

# Práctica 1

## Usando RSNNS

- Descripción de R.
- Abrir y usar RStudio.
- Manejo sencillo del script para la práctica.

# Lenguaje R

- Lenguaje de programación abierto para la computación estadística y el análisis de datos.
- Se puede encontrar en: <https://www.r-project.org/about.html>
- Contiene multitud de operaciones para el procesamiento de distintas estructuras de datos, especialmente para matrices.
- Amplio abanico de librerías de herramientas de análisis de datos desarrolladas en este lenguaje.

# RStudio

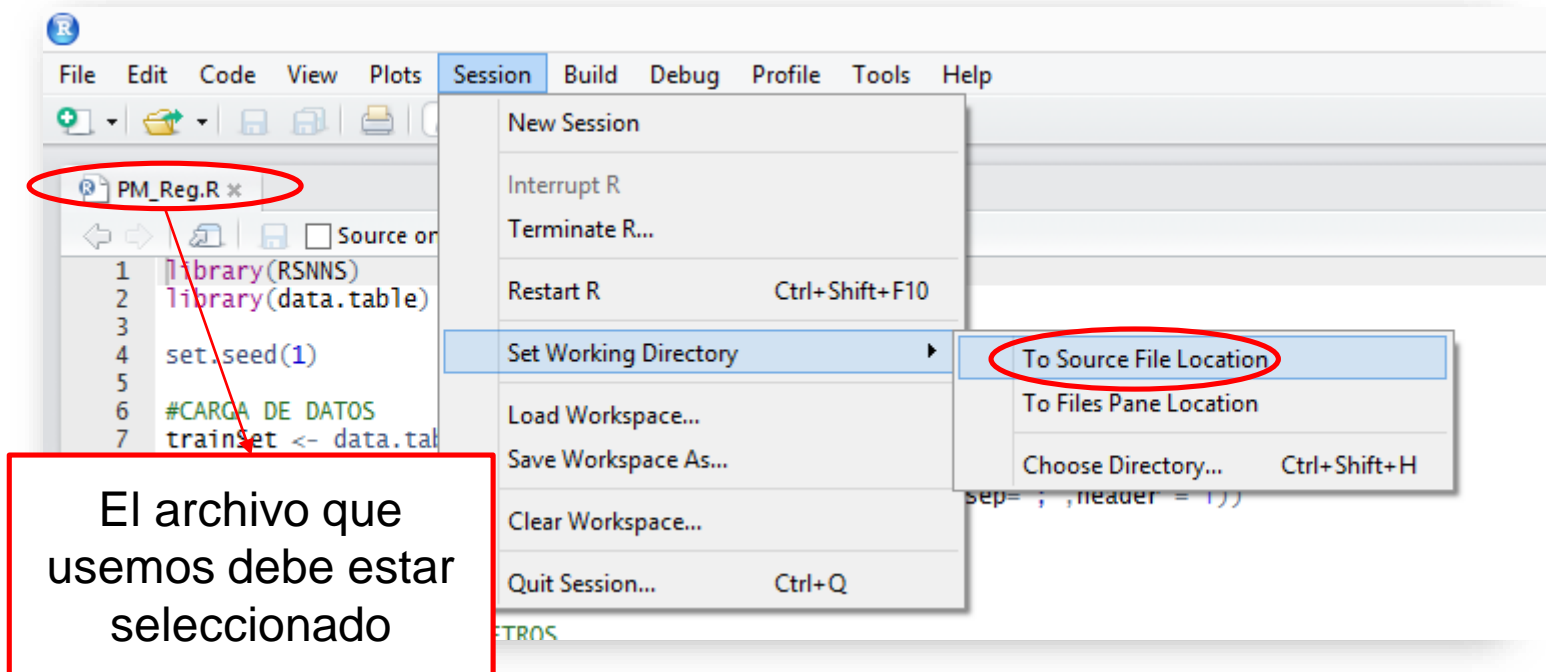
- Entorno de desarrollo para el lenguaje R con consola, desarrollo de scripts, visor de variables y diversas utilidades para facilitar el uso de R.
- Se puede encontrar en: <https://rstudio.com>

# Preparación de RStudio

- Abrir RStudio en Windows
- Desde RStudio, abrir el archivo de la práctica PM2019\_Reg.R
- Asegurarse de que los archivos de datos y el script estén en el mismo directorio

# Preparación de RStudio

- Configurar el directorio de trabajo



# Preparación de RStudio

- Instalar los paquetes necesarios:
  - RSNNS
  - Es una adaptación para R del conocido simulador “The Stuttgart Neural Network Simulator” (SNNS)
  - Es un paquete que contiene muchas funciones para usar el simulador
  - <https://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf>

# Ficheros de datos

- El script está preparado para cargar los siguientes ficheros de datos (los nombres y extensiones de los ficheros pueden variar):
  - Train.dat
  - Validacion.dat
  - Test.dat
- Los ficheros de datos deben tener el formato siguiente:
  - ‘,’ coma para marcar separación entre dos campos (sep=“,”)
  - ‘.’ punto para marcar el decimal de los números (dec=“.”)
  - Con cabecera (header=T)
- Pueden usarse otros formatos cambiando los valores de los parámetros dec (separador decimal), sep (separador entre atributos) y header (si tiene cabecera o no). (Si no tenemos cabecera será header = F)
- Si los campos están separados por espacios o tabulador puede usarse read.table() en lugar de read.csv()



# Ejecutar el script

- Se puede ejecutar todo el script de una vez

- Pulsar “Source”



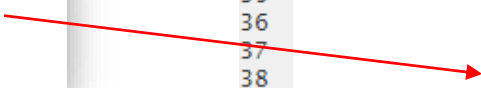
- También se puede ejecutar la parte del código que se desee seleccionándolo y pulsando “Run”.



# Usar el MLP con RSNNS

- El código que ejecuta MLP es:

```
13  
14 #SELECCION DE LOS PARAMETROS  
15 topologia <- c(20)  
16 razonAprendizaje <- 0.01  
17 ciclosMaximos <- 200  
18  
30  
31 #EJECUCION DEL APRENDIZAJE Y GENERACION DEL MODELO  
32 # En RSNNS se llama test a nuestro fichero de validación  
33  
34 model <- mlp(x= trainset[,-salida],  
35               y= trainset[, salida],  
36               inputsTest= validset[,-salida],  
37               targetsTest= validset[, salida],  
38               size= topologia,  
39               maxit=ciclosMaximos,  
40               learnFuncParams=c(razonAprendizaje),  
41               shufflePatterns = F  
42               )  
43
```



- Los parámetros que hay que modificar son los indicados:
  - topologia:** Define las neuronas ocultas del MLP, se define usando una concatenación de valores, indicando el número de neuronas ocultas por cada capa: c(20), c(10,20,15), etc. Si hay solo una capa oculta se puede usar un escalar
  - razonAprendizaje:** Define la razón de aprendizaje
  - ciclosMaximos:** Define el número máximo de iteraciones del algoritmo de aprendizaje

# Usar el MLP con RSNNs

- El modelo completo se guarda en el objeto “model”, incluida la evolución de los errores de entrenamiento y validación.

```
44 #GRAFICO DE LA EVOLUCION DEL ERROR  
45 plotIterativeError(model)
```

- A la derecha en Rstudio se muestra la evolución del error. La línea roja marca el error de validación y la línea negra representa el error de entrenamiento.
- Se pueden consultar:
  - Errores finales: *errors*
  - Evolución del MSE: *iterativeErrors*
  - Salidas de la red: *outputs*
- El resultado del entrenamiento se guarda al final en ficheros csv.

