Datos sobre mis alumnos preferidos en R

50 observaciones sobre su altura, peso, edad, género (1 hombre), número de hermanos y lateralidad (1 zurdo)

*** DATOS ***

Si bien en la web WebTC1 está la asignatura desarrollada en el entorno de programación **R**, a lo largo de las transparencias trabajaremos cómo afrontar la asignatura con éxito.

En Google es fácil encontrar manuales de ayuda generales sobre **R**, o específicos sobre la materia de la asignatura, uno de ellos podría ser el siguiente: R-intro-1.1.0-espanol.pdf.

Existen diversas formas de incorporar información a **R**. Puesto que normalmente los disponemos en formato *.txt* o *.xls* usaremos el comando *read.table* para incorporarlos (realmente esta orden sólo sirve con los primeros, pero guardar los segundos como los primeros es inmediato).

Para incorporar (y ver) los datos del ejemplo anterior es suficiente con escribir en la consola de **R**:

```
datos = read.table("https://www.ugr.es/local/romansg/material/WebTC1/TC1/datos.txt",
    header=T, sep=";")
head(datos)
```

Para obtener información sobre cualquier orden usar el comando *help*. Por ejemplo: *help*(*read.table*).

***

- **Altura**, **Peso** y **Edad** son variables cuantitativas de tipo continuo.
- **Hermanos** es una variable cuantitativa representada por una variable discreta con 6 modalidades.
- **Género** y **Lateralidad** son de tipo cualitativo, en escala nominal y presentan 2 modalidades.

Altura	Peso	Edad	Género	Hermanos	Lateralidad
172	60	261	0	2	2
167	65	261	0	1	1
174	75	288	0	1	2
169	72	360	0	2	2
178	78	360	1	3	2
196	95	288	1	3	2
160	50	288	0	2	2
168	60	264	0	1	1
172	74	288	1	1	2
181	72	276	1	5	2
180	70	300	0	3	2
181	89	288	1	0	2
188	100	312	1	2	2
170	60	254	0	2	2
171	73	242	1	1	2
160	57	263	0	2	2
180	65	258	0	1	2
177	95	338	1	1	1
175	67	288	0	4	2
186	82	288	1	2	2
165	59	300	0	1	2
180	97	311	1	2	1
174	68	264	0	2	2
180	75	294	1	2	2
178	71	336	1	1	2

Altura	Peso	Edad	Género	Hermanos	Lateralidad
179	95	264	1	1	1
180	85	276	1	2	2
180	82	270	1	1	2
173	60	324	0	2	2
165	61	252	0	2	2
185	91	345	0	1	2
165	53	285	0	3	2
170	60	252	0	1	2
174	67	255	0	3	2
165	52	259	0	1	2
168	60	264	0	1	1
172	74	288	1	1	2
181	72	276	1	5	2
180	70	300	0	3	2
181	89	288	1	0	2
188	100	312	1	2	2
170	60	254	0	2	2
181	89	288	1	0	2
180	97	311	1	2	1
174	68	264	0	2	2
180	75	294	1	2	2
178	71	336	1	1	2
179	95	264	1	1	1
180	85	276	1	2	2
180	82	270	1	1	2

*** Conceptos de ESTADÍSTICA DESCRIPTIVA ***

A continuación se codifican las variables cualitativas **Género** y **Lateralidad** de acuerdo a su definición, para a continuación crear un objeto del tipo *data.frame*, que permite trabajar tanto con números como con texto:

```
attach(datos)
Gen = ifelse(Genero == 1, "Hombre", "Mujer")
Late = ifelse(Lateralidad == 1, "Zurdo", "Diestro")

data = data.frame(Altura, Peso, Edad, Gen, Hermanos, Late)
head(data)
```

¿Para que sirve el comando *ifelse()*? ¿Y *attach()*?

