1 Visualización de datos

Las primeras acciones relativas al manejo de un conjunto de datos que solemos realizar comprenden la lectura y visualización de los mismos.

Comenzaremos por leer un conjunto de datos en un data.frame, que es una estructura de datos muy libre de R que generaliza el concepto de matriz y que sirve para almacenar información.

Primero indicamos el subdirectorio de trabajo, leeremos los datos y veremos los primeros registros. Podemos inspeccionar en que directorio estamos mediante la instrucción getwd().

Con frecuencia los datos están en un archivo que leemos y alojamos en un data frame o marco de datos, que no es más que una lista de variables con el mismo número de filas. R puede leer datos guardados como archivos de texto con las funciones read.table(), scan(), entre otras. También puede leer archivos en otros formatos como excell, SPSS, etc., mediante el uso de paquetes específicos.

La función read.table(), que suele ser la más usada, crea un marco de datos (o data frame). Si en el primer renglón de nuestro archivo tenemos los nombres de las variables, ponemos header=T en el comando de lectura, como se muestra a continuación.

```
> rm(list=ls())
> # rm es el comando remove: borramos todo lo que pueda estar almacenado
> getwd()

[1] "C:/Users/Ana/Dropbox/lanueva/listas/Graficos"

> # En que directorio estamos?
> setwd("C:\\Users\\Ana\\Dropbox\\lanueva\\listas\\Graficos")
> # establezco directorio de trabajo
> Alumnos<- read.table("StudentSurvey.txt",header=T)
> head(Alumnos) # devuelve las primeras 6 lineas de los datos y sus nombres

       Year Gender Smoke   Award HigherSAT Exercise TV Height Weight Siblings
1    Senior      M    No Olympic      Math       10  1     71    180        4
2 Sophomore      F   Yes Academy      Math        4  7     66    120        2
```

```
3 FirstYear      M    No   Nobel      Math       14  5     72    208         2
4    Junior      M    No   Nobel      Math        3  1     63    110         1
5 Sophomore      F    No   Nobel    Verbal        3  3     65    150         1
6 Sophomore      F    No   Nobel    Verbal        5  4     65    114         2
  BirthOrder VerbalSAT MathSAT  SAT  GPA Pulse Piercings
1          4       540     670 1210 3.13    54         0
2          2       520     630 1150 2.50    66         3
3          1       550     560 1110 2.55   130         0
4          1       490     630 1120 3.10    78         0
5          1       720     450 1170 2.70    40         6
6          2       600     550 1150 3.20    80         4
```

Otra opción de lectura muy útil y flexible es scan(), que permite tanto la lectura de un archivo como desde la pantalla.

Notemos que este data frame contiene variable de distinto tipo: numéricas y caracteres. Podemos acceder a cada variable individualmente escribiendo: Alumnos$Year, Alumnos$Gender, etc. Si el archivo no contuviera los nombres de las variables, éstas recibirían por default el nombre V1, V2, etc. y en ese caso accederíamos a ella escribiendo Alumnos$V1, Alumnos$V2, etc. Hacer un attach() del archivo nos permite llamar a las variables por su nombre:

```
> attach(Alumnos)
```

Una vez ejecutado el attach(), podemos referirnos a las variables sin el el prefijo Alumnos$, ahora podemos acceder a las variables llamando Year, Gender, etc. Es recomendable revisar antes si ya existen variables definidas con el mismo nombre en el entorno de trabajo.

Inspeccionemos los datos con el editor, que permite hacer cambios en el data frame. Con la siguiente instrucción se abre un editor en el que vemos los datos y podemos modificralos.

```
> fix(Alumnos)
```

Observemos que en algunos casos hay valores NA. Esta es la forma de representar en R que ese es un valor faltante, independientemente del tipo de datos. Por lo tanto, NA (Not Available) es un símbolo reservado en R.

Las variables registradas en nuestro data frame son:

1. **Year**: Años en la institución: FirstYear, Sophomore, Junior, o Senior
2. **Gender**: Sexo: F o M

3. **Smoke**: Fuma? No o Yes
4. **Award**: Premio preferido: Academy, Nobel, o Olympic
5. **HigherSAT**: Qué puntaje SAT es más alto? Math o Verbal
6. **Exercise**: Horas semanales de ejercicio
7. **TV**: Horas semanales de TV
8. **Height**: altura (en pulgadas)
9. **Weight**: peso (en libras)
10. **Siblings**: número de hermanos
11. **BirthOrder**: orden de nacimiento, 1 = mayor, 2 = segundo mayor, etc.
12. **VerbalSAT**: puntaje SAT en lengua
13. **MathSAT**:puntaje SAT en matenáticas
14. **SAT**: puntaje SAT combinado de lengua y matemáticas
15. **GPA**: puntaje promedio obtenido en el colegio
16. **Pulse**: pulso cardíaco (latidos por minuto)
17. **Piercings**: número de piercings

Como vemos son variables de distinta naturaleza, categóricas y cuantitativas: dicotómicas (Gender, Smoke), categóricas (Award), categóricas ordinales (Year), discretas (Siblings) y continuas (GPA). Es importante esta distinción porque la naturaleza de las variables puede determinar el tipo de tratamiento que reciban.

Respecto de este conjunto de datos pordíamos tener algunas preguntas, por ejemplo:

- ¿Cuál es el porcentaje de hombres y de mujeres?
- ¿Qué porcentaje de los alumnos fuma?
- ¿Quiénes fuman más, hombres o mujeres?
- ¿Cuál es el promedio de horas de ejercicio?

- ¿Hay alumnos con un puntaje de VerbalSAT inusualmente alto o bajo?
- ¿Cuál es el premio más deseado?
- ¿Qué relación hay entre peso y altura? ¿Es la misma en hombres que en mujeres?
- ¿Los alumnos que hacen más ejercicio tienden a tener pulso más bajo?

1.1 Variables Categóricas (factor en R)

Comencemos por las variables cualitativas. Podríamos calcular la cantidad de hombres y mujeres, la cantidad de individuos que fuma o no fuma y como se cruzan estas variables.

```
> names(Alumnos)

 [1] "Year"       "Gender"     "Smoke"      "Award"      "HigherSAT"
 [6] "Exercise"   "TV"         "Height"     "Weight"     "Siblings"
[11] "BirthOrder" "VerbalSAT"  "MathSAT"    "SAT"        "GPA"
[16] "Pulse"      "Piercings"

> summary(Gender)

  F   M
169 193

> #summary de una var. caregorica da la frecuencia de cada categoria
> summary(Smoke)

 No Yes
319  43

> # Otra opcion es
> table(Gender)

Gender
  F   M
169 193
```

Podemos hacer distintos gráficos representativos. Gráficos de barras como el siguiente, en el que se han elegido opciones de color y de líneas

de sombreado oblicuas. Observemos que la entrada a este gráfico es una tabla de frecuencias calculada, en este caso con el comando table():

```
> table(Gender)

Gender
  F   M
169 193

> counts.gender <- table(Gender)
> barplot(counts.gender,col="blue",density=4)
```

