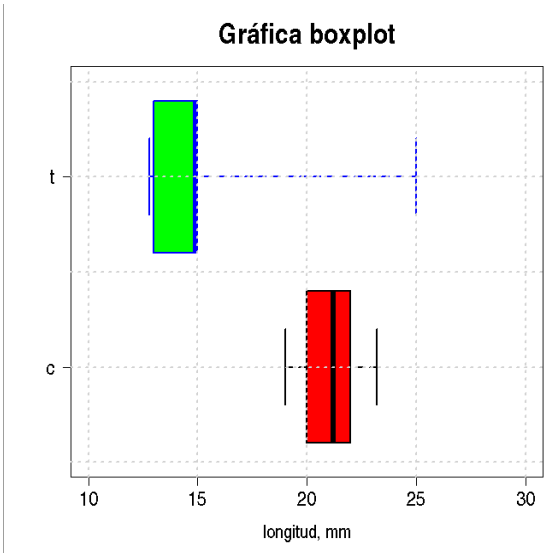


Ahora incluiremos una grilla. Esta función es un poco más compleja y tiene varios argumentos que fijar. Primero se hace el gráfico y luego con la función grid() se coloca la grilla.

>boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,ylim=c(10,30),horizontal=TRUE)

> grid(nx=NULL,ny=NULL,col="lightgray",lty="dotted",lwd=2.5)

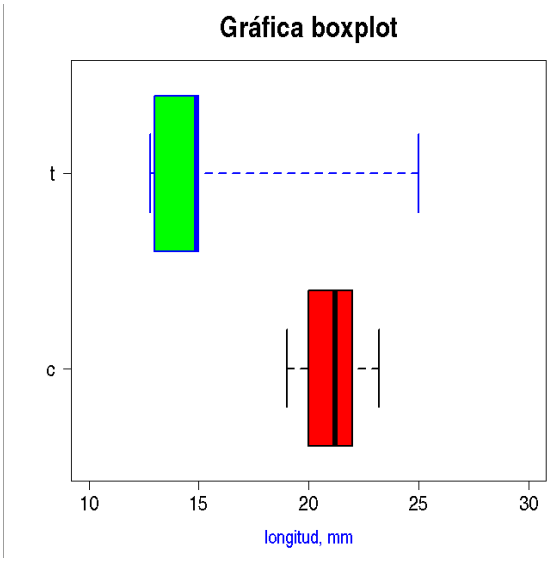
nx=NULL, hace coincidir la grilla con los ticks del eje x de la gráfica. ny=NULL hace que las lineas de la gráfica coincidan con los ticks del eje y. Luego veremos como hacer para obtener resultados diferentes. El argumento col fija el color de las líneas de la grilla. El argumento lty (line type): fija el tipo de línea (continua, puntos, etc) y el argumento lwd (line width) fija el grosor de la línea de la grilla)



Otros argumentos. Veamos col.lab

>boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,ylim=c(10,30),horizontal=TRUE,col.lab="blue")

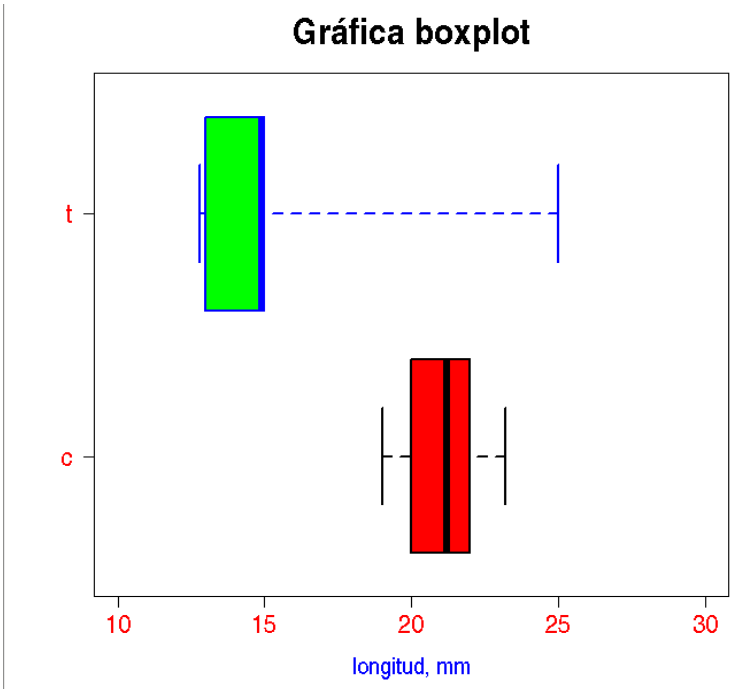
col.lab permite cambiar el color de los detalles de los ejes.



Ahora veamos el argumento col.axis

>boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica boxplot",cex.main=2,xlab="longitud, mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=0,ylim=c(10,30),horizontal=TRUE,col.lab="blue",col.axis="red")

col.axis: permite cambiar el color de los rótulos de los ejes.



2.2. Exportar gráficas en diversos formatos

Bajo Windows las gráficas se pueden grabar como cualquier archivo. Esta opción no existe en R para Linux. Sin embargo el mecanismo es sencillo. Adicionalmente estos mecanismos permiten fijar algunos parámetros de tamaño y resolución que pueden ser de mucha utilizada.

Primero se debe abrir el dispositivo de guardado de gráfico con el siguiente código si deseamos una gráfica en formato .jpg

```
>jpeg(filename= "boxplot3.jpg",width=30,height=30,units= "cm",res=300)
```

filename: lleva el nombre con que aparecerá el gráfico en el mismo directorio donde estamos ejecutando R.

width y height: indican ancho y alto

units: las unidades del ancho y el alto

res: la resolución de la imagen. este último parámetro es muy importante especialmente cuando la imagen es requerida con alguna resolución específica

Luego se hace el gráfico con los argumentos deseados.

```
>boxplot(longitud~tratamiento,data=tablaR221,main="Gráfica
boxplot",cex.main=2,xlab="longitud,
mm",cex.lab=1.3,cex.axis=1.5,las=1,lwd=2,col=c("red","green"),border=c("black","blue"),range=
0,ylim=c(10,30),horizontal=TRUE,col.lab="blue",col.axis="red")
```

Finalmente se cierra el dispositivo

```
> graphics.off()
```

y hallará la figura en formato .jpg en el directorio donde se halla el espacio de trabajo.

Si deseara guardar en otros formatos, debe abrir otros dispositivos con

Si desea el gráfico en formato tif, ejecute

```
tiff(filename= "boxplot3.tif",width=30,height=30,units= "cm",res=300)
```

mientras que si lo desea en bmp, debe ejecutar antes del gráfico

```
bmp(filename= "boxplot3.bmp",width=30,height=30,units= "cm",res=300)
```

2.3. Graficas con restricción de datos

En la tablaR221 tenemos datos de longitud para diferentes unidades experimentales donde se dan varias combinaciones de factores. Tenemos plantas que la variable germinación fue si y en otras no. A su vez tenemos plantas con tratamiento c y t. Determinando así individuos con la siguiente combinación de factores

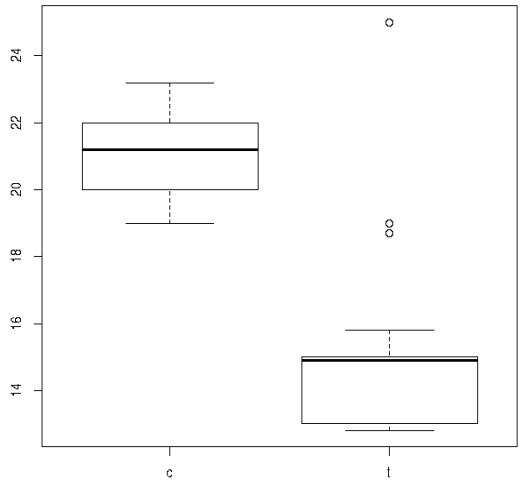
germinación	tratamiento
si	c
si	t
no	c
no	t

supongamos que deseamos hacer un boxplot de longitud en función de tratamiento pero solo para aquellas unidades donde germinación vale si. Es decir no incluir en el gráfico las unidades que no germinaron. El código básico sin argumentos ya utilizado fue

boxplot(longitud~tratamiento,data=tablaR221)

introducimos entonces la restricción.

boxplot(longitud[tablaR221$germinacion=='si']~tratamiento[tablaR221$germinacion=='si'],data=tablaR221)



2.4. Resumen de argumentos

1. boxplot(y~x,data=objeto del espacio de trabajo que contiene los datos)
2. main= "entre comillas se coloca el título de la gráfica"
3. cex.main= con un número indica el tamaño de letra del titulo.
4. font.main= con un número indica el tipo: 1- normal, 2- negrita, 3- cursiva, 4-negrita cursiva
5. sub: "entre comillas se coloca el pie de la figura"
6. cex.sub= con un número indique el tamaño de la letra de sub.
7. font.sub= con un número indica el tipo: 1- normal, 2- negrita, 3- cursiva, 4-negrita cursiva
8. xlab= "entre comillas coloque la descripción del eje x"
9. ylab= "entre comillas coloque la descripción del eje y"
10. col.lab="entre comillas el color" o bien el número obtenido con colours().
11. cex.lab= con un número indica el tamaño de los rótulos de los ejes
12. cex.axis= con un número indica el tamaño de la escala de los ejes
13. col.axis= "entre comillas el color de los rótulos de los ejes"

14. col= con un número o un vector de números (o palabras) indica el color de las series de datos
15. border= con palabras o un vector indica el color de los bordes de la caja de cada serie de datos
16. las= con un número coloca los números de las escalas de los ejes de diferente forma. las= 0 rótulos de escala de los ejes paralelos a ambos ejes. las= 1 rótulos de eje y perpendicular al eje, rótulos de x paralelos al eje. las= 2 rótulos de los eje perpendiculares a ambos ejes. las =3 rótulos de eje y paralelo al eje, rótulos de x perpendicular al eje.
17. range= 0 incluye todos los datos en las cajas y bigotes, sin colocar range deja fuera los outliers.
18. legend(con los argumentos necesarios) coloca la leyenda
19. text(x, y, "entre comillas el texto a colocar") coloca un texto dentro del grafico en la posición x y indicada en el argumento.
20. mtext("texto que se desea colocar sobre el borde superior del gráfico", adj=1 (a la derecha), col= color, cex=tamaño). adj=0 queda a la izquierda, adj=0.5 queda al medio

3. Clase 3

Vídeo: https://youtu.be/pS8J34T8EsY

Tabla de datos: http://hdl.handle.net/2133/10515

3.1. Gráficos scatter plots

En esta clase veremos las gráficas de dispersión, gráficas x-y o scatter plots.

Introduzcamos las tablaR231 de la hoja de cálculo de esta clase (tablaR2-3.ods/xls) en el espacio de trabajo

```
> tablaR231<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

Realicemos la auditoría de los datos introducidos. Siempre es una buena inversión de tiempo controlar que los datos hayan ingresado en el formato adecuado

```
> summary(tablaR231)
       x               y               z          
 Min.   : 1.00   Min.   : 1.800   Min.   :2.00  
 1st Qu.: 3.25   1st Qu.: 3.275   1st Qu.:4.25  
 Median : 5.50   Median : 5.250   Median :7.00  
 Mean   : 5.50   Mean   : 5.490   Mean   :6.10  
 3rd Qu.: 7.75   3rd Qu.: 7.000   3rd Qu .:8.00  
 Max.   :10.00   Max.   :11.000   Max.   :9.00  
> names(tablaR231)

[1] "x" "y" "z"

> ncol(tablaR231)

[1] 3

> nrow(tablaR231)

[1] 10

> head(tablaR231)
   x   y   z
1 1  1.8  8
2 2  2.5  9
3 3  3.0  4
4 4  4.5  5
5 5  4.1  7
6 6  6.0  2
> str(tablaR231)
'data.frame':        10 obs. of  3 variables:
 $ x: int  1 2 3 4 5 6 7 8 9 10
 $ y: num  1.8 2.5 3 4.5 4.1 6 7 7 8 11
 $ z: int  8 9 4 5 7 2 3 8 7 8
```

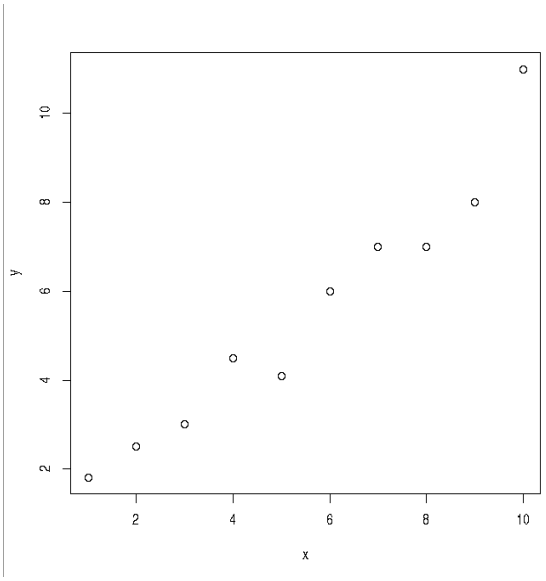
Luego de comprobar que los datos ingresados no tienen errores, procedemos a realizar un gráfico de dispersión (scatterplot)

3.2. Gráficos scatterplot simple

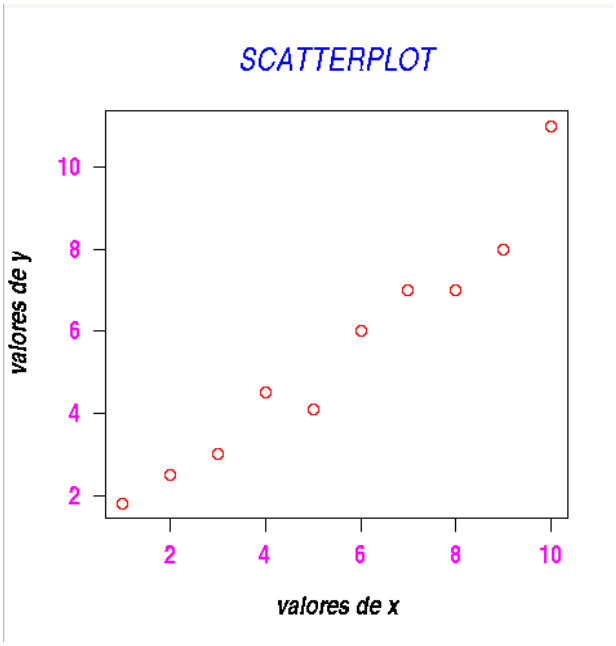
Realicemos el gráfico scatter plot más sencillo

```
> plot(y~x,data=tablaR231)
```

el símbolo ~, se lee "en función". Es decir que como escribimos la línea de comando estamos orientando y al eje vertical y x al eje horizontal.
El código también puede escribirse con los mismos resultados de la siguiente manera
> plot(x,y,data=tablaR231)
indicando primero el eje horizontal y luego el vertical sin el símbolo **~.**



mejoremos y agreguemos información con los argumentos estudiados en la lección anterior. Utilizaremos los argumentos
Relativos al título de la gráfica
main
cex.main
font.main
col.main
Relativos a nombres de los ejes
xlab
ylab
cex.lab
col.lab
font.lab
Relativos a las escalas de los ejes
cex.axis
col.axis
font.axis
las
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red")

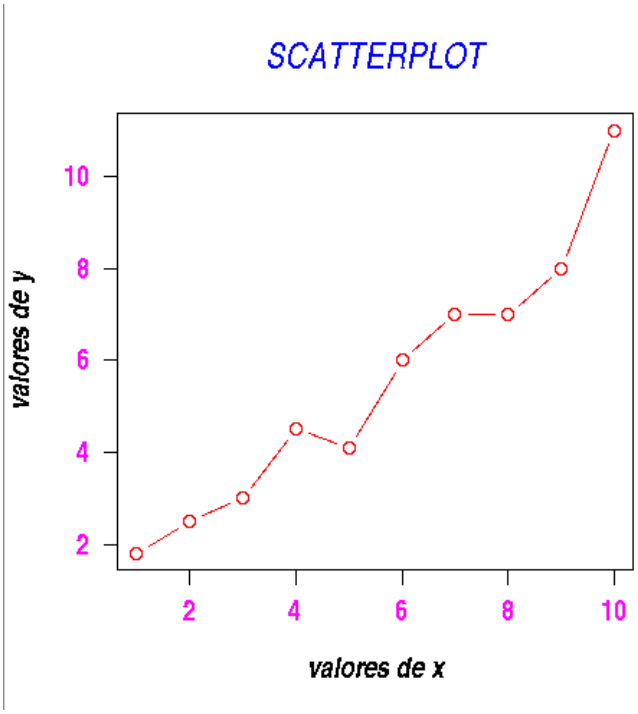


A continuación desarrollamos nuevos argumentos

3.2.1. type

indica la representación de la serie
type=
"p": puntos (default. Si no se especifica type=, plot coloca puntos)
"l": líneas
"b": puntos y líneas
"h": alturas del eje al punto
"s". escalera anticipada
"S": escalera retrasada
"n" : no grafica los valores. Más adelante veremos situaciones en que es útil este valor del argumento.
Pruebe cada opción, para tener idea de sus posibles utilidades. Probaremos el valor "b" que grafica puntos y líneas

```
>plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b")
```



3.2.2. pch

Indica el tipo de símbolo utilizado para marcar los datos. Indique en el código, por ejemplo pch=15

3.2.3. lwd

De la misma manera que en boxplot indica el ancho de la línea de la caja, en este caso, indica el grosor de la línea que une a los puntos. Indique en el código por ejemplo: lwd=1.5 y compare con lwd=1-

3.2.4. cex

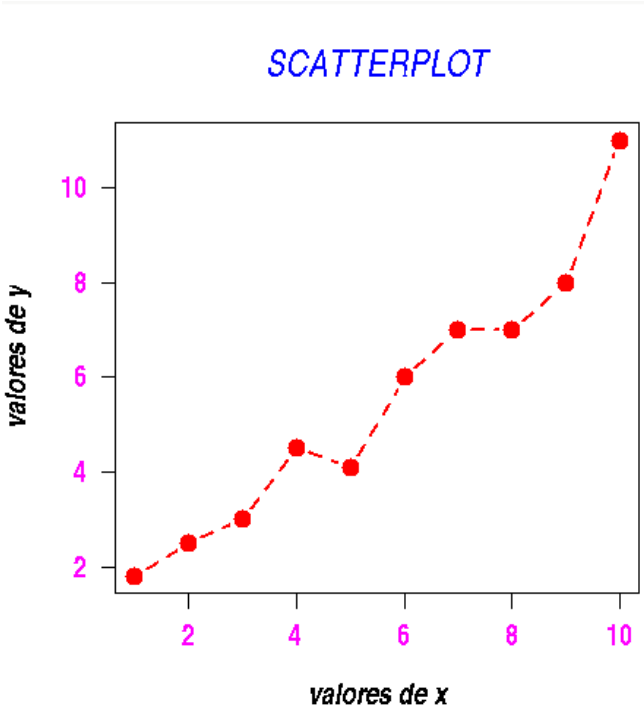
Indica el tamaño de los símbolos que utiliza para indicar los datos en la gráfica. Indique en el código por ejemplo: cex=2 y compare con cex=1.

3.2.5. lty

Indica el tipo de línea que une los puntos. Indique en el código por ejemplo: lty=2. También puede indicarla con palabras, por ejemplo "dotted". Pruebe diferente valores del argumento lty.

A continuación introducimos los cuatro argumentos con sus valores.

```
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2)
```



A continuación vemos otra lista de argumentos y sus valores.

3.2.6. frame

Sus valores posibles son TRUE o FALSE. deja o coloca el recuadro completo al gráfico

3.2.7. fg

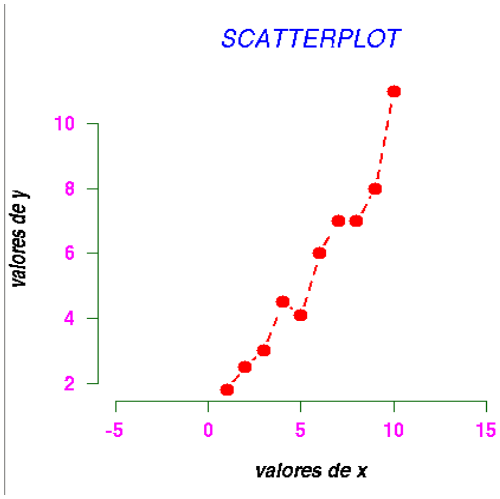
Permite cambiar el color a los ejes

3.2.8. asp

Cambia la relación entre los ejes. 1 da igual proporción horizontal-vertical. Mayor que 1 amplia el rango del eje horizontal. Menor que 1 amplia el rango del eje vertical. Se puede manejar también con xlim e ylim, pero estos últimos mantienen la proporción de los ejes.

Veamos a continuación los tres argumentos descriptos

```
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2,frame=F, fg="darkgreen",asp=2)
```



pruebe los argumentos conocidos

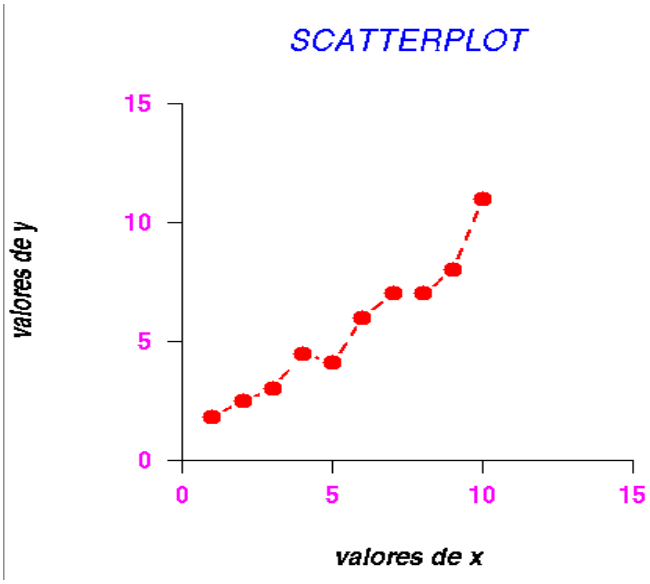
xlim, por ejemplo xlim=c(0,15)

ylim, por ejemplo ylim=c(0,20)

3.2.9. pos

Indica en que valor se cortan los ejes. pos=0, determina que los ejes se corten en el valor 0 de cada eje. Por default los ejes suelen no cortarse en el valor 0

```
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2, frame=F,fg="black",asp=1,pos=0, xlim=c(0,15),ylim=c(0,15))
```



3.2.10. axes

El argumento axes=T o axes=F, permite dejar o sacar las líneas que representan los ejes

3.2.11. axis

Axis es una función que permite agregar ejes. No se coloca dentro de la función plot(), sino que en la función plot() colocamos el argumento axes=FALSE y luego ejecutamos la función axis() indicando los ejes que queremos hacer visibles en la gráfica. La función axis() lleva el argumento labels= T o F, dependiendo que deseemos que se vean divisiones o no de los ejes agregados. Cada eje se identifica con un número arábigo

1: eje x abajo

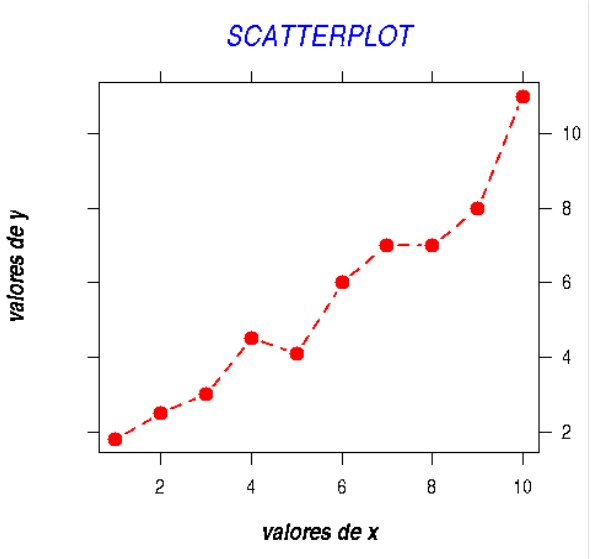
2: eje y a la izquierda

3: eje x arriba

4: eje y a la derecha

En primer lugar hacemos la gráfica con el argumento axes=FALSE. Luego agregamos los ejes con la función axis() indicando en cual deseamos colocar los valores de las divisiones. En el ejemplo siguiente deseamos que el eje vertical tenga las divisiones en le eje de la derecha.

```
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2, frame=T,axes=F)

> axis(1,labels=T)

> axis(3,labels=F)

> axis(2,labels=F)

> axis(4,labels=T,las=1)
```



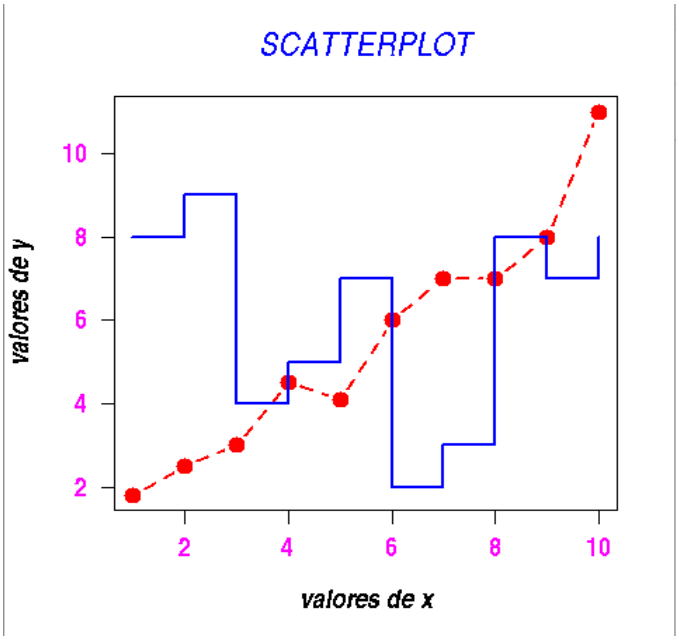
3.3. Agregar serie de datos

Muchas veces tenemos más de una serie de datos que deseamos colocar con características diferentes, por ejemplo símbolos o colores distintos. Para ello utilizaremos luego de la función plot(), la función points()
En primer lugar hacemos la gráfica con una serie de datos, por ejemplo y en función de x

>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2)

Luego agregamos los otros datos (z en función de x) con el formato deseado, en este caso elegiremos la escalera

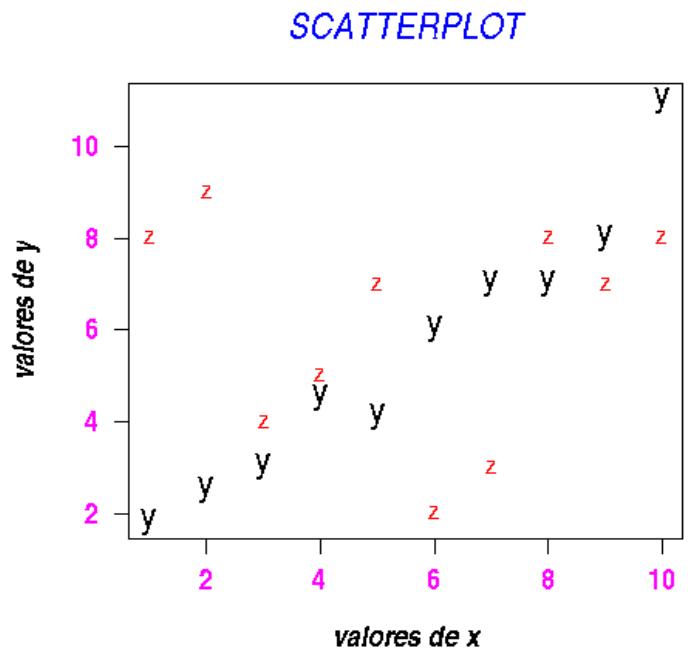
> points(z~x,data=tablaR231,col="blue",type="s",lwd=2)



Un recurso interesante de R es utilizar letras para como símbolos para identificar una serie. En el caso que cada tratamiento se identifica con una letra se puede hacer un gráfico utilizando las letras de cada tratamiento para identificarlas. Para ello en el argumento pch colocamos la letra que deseamos que actúe como símbolo. Utilizamos pch='y' para representar los valores de y en función de x. y en la función points() utilizamos el argumento pch='z' para identificar la serie z de la tablaR231.

```
>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,type="p",pch="y",lwd=2,cex=1.5,lty=2, frame=T)

> points(z~x,data=tablaR231,col="red",pch="z")
```

claramente vemos los datos de la serie "y" y de la serie "z"



3.4. Agregado de leyenda

Ya hemos visto el agregado de leyenda, pero recordemos el mecanismo. Primero realizamos el gráfico con los datos de una serie

>
plot(y~x,data=tablaR231,main="SCATTERPLOT",cex.main=1.5,font.main=3,col.main="blue",xlab="valores de x",ylab="valores de y",cex.lab=1.2,col.lab="black",font.lab=4,cex.axis=1.2,col.axis=6,font.axis=2,las=1,col="red",type="b",pch=19,lwd=2,cex=1.5,lty=2)

luego agregamos los puntos de la segunda serie

>points(z~x,data=tablaR231,col="blue",type="p")

por último agregamos la leyenda

> legend(1,11,legend=c("y","z"),pch=c(19,1),col=c("red","blue"),horiz=TRUE,title="series")

En la lunción leyenda:

Los números 1 y 11, están indicando la posición x e y donde se ubicará el vértice superior izquierdo

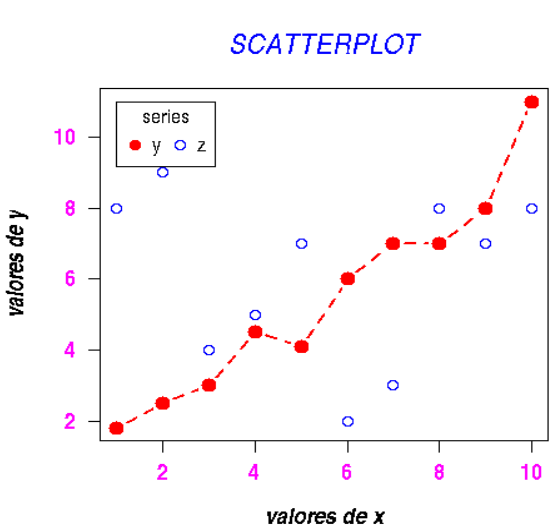
legend=c("y","z"), indica que aparecerá como texto en cada serie

pch=c(19,1), asigna a la serie y el símbolo 19 y a la serie x el símbolo 1

col=c("red","blue"), asigna color rojo y azul a las series "y" y "z" respectivamente

horiz= TRUE: dispone la leyenda de manera horizontal

title, indica que dirá dentro del cuadro de la leyenda

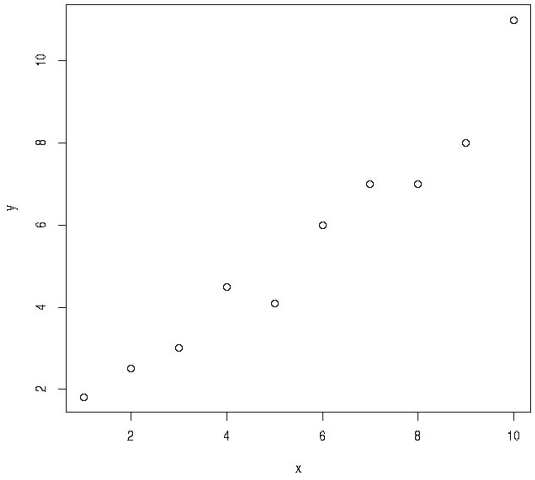


3.5. Colocar linea de ajuste

En muchas circunstancias necesitamos además de los datos, agregar líneas de tendencia o regresión, obtenidas con diferentes modelos. Para este procedimiento deberemos utilizar algunos conceptos previos y otros de la matemática y estadística que serán profundizado en clases y módulos siguientes.

Hagamos una gráfica sencilla. En primer lugar graficamos y en función de x, sin ningún argumento que pueda distraernos

> plot(y~x,data=tablaR231)



vemos una clara relación lineal, que en términos matemáticos indica que

y = a*x + b

donde ***a*** es la pendiente de la recta y ***b*** su ordenada al origen

Para hacer la gráfica (en este caso lineal, pero podría ser cualquier otra), primero debemos hacer el cálculo de los parámetros de la función, en este caso ***a*** y ***b***.

R tiene diversos mecanismo que veremos más adelante, aunque el más útil es aplicar un modelo lineal. Creamos un objeto para el modelo lineal y lo hacemos con el siguiente código. La aplicación de la función lm() es objeto de estudio de próximos módulos

```
> misparametros<-lm(y~x,data=tablaR231)
```

en el objeto misparametros están almacenados con otros datos importantes los parámetros de la recta. Los pedimos con el siguiente código

En primer lugar la ordenada al origen (para R es el coefficient[1])

```
> misparametros$coefficient[1]

(Intercept)

  0.4666667
```

luego la pendiente (para R es el coefficient[2])

```
> misparametros$coefficient[2]

      x

0.9133333
```

intercept= ordenada al origen = 0,4666667

x= pendiente = 0,9133333

nuestro puntos en el eje x van de 1 a 10 (aproximadamente).

Si bien existen diferentes formas de representar una función en una gráfica, mostraremos uno de ellos y el resto será profundizado en módulos posteriores.

Creamos entonces un vector de datos de x para ajustar la recta, que abarque de 1 a 10 y tome valores intermedios cada 0,01). Lo hacemos con la función seq(), vista en módulo 1

```
> xfit<-seq(1,10,0.01)
```

vemos que me creo un vector con datos espaciados en 0,01 unidades y que va de 1 a 10

```
> xfit
  [1] 1.00 1.01 1.02 1.03 1.04 1.05 1.06 1.07 1.08 1.09 1.10 1.11
 [13] 1.12 1.13 1.14 1.15 1.16 1.17 1.18 1.19 1.20 1.21 1.22 1.23
 [25] 1.24 1.25 1.26 1.27 1.28 1.29 1.30 1.31 1.32 1.33 1.34 1.35
 [37] 1.36 1.37 1.38 1.39 1.40 1.41 1.42 1.43 1.44 1.45 1.46 1.47
 [49] 1.48 1.49 1.50 1.51 1.52 1.53 1.54 1.55 1.56 1.57 1.58 1.59
 [61] 1.60 1.61 1.62 1.63 1.64 1.65 1.66 1.67 1.68 1.69 1.70 1.71
 [73] 1.72 1.73 1.74 1.75 1.76 1.77 1.78 1.79 1.80 1.81 1.82 1.83
 [85] 1.84 1.85 1.86 1.87 1.88 1.89 1.90 1.91 1.92 1.93 1.94 1.95
 [97] 1.96 1.97 1.98 1.99 2.00 2.01 2.02 2.03 2.04 2.05 2.06 2.07
.....................................