***

Las tablas estadísticas de los dos caracteres cualitativos **Género** y **Lateralidad** son las que aparecen en las tablas siguientes:

```
cbind(table(Gen), prop.table(table(Gen)))
```

Tabla estadística para **Género**

a_i	n_i	f_i
Hombre (1)	24	0.48
Mujer (0)	26	0.52
Total	50	1

```
cbind(table(Late), prop.table(table(Late)))
```

Tabla estadística para **Lateralidad**

a_i	n_i	f_i
Zurdo (1)	8	0.16
Diestro (2)	42	0.84
Total	50	1

***

La tabla estadística del carácter cuantitativo **Número de hermanos** es:

x_i	n_i	N_i	f_i	F_i
0	3	3	0.06	0.06
1	19	22	0.38	0.44
2	19	41	0.38	0.82
3	6	47	0.12	0.94
4	1	48	0.02	0.96
5	2	50	0.04	1
Total	50		1	

```
n.i = table(Hermanos)
f.i = prop.table(n.i)
N.i = cumsum(n.i)
F.i = cumsum(f.i)
data.frame(n.i, f.i, N.i, F.i)
```

***

¿Qué se obtiene a partir del siguiente código?

```
cbind(table(Hermanos[Gen=="Hombre"]), prop.table(table(Hermanos[Gen=="Hombre"])))
range(Altura)
cbind(table(Hermanos[Altura<180]), prop.table(table(Hermanos[Altura<180])))
```

Esto se puede hacer gracias a *data.frame*.

***

La tabla estadística del carácter cuantitativo **Altura** es:

$l_{i-1} - l_i$	x_i	n_i	N_i	f_i	F_i	a_i	d_i
160 - 170	165	13	13	0.26	0.26	10	1.3
170 - 180	175	27	40	0.54	0.80	10	2.7
180 - 190	185	9	49	0.18	0.98	10	0.9
190 - 200	195	1	50	0.02	1	10	0.1
Total		50		1			

```
range(Altura)
int = cut(Altura, breaks = seq(160, 200, 10),
    include.lowest=T)
cbind(table(int), prop.table(table(int)))
```

$l_{i-1} - l_i$	x_i	n_i	N_i	f_i	F_i	a_i	d_i
160 - 165	162.5	6	6	0.12	0.12	5	1.2
165 - 170	167.5	7	13	0.14	0.26	5	1.4
170 - 175	172.5	10	23	0.2	0.46	5	2
175 - 180	177.5	17	40	0.34	0.8	5	3.4
180 - 185	182.5	6	46	0.12	0.92	5	1.2
185 - 190	187.5	3	49	0.06	0.98	5	0.6
190 - 195	192.5	0	49	0	0.98	5	0
195 - 200	197.5	1	50	0.02	1	5	0.2
Total		50		1			

```
range(Altura)
int = cut(Altura, breaks = seq(160, 200, 5),
    include.lowest=T)
cbind(table(int), prop.table(table(int)))
```

*** Medidas de POSICIÓN, DISPERSIÓN, FORMA y CONCENTRACIÓN ***

¿Cuáles son las medidas de posición de la variable **Altura**?

```
mean(Hermanos)
mean(Altura)
moda = table(int.Altura2)[which.max(table(int.Altura2))]
moda
median(Altura)
```

```
cuartiles = quantile(Altura, probs = seq(0.25, 0.75, 0.25))
cuartiles
deciles =  quantile(Altura, probs = seq(0.1, 0.9, 0.1))
deciles
percentiles =  quantile(Altura, probs = seq(0.01, 0.99, 0.01))
percentiles
```

¿Cómo se calcula la moda?

***

Con la siguiente función se puede obtener el percentil α de una variable no agrupada en intervalos:

```
percentil = function(x_i, n_i, alfa = 0.5){
        N_i = cumsum(n_i)
        n = length(N_i)
        alfa.n = alfa*N_i[n]
        cuartil = -5
        for (i in 1:n){
                if (N_i[i] == alfa.n){
                        cuartil = (x_i[i]+x_i[i+1])/2
                        break
                }
                if (N_i[i] > alfa.n){
                        cuartil = x_i[i]
                        break
                }
        }
        return(cuartil)
}
```

***

Medidas de dispersión:

```
Re = range(Altura)[2] - range(Altura)[1]
ReI = quantile(Altura, probs = 0.75) - quantile(Altura, probs = 0.25)

var(Altura) \# cuasi varianza
n = dim(data)[1] \\
((n-1)/n)*var(Altura) \# varianza

sd(Altura) \# cuasi desviacion tipica
sqrt(((n-1)/n)*var(Altura)) \# desviacion tipica

CA = range(Altura)[2]/range(Altura)[1]
ReR = (range(Altura)[2] - range(Altura)[1])/abs(mean(Altura))
CV = sd(Altura)/abs(mean(Altura))
```

