

# Curso de introducción al software estadístico R. Introducción a la interfaz Rstudio.

Profesores: Carlos P.G., Roberto D.G. y Marcos C.S.



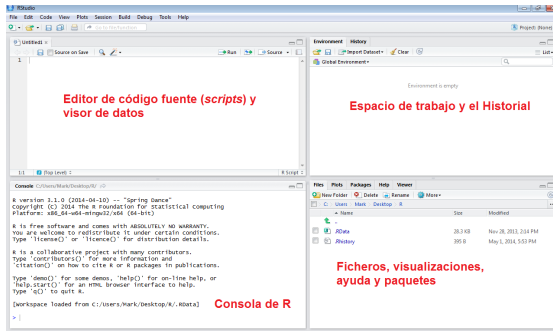
## 5. Gráficos en R: el comando `plot()` y librerías específicas. Introducción a la interfaz Rstudio.

## 5.2 Introducción a la interfaz Rstudio.

- ¿Qué es RStudio?
- Vistazo rápido y características de RStudio
- Instalación de RStudio
- Descripción del entorno
- Cargar datos en Rstudio
- Gestión del historial
- Primer script en RStudio
- Usando la ayuda
- Extracción y ejecución de funciones
- Secciones, plegado y navegación de código
- Visualización de datos
- Los comandos `plot`, `qplot` y `manipulate`
- Instalación de paquetes en RStudio
- Generando informes HTML a partir de scripts de R
- Otros aspectos de R
- Proyecto dentro de RStudio
- Atajos de teclado
- Ejercicios
- Futuro de RStudio
- Referencias

# ¿Qué es RStudio?

- RStudio es una herramienta **IDE (Integrated Development Environment)** para R, libre y gratuita que facilita:
  - Trabajar con R y gráficos de R de forma interactiva.
  - Organizar el código y mantener múltiples proyectos.
  - Gestión de los paquetes de R (instalación, actualización,...).
  - Crear y compartir informes (utilizando un lenguaje llamado markdown).
  - Compartir código y colaborar con otros usuarios.
- RStudio también se ofrece como herramienta comercial para empresas (ofreciendo la empresa desarrolladora servicios adicionales de consultoría y formación).
- RStudio **NO** realiza ninguna operación estadística. Solo facilita realizar dichas operaciones sobre R.



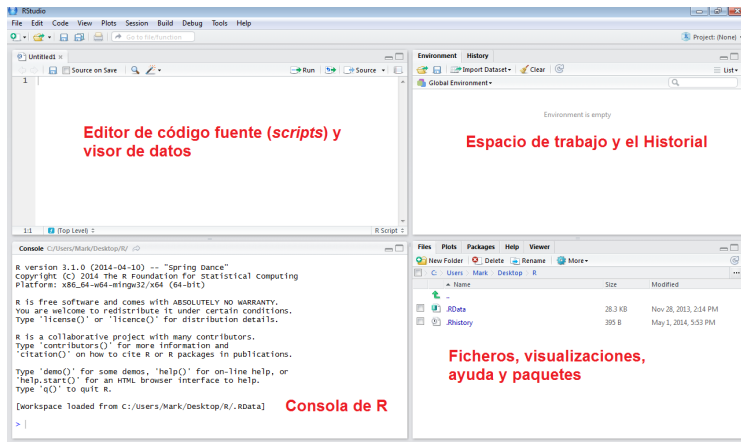
# Instalación de RStudio

- Antes de instalar RStudio, necesitas instalar **R**. Se puede obtener desde:
  - [cran.r-project.org](https://cran.r-project.org)
- Una vez instalado R, se puede instalar RStudio en tu ordenador (versión **Desktop**) dependiendo del sistema operativo:
  - [www.rstudio.com/ide/download/desktop](https://www.rstudio.com/ide/download/desktop)
- Solo en Linux, también se puede instalar **RStudio Server**, el cual permite acceder al entorno desde un navegador web:
  - [www.rstudio.com/ide/server](https://www.rstudio.com/ide/server)

# Características de Rstudio

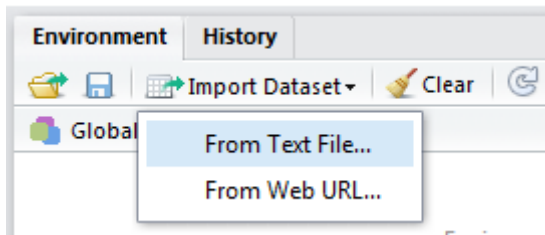
- Integración de la consola de R dentro de Rstudio (se pueden teclear comandos de R directamente).
- Ejecución de código (permite ejecutar código directamente desde un fichero de script).
- Resaltado de la sintaxis (realiza una coloración automática de los instrucciones y de las funciones).
- Ayuda con los paréntesis, corchetes y comillas (autocompleta estos símbolos al abrirlos).
- Completado de comandos (completa los comandos mientras se escriben usando la tecla `Tab`).
- Atajos de teclado (al final de la presentación).
- Navegador de objetos (se pueden inspeccionar todos los objetos de la sesión de R).
- Gestión del historial de comandos.
- Navegación del código en la ventana de script (mediante secciones que finalizan en ----).
- Importación y visualización de datos.
- Integración de gráficos (manipulación, zoom y exportación).
- Otros aspectos destacables son la gestión de proyectos (se puede cambiar de un proyecto a otro fácilmente), el control de versiones (se integra bien con `git` y `svn`) y la generación de documentos (tipo PDF, HTML y otros más avanzados con un solo click).