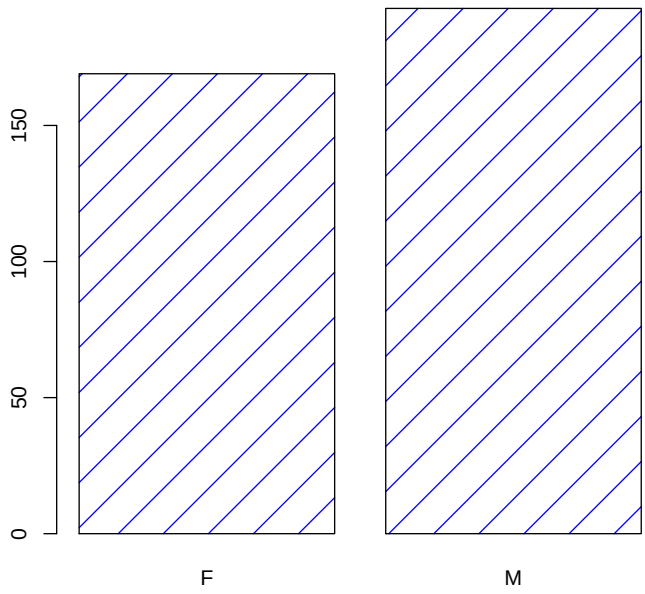

Figure 1: Barplot

o gráficos de torta:

```
> table(Smoke)

Smoke
 No Yes
319  43

> counts.smoke <- table(Smoke)
> pie(counts.smoke, col=c("blue","green"), main="Grafico de Torta de Smoke")
```

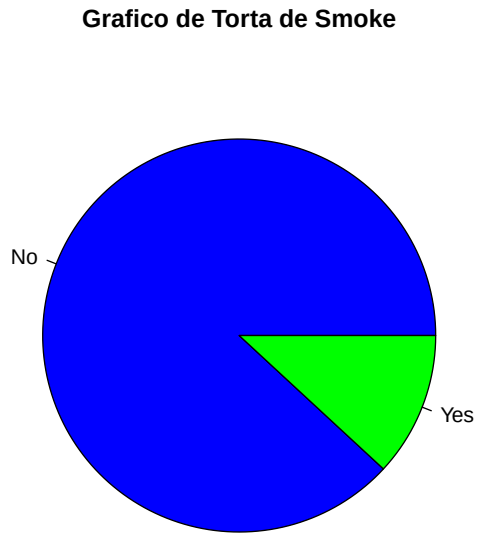

Figure 2: Pie Chart

Si queremos realizar un diagrama de torta 3D, podemos hacerlo usando el paquete plotrix, que debe instalarse y cargarse previamente:

```
> library(plotrix)
> pie3D(counts.smoke,col=c("blue","green"), main="Grafico de Torta 3D de Smoke")
```

Figure 3: Pie 3D

Crucemos estas dos variables, es decir, consideremos como se distribuyen los 362 estudiantes entre las categorías que surgen al combinar el sexo y la condicion de fumador. Por ejemplo, tenemos 153 estudianes femeninos que no fuman y 16 femeninos que si fuman. Con el siguiente comando R construye una tabla de contingencia en la que clasifica a todos los alumnos teniendo en cuenta todas las categorías posibles al cruzar las dos variables.

```
> xtabs(~ Smoke+Gender)

     Gender
Smoke   F   M
  No  153 166
  Yes  16  27
```

Este tipo de tablas se suele utilizar, entre otras cosas, para estudiar la independencia entre las variables involucradas: Gender y Smoke en nuestro caso. Podemos graficar las frecuencias mediante barplot():

```
> counts <- table(Smoke,Gender)
> barplot(counts,col=c("blue","red"),main="Smoke vs. Gender")
```

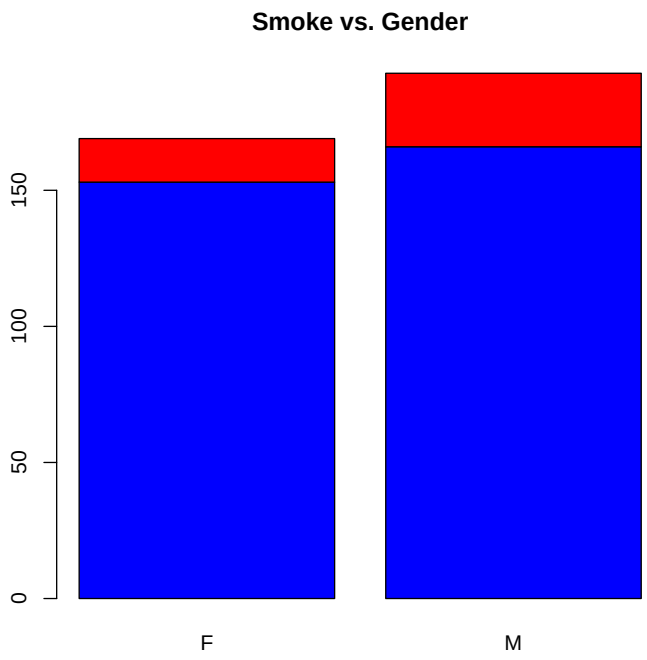

Figure 4: Barplot

También podemos graficar las frecuencias relativas dentro de cada clase de Gender con la instrucción plot() como sigue:

```
> plot(Gender,Smoke,col=c("blue","red"))
```

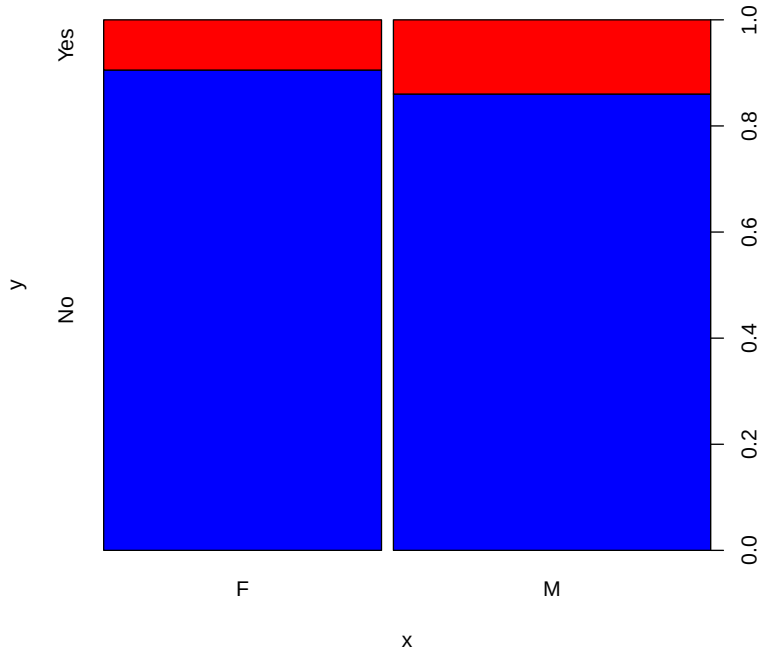

Figure 5: Plot

Asimismo, podríamos interesarnos en Gender y Award, :

```
> xtabs(~ Gender+Award)
```

```
      Award
Gender Academy Nobel Olympic
     F      20    76      73
     M      11    73     109
```

Los barplots agrupados pueden tener distintas orientaciones como se muestra a continuación. Si a los fines de la compración, queremos disponer los dos barplots uno al lado del otro en un mismo gráfico, podemos utilizar el comando par() con el parámetro mfrow=c(nr,nc) que permite dividir la ventana gráfica como una matriz con nr filas y nc columnas. Con el comando

par() establecemos cambios en la ventana gráfica que permanecerán vigentes hasta que los volvamos a cambiar.

```
> counts <- table(Gender,Award)
> par(mfrow=c(1,2))
> barplot(counts, names.arg=c("A","N","O"),main="Award vs. Gender",
+         col=c("blue","red"),xlab="Award", legend = rownames(counts),
+         args.legend = list(x = "topleft", bty = "n"))
> barplot(counts, names.arg=c("A","N","O"),main="Award vs. Gender",
+         col=c("blue","red"),xlab="Award", legend = rownames(counts),
+         args.legend = list(x = "topleft", bty = "n"),beside=TRUE)
> par(mfrow=c(1,1)) # restablecemos la ventana grafica sin divisiones
```