[829] 9.28 9.29 9.30 9.31 9.32 9.33 9.34 9.35 9.36 9.37 9.38 9.39
[841] 9.40 9.41 9.42 9.43 9.44 9.45 9.46 9.47 9.48 9.49 9.50 9.51
[853] 9.52 9.53 9.54 9.55 9.56 9.57 9.58 9.59 9.60 9.61 9.62 9.63
[865] 9.64 9.65 9.66 9.67 9.68 9.69 9.70 9.71 9.72 9.73 9.74 9.75
```

[877] 9.76 9.77 9.78 9.79 9.80 9.81 9.82 9.83 9.84 9.85 9.86 9.87
[889] 9.88 9.89 9.90 9.91 9.92 9.93 9.94 9.95 9.96 9.97 9.98 9.99
[901] 10.00

creamos ahora otro vector utilizando los datos de xfit y los parámetros de la recta, asignado dichos valores a un vector yfit por medio de la ecuación de una recta

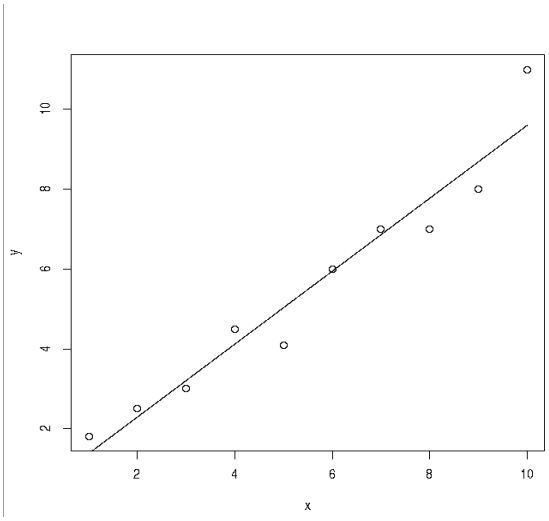
> yfit<-misparametros$coefficients[2]*xfit+misparametros$coefficients[1]

luego hacemos la gráfica, que ya habíamos hecho

plot(y~x,data=tablaR231)

y le colocamos la función de ajuste con el código

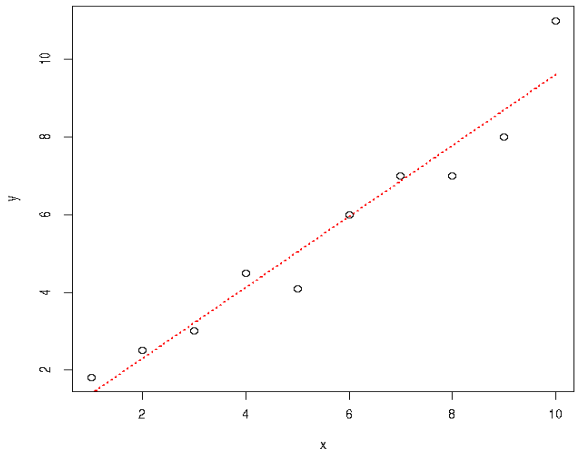
> lines(spline(xfit,yfit))



podemos hacer las mejorías necesarias a la gráfica y la línea

> plot(y~x,data=tablaR231)

> lines(spline(xfit,yfit),lwd=2,lty=3,col=2)



3.6. Gráficos combinados

R tiene gran versatilidad a la hora de realizar gráficos y será prácticamente nuestra imaginación la que podrá poner límites a la creatividad.

Volvamos a la tablaR231

```
> tablaR231
   x  y    z
1  1 1.8   8
2  2 2.5   9
3  3 3.0   4
4  4 4.5   5
5  5 4.1   7
6  6 6.0   2
7  7 7.0   3
8  8 7.0   8
9  9 8.0   7
10 10 11.0 8
```

Supongamos que deseamos colocar en un solo gráfico los datos de z de manera que nos muestre los mismos como una caja con bigotes, de manera de ver mediana, rango intercuartilo y rango de datos. Por otra parte deseamos graficar en color rojo los datos de y en función de x.

Ejecutamos los siguientes comando

```
> boxplot(tablaR231$z,xlim=c(min(tablaR231$x),max(tablaR231$x)),las=1,col="blue")
```

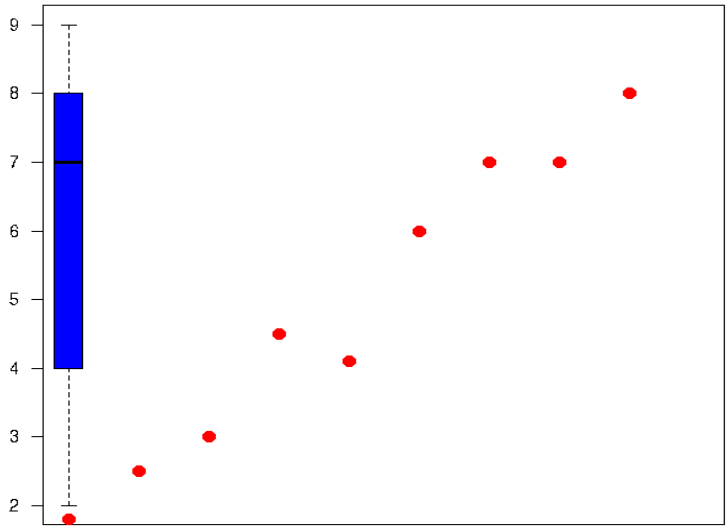
la línea de comandos anterior nos permite hacer el box de los datos de z. Ya en este comando fijamos el rango del eje x para que luego podamos colocar los valores. Para ello utilizamos el argumento xlim, fijando el menor valor como el mínimo de x: min(tablaR231$x) y el máximo como máximo de x: max(tablaR231$x). Ordenamos a nuestro gusto los números de los ejes con el argumento las y le dimos el color azul al box con col.

Ahora colocamos los puntos con la función points()

```
> points(tablaR231$x,tablaR231$y,col="red",pch=20,cex=2)
```

esta función coloca los valores de y en el eje vertical y los de x en el horizonal, con color rojo, utilizando la forma de símbolo 20 fijada por el argumento pch y el tamaño de los mismos con cex.

El gráfico obtenido es



Por supuesto le faltan detalles que se podrían agregar para darle mejor aspecto e información como los rótulos de los ejes, las divisiones del eje horizontal, alguna leyenda, textos, etc.

4. Clase 4

Vídeo: https://youtu.be/Jav1KWxKj5c

Tabla de datos: http://hdl.handle.net/2133/10516

4.1. Gráficos de sectores

En esta clase veremos las gráficas de sectores.

Introduzca los datos de la tabla tablaR241 de la planilla de cálculo tablaR2-4.xls/ods. Esta tabla tiene individuos de ambos sexos (h – m), el número de idiomas que hablan (1, 2, 3, ...) y la edad en años.

```
> tablaR241<-read.table("clipboard",header=TRUE,dec=",",sep="\t",encoding="latin1")

> tablaR241
```

	individuo	sexo	idiomas	edad
1	1	h	2	25
2	2	h	1	25
3	3	h	1	20
4	4	h	2	17
5	5	h	2	45
6	6	h	3	55
7	7	h	3	51
8	8	h	3	20
9	9	h	2	15
10	10	h	2	15
11	11	h	1	18
12	12	h	1	25
13	13	h	1	26
14	14	h	1	29
15	15	h	1	28
16	16	h	2	29
17	17	h	1	23
18	18	h	1	23
19	19	h	1	23
20	20	h	1	26
.....				
37	37	m	3	18
38	38	m	2	25
39	39	m	1	26
40	40	m	2	29
41	41	m	1	28
42	42	m	1	29
43	43	m	1	15
44	44	m	1	15
45	45	m	2	18
46	46	m	1	25
47	47	m	1	26
48	48	m	1	29
49	49	m	1	28
50	50	m	1	29

realicemos una rápida auditoría para corroborar el formato numérico.

```
> summary(tablaR241)
      individuo    sexo     idiomas          edad
 Min.   : 1.00    h:24   Min.   :1.0   Min.   :15.00
 1st Qu.  :13.25  m:26   1st Qu.:1.0   1st Qu.:18.50
 Median  :25.50          Median :1.0   Median :25.00
 Mean    :25.50          Mean   :1.6   Mean   :24.96
 3rd Qu  .:37.75          3rd Qu.:2.0    3rd Qu.:28.75
 Max.    :50.00          Max.   :3.0   Max.   :55.00
```

Como vemos idiomas lo ha tomado como un número real, pero a los fines de análisis es conveniente tomarlo como un factor. Esta decisión se toma habitualmente analizando si la media de la variable tiene sentido. Por ejemplo en este caso la media vale 1.6. ¿Qué significa que en promedio una persona habla 1.6 idiomas?. Realmente carece de sentido, por lo que lo transformamos en factor con la función as.factor(). La variable individuo tiene el mismo sentido. Cada número es asignado como un número de orden a cada uno de ellos. Contrariamente, la edad es correcto mantenerla como un número real.

Entonces transformamos las variables idiomas e individuos en factores

```
> tablaR241$idiomas<-as.factor(tablaR241$idiomas)
```

```
> tablaR241$individuo<-as.factor(tablaR241$individuo)
```

Confirmamos que la transformación se haya realizado

```
> summary(tablaR241)
  individuo  sexo    idiomas       edad
 1      : 1    h:24   1:28     Min.    :15.00
 2      : 1    m:26   2:14     1st Qu .:18.50
 3      : 1            3: 8     Median :25.00
 4      : 1                     Mean    :24.96
 5      : 1                     3rd Qu. :28.75
 6      : 1                      Max.   :55.00
 (Other):44
```

Realizamos la auditoría de los datos controlando como siempre número de filas y columnas, nombre de las columnas, etc.

```
> ncol(tablaR241)
```

```
[1] 4
```

```
> nrow(tablaR241)
```

```
[1] 50
```

El tipo de objeto, que para los data.frame es list.

```
> mode(tablaR241)
```

```
[1] "list"
```

y verificamos si realmente es un data.frame

```
> is.data.frame(tablaR241)
```

```
[1] TRUE
```

Ahora el nombre de cada columna

```
> names(tablaR241)
[1] "individuo" "sexo"      "idiomas"   "edad"
```

o podríamos hacer una visualización de las dos primeras filas de la tabla

```
> head(tablaR241,2)
  individuo sexo idiomas edad
1         1    h       2   25
2         2    h       1   25
```

Contrariamente a lo que puede pensar, el exceso de controles en el momento de ejecutar comando, le harán ahorrar mucho tiempo y esfuerzo.

4.2. Gráficas de sectores simple

Las gráficas de sectores, también conocida como gráfica de tortas, son útiles para realizar descripciones de muestras, especialmente cuando existen datos cualitativos.

Trabajaremos con el data.frame tablaR241. Recordemos el uso de la función table(). Esta función nos permite clasificar los individuos por un criterio, en este caso el sexo.

```
> table(tablaR241$sexo)
 h  m
24 26
```

nos da la cantidad de hombres (h) y mujeres (m) en la tabla

la función prop.table(), nos permite conocer la fracción de cada sexo.

```
> prop.table(table(tablaR241$sexo))
   h    m
0.48 0.52
```

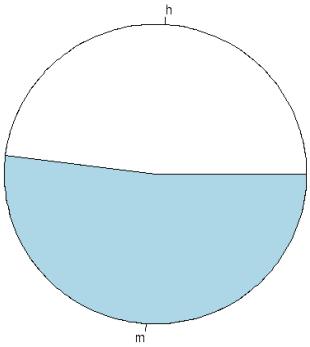
Si a la función la multiplicamos por 100 nos dará el porcentaje de individuos de cada sexo en la muestra en estudio.

```
> prop.table(table(tablaR241$sexo))*100
 h  m
48 52
```

Para realizar un gráfico de sectores o tortas se utiliza la función pie(). Resaltamos en amarillo los cambios sobre los códigos anteriores. La función pie() podemos aplicarla a la función table() y como esta última nos mostraba el número de individuos, la función pie() lo mostrará gráficamente

```
> pie(table(tablaR241$sexo))
```

nos da el gráfico de sectores. Lo mismo que prop.table, pero de manera visual nos indica que hay una mayor proporción de mujeres que hombres



Veamos ahora como mejorar su aspecto.

Argumentos

labels

Permite cambiarle el nombre a cada sector.

edges

El formato circular de la gráfica es aproximado por un polígono cuanto más grande el número más circular el gráfico.

radius

Determina el radio del círculo.

clockwise

Este argumento puede tomar valores TRUE o FALSE. Indica si los sectores se orientan en sentido o en contra del reloj.

init.angle

Permite girar el gráfico. El valor 0 corresponde al inicio del primer sector ubicado a las 3 o'clock.

density

Si este argumento toma el valor NULL si se coloreará cada sector con colores plenos. Si toma un valor numérico indica que se llenará con líneas.

angle

Es un valor que indica la inclinación de la líneas de relleno de cada sector.

col

Da el color de cada sector. Se puede especificar como un vector, con un número para cada sector.

border

Es un vector que le da los colores al borde del sector.

lty

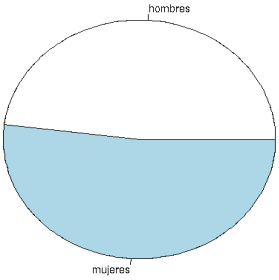
Es un vector que le da la características a las líneas de cada sector. NULL deja la línea continua.

cex

Especifica el tamaño de textos

Iremos viendo cada argumento, resaltando en amarillo el cambio en la línea de comandos. Veamos el argumento labels

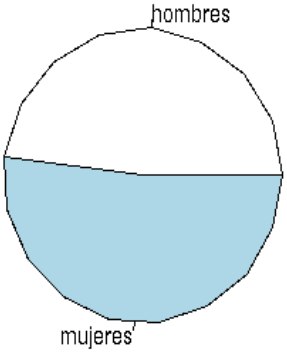
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"))



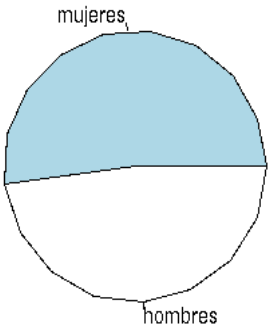
Ahora el argumento edges:

> pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20)

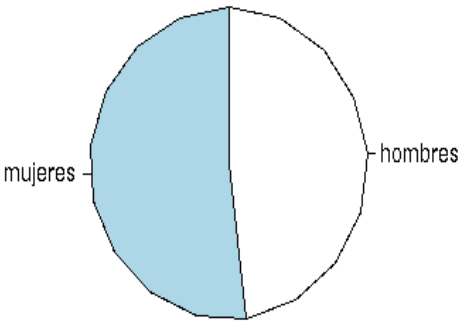
El argumento radius
> pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1)



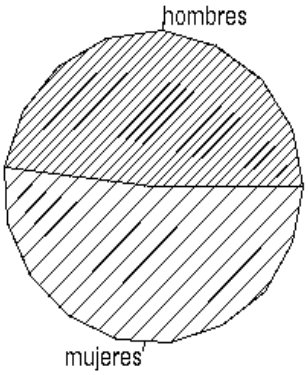
> pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=T, init.angle=0)



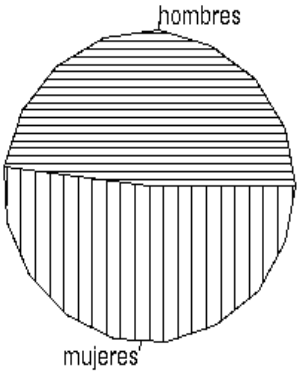

```
> pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=T)
```



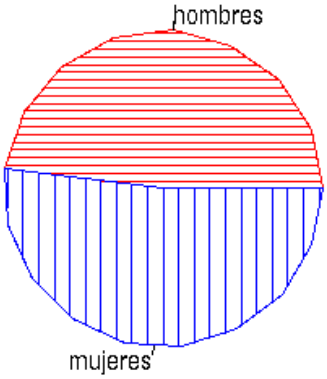

```
>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10))
```



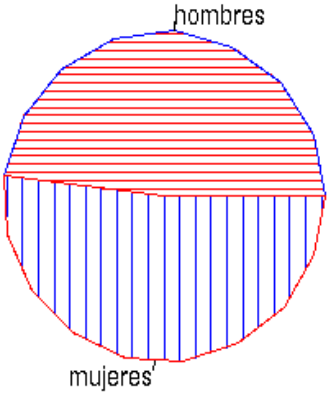

```
>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,90))
```



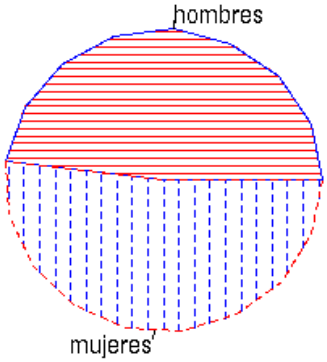
>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,90),col=c("red","blue"))



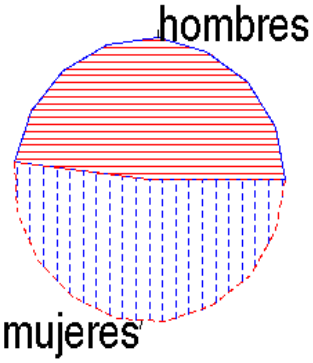
>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,90),col=c("red","blue"),border=c("blue","red"))



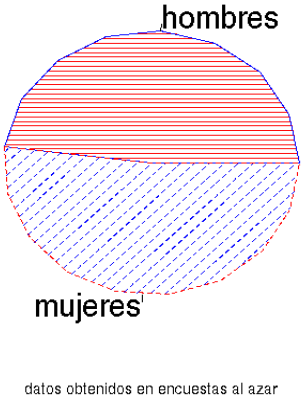
>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,90),col=c("red","blue"),border=c("blue","red"),lty=c(1,2))



>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,90),col=c("red","blue"),border=c("blue","red"),lty=c(1,2),cex=2
)



>
pie(table(tablaR241$sexo),labels=c("hombres","mujeres"),edges=20,radius=1,clockwise=F,init.angle=0,density=c(20,10),angle=c(0,45),col=c("red","blue"),border=c("blue","red"),lty=c(1,2),cex=2
,sub="datos obtenidos en encuestas al azar")



4.3. Dot-chart

Este tipo de gráficas permite mirar desde otro aspecto la misma tabla de datos, haciendo hincapié

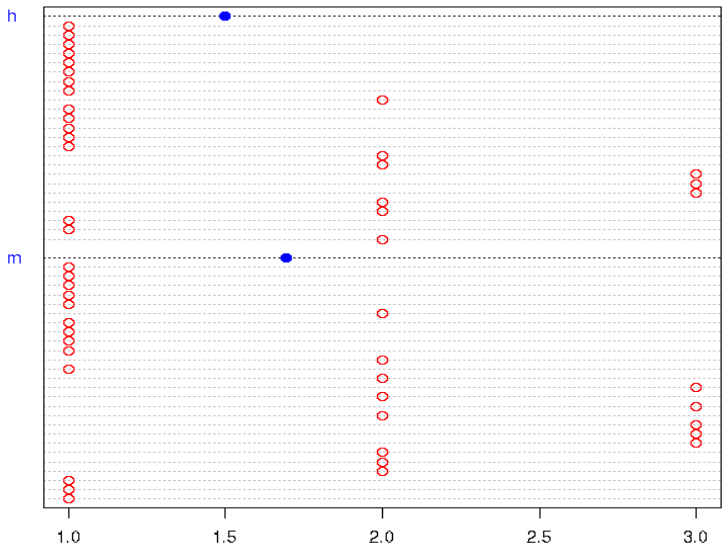
en valores individuales. Cada individuo se identifica con un punto según una categoría y su valor de variable analizada. Además se puede mostrar el valor de la media u otra estadística como la mediana. En las gráficas siguientes se eligió punto rojo para cada valor individual y azul para la estadística mencionada.

Para poder realizar este tipo de tabla, las variables tienen que ser numéricas. Anteriormente para el gráfico de sectores transformamos idioma a variable categórica. Ahora la necesitaríamos nuevamente en formato numérico, cosa que hacemos con el siguiente código.

```
> tablaR241$idiomas<-as.numeric(tablaR241$idiomas)
```

Ahora si hacemos la gráfica

```
>
dotchart(tablaR241$idioma,groups=tablaR241$sexo,gdata=tapply(tablaR241$idioma,tablaR241$sexo,mean),pch=1,gpch=19,col="red",gcol="blue")
```



Los datos a graficar son el número de idiomas que se representa con tablaR241$idioma.

groups: indica como se dividirán los datos, en este caso por sexo. Se representan en el panel de arriba los hombres y abajo las mujeres.

gdata es el valor de la estadística a graficar. Se puede hacer con la función tapply como se muestra en el código.

col= da el color a los datos

gcol= da el color a la estadística que se agrega a la gráfica.

pch: tipo de punto para los datos

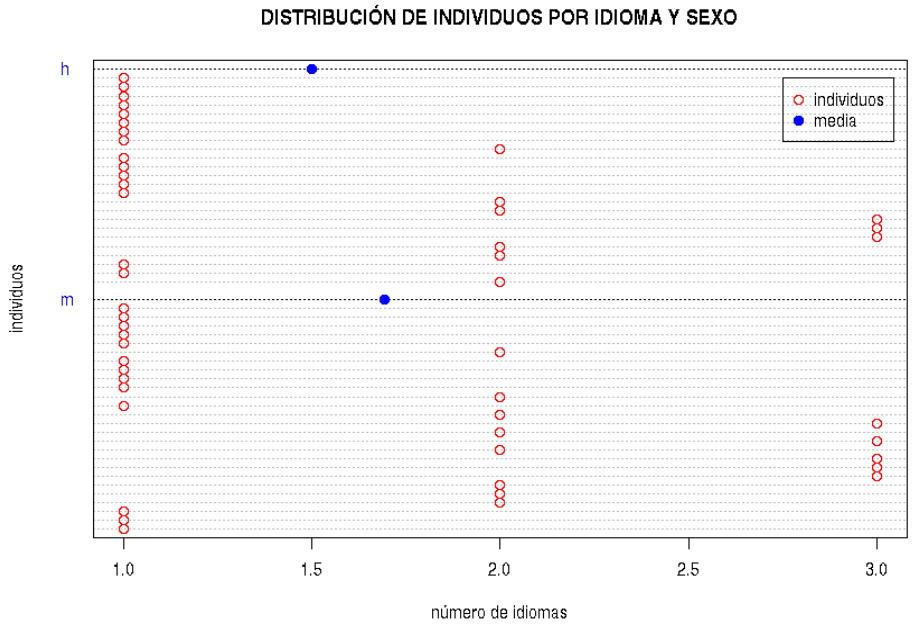
gpch: tipo de punto para la estadística agregada.

Agregaremos ahora algunos detalles de ejes, título y leyenda

```
>
```

dotchart(tablaR241$idioma,groups=tablaR241$sexo,gdata=tapply(tablaR241$idioma,tablaR241$sexo,mean),pch=1,gpch=19,col="red",gcol="blue",ylab="individuos",xlab="número de idiomas",main="DISTRIBUCIÓN DE INDIVIDUOS POR IDIOMA Y SEXO")

> legend(2.75,52,legend=c("individuos","media"),pch=c(1,19),col=c("red","blue"))

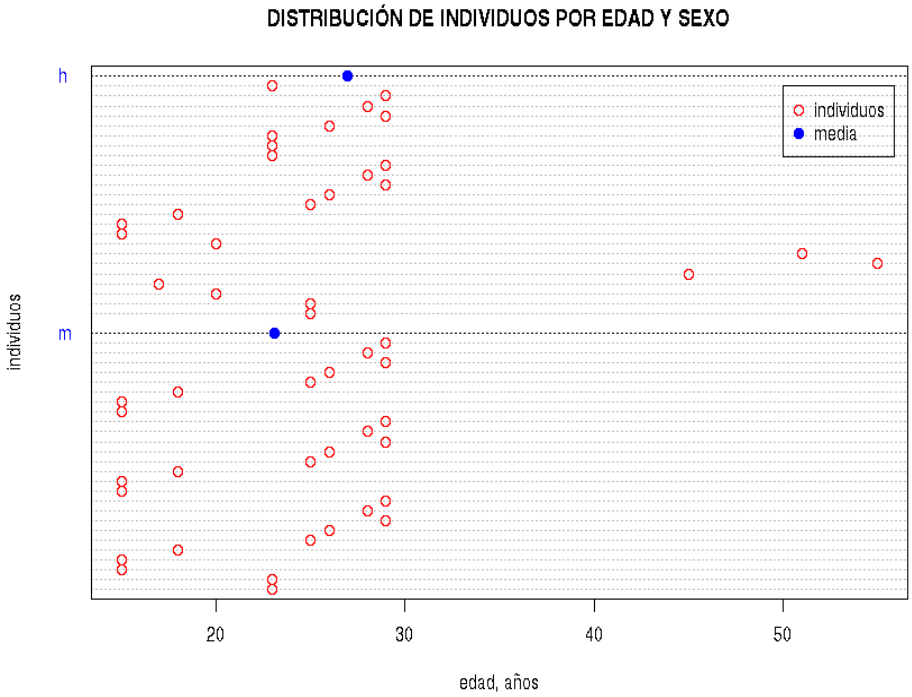


La gráfica claramente nos muestra que en promedio los hombres hablan menos número de idiomas que las mujeres (ver ubicación del punto azul que representa la media). Se ve claramente que en ambos sexos hay una mayor proporción de personas que hablan 1 idioma, menos 2 y aun menos tres idiomas.

Cambiamos a continuación el tipo de gráfico, en el eje x colocamos la edad y agrupamos los datos por sexo.

>
dotchart(tablaR241$edad,groups=tablaR241$sexo,gdata=tapply(tablaR241$edad,tablaR241$sexo,mean),pch=1,gpch=19,col="red",gcol="blue",ylab="individuos",xlab="edad,
años",main="DISTRIBUCIÓN DE INDIVIDUOS POR EDAD Y SEXO")

> legend(50,52,legend=c("individuos","media"),pch=c(1,19),col=c("red","blue"))

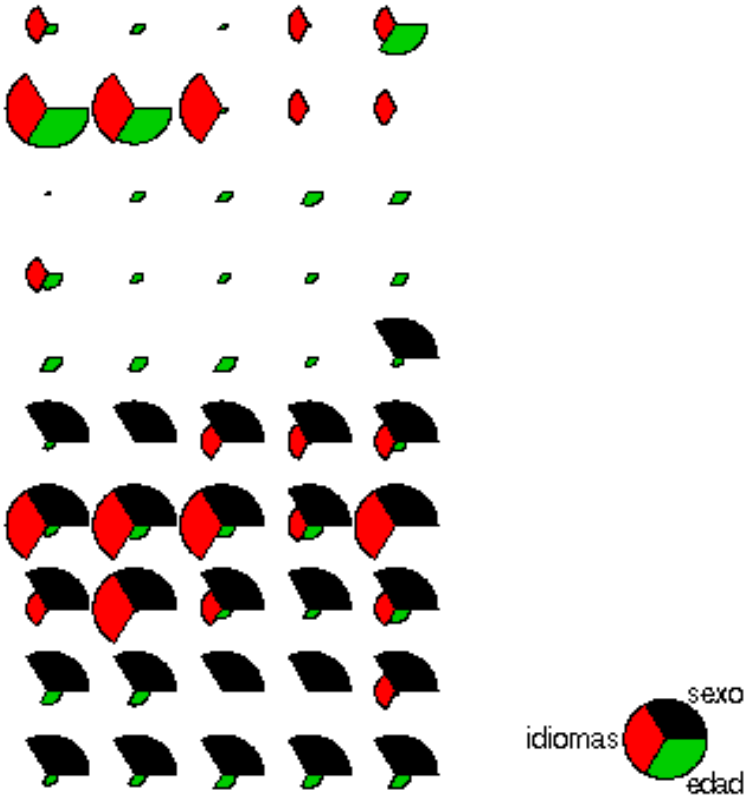


4.4. Star plot

Este tipo de gráfico permite observar también de manera gráfica e individual a las variables de cada unidad experimental. Es muy útil para observar individualmente a cada unidad experimental. El color y tamaño de cada sector indica la magnitud del valor. El radio máximo del sector corresponde al mayor valor de la variable estudiada y la ausencia de sector al menor valor. En este tipo de gráfico se pueden observar todas las variables, tanto cuali como cuantitativas. Utilizamos para este tipo de gráfica la función stars().

stars(tablaR241[2:4],full=T,locations=NULL,len=1,nrow=10,ncol=5,key.loc=c(17,3),key.labels=colnames(tablaR241)[2:4],draw.segments=T)

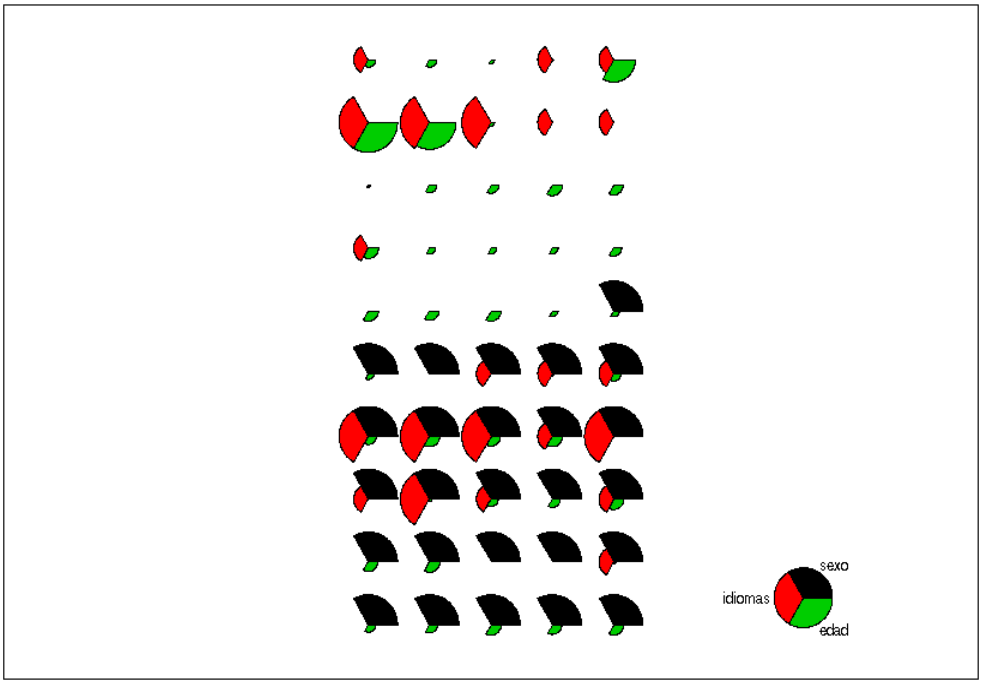
En la gráfica obtenida, se observan 50 combinaciones de sectores. Cada uno de ellos representa un individuo de la tabla.



A la derecha aparece la leyenda del gráfico, dividida en tres sectores (las variables): sexo, idiomas y edad. Para cada individuo se grafican las tres variables con la siguiente característica. Si la variable tiene el valor máximo dentro de todos los individuos, el sector tendrá el radio máximo. En cambio si tiene el valor mínimo, el sector no se visualiza. En el caso del sexo que tiene dos niveles, el sector se da en negro. Para hombre no se visualiza el sector, para mujer toma el radio máximo. Para idioma, que se visualiza en rojo, tiene tres niveles: 1, 2, y 3. Aquellos individuos que hablan tres idiomas tienen el sector rojo con máximo radio, mientras que en los que hablan 1, no se visualiza el sector. La misma interpretación es para edad. El individuo de mayor edad tendrá el radio del sector verde en su valor máximo y el más joven no tendrá sector verde.

Le hacemos algunas mejorías al gráfico

```
>
stars(tablaR241[2:4],full=T,locations=NULL,len=1,nrow=10,ncol=5,key.loc=c(17,3),key.labels=colnames(tablaR241)[2:4],draw.segments=T,col.segments=c(1,2,3),main="DISTRIBUCIÓN POR SEXO,IDIOMA Y EDAD",frame=T)
```



Interpretemos:

sector sexo: color negro= mujer. Ausencia de color: hombre

sector idiomas: máximo radio: 3 idiomas, ausencia de sector: 1 idioma

sector edad: máximo radio: individuo de mayor edad. Ausencia de sector: individuo de menor edad.

los primeros 24 individuos son hombres: no tienen sector negro.

los 26 siguientes son mujeres: tienen sector negro.

Hagamos ahora algunas comprobaciones y para ello busquemos en la tabla los individuos de menor edad. Realizaremos esta búsqueda con un código ya estudiado

```
> tablaR241[tablaR241$edad==min(tablaR241$edad),]
   individuo sexo idiomas edad
9          9    h       2   15
10        10    h       2   15
27        27    m       1   15
28        28    m       2   15
35        35    m       3   15
36        36    m       2   15
43        43    m       1   15
44        44    m       1   15
```

Si buscamos en la figura el individuo 9 veremos que solo tiene sector rojo: es decir habla dos idiomas, es hombre de 15 años. Verifique con los otros individuos.

Cuando realizamos los gráficos de tipo stars, es conveniente ordenar todas las columnas, ya sea

creciente o decreciente.

Revisemos un summary

```
> summary(tablaR241)
  individuo       sexo     idiomas          edad
 Min.   : 1.00   h:24   Min.   :1.0   29     :10
 1st Qu.:13.25   m:26   1st Qu.:1.0   15     : 8
 Median :25.50          Median :1.0   23     : 6
 Mean   :25.50          Mean   :1.6    25    : 6
 3rd Qu.:37.75          3rd Qu.:2.0   26     : 5
 Max.   :50.00          Max.   :3.0   28     : 5
                                      (Other):10
```

Ordenaremos la tabla primero por sexo, luego por número de idiomas y finalmente por edad

Podemos ordenar la tabla y asignarla a otro data.frame y luego graficar o bien colocar el código de ordenamiento dentro del propio gráfico.

El código para realizar el ordenamiento sería

