Trabajo 1

Alejandro García Montoro

20 de marzo de 2016

Generalización y visualización de datos

Ejercicio 1

La siguiente función genera una lista de `N` vectores. Cada uno de ellos contiene una muestra de tamaño `dim` de una distribución uniforme entre los valores especificados por `rango`.

La implementación es sencilla: como `runif` —el generador aleatorio de muestras de una distribución uniforme— recibe el número de muestras que se desean tomar, basta aplicar esta función al número `dim` tantas veces como indique `N`. Esto se consigue con la función `lapply`, que genera una lista del mismo tamaño que la list que se le pasa como primer argumento, ejecutando entonces la función pasada como segundo argumento sobre cada uno de los valores de la lista.

```
simula_unif <- function(N, dim, rango){
    lapply(rep(dim, N), runif, min = rango[1], max = rango[2])
}
```

Ejercicio 2

La siguiente función genera una lista de `N` vectores. Cada uno de ellos contiene una muestra de tamaño `dim` de una distribución gaussiana de media 0 y desviación típica correspondiente al elemento especificado en el vector `sigma`.

La implementacióne es igual que la anterior, cambiando la función `runif` por `rnorm`.

```
simula_gauss <- function(N, dim, sigma){
    lapply(rep(dim, N), rnorm, mean = 0, sd = sigma)
}
```

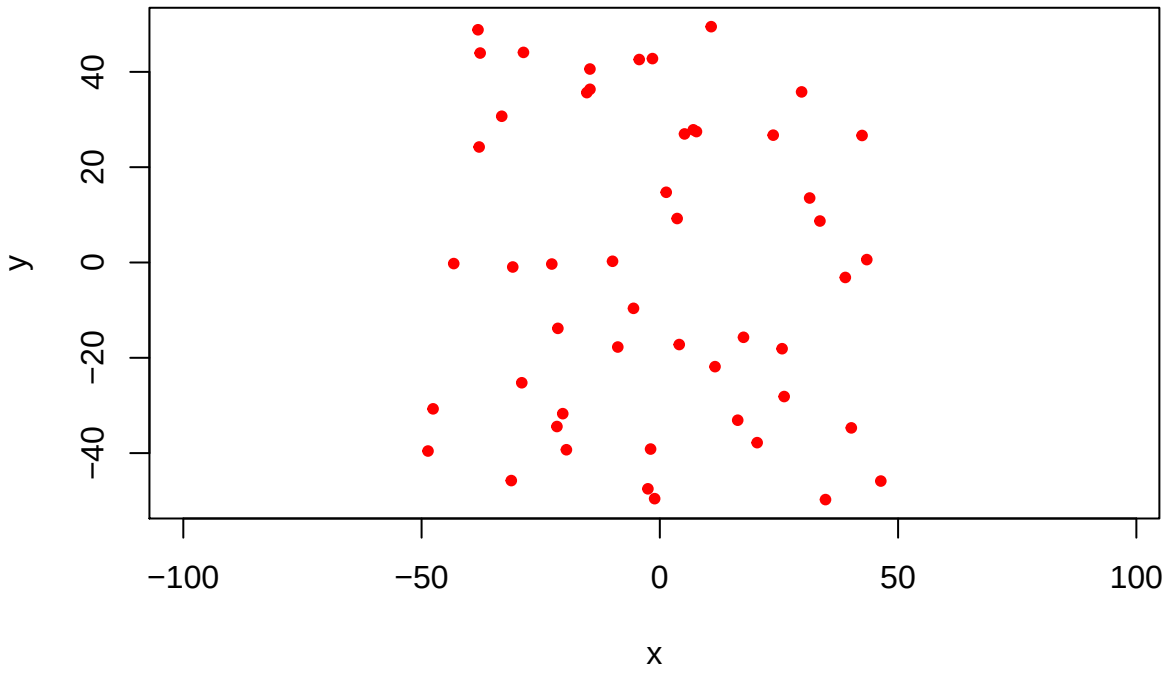
Ejercicio 3

Primero generamos los datos solicitados, usando la función anterior y tomando como `x` (resp. `y`) los primeros (resp. segundos) valores de cada vector generado:

```
datos_unif <- simula_unif(50, 2, c(-50,50))
x <- unlist(lapply(datos_unif, '[[', 1))
y <- unlist(lapply(datos_unif, '[[', 2))
```

