# BIOINFORMÁTICA
# Práctica 4.5
# Introducción al lenguaje R y a Bioconductor

Juan Miguel García Gómez
(juanmig@upvnet.upv.es)
Departamento de Física Aplicada
Facultad de Informática-UPV

January 31, 2011

## Contents

**1 Objetivo de la práctica** **1**

**2 Material de la práctica** **1**

**3 Memoria de la práctica** **2**

**4 Ejercicios** **2**
4.1 Ejercicio. Introducción a R . . . . . 2
4.1.1 Objetivo . . . . . 2
4.1.2 Desarrollo . . . . . 2
4.2 Ejercicio. Bioconductor . . . . . 14
4.2.1 Objetivo . . . . . 14
4.2.2 Desarrollo . . . . . 14

## 1 Objetivo de la práctica

R (http://www.r-project.org) es un software estadístico conjunto a un lenguaje de programación para manipulación de datos, cálculo y representación gráfica.

El proyecto Bioconductor (http://www.bioconductor.org) se integra completamente con R, proveyendo paquetes R adicionales para el análisis de datos procedentes de las ciencias de la biología de y la salud.

Esta sesión práctica pretende introducir al estudiante en los aspectos concretos del lenguaje R, centrando nuestro interés en su uso en Bioinformática, para lo cual, utilizaremos también librerias del proyecto Bioconductor.

## 2 Material de la práctica

- R project (http://www.r-project.org)

- Bioconductor project (http://www.bioconductor.org)

# 3 Memoria de la práctica

La memoria de la práctica será el script "practica4.5.R" que contenga las instrucciones de los diversos ejercicios.

Para ello, en lugar de introducir directamente las instrucciones en el interprete de comandos de R editaremos con cualquier programa de textos (emacs, wordpad, vim) el fichero "practica4.5.R". Ejecutaremos el contenido del fichero desde el interprete de R con:

> `source("practica4.5.R")`

# 4 Ejercicios

## 4.1 Ejercicio. Introducción a R

### 4.1.1 Objetivo

- Utilizar instrucciones básicas de R con ejemplos sencillos
- Manejar los mecanismos de ayuda del entorno R
- Importar datos desde ficheros externos
- Manejar los tipos de datos en R con las funciones proporcionadas por el entorno de programación
- Realizar los primeros scripts y funciones en el lenguaje de programación R
- Utilizar las herramientas gráficas de R

### 4.1.2 Desarrollo

1. La asignación `<-`, el listado de objetos `ls()` y destrucción de objetos `rm()`

```
> v1 <- 10.56
> v2 <- 3 + rnorm(1)
> vector1 <- c(-2, -2, -2)
> matriz1 <- matrix(c(v1, v2, 3, 4, 5, 6), nrow = 2, ncol = 3)
> matriz1

          [,1] [,2] [,3]
[1,] 10.560000    3    5
[2,]  4.569625    4    6

> matriz1[1, 3] <- -1
> matriz1[2, ] <- vector1
> matriz1 <- edit(matriz1)
> ls()

[1] "matriz1" "v1"      "v2"      "vector1"
```

```
> ls(pat = "v")

[1] "v1"      "v2"      "vector1"
```

`pat="v"` restringe el listado a los objetos cuyo nombre comienza por "v"

```
> ls.str()

matriz1 :  num [1:2, 1:3] 10.6 -2 3 -2 -1 ...
v1 :  num 10.6
v2 :  num 4.57
vector1 :  num [1:3] -2 -2 -2
```

`ls.str()` muestra los tipos de los objetos en memoria

```
> rm(v1)
> ls()

[1] "matriz1" "v2"      "vector1"

> rm(list = ls(pat = "v"))
> gc()

         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 137819  7.4     350000 18.7   350000 18.7
Vcells 132750  1.1     786432  6.0   457873  3.5

> ls()

[1] "matriz1"
```

`rm` borra el objeto de memoria, `gc` invoca al recolector de basura para liberar memoria

2. Leyendo y escribiendo datos desde y en archivos

```
> frame1 <- read.table(file = "datosPrueba.txt")
> frame2 <- frame1[1:3, 1:10]
> write.table(frame2, file = "datosReducido.txt", sep = " ", row.names = FALSE,
+     col.names = FALSE)
> file.show("datosReducido.txt")
```

