¿Qué es R?

R (http://www.r-project.org/) es un potente paquete de análisis numérico y estadístico. Es completamente open-source y está disponible para Linux, Windows y Mac. Por supuesto también puedes descargar el código y compilarlo.

En su instalación normal nos proporciona una consola sobre la que trabajar, en la que procesamos los datos, mostrándose lo gráficos que hagamos en una ventana independiente.

Instalación

Para instalar **R** en ubuntu

```
sudo apt-get install r-base
```

(Son unos 100Mb de instalación).
Una vez instalado podemos abrir la consola ejecutando

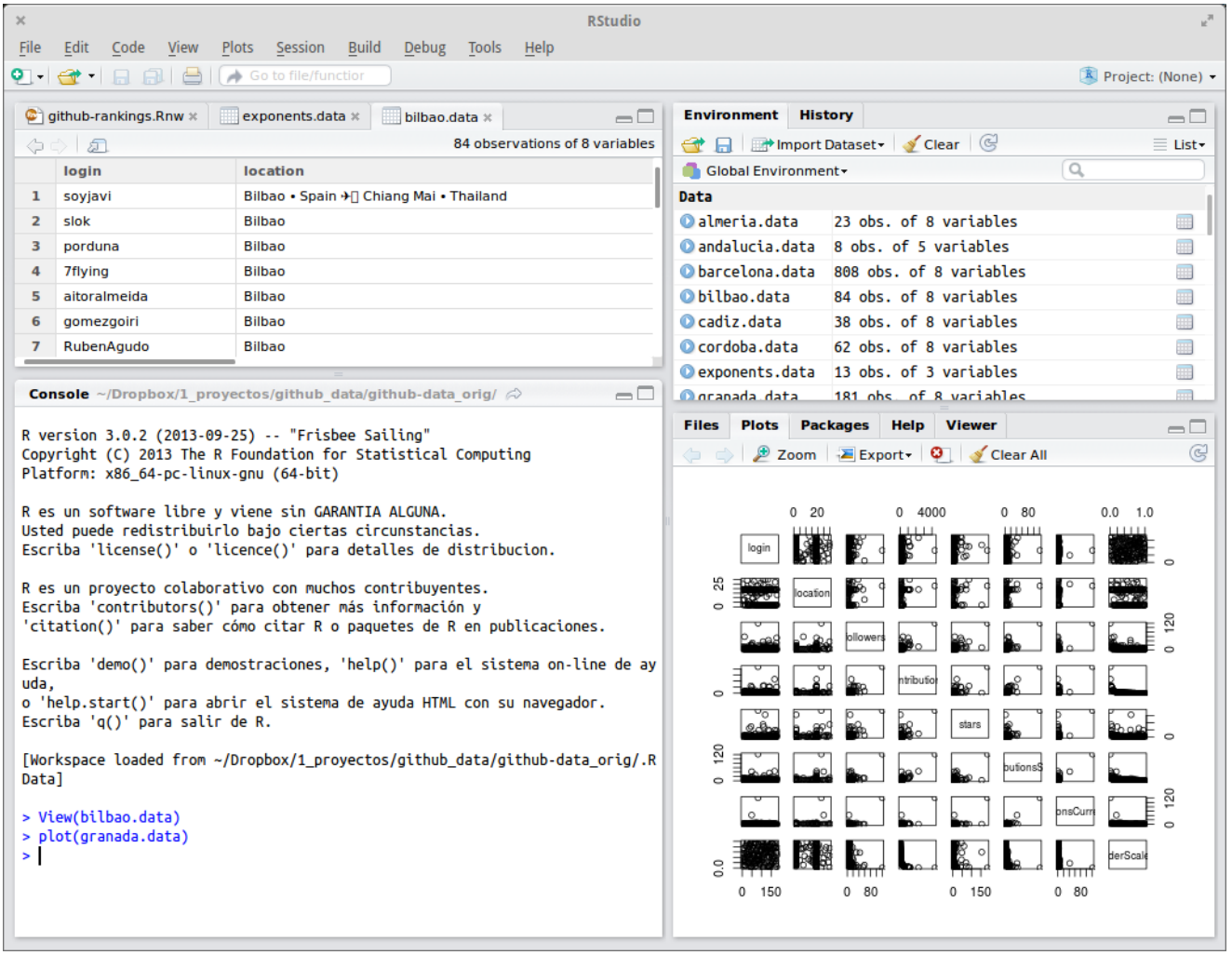
```
R
```

Podemos abrir la ayuda en un navegador sin más que ejecutar **help.start()**

Entornos más *amigables*

Como hemos comentado, R es una herramienta interativa, podíamos decir que un intérprete, con el que interaccionamos mediante una consola. Todas las instrucciones que le vamos dando se guardan en un ficherro denominado .Rhistory que se crea en el directorio desde el que lo hemos arrando. Por esto podemos tener varios ficheros .Rhistory. Personalmente uso este hecho para guardar los pasos que voy dando en diferentes proyectos.