Para el dibujo de la gráfica usamos la función `plot`:

```
plot(x, y, col = 'red', pch = 20, asp=1)
```

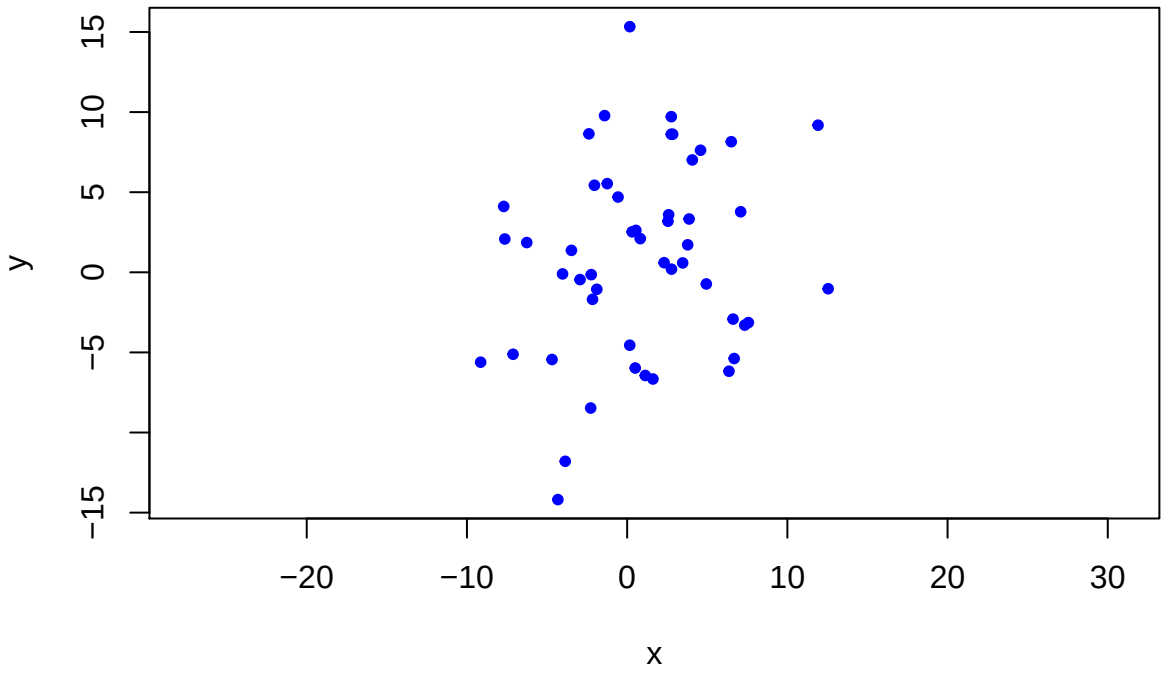


Ejercicio 4

```
datos_norm <- simula_gauss(50, 2, c(5,7))
x <- unlist(lapply(datos_norm, '[[', 1))
y <- unlist(lapply(datos_norm, '[[', 2))
```

Para la generación de la gráfica necesitamos conocer el intervalo en el que se mueven los datos. La función **range** nos devuelve justo eso, y con su resultado podemos definir los intervalo de cada eje:

```
plot(x, y, xlim = range(x), ylim = range(y), asp = 1, col = 'blue', pch = 20)
```



Ejercicio 5

```
simula_recta <- function(intervalo){
    # Simulamos dos puntos dentro del cuadrado intervalo x intervalo
    punto1 <- runif(2, min=intervalo[1], max=intervalo[2])
    punto2 <- runif(2, min=intervalo[1], max=intervalo[2])

    # Generamos los parámetros que definen la recta
    a <- (punto2[2] - punto1[2]) / (punto2[1] - punto1[1])
    b <- -a * punto1[1] + punto1[2]

    # Devolvemos un vector concatenando ambos parámetros
    c(a,b)
}
```