3. Vectores

```
> x <- c(1.1, 2.3, 3.3, 4.4, 5.5, 6.6, 7.7)
> x[2]

[1] 2.3

> x[2:3]

[1] 2.3 3.3

> x[-2]
```

```
[1] 1.1 3.3 4.4 5.5 6.6 7.7
```

```
> sample(x, 5)
```

```
[1] 1.1 4.4 5.5 2.3 3.3
```

```
> sort(runif(100, min = 1, max = 5))
```

```
  [1] 1.005142 1.013009 1.014868 1.124146 1.266444 1.348073 1.393488 1.423938
  [9] 1.430730 1.451128 1.468596 1.558176 1.632592 1.639639 1.662389 1.740828
 [17] 1.762429 1.800782 1.909147 1.985292 2.001893 2.064599 2.092695 2.110992
 [25] 2.147288 2.173517 2.306898 2.336242 2.382775 2.429869 2.436721 2.473624
 [33] 2.487135 2.523328 2.540463 2.612706 2.631163 2.751928 2.807518 2.815615
 [41] 2.870863 2.874293 2.878741 2.898576 2.932669 3.079450 3.182069 3.205907
 [49] 3.223584 3.277571 3.319206 3.324553 3.372879 3.407806 3.408125 3.410449
 [57] 3.412214 3.432522 3.458411 3.540836 3.548588 3.589997 3.612446 3.616743
 [65] 3.625418 3.637544 3.648968 3.701140 3.770734 3.777829 3.782413 3.793452
 [73] 3.820862 3.842990 3.850774 3.970259 4.074853 4.145161 4.172634 4.182829
 [81] 4.226968 4.256337 4.271406 4.298623 4.312971 4.363400 4.365384 4.386706
 [89] 4.421708 4.542711 4.543883 4.564489 4.570876 4.585925 4.625311 4.635516
 [97] 4.659352 4.750345 4.830256 4.989801
```

```
> x <- rep(1:10, times = 2)
> x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10
```

```
> unique(x)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

4. Cambio de tipos

```
> x <- as.integer(runif(100, min = 1, max = 5))
> xc <- as.character(x)
> xn <- as.numeric(xc)
> x <- as.integer(runif(100, min = 1, max = 5))
> x[x == 1] <- "A"
> x[x == 2] <- "T"
> x[x == 3] <- "G"
> x[x == 4] <- "C"
> paste(x, sep = "", collapse = "")
```

```
[1] "CGCATAGCCATTTCTCTCGGTGGGTGCTTGGTCTGTTCGGTATGAACGAGGTGCGACTATCAACCCCAGCACCAGTCGCC
```

5. Búsquedas y comparaciones en conjuntos

```
> letters
```

```
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
```

```
> letters == "c"
```

```
 [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE
```

```
> which(rep(letters, 2) == "c")
```

```
[1]  3 29
```

```
> match(c("c", "g"), letters)
```

```
[1] 3 7
```

```
> x <- rep(1:10, 2)
> y <- c(2, 4, 6)
> x %in% y
```

```
 [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
[13] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
> intersect(month.name[1:4], month.name[3:7])
```

```
[1] "March" "April"
```

```
> setdiff(month.name[1:4], month.name[3:7])
```

```
[1] "January"  "February"
```

```
> union(month.name[1:4], month.name[3:7])
```

```
[1] "January"  "February" "March"    "April"    "May"      "June"     "July"
```

```
> x <- c(month.name[1:4], month.name[3:7])
> x[duplicated(x)]
```

```
[1] "March" "April"
```

```
> unique(x)
```

```
[1] "January"  "February" "March"    "April"    "May"      "June"     "July"
```

6. Factores

   Los factores son vectores usados para especificar agrupamientos (clases) de de los componentes de vectores de la misma longitud. Las clases de los factores son denominados niveles ”levels”.

```
> diagnostico <- factor(c("glioblastoma", "glioblastoma", "metastasis",
+     "metastasis", "glioblastoma", "meningioma", "meningioma"))
> diagnostico
```

```
[1] glioblastoma glioblastoma metastasis   metastasis   glioblastoma
[6] meningioma   meningioma
Levels: glioblastoma meningioma metastasis
```

La función `table` calcula las frecuencias para clases de un factor.

```
> diagnosticofr <- table(diagnostico)
> diagnosticofr

diagnostico
glioblastoma   meningioma   metastasis
           3            2            2
```

La función `tapply` ejecuta una función `mean` sobre los elementos del vector `expresion` agrupados por el factor `diagnostico`.