```
tablaR241[order(tablaR241[,2],tablaR241[,3],tablaR241[,4]),]
```

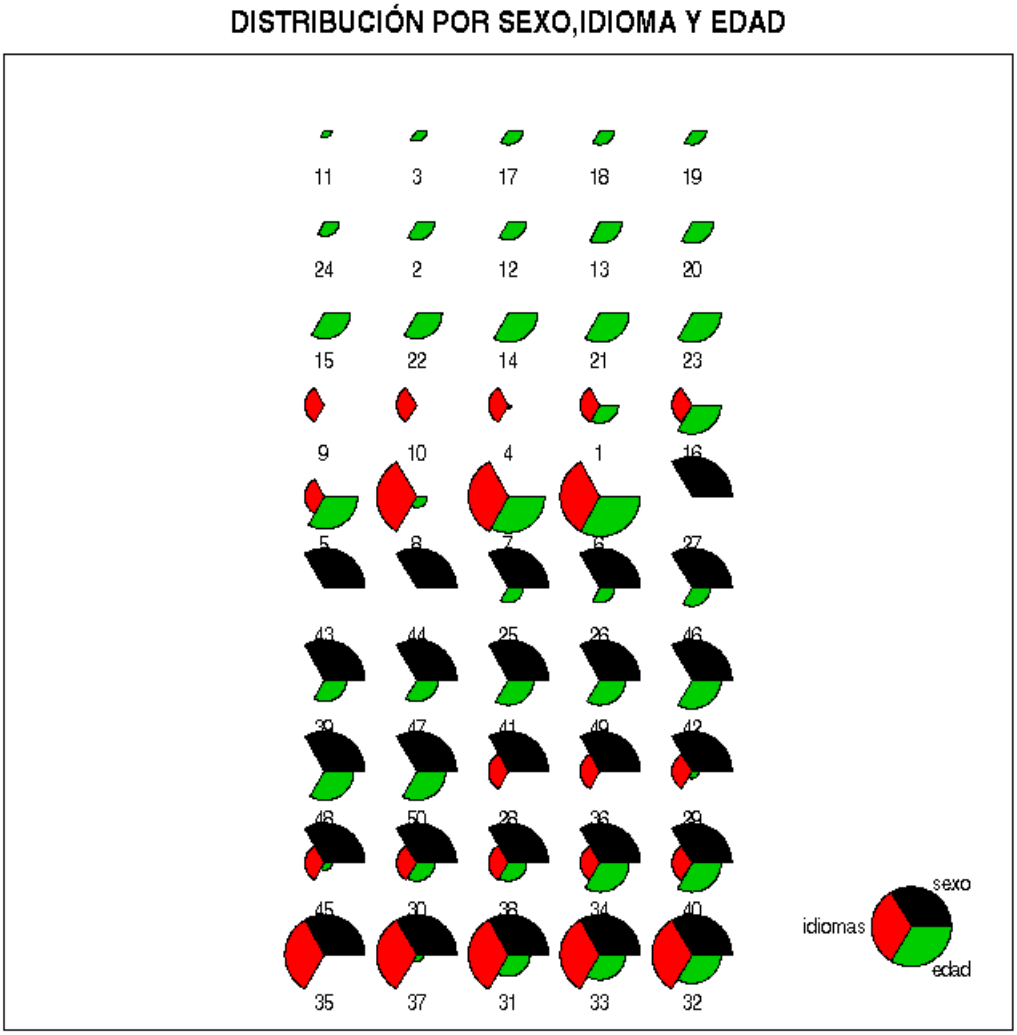
Revise este concepto en el módulo 1 de este curso.

Podemos colocar el código dentro del código para hacer el gráfico. Veamos el código que utilizamos inicialmente sin ningún tipo de orden

```
>
stars(tablaR241[2:4],full=T,locations=NULL,len=1,nrow=10,ncol=5,key.loc=c(17,3),key.labels=colnames(tablaR241)[2:4],draw.segments=T,col.segments=c(1,2,3),main="DISTRIBUCIÓN POR SEXO,IDIOMA Y EDAD",frame=T)
```

entonces hacemos el reemplazo marcado en amarillo

```
stars(tablaR241[order(tablaR241[,2],tablaR241[,3],tablaR241[,4]),]
[2:4],full=T,locations=NULL,len=1,nrow=10,ncol=5,key.loc=c(17,3),key.labels=colnames(tablaR241)[2:4],draw.segments=T,col.segments=c(1,2,3),main="DISTRIBUCIÓN POR SEXO,IDIOMA Y EDAD",frame=T)
```



Todas las variables están ordenadas en forma creciente. El sexo (sector negro) tiene primero "h" y luego "m". A "h" no lo muestra y "m" lo muestra con el radio máximo. Se distingue claramente en el gráfico los hombres en la parte superior y las mujeres en la parte inferior.
El segundo ordenamiento es el número de idiomas (sector rojo). Los 15 primeros hombres hablan 1 idioma, luego hay 6 que hablan 2 y solo 3 que hablan 3 idiomas. Con respecto a la edad, vemos que hay individuos de casi todas las edades. El de mayor edad de la tabla es el individuo numero 6, es hombre y además habla 3 idiomas.

4.5. Gráficos múltiples

4.5.1. Con función layout

Para hacer gráficos múltiples básicamente lo que hacemos es crear una matriz con celdas donde colocaremos cada gráfico. Para ello se utiliza la función layout().

La sucesión siguiente de comandos conducen a la gráfica que se halla al final de los mismos. El mecanismo es igual si se quisieran realizar gráficos múltiples con boxplot o scatter-plots, inclusive

se pueden combinar diferentes tipos de gráficos.

> layout(matrix(1:3,1,3),widths=c(10,10,10),heights=c(5,5,5))

explicación del argumento (1:3,1,3). 1:3,1,3: se colocarán tres imágenes. 1:3,1,3: en una sola fila, 1:3,1,3: en tres columnas.

Luego se ejecutan los gráficos, que irán colocándose desde el primero hasta el último espacio

> pie(table(tablaR241$edad),radius=1)

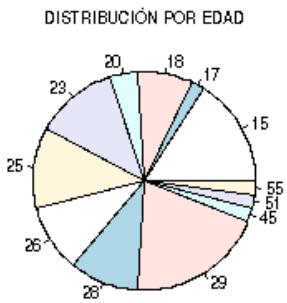
> text(0,1.5,"DISTRIBUCIÓN POR EDAD")

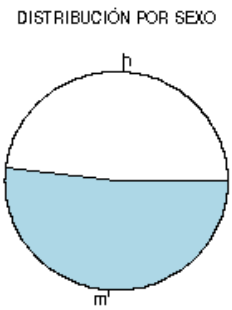
> pie(table(tablaR241$sexo),radius=1)

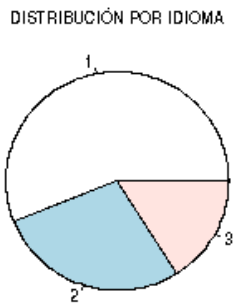
> text(0,1.5,"DISTRIBUCIÓN POR SEXO")

> pie(table(tablaR241$idioma),radius=1)

> text(0,1.5,"DISTRIBUCIÓN POR IDIOMA")







Los comandos siguientes dan numéricamente la misma información. En el primer caso aplicado a la edad tenemos para edad el porcentaje de individuos.

> prop.table(table(tablaR241$edad))*100
15 17 18 20 23 25 26 28 29 45 51 55
16 2 8 4 12 12 10 10 20 2 2 2

Con la siguiente vemos los porcentajes de hombres y mujeres, representados gráficamente por los tamaños de los sectores.

> prop.table(table(tablaR241$sexo))*100
h m
48 52

Con el siguiente el porcentaje de individuos que habla 1, 2 o 3 idiomas.

> prop.table(table(tablaR241$idiomas))*100
1 2 3
56 28 16

4.5.2. Con modificación de parámetros

Habitualmente R está configurado para hacer un solo gráfico por vez. Podemos modificar algunos parámetros que permitan hacer más de uno. A tal fin podemos utilizar el parámetro mfrow.

El valor por defecto de este parámetro es

mfrow=c(1,1),

es decir, define una grilla de una fila y una columna, dejando un espacio para el gráfico. Cuando este se llena se cierra la grilla y se abre nuevamente para un nuevo gráfico. Por ello lo persiste el anterior.

Si modificamos el parámetro mfrow, podemos hacer varios gráficos a la vez. Veremos este tema en la clase en que introducimos el manejo de parámetros, adelante en este curso.

5. Clase 5

Vídeo: https://youtu.be/m2cebKoOezE

Tabla de datos: http://hdl.handle.net/2133/10517

5.1. Gráficos de distribución de probabilidad

Recordemos que muchas de las funciones y argumentos son comunes a todos los tipos de gráficos

En esta clase veremos la construcción de gráficas que puedan contribuir a visualizar la distribución de probabilidad de los datos que disponemos.

En primer lugar introduzcamos en nuestro espacio de trabajo los datos que se hallan en la hoja tablaR251 de la planilla de cálculo tablaR2-5.xls/ods

```
> tablaR251<-read.table("clipboard",header=TRUE,dec=",",sep="\t",encoding="latin1")
```

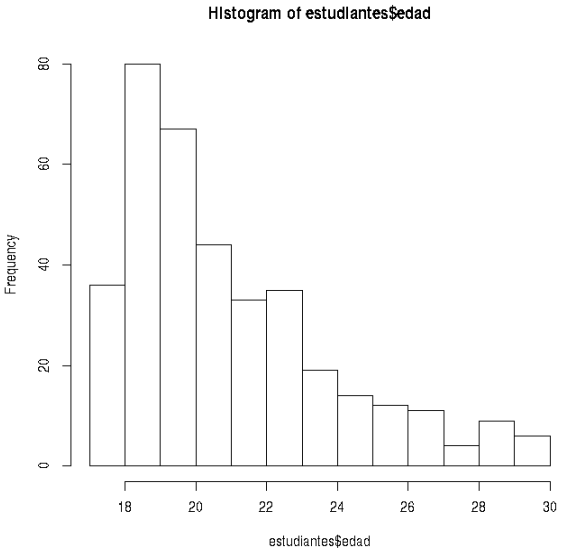
y realicemos la auditoría de datos

```
> summary(tablaR251)
      edad            sexo         peso                altura           ocupacion        actividadfisica
 Min.   :17.00   f:239    Min.   : 41.00    Min.   :1.500   estudiante:370   n:112
 1st Qu.:19.00   m:131    1st Qu.: 56.00    1st Qu.:1.620                    s:258
 Median :21.00            Median : 64.00    Median :1.680
 Mean   :21.42            Mean   : 66.93    Mean   :1.689
 3rd Qu.:23.00            3rd Qu.: 74.00    3rd Qu.:1.750
 Max.   :30.00            Max.   :139.00    Max.   :1.940
> nrow(tablaR251)

[1] 370

> ncol(tablaR251)

[1] 6

> is.data.frame(tablaR251)

[1] TRUE

> str(tablaR251)
'data.frame':        370 obs. of  6 variables:
 $ edad          : int  26 19 20 25 23 20 29 24 23 23 ...
 $ sexo          : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1 1 1 2 ...
 $ peso          : int  66 85 55 51 100 62 55 59 68 107 ...
 $ altura        : num  1.74 1.76 1.72 1.59 1.84 1.67 1.83 1.65 1.73 1.63 ...
 $ ocupacion     : Factor w/ 1 level "estudiante": 1 1 1 1 1 1 1 1 1 1 ...
 $ actividadfisica: Factor w/ 2 levels "n","s": 2 2 2 2 2 2 1 2 2 1 ...
```

5.1.1. Histogramas

Un histograma permite observar la frecuencia absoluta o relativa de una variable cuantitativa agrupada por intervalos. Hallemos el histograma para la variable edad.

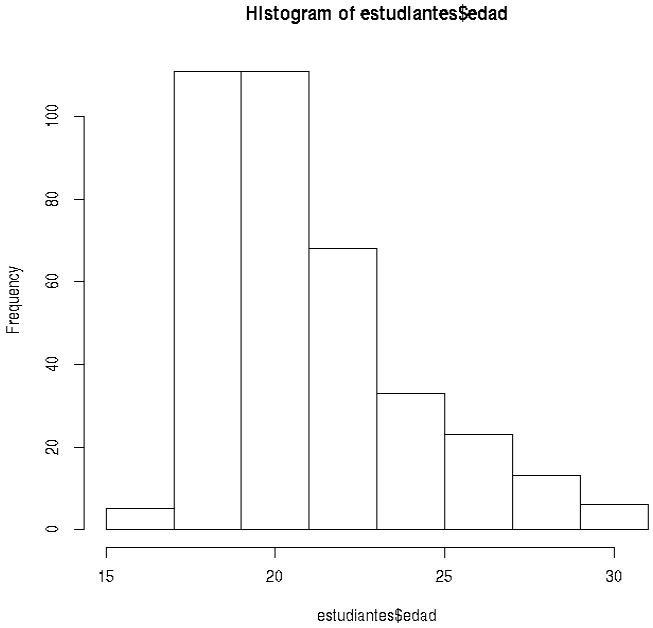
```
> hist(tablaR251$edad)
```



la gráfica nos está indicando el número de estudiantes para cada intervalo de edades. De observar el histograma vemos que para edades entre 18 y 19 años tenemos aproximadamente 80 estudiantes.

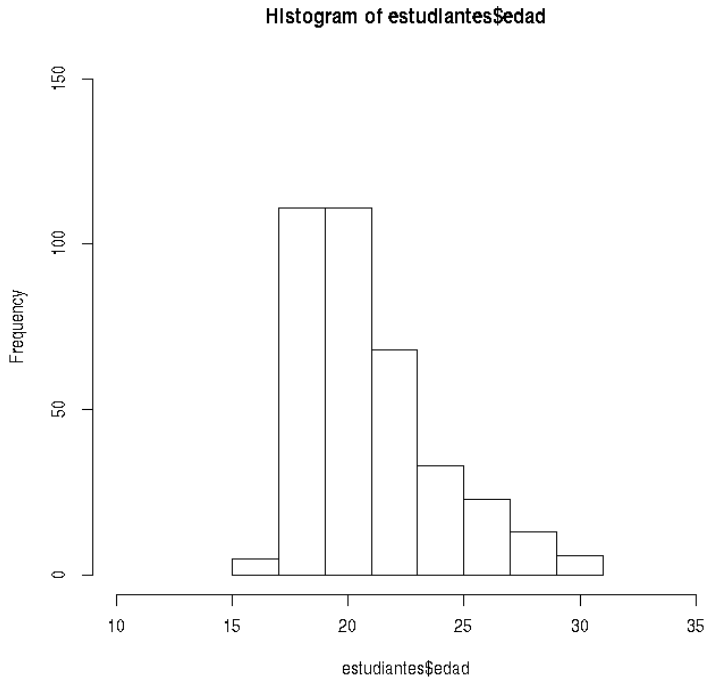
Podemos establecer los intervalos para conocer la frecuencia utilizando el argumento breaks.

```
> hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31))
```



Con los argumentos xlim e ylim podemos mejorar la presentación

> hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,150))



Un problema habitual en los histogramas se presenta con valores como por ejemplo un estudiante que tenga 25 años. A un individuo con esta edad lo incluimos en el intervalo 23-25 o en el intervalo 25-27 años? Estos problemas se solucionan con los argumentos include.lowest y right.

>
hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,150),include.lowest=T,right=F)

para entender estos argumentos tomemos por ejemplo el valor 21, que coincide con uno de los breaks o límites de cada intervalo. Include.lowest=T indica que el en el intervalo que va de 21 a 32, el 21 será incluido en este intervalo. Sin embargo, al ser right=F, el extremo derecho del intervalo, es decir el valor 23 no será incluido en este intervalo sino en el siguiente. Es importante recordar que hay solo dos configuraciones posibles para ambos argumentos

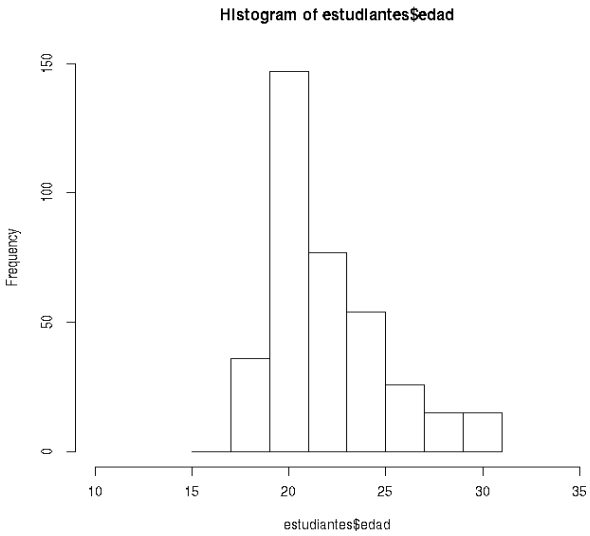
include.lowest= TRUE, right =FALSE

o

include.lowest=FALSE, right=TRUE

cualquier otra combinación dará un gráfico erróneo, ya que un valor como el 21 que pertenece a un valor de breaks, será incluido en el intervalo anterior y posterior.

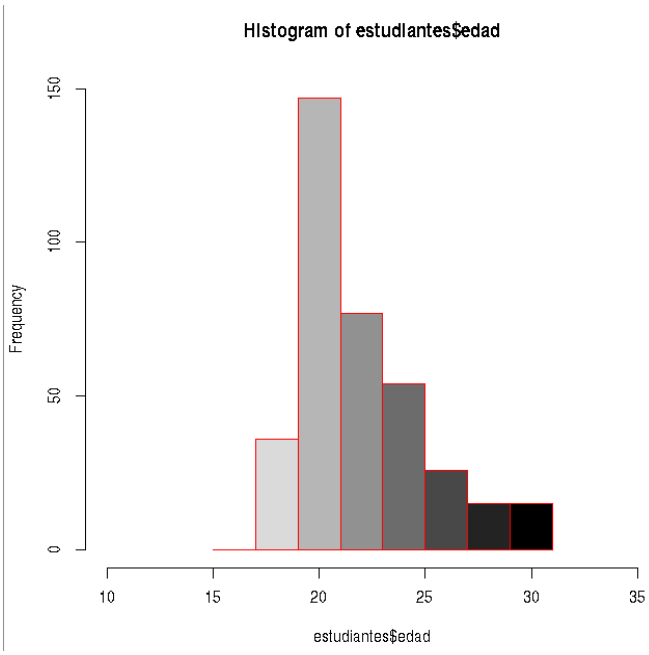
Recuerde que al modificar los argumentos include.lowest y right, puede haber cambios en su histograma, ya que ciertos individuos pueden pasar de uno a otro intervalo



Veamos ahora algunos detalles visuale como color y bordes de las barras. Para ello utilizaremos argumentos ya conocidos como col y border.

>
hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,150),include.lowest=T,right=F,col=gray(seq(1,0,length=8)),border="red")

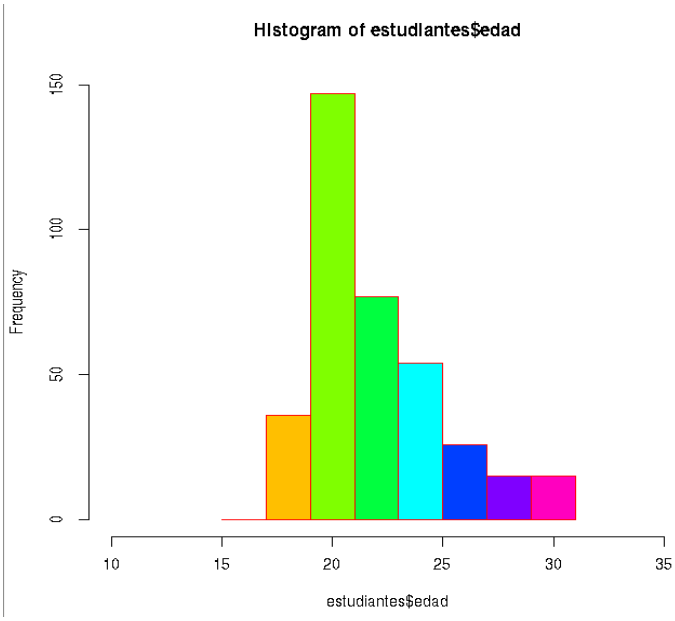
elegimos en este caso colorear los escalones (que son 8 los dibujados) con una escala de grises de 0 a 1 y envolverlos en rojo



La función rainbow() es útil para lograr un espectro de colores atractivos. En esta función que da colores varios, solo debe indicarse cuantos colores se desean, como lo muestra el ejemplo

>
hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,150),include.lowest=T,right=F,col=rainbow(8),border="red")

rainbow (8) da 8 colores diferentes. Podrían ser más. Esta función se puede usar como argumento en otras gráficas.



El argumento plot, permite ver el gráfico si su valor es TRUE o no verlo si es FALSE. En tal caso nos mostrará detalles analíticos del histograma

>
hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,150),include.lowest=T,right=F,col=rainbow(8),border="red",plot=F)

no da la gráfica y da un histograma analítico y numérico.

$breaks

[1] 15 17 19 21 23 25 27 29 31

son los números fijados al hacer el histograma, en base a los cuales se agruparan los individuos por sus edades.

$counts

[1] 0 36 147 77 54 26 15 15

cantidad de individuos en cada intervalo

$density

[1] 0.00000000 0.04864865 0.19864865 0.10405405 0.07297297 0.03513514 0.02027027

[8] 0.02027027

la frecuencia de individuos por unidad de la variable analizada. Por ejemplo el valor 0,04864865 = 36 / (370*2). Es decir el el número de individuos en un intervalo dividido el número de individuos totales y la amplitud del intervalo. En otras palabras el 4,86% de los individuos tienen entre 17 y 18 años.

$mids

[1] 16 18 20 22 24 26 28 30

los puntos medios de cada intervalo

$xname

[1] "tablaR251$edad"

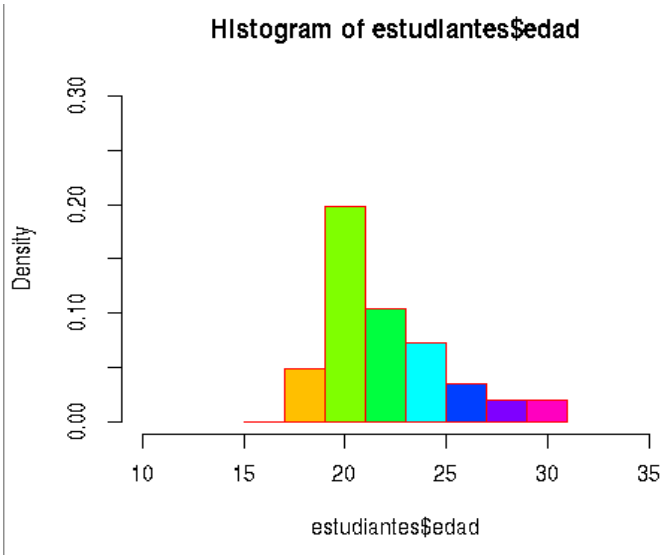
data.frame y columna analizada

$equidist

[1] TRUE

El histograma que estuvimos graficando nos mostraba la frecuencia, es decir la cantidad de individuos por cada intervalo. El argumento freq, permitirá mostrarnos la cantidad o la densidad.

>
hist(tablaR251$edad,breaks=c(15,17,19,21,23,25,27,29,31),xlim=c(10,35),ylim=c(0,0.3),include.lowest=T,right=F,col=rainbow(8),border="red",plot=T,freq=F)



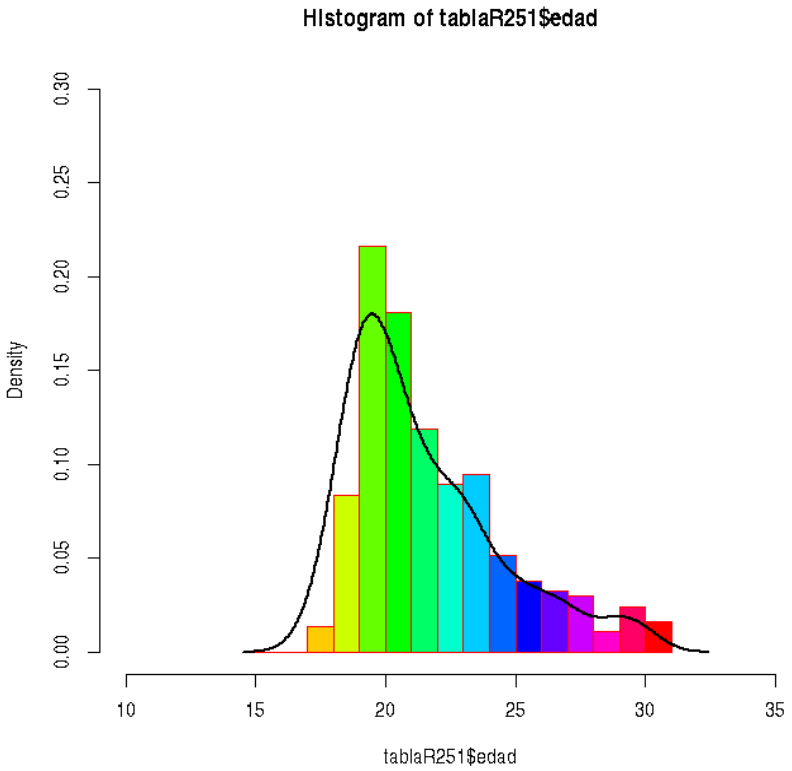
cambiemos los breaks de manera que estratifiquemos cada un año

>
hist(tablaR251$edad,breaks=c(15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31),xlim=c(10,35),ylim=c(0,0.3),include.lowest=T,right=F,col=rainbow(15),border="red",plot=T,freq=F)

podemos introducir la línea que nos ajusta los valores de densidad y dará un aspecto interesante al histograma

> lines(density(tablaR251$edad),lwd=2)

agrega la función de ajuste al histograma



El histograma nos muestra una distribución de probabilidad, es decir que porcentaje de la población tiene valores mayores, menores o están en un intervalo respecto del total de datos. Por ejemplo en el histograma anterior aproximadamente podemos decir que el 50% de los individuos tienen edad menor a 21 años. Para ello sume la densidad del grupo 17-18 años (aprox=0,01), del grupo 18-19; 0,08, del grupo 19-20: 0,22 y del grupo 20-21 años (aprox 0,18), esto da 0,49, es decir un 49% tiene edad menor a 21 años.

Por supuesto esto datos que sacamos de mirar el histograma también los podemos obtener cambiando el argumento y observando los valores de $density para los $breaks requeridos

>
hist(tablaR251$edad,breaks=c(15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31),xlim=c(10,35),ylim=c(0,0.3),include.lowest=T,right=F,col=rainbow(8),border="red",plot=F,freq=F)

$breaks

[1] 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

$counts

[1] 0 0 5 31 80 67 44 33 35 19 14 12 11 4 9 6

$density

[1] 0.00000000 0.00000000 0.01351351 0.08378378 0.21621622 0.18108108

[7] 0.11891892 0.08918919 0.09459459 0.05135135 0.03783784 0.03243243

[13] 0.02972973 0.01081081 0.02432432 0.01621622

Podemos resolver esto analítica y elegantemente con códigos y recursos ya conocidos. En primer lugar asignamos el código anterior al un objeto que

histtabla251<-
hist(tablaR251$edad,breaks=c(15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31),xlim=c(10,35),ylim=c(0,0.3),include.lowest=T,right=F,col=rainbow(8),border="red",plot=F,freq=F)

para ver cada uno de los detalles del objeto, por ejemplo density, utilizamos el nombre del objeto y separado por $ el atributo

> qhisttabla251$density

[1] 0.00000000 0.00000000 0.01351351 0.08378378 0.21621622 0.18108108

[7] 0.11891892 0.08918919 0.09459459 0.05135135 0.03783784 0.03243243

[13] 0.02972973 0.01081081 0.02432432 0.01621622

histtabla251$density es un vector. Lo podemos presentir por la forma en que se nos muestra y lo podemos verificar con

> is.vector(histtabla251$density)

[1] TRUE

Entonces si es un vector conocer la suma de los primeros 6 intervalos es sencillo con códigos aprendidos en el módulo 1 de este curso.

> sum(histtabla251$density[c(1:6)])

[1] 0.4945946

comprobemos esto con otro recursos conocidos.

Primero busquemos en nuestra tabla cuantos individuos tienen menos de 21 años, para ello utilizamos el código

> menores21<-tablaR251[tablaR251$edad<21,]

este código nos seleccionó los individuos menores de 21. Tener en cuenta que en nuestro histograma el argumento right=F. Que indica que los individuos de 21 años fueron incluidos en el intervalo 21-22.

Con el código siguiente podemos calcular cuantos individuos tienen menos de 21 años

> nrow(menores21)

[1] 183

y con el siguiente el porcentaje

> nrow(menores21)*100/nrow(tablaR251)

[1] 49.45946

Ya debe ir comprendiendo lo redundante que es R para alcanzar un resultado.

Aclaración: El valor $density que se obtiene al aplicar la función hist() con el argumento plot=F, da la probabilidad acumulada hasta cierto valor de la variable analizada. Sin embargo esto es cierto por la forma en que está definida si el break tiene un intervalo de 1

En el caso que el intervalo de break no sea uno, se puede corregir fácilmente creando breaks cada una unidad. Cosa sencilla si hablamos del peso, lo que corregiríamos con el siguiente código que incluye un generador de números secuenciales aprendidos en módulo 1. Como el peso mínimo es 41 y el máximo 139, podemos escribir

> sum(hist(tablaR251$peso,breaks=seq(41,139,1),plot=F)$density)

[1] 1

y como se puede comprobar ahora da 1. Pues bien y que pasa con la altura cuyo mínimo es 1.5 y el máximo 1.94? Si tomamos un break de 1, en ese intervalo caen todos los valores. Engañemos a R, trabajando con valores multiplicados por 100 de manera que el mínimo sea 150 y el máximo 194 Asi quedaría

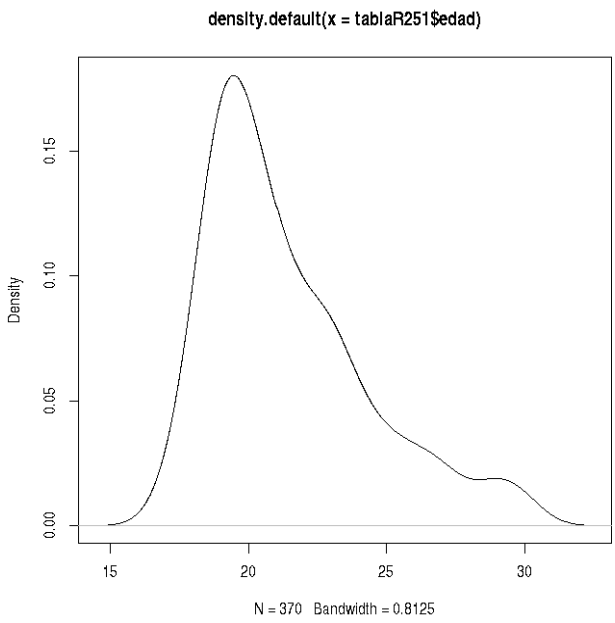
> sum(hist((tablaR251$altura)*100,breaks=seq(150,194,1),plot=F)$density)

[1] 1

5.1.2. Density

Permite ajustar una distribución de probabilidad empírica. Como se vio en el ítem anterior, puede ajustarse también sobre un histograma

> plot(density(tablaR251$edad))



5.1.3. steam and leaf,

Este tipo de representación (mitad gráfica y mitad numérica) permite obtener una distribución de los valores de una variable numérica entre los individuos de una muestra.

Analizaremos la edad. El argumento scale es clave en esta representación. En este caso toma el valor 1 que nos permite ver la distribución por años. Se eligió 1 como valor, ya que los años son números enteros y el rango solo abarca de 17 a 30. Se debe jugar con este argumento según la muestra.