# Descripción del entorno



## Cargar datos en Rstudio

- Vamos a cargar unos datos para poder empezar a trabajar:



- Aquí podemos elegir:
  - Cargar los datos desde un fichero de texto (**From Text File...**).
  - O desde una URL de una web (**From Web URL...**).
- Desde esta URL podemos descargar el fichero `ddt.txt`, o usarla directamente:
  - <http://cpgonzal.github.io/cursIntroR/ddt.txt>



- **group:** Grupo de observaciones (mediciones en dos temporadas).
- **location:** Código del lugar de medición (hay tres lugares: 1,2,3).
- **location\_name:** Lugar de medición (Afluente, CursoM, Desemboca).
- **species:** Código de especie (de 1 a 3).
- **species\_name:** Nombre de la especie del pez.
- **length:** Longitud del pez.
- **weight:** Peso del pez.
- **DDT\_conc:** Concentración de DDT (% de peso) medida.

Import Dataset

Name

ddt

Heading

☒ Yes
 ☐ No

Separator

Whitespace

Decimal

Period

Quote

Double quote (")

na.strings

NA

☒ Strings as factors

Input File

```

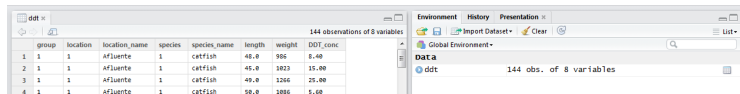
group location location_name species species_name length
1 1 Afluyente 1 catfish 48 986 8.4
1 1 Afluyente 1 catfish 45 1023 15
1 1 Afluyente 1 catfish 49 1266 25
1 1 Afluyente 1 catfish 50 1086 5.6
1 1 Afluyente 1 catfish 46 1044 4.6
1 1 Afluyente 1 catfish 52 1770 8.2
1 1 Afluyente 1 catfish 48 1048 6.1
1 1 Afluyente 1 catfish 51 1641 13
1 1 Afluyente 1 catfish 48.5 1331 6
1 1 Afluyente 1 catfish 51 1728 6.6
1 1 Afluyente 1 catfish 44 917 5.5
1 1 Afluyente 1 catfish 51 1398 11
1 1 Afluyente 2 buffalo 49 1763 4.5
1 1 Afluyente 2 buffalo 46 1459 4.2
1 1 Afluyente 2 buffalo 52 2302 3
1 1 Afluyente 2 buffalo 46 1614 2.3

```

Data Frame

group	location	location_name	species	species_name	length
1	1	Afluyente	1	catfish	48
1	1	Afluyente	1	catfish	45
1	1	Afluyente	1	catfish	49
1	1	Afluyente	1	catfish	50
1	1	Afluyente	1	catfish	46
1	1	Afluyente	1	catfish	52
1	1	Afluyente	1	catfish	48
1	1	Afluyente	1	catfish	51
1	1	Afluyente	1	catfish	48.5
1	1	Afluyente	1	catfish	51
1	1	Afluyente	1	catfish	44
1	1	Afluyente	1	catfish	51
1	1	Afluyente	2	buffalo	49
1	1	Afluyente	2	buffalo	46
1	1	Afluyente	2	buffalo	52
1	1	Afluyente	2	buffalo	46

# Datos cargados en RStudio



ddt x

144 observations of 8 variables

	group	location	location_name	species	species_name	length	weight	DDT_conc
1	1	1	Afluyente	1	catfish	48.0	986	8.40
2	1	1	Afluyente	1	catfish	45.0	1023	15.00
3	1	1	Afluyente	1	catfish	49.0	1266	25.00
4	1	1	Afluyente	1	catfish	50.0	1086	5.60

Environment History Presentation x

Global Environment

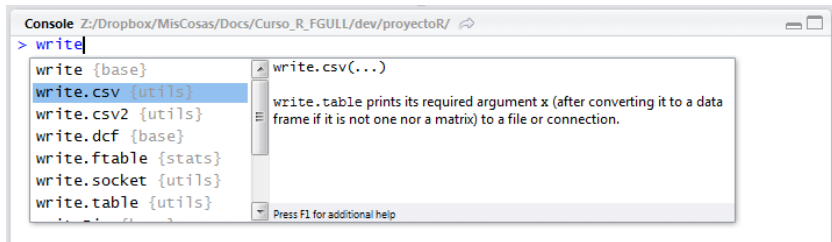
Data

ddt 144 obs. of 8 variables

- Los datos cargados se muestran en el panel superior izquierdo, y en el panel de espacio de trabajo (o entorno).
- Vamos a guardar estos datos en un fichero CSV (*comma separated values*).
- Para ello, empezaremos a usar la consola de R, y las funciones incorporadas de completado de comandos y de nombres de carpetas/ficheros.