```
> expresion <- c(10.5, 11.4, 5.2, 3.2, 9.3, 1.2, 4.2)
> tapply(expresion, diagnostico, mean)

glioblastoma   meningioma   metastasis
        10.4          2.7          4.2
```

7. Matrices

```
> x <- matrix(1:30, 3, 10, byrow = T)
> x <- matrix(1:30, 10, 3, byrow = F)
> x[c(1:5), 3]

[1] 21 22 23 24 25

> mean(x[c(1:5), 3])

[1] 23

> x[9, 2] <- 50
> dim(x)

[1] 10  3

> dim(x) <- c(3, 5, 2)
```

8. Listas

```
> expresion <- c(10.5, 11.4, 5.2, 3.2, 9.3, 1.2, 4.2)
> diagnostico <- factor(c("glioblastoma", "glioblastoma", "metastasis",
+     "metastasis", "glioblastoma", "meningioma", "meningioma"))
> corpus <- list(variable = expresion, clase = diagnostico)
> corpus$variable

[1] 10.5 11.4  5.2  3.2  9.3  1.2  4.2

> corpus[[1]]

[1] 10.5 11.4  5.2  3.2  9.3  1.2  4.2

> corpus$clase
```

```
[1] glioblastoma glioblastoma metastasis   metastasis   glioblastoma
[6] meningioma   meningioma
Levels: glioblastoma meningioma metastasis
```

```
> corpus[[2]][1]
```

```
[1] glioblastoma
Levels: glioblastoma meningioma metastasis
```

9. Data frames: tablas de datos y tratamiento de datos perdido (NA)

```
> frame2 <- data.frame(y1 = rnorm(12), y2 = rnorm(12), y3 = rnorm(12),
+     y4 = rnorm(12))
> rownames(frame2) <- month.name[1:12]
> names(frame2) <- c("y4", "y3", "y2", "y1")
> dim(frame2)
```

```
[1] 12  4
```

```
> data.frame(frame2[, 2:4])
```

```
                  y3          y2          y1
January   -1.7167389 -0.45640649  1.10290833
February  -0.5614042 -0.55187164 -0.44109658
March      0.1265098  0.77557361  1.39666774
April     -0.8633006  0.04484643  1.78202605
May       -0.7193731  1.00906560  0.22534800
June       0.1384573  0.10927022 -0.31906008
July      -1.3048358 -0.95546792  0.51510754
August     0.5039372 -0.49474464  1.21199322
September  1.8166845 -0.11568180 -0.09057895
October   -0.5145130  0.10879788  0.54429213
November  -0.1586320  1.74486772  1.05962296
December   0.6845057 -0.58378848 -0.97073137
```

```
> frame2[1:5, 1:2]
```

```
                y4         y3
January  -0.9784822 -1.7167389
February -0.4226205 -0.5614042
March    -1.1529539  0.1265098
April     0.7233967 -0.8633006
May      -0.9168725 -0.7193731
```

```
> frame2["August", ]
```

```
              y4        y3         y2       y1
August -1.107122 0.5039372 -0.4947446 1.211993
```

```
> summary(frame2)
```

```
       y4                y3                y2                y1
 Min.   :-1.1530   Min.   :-1.7167   Min.   :-0.95547   Min.   :-0.9707
 1st Qu.:-0.9323   1st Qu.:-0.7554   1st Qu.:-0.50903   1st Qu.:-0.1477
 Median :-0.3078   Median :-0.3366   Median :-0.03542   Median : 0.5297
 Mean   :-0.1642   Mean   :-0.2141   Mean   : 0.05287   Mean   : 0.5014
 3rd Qu.: 0.7028   3rd Qu.: 0.2298   3rd Qu.: 0.27585   3rd Qu.: 1.1302
 Max.   : 1.3593   Max.   : 1.8167   Max.   : 1.74487   Max.   : 1.7820
```

```
> mean(frame2)

         y4          y3          y2          y1
-0.16420981 -0.21405858  0.05287171  0.50137492
```

```
> cor(frame2)

            y4          y3         y2          y1
y4  1.00000000 -0.14356995 0.07387835 -0.08533176
y3 -0.14356995  1.00000000 0.07730228 -0.36968919
y2  0.07387835  0.07730228 1.00000000  0.30119150
y1 -0.08533176 -0.36968919 0.30119150  1.00000000
```

```
> x <- 1:10
> x <- x[1:12]
> frame3 <- data.frame(x, y = 12:1)
> is.na(x)

 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
> sapply(frame3, mean, na.rm = T)

  x   y
5.5 6.5
```

```
> apply(frame3, 1, mean, na.rm = T)