```
> stem(tablaR251$edad,scale=1)
 The decimal point is at the |
 17 | 00000
  18 | 000000000000000000000000000000
  19 | 000000000000000000000000000000000000000000000000000000000000000000000000
  20 | 000000000000000000000000000000000000000000000000000000000000000000000000
  21 | 000000000000000000000000000000000000000000
  22 | 00000000000000000000000000000000
  23 | 000000000000000000000000000000000
  24 | 0000000000000000000
  25 | 00000000000000
  26 | 000000000000
  27 | 00000000000
  28 | 0000
  29 | 000000000
  30 | 000000
```

¿Cómo se interpreta?. Para 17 años, luego del valor observamos el signo | y luego observamos 5 ceros. Esto indica que en la muestra hay 5 individuos de 17 años. Información que podemos comprobar para tomar confianza en la representación con códigos de selección ya conocidos

```
> tablaR251[tablaR251$edad==17,]
     edad sexo peso  altura ocupacion  actividadfisica
106  17   f    58  1.61   estudiante          n
174  17   f    49  1.60   estudiante          s
237  17   f    48  1.63   estudiante          n
251  17   f    57  1.66   estudiante          s
294  17   f    45  1.70   estudiante          n
```

entendiendo el argumento scale. Vemos el siguiente ejemplo, pero aplicado a la altura

```
> stem(tablaR251$altura,scale=0.5)
 The decimal point is 1 digit(s) to the left of the |
  15 | 00222223333344444
  15 | 55555556666666777777777788888889999999
  16 | 000000000000000000000011111112222222222233333333333333333444444444
  16 | 555555555555555555555555566666666777777777777777777778888888888888
  17 | 000000000000000000000000000001112222222233333333333333333333344444
  17 | 555555555555555566666666666677777888899999
  18 | 000000000000000000000000112223333344444
  18 | 55555566699999
  19 | 01144
```

¿Cómo se interpreta?

veamos 15|

15 | 00222223333344444

Los números a la derecha del símbolo | indican el número que acompaña a 15

Así nos indica que hay

2 individuos que miden 1,50 (ya que hay dos ceros a las derecha de |)

5 que miden 1,52 (hay cinco números 2 a la derecha de |)

5 que miden 1,53

5 que miden 1,54

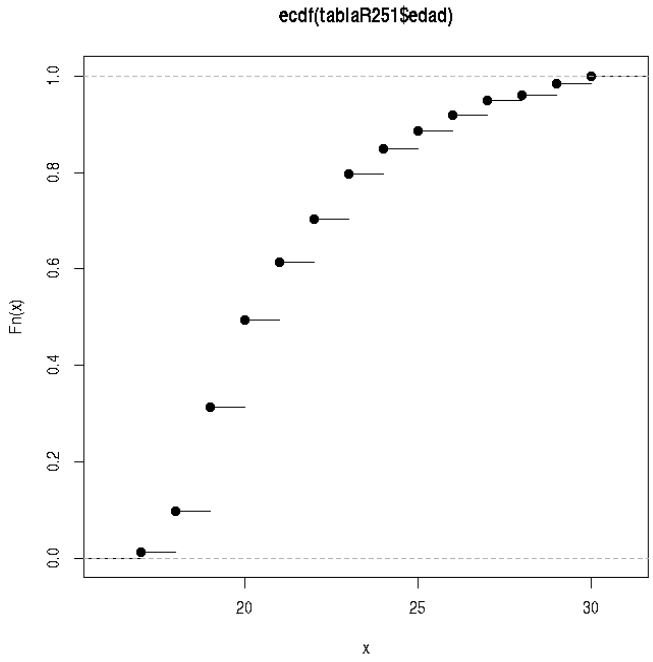
5.1.4. ecdf

ecdf: empirical cummulative distribution

Esta función hace la distribución acumulada de los datos en función del valor de la variable. Luego se puede dibujar sobre ella la línea que correspondería a una distribución normal acumulada tomando la media y el desvío estándar de los datos. Se evalúa a simple vista por la desviación de los puntos respecto de la línea. Si no coinciden los puntos con la recta, la muestra no tiene o se aleja de la distribución normal. Es un recurso más para evaluar si una muestra en particular tiene distribución normal o no.

tomamos la variable edad del data.frame tablaR251, con el que venimos trabajando

> plot(ecdf(tablaR251$edad))



Calculamos el rango de edades

> range(tablaR251$edad)

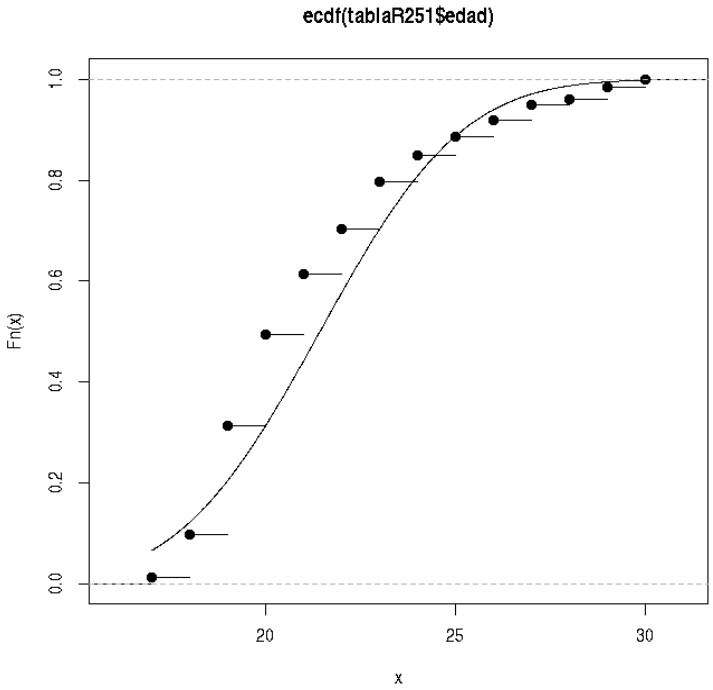
[1] 17 30

generamos un vector con edades que van de 17 a 30 años de a 0,01 año

> x<-seq(17,30,0.01)

graficamos sobre el gráfico anterior una línea que tendrá como x la edad y como valor de la ordenada los valores acumulados de una distribución normal con la media y desvío estándar de nuestros datos.

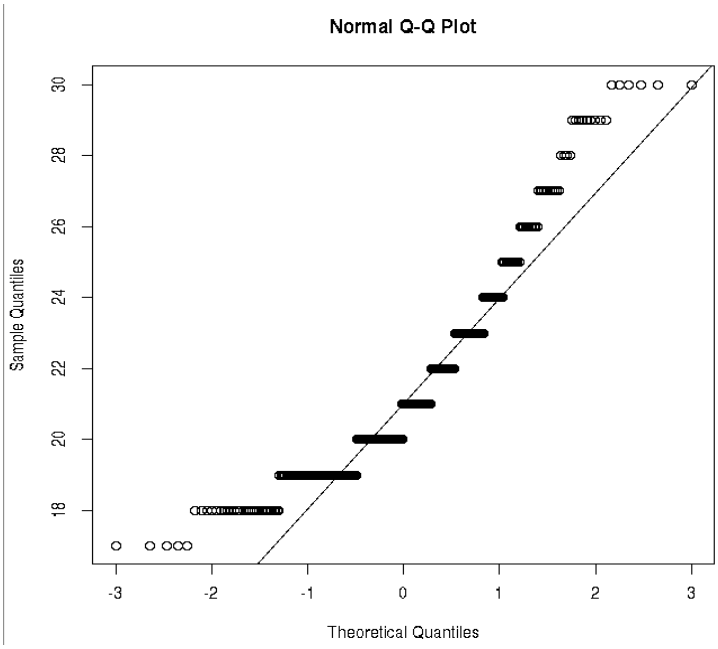
> lines(x,pnorm(x,mean=mean(tablaR251$edad),sd=sd(tablaR251$edad)))



Cuanto más al azar se hallan distribuidos los puntos respecto de la línea, más cercana es la distribución de nuestros datos a una distribución normal. En nuestro caso vemos que en un tramo entre 20 y 25 años todos los puntos están de un solo lado. Completaremos este análisis visual de distribución de probabilidad con formas analíticas que veremos en el módulo 3 de este curso.

5.1.5. Q-Q plots

Quantile – quantile plot. Permite también gráficamente ver si una distribución es o no normal
Aplicamos a nuestra tabla las funciones qqnorm() y qqline()
> qqnorm(tablaR251$edad)
> qqline(tablaR251$edad)



Si la distribución es normal los puntos deberían superponerse a la recta. En este caso se ven desviaciones de la distribución normal.

Veamos como darían las gráfica ecfd y QQplot en el caso que los datos realmente se distribuyeran normalmente.

Generemos una muestra de 100 valores de una muestra proveniente de una población normal con media 2 y sd=0.3. Revisar clases anteriores como se realiza este procedimiento. Para esto utilizamos la función rnorm(n,mean,sd) que generará n=100 datos aleatorios, con media= 2 y sd igual a 0.3

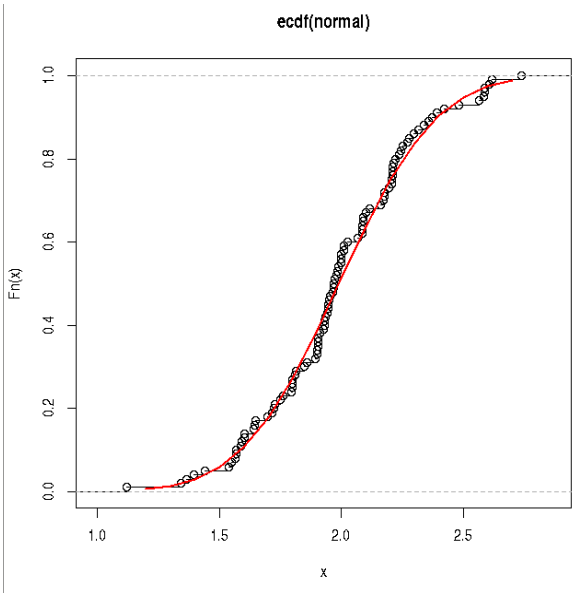
```
> normal<-rnorm(100,2,0.3)
```

Veamos los datos generados y guardados en el objeto normal. Como los datos generados por rnorm() son aleatorios, sus datos no coincidirán con los que se muestran a continuación, pero tendrán distribución normal.

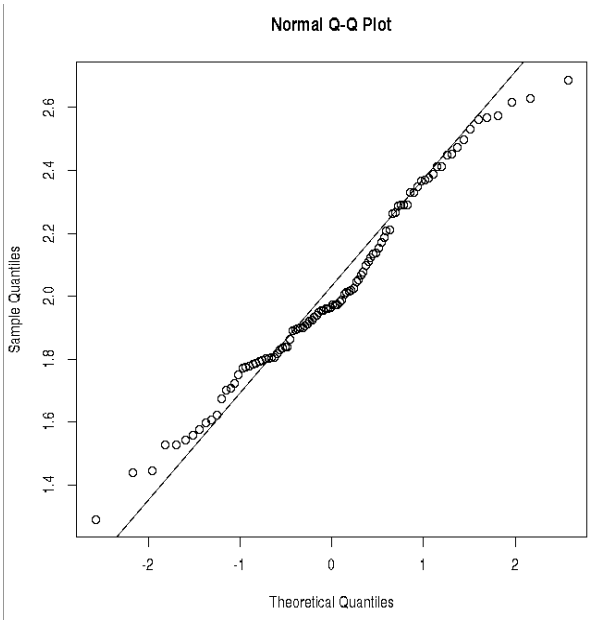
```
> normal
  [1] 1.721397 1.804619 2.617405 2.019167 1.784730 1.621783 1.598194 2.376650
  [9] 1.974220 1.576881 2.110506 2.388382 2.331047 1.890877 2.473789 1.528753
 [17] 2.330752 1.934641 2.015431 1.557166 1.829804 2.367172 1.901244 1.912632
 [25] 2.413275 1.900921 1.960582 1.940386 2.264486 2.153098 2.567109 1.980787
 [33] 2.560776 1.707086 1.772444 1.953415 2.045183 1.674991 1.801577 2.209507
 [41] 1.837591 2.531662 2.497439 2.291387 1.862449 1.607118 2.349014 1.948021
 [49] 1.443889 1.792283 2.688028 1.541746 2.451641 2.628958 1.701574 2.133967
 [57] 1.898145 1.835054 2.291552 1.786078 2.207707 2.288987 2.371135 1.816751
 [65] 2.051901 2.136439 2.098097 1.893219 1.806032 1.920726 1.802481 1.796406
 [73] 1.776152 1.962971 1.289196 2.024831 1.986819 2.068249 2.012244 2.450465
 [81] 2.263026 2.187623 2.287422 2.005468 1.972453 1.953783 2.170149 1.526338
 [89] 1.971638 1.838702 1.438058 1.751444 2.573515 2.411313 1.904761 1.960689
 [97] 2.075837 1.778564 2.121524 1.924378
```

veamos primero la curva ecdf

```
> range(normal)
[1] 1.289196 2.688028
> x<-seq(1.2,2.7,0.1)
> plot(ecdf(normal),pch=1)
> lines(x,pnorm(x,mean=mean(normal),sd=sd(normal)),lwd=1.5,col='red')
```




```
> qqnorm(normal)
> qqline(normal)
```



6. Clase 6

Vídeo: https://youtu.be/u7WZmQhWQa0

Tabla de datos: http://hdl.handle.net/2133/10518

6.1. Bar plots

Los gráficos de barras habitualmente se utilizan para mostrar medias y desvíos estándar de dos o más grupos de datos. El límite superior de cada barra coincide con el valor de la media y la longitud del segmento habitualmente en "T" representa la magnitud del desvío estándar. Por ejemplo en la Figura 6.1 se muestran las medias de tres grupos: control, t1 y t2. La media de t1 es aproximadamente 2 g/l y su desvío estándar de aproximadamente 0,4 g/l. Además se suele marcar con asterisco, letras o cualquier otro símbolo si existe o no diferencia entre los grupos. Por supuesto que en este tipo de gráficos se puede marcar también si se lo desea mediana, rango, percentilos, etc, pero no es lo más común. De todas maneras en el pie del gráfico se debe indicar qué representa cada elemento del gráfico, el número de unidades analizadas en cada grupo, y la estadística aplicada. Recuerde que en general la gráfica debe ser autoexplicativa.

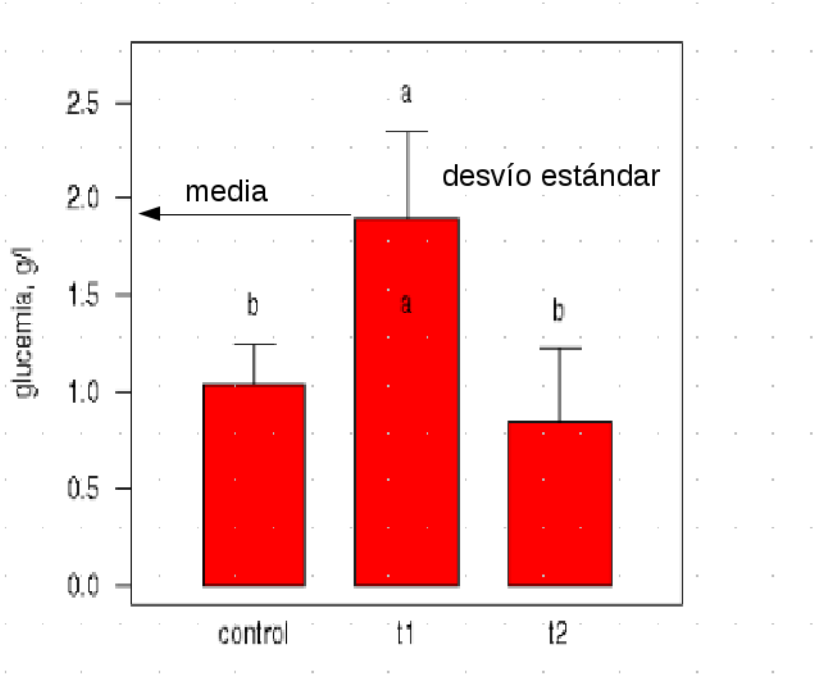


Figura 6.1: Glucemia en g/l de los grupos experimentales. Las barras representan las medias de cada grupo y los segmentos los desvío estándar. Las letras sobre los desvío indican el significado del análisis estadístico: letras distintas entre dos grupos indican diferencias significativas entre los grupos, p<0,05 ANOVA, post-test LSD, n=8 por grupo.

6.2. Uso de funciones prearmadas y bibliotecas

6.2.1. barplot()

Esta función es la más sencilla y a la vez limitada, pertenece al paquete graphics que se instala automáticamente con R. Veamos como funciona.

Introduzcamos la tablaR262 en nuestro espacio de trabajo

```
> tablaR262<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

```
> tablaR262
   tratamiento glucemia
1      control     1.10
2      control     0.80
3      control     0.90
4      control     0.85
5      control     0.84
6      control     0.97
7      control     1.20
8      control     1.20
9      control     1.10
10     control     1.00
11     control     1.50
12          t1     1.10
13          t1     1.50
14          t1     1.60
15          t1     1.70
16          t1     1.80
17          t1     2.00
18          t1     2.10
19          t1     2.00
20          t1     2.00
21          t1     2.30
22          t1     2.80
23          t2     0.70
24          t2     0.80
25          t2     0.80
26          t2     0.70
27          t2     0.90
28          t2     0.50
29          t2     0.60
30          t2     0.40
31          t2     1.50
32          t2     1.60
33          t2     0.80
```

Haga las auditorías correspondientes

```
> summary(tablaR262)
  tratamiento    glucemia
 control:11   Min.   :0.400
 t1      :11   1st Qu.:0.800
 t2      :11   Median :1.100
```

```
Mean   :1.262
3rd Qu.:1.600
 Max.  :2.800
```

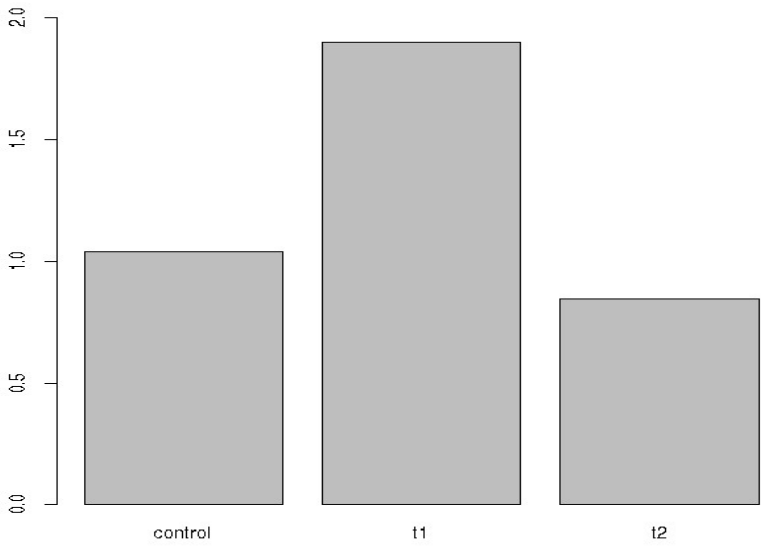
queremos graficar la media de los valores de glucemia para cada tratamiento

Recordemos que con la función tapply() podemos conocer las medias por grupos

```
> tapply(tablaR262$glucemia,tablaR262$tratamiento,mean)
  control        t1        t2
1.0418182 1.9000000 0.8454545
```

grafiquemos estos datos en formato de barras utilizando la función barplot(), solo fijando el argumento del largo del eje vertical con ylim.

```
barplot(tapply(tablaR262$glucemia,tablaR262$tratamiento,mean),ylim=c(0,2))
```



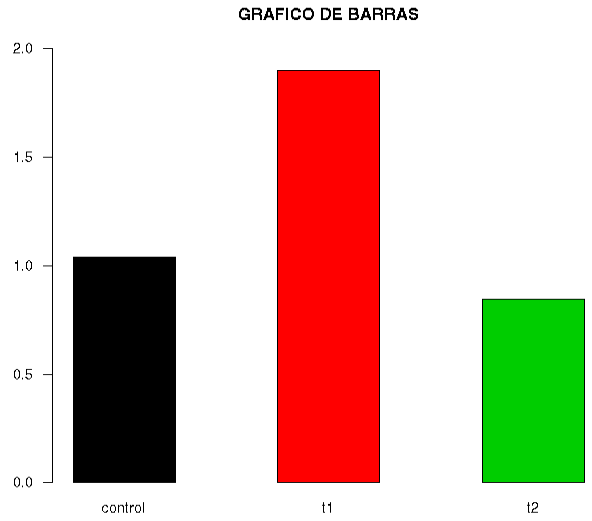
veamos los argumentos que podemos utilizar para mejorar las gráficas

col: permite cambiar el color de las barras

space: permite cambiar el grosor de las barras

border: permite asignar un color al borde del recuadro. En este caso colocaremos un mismo color a todos

```
> barplot(tapply(tablaR262$glucemia,tablaR262$tratamiento,mean),ylim=c(0,2),col=c(1,2,3),
space=1,border="black",main="GRAFICO DE BARRAS",las=1)
```



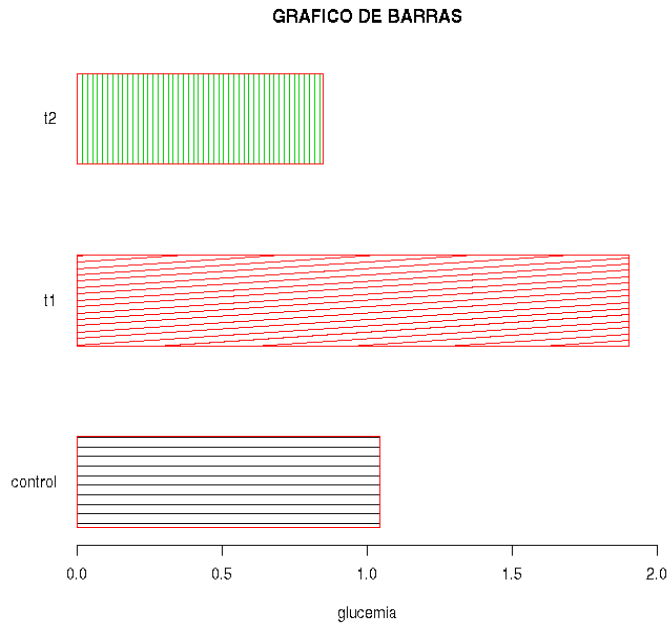
Otros argumentos

density: permite cambiar el patrón de relleno de las barras, por líneas

angle: nos permite ajustar el ángulo de inclinación de las líneas de relleno

horiz: permite cambiar la orientación de las barras. Cuando usa este argumento debe cambiar ylim por xlim, ya que la variable dependiente se graficará en el eje horizontal.

barplot(tapply(tablaR262$glucemia,tablaR262$tratamiento,mean),col=c(1,2,3),border="red",density=c(10,15,20), angle=c(0,4,90),horiz=T,xlim=c(0,2),space=1,main="GRAFICO DE BARRAS",las=1,xlab="glucemia")



6.2.2. Sin bibliotecas

Si bien existen diferentes bibliotecas (ggplots, gplots, lattice, etc) y funciones que permiten hacer gráficas de barras. Encaremos este tema con objetivos puesto en el futuro. En esta clase mostraremos como hacer gráficas barplot con recursos mínimos y mucha imaginación. Este mecanismo luego con el uso de script le permitirá construir la gráfica que desee.

Haremos primero un ejercicio simple para entender el mecanismo.

Introduzcamos la tablaR261 de la planilla de cálculo tablaR2-6

```
> tablaR261<-read.table("clipboard",header=TRUE,dec=",",sep="\t",encoding="latin1")

> tablaR261
   glucemia
1      1.10
2      0.80
3      0.90
4      0.85
5      0.84
6      0.97
7      1.20
8      1.20
9      1.10
10     1.00
11     1.50

> summary(tablaR261)
    glucemia
 Min.    :0.800
 1st Qu .:0.875
 Median :1.000
 Mean   :1.042
 3rd Qu.:1.150
 Max.   :1.500
> ncol(tablaR261)

[1] 1

> nrow(tablaR261)

[1] 11

> head(tablaR261)
  glucemia
1     1.10
2     0.80
3     0.90
4     0.85
5     0.84
6     0.97

> names(tablaR261)
[1] "glucemia"
```

> is.data.frame(tablaR261)
[1] TRUE

graficaremos la media como una barra y el desvío estándar como una línea en T.

calculemos la media y el desvío estándar con funciones conocidas

> mean(tablaR261$glucemia)

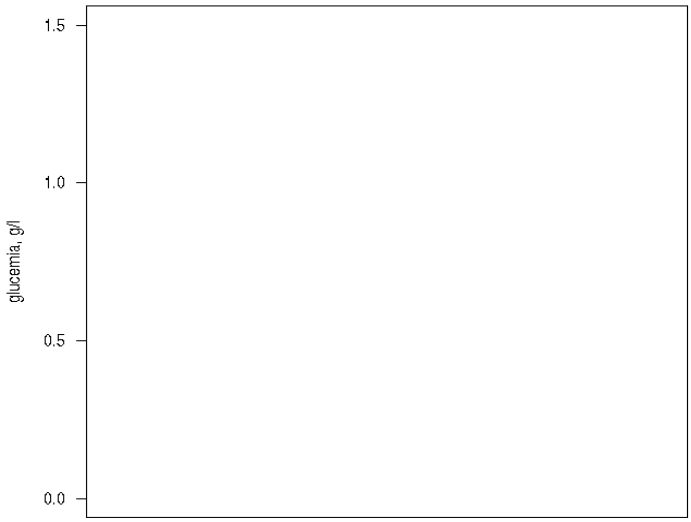
[1] 1.041818

> sd(tablaR261$glucemia)

[1] 0.2073074

Hagamos un gráfico vacío. Para ello asignamos los valores 0 y 0 a los dos primeros elementos del gráfico y además al argumento type lo fijamos en "n". Además introducimos el argumento xaxt que al tomar el valor "n" , elimina rótulos del eje horizontal.

> plot(0,0,type="n",xlim=c(0,5),ylim=c(0,1.5),ylab="glucemia, g/l",xaxt="n",xlab="",las=1)



Ahora grafiquemos la barra. Para ello utilizamos la función rect(). Esta función hace un rectángulo. La misma indica los vértices del rectángulo comenzando por el inferior izquierdo (son los dos primeros valores x e y) y siguiendo por el superior derecho (son los valores tercero y cuarto que representan respectivamente los valores x e y)

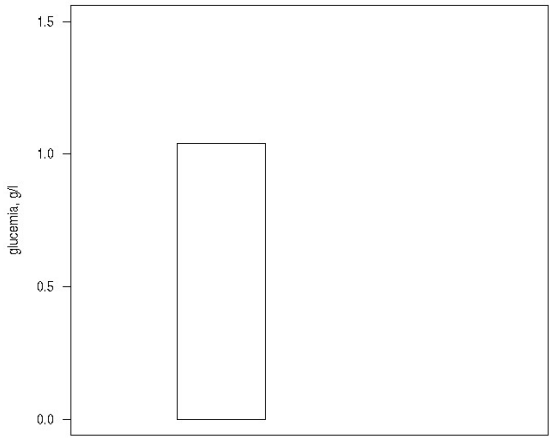
Queremos que la barra arranque en y=0 y llegue hasta y=media de las glucemias

> rect(xlef=1,ybottom=0,xright=2,ytop=mean(tablaR261$glucemia))

podemos escribirlo sin indicar la ubicación de los puntos y se asume en ese orden.

> rect(1,0,2,mean(tablaR261$glucemia))

este código dibuja un rectángulo que está apoyado sobre y=0 y llega hasta el valor de la media y tiene un ancho de 1 unidad, ubicándose la barra en el valor 1.



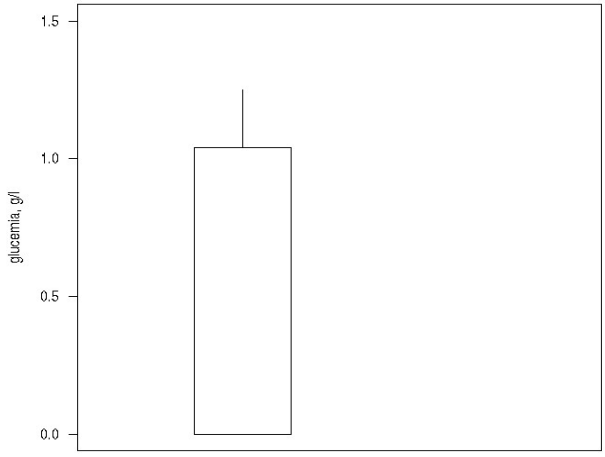
ahora grafiquemos el desvío estándar que lo haremos como una T. para ello tenemos que dibujar dos líneas, cosa que haremos con la función segments(). En esta función se expresa el punto desde donde se inicia el segmento (x0 e y0) y hasta donde llega (x1 e y1)

> segments(x0=1.5,y0=mean(tablaR261$glucemia),x1=1.5,y1=mean(tablaR261$glucemia) +sd(tablaR261$glucemia))

Este código nos está diciendo que trazaremos una línea que se origina en el punto (1.5,media de los valores), es decir este punto está en el medio de la barra y la parte superior. El segmento termina en el punto (1.5,media+desvío estándar)

puede escribirlo sin colocar x e y con sus subíndices.

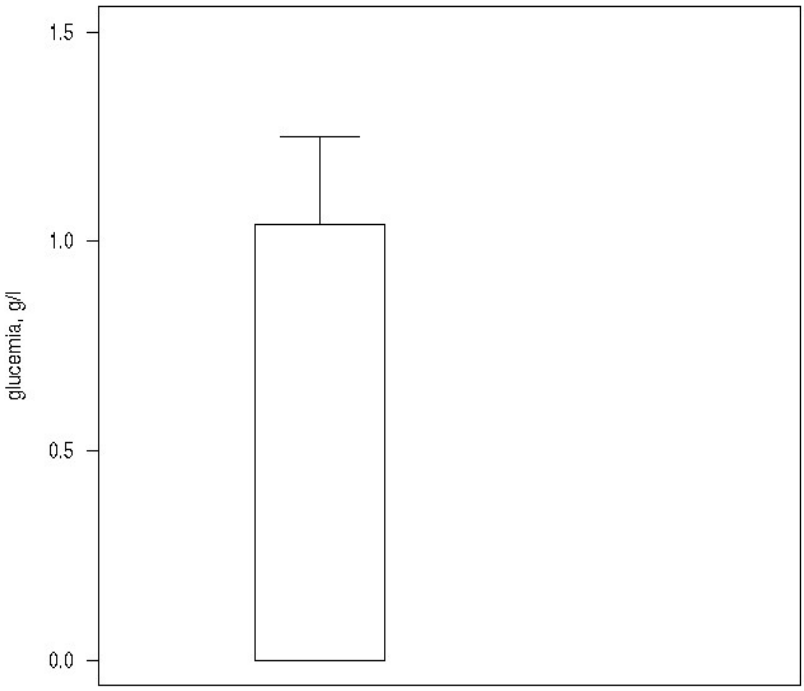
> segments(1.5,mean(tablaR261$glucemia),1.5,mean(tablaR261$glucemia) +sd(tablaR261$glucemia))



ahora graficaremos la línea horizontal para que el desvío tenga forma de T. Esta línea estará ubicada a la altura media+sd (es decir al final de la línea anterior) y se extenderá hacia la derecha e izquierda del segmento trazado una cierta longitud, por ejemplo 0,3 unidades. Como la línea

vertical estaba en x=1.5, haremos que este segmento se origine en 1.2 y a la altura de media+sd, y se extienda hasta 1,8 (es decir 0,3 unidades más que donde está la línea vertical) y a la misma altura (media+sd)

```
> segments(1.2,mean(tablaR261$glucemia)
+sd(tablaR261$glucemia),1.8,mean(tablaR261$glucemia)+sd(tablaR261$glucemia))
```



Veamos algunos parámetros de la función rect()
density: Si tiene el valor NULL, permite colores plenos. Si se coloca un número coloca líneas.

angle: solo tiene valor si density es distinto de NULL, el número indica la inclinación de las líneas de relleno.

col: da el color de relleno del rectángulo.

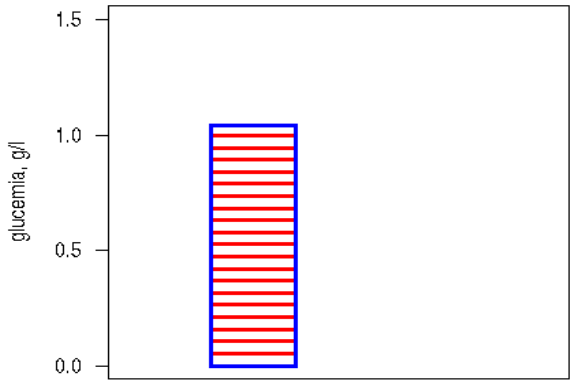
border: da el color del borde del rectángulo.

lty: tipo de línea del borde y el relleno.

lwd: permite modificar el grosor de las líneas de bordes y relleno.

Veamos una barra utilizando estos argumentos

```
> plot(0,0,type="n",xlim=c(0,5),ylim=c(0,1.5),ylab="glucemia, g/l",xaxt="n",xlab="",las=1)
>rect(1,0,2,mean(tablaR261$glucemia),density=10,angle=0,col="red",border="blue",lty=1,lwd=3)
```



Si deseamos escribir el nombre de la serie de datos, tenemos que habilitar la escritura fuera del recuadro. Para ello debemos modificar un parámetro de las gráficas. Cabe aclarar acá que las gráficas tiene argumentos (cosas que podemos cambiar modificando valores dentro de la función, por ejemplo col="red") y además tiene parámetros, que debemos modificar previamente y permiten hacer si se lo desea cambios permanentes a las gráficas de acuerdo a nuestra conveniencia. Ya veremos este complejo pero efectivo tema en otra clase. En este caso modificaremos temporariamente la posibilidad de escribir fuera de la gráfica. Veamos como funciona esto.
Reconstruimos la gráfica anterior

```
> plot(0,0,type="n",xlim=c(0,5),ylim=c(0,1.5),ylab="glucemia, g/l",xaxt="n",xlab="",las=1)
>rect(1,0,2,mean(tablaR261$glucemia),density=10,angle=0,col="red",border="blue",lty=1,lwd=3)
```

Ahora probamos a escribir algunos textos en la gráfica. El primero está en la posición 3 del eje horizontal y 1.5 del eje vertical. Recuerde que la escala del eje horizontal fue fijada en la función plot() con el argumento xlim=c(0,5).

```
> text(3,1.5,labels="glucemia",cex=0.9)
```

podemos verlo en la parte superior interna del recuadro. Modifiquemos algún argumento de la función text()

```
> text(3,1,labels="glucemia",cex=0.9)
```

aparece un poco más abajo, al igual que el siguiente

```
> text(3,0.5,labels="glucemia",cex=0.9)
```

```
> text(3,0,labels="glucemia",cex=0.9)
```

este apareció en la parte inferior. Veamos el siguiente

```
> text(3,-0.2,labels="glucemia",cex=0.9)
```

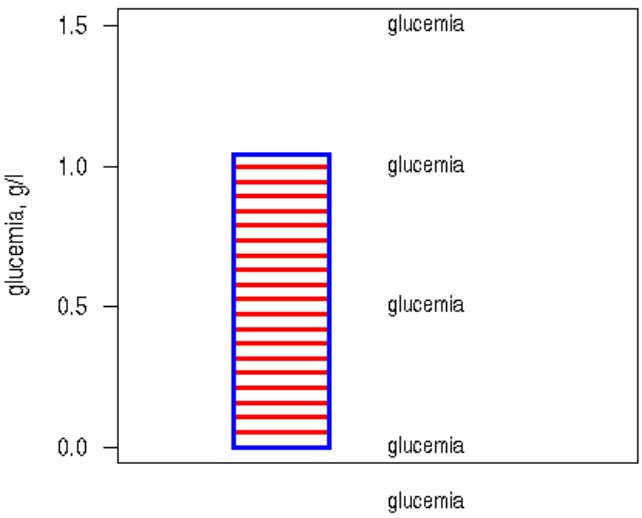
vemos que este texto no apareció en la gráfica, ya que el parametro xpd=F (no permite escritura fuera del gráfico).

Entonces habilitamos esta posibilidad. Para ello utilizamos el siguiente código.

> par(xpd=T)

Ahora tenemos habilitada la escritura fuera del gráfico. En clases venideras veremos uso de parámetros. Volvamos a escribir ahora el texto

> text(3,-0.2,labels="glucemia",cex=0.9)



Vemos que ahora la palabra glucemia nos apareció fuera del gráfico.

Veamos una gráfica sencilla con un poco de estadística. Supongamos que tenemos tres grupos que se hallan en la hoja tablaR262 de la planilla de cálculo tablaR2-6.xls, que ya introdujo al principio de esta clase. Si no lo hizo, hágalo ahora

> tablaR262<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")

```
> tablaR262
   tratamiento glucemia
1      control     1.10
2      control     0.80
3      control     0.90
4      control     0.85
5      control     0.84
6      control     0.97
.............
30          t2     0.40
31          t2     1.50
32          t2     1.60
33          t2     0.80
```

calculamos la media de la glucemia de cada tratamiento

```
> tapply(tablaR262$glucemia,tablaR262$tratamiento,mean)
  control       t1       t2
1.0418182 1.9000000 0.8454545
```

y calculamos el desvío estándar de cada tratamiento

```
> tapply(tablaR262$glucemia,tablaR262$tratamiento,sd)
  control       t1       t2
0.2073074 0.4449719 0.3777926
```

Comparemos las medias utilizando una análisis de la variancia a un criterio (ANOVA a un criterio), utilizando la función aov(). El análisis estadístico que aplicaremos a continuación usted lo verá en detalle y con sus fundamentos en el módulo 3. Por lo tanto no se preocupe por los códigos sino solo concéntrese en los datos que obtendremos y cómo lo incluiremos en los gráficos. Resaltaremos los datos a utilizar

```
> aovtablaR262<-aov(glucemia~tratamiento,data=tablaR262)

> summary(aovtablaR262)
             Df Sum Sq Mean Sq F value   Pr(>F)
tratamiento   2  6.919   3.460   27.05 1.93e-07 ***
Residuals    30  3.837   0.128
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> library(agricolae)
> postest<-LSD.test(aovtablaR262,"tratamiento")

> postest
$statistics
    Mean       CV          MSerror       LSD
  1.262424 28.32905 0.1279012 0.3114366

$parameters
  Df  ntr  t.value  alpha  test        name.t
  30  3    2.042272 0.05   Fisher-LSD  tratamiento

$means
            glucemia       std          r  LCL        UCL      Min  Max
control     1.0418182  0.2073074  11  0.8215992  1.262037  0.8  1.5
t1          1.9000000  0.4449719  11  1.6797811  2.120219  1.1  2.8
t2          0.8454545  0.3777926  11  0.6252356  1.065673  0.4  1.6
```

resaltados en amarillo vemos las medias y desvíos estándar de cada grupo

```
$comparison
NULL

$groups
   trt       means       M
1 t1        1.9000000   a
2 control   1.0418182    b
```

3 t2 0.8454545 b

Las letras de la columna M indican la diferencia entre los grupos. Todas las letras distintas entre dos filas indica diferencias significativas con p-value<0,05. En este caso t1 es diferente de control y t2, pero no los son control y t2.

Grafiquemos todo esto!!!!. Para ello utilizaremos los valores de las medias y desvíos estándar generados anteriormente con la función tapply() y los significados estadísticos resaltados en amarillo en el objeto postest.