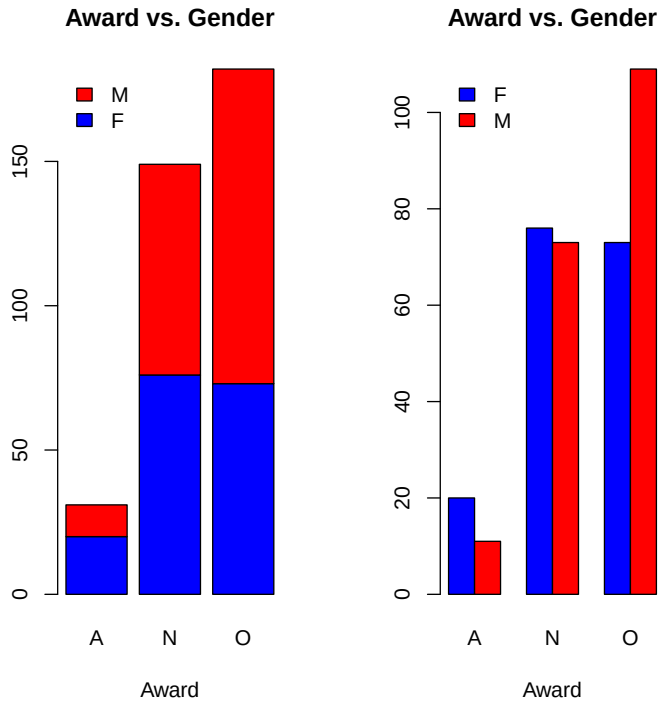

Figure 6: Barplot

Se podrían agregar más variables, por ejemplo Smoke, en cuyo caso quedarían las siguientes tablas:

```
> xtabs(~ Award+Gender+Smoke)
```

```
, , Smoke = No

         Gender
Award      F  M
  Academy 19 10
  Nobel   68 61
  Olympic 66 95

, , Smoke = Yes

         Gender
Award      F  M
  Academy  1  1
  Nobel    8 12
  Olympic  7 14
```

1.2 Variables Cuantitativas (numeric en R)

Sigamos con las variables cuantitativas. Consideremos la variable GPA, que es continua. Lo primero que podríamos calcular es un summary, o sea un resumen de esta variable.

Miremos los primeros valores con head(), veamos el rango de variación de la variable usando el comando range() y calculemos el promedio muestral mediante mean():

```
> head(GPA)

[1] 3.13 2.50 2.55 3.10 2.70 3.20

> range(GPA)

[1] NA NA

> mean(GPA)

[1] NA
```

Como vemos la respuesta a range(GPA) y mean(GPA) es NA y esto se debe a que esta variable presenta missings. Para saber cuantos missing tiene la variable GPA podemos usar la función lógica is.na() que devuelve un FALSE si el valor está presnte y TRUE si está ausente.

```
> #is.na(GPA)
> #[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
> # FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> # FALSE FALSE FALSE .......
```

Podemos sumar para saber cuántos son:

```
> sum(is.na(GPA))

[1] 17

> # is.na(GPA) toma valores logicos, sin embargo se puede sumar dado que
> # en R el TRUE y el FALSE tienen una dualidad logico/numerica (1 y 0)
```

Ante la presencia de missings, deberíamos excluirlos para realizar las operaciones deseadas:

```
> range(GPA,na.rm=TRUE)

[1] 2 4

> mean(GPA,na.rm=TRUE)

[1] 3.157942
```

El comando summary() calcula medidas de resumen del argumento.

```
> summary(GPA,na.rm=TRUE)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.000   2.900   3.200   3.158   3.400   4.000      17
```

summary() informa el mínimo, el primer cuartil (o percentil 25%), la mediana (o segundo cuartil), la media, el tercer cuartil (o percentil 75%), el máximo y el número de missings.

Las 5 medidas en azul son lo que se llama los **5 números resumen**. Podríamos calcular las medidas clásicas de media, varianza o desvío muestrales por separado:

```
> mean(GPA,na.rm=TRUE)

[1] 3.157942
```

```
> var(GPA,na.rm=TRUE)

[1] 0.1586594

> sd(GPA,na.rm=TRUE)

[1] 0.3983207
```

También podríamos calcular la mediana, las medias $\alpha-$podadas y la MAD:

```
> median(GPA,na.rm=TRUE)

[1] 3.2

> mean(GPA,na.rm=TRUE,trim=0.1) # trim=porcentaje de poda

[1] 3.174368

> mean(GPA,na.rm=TRUE,trim=0.2)

[1] 3.181643

> mad(GPA,na.rm=TRUE)           # calcula la mad estandarizada

[1] 0.429954
```

R tiene un número enorme de funciones predefinidas que de gran utilidad en estadística, la siguiente tabla detalla algunas de ellas:

- n datos: $x_1, x_2, \ldots, x_n$
- datos ordenados $x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)}$ - en R: sort(datos)

sum(datos)	suma de los elementos de datos
prod(datos)	producto de los elementos de datos
sort(datos)	ordena en forma ascendente los elementos de datos
rev(sort(datos))	ordena en forma descendente los elementos de datos
order(datos)	vector de rangos de los elementos de datos
prod(datos)	producto de los elementos de datos
mean(datos)	$\overline{x}$
median(datos)	$\tilde{x}$
quantile(datos, alfa)	$x_{([n\alpha])}$
quantile(datos,0.25)	Q_1 primer cuartil
quantile(datos,0.75)	Q_3 tercer cuartil
min(datos), max(datos)	$x_{(1)}$, $x_{(n)}$
which.max(datos)	devuelve el índice del elemento máximo de x
which.min(datos)	devuelve el índice del elemento mínimo de x
mean(datos,trim = alfa)	$\overline{x}_\alpha = \{x_{([n\alpha])} + \ldots + x_{(n-[n\alpha])}\}/(n - 2[n\alpha])$
var(datos)	$s^2 = \sum(x_i - \overline{x})^2/(n-1)$
sd(datos)	$\sqrt{s^2}$
IQR(datos)	$d_I = Q_3 - Q_1$
mad(datos)	1.4826 mediana$(\lvert x_i - \tilde{x}\rvert)$

* $1.4826 = 1/\Phi^(-1)(3/4)) = 1/qnorm(3/4))$, si no poner mad(datos,constant=1)
* indicar acción si hay NA

Si quisiéramos aplicar estas operaciones a un subconjunto de variables determinado podríamos primero definir el subconjunto de variables de interés y luego aplicar la función que nos interesa. Una forma eficiente de hacerlo es mediante la función apply() que actúa sobre las filas o columnas de una matriz o un data frame aplicando una determinada función.