 [1] 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 2.0 1.0
```

```
> frame3[is.na(frame3), 1] <- 0
> apply(frame3, 1, mean, na.rm = T)

 [1] 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 1.0 0.5
```

10. Creación de una función

```
> funcionEjemplo <- function(argumento1, argumento2, argumento3 = 100.5) {
+     media <- mean(c(argumento1, argumento2, argumento3))
+     mensaje <- paste("La media de:", argumento1, ",\n\t", argumento2,
+         "y", argumento3, "es", media)
+     print(mensaje)
+     return(media)
+ }
> r <- funcionEjemplo(10, 50, 60)
```

```
[1] "La media de: 10 ,\n\t 50 y 60 es 40"
```

```
> r
```

```
[1] 40
```

```
> r <- funcionEjemplo(10, 50)
```

```
[1] "La media de: 10 ,\n\t 50 y 100.5 es 53.5"
```

```
> r
```

```
[1] 53.5
```

11. Manejo de directorios

    `dir` lista el contenido del directorio de trabajo, `getwd` devuelve el directorio de trabajo, `setwd` establece el directorio de trabajo

```
> dir()
```

```
[1] "45IntroRyBioC"     "45IntroRyBioC.Rnw" "45IntroRyBioC.tex"
[4] "datosPrueba.txt"   "datosReducido.txt" "Sweave.sty"
```

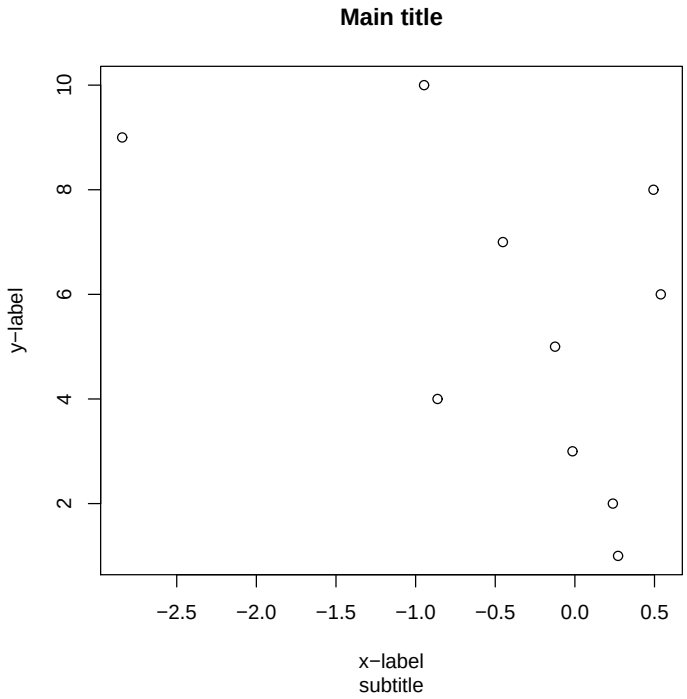
```
> setwd("..")
> getwd()
```