1- gráfico vacío. Haremos barras de 1 unidad de ancho, separadas 0.5 unidades, dejando 0,5 unidades a cada lado, por lo tanto necesitamos en total 5 unidades en el eje horizontal, por ello en el gráfico fijaremos el argumento xlim=c(0,5). La altura de la gráfica nos tiene que permitir graficar la media mas el desvío estándar, que lo tenemos en las tablas superiores, indicado por std. Mirando dicha tabla llegaríamos a un valor de 2,4 para el grupo t1 ..."t1 1.9000000 + 0.4449719". Dejaremos un lugar sobre el desvío estándar para colocar la letra del significado. Por lo tanto ylim=c(0,2.7) y por otro lado dejaremos los parámetros que ya tenemos. Además habilitamos escritura fuera del gráfico con par(xpd=T). Si ya lo hizo en su sesión de trabajo no debería volver a hacerlo. La figura siguiente esquematiza el texto anterior

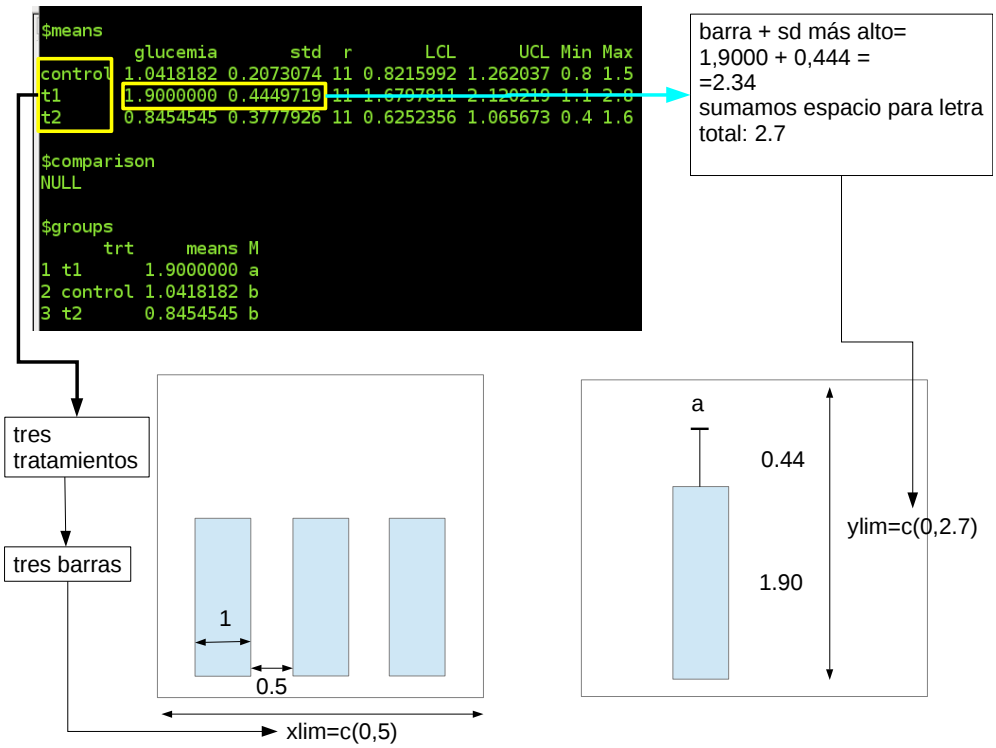


Figura 6.2. esquema en como fijar xlim e ylim en base a sus datos de medias y sd hallados luego de hacer el análisis de la variancia y su postest correspondiente.

Con estos datos iniciamos el gráfico, haciendo uno sin datos, pero con los límites establecidos

> plot(0,0,type="n",xlim=c(0,5),ylim=c(0,2.7),ylab="glucemia, g/l",xaxt="n",xlab="",las=1)

haremos los rectángulos que representarán a las medias de cada grupo. El primer rectángulo que

representa a la media del grupo control deja 0.5 unidades a la izquierda, es decir se extiende desde 0,5 a 1,5 (ya que el ancho es 1). Lo haremos con el código siguiente, en color rojo y bordes negros

>
rect(0.5,0,1.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"]),density=NULL,col="red",border="black",lty=1,lwd=1)

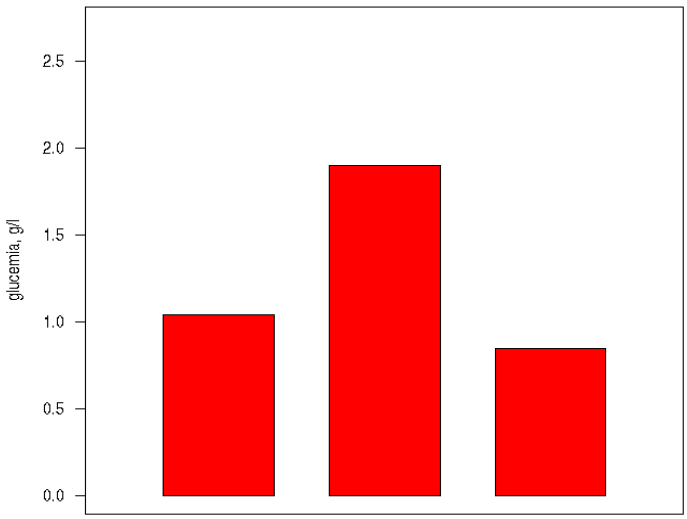
el segundo rectángulo, que representará al grupo t1, también tiene 1 de ancho, pero como dejamos 0,5 unidades a partir del rectángulo anterior, por lo que se extenderá entre 2 a 3

>
rect(2,0,3,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),density=NULL,col="red",border="black",lty=1,lwd=1)

ahora la media del grupo t2

>
rect(3.5,0,4.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),density=NULL,col="red",border="black",lty=1,lwd=1)

hasta acá hemos graficado las medias y deberíamos tener la siguiente figura



Ahora graficaremos los segmentos que representarán a los desvíos estándar. En primer lugar graficamos el desvío del grupo control, que colocaremos en la parte superior del del rectángulo, apoyado sobre él y en su medio

> segments(1,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"]),1,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),lty=1,col="black")

luego graficamos el desvío estándar de t1

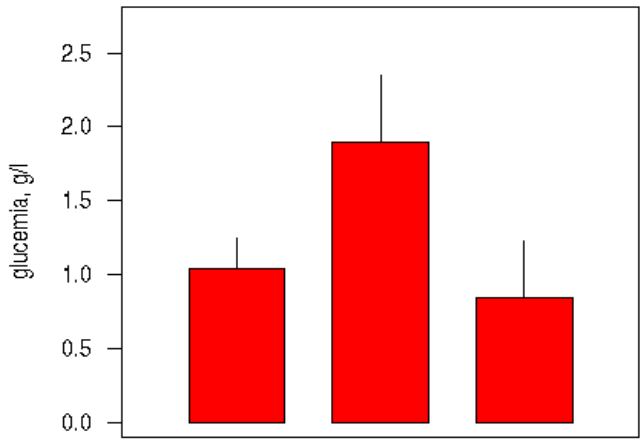
> segments(2.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),2.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])

+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),lty=1,col="black")

y finalmente t2

> segments(4,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),4,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),lty=1,col="black")

así debe ir quedando a esta altura del trabajo



ahora haremos los segmentos para que la representación del desvío estándar tenga forma de "te".

Primero para el grupo control

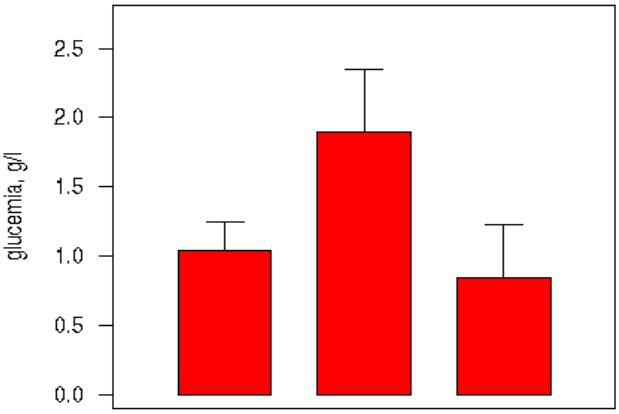
> segments(0.8,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),1.2,
mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),lty=1,col="black")

luego para t1

> segments(2.3,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),2.7,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),lty=1,col="black")

y finalmente para t2

> segments(3.8,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),4.2,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),lty=1,col="black")



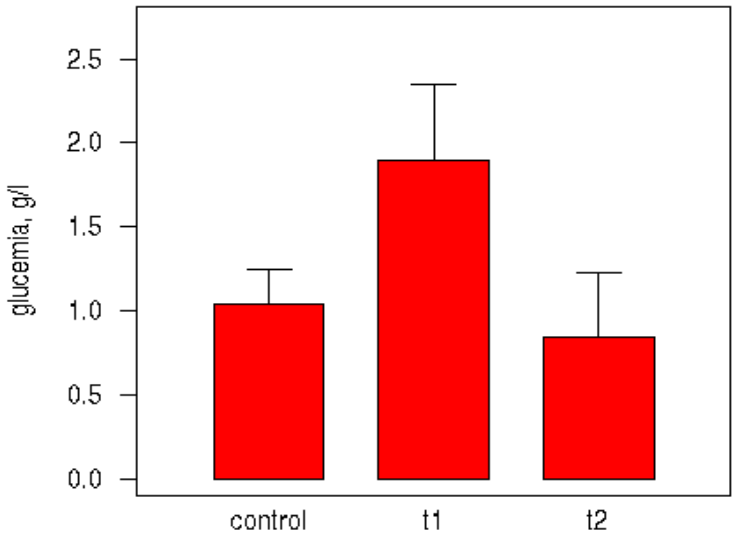
ahora ponemos los rótulos de las barras. Como debemos escribir fuera del gráfico habilitemos esta posibilidad con

par(xpd=T)

> text(1,-0.25,"control")

> text(2.5,-0.25,"t1")

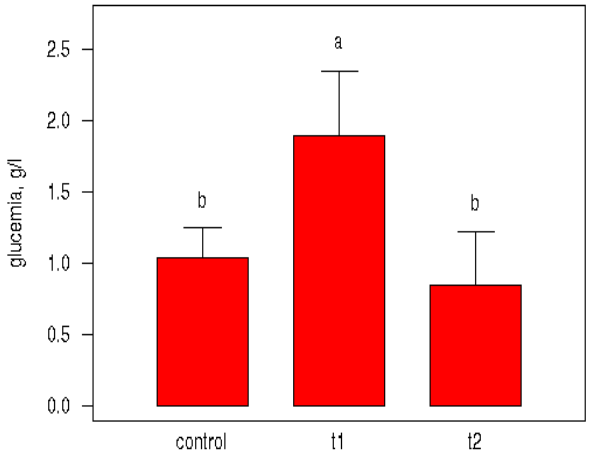
> text(4,-0.25,"t2")



ahora el significado estadístico, es decir las letras que indicarán si son o no diferentes las medias de cada grupo. Las pondremos sobre las "Tes" , ubicada a 0,2 unidades de las mismas. Estas letras

las obtenemos del objeto postest generado anteriormente y expresadas en la columna M de la tabla $groups

```
> text(1,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"])+0.2,"b")

> text(2.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"])+0.2,"a")

> text(4,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
 +sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"])+0.2,"b")
```



Seguramente le puede resultar complejo esta forma de realizar un gráfico, existiendo programas que con solo apretar un botón permiten hacerlo. Pero es una nueva metodología, haciendo lo que uno desea y no solo lo que un programa puede proveerle. En el módulo 5 de este curso verá como realizarlo utilizando scripts. Ahora veremos un breve anticipo de este mecanismo

6.3. Script: primeros pasos

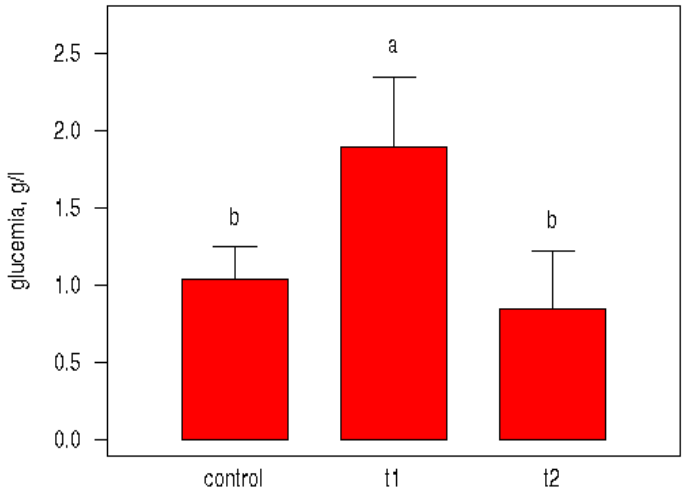
Primeros pasos hacia un script (instrucciones programadas en R, que veremos en módulos posteriores).
Organicemos todas las instrucciones explicadas anteriormente y coloquemos un signo de comentario (#) a todo texto que no es instrucción para R. Además saquemos las figuras y el prompt ">".
Le debería quedar algo como lo que se muestra a continuación resaltado en amarillo.

```
plot(0,0,type="n",xlim=c(0,5),ylim=c(0,2.7),ylab="glucemia, g/l",xaxt="n",xlab="",las=1)

#haremos los rectángulos de las medias
#el primer rectángulo deja 0.5 unidades a la izquierda, es decir se extiende desde 0,5 a 1,5 (ya que el ancho es 1)
  rect(0.5,0,1.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"]),density=NULL,col="red",border="black",lty=1,lwd=1)
```

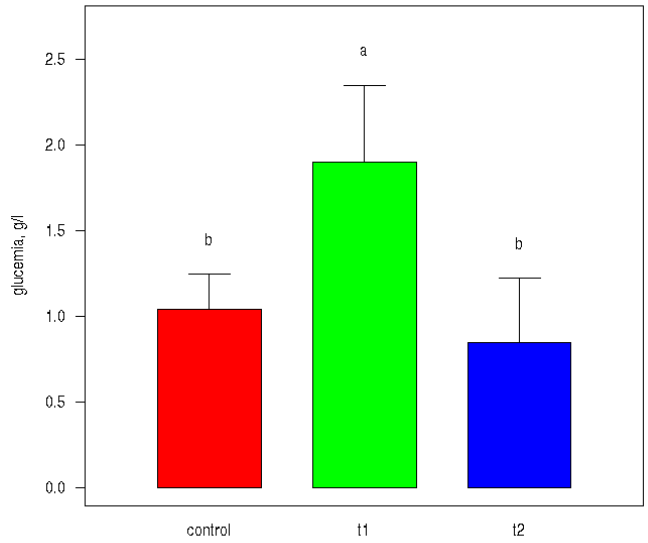
```
#el segundo rectángulo tambien tiene 1 de ancho, pero como dejamos 0,5 entre barras irá de 2  a 3
  rect(2,0,3,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),density=NULL,col="red",b
order="black",lty=1,lwd=1)

rect(3.5,0,4.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),density=NULL,col="red
",border="black",lty=1,lwd=1)

segments(1,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"]),1,
mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),lty=1,col="black")

segments(2.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),2.5,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),lty=1,col="black")

segments(4,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),4,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),lty=1,col="black")

#así debe ir quedando a esta altura del trabajo

#ahora haremos la "tes"
segments(0.8,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),1.2,
mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"]),lty=1,col="black")

segments(2.3,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),2.7,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"]),lty=1,col="black")

segments(3.8,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),4.2,
mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"]),lty=1,col="black")

#ahora ponemos los rótulos de las barras. Como debemos escribir fuera del gráfico habilitemos
esta posibilidad con
par(xpd=T)
text(1,-0.25,"control")
text(2.5,-0.25,"t1")
text(4,-0.25,"t2")

#ahora el significado estadístico, es decir las letras que indicarán si son o no diferentes. Las
pondremos sobre las "Tes" , a 0,2 unidades de las mismas

text(1,mean(tablaR262$glucemia[tablaR262$tratamiento=="control"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="control"])+0.2,"b")
```

```
text(2.5,mean(tablaR262$glucemia[tablaR262$tratamiento=="t1"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t1"])+0.2,"a")

text(4,mean(tablaR262$glucemia[tablaR262$tratamiento=="t2"])
+sd(tablaR262$glucemia[tablaR262$tratamiento=="t2"])+0.2,"b")
```

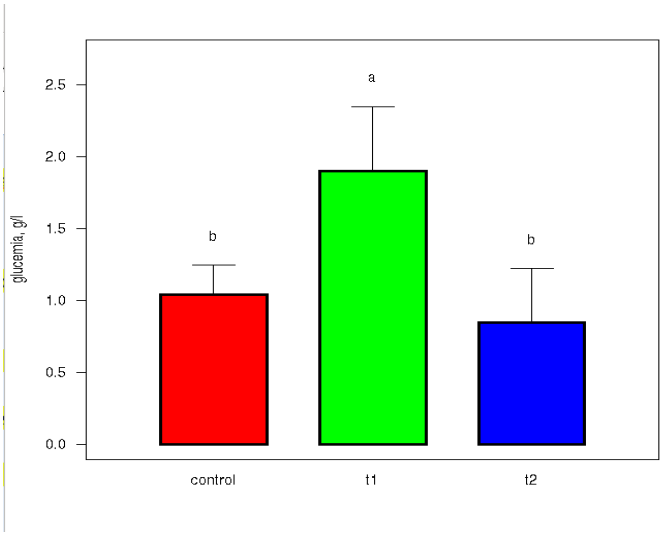
Copie todo el texto resaltado en amarillo. Pase a la consola, pegue este texto y oprima enter.
debería aparecerle esto



Usted termina de elaborar su primer script, es decir una serie de instrucciones para el procesamiento de sus datos. Aunque ahora parece ineficiente, ya verá en el futuro como aumentar dicha eficiencia.
Realice algunas modificaciones al texto, copie y ejecute como para ir entendiendo esta nueva filosofía de hacer gráficos. El uso cotidiano de este mecanismo de gráfico irá transformando a usted en un experto, transportándolo de su ubicación el el teclado a una posición donde se sentirá parte del procesador de su computadora. Una cuestión de paciencia, constancia y un poco de fe.
Le propongo algunas preguntas
¿ Qué parte del script modificó si la gráfica obtenida fue?



¿ y esta?



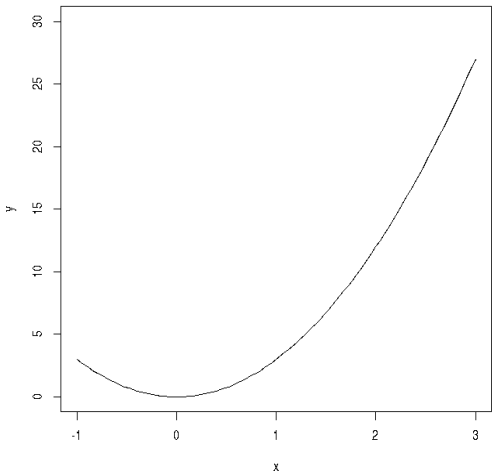
6.4. Gráficas de funciones

Si bien el tema con mayor detalle será desarrollado en clases posteriores, haremos en este módulo una breve introducción, de como definir una función R, obtener su gráfica y marcar un área bajo la gráfica de una función. Tenemos una función en estudio, por ejemplo

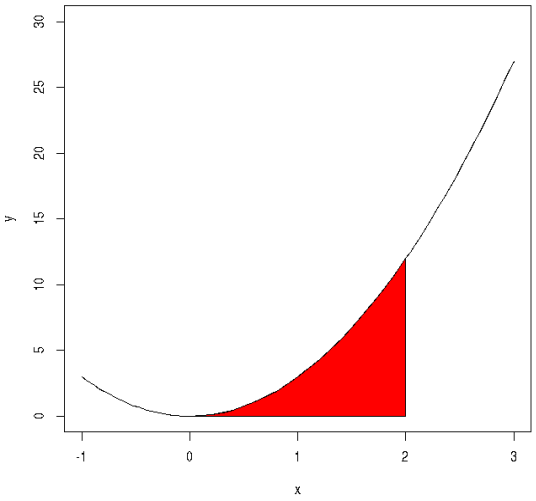
$$y = 3x^2$$

la introducimos en R con el siguiente código

> y<-function(x){3*x^2}
luego graficamos la función en un rango deseado de ambas variables. Supongamos que deseamos ver la gráfica de la función en el intervalo [-1,3] y resaltar el área entre los valores de x pertenecientes al intervalo [0,2]. Como la función el el valor 3 tendrá un valor de 27 para la y, tomamos también un rango adecuado para el argumento ylim.
> plot(y,-1,3,ylim=c(0,30))
hallamos



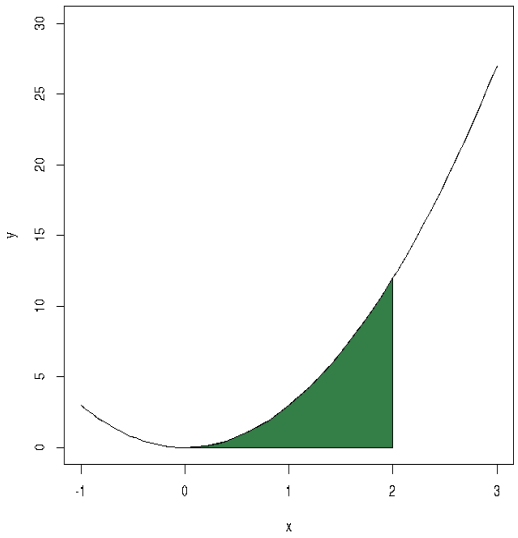
luego demarcamos el área con la función polygon(). Esta función incluye dos vectores que indican la sucesión de puntos de un polígono de la cantidad de lados deseado. Nuestro polígono irá recorriendo la línea de la función desde el valor 0 hasta el valor 2 y luego agregará dos puntos en el valor (2,0) y (0,0) para cerrar el polígono, al que le daremos color rojo.
definimos un vector con los valores de x indicados
> x=c(seq(0,2,0.1))
y ahora definimos el polígono indicando los puntos.
> polygon(x=c(x,2,0),y=c(y(x),0,0),col='red')
con lo que obtenemos



Podemos cambiar el color del área, modificando el color utilizado en el argumento col. O bien utilizando el argumento rgb(a,b,c), donde a, b y c son número entre 0 y 1. Si a, b y c toman simultáneamente el valor 0, el área será negra. Si a, b y c toman el valor 1, el área será blanca. Si ellos toman valores donde cada uno es un cociente entre un número del intervalo [0,255] y 255, podrá lograr cualquier color, por ejemplo

```
> polygon(x=c(x,2,0),y=c(y(x),0,0),col=rgb(50/250,125/250,70/250))
```

obtendrá



7. Clase 7

Vídeo: https://youtu.be/3r2IIkgas7I

Tabla de datos: http://hdl.handle.net/2133/10519

7.1. Grafico 3D

A continuación veremos gráficos que nos permiten visualizar tres variables

7.1.1. Scatter plot 3D

Scatterplot 3D nos permite graficar puntos cuyas coordenadas son tres variables. Utiliza la biblioteca (scatterplot3d)

Introduzcamos la tablaR271 de la planilla de cálculo tablaR2-7.ods/xls

```
> tablaR271<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

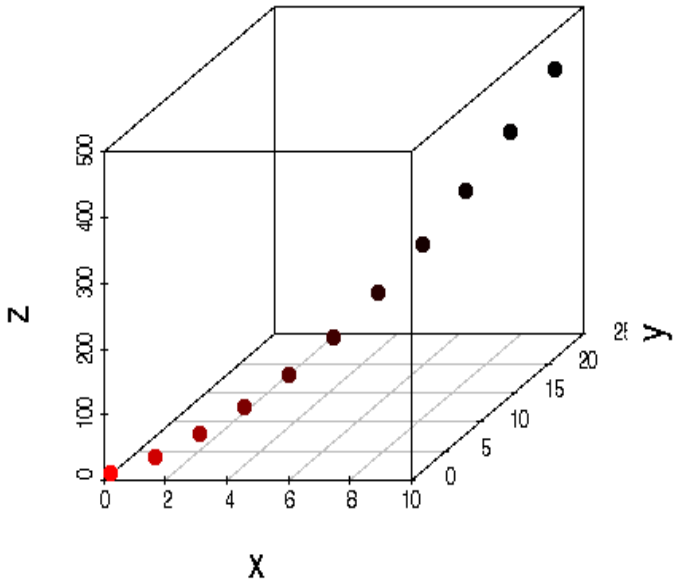
```
> summary(tablaR271)
       x               y               z               w        
 Min.   : 0.0   Min.   :  1.0   Min.   :  1.0   Min.   :  2.0  
 1st Qu.: 2.5   1st Qu.:  6.0   1st Qu.: 37.0   1st Qu.: 23.5  
 Median : 5.0   Median : 11.0   Median :121.0   Median :111.0  
 Mean   : 5.0   Mean   : 11.0   Mean   :161.0   Mean   :156.0  
 3rd Qu.: 7.5   3rd Qu.: 16.0   3rd Qu.:257.0   3rd Qu.:260.0  
 Max.   :10.0   Max.   : 21.0   Max.   :441.0   Max.   :400.0  
```

Graficaremos con mínimos argumentos los valores x, y, z de la tabla. En primer lugar cargamos la biblioteca scatterplot3d

```
> library(scatterplot3d)
```

A continuación con la función scatterplot3d() graficamos los puntos (x,y,z) de la tablaR271

```
>
scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=40,type="p",highlight.3d=TRUE,pch=19,xlab="x",ylab="y",zlab="z",cex.lab=1.5)
```



7.1.1.1 argumentos utilizados

angle: gira el gráfico sobre un eje. Es el ángulo entre eje x e y

type: tipo de visualización de los datos. "p" puntos, "l" línea, "h" vertical line

highlight.3d: da diferente coloración a los puntos, indicando la altura. En el ejemplo anterior. los puntos negros son los valores más altos y los rojos los más bajos.

7.1.1.2 Argumentos no utilizados

Hay otros argumentos que no fueron utilizados y podrán ser probados para observar el efecto que tiene sobre la gráfica. En el caso de argumentos que tienen valor T o F se indica primero el valor por defecto. De no especificarse estos argumentos en la función, ese es el valorque tomará.

axis: T o F. dibuja los ejes

tick.marks: coloca los ticks en el eje, si axis=T

tick.labels: coloca los números en los ticks de los ejes, si axis=T

grid= T o F. Coloca o saca la grilla

box= T o F. Dibuja o no una caja para el gráfico.

col.axis: color de los ejes

col.grid: color de la grilla

col.lab: color de los rótulos de los ejes.

cex.simbols: tamaño de puntos

ces.axis: tamao de números de los ejes

cex.lab: tamaño de los rótulos de los ejes.

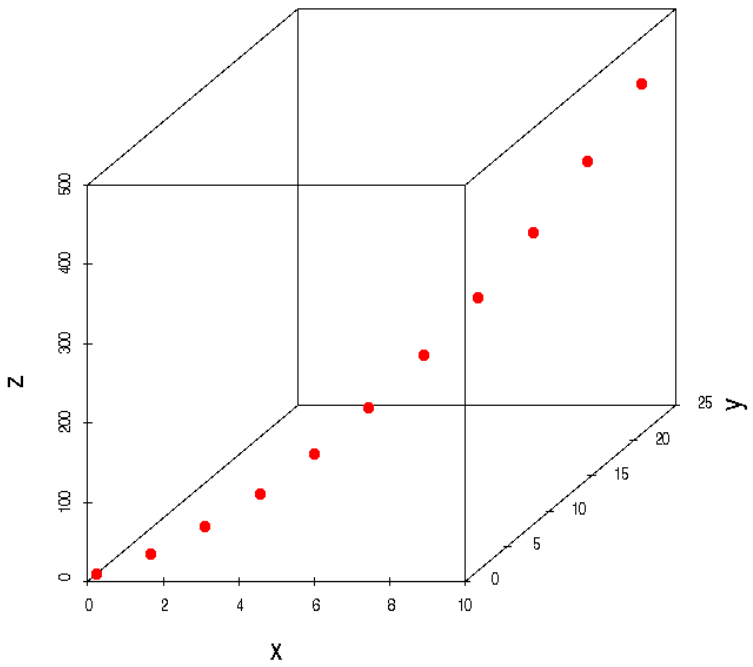
font.lab: tipo de letra de los rótulos de los ejes: 1 normal, 2 negrita, 3 cursiva.

font.axis: tipo de letra de los números de los ejes: 1 normal, 2 negrita, 3 cursiva.

Agreguemos o modifiquemos argumentos

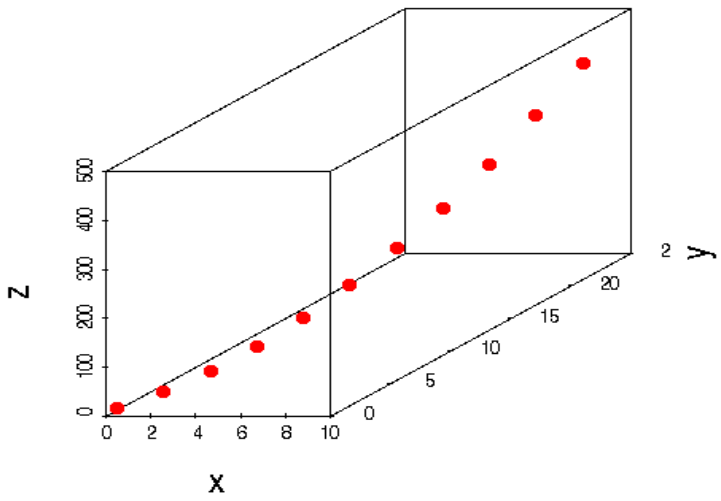
Colocaremos los puntos solo en color rojo y sacaremos la grilla.

```
>
scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=40,type="p",color="red",pch=19,xlab
="x",ylab="y",zlab="z",cex.lab=1.5,grid=F)
```



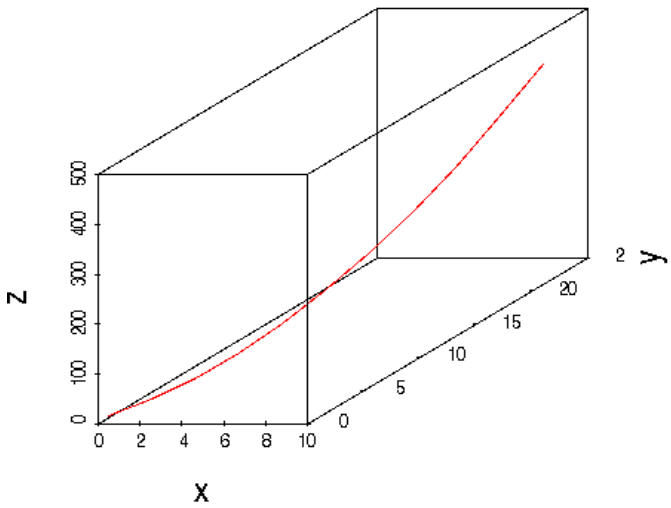
Por defecto, los tres ejes tienen igual longitud, con el argumento scale.y, se puede cambiar dicha relación. Graficaremos con un eje y que es el doble de x y un ángulo de 30 para la ubicación de los mismos.

```
>
scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=30,type="p",color="red",pch=19,xlab
="x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=2)
```



En lugar de puntos graficaremos una línea, utilizando el argumento type

>
scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=30,type="l",color="red",pch=19,xlab="x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=2)



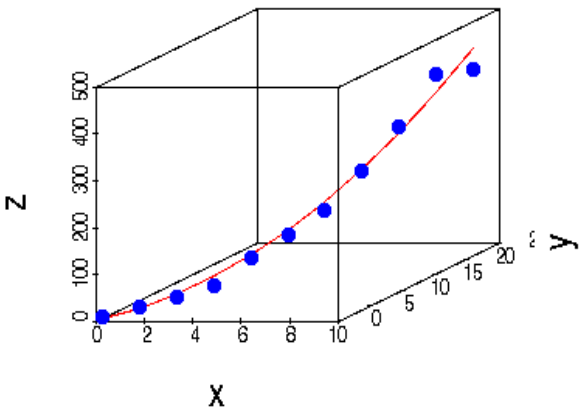
7.1.2. Agregar puntos en gráficos 3d

para ello es necesario graficar utilizando la función scatterplot3d, pero asignando el código a un objeto, en este caso lo llamamos sp3d.

```
> sp3d<-scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=30,type="l",color="red",pch=19,xlab="x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=1)
```

Con el código anterior representamos z como función de x e y con una línea roja. Con el código siguiente graficamos los puntos que representan w en función de x e y como puntos azules.

```
> sp3d$points3d(tablaR271$x,tablaR271$y,tablaR271$w,type="p",col="blue",pch=19)
```

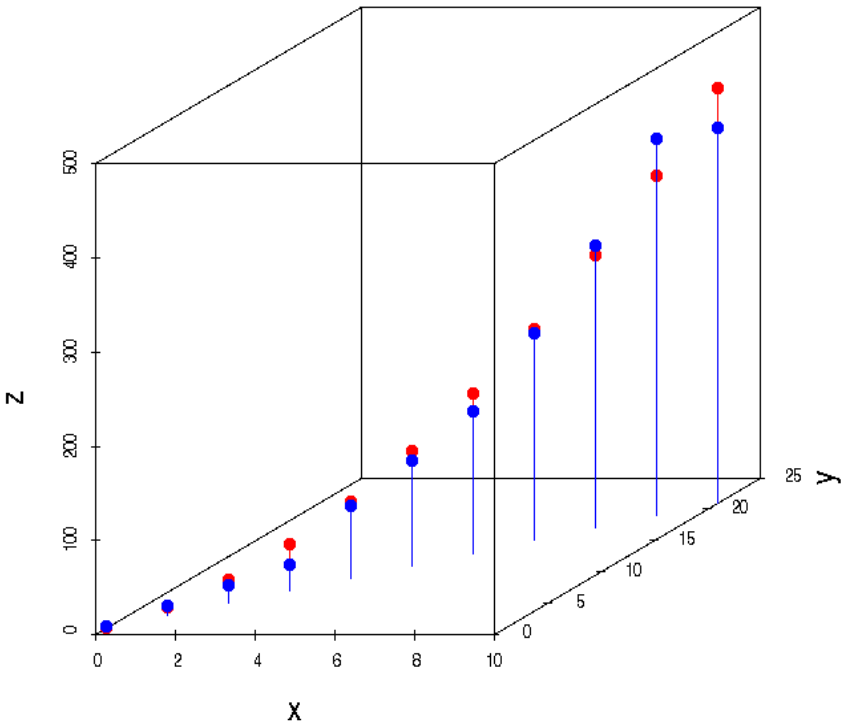


En ciertas ocasiones para ver mejor la distribución es adecuado el gráfico con el argumento type="h", que muestra el punto y una línea hasta la base del gráfico. Veamos esta opción. Primero graficamos los puntos (x,y,z) en color rojo

```
> sp3d<-scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=30,type="h",color="red",pch=19,xlab="x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=1)
```

Luego los puntos (x,y,w) como puntos azules.