Teniendo en cuenta, que algunas variables contienen algunos registros faltantes, deberíamos calcular, por ejemplo, la media muestral de cada variable omitiendo para cada una de ellas los casos NA. En las siguientes líneas calculamos la media y desvío muestrales para cada variable de SUB, matriz que hemos formado mediante el comando cbind() que combina columnas (rbind() combina filas):

```
> SUB<-cbind(Exercise,TV,Height,Weight,VerbalSAT,MathSAT,SAT,
+            GPA,Pulse,Piercings)
> apply(SUB,2,mean,na.rm=TRUE) # 2 indica accion por columna y 1 por fila

   Exercise          TV      Height      Weight   VerbalSAT     MathSAT
   9.054017    6.504155   68.422535  159.798319  594.190608  609.436464
```

```
        SAT         GPA       Pulse   Piercings
1203.627072    3.157942   69.574586    1.673130
```

```
> apply(SUB,2,sd,na.rm=TRUE)
```

```
   Exercise          TV      Height      Weight   VerbalSAT     MathSAT
  5.7407405   5.5837671   4.0785437  31.6194667  74.1763984  68.4900672
        SAT         GPA       Pulse   Piercings
121.2852074   0.3983207  12.2051356   2.1727027
```

Consideremos los resultados en lengua y matemáticas, VerbalSAT y MathSAT, y realicemos sendos histogramas:

```
> par(mfrow=c(1,2))
> hist(VerbalSAT,freq=F)
> hist(MathSAT,freq=F)
> par(mfrow=c(1,1)) #Volvemos a hacer una ventana grafica sin divisiones
```

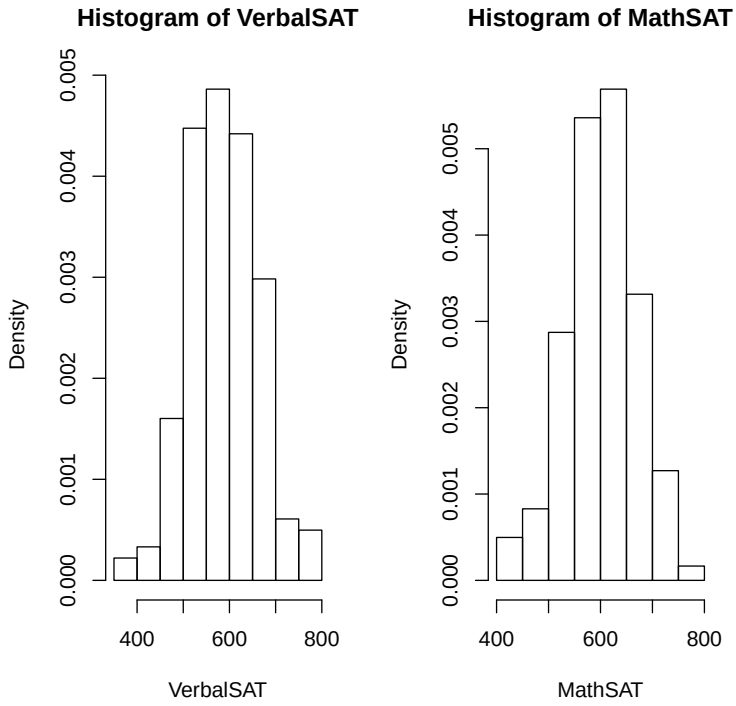

Figure 7: Histogramas

Si quisiéramos guardar en disco este gráfico podríamos hacerlo de las siguientes maneras:

En pdf:

```
> pdf ("histogramas.pdf ")
> par(mfrow=c(1,2))
> hist(VerbalSAT,freq=F)
> hist(MathSAT,freq=F)
> graphics.off()
```

En Encapsulated PostScript:

```
> postscript("histogramas.eps")
> par(mfrow=c(1,2))
> hist(VerbalSAT,freq=F)
> hist(MathSAT,freq=F)
> graphics.off()
```

Por lo tanto, deberían aparecer en el directorio de trabajo los archivos histogramas.pdf y histogramas.eps.

Si quisiéramos sobreimprimir una curva normal, que es una práctica bastante común, ¿cómo podríamos hacerlo? Primero, deberíamos decidir qué normal tiene que ser. Es decir, ¿con qué media y con qué desvío stantard?

La densidad de la norma con media μ y desvío standard σ

$$f(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{\sigma^2}}$$

está programada en R en la función dnorm.

Por lo tanto, una forma de realizar el plot deseado para VerbalSAT hacemos un histograma de densidad y superponemos la densidad normal, con lo que tendríamos:

```
> media<-mean(VerbalSAT)
> desvio<-sd(VerbalSAT)
> grilla<-seq(range(VerbalSAT)[1],range(VerbalSAT)[2],length=100)
> names(hist(VerbalSAT,freq=F))

[1] "breaks"   "counts"   "density"  "mids"     "xname"    "equidist"

> maximo<-max(hist(VerbalSAT,freq=F)$density)+0.0005
```

```
> hist(VerbalSAT,ylim=c(0,maximo),freq=F,main=
+        "Histograma de Densidad de VerbalSAT")
> lines(grilla,dnorm(grilla,media,desvio),col="blue")
```

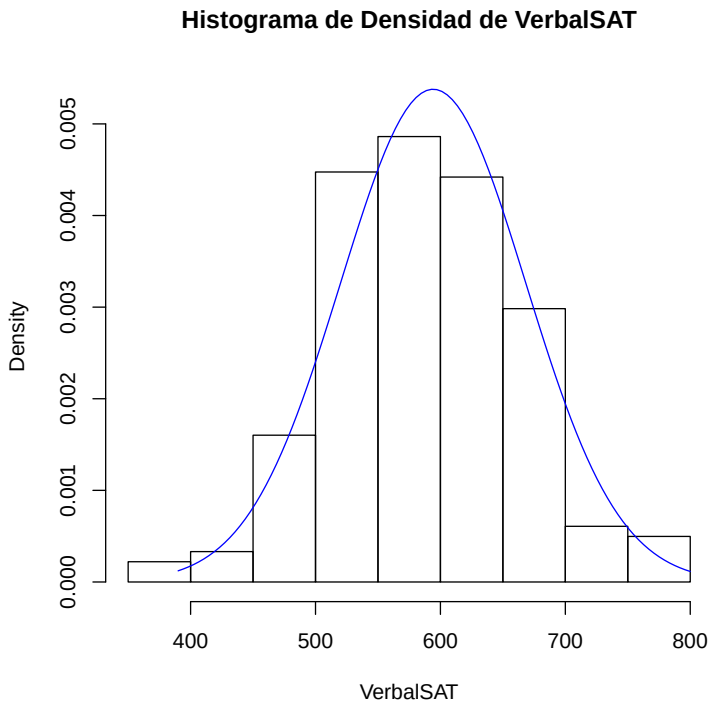

Figure 8: Histogramas+Curva Normal

Grafiquemos el boxplot de VerbalSAT de todos los alumnos juntos y en boxplots paralelos clasificados de acuerdo con su condición de fumador:

```
> par(mfrow=c(1,2))
> boxplot(VerbalSAT, main="VerbalSAT")
> boxplot(VerbalSAT~Smoke, xlab="Smoke",main="VerbalSAT")
> # ~ selecciona la condicion de fumador
> par(mfrow=c(1,1))
```

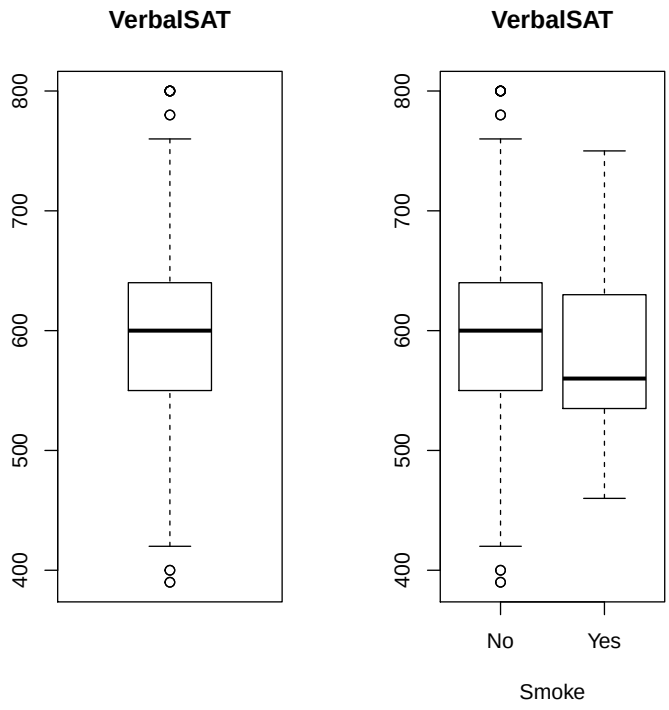

Figure 9: Boxplots

Observemos que el objeto creado con el comando boxplot(), tiene información que nos permite identificar los outliers. En efecto, los índices de los outliers los podemos obtener como sigue:

```
> names(boxplot(VerbalSAT)) # devuelve nombres del objeto

[1] "stats" "n"     "conf"  "out"   "group" "names"

> names(VerbalSAT)=1:length(VerbalSAT) # ponemos nombre a cada observacion
> boxplot(VerbalSAT)$out
```

```
 93 116 146 148 162 200 236 240 242 278 285 310
800 800 400 400 800 390 780 800 780 800 390 800
```