***

Cálculo de media y varianza dado el par (x_i, n_i) de una tabla estadística:

```
x.i = c(162.5, 167.5, 172.5, 177.5, 182.5, 187.5, 192.5, 197.5)
x.i = (x.i-177.5)/5 # (cambio variable)
n.i = c(6, 7, 10, 17, 6, 3, 0, 1)
N = 50
xn.i = x.i*n.i
xmenosmedia = x.i - sum(xn.i)/N
xmenosmedia2 = xmenosmedia^2
xmenosmedia2n.i = xmenosmedia2*n.i
data.frame(x.i, n.i, xn.i, xmenosmedia, xmenosmedia2, xmenosmedia2n.i)
```

```
media = sum(xn.i)/N
media
varianza = sum(xmenosmedia2n.i)/N
varianza
coeficiente.variacion = sqrt(varianza)/abs(media)
coeficiente.variacion
```

Este código se puede adaptar para calcular características directamente sobre los datos y para cualquier momento centrado o no centrado:

```
x.i = c(1, 1, 2, 1, 2, 1, 2, 1, 2, 1)
x.i = c(3, 0, 2, 0, 3, 2, 2, 0, 0, 2)
N = length(x.i)
n.i = rep(1, N)
xn.i = x.i*n.i

s = 1 # media
xn.s = x.i^(s)
xn.s.i = xn.s*n.i

xmenosmedia = x.i - sum(xn.i)/N
r = 2 # varianza
xmenosmedia.r = xmenosmedia^r
xmenosmedia.r.n.i = xmenosmedia.r*n.i

data.frame(x.i, n.i, xn.s, xn.s.i, xmenosmedia, xmenosmedia.r, xmenosmedia.r.n.i)

momentonocentrado.r = sum(xn.s.i)/N
momentonocentrado.r
momentocentrado.r = sum(xmenosmedia.r.n.i)/N
momentocentrado.r
```

***

La siguiente función permite calcular cualquier momento centrado o no a partir del par (x_i, n_i) (variable no agrupada en intervalos):

```
momentos1 = function(x_i, n_i, r, mp = 0){
        N = sum(n_i)
        xmenos = x_i - mp
        momento_r = sum((xmenos^r)*n_i)/N
        return(momento_r)
}
```

***

Tabla para calcular coeficiente de Fisher a partir del par (x_i, n_i):

```
x.i = c(162.5, 167.5, 172.5, 177.5, 182.5, 187.5, 192.5, 197.5)
N = 50
n.i = c(6, 7, 10, 17, 6, 3, 0, 1)
media = sum(x.i*n.i)/N

xmenosmedia = x.i - media
varianza = sum(((x.i - media)^(2))*n.i)/N
r = 3
xmenosmedia.r = xmenosmedia^r
xmenosmedia.r.n.i = xmenosmedia.r*n.i

data.frame(x.i, n.i, xmenosmedia, xmenosmedia.r, xmenosmedia.r.n.i)

momentocentrado.r = sum(xmenosmedia.r.n.i)/N
```

```
momentocentrado.r
coef.Fisher = momentocentrado.r/((sqrt(varianza))^3)
coef.Fisher
```

Y el de Curtosis:

```
x.i = c(162.5, 167.5, 172.5, 177.5, 182.5, 187.5, 192.5, 197.5)
N = 50
n.i = c(6, 7, 10, 17, 6, 3, 0, 1)
media = sum(x.i*n.i)/N

xmenosmedia = x.i - media
varianza = sum(((x.i - media)^(2))*n.i)/N
r = 4
xmenosmedia.r = xmenosmedia^r
xmenosmedia.r.n.i = xmenosmedia.r*n.i

data.frame(x.i, n.i, xmenosmedia, xmenosmedia.r, xmenosmedia.r.n.i)

momentocentrado.r = sum(xmenosmedia.r.n.i)/N
momentocentrado.r
coef.Curtosis = momentocentrado.r/(varianza^2) - 3
coef.Curtosis
```