```
> sp3d$points3d(tablaR271$x,tablaR271$y,tablaR271$w,type="h",col="blue",pch=19)
```



Esta forma nos permite apreciar con más facilidad la diferencia de valores entre las dos series graficas.

Podemos graficar también puntos de otras tablas siempre y cuando las variables no estén fueran del rango de la gráfica ya realizada. Tomará la primer variable como la x, la segunda como la y, la tercera como z, independientemente de como se llamen.

Introduzcamos la tablaR272 de la planilla de cálculo tablaR2-7.xls/ods

> tablaR272<-read.table("clipboard", header=T,dec=",",sep="\t",encoding="latin1")

hacemos una gráfica con los datos del data.frame tablaR271, graficando z en función de x e y

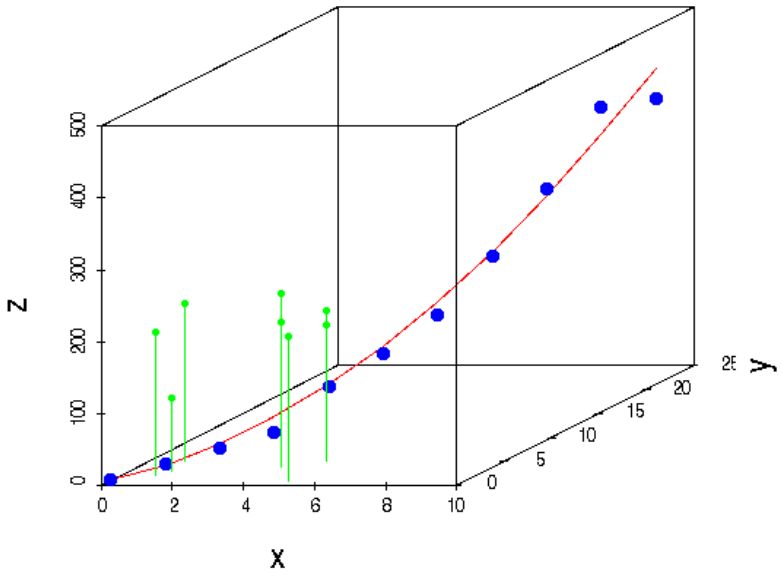
> sp3d<-scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=30,type="l",color="red",pch=19,xlab="x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=1)

luego agregamos los datos de la columna w, del mismo data.frame

> sp3d$points3d(tablaR271$x,tablaR271$y,tablaR271$w,type="p",col="blue",pch=19)

agregamos ahora los puntos de la columna u del data.frame tablaR272

> sp3d$points3d(tablaR272$x,tablaR272$y,tablaR272$u,type="h",col="green",pch=20)



Ahora un poco de entretenimiento y de vista al futuro. A continuación mostramos lo que sería un "script", es decir un conjunto de instrucciones que se ejecutan automáticamente. Esta parte de la clase está puesta a modo ilustrativo, en el módulo 4 y 5 del curso se impartirán los conocimientos para este tipo de aplicaciones. Para ejecutar el scritp marque todo lo que está dentro del rectángulo, cópielo (Control+C) y péguelo en la consola. Sobrevuele el gráfico en búsqueda de relaciones entre las variables

```
for(i in 10:70){
sp3d<-
scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z,angle=i,type="l",color="red",pch=19,xlab="
x",ylab="y",zlab="z",cex.lab=1.5,grid=F,scale.y=1)
sp3d$points3d(tablaR271$x,tablaR271$y,tablaR271$w,type="p",col="blue",pch=19)
sp3d$points3d(tablaR272$x,tablaR272$y,tablaR272$u,type="h",col="green",pch
=20)
Sys.sleep(0.2)
}
```

En el módulo 5 específicamente comenzará el tema de script y programación en R.

7.1.3. Persp

Persp es una función que permite obtener gráficos 3D especialmente de superficies. Se requiere del uso de matrices en lugar de data.frame. Los ejes x e y son los valores de las filas y las columnas de una matriz. Los valores que contiene la matriz son los valores de z. Veamos una construcción sencilla

definimos un vector ***x*** que tiene valores en el rango [-1,1] en escalones de 0,01. Para ello

utilizamos la función seq()

x<-seq(-1,1,0.1)

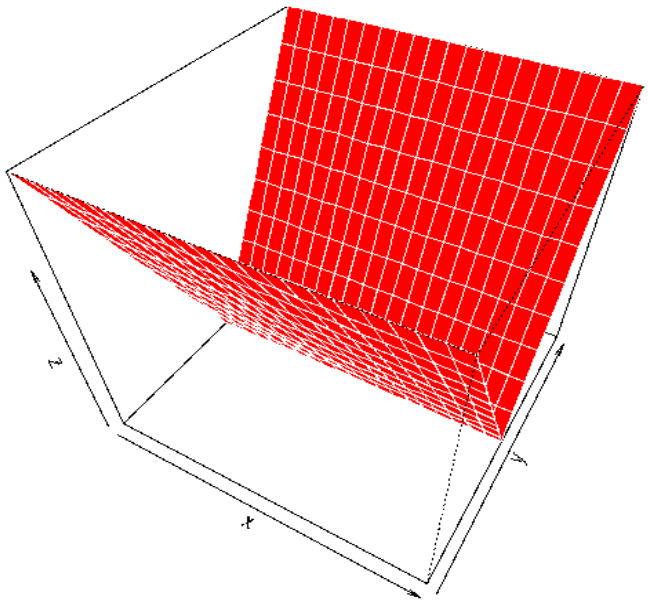
definimos otro vector, al que llamamos ***y*** que tiene valores en el rango [-1,1] en escalones de 0,01, tambien con la función seq().

y<-seq(-1,1,0.1)

definimos los valores de z que son función de x e y con el siguiente código, que asigna el valor a una matriz. Los valores de la matriz se obtienen como la raíz cuadrada de la suma del cuadrado de x e y

> xy<-matrix(data=sqrt(x^2+y^2),nrow=length(x),ncol=length(y),byrow=T)

Hagamos una gráfica sencilla. Graficamos los valores de xy en función de x e y
> persp(x,y,xy,theta=30,phi=45,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")



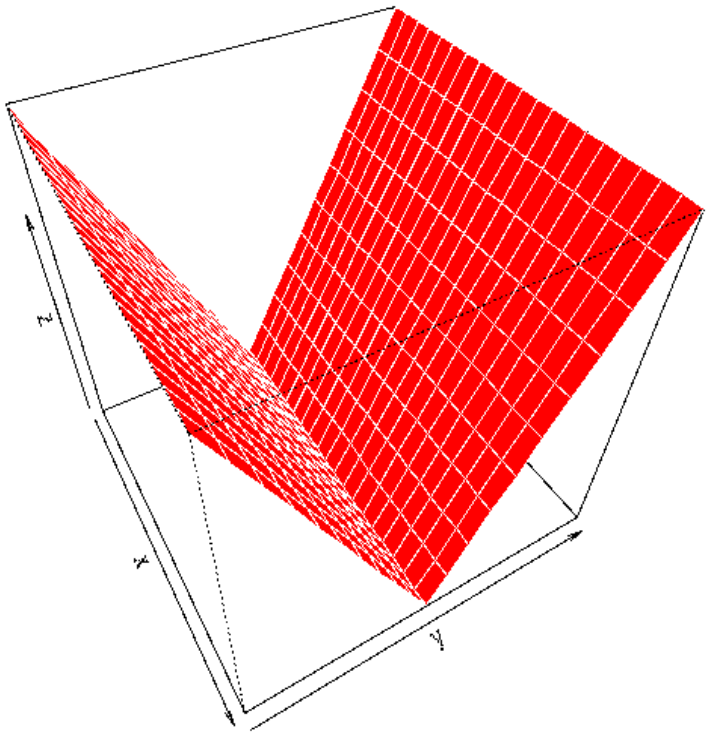
7.1.3.1 Rotar gráficos

La rotamos. Para ello hay dos ángulos, theta y phi. Ambos ángulos producen movimientos diferentes

> persp(x,y,xy,theta=60,phi=45,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")

anticipando el módulo 5 del curso. Utilizamos lo que se llama un bucle for{}. R repetirá 360 veces en este caso, lo que se halla entre llaves. tomando la variable i valores que van desde 0 a 360. Por otra parte cada valor que tome se lo asignará al argumento theta y mostrará el gráfico con ese ángulo. Creará un gráfico cada 0,01 seg. Copie con Control+C el código del recuadro siguiente y péguelo en el sector de códigos de R. Si desea verlo más rápido o lento modifique el valor de Sys.sleep()

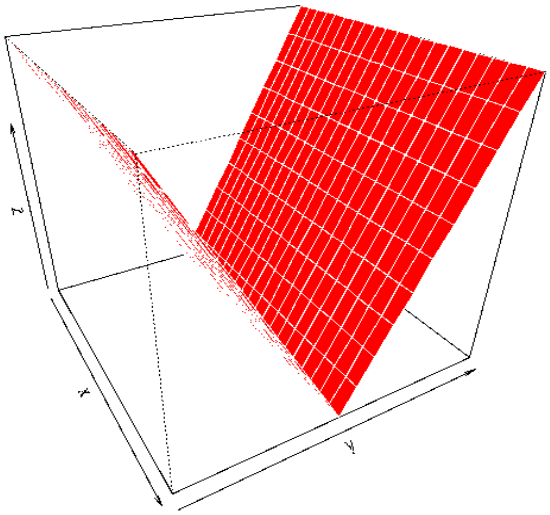
```
for (i in 0:360){
persp(x,y,xy,theta=i,phi=45,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")
Sys.sleep(0.01)
}
```



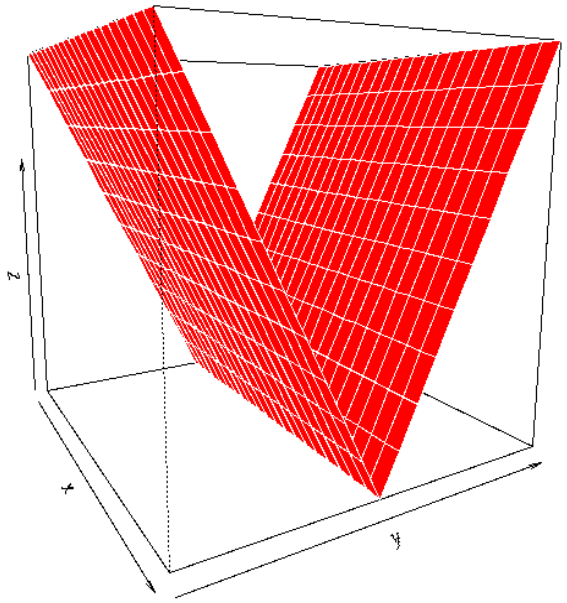
El argumento theta rota al gráfico, dejando en su posición el plano xy.

Veamos ahora el argumento phi. Este argumento fija también un ángulo desde donde miramos el gráfico.

```
> persp(x,y,xy,theta=60,phi=30,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")
```



> persp(x,y,xy,theta=60,phi=10,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")



el ángulo phi le permite dejar fija la caja y rotar el gráfico sobre una línea que atraviesa perpendicularmente el eje z. Pruebe, copie el recuadro y péguelo en la consola, luego oprima enter.

```
for(i in 0:360){
persp(x,y,xy,theta=60,phi=i,col="red",border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")
Sys.sleep(0.05)
}
```

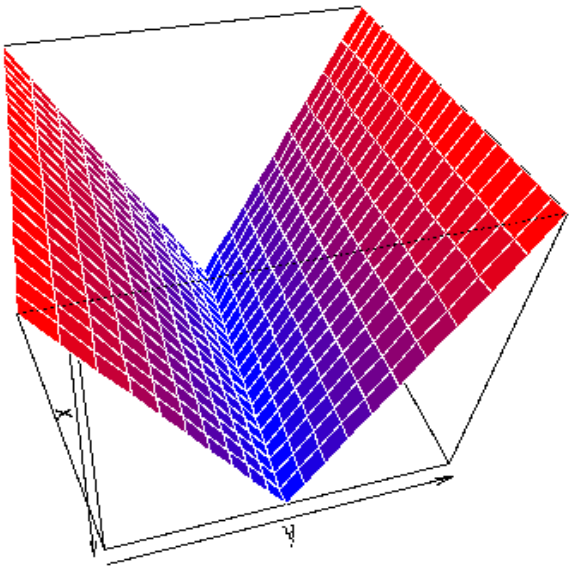
7.1.3.2 Color de la superficie

La utilización de diversos colores en la superficie es útil para resaltar valores diferentes. Para hacer gráfico con graduación indicando valor máximo y mínimo de la variable, utilizamos la función colorRampPalette() y la función jet.colors(). Como se indica a continuación, asignamos el valor de la función colorRampPalette al objeto jet.colors.

```
jet.colors <- colorRampPalette( c("blue", "red") )
```

Asignará color azul a los valores menores y rojo a los mayores. Asignamos a la variable nbcol el valor 100, que nos indicará la cantidad de tonos entre azul y rojo

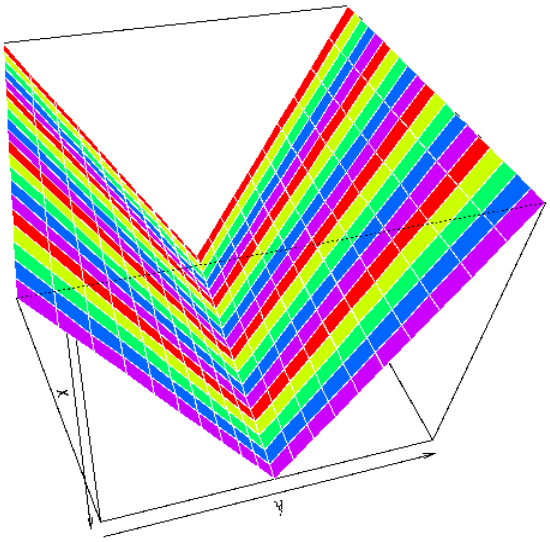
```
nbcol <- 100
color <- jet.colors(nbcol)
zfacet <- xy[-1, -1] + xy[-1, -ncol(xy)] + xy[-nrow(xy), -1] + xy[-nrow(xy), -ncol(xy)]
facetcol <- cut(zfacet, nbcol)
>
persp(x,y,xy,theta=75,phi=45,col=color[facetcol],border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")
```



pruebe otros colores.

En el ejemplo siguiente damos 5 colores del arco iris

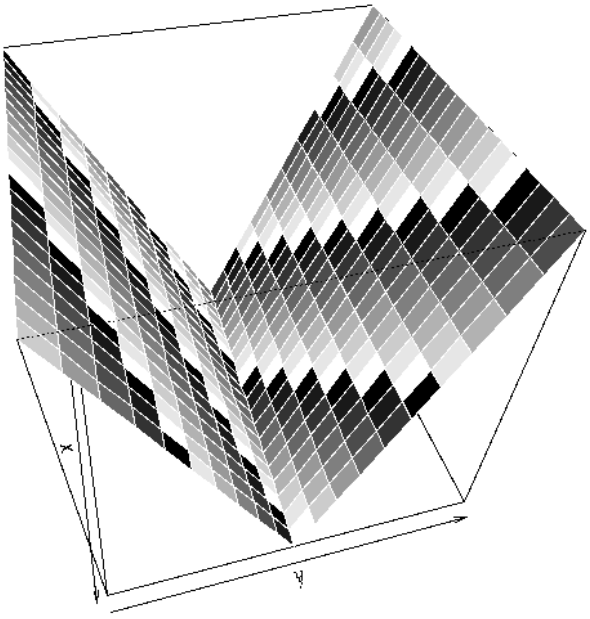
```
>
persp(x,y,xy,theta=75,phi=45,col=rainbow(5),border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")
```



En el siguiente escala de grises, que van desde el negro al blanco

>

persp(x,y,xy,theta=75,phi=45,col=gray(seq(0,1,0.1)),border="white",lwd=0.001,xlab="x",ylab="y",zlab="z")



Y en este lo hacemos con en forma transparente.

> persp(x,y,xy,theta=75,phi=45,col=gray,border="black",lwd=0.001,xlab="x",ylab="y",zlab="z")

Veamos ahora un ejemplo de aplicación. Supongamos que hemos medido la altura promedio de la vegetación una superficie de 30x40 m, los datos se halla en la hoja tablaR273 de la planilla de cálculo tablaR2-7.ods/xls. Introduzca la planilla en su espacio de trabajo como una matriz, es decir no introduzca los números de la primer fila y la primer columna. Copie en el portapapeles las filas 2-42 y las columnas B a AF.

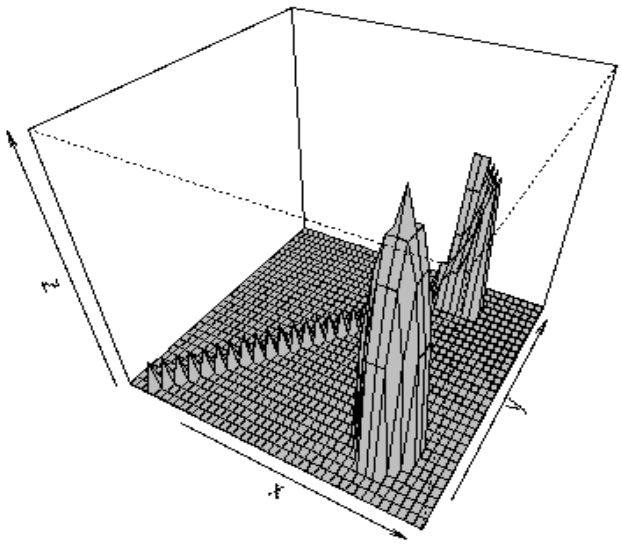
```
> tablaR273<-as.matrix(read.table("clipboard",header=F,dec=",",sep="\t",encoding="latin1"))
```

comprobemos que los datos ingresados son los correspondientes a una matriz

```
> is.matrix(tablaR273)
[1] TRUE
```

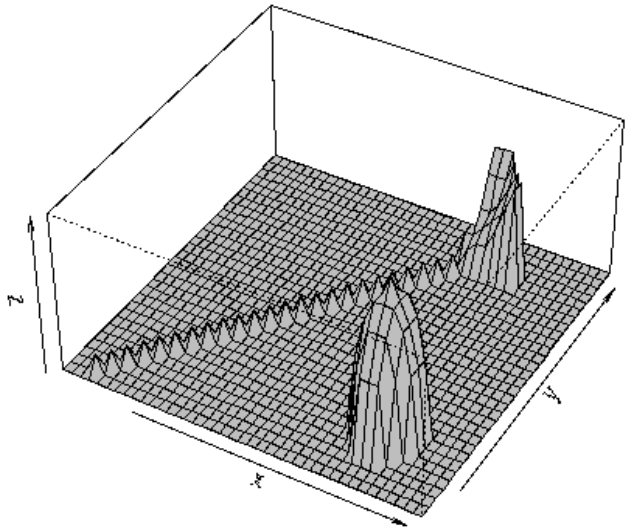
Grafiquemos utilizando la función persp()

```
>
persp(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,theta=30,phi=40,col="gray",border="black",lwd=0.001,xlab="x",ylab="y",zlab="z")
```



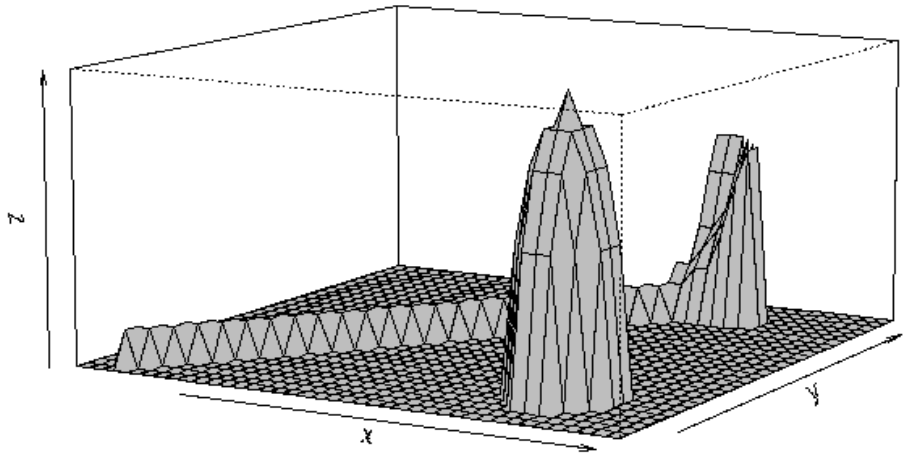
La gráfica claramente nos muestra la altura de vegetación. Si cambiamos algunos argumentos veremos mejor el terreno

```
>
persp(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,theta=30,phi=40,col="gray",border="black",lwd=0.001,xlab="x",ylab="y",zlab="z",expand=0.5,r=10)
```



Si cambia el argumento phi, hasta sabrá si tiene que podar alguno de los árboles o emparejar el cerco de arbustos.

```
>
persp(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,theta=30,phi=10,col="gra
y",border="black",lwd=0.001,xlab="x",ylab="y",zlab="z",expand=0.5,r=10)
```

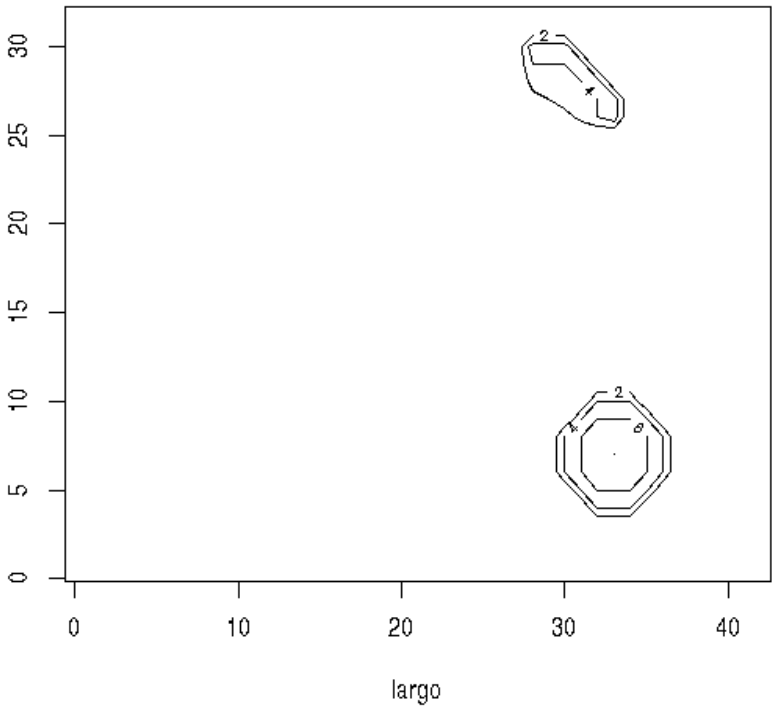


7.1.4. Contour

La función Contour permite observar cortes transversales al plano xy, es decir curvas de nivel. Usaremos como ejemplo la gráfica y datos anteriores. Es claro de observar el gráfico que si

produjeramos cortes a diferentes alturas de z, en algunos casos solo tocaríamos los dos picos que se hallan graficados, mientras que en otras alturas de z tocaríamos los picos y el cerco. Veamos como funciona la función contour()

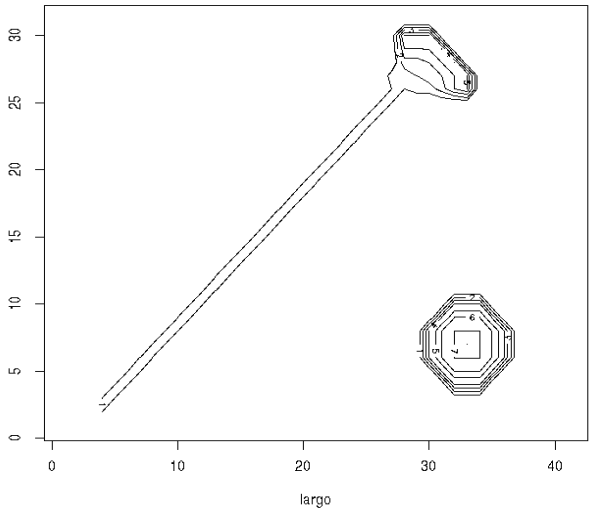
>
contour(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,nlevels=4,xlab="largo",ylab="ancho")



El gráfico nos indica que en una zona tenemos vegetación que supera los 4 m pero no los 6 metros, mientras que otros sectores superan los 6 m. El argumento nlevels=4 divide a los valores de z en 4 niveles: menores que 2, de 2 a 4, entre 4 y 6 y mayores que 6.

Cambiemos un argumento

contour(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,nlevels=6,xlab="largo",ylab="ancho")

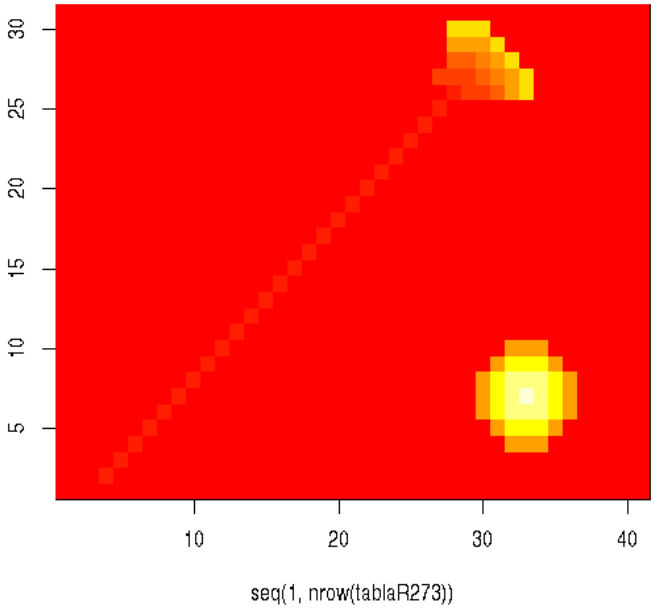


Los valores quedaron divididos en 6 intervalos.

7.1.5. Image

La funciom image(), permite obtener un gráfico de dos dimensiones, pero donde la tercer dimensión se observa por una graduación de color. En la gráfica siguiente vemos una aplicación

image(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273)



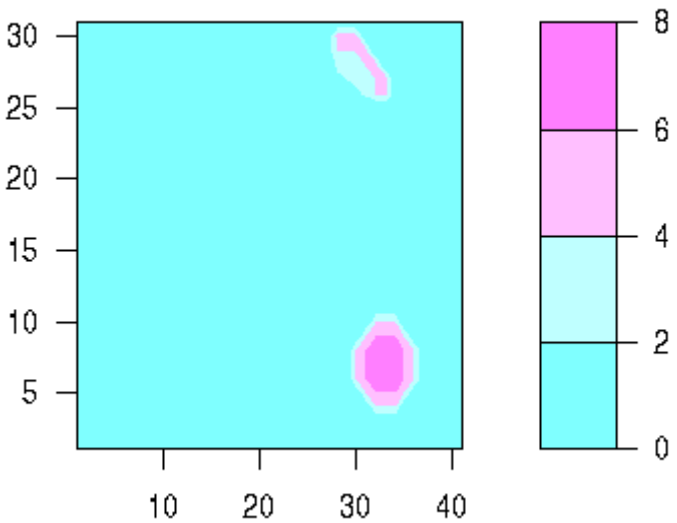
veamos un poco de manejo de color. Copie el recuadro y ejecute en consola como ya aprendimos.

```
for(i in 1:6){
image(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,col=heat.colors(i))
Sys.sleep(1)
}
```

7.1.6. Filled.contour

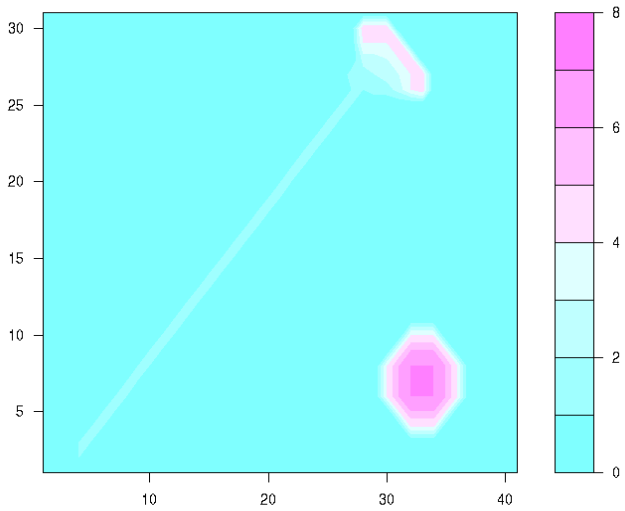
Esta función también permite ver en dos dimensiones valores de la tercera dimensión, la que se presenta con diversos colores respecto de su valor. Lo interesante de esta función es que dispone de un escala que nos permite interpretar el valor de la variable z, en este caso, de acuerdo al color que se presenta..

```
> filled.contour(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,nlevels=4)
```



Cláramente podemos ver en nuestro terreno dónde tenemos vegetación de más de 6 m, menos de 4, etc. Modificamos el argumento nlevels.

```
> filled.contour(x=seq(1,nrow(tablaR273)),y=seq(1,ncol(tablaR273)),z=tablaR273,nlevels=8)
```



¿En que sitio del terreno se ubicaría si deseara tener un poco de sombra? ¿Qué altura tiene allí la vegetación?

8. Clase 8

Vídeo: https://youtu.be/Z1kw1VV-7aI

Tabla de datos: http://hdl.handle.net/2133/10520

8.1. Gráficos de alta densidad de datos

Si bien no es común, podemos enfrentarnos a gráficos del tipo xy (scatterplot o dispersión), donde los pares de valores son tan numerosos que impiden identificar individuos o reconocer patrones porque los mismos están superpuestos.

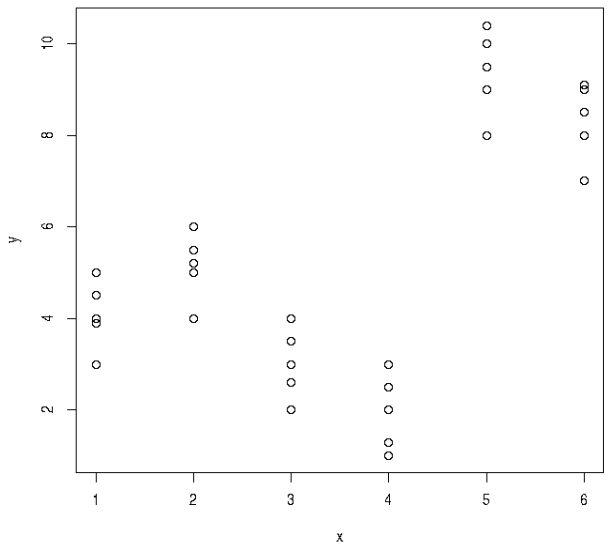
Veamos un caso sencillo, introduciendo los datos de la tablaR281 de la planilla de cálculo tablaR2-8.xls/ods

```
> tablaR281<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

haga la auditoría y grafique

```
> plot(tablaR281$y~tablaR281$x, xlab="x",ylab="y")
```

la tablaR281 tiene 30 datos y los mismos son visible e identificables en la gráfica.

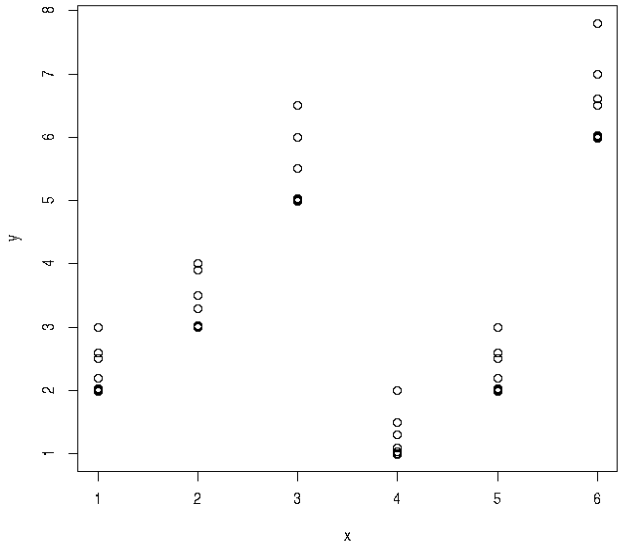


veamos ahora la gráfica de la tablaR282. Introduzcamos los datos

```
> tablaR282<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

y grafique

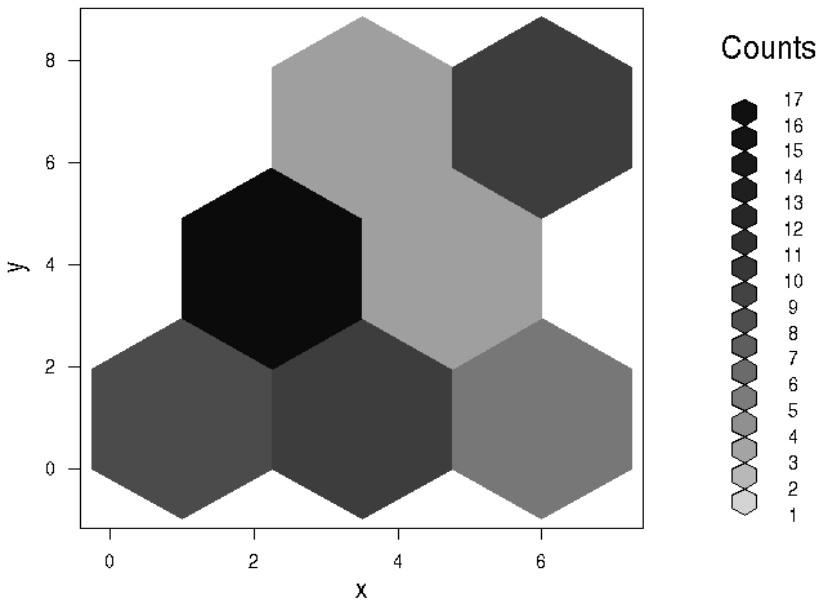
```
> plot(tablaR282$y~tablaR282$x, xlab="x",ylab="y")
```



La gráfica puede engañarnos ya que hay muchos puntos superpuestos. Parecería haber 30 unidades experimentales, mientras que la tablaR282 tiene 60. El problema es que los valores si bien no son iguales están superpuestos.

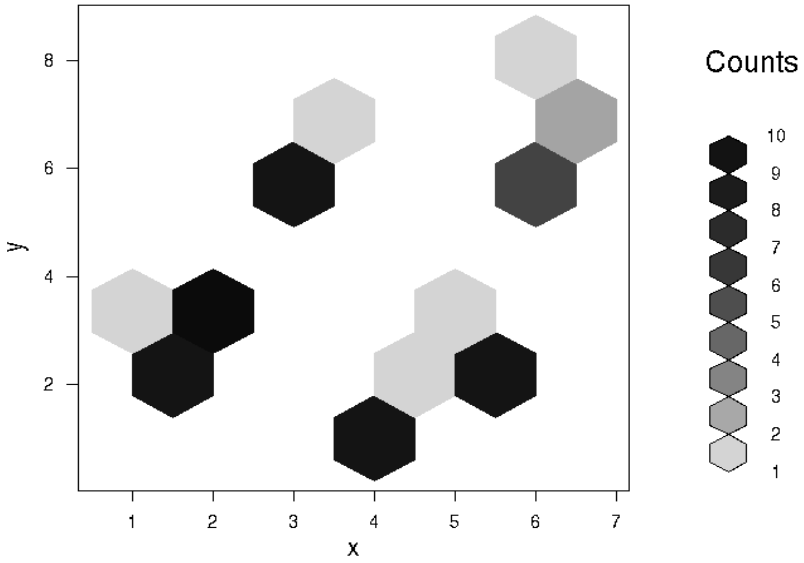
Para solucionar este problema tenemos la biblioteca hexbin. Descargue e instale la biblioteca hexbin y realizaremos un gráfico xy de alta densidad.