Si eres más de ratón que de teclado puedes usar RStudio (www.rstudio.com) que nos proporciona un entorno visual, multi documento donde es fácil ver los objetos (datos) que tenemos disponibles, los diferentes comandos empleados, los gráficos realizados, etc.



Captura de rstudio donde se ven sus posibilidades de uso

CookBook

En este cajón de sastre iré poniendo lo que vaya aprendido antes de estructurarlo.

- **q()** Salir de R
- Cargar un fichero en formato csv
  granada.data <- read.csv('user-data-Granada.csv',sep=';')
- **summary(granada.data)** Resumen de los datos
- Para instalar un package (por ejemplo ggplot2 para hacer gráficos)

  ```
  install.package("ggplot2")
  ```

  Si no lo estamos ejecutando como root (lo cual está bien), nos preguntará si queremos utilizar un repositorio de paquetes local. Seleccionamos la carpeta (por defecto ~/R/) y se descargará el paquete, se compilará (no he probado lo que ocurre si no tenemos

instalado el entorno de compilación) y ya podemos usarlo. Más detalles (http://www.r-bloggers.com/installing-r-packages/)

- **objects()** muestra los datasets disponibles
- Nos referimos a una columna de un dataset con **dataset$columna** (como lo haríamos en excel, upsss)
- Cuando lo usamos se genera un fichero llamado **.Rhistory** con los distintos comandos, y ¿ cuando abrimos R desde una carpeta con un fichero así tenemos accesible el histórico de comandos
- Si usamos **plot(dataset)** se representan todos los gráficos posibles de cada columna con todas las demas. En la diagonal aparecen los nombres de esa columna/fila. plot (http://stat.ethz.ch/R-manual/R-devel/library/graphics/html/plot.html) permite dibujar también funciones.

```
fun1 <- function(x) sin(cos(x)*exp(-x/2))
plot (fun1, -8, 5)
```

Podemos añadir varios gráficos usando el argumento **add=TRUE**

```
fun2 <- function(x) sin(sin(x)*exp(-x/2))
plot (fun2, -8, 5,add=TRUE)
plot (fun1, -8, 5,add=TRUE)
```

Para seleccionar el color usarmos **col='red'**

- Para representar un gráfico hacemos **plot(dataset$columanY,dataset$columnaY)** donde podemos usar cualquier operación matemática
- Podemos añadir una columna sin más que utilizar **dataset$NuevaColumna<-funcion(otros datos)** . Por ejemplo

```
granada.data$orderScaled<-seq_along(granada.data$contributions)/length(gran
```

- **scale(x)** devuelve una columna con los valores centrados en la media scale (https://stat.ethz.ch/R-manual/R-patched/library/base/html/scale.html)
- Para normalizar una columna al rango 0,1 hacemos fuente (http://stackoverflow.com/questions/15468866/scaling-a-numeric-matrix-in-r-with-values-0-to-1):

```
apply(m, MARGIN = 2, FUN = function(X) (X - min(X))/diff(range(X)))
```

- Para normalizar una columna que es una secuencia entre 1 y n basta con hacer

```
granada.data$orderScaled<-seq_along(granada.data$contributions)/length(gran
```

- Exportar **write.csv(dataset, "filename.csv")** para formato csv o **write.table(dataset, "filename.txt, sep="\t")** para exportar a otro formato

- Importar **dataset<-read.csv('fichero.csv',sep=';')** en un dataset

```
madrid.data <- read.csv('user-data-Madrid.csv',sep=';')
```

- Podemos hacer una **regresión lineal** (linear model) usando **lm(waiting ~ duration)** y representar la línea con abline(lm(waiting ~ duration)) detalles (http://msenux.redwoods.edu/math/R/regression.php) abline (https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/abline.html). Podemos hacer que no se transforme ¿según los ejes? con **untf='true'**

- Para ajustar por mínimos cuadrados **lsfit (https://stat.ethz.ch/R-manual/R-patched/library/stats/html/lsfit.html)** más detallado (http://sites.stat.psu.edu/~jls/stat511/lectures/lec10.pdf)
  La diferencia entre ls y lsfit es que en lsfit tenemos que proporcionar la matriz del ajuste y en ls sólo damos los datos

- Para guardar una imagen podemos hacer ggsave("image.png")

- Para usar una librería haremos **library("ggplot2")** lo que nos permitirá usar nuevos métodos

- Para incluir varias series de valores con distintos tipos de puntos:

```
qplot(zaragoza.data$orderScaled,log10(zaragoza.data$contributions)) +
geom_point(aes(y=log10(almeria.data$contributions),x=almeria.data$orderScal
```

- Los datasets usados se guardan en *.Rdata* . Si guardamos el workspace al salir se irán guardando.

- Podemos hacer que se muestren los ejes en escala logarítmica usando el parámetro de plot log="xy" (para los dos ejes o log="x" para uno)

- Visualizamos un objeto con **view(objeto)**
- Podemos representar una función con **curve (http://astrostatistics.psu.edu/su07/R/html/graphics/html/curve.html)**