```
[1] "/Users/yossua54/Docencia/bioinformatica/bioinformatica20102011/practicas"
```

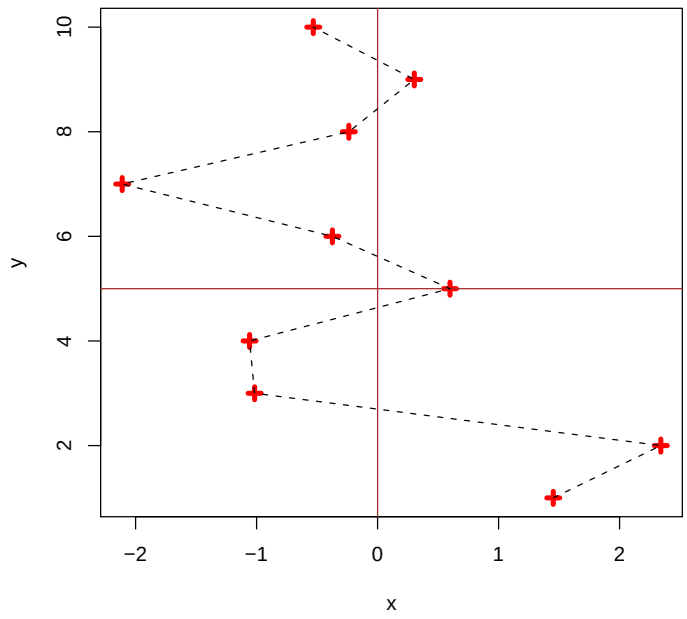
```
> setwd("45IntroduccionRyBioC")
```

12. Representaciones gráficas

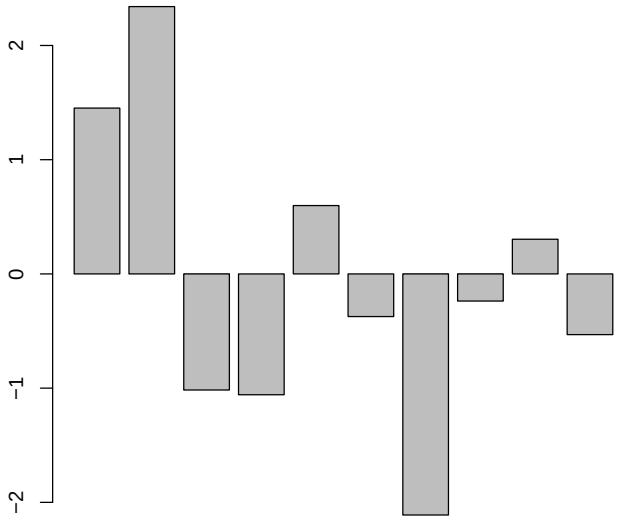
```
> x <- rnorm(10)
> y <- 1:10
> z <- 10:1
> plot(x, y, main = "Main title", sub = "subtitle", xlab = "x-label",
+     ylab = "y-label")
```



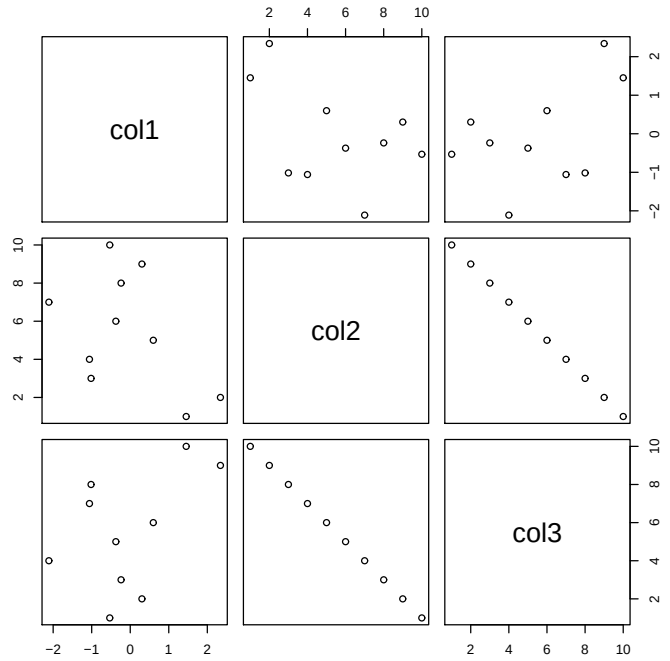

```
> plot(x, y, type = "p", col = "red", lwd = 4, pch = 3)
> abline(h = 5, col = "brown")
> abline(v = 0, col = "brown")
> lines(x, y, lty = 2)
```



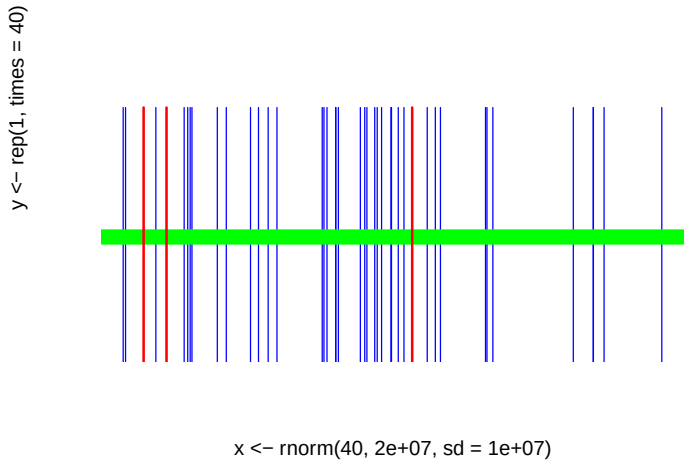

