Actividad: el ajuste a una recta

En la actividad "Ajusta la recta", discutimos sobre cuál es la recta que mejor se ajusta o aproxima a una nube de puntos. Los conceptos básicos que se han tratado son:  
\* El ajuste.  
\* La recta de mínimos cuadrados.  
\* El residual.  
\* La bondad del ajuste.  
Ahora vamos a revisar esos conceptos, pero echando mano de los recursos que nos proporciona el programa R.

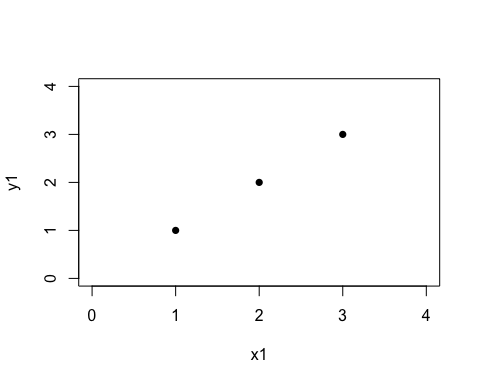
# Diagrama A

En primer lugar, vamos a introducir los valores que se presentan en cada diagrama. Para ello vamos a definir dos variables. En el primer diagrama las vamos a llamar **x1** e **y1**.  
Tenemos que anotar los valores de cada punto en las respectivas variables. El primer punto es (1,1), por lo que hay que anotar el valor 1 tanto en x1 como en 1; y así sucesivamente.  
Recordemos que para asignar valores a una variable, en R usamos el comando c().

x1 <- c(1,2,3)  
y1 <- c(1,2,3)

A continuación vamos a reproducir el diagrama de dispersión, usando la orden plot(). Una opción que se ha incluido es la que limita el número de valores que se presentan en cada eje. También se ha incluido una opción para representar los valores como puntos.

plot(x1,y1, xlim=c(0,4), ylim=c(0,4), pch=16)



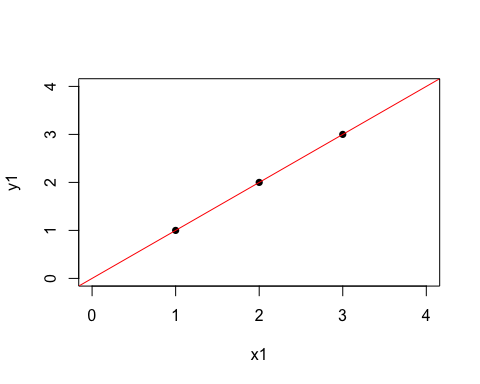
En R disponemos de un comando, **lm** (de *linear model*), para ajustar unos valores a una recta de regresión. Para escribir la orden, tienes que usar el símbolo "~" (tilde). Fíjate que en primer lugar se escribe la variable dependiente, y detrás de la tilde, la variable independiente. Para obtener la tilde, en Windows debes usar la combinación de teclas AltGr+4; en Mac, ese signo lo obtienes pulsando alt+Ñ.

lm(y1 ~ x1)

##   
## Call:  
## lm(formula = y1 ~ x1)  
##   
## Coefficients:  
## (Intercept) x1   
## 0 1

El valor que aparece bajo la etiqueta "Intercept" corresponde a la *ordenada en el origen*, (el parámetro "a" en la ecuación y=a+bx). El segundo valor corresponde a la pendiente de la recta.  
Por último, vamos a representar la recta que hemos ajustado en el paso anterior, usando la orden **abline**. Combinando esta orden con el resultado de la anterior, que nos proporciona los parámetros de la recta de mínimos cuadrados, dibujamos la **recta de regresión**. Para diferenciar bien la recta, incluimos la opción *col* para que la dibuje en rojo.

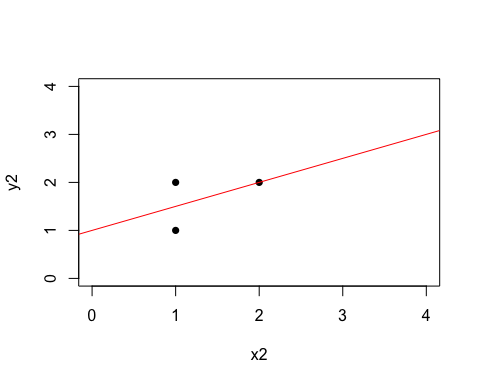
abline(lm(y1 ~ x1), col="RED")



# Diagrama B

En este segundo diagrama vamos a repetir los mismos pasos que en el anterior, pero atendiendo a que la distribución de los valores en las dos variables. Para distinguirlas del caso anterior vamos a llamarlas **x2** e **y2**.

x2 <- c(1,1,2)  
y2 <- c(1,2,2)  
plot(x2,y2, xlim=c(0,4), ylim=c(0,4), pch=16)  
abline(lm(y2 ~ x2), col="RED")



Para obtener los parámetros de la recta ajustada escribiremos la orden correspondiente:

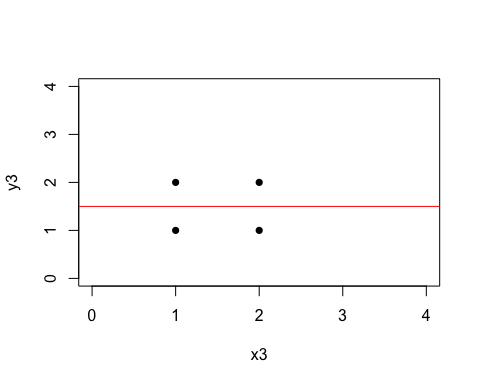
lm(y2 ~ x2)

A diferencia del diagrama A, en este caso el ajuste no es perfecto. La recta ajustada pasa por el punto medio de x e y.