Ejercicio 6

Para este ejercicio y el siguiente vamos a crear una estructura abstracta que genera funciones etiquetadoras. Para ello, definimos `generador_etiquetados`, una función que al ser llamada con otra función `f` como parámetro, devuelve una función etiquetadora; es decir, una función que devuelve 1 o -1 según el signo que toma la función `f` al recibir los parámetros `x` e `y`:

```
generador_etiquetados <- function(f){
    function(x,y){
        sign(f(x,y))
    }
}
```

Resolver el ejercicio con la estructura anterior es ahora más sencillo: simplemente tenemos que definir la función $f(x, y) = y - ax - b$, donde a y b son los parámetros que definen la recta $y = ax + b$.

Por tanto, basta simular una recta haciendo uso de la función implementada anteriormente y definir la función `f1`, que será la que pasaremos al generador de funciones etiquetadoras:

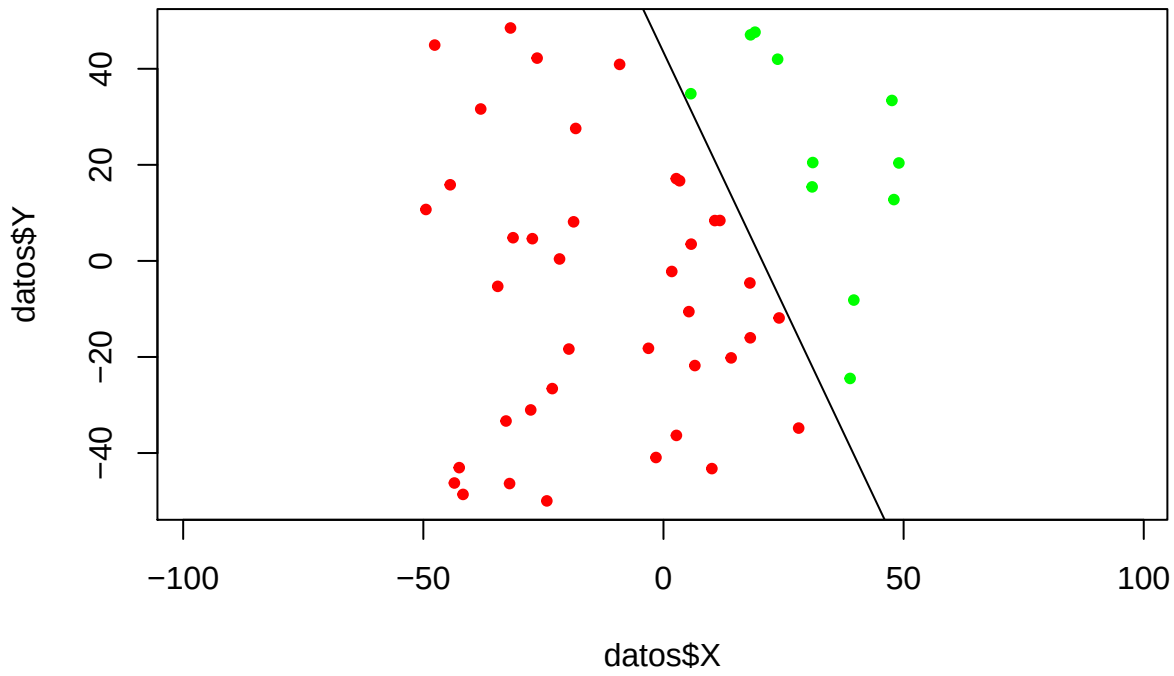
```
recta <- simula_recta(c(-50,50))

f1 <- function(x,y){
    y - recta[1]*x - recta[2]
}

etiquetado1 <- generador_etiquetados(f1)
```

Por último, generamos la gráfica solicitada simulando datos de una distribución uniforme: extraemos los datos de las ordenadas, los de las abscisas y generamos los etiquetados con la función `etiquetado1`:

```
datos_unif <- simula_unif(50, 2, c(-50,50))

x <- unlist(lapply(datos_unif, '[[', 1))
y <- unlist(lapply(datos_unif, '[[', 2))

datos <- data.frame(X = x, Y = y, Etiqueta = etiquetado1(x,y))
```