También podemos realizar los boxplots paralelos de dos variables:

```
> boxplot(VerbalSAT,MathSAT,names=c("VerbalSAT","MathSAT"))
```

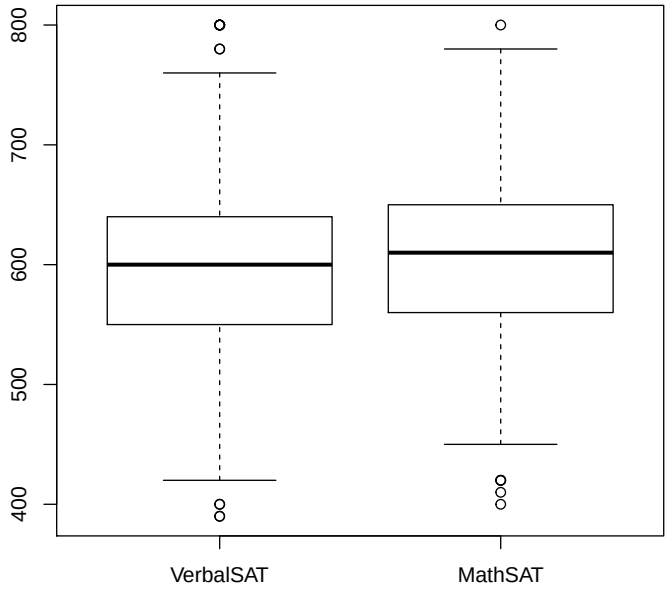

Figure 10: Boxplots

El boxplot tal como lo conocemos está fuertemente ligado a la distribución normal. Existen otras opciones, tal como el adjbox del paquete robustbase, introducido por Vandervieren y Hubert (2004), diseñado para distribuciones fuertemente asimétricas.

Existen extensiones del boxplot a tipos de datos con estructura más complejos, tales como datos bivariados o datos funcionales. En el paquete alplack se encuentra el comando bagplot(), que generaliza el boxplot usando el concepto de profundidad a datos bivariados y que permite visualizar cómo varían conjuntamente dos variables e identificar outliers en el espacio bidi-

mensional. Graficaremos a continuación los boxplots de las variables Height y Weight y su bagplot.

```
> m<-matrix(c(1:3, 3), 2, 2)
> m
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    3
```

```
> layout(m)                 # asi dividimos la pantalla en 3 partes
> layout.show(3)            # Veamos como queda
```

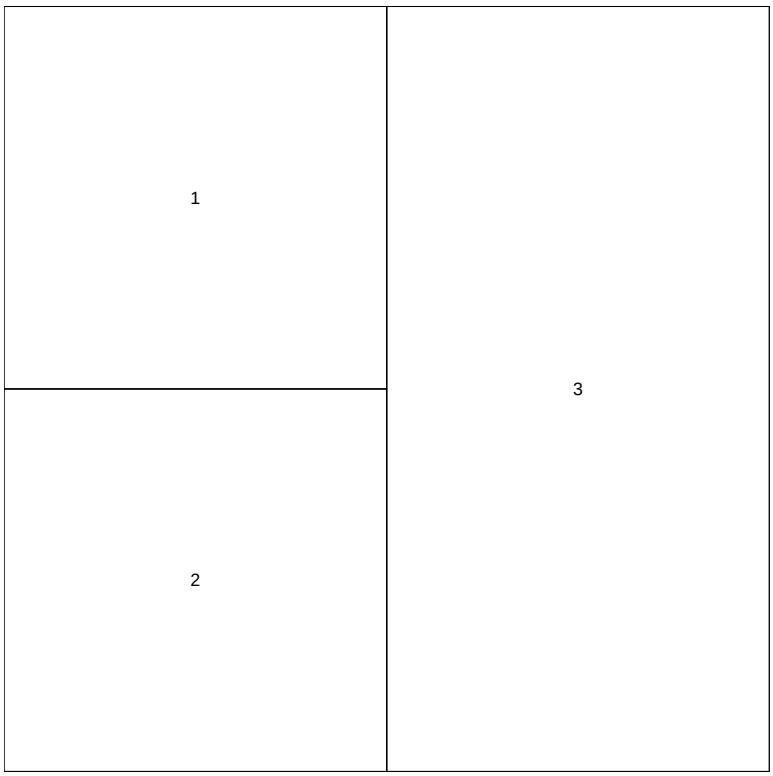

Figure 11: División de pantalla

```
> library(aplpack)
> boxplot(Height,xlab="Height")
> boxplot(Weight,xlab="Weight")
> bagplot(Height,Weight,xlab="Height",ylab="Weight",na.rm=T)

[1] "Warning: NA elements have been removed!!"

> layout(c(1,1))
```

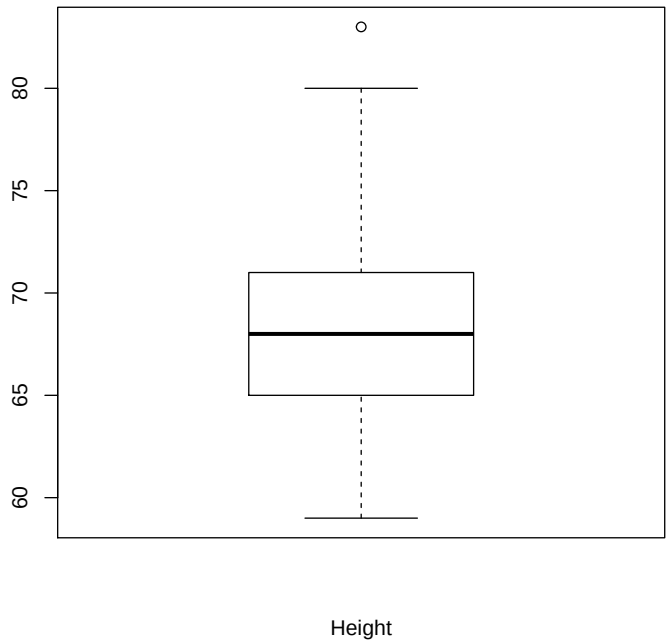

Figure 12: Bagplot

Como vemos estas figuras dan una idea de la distribución conjunta de los valores observados de dos variables y permite visualizar outliers teniendo en cuenta esta distribución.

```
> bagplot(Height,Weight,xlab="Height",ylab="Weight",na.rm=T)

[1] "Warning: NA elements have been removed!!"
```