```
> plot(hexbin(tablaR282$x,tablaR282$y,xbins=2),xlab="x",ylab="y")
```



¿Cómo se interpreta este gráfico?. Dentro de cada hexágono hay tantos individuos como indica la escala. Quizás cambiando el argumento xbins puede obtener mejores resultados.

> plot(hexbin(tablaR282$x,tablaR282$y,xbins=5),xlab="x",ylab="y")



las zonas blancas no tiene puntos. Los hexágonos más oscuros nos indican alrededor de 10 datos dentro de cada uno de los hexágonos de color gris claro, contienen 1 punto. Los diferentes tonos de grises tienen diferentes cantidades que se leen en la escala Counts, de la derecha

8.1.1. Matrices de scatterplots

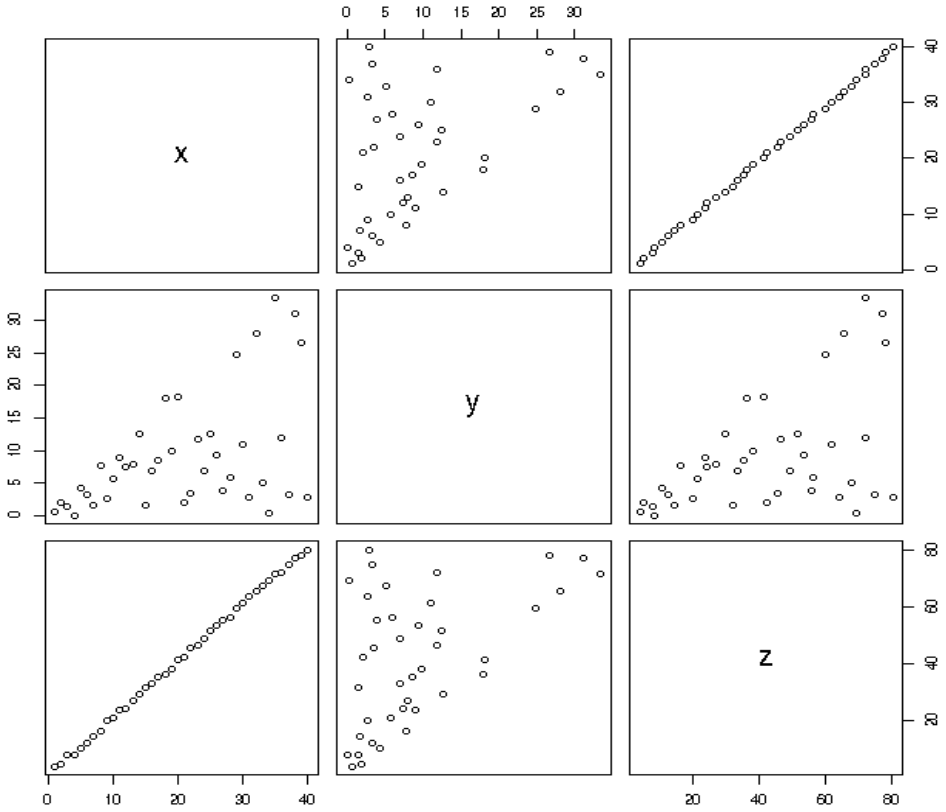
Esta gráfica es un tipo de gráfico múltiple y permite ver varias variables simultáneamente. Si bien puede graficarse cualquier número de variables, números muy elevados (mayor a 5 o 6) de variables hace casi imposible sacar conclusiones.

Introduzcamos y veamos la tablaR283

> tablaR283<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")

Veamos una gráfica que contenga la representación de todas las variables contra todas. Como la tabla tiene 3 variables: x, y, z, tendremos 9 gráficas.

> plot(tablaR283)



¿Cómo se interpreta?

La primer fila de la gráfica tienen a la variable x como eje horizontal, la segunda línea de gráfica tiene la variable y como eje horizontal y la tercera línea los valores de z.

La primer columna de gráficas tiene los valores de x como eje vertical, la segunda los valores de y, la tercera los valores de z.

La diagonal no contiene gráfica ya que es x en función de x, y en función de y , etc.

Analicemos una parte. La primer gráfica de la izquierda de la segunda fila está mostrando la relación entre x e y , con la variable y en las abscisas y la variable x en las ordenada. Básicamente es la misma gráfica que la que ocupa el segundo lugar de la primer fila, salvo que en ésta la variable x es la abscisa e y la ordenada.

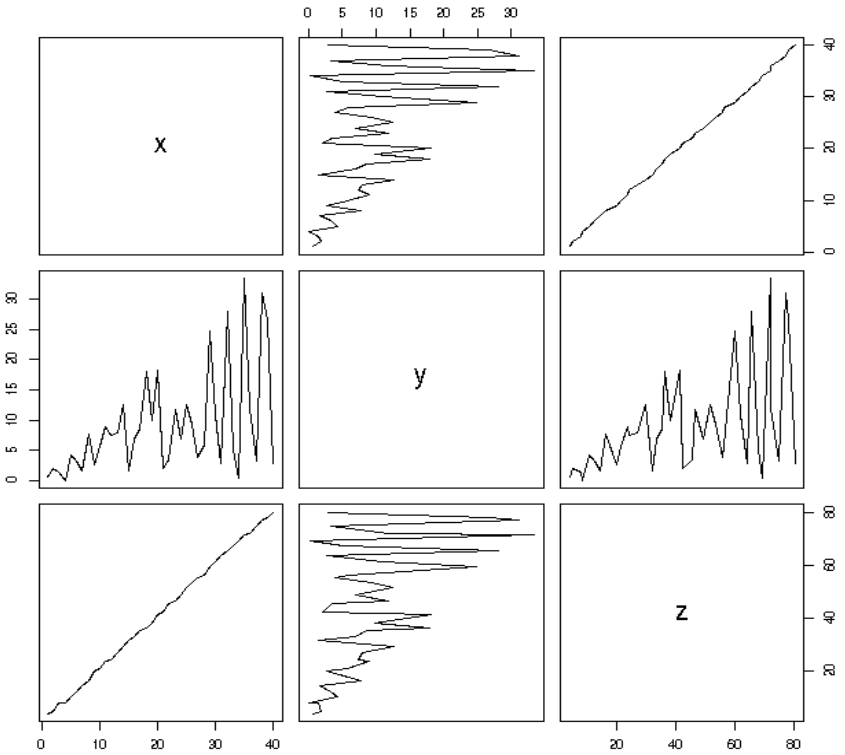
Es decir que básicamente debemos mirar de la diagonal con los valores x, y, z para arriba o para abajo, ya que son simétrica, cambiando la posición de los ejes. Esto ayuda a la búsqueda de relaciones y asociaciones entre variables.

La función pairs() realiza la misma gráfica

```
> pairs(tablaR283)
```

pero podemos darles argumentos.

```
> pairs(tablaR283,panel="lines")
```



8.1.2. Bandplot

Este tipo de gráficos utiliza library gplots.

> library(gplots)

Permite graficar valores de series de tiempo, es decir valores que se obtienen o generan a lo largo del tiempo. En el ejemplo veremos registros de temperatura a lo largo de cuatro meses

Introduzcamos la hoja tablaR284 de la planilla tablaR2-8.

Permite ver los valores, la media acumulada a lo largo del tiempo y 1 y 2 desvío estándar. Gráfica ideal para el control de series de tiempo.

> tablaR284<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")

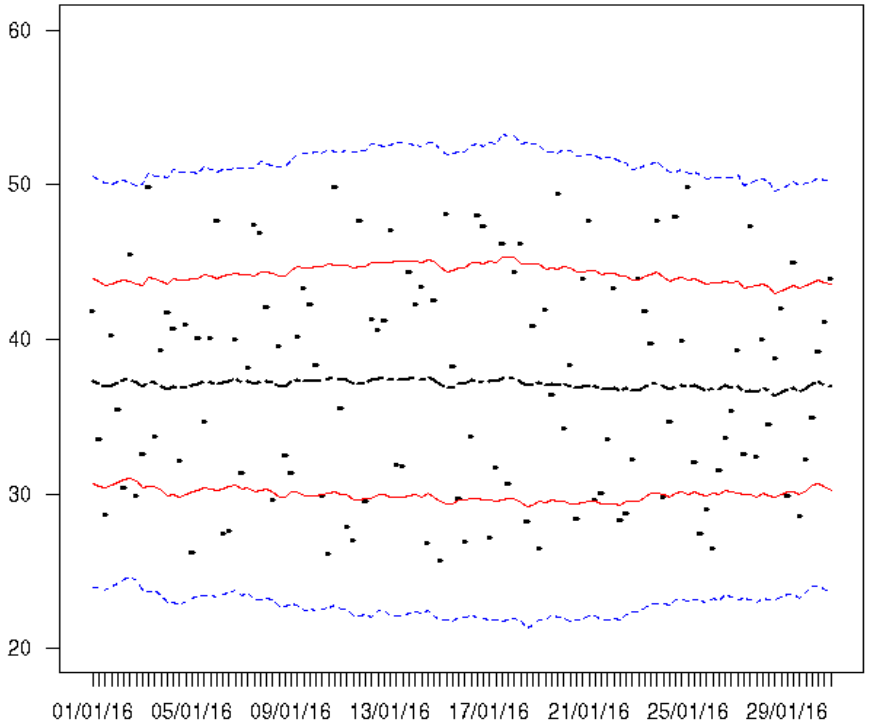
>
bandplot(tablaR284$fecha,tablaR284$temperatura,sd.col=c("blue","red","black","red","blue"),sd.lty=c(2,1,6,1,2),sd.lwd=c(1,1,2,1,1),pch=1,col="black",cex=0.8,las=1,ylim=c(20,60))

En la gráfica además de los puntos vemos líneas. La línea del medio (en color negro) representa a la media, las líneas rojas representan las medias más y menos un SD (+1SD, -1SD). Las líneas azules representan las medias más y menos 2 SD (+2SD, -2SD).

sd.col=establece los colores para -2SD,-1SD, media, +1SD,+2SD

sd.lty= establece el tipo de línea para -2SD,-1SD, media, +1SD,+2SD

sd.lwd= establece el grosor de línea para -2SD,-1SD, media, +1SD,+2SD



Se le pueden agregar leyendas para indicar los detalles. A tal fin se utilizan los códigos conocidos para detalles como títulos, ejes, leyendas, etc.

8.1.3. Barplot con desvíos

Si bien ya hemos visto gráficos sencillos con barras y veremos más aun cuando manejemos scripts, la biblioteca gplots tiene una función sencilla (barplot2) que hace gráficas de barras sencillas en las que se pueden incluir fácilmente las medias, desvíos estándar, intervalos de confianza, etc, combinándola con el uso de la función tapply.

Introduzcamos la hoja tablaR285

```
> tablaR285<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
```

dado que la columna calcemia tiene excesivos decimales redondeamos a dos decimales

```
> tablaR285$calcemia<-round(tablaR285$calcemia,digits=2)
```

```
> tablaR285
  tratamiento calcemia
1           c    10.50
2           c    10.47
3           c    10.58
4           c    10.10
5           c    10.35
```

6	c	10.15
7	c	10.73
8	c	10.81
9	c	10.31
10	t	12.49
11	t	12.60
12	t	12.01
13	t	11.46
14	t	12.02
15	t	12.63
16	t	11.74
17	t	12.10
18	t	11.30
19	t	11.10

con la función tapply calculamos media y sd para cada tratamiento y las asignamos a objetos.

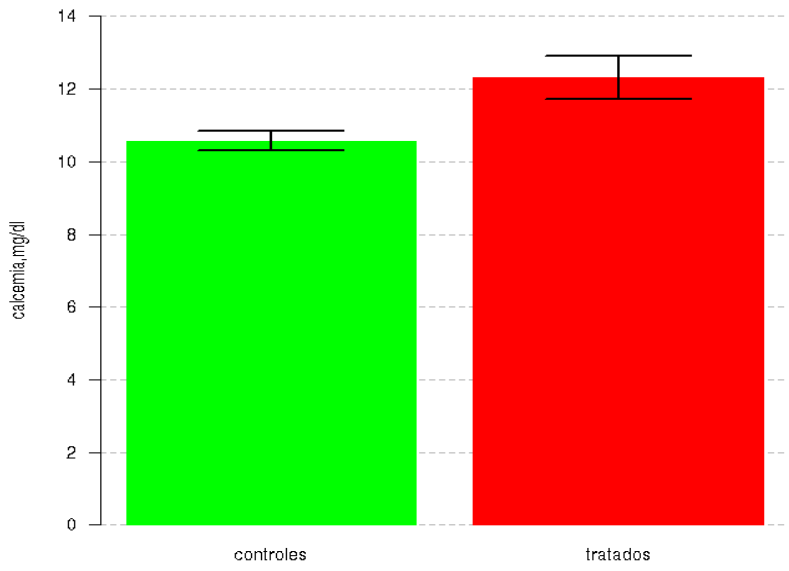
```
> medias<-tapply(tablaR285$calcemia,tablaR285$tratamiento,mean)
```

```
> sd<-tapply(tablaR285$calcemia,tablaR285$tratamiento,sd)
```

los nombres de los objetos medias y sd deben coincidir con los nombres (resaltados en amarillo) del código siguiente

y graficamos con barplot2()

```
>
barplot2(medias,names.arg=c("controles","tratados"),ylim=c(0,14),col=c("green","red"),border=c("green","red"),ylab="calcemia,mg/dl",las=1,plot.ci=TRUE,ci.l=c(medias-sd),ci.u=c(medias+sd),ci.color="black",ci.lty=1,ci.lwd=2,plot.grid=TRUE,grid.inc=8,grid.lty="dashed",grid.lwd=0.5,grid.col="gray70")
```



Si lo desea podría graficas con dos SD, sin SD o bien intervalo de confianza. Solo debe modificar

los argumentos ci.l y ci.u.

8.1.4. Scatterplot ampliado

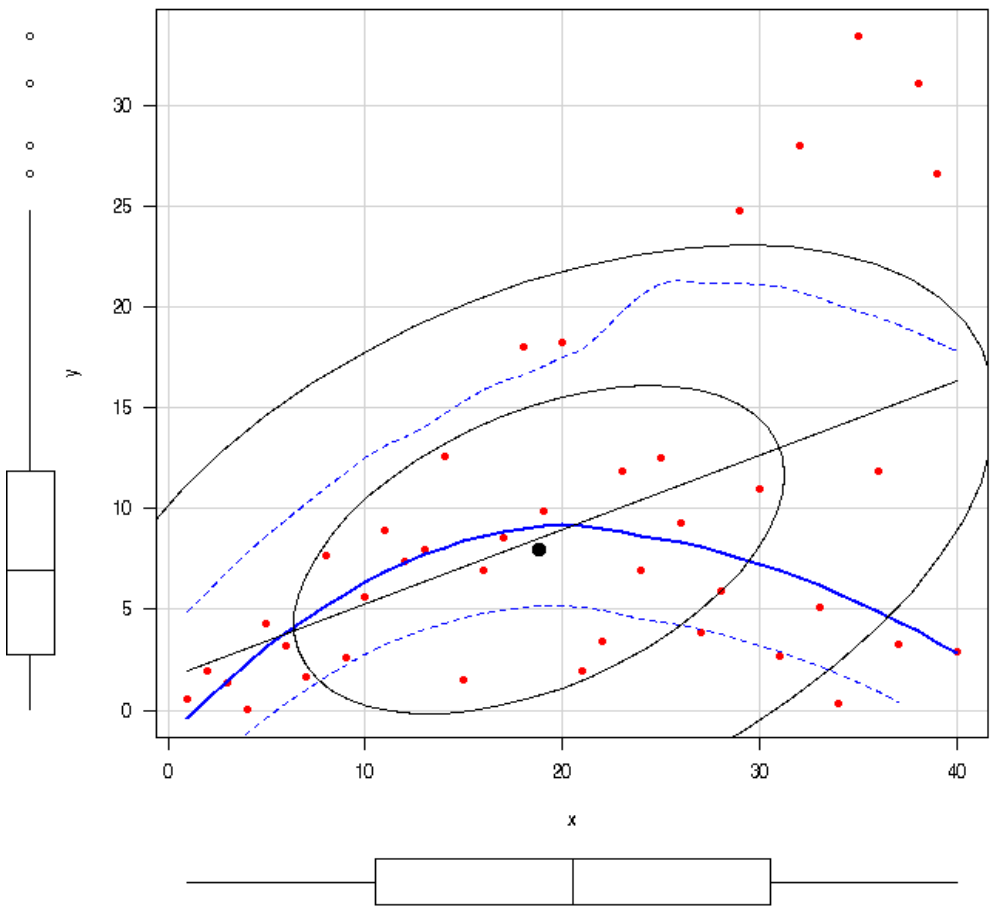
Este tipo de gráficos scatterplot requiere la biblioteca car.

```
> library(car)
```

Esta gráfica es interesante ya que a la hora de análisis permite ver muchas cosas simultáneamente.

Utilizaremos el data.frame tablaR283 y graficaremos y en función de x.

```
>
scatterplot(y~x,pch=19,col=c("black","blue","red"),data=tablaR283,ellipse=T,levels=c(0.5,0.9),las
=1,grid=T,span=1)
```



La gráfica muestra en los márgenes los rangos, medianas y cuartilos de cada variable.

En azul la curva de ajuste de los datos.

En azul con puntos: la curva de ajuste con un intervalo de confianza del 95%

La recta de regresión lineal es mostrada con una línea negra de trazo continuo.

Las dos elipses. La interna agrupa el 50 %de los datos y la más amplia el 90%

9. Clase 9

Vídeo: https://youtu.be/OOG7VeUzYBk

Tabla de datos: http://hdl.handle.net/2133/10521

9.1. Parámetros gráficos

Los parámetros de un gráfico son valores que dan las características a un gráfico. Estos se hallan en un archivo. Algunos se pueden modificar durante la construcción del gráfico, por ejemplo con la función plot(), barplot(), etc, tal es el caso de cex, pch, etc. Otros solo pueden modificarse cambiando el parámetro a través de un mecanismo que veremos en esta clase y otros no son modificables.

9.1.1. Conocer los parámetros gráficos

help(par)
Allí encontrará la descripción de cada uno. La modificación de los parámetros es solo útil en casos especiales, sin embargo requiere mucho tiempo y esfuerzo. Los valores por defecto de los parámetros generan gráficos de buena calidad e información.

9.1.2. Conocer el valor de los parámetros

Para conocer los valores de los parámetros gráficos que tiene su instalación de R escriba
> par()

a continuación se listan los valores por defecto. Cada uno de ellos da una característica del gráfico. Por ejemplo el primero $xlog, puede tomar dos valores TRUE o FALSE. En tales casos el gráfico tendrá el eje x en escala logarítmica o no.

```
$xlog
[1] FALSE

$ylog
[1] FALSE

$adj
[1] 0.5

$ann
[1] TRUE

$ask
[1] FALSE

$bg
[1] "transparent"

$bty
[1] "o"

$cex
[1] 1
```

```
$cex.axis
[1] 1

$cex.lab
[1] 1

$cex.main
[1] 1.2

$cex.sub
[1] 1

$cin
[1] 0.15 0.20

$col
[1] "black"

$col.axis
[1] "black"

$col.lab
[1] "black"

$col.main
[1] "black"

$col.sub
[1] "black"

$cra
[1] 14.39646 19.21892

$crt
[1] 0

$csi
[1] 0.2

$cxy
[1] 124.0346513   0.3353315

$din
[1] 6.991303 6.993110

$err
[1] 0

$family
[1] ""
```

```
$fg
[1] "black"

$fig
[1] 0 1 0 1

$fin
[1] 6.991303 6.993110

$font
[1] 1

$font.axis
[1] 1

$font.lab
[1] 1

$font.main
[1] 2

$font.sub
[1] 1

$lab
[1] 5 5 7

$las
[1] 0

$lend
[1] "round"

$lheight
[1] 1

$ljoin
[1] "round"

$lmitre
[1] 10

$lty
[1] "solid"

$lwd
[1] 1

$mai
[1] 1.02 0.82 0.82 1.60
```

```
$mar
[1] 5.1 4.1 4.1 8.0

$mex
[1] 1

$mfcol
[1] 1 1

$mfg
[1] 1 1 1 1

$mfrow
[1] 1 1

$mgp
[1] 3 1 0

$mkh
[1] 0.001

$new
[1] FALSE

$oma
[1] 0 0 0 0

$omd
[1] 0 1 0 1

$omi
[1] 0 0 0 0

$page
[1] TRUE

$pch
[1] 1

$pin
[1] 4.571303 5.153110

 $plt
[1] 0.1172886 0.7711442 0.1458578 0.8827417

$ps
[1] 12

$pty
[1] "m"
```

```
$smo
[1] 1

$srt
[1] 0

$tck
[1] NA

$tcl
[1] -0.5

$usr
[1] 27360.00 31140.00    27.68    36.32

$xaxp
[1] 27500 31000     7

$xaxs
[1] "r"

$xaxt
[1] "s"

$xpd
[1] TRUE

$yaxp
[1] 28 36  4

$yaxs
[1] "r"

$yaxt
[1] "s"

$ylbias
[1] 0.2
```

9.1.3. Qué significan algunos de ellos?

A continuación se da la explicación de algunos de ellos y los valores que pueden tomar. Para acceder a un parámetro escriba, por ejemplo

```
> par('adj')
```

le mostrará el valor que tiene asignado

```
[1] 0.5
```

para conocer todos los valores escriba

> help(par)

y busque el parámetro en cuestión. Para adj, surje de la lectura que puede tomar 3 valores y cada uno de ellos tiene una finalidad

adj= 0, 0.5 o 1

Se puede usar con los ejes, mtext y title

adj=0 coloca los títulos de los ejes próximos al origen

adj=0.5 coloca los títulos al medio del eje

ajd=1 coloca los títulos en la punta de los ejes

bty

Especifica el tipo de recuadro del gráfico

“o”= default. recuadro completo, donde dos de sus lados son los ejes.

“l”= solo los ejes

“7” recuadra pero deja los ejes separados del recuadro

“c” cierra el gráfico por arriba pero no por la derecha

“u” cierra el gráfico por la derecha pero no por arriba

“]” cierra el gráfico por ambos lados pero deja despegado el lado superior del eje y

ann= T o F
T: pone los títulos de los ejes
F: no los pone

ask= T o F
si esta en T, cuando se coloca el código de un gráfico y se da enter, lo muestra directamente, Si está en F, al dar enter aparece un mensaje que dice <enter> para ver el próximo mensaje.
Solo se puede cambiar modificando escribiendo la siguiente línea de comando
par(ask= T o F)

$xlog
fija a los ejes x o y en escala logarítmica
para cambiarlo, se debe incluir dentro del código del gráfico
> plot(variable1~variable2,data=datos,log=”x”)
el par log no queda en T, hay que fijarlo cada vez
si se desean cambiar los dos ejes simultáneamente
> plot(variable1~variable2,data=datos,log=”xy)”

$ylog
para saber el valor del parámetro ejecutar
para cambiarlo, se debe incluir dentro del código del gráfico
> plot(variable1~variable2,data=datos,log=”y”)
el par log no queda en T, hay que fijarlo cada vez

$bg
da el color de fondo de toda el área del gráfico

para cambiarlo hay que hacerlo antes de graficar
par(bg=”blue”) #ojo que queda en azul, el default es “transparent”

$cex
cambia el tamaño del signo utilizado para representar el par de valores en el gráfico

$cex.axis
cambia el tamaño de los números de los ejes

$cex.lab
cambia el tamaño de los nombres de los ejes

$cex.main
cambia el tamaño del titulo principal del gráfico (main)

$cex.sub
cambia el tamaño del texto que actúa como pie de gráfico (sub)

$cin
[1] 0.15 0.20
este parámetro no puede ser fijado es un R.O. (read only argument)

$col
especifica el color del punto del par de valores

$col.axis
especifica el color de los números de los ejes

$col.lab
especifica el color de los nombres de los ejes

$col.main
especifica el color del título principal

$col.sub
especifica el color del texto que actúa como pie de figura

$cra
[1] 12.46466 16.81655
este parámetro no puede ser fijado es un R.O. (read only argument)

$csi
[1] 0.2
este parámetro no puede ser fijado es un R.O. (read only argument)

$cxy
[1] 0.3026788 0.2278035
este parámetro no puede ser fijado es un R.O. (read only argument)

$din

[1] 4.344281 4.305282
este parámetro no puede ser fijado es un R.O. (read only argument)

$err
[1] 0
no esta implementado en R aun

$family
fija el tipo de letra
[1] "" valor default
R trae cuatro tipos básicos
Courir New = mono
Times New Roman = serif
Arial = sans (valor default) que es equivalente a family=””
Symbol = symbol
si deseo cambiar el tipo de todo el gráfico
par(family=”serif”)
plot(..... #me hará el gráfico en el tipo deseado
pero si deseo otros puedo usar la biblioteca Hershey
> par(family="HersheyGothicGerman")
plot(..... # me hará el gráfico en el tipo elegido. Para variedad ver la tabla siguiente

`"HersheySerif"` plain, bold, italic, bold-italic
`"HersheySans"` plain, bold, italic, bold-italic
`"HersheyScript"` plain, bold
`"HersheyGothicEnglish"` plain
`"HersheyGothicGerman"` plain
`"HersheyGothicItalian"` plain
`"HersheySymbol"` plain, bold, italic, bold-italic
`"HersheySansSymbol"` plain, italic

Se pueden colocar estos parámetros directamente dentro del gráfico. Ej:
> plot(variable1~variable2,data=datos)
> text(5,0.5,"A. Rigalli",srt=30,family="HersheyGothicGerman")

$fg
cambia el color de ejes, recuadros, lineas, etc, pero no de labels, números y títulos
par(fg="red") #el default es “black”

$fig
[1] 0 1 0 1
Parámetro que define el tamaño del gráfico dentro del área de dibujo. Los valores están entre 0 y 1. Es un vector c(posición del lado izquierdo, lado derecho, lado inferior, lado superior)
> par(fig=c(0,1,0,1)) # Cubre todo el área de dibujo
Si uno desea un gráfico que ocupe el 25% del área ubicado el en vértice inferior izquierdo, se tendría que especificar:
> par(fig=c(0,0.5,0,0.5))
plot(....)

$fin

Parámetro que da la dimensión (alto y ancho) de la zona de dibujo. Se debe definir previo a ejecutar el gráfico, y no queda almacenado para los siguientes gráficos.
> par("fin")
[1] 6.991765 6.993110

$font
fija el tipo de letra: normal, negrita, cursiva
[1] 1

$font.axis
fija el tamaño de los números de los ejes
[1] 1

$font.lab
fija el tamaño de los rótulos de los ejes
[1] 1

$font.main
fija el tamaño del título del gráfico
[1] 2

$font.sub
fija el tamaño del texto pie de gráfico
[1] 1

$lab
[1] 5 5 7

$las
fija la orientación de los números de los ejes respecto de los ejes.
[1] 0

$lend
[1] "round"

$lheight
[1] 1

$ljoin
[1] "round"

$lmitre
[1] 10

$lty
fija el tipo de linea
[1] "solid"

$lwd
fija el grosor de la línea

[1] 1

$mai
[1] 1.02 0.82 0.82 0.42

$mar
[1] 5.1 4.1 4.1 2.1
este parámetro permite modificar los margenes de la figura.
El primer número es el margen inferior
El segundo: izquierdo, el tercero: superior y el cuarto: derecho.
para modificarlo
par(mar=c(5.1 4.1 4.1 2.1))
para poder escribir en los margenes se debe incluir xpd=T
par(mar=c(5.1 4.1 4.1 2.1), xpd=T)
para escribir por ejemplo una leyenda en el margen derecho, se debe especificar ciertas cosas (posicion: topright, bottom, etc, inset=c(-0.5, 0)
El primer numero de inset da la posición horizontal. el valor 0 indica el vértice derecho del gráfico, un valor negativo mueve fuera del gráfico la leyenda. El segundo numero indica la posición vertical. el valor 0 indica el vértice superior
por ejemplo para poner una leyenda bajo de la gráfica
par(mar=c(8 4.1 4.1 2.1))
> legend("bottom",inset=c(0,-0.5),pch=c(3,5),legend=c("control","tratado"),title="Tratamiento",horiz=T)

$mex
[1] 1

$mfcol
[1] 1 1

$mfg
[1] 1 1 1 1

$mfrow
[1] 1 1

$mgp
[1] 3 1 0

$mkh
[1] 0.001

$new
[1] FALSE

$oma
[1] 0 0 0 0

$omd
[1] 0 1 0 1

$omi
[1] 0 0 0 0

$pch
[1] 1

$pin
[1] 3.104281 2.465282

$plt
[1] 0.1887539 0.9033212 0.2369183 0.8095363

$ps
[1] 12

$pty
[1] "m"

$smo
[1] 1

$srt
[1] 0

$tck
Indica posición y tamaño del tick. Si es positivo (ej 0.1) grafica los ticks hacia arriba del x y la derecha del y,siendo el tick el 10% del tamaño del gráfico. Si es negativo (ej -0.1) grafica los ticks hacia abajo del x y la izquierda del y,siendo el tick el 10% del tamaño del gráfico.
Si vale NA, tcl = -0.5, y no cambia el tamaño de los ticks aun cuando modifiquemos tcl.
tck por defecto es NA

$tcl
Indica posición y tamaño del tick. Si es positivo (ej 0.1) grafica los ticks hacia arriba del x y la derecha del y,siendo el tick el 10% del tamaño de una linea de texto. Si es negativo (ej -0.1) grafica los ticks hacia abajo del x y la izquierda del y,siendo el tick el 10% del tamaño del gráfico.
Si vale NA, tcl = -0.01.

$usr
es un vector donde los dos primeros números están muy cercanos a los valores extremos de la variable horizontal, y los dos últimos muy cercanos a los extremos de la variable vertical. Cambiando los parámetros no ha cambiado la gráfica. Quizás dependa de otro parámetro.
> par(usr=c(0,7,0,3))

$xaxp
es un vector de tres números, el primero indica el tick extremo izquierdo del eje, n2: el tick extremo derecho del eje x, n3 indica el numero de intervalos
> par(xaxp=c(0,6,6))
Esto funciona si el parámetro xlog=F, si xlog=T es diferente.

no anduvo quizas por otro parámetro que falta modificar

$xaxs= “r”, “i”, “s”
r: deja un espacio entre los puntos extremos del rango de valores y los ejes y .
i: no deja un espacio entre los puntos extremos del rango de valores y los ejes.

> par(xaxs="r")
$xaxt= “s” o “n”
s= pone los números y los tick del eje x.
n= no los pone
> par(xaxt="n")

$xpd= T or F
Cuando xpd es T (par(xpd=T)), permite trabajar fuera del recuadro del gráfico
la gráfica originariamente no tiene grilla y en caso de quererla se debería colocar. primero se hace el gráfico y luego se ejecuta el comando para colocar la grilla
grilla(nx=NULL, ny=NULL.....)

$yaxp=c(n1,n2,n3)
es un vector de tres números, el primero indica el tick extremo inferior del eje, n2: el tick extremo superior del eje y, n3 indica el numero de intervalos
Esto funciona si el parámetro ylog=F, si ylog=T es diferente.
no anduvo quizás por otro parámetro que falta modificar

$yaxs= “r”, “i”, “s”
r: deja un espacio entre los puntos extremos del rango de valores y los ejes x. Funciona con plot, boxplot
i: no deja un espacio entre los puntos extremos del rango de valores y los ejes. funciona con plot, boxplot

$yaxt= “s” o “n”
“s”= pone los números y los tick del eje y, “n”= no los pone (entre comillas)
> par(yaxt="n")

ylbias=número
el numero indica la separación entre ticks y números de la escala. Es poco práctica (al menos hasta ahora) ya que números positivos acercan números del eje x al eje, pero alejan los números del eje y al eje. Números negativos lo hacen en sentido contrario. El valor 0 es la colocación clásica
> par(ylbias=0)

9.1.4. Uso de parámetros

Veremos algunos detalles y uso de los parámetros y quedará para usted profundizar en caso que lo necesite. El manejo completo de todos los parámetros llevaría varios meses. El aprendizaje del manejo de algunos de ellos lo habilitará para trabajar con los otros cuando los necesite.

Si se va a modificar algún parámetro es conveniente guardar los parámetros que se hallan por defecto. Para modificar algún parámetro, es conveniente guardar el objeto par en otro, que por ejemplo podemos llamar oldpar

> oldpar<-par()

en oldpar quedan guardados los parámetros por defecto. Al finalizar el trabajo restablezca los parámetros originales, salvo que el cambio realizado le sea beneficioso. Para restablecer los parámetros guardados en oldpar ejecute

>par(oldpar)

9.1.4.1 Modificar un parámetro.

Como regla general, la línea de comando es

par(nombre del parámetro=valor)

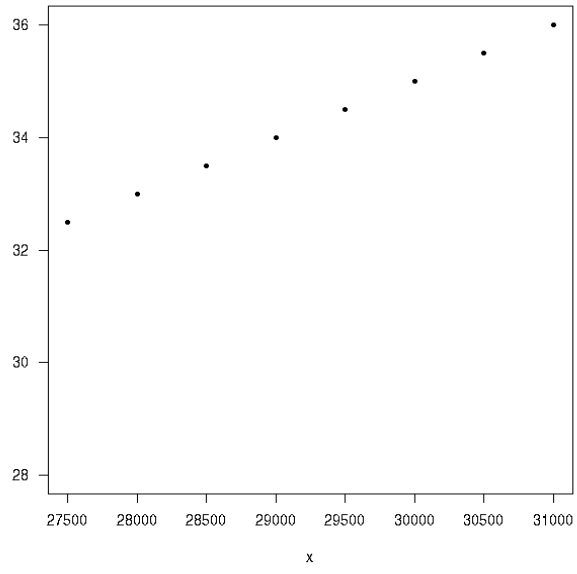
utilicemos el parámetro bg (que fija el color del fondo del gráfico)

vemos el valor del parámetro bg en la configuración actual

```
> par("bg")
[1] "transparent"
```

introduzcamos la hoja tablaR292 de la planilla de calculo tablaR2-9