Por último, generamos la gráfica con `plot`, asignando un color diferente según el etiquetado, y añadiendo la gráfica de la recta simulada:

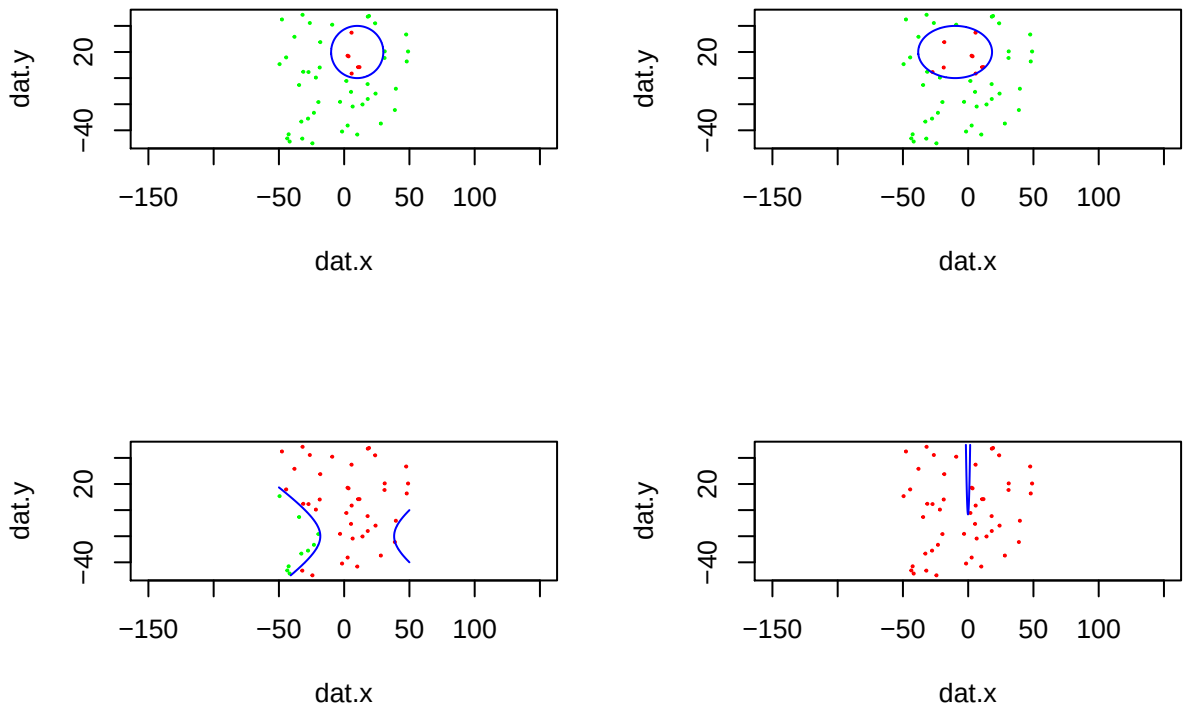
```
colores <- ifelse(datos$Etiqueta == 1, "green", "red")

plot(datos$X, datos$Y, xlim = range(datos$X), ylim = range(datos$Y), asp = 1, col = colores, pch = 20)
abline(rev(recta))
```



Ejercicio 7

```
f2 <- function(x,y){
    (x-10)**2 + (y-20)**2 - 400
}

f3 <- function(x,y){
    0.5*(x+10)**2 + (y-20)**2 - 400
}

f4 <- function(x,y){
    0.5*(x-10)**2 - (y+20)**2 - 400
}

f5 <- function(x,y){
    y - 20*x**2 - 5*x + 3
}
```

```
genera_grafico <- function(dat.x, dat.y, f, colores, ...){
    etiquetado <- generador_etiquetados(f)
    etiquetas <- etiquetado(dat.x, dat.y)

    if(missing(colores))
        colores <- ifelse(etiquetas == 1, "green", "red")

    plot(dat.x, dat.y, asp = 1, col = colores, pch = 20, ...)

    f.x <- seq(-50, 50, length=1000)
    f.y <- seq(-50, 50, length=1000)
    f.z <- outer(f.x, f.y, f)
    contour(f.x, f.y, f.z, levels=0, col = "blue", add=T, drawlabels=F)
}

par(mfrow=c(2, 2))

genera_grafico(x, y, f2, cex=0.25)
genera_grafico(x, y, f3, cex=0.25)
genera_grafico(x, y, f4, cex=0.25)
genera_grafico(x, y, f5, cex=0.25)
```