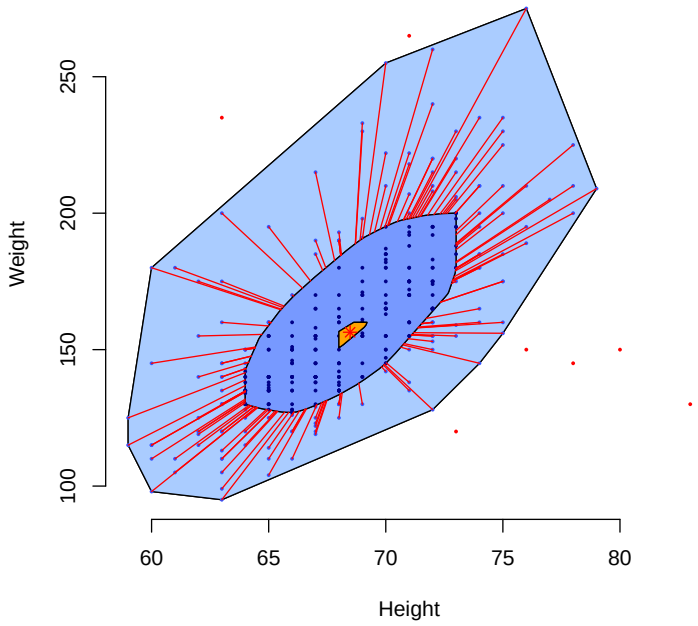

Figure 13: Bagplot

Con el mismo comando plot() que ya usamos más arriba, podemos graficar ahora dos variables numéricas, una versus otra.

```
> plot(VerbalSAT,MathSAT,xlab ="Esta es VerbalSAT",
+      ylab ="Esta es MathSAT",main ="Diagrama de VerbalSAT vs. MathSAT",pch=16)
```

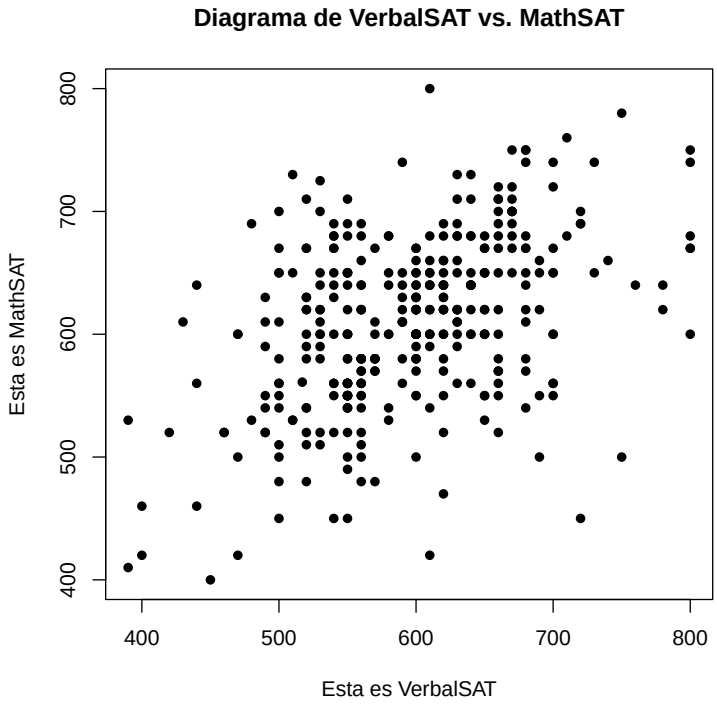

Figure 14: Scatterlot: con plot()

R tiene muchísimas opciones para personalizar los gráficos de acuerdo a nuestra necesidad o nuestro deseo. A través de los comandos par() y plot() se pueden establecer parámetros gráficos, de forma permanente o para un gráfico determinado, respectivamente. Estas opciones permiten controlar opciones tales como color de fondo (bg), márgenes (mar), ejes (axis), etc. La lista completa de parámetros gráficos puede consultarse con ?par. En el siguiente ejemplo mostramos como modificamos el gráfico anterior mediante estos parámetros.

```
> par(bg="lightgray",mar=c(4,2,3.5, 4))
> #c(bottom, left, top, right)   defalut es c(5, 4, 4, 2) + 0.1.
> plot(VerbalSAT,MathSAT,type="n",xlim=c(350,850),ylim=c(350,850),
+      xlab="", ylab="",xaxt="n", yaxt="n")
> #solo graficamos la caja
> points(VerbalSAT,MathSAT,pch=20,col="magenta")
> #solo graficamos los puntos con el simbolo deseado
> #Ahora nos encargamos de los ejes
> axis(1,c(400,600,800),cex=2)
> mtext("VerbalMAT",side=1,cex=0.8,line=3)
> axis(4,cex=0.8,col="blue",labels=FALSE)
> mtext(c(400,500,600,700,800),side=4,at=c(400,500,600,700,800),col="blue",line=0.3)
> mtext("MATHMAT",side=4,cex=0.8,line=2.5,col="blue")
> #titulo
> title(" Diagrama Personalizado de VerbalSAT vs. MathSAT",cex.main=0.8)
```

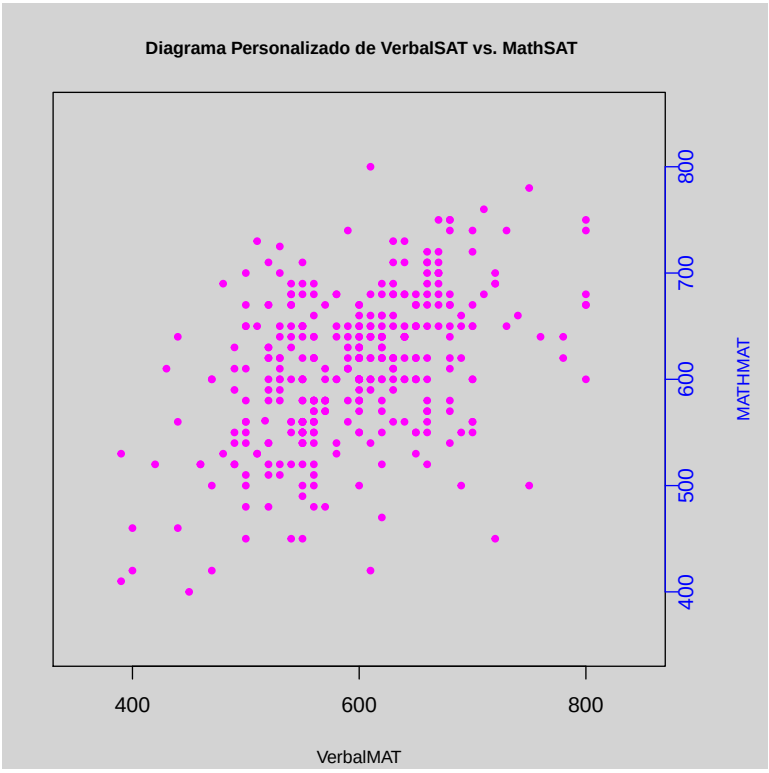

Figure 15: Scatterlot: Personalizado con plot()

Cuando se utiliza el comando par() para establecer otros parámetros gráficos es útil guardar los previos. Esto se puede realizar mediante las instrucciones

```
par()              # muestra los parametros existentes

oldpar <- par()    # copia los par\'ametros actaules

de manera de restablecerlos cuando se desea mediante:

par(oldpar)        # restabelece parametros originales
```

Si queremos hacer un scatter plot de todas las variables de SUB versus ellas mismas, podemos usar el comando pairs():

```
> par(mar=c(5, 4, 4, 2) + 0.1)
> pairs(SUB,col="magenta")
> par(bg="white")
```