***

Momentos centrados y no centrados: cálculo a partir de los datos directamente:

```
momentos = function(x, r, mp = 0) {
    m = sum((x-mp)^r)/length(x)
    return(m)
}

variable = Hermanos
media = momentos(variable, 1)
media
varianza = momentos(variable, 2, mean(variable))
varianza
momentos(variable, 2) - momentos(variable, 1)^2 # varianza
desv.tip = sqrt(varianza)
desv.tip
CV = desv.tip/abs(media)
CV
m3 = momentos(variable, 3, mean(variable))
asimetria = m3/(desv.tip^3)
asimetria
m4 = momentos(variable, 4, mean(variable))
kurtosis = m4/(varianza^2) - 3
kurtosis
```

***

La siguiente función permite calcular el índice de Gini a partir del par (x_i, n_i) (variable no agrupada en intervalos):

```
Gini = function(x_i, n_i, salida = 0, h = 1){
       N = sum(n_i)
       N_i = cumsum(n_i)
       p_i = N_i/N
       xn_i = x_i*n_i
       u_i = cumsum(xn_i)
       q_i = u_i/u_i[length(x_i)]
       IG = 1 - (sum(q_i)-1)/(sum(p_i)-1)
```

```
        if (salida == 0){return(IG)}
        if (salida != 0){
                p = p_i[h]
                q = q_i[h]
                return(c(p,q))
        }
}
```

También permite obtener obtener un par (p_i, q_i) concreto.

***

Función para clacular TODO:

```
TODO = function(x, alfa = 0.95, graf = F) {
        n = length(x)
        media = momentos(x, 1)
        mediana = median(x)
        tabla.estadistica = table(x)
        posicion = which.max(table(x))
        moda = as.double(rownames(tabla.estadistica)[posicion])
        varianza = momentos(x, 2, mean(x))
        desv.tip = sqrt(varianza)
        Q1 = quantile(x, probs = 0.25)
        Q3 = quantile(x, probs = 0.75)
        percentil.alfa = quantile(x, probs = alfa)
        minimo = min(x)
        maximo = max(x)
        rango = maximo - minimo
        rango.int = as.numeric(Q3) - as.numeric(Q1)
        ca = maximo/minimo
        rango.re = rango/abs(media)
        cv = desv.tip/abs(media)
        pearson = (media-moda)/desv.tip
        m3 = momentos(x, 3, mean(x))
        simetria = m3/(desv.tip^3)
        m4 = momentos(x, 4, mean(x))
        kurtosis = m4/(varianza^2) - 3
        library(ineq)
        gini = ineq(x, type="Gini")
        if(graf == T) {plot(Lc(x), col="red", lwd=2)}
        salida = data.frame(n, media, mediana, moda, varianza, desv.tip, Q1, Q3, percentil.alfa,
            minimo, maximo, rango, rango.int, ca, rango.re, cv, pearson, simetria, kurtosis,
            gini)
        rownames(salida) = c("")
        return(salida)
}

    variable = Altura

        tapply(variable, Gen, mean)
        tapply(variable, Late, mean)
        tapply(variable, Gen, var)
        tapply(variable, Late, var)

        TODO(variable)
        tapply(variable, Gen, TODO)
        tapply(variable, Late, TODO)

        summary(variable)

    tabla = data.frame(t(TODO(Altura)), t(TODO(Peso)), t(TODO(Edad)), t(TODO(Hermanos)))
```

```
colnames(tabla) = c("Altura", "Peso", "Edad", "Hermanos")
round(tabla, digits = 3)
```

***

Medidas de asimetría:

```
d = rnorm(200)
plot(density(d), lwd=2, col="blue", xlab="", ylab="", main="") # simetrica
lines(density(c(rep(-4,1),rep(-3,2),rep(-2,5),rep(-1,10),rep(0,15),d+2)), lwd=2, col="darkgreen")
    # asimetrica izquierda
lines(density(c(d-2,rep(4,1),rep(3,2),rep(2,5),rep(1,10),rep(0,15))), lwd=2, col="red")
    # asimetrica derecha
```

Es interesante aprender más sobre *density* y *plot* mediante *help(plot)*.
Medidas de forma:

```
plot(density(rnorm(200,0,0.7)), lwd=2, col="darkgreen", xlab="", ylab="", main="") # leptocurtica
lines(density(rnorm(200)), lwd=2, col="blue") # mesocurtica
lines(density(rnorm(200,0,1.3)), lwd=2, col="red") # platicurtica
```

(la simetría es con respecto al cero)