```
> tablaR292<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
> tablaR292
      x    y     z
1 27500 32.5 29.25
2 28000 33.0 29.70
3 28500 33.5 30.15
4 29000 34.0 30.60
5 29500 34.5 31.05
6 30000 35.0 31.50
7 30500 35.5 31.95
8 31000 36.0 32.40
> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",ylab="",ylim=c(28,36))
```



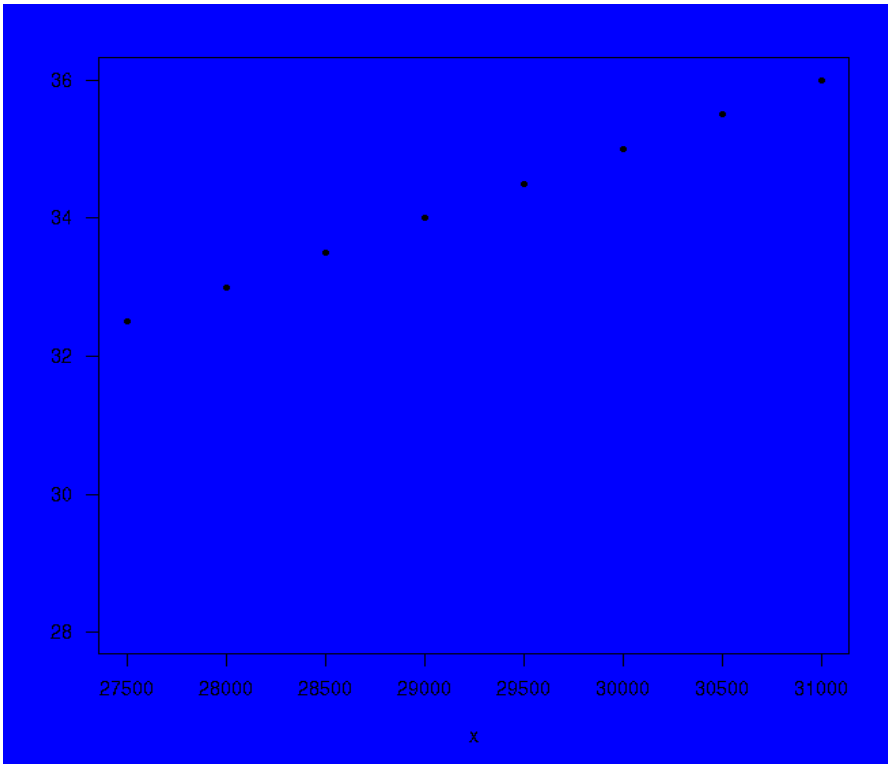
primero

> oldpar<-par()

cambiamos el parámetro

par(bg="blue")

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",ylab="",ylim=c(28,36))



Si deseamos restablecer los valores por defecto de los parámetros utilizamos el objeto oldpar que hemos creado y tiene todos los valores originales

> par(oldpar)

si ahora comprobamos nuevamente el valor del parámetro bg

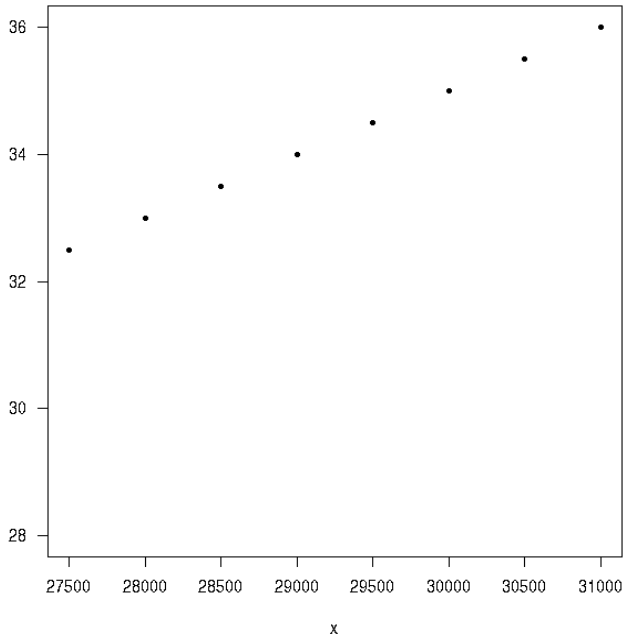
> par("bg")

[1] "transparent"

Si hacemos nuevamente la gráfica

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",ylab="",ylim=c(28,36))

obtendremos



también se pueden fijarse y restablecer los valores por defecto de los parámetros con el comando correspondiente a un parámetro en cuestión.

Por ejemplo comprobemos que valor tiene el parámetro bg en este caso

> par("bg")

[1] "transparent"

el recuadro del gráfico es transparente

Ahora lo pasamos a azul

> par(bg="blue")

Comprobamos si está en azul

> par("bg")

[1] "blue"

Con el siguiente comando restablecemos el valor

> dev.off()

null device

1

> par("bg")

[1] "transparent"

El cambio del valor del valor de un parámetro se puede hacer también en la función que utilizamos para graficar. En la misma tenemos argumentos que se modifican directamente, como por ejemplo pch. Para aquellos parámetros que no se puede hacer directamente en el código se debe utilizar una escritura que indique la modificación del parámetro, como se ve resaltado en amarillo.

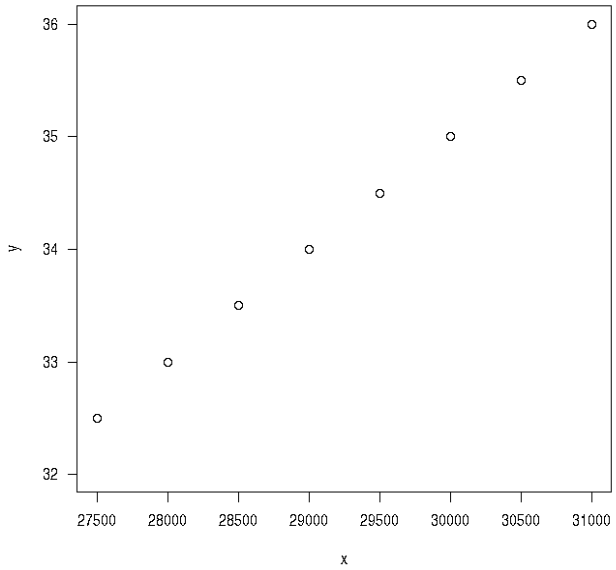
>
plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",ylab="",ylim=c(28,36),bg=par(bg="blue"))

Esto cambiará el valor del parámetro, que puede o no ser restablecido, según las necesidades.

9.2. Rotar rótulos de ejes

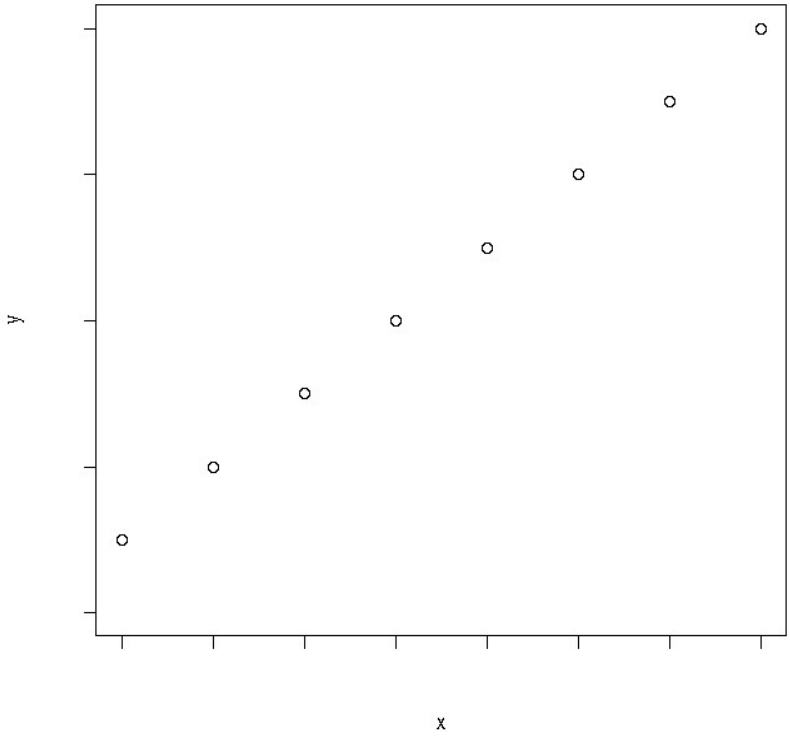
En algunas situaciones requerimos rotar los rótulos de los ejes para tener una mejor presentación de los resultados. No es fácil con R, pero si posible.
Introduzcamos los datos de la hoja tablaR291 de la planilla de cálculo tablaR2-9

```
> tablaR291<-read.table("clipboard",header=T,sep="\t",dec=",",encoding="latin1")

> tablaR291
     x    y
1 27500 32.5
2 28000 33.0
3 28500 33.5
4 29000 34.0
5 29500 34.5
6 30000 35.0
7 30500 35.5
8 31000 36.0
> plot(y~x,data=tablaR291,xlab="x",ylab="y",las=1,xlim=c(27500,31000),ylim=c(32,36))
```



supongamos que queremos rotar los rótulos del eje x de manera que no queden horizontales. Realizamos nuevamente la gráfica, pero le eliminamos los rótulos de los ejes

>
plot(y~x,data=tablaR291,xlab="x",ylab="y",las=1,labels=F,xlim=c(27500,31000),ylim=c(32,36))



creamos dos vectores con los números correspondientes a los ticks de cada eje.

eje x

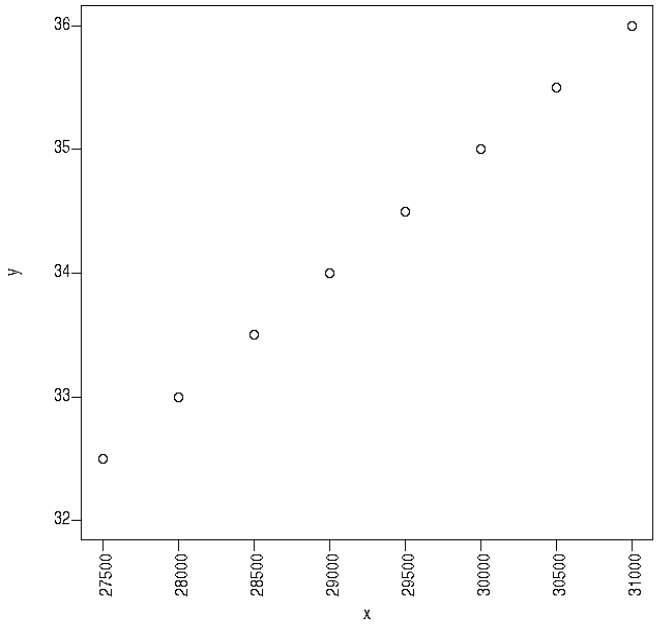
```
> lablistx<-seq(27500,31000,by=500)
> lablistx
[1] 27500 28000 28500 29000 29500 30000 30500 31000
```

y para el eje y

```
> lablisty<-seq(32,36,by=1)
> lablisty
[1] 32 33 34 35 36
```

colocamos los rótulos en su posición, pero para el eje x lo rotamos 90 grados con el argumento srt.

```
> text(x=seq(27500,31000,by=500),par("usr")[3]-0.2,labels=lablistx,srt=90,pos=1,xpd=TRUE)
> text(y=seq(32,36,by=1),par("usr")[1]-2,labels=lablisty,pos=2,xpd=TRUE)
```



xpd= T #permite escribir fuera del gráfico.

Esta opción permite no solo colocar en 90 grados, sino en el angúlo que lo desee. Cuando solo se requiere ángulos de 90 grados, recuerde que el argumento las permite esto

las=0 genera rótulo paralelos a los ejes

las=1 genera rótulos horizontales

las=2 genera rótulos perpendiculares a los ejes

las=3 genera rótulos verticales

para generar la gráfica anterior necesitaríamos entonces

```
> plot(y~x,data=tablaR291,xlab="x",ylab="y",las=2,xlim=c(27500,31000),ylim=c(32,36))
```

9.3. Legenda fuera del gráfico

Utilizaremos el data.frame tablaR292

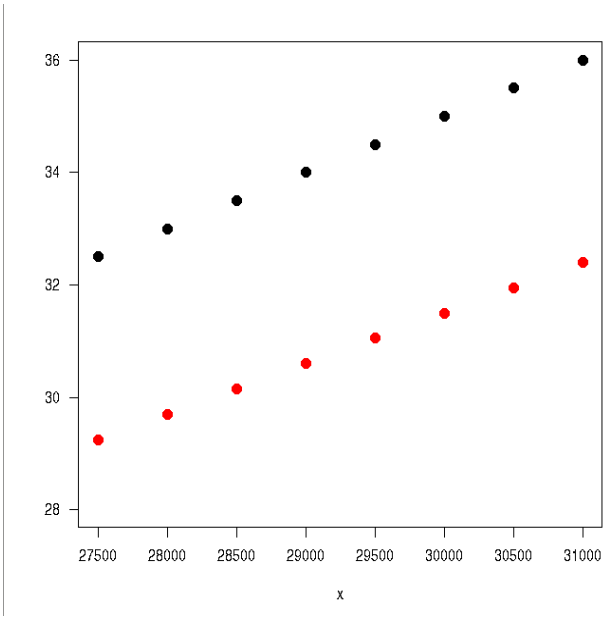
```
> tablaR292
      x    y     z
1 27500 32.5 29.25
2 28000 33.0 29.70
3 28500 33.5 30.15
4 29000 34.0 30.60
5 29500 34.5 31.05
6 30000 35.0 31.50
7 30500 35.5 31.95
8 31000 36.0 32.40
```

graficamos los datos y en función de x

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",cex=2,ylab="",ylim=c(28,36))

agregamos los datos z en función de x

> points(z~x,data=tablaR292,col="red",pch=20,cex=2)



deseamos colocar la leyenda debajo

analicemos el parámetro "mar", que fija los márgenes del gráfico por fuera de los ejes

> par("mar")

[1] 5.1 4.1 4.1 2.1

estos son los valores de los márgenes por debajo del eje x, a la izquierda del y, arriba y a la derecha.

Guardamos los parámetros originales

> oldpar<-par()

fijamos nuevos márgenes

> par(mar=c(3,3,3,3))

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",cex=2,ylab="",ylim=c(28,36))

vemos que tenemos a cada uno de los cuatro lados del gráfico igual distancia hasta el rectángulo que lo contiene. Fijamos ahora

> par(mar=c(7,3,3,3))

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",cex=2,ylab="",ylim=c(28,36))

hemos ganado espacio debajo. Ampliemos aun más el espacio

> par(mar=c(9,3,3,3))

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",cex=2,ylab="",ylim=c(28,36))

modifiquemos todos dejando un buen espacio abajo, izquierda y derecha y poco arriba

> par(mar=c(11,5,3,5))

graficamos la serie y en función de x

> plot(y~x,data=tablaR292,las=1,col="black",pch=20,xlab="x",cex=2,ylab="",ylim=c(28,36))

y luego la serie z en función de x

> points(z~x,data=tablaR292,col="red",pch=20,cex=2)

En definitiva el parámetro mar es un vector de cuatro números que indican los espacios que tiene la gráfica desde el borde de la figura a los ejes, de la siguiente manera:

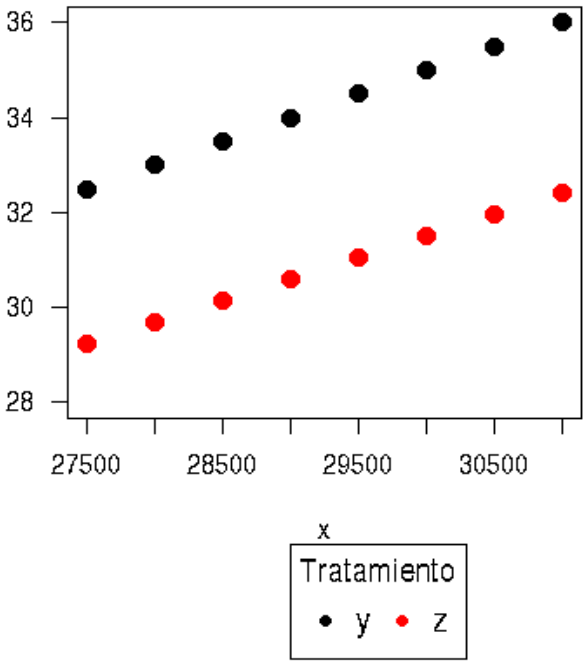
espacio inferior,espacio izquierdo, espacio superior, espacio derecho

Así hicimos un espacio debajo de la gráfica. Con el parámetro xpd en TRUE permitiremos acceder con textos a esos espacios. Si xpd=F no se podrán utilizar los espacios

> par(xpd=T)

para poder escribir fuera del gráfico

>
legend(29000,25,pch=c(20,20),cex=1.2,col=c(1,2),legend=c("y","z"),title="Tratamiento",horiz=T)



Por supuesto puede embellecer mucho aun la figura jugando ahora con argumentos y parámetros.

9.4. Gráficos dentro de gráficos

Muchas veces es de utilidad mostrar un gráfico con más de un resultado o forma de presentación. Para ello poder colocar un gráfico dentro de otro es un buen recurso, especialmente si el gráfico lo permite. Esto requiere utilizar un recurso bastante avanzado que es la modificación de parámetros, para lo cual estamos en condiciones de hacer

El parámetro new, que puede adoptar los valores T o F, permite o no graficar un gráfico sobre otro. Si su valor es T, luego de un gráfico podremos hacer otro sobre el mismo. En cambio si el valor de new es F, cada vez que ejecutemos un gráfico se cerrará el anterior, salvo que utilicemos recursos para gráficos múltiples.

También modificaremos el parámetro fig, que es un vector con cuatro números fig=c(n1,n2,n3,n4). los números n1 y n2 indican desde donde hasta donde se extiende el gráfico en el sentido horizontal dentro del área de dibujo. Lo default es n1=0, n2=1, es decir va de extremo a extremo. Para n3 y n4 los valores son n3=0, n4=1, es decir que en sentido vertical va de extremo a extremo.

Si cambiaramos el parámetro a valores

par(fig=c(0,0.5,0,0.5), obtendríamos un gráfico que ocupa un cuarto del área de dibujo y se hallará a la izquierda abajo. Pero si los valores fueran par(fig=c(0.5,1,0.5,1) también ocuparía un cuarto, pero sería en el vértice superior derecho. Entenderemos mejor esto con un ejemplo.

Graficaremos a continuación en un mismo gráfico los datos de la tablaR292, mostrando con

- la función plot a ***y*** en función de ***x***

- la función points a ***z*** en función de ***x***

- la función boxplot el rango, cuartilos y mediana de los valores de ***z*** e ***y***

Fijamos los parámetros mencionados en valores default

```
> par(new=F)

> par(fig=c(0,1,0,1))
```

o bien ejecutemos

```
dev.off()
```

graficamos y en función de x, fijando ylim adecuadamente, para que luego z en función de x aparezca en el gráfico, al utilizar la función points.

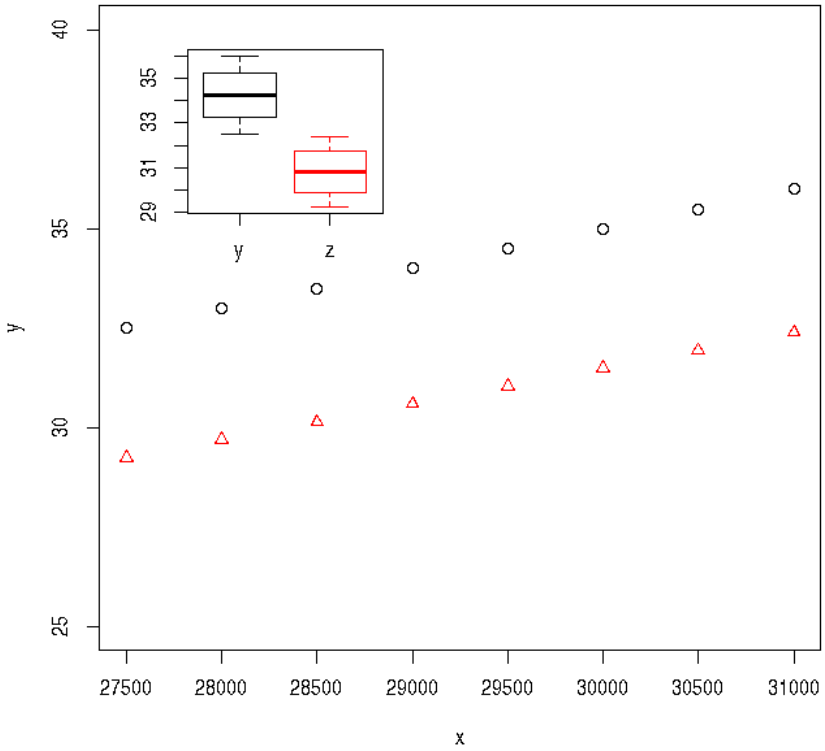
```
> plot(y~x,tablaR292,ylim=c(25,40))

> points(z~x,tablaR292,pch=2,col='red')
```

ahora cambiamos los parámetros de manera que se pueda hacer una segunda gráfica

```
par(new=T)
```

y ubicaremos el boxplot en el vértice superior izquierdo, ya que allí tenemos lugar

```
par(fig=c(0.1,0.5,0.5,0.95))

boxplot(tablaR292[,c(2:3)] ,cex.lab=0.8,border=c(1,2))
```



La gráfica no solo nos está mostrando en puntos negros como varía z e y en función de x, sino que nos está mostrando como son las distribución de valores de z e y en el inserto superior izquierdo.

9.5. Gráficos múltiples

Ya hemos visto la construcción de gráficos múltiples utilizando la función layout(). En esta oportunidad veremos como hacer gráficos múltiples con modificación de parámetros. Debemos ser conscientes que la modificación de parámetros va produciendo cambios en los gráficos que en algunas situaciones puede no ser útiles. Por ello siempre se recomienda modificar el parámetro temporariamente y volver siempre a los valores por defecto, que más se adaptan a condiciones estándar.

Veremos el parámetro mfrow. Este parámetro es un vector de dos números que indica el número de filas y columnas de una matriz donde se ubicarán los datos. El valor por defecto es

```
> par('mfrow')
[1] 1 1
```

esto, como dijimos permite hacer un gráfico a la vez. Cambiemos el parámetro a una matríz de 2 filas y 2 columnas. Es decir que nos dará un espacio para insertar cuatro gráficos

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

Utilizando los datos de la tablaR292 graficamos en una grilla de 2 por 2

y en función de x

```
> plot(tablaR292$x,tablaR292$y,xlab='x',ylab='y')
```

z en función de x

```
> plot(tablaR292$x,tablaR292$z,xlab='x',ylab='z')
```

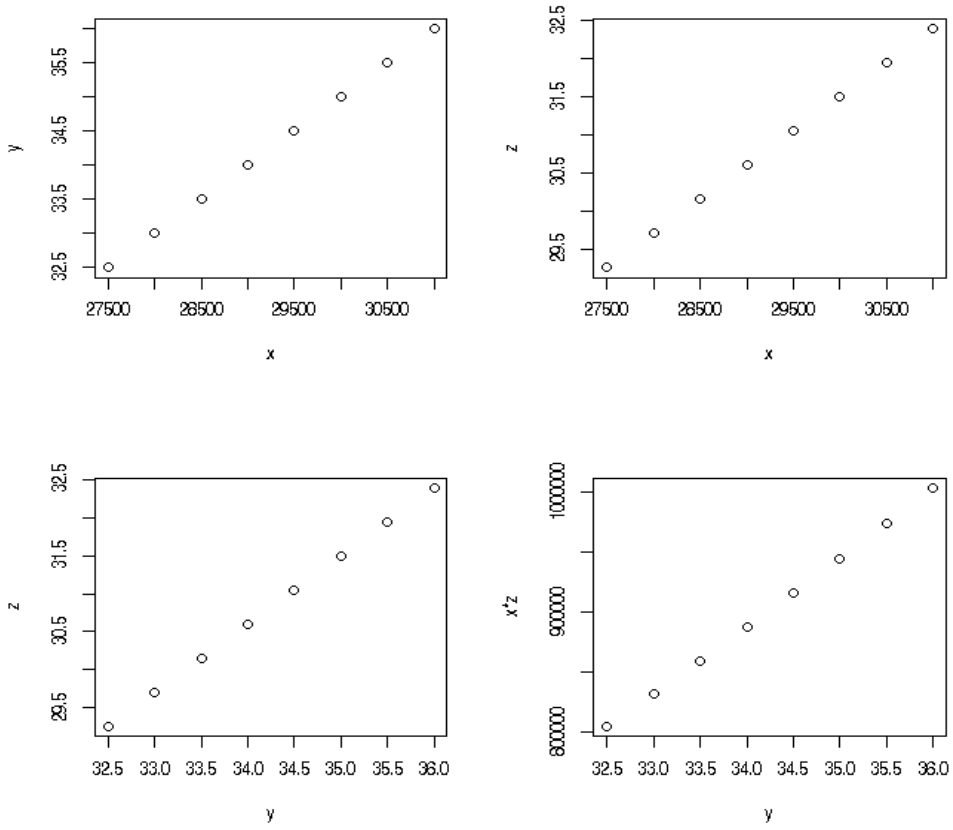
z en función de y

```
> plot(tablaR292$y,tablaR292$z,xlab='y',ylab='z')
```

el producto de x por z en función de y

```
> plot(tablaR292$y,tablaR292$x*tablaR292$z,xlab='y',ylab='x*z')
```

así obtenemos



recuerde regrese el parámetro a valores iniciales.

```
> par(mfrow=c(1,1))
```

o dev.off()

Es una práctica ordenada, trabajar con los valores por defecto y solo cambiarlo a situaciones específicas cuando se da la necesidad.

10. Resumen de códigos para gráficos

10.1.1. Ingreso y auditoría de datos

1. tablaR231[1]<-read.table("clipboard",header=T,dec=",",sep="\t",encoding="latin1")
2. summary(tablaR231)
3. names(tablaR231)
4. ncol(tablaR231)
5. nrow(tablaR231)
6. head(tablaR231)
7. str(tablaR231)
8. is.data.frame(tablaR231)

10.1.2. boxplot

1. boxplot(y~x,data=objeto del espacio de trabajo)
2. border= con palabras o un vector indica el color de los bordes de la caja de cada serie de datos. Ej: border="red", border=c("red","blue"), border=c(1,2)

10.1.3. scatter plots

1. plot(y~x,data=tablaR231)
2. type= "p" puntos, "l" lines, "b" puntos y lineas, "s" y "S" escalera, "h" barra o altura. Ej: type="p"
3. pch= número que indica el tipo de símbolo para el punto. Ej: pch=20, pch=c(12,29)
4. lwd= número que indica el ancho de la línea. Ej: lwd=2
5. cex= tamaño del símbolo que representa cada punto. Ej: cex=3
6. lty= número que indica el tipo de línea. 1: continua, 2: puntos, etc. Ej: lty=2
7. points(z~x,data=objeto del espacio de trabajo) agrega serie de datos en formato scatter plots
8. lines(spline(x,y)) permite dibujar una línea de ajuste sobre un gráfico, a partir de dos vectores x e y.

10.1.4. Scatterplot ampliado

(requiere biblioteca car)
scatterplot(y~x,data=tablaR283,ellipse=T,levels=c(0.5,0.9),span=1)

10.1.5. gráficos de sectores o tortas

1. pie(table(tablaR241$sexo))
2. labels= string o vectores de strings que lleva los nombres o detalles de cada sector. Ej: labels=c("h","m")
3. edges: número que indica el número de lados del polígono para aproximar el círculo de la torta. Ej. edges=50
4. radius: número que indica el radio de la torta. Cuanto mayor el número más grande la torta. Ej: radius=1
5. clockwise: T o F. Indica si los sectores se grafican a favor o en contra de las agujas del reloj.
6. init.angle: número que indica desde donde se comienza a dibujar el primer sector. Ej. init.angle=30.

1 tablaR231, nombre del objeto tomado como ejemplo. Podría ser cualquier otro nombre

7. density= número o vector de números que indica la cantidad de líneas que tendrá el relleno de cada sector. Ej. density=c(20,10,15). Habrá tres sectores con diferente número de líneas.
8. angle: número o vector numérico que indica la inclinación de las líneas que llenan cada sector. Ej. angle=c(0,30,90)
9. border: número o vector numérico (tambien puede ser con los nombres entre comillas de los colores) que indica el color del borde externo de cada sector.
10. lty: tipo de linea, que en el gráfico de sectores se refiere a las líneas de relleno y border.

10.1.6. dotchart

1. dotchart(tablaR241$idioma,groups=tablaR241$sexo,gdata=tapply(tablaR241$idioma,tablaR241$sexo,mean)). Grafica los idiomas separados por sexo y muestra vlores individuales y la media.
2. gpch: tipo de punto para identifica la estadística definida en gdata.
3. gcol= color que se le dará a la estadística definina en gdata.

10.1.7. starplots

1. stars(tablaR241[2:4].....). grafica las variables de las columnas 2 a la cuatro de la tabla indicada.
2. key.lock: vector que indica coordenadas x e y en el gráfico donde aparezca la leyenda que indica las variables. Ej: ,key.loc=c(17,3)
3. key.labels: vector que permite asignar los nombres a los sectores. Ej. key.labels=colnames(tablaR241)[2:4]
4. draw.segments= T o F, indica si el gráfico se hace con polígonos (F) o sectores circulares (T)
5. col.segments: vector numérico o con palabras que permite asignar color a cada sector circular. Ej. col.segments=c(1,2,1)

10.1.8. histograma

1. hist(estudiantes$edad) hace el histograma de la columna edad del data.frame estudiantes.
2. breaks: vector que indica los número en los se dividen las categorías del histograma. Ej. breaks=c(0,5,10,15,20,25)
3. include.lowest: TRUE o FALSE. Si TRUE, un valor que cae sobre el valor del break, es incluido en el grupo para el cual dicho break es el extremo inferior
4. right=TRUE o FALSE. Si TRUE,un valor que cae sobre el valor del break, es incluido en el grupo para el cual dicho break es el extremo superior.
5. freq= TRUE o FALSE. Si TRUE, el histograma muestra el número de individuos. Si FALSE muestra la densidad.

10.1.9. Barplot

1. barplot(tapply(tablaR262$glucemia,tablaR262$tratamiento,mean).....
2. horiz: T o F, hace las barras horizontales o verticales
3. density: número o vector. determina el número de lineas de relleno de cada barra en lugar de colores plenos.
4. angle: número o vector. determina la inclinación de las líneas de relleno.
5. rect(xleft,yleft,xright,yright....) dibuja un rectángulo con los vertices indicados.
6. segments(x0,y0,x1,y1,....) dibuja un vector que va desde el punto (x0,y0) al punto (x1,y1)

10.1.10. Barplots con desvíos

usa biblioteca gplots

```
> medias<-tapply(tablaR285$calcemia,tablaR285$tratamiento,mean)

> sd<-tapply(tablaR285$calcemia,tablaR285$tratamiento,sd)
```

barplot2(medias,names.arg=c("controles","tratados"),plot.ci=TRUE,ci.l=c(medias-sd),ci.u=c(medias+sd))

10.1.11. scatterplot3D

hacer gráfica a partir de una tabla con tres columnas: x, y,z

1. > sp3d<-scatterplot3d(tablaR271$x,tablaR271$y,tablaR271$z)

agregar puntos a un gráfico con valores x, y, w

2. > sp3d$points3d(tablaR271$x,tablaR271$y,tablaR271$w)
3. angle: rota el grafico
4. type: "p", "l","h" grafica puntos, líneas o alturas desde la base hasta el punto
5. scale.y=1: cambia la dimensión del eje y respecto a x.
6. grid= T o F: coloca o saca la grilla

10.1.12. grafico 3D: persp

grafica los valores de una matriz (xy) de datos donde x e y son los valores de las filas y columnas.

1. persp(x,y,xy,theta=30,phi=45,...)
2. theta: ángulo de giro sobre la base del gráfico
3. phi: ángulo de giro sobre la cara trasera.

10.1.13. gráfico 3D: contour

grafica valores de una tercer variable contenida en una matriz de datos, pero en el plano a través de curvas de nivel.

1. contour(x,y,xy, nlevels=4.....)
2. nlevels: número que indica la cantidad de niveles que se graficarán

10.1.14. gráfico 3D: image

grafica valores de una tercer variable contenida en una matriz z de datos, pero en el plano a través de niveles de color

1. image(x=seq(1,nrow(z)),y=seq(1,ncol(z)),z,col=heat.colors(4))
2. heat.colors: número que indica la cantidad de valores de color en que se dividirá entre los mayores y los menores valores.

10.1.15. gráfico 3D: filled.contour

grafica valores de una tercer variable contenida en una matriz z de datos, pero en el plano a través de niveles de color

1. filled.contour(x=seq(1,nrow(z)),y=seq(1,ncol(z)),z,nlevels=4)
2. nlevels: número que indica

10.1.16. Gráficos de alta densidad de puntos

plot(hexbin(tablaR282$x,tablaR282$y,xbins=2),xlab="x",ylab="y")

10.1.17. bandplot

bandplot(tablaR284$fecha,tablaR284$temperatura,sd.col=c("blue","red","black","red","blue"),sd.lty=c(2,1,6,1,2),sd.lwd=c(1,1,2,1,1),pch=1,col="black",cex=0.8,las=1,ylim=c(20,60))

10.1.18. Generales para todo tipo de gráficos

10.1.19. títulos

1. main= "entre comillas se coloca el título de la gráfica". Ej: main="mi gráfico"
2. cex.main= con un número indica el tamaño de letra. Ej: cex.main= 2
3. font.main= con un número indica el tipo: 1- normal, 2- negrita, 3- cursiva, 4-negrita cursiva. Ej: font.main=3
4. sub: "entre comillas se coloca el pie de la figura". Ej: sub="pie de la figura"
5. cex.sub= con un número indique el tamaño de la letra. Ej: cex.sub=2
6. font.sub= con un número indica el tipo: 1- normal, 2- negrita, 3- cursiva, 4-negrita cursiva. Ej: font.sub=3

10.1.20. formato ejes

7. xlab= "entre comillas coloque la descripción del eje x". Ej: xlab="rotulo eje x"
8. ylab= "entre comillas coloque la descripción del eje y"Ej: ylab="calcemia, mg/dl"
9. col.lab="entre comillas el color" o bien el número obtenido con colours(). Ej: col.lab= "green"
10. cex.lab= con un número indica el tamaño de los rótulos de los ejes. Ej: cex,lab=1.5
11. cex.axis= con un número indica el tamaño de la escala de los ejes. Ej: cex.axis= 0.8
12. col.axis= "entre comillas el color de los rótulos de los ejes" Ej: col.axis= "blue"
13. las= con un número coloca los números de las escalas de los ejes de diferente forma. las= 0 rótulos de escala de los ejes paralelos a ambos ejes. las= 1 rótulos de eje y perpendicular al eje, rótulos de x paralelos al eje. las= 2 rótulos de los eje perpendiculares a ambos ejes. las =3 rótulos de eje y paralelo al eje, rótulos de x perpendicular al eje. Ej: las=2
14. xlim=c(limite inferior,limite superior del eje x). Ej: xlim=c(0,15)
15. ylim=c(limite inferior,limite superior del eje y). Ej: ylim=c(-2,13)
16. frame= T o F. coloca o no un recuadro a la gráfica. Ej: frame= TRUE
17. fg= número o color entre comillas que indica el color de los ejes. Ej: fg= 4
18. asp= relación entre los ejes. Ej: asp=2
19. pos= número que indica en que valor se cortan los ejes. Ej: pos= 1.2
20. axes= T o F permite visualizar o no los ejes. Ej: axes= FALSE
21. axis(1,), permite según el número 1,2, 3, o 4 colocar ejes abajo, izq, arriba o derecha.

10.1.21. Formato series

22. col= con un número o un vector de números (o palabras) indica el color de las series de datos.
23. range= 0 incluye todos los datos en las cajas y bigotes, sin colocar range deja fuera los outliers. (exclusivo boxplot)

10.1.22. Textos y leyendas

24. legend(con los argumentos necesarios) coloca la leyenda
25. text(x, y, "entre comillas el texto a colocar") coloca un texto dentro del gráfico en la posición x y indicada en el argumento.
26. mtext("texto que se desea colocar sobre el borde superior del gráfico", adj=1 (a la derecha), col= color, cex=tamaño). adj=0 queda a la izquierda, adj=0.5 queda al medio