# Diagrama C

Al igual que en los dos diagramas anteriores, vamos a definir dos nuevas variables para representar a los valores; en este caso las llamaremos **x3** e **y3**.

x3 <- c(1,1,2,2)  
y3 <- c(1,2,1,2)  
plot(x3,y3, xlim=c(0,4), ylim=c(0,4), pch=16)  
abline(lm(y3 ~ x3), col="RED")



Para obtener los parámetros de la recta ajustada escribiremos la orden correspondiente:

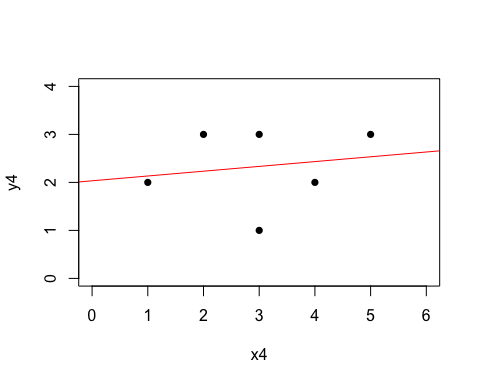
lm(y3 ~ x3)

En este diagrama, la pendiente de la recta es 0, lo que indica que no podemos ajustar una relación lineal entre las dos variables.

# Diagrama D

Como ya hemos hecho antes, definimos las dos variables **x4** e **y4**, representamos la distribución y ajustamos la recta.

x4 <- c(1,2,3,3,4,5)  
y4 <- c(2,3,1,3,2,3)  
plot(x4,y4, xlim=c(0,6), ylim=c(0,4), pch=16)  
abline(lm(y4 ~ x4), col="RED")



Para obtener los parámetros de la recta ajustada escribiremos la orden correspondiente:

lm(y4 ~ x4)

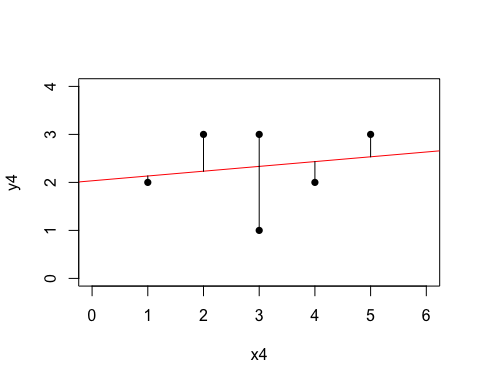
En este ejemplo vamos a estudiar en más profundidad el concepto de *residual*, que es la distancia entre el valor observado y el que predice la recta que hemos ajustado. En R contamos con la orden predict()que nos calcula los valores de la recta que le corresponden a cada valor de x. Escribimos los valores en una variable que llamamos \*\*y4\_p\*.

y4\_p <- predict(lm(y4 ~ x4))  
y4\_p

## 1 2 3 4 5 6   
## 2.133333 2.233333 2.333333 2.333333 2.433333 2.533333

El valor que corresponde al primer valor de x4 es 2,13, y el valor observado es 2.0. Si queremos representar esa distancia entre los dos valores, usamos la orden lines(), en la que tenemos que indicar las coordenadas de la línea. La orden se escribe así: lines(c(1,1),c(2,2.13)). Pero podemos hacerlo para todos los valores, introduciendo una nueva orden for:

plot(x4,y4, xlim=c(0,6), ylim=c(0,4), pch=16)  
abline(lm(y4 ~ x4), col="RED")  
for (i in 1:6) lines(c(x4[i],x4[i]),c(y4[i],y4\_p[i]))



¿Por qué se llama *recta de minimos cuadrados*? Si sumamos los residuales de esta recta (y lo mismo ocurriría con cualquier otra recta), el valor obtenido es:

sum(y4\_p-y4)

## [1] 3.996803e-15

Bueno, ese valor no es igual a cero por cuestiones de redondeo. Pero prácticamente se puede tomar como tal. Y ese es el problema, que los residuales negativos se compensan con los que tienen signo positivo, y siempre se anulan. Para medir el grado en el que una recta se ajusta a unos valores, lo que hacemos es elevar al cuadrado esos residuales, y obtenemos la **varianza residual**. De ahí viene el nombre del método: *cuadrados*, porque elevamos al cuadrado a los residuales, y *mínimos*, porque corresponde a la recta que está más cerca de los puntos estudiados.

sum((y4\_p-y4)^2)

El valor que se obtiene es la suma de los cuadrados de las distancias a la recta.

# Órdenes de R

En esta actividad has aprendido a usar las siguientes órdenes de R:  
abline() Representa gráficamente la recta que se ajusta a una nube de puntos.  
c() Asigna una serie de datos a una variable.  
lm() Estima los parámetros (pendiente y ordenada en el origen) de la recta de mínimos cuadrados.  
plot() Representa gráficamente los valores de una o más variables.  
predict() Estima los valores individuales a partir de una recta de mínimos cuadrados.  
sum() Suma todos los valores almacenados en una variable.