***

La variable estadística **Altura** es asimétrica a la izquierda y leptocúrtica:

```
install.packages("moments") # solo una vez
library(moments)
skewness(Altura)
kurtosis(Altura) - 3
plot(density(Altura))
```

***

Tabla para calcular el Índice de Gini a partir del par (x_i, n_i) para el número de hermanos:

```
x.i = c(0,1,2,3,4,5)
N = 50
n.i = c(3,19,19,6,1,2)

N.i = cumsum(n.i)
N.i
p.i = N.i/N

xn.i = x.i*n.i
u.i = cumsum(xn.i)
q.i = u.i/u.i[length(x.i)]

data.frame(x.i, n.i, p.i, xn.i, u.i, q.i)

IG = 1 - (sum(q.i)-1)/(sum(p.i)-1)
IG
```

***

Calcular las características de concentración a partir de la tabla estadística es una tarea laboriosa, sin embargo a partir de **R** es suficiente con escribir:

```
install.packages("ineq") # solo una vez
library(ineq)
ineq(Altura, type="Gini")
plot(Lc(Altura), col="darkred", lwd=2)
```

En este caso el índice de Gini es muy próximo a cero por lo que la altura se reparte de forma equitativa entre todos los individuos.

*** Representaciones GRÁFICAS ***

Diagrama de sectores:

```
pie(table(Lateralidad), labels=Late, col=c("darkblue","lightblue"),
    main="Diagrama de Sectores de LATERALIDAD")

pie(table(Genero), labels=Gen, col=c("lightpink","lightblue"),
    main="Diagrama de Sectores de GENERO")
```

***

Diagrama de Barras:

```
barplot(table(Genero), col="red", xlab="Genero", main="Diagrama de Barras de GENERO")

barplot(prop.table(table(Gen)), col=c("lightblue","lightpink"), ylab="Frecuencias relativas",
    main="Diagrama de Barras de GENERO")

barplot(prop.table(table(Late)), col=c("red","orange"), ylab="Frecuencias relativas",
    main="Diagrama de Barras de LATERALIDAD")
```

***

Histograma:

```
hist(Altura)
hist(Altura, breaks=seq(160, 200, length=10), ylab="Frecuencias absolutas", col="red",
    main="Histograma de ALTURA")

hist(Altura, ylab="Frecuencias relativas", col="lightblue", freq=F, main="Histograma
    y curva de distribucion de ALTURA")
lines(density(Altura), col="blue", xlab="Altura", lwd=2)
```

***

Diagrama de tallo y hojas:

```
stem(Altura)
stem(Peso)
stem(Edad)
```

Diagrama Box-Whisker (Cajas y Bigotes):

```
boxplot(Altura, col="red", ylab="Altura", main="Diagrama Box-Whisker de ALTURA")
boxplot(Altura ~ Late, col=c("darkblue","lightblue"), ylab="Altura",
    main="Diagrama Box-Whisker de ALTURA")
```

*** Casualidad vs Causalidad (causa-efecto) ***

Para ilustrar la dicotomica casualidad/causalidad se generan a continuación de forma aleatoria dos conjuntos de datos altamente relacionados:

```
n = 100
e = rnorm(n)
a = rep(0.1, n)
b = rep(0.1, n)
for (i in 2:n){
    a[i] = 0.2*a[i-1] + e[i]
        b[i] = 0.5*b[i-1] + e[i]
}
```

```
plot(a, b, lwd=2, col="blue")
abline(v=0, lwd=2)
abline(h=0, lwd=2)
abline(a=0, b=1, lwd=2)

cor(a,b)
```

¿Se podría decir que uno aumenta *porque* el otro también lo hace?