```
> barplot(x)
```



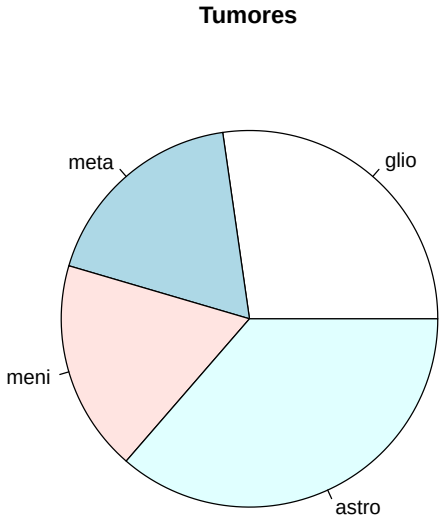

```
> plot(data.frame(col1 = x, col2 = y, col3 = z))
```




```
> plot(x <- rnorm(40, 2e+07, sd = 1e+07), y <- rep(1, times = 40),
+     type = "h", col = "blue", xaxt = "n", yaxt = "n", bty = "n")
> abline(h = 0.78, col = "green", lwd = 12)
> lines(a <- rnorm(5, 2e+07, sd = 1e+07), b <- rep(1, times = 5),
+     type = "h", col = "red", lwd = 2)
```




```
> pie(c(glio = 3, meta = 2, meni = 2, astro = 4), main = "Tumores")
```




```
> jpeg("frecuenciaTumores.jpeg")
```

```
> pie(c(glio = 3, meta = 2, meni = 2, astro = 4), main = "Tumores")
> dev.off()

quartz
     2
```

## 4.2 Ejercicio. Bioconductor

### 4.2.1 Objetivo

Ejecutaremos únicamente un ejemplo con las librerías de Bioconductor, para ilustrar su utilidad específica en bioinformática.

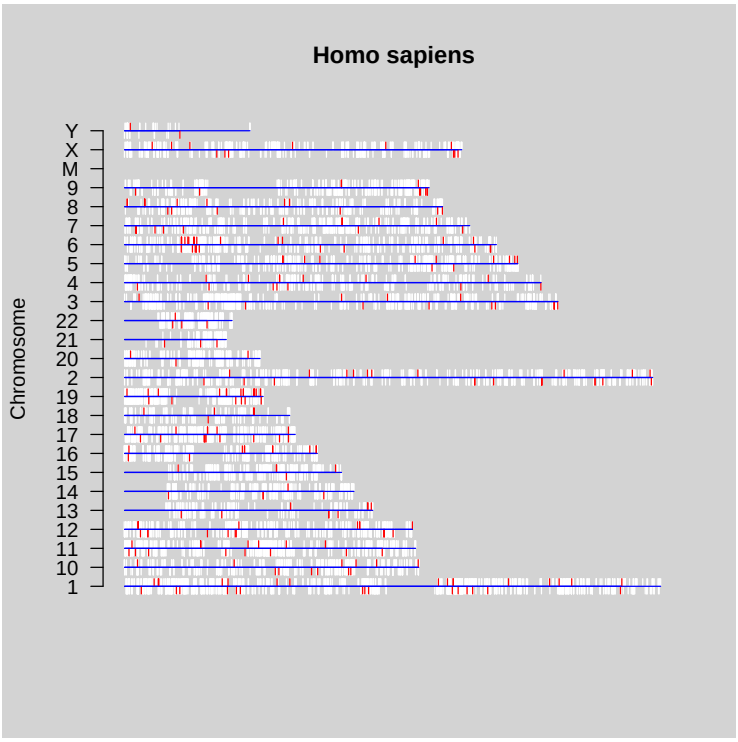
### 4.2.2 Desarrollo

Realizaremos la representación gráfica de las posiciones de los genes en los cromosomas del homo-sapiens. Destacando en rojo aquellos genes estudiados en la base de datos `eset`.

La librería `annotate` contiene, entre otras, información sobre la localización de los genes en los cromosomas y la librería `geneplotter` incluye la función `cPlot` que dibuja el objeto de localización de los genes.

La librería `hgu95av2.db` contiene, entre otras funciones y datos, información sobre la localización de los probes para los genes humanos. Por su parte, `data:sample.ExpressionSet` de la librería `Biobase` contiene una base de datos real anonimizada de 26 microarrays affymetrix U95v2 y 500 genes estudiados.Visualizaremos parte de la información cruzada de ambos paquetes.

```
> library(annotate)
> library(geneplotter)
> library("hgu95av2.db")
> newChrom <- buildChromLocation("hgu95av2.db")
> cPlot(newChrom)
> data(sample.ExpressionSet)
> cColor(featureNames(sample.ExpressionSet), "red", newChrom)
```




```
> cPlot(newChrom, c("2"), fg = "yellow", scale = "relative")
> cColor(featureNames(sample.ExpressionSet), "red", newChrom)
```