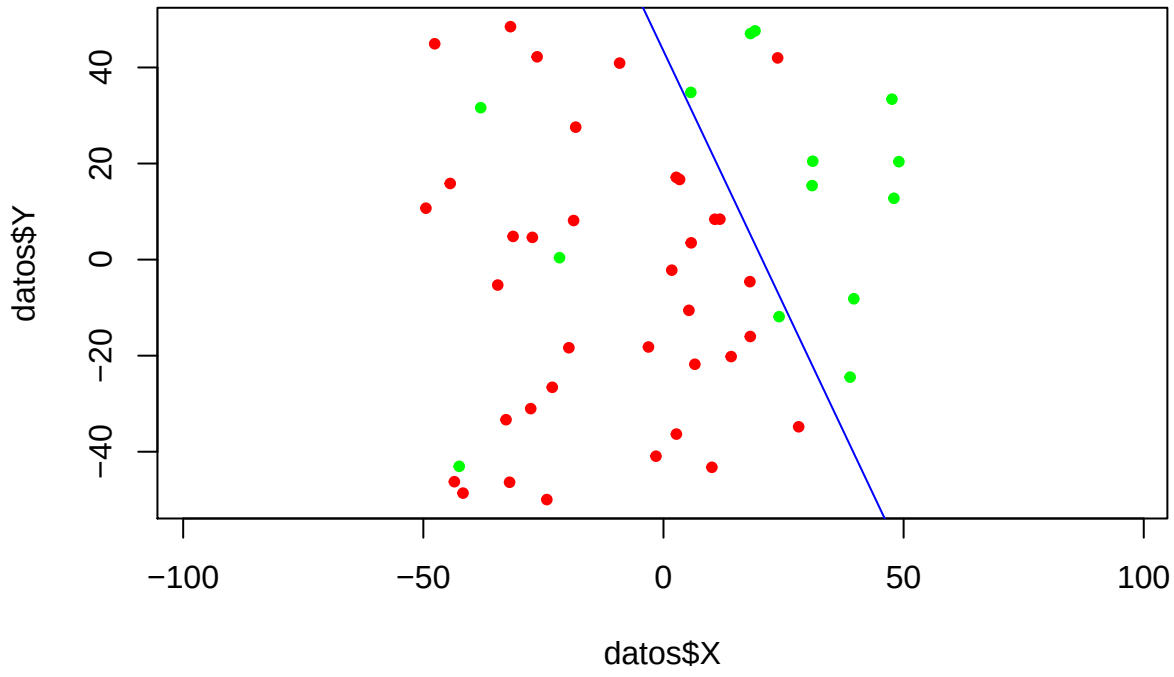



```
par(mfrow=c(1, 1))
```

Ejercicio 8

```
# Get indices of the labelled data, divided into positive and negative
indices_pos <- which(datos$Etiqueta %in% 1)
indices_neg <- which(datos$Etiqueta %in% -1)

# Get a random sample of those indices, of size 10%
cambiar_pos <- sample(indices_pos, round(0.1*length(indices_pos)))
cambiar_neg <- sample(indices_neg, round(0.1*length(indices_neg)))

# Change those positions
datos$Etiqueta[cambiar_pos] <- -1
datos$Etiqueta[cambiar_neg] <- 1
```

```
colores <- ifelse(datos$Etiqueta == 1, "green", "red")

plot(datos$X, datos$Y, xlim = range(datos$X), ylim = range(datos$Y), asp = 1, col = colores, pch = 20)
abline(rev(recta), col="blue")
```




```
genera_grafico(x, y, f2, colores)
```