*** Tablas de contingencia ***

Tablas de contingencia (tablas estadísticas bidimensionales):

```
table(Lateralidad, Genero)
table(Hermanos, Genero)

lim.intA = seq(min(Altura), max(Altura), length=5)
lim.intP = seq(min(Peso), max(Peso), length=5)
int.Altura = cut(Altura, breaks=lim.intA, include.lowest=TRUE)
int.Peso = cut(Peso, breaks=lim.intP, include.lowest=TRUE)
table(int.Altura, int.Peso)
prop.table(table(int.Altura, int.Peso))
table(int.Altura, Genero)
table(int.Peso, Lateralidad)

ftable(table(Lateralidad, Hermanos, Genero))
```

***

Distribuciones marginales y condicionadas:

```
install.packages("descr") # solo una vez
library(descr)
help(crosstab)

crosstab(Hermanos, Genero, plot=F)
crosstab(Lateralidad, Genero, plot=F)
crosstab(Lateralidad, Genero, plot=F, prop.t=T)
crosstab(int.Altura, int.Peso, plot=F)

crosstab(Hermanos[Genero==1], Genero[Genero==1], plot=F)
crosstab(Hermanos[Hermanos>1], Genero[Hermanos>1], plot=F)
crosstab(Hermanos[Altura>mean(Altura)], Genero[Altura>mean(Altura)], plot=F)
crosstab(Hermanos[(Genero==1)&(Hermanos<=3)&(Hermanos>=1)],
    Genero[(Genero==1)&(Hermanos<=3)&(Hermanos>=1)], plot=F)
```

*** Covarianza y coeficiente de correlación lineal ***

Correlación lineal alta positiva:

```
x = seq(-15,15,0.1)
p = rep(1, length(x))
for (i in 1:length(x)) {
    p[i] = sample(c(2.7, 2.9, 3.1, 3.3),1)*x[i] - sample(c(1.7, 1.9, 2.1, 2.3),1)
}
plot(x, p, lwd=2, col="blue", xlab="", ylab="")
abline(v=0, lwd=2)
abline(h=0, lwd=2)
```

Correlación lineal alta negativa:

```
x = seq(-15,15,0.1)
n = rep(1, length(x))
for (i in 1:length(x)) {
    n[i] = -sample(c(2.7, 2.9, 3.1, 3.3),1)*x[i] + sample(c(1.7, 1.9, 2.1, 2.3),1)
}
plot(x, n, lwd=2, col="blue", xlab="", ylab="")
abline(v=0, lwd=2)
abline(h=0, lwd=2)
```

Ausencia de relación lineal (y de cualquier otro tipo):

```
x = seq(-15,15,0.1)
nr = rep(1, length(x))
for (i in 1:length(x)) { nr[i] = rnorm(1) }
plot(x, nr, lwd=2, col="blue", xlab="", ylab="")
abline(v=0, lwd=2)
abline(h=0, lwd=2)
```

Ausencia de relación lineal pero relación cuadrática alta:

```
x = seq(-15,15,0.1)
rc = rep(1, length(x))
for (i in 1:length(x)) {
    rc[i] = sample(c(2.9, 3.1),1)*(x[i]^2) - sample(c(1.9, 2.1),1)
}
plot(x, rc, lwd=2, col="blue", xlab="", ylab="")
abline(v=0, lwd=2)
abline(h=0, lwd=2)
```

Covarianzas y correlaciones en cada uno de los casos anteriores:

```
cov(x, p)
cov(x, n)
cov(x, nr)
cov(x, rc)

cor(x, p) # correlacion lineal alta positiva
cor(x, n) # correlacion lineal alta negativa
cor(x, nr) # ausencia de relacion lineal
cor(x, rc) # ausencia de relacion lineal
```

***

¡Ojito con lo que hace el ordenador!
Recordemos que mediante *var*() calcula la cuasi-varianza, no la varianza. ¿Qué se calculará con *cov*()?
Con la siguiente función se clacula cualquier momento bidimensional centrado o no centrado:

```
momentos.b = function(x, y, r, s, mp.x = mean(x), mp.y = mean(y)){
        m = sum(((x-mp.x)^r)*((y-mp.y)^s))/length(x)
        return(m)
}
```

La covarianza se obtendría según $momentos.b(x, y, 1, 1)$.
Podemos observar que con *cov*() se calcula la cuasi-covarianza:

```
x = c(161,203,235,176,201,188,228,211,191,178)
y = c(159,206,241,163,197,193,209,189,169,201)

momentos.b(x,y,1,1)
cov(x,y)
((length(x)-1)/length(x))*cov(x,y)
```

Sin embargo, en el caso de la correlación simple, *cor*() sí obtiene el verdadero valor:

```
momentos.b(x,y,1,1)/sqrt(momentos(x,2,mean(x))*momentos(y,2,mean(y)))
cor(x,y)
```