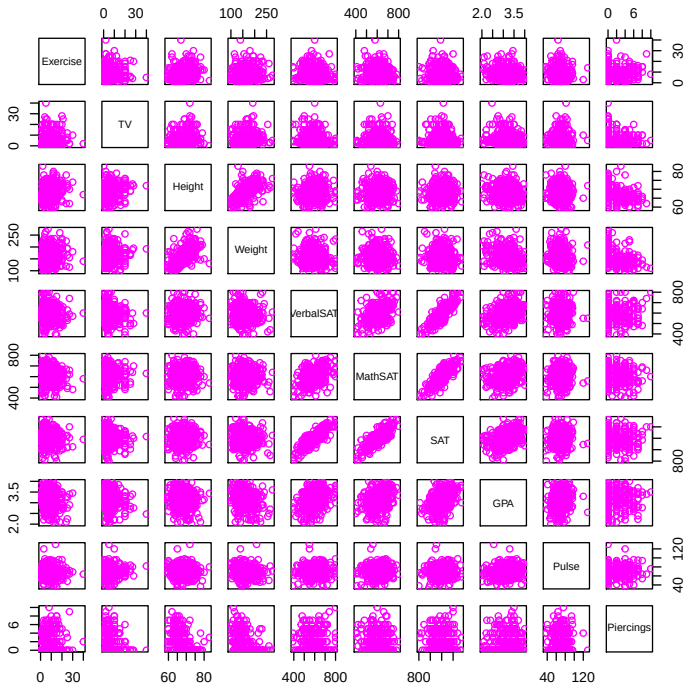

Figure 16: Pairs

En algunas oportunidades puede interesarnos visualizar datos para explorar posibles interacciones entre más variables. Para ello son útiles las gráficas bivariadas de tipo xyplot, que son muy flexibles y se implementan con el paquete lattice. Estos plots permiten realizar de manera muy sencilla gráficos múltiples condicionales como se muestra a continuación.

En xyplot se proporciona una fórmula que es utilizada para la realización del mismo. En el primer ejemplo se muestran los gráficos correspondientes a la fórmula condicional MathSAT VerbalSAT|Award, que significa que se harán gráficos de MathSAT con respecto a VerbalSAT para cada nivel de Award. En el segundo, la implementación con la fórmula Pulse Exercise|Gender.

```
> library(lattice)
> xyplot(MathSAT~VerbalSAT|Award)
```

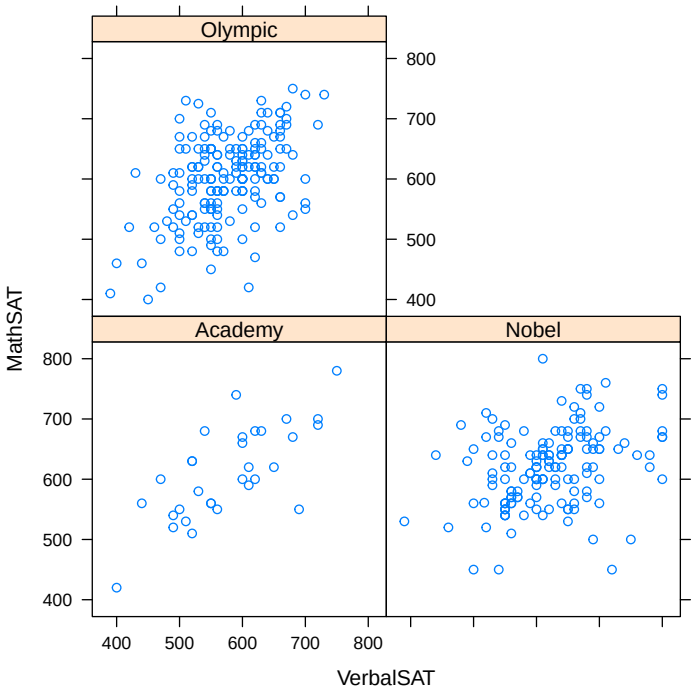

Figure 17: scatterplots condicionales: con xyplot()

```
> library(lattice)
> xyplot(Height~Weight|Gender,pch=16,col="darkblue")
```

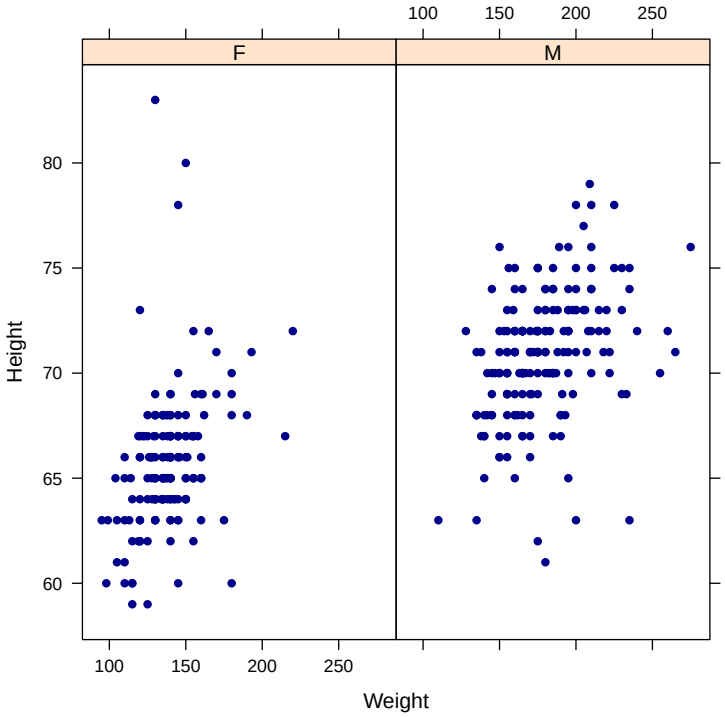

Figure 18: scatterplots condicionales: con xyplot()

Un par de ejemplos con un uso más sofisticado y aprovechando el potencial de xyplot:

```
> library(lattice)
> xyplot(MathSAT~VerbalSAT,groups=Award, pch=c(1,2,3),type="p",
+         col=c("red","blue","green"),key=list(columns=3,
+         text=list(levels(Award)),points=list(col=c("red","blue","green"),
+        pch=c(1,2,3))))
```

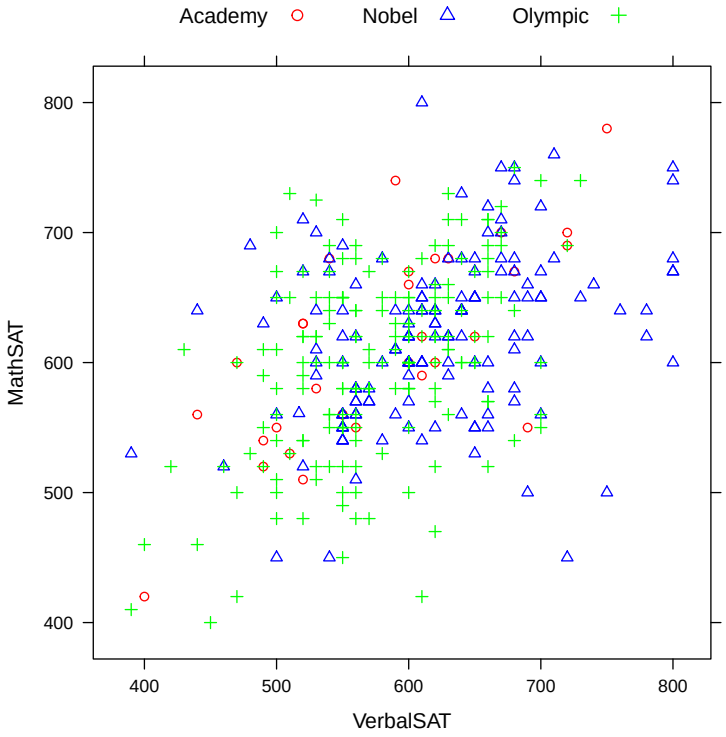

Figure 19: Scatterlot: con xyplot()

```
> library(lattice)
> xyplot(Height~Weight,groups=Gender, pch=c(16,16),type="p", col=c("red","blue"),
+        key=list(columns=2,text=list(levels(Gender)),points=list(col=c("red","blue")
+        pch=c(16,16))))
```

