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)
}
```

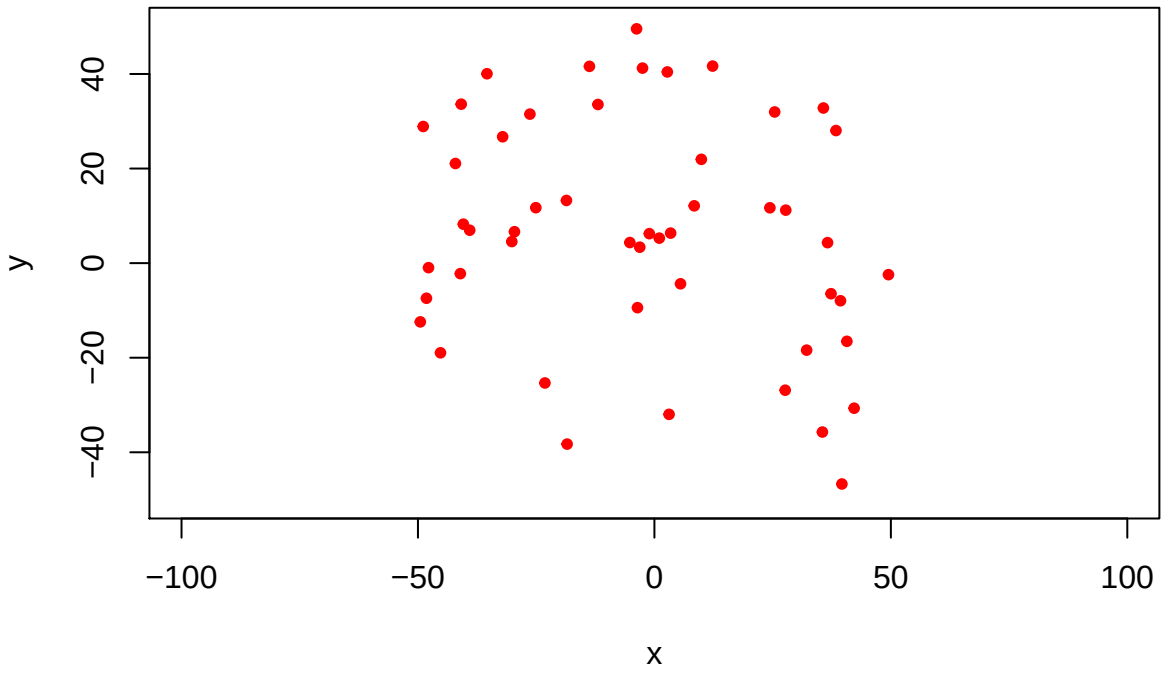
Ejercicio 3

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, xlim = c(-50, 50), ylim = c(-50,50), asp = 1, col = 'red', pch = 20)
```

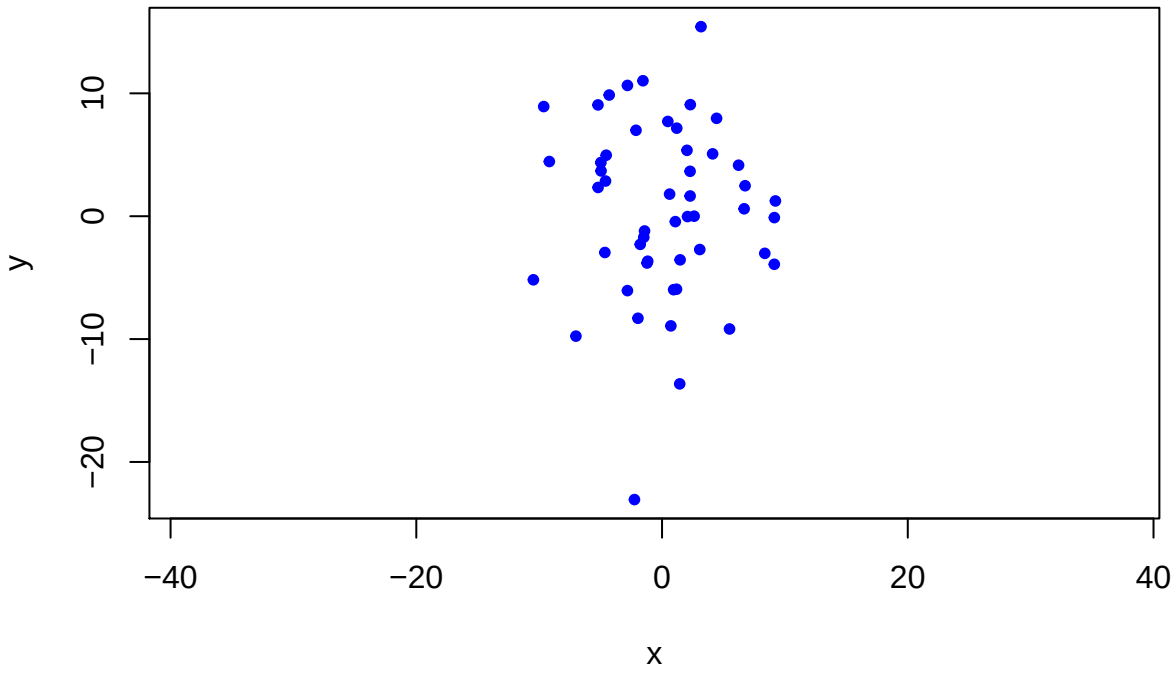


Ejercicio 4

```
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)
```


y
10
0
−10
−20
−40
−20
0
20
40
x