***

¡Ojito con lo que hace el ordenador! (bis)

```
cuantitativas = cbind(Altura, Peso, Edad, Hermanos)
pairs(cuantitativas)
cor(cuantitativas)

cor(Peso, Genero)
cor(Altura, Lateralidad)
```

*** Regresión Lineal ***

Recta de regresión de **Peso** sobre **Altura** de andar por casa:

```
regresion = function(y, x, recta = T){
        media.y = mean(y)
        media.x = mean(x)
        n = length(x)
        varianza.x = momentos(x, 2, mp=media.x)
        covarianza = momentos.b(x, y, 1, 1)
        pendiente = covarianza/varianza.x
        origen = media.y - pendiente*media.x
        resultados = data.frame(media.y, covarianza, varianza.x, media.x)
        ifelse (recta == T, paste("y =", pendiente, "* x +", origen), return(resultados))
}

regresion(Peso, Altura, T)
regresion(Peso, Altura, F)
```

Combinando las funciones *momentos*(), *momentos.b*() y *regresion*():

```
BID = function(x, y, graf = T){
        covarianza = momentos.b(x,y,1,1)
        correlacion = momentos.b(x,y,1,1)/sqrt(momentos(x,2,mean(x))*momentos(y,2,mean(y)))
        recta = regresion(y, x, T)
        if (graf == T){plot(x, y, lwd=2, col="blue")}
        salida = data.frame(covarianza, correlacion, recta)
        rownames(salida) = c("")
        return(salida)
}

BID(Peso, Altura, T)
BID(Altura, Peso, F)
```

***

Aunque a continuación aparece información que aún no sabremos interpretar (todo se andará), desde un punto de vista profesional...

Recta de regresión de **Peso** sobre **Altura**:

```
reg1 = lm(Peso~Altura)
coef(reg1)[[1]]+coef(reg1)[[2]]*180 # prediccion

plot(Altura, Peso, col="brown", lwd=2)
abline(reg1, col="blue", lwd=2)
summary(reg1)
```

Recta de regresión de **Altura** sobre **Peso**:

```
reg2 = lm(Altura~Peso)
coef(reg2)[[1]]+coef(reg2)[[2]]*77 # prediccion

plot(Peso, Altura, col="brown", lwd=2)
abline(reg2, col="blue", lwd=2)
summary(reg2)
```

*** Regresión NO Lineal ***

Se simulan datos y vemos qué ajuste es mejor, lineal o no lineal: en cada caso se generan unas series de observaciones para la variable x. En función del tipo de modelo se combinan con los valores concretos $a = 5 = b$, se le añade un valor de error e para obtener la variable y y así evitar el ajuste perfecto.

***

Hipérbola:

```
obs = 100
e = rnorm(obs,0,0.1)
a = 5
b = 5
x = seq(1,obs)

w = 1/x
y = a + b*w + e
plot(x, y, xlab="", ylab="", col="blue", lwd=2)
abline(lm(y~x), col="red", lwd=2)

summary(lm(y~x))
summary(lm(y~w))
```

***

Potencial:

```
obs = 100
e = rnorm(obs,0,0.1)
a = 5
b = 5
x = seq(1,obs)

y = a*(x^b) + e
plot(x, y, xlab="", ylab="", col="blue", lwd=2)
abline(lm(y~x), col="red", lwd=2)

summary(lm(y~x))
z = log(y, base=exp(1))
w = log(x, base=exp(1))
summary(lm(z~w))
exp(coefficients(lm(z~w))[[1]])
```

***

Exponencial:

```
obs = 100
e = rnorm(obs,0,0.1)
a = 5
b = 5
x = seq(1,obs)/10

y = a*(b^x) + e
plot(x, y, xlab="", ylab="", col="blue", lwd=2)
abline(lm(y~x), col="red", lwd=2)

summary(lm(y~x))
z = log(y, base=exp(1))
summary(lm(z~x))
exp(coefficients(lm(z~x)))
```

*** Todo en UNO ***

```
TC1 = function(x, y, graf = T){
        todo.x = TODO(x)
        todo.y = TODO(y)
        bid = BID(x, y, graf)
        salida = list(todo.x, todo.y, bid)
        names(salida) = c("Descriptivos X", "Descriptivos Y", "Bidimensional")
        return(salida)
}

TC1(Peso, Altura)
```

*** FIN ***