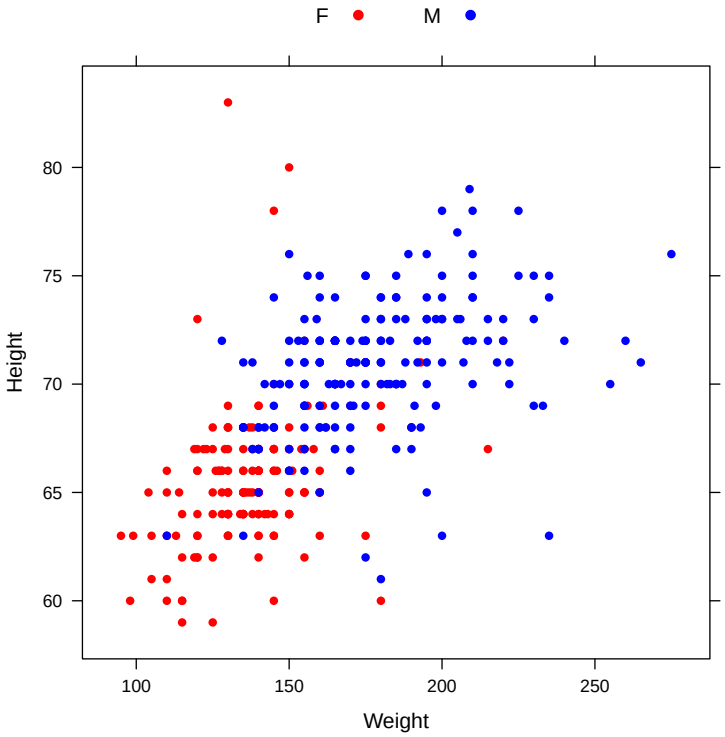

Figure 20: Scatterlot: con xyplot()

2 Otros gráficos

Veamos los comnandos outer, image y persp que sirven para representar datos tridimensionales

```
> x<-seq(-pi,pi,length =50)
> y<-x
> f<-outer(x,y, function(x,y) cos(y)/(1+x^2) )
```

```
> par(mfrow=c(1,3),bg="lightblue")
> contour(x,y,f)                # graficamos cortes de nivel
> contour(x,y,f,nlevels =15)
> contour(x,y,f, nlevels =45)
```

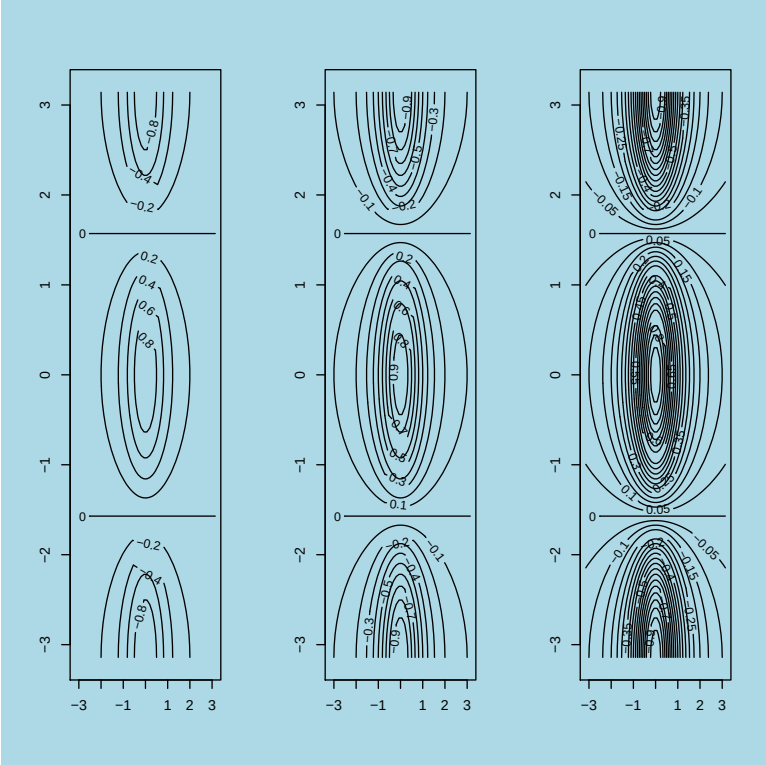

Figure 21: contour

```
> par(mfrow=c(2,2),bg="white")
> par(mar=c(1,1,1, 1))
> persp (x,y,f)
> persp (x,y,f , theta =30) #theta: angulo de direccion de vista
> persp (x,y,f , theta =90)
> persp (x,y,f , theta =90, phi =45) # phi: angulo de altura de vista
> par(mfrow=c(1,1),mar=c(5, 4, 4, 2) + 0.1) #valores de default
```

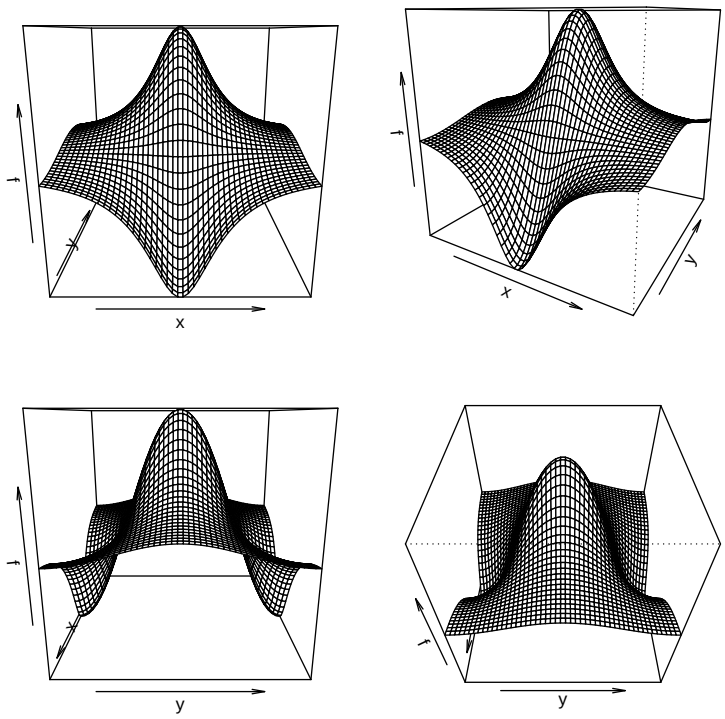

Figure 22: image y persp

Referencias

- Lock, R., Lock, P., Frazer Lock , K., Lock Morgan, E. and Lock, D. (2013). Statistics: unlocking the power of the data, Wiley.
- Paradis, E. (2003). R para principiantes. Disponible en *http* : *//cran.r − project.org/doc/contrib/rdebuts_es.pdf*
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). An Introduction

to Statistical Learning with Applications in R. Springer.

- Vasishth, A. (2014). An introduction to statistical data analysis (Summer 2014) Lecture note. Disponible en $http://www.ling.uni-potsdam.de/\sim vasishth/$
- .... y muchos más