## Guardar datos en formato CSV

- Tecleamos el comando `write` en la consola R, y pulsamos la tecla de tabulación `Tab`:

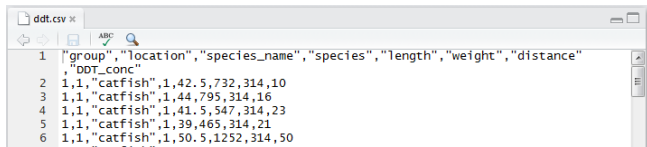


- La característica de RStudio que hemos usado se denomina **completado de comandos**.
- De las opciones que nos ofrece, seleccionamos `write.csv`, y seguimos escribiendo un paréntesis `(`, que RStudio completará con otro paréntesis `)`.
- Continuamos escribiendo el comando, indicando en el primer parámetro el objeto `ddt`, y en el segundo parámetro, el nombre del fichero a guardar.
- La función que hemos usado ahora se denomina **completado de carpetas/ficheros**.
- Como nombre del fichero, ponemos el nombre `ddt.csv`.
- Terminamos de escribir el comando con el último parámetro, para indicar que no queremos que guarde el nombre (número) de cada fila:

```
write.csv(ddt, "ddt.csv", row.names=FALSE)
```

## El directorio de trabajo

- Si no lo hemos configurado previamente, este archivo csv no se ha copiado en el directorio `cursoR`.
- En R (y Rstudio), existe lo que se conoce como **directorio de trabajo** y que representa el directorio donde (por defecto) se cargan, se graban, etc.. los archivos en caso de no especificar una ruta (path) para el archivo.
- Este directorio se consulta con el comando `getwd()`. Si queremos cambiarlo utilizamos el comando `setwd(ruta_a_carpeta)`.
- En Rstudio lo cambiamos fácilmente en la opción `More` de la ventana de ficheros, una vez que nos hemos situado dentro del directorio deseado.
- También podemos especificar la ruta de búsqueda del fichero en `write.csv` pulsando `Tab` en el segundo argumento de la función.
- Podemos ver el fichero creado, y haciendo clic sobre él, nos mostrará el contenido:

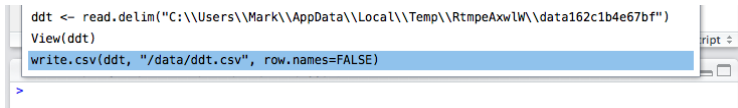


The screenshot shows the RStudio interface with a file named 'ddt.csv' open. The file contains a CSV dataset with 6 rows of data. The first row is the header, and the subsequent rows contain data for 'catfish' species, including location, species name, length, weight, distance, and DDT concentration.

```
1 "group", "location", "species_name", "species", "length", "weight", "distance"  
2 "DDT_conc"  
3 1,1,"catfish",1,42.5,732,314,10  
4 1,1,"catfish",1,44,795,314,16  
5 1,1,"catfish",1,41.5,547,314,23  
6 1,1,"catfish",1,39,465,314,21  
7 1,1,"catfish",1,50.5,1252,314,50
```

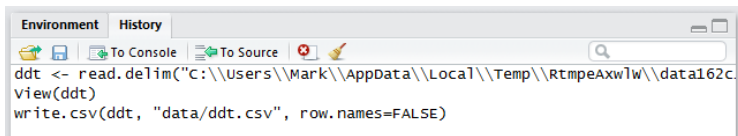
# Historial de RStudio

- Hay tres formas para re-usar los comandos ya tecleados en RStudio:
  - Pulsando las teclas de flecha arriba o abajo.
  - Pulsando **Ctrl+Up**.



```
ddt <- read.delim("C:\\Users\\Mark\\AppData\\Local\\Temp\\RtmpeAxwIw\\data162c1b4e67bf")
View(ddt)
write.csv(ddt, "/data/ddt.csv", row.names=FALSE)
```

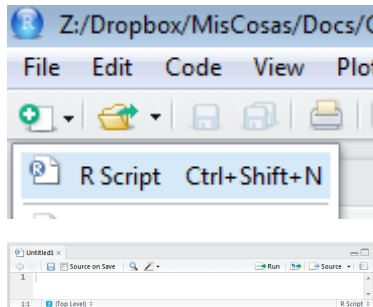
- Explorando la pestaña **History** en el panel derecho superior:



- Se puede seleccionar un comando o varios (usando la tecla **Shift+Mayúsc**), y ejecutar pulsando **Enter**.
- Asimismo, se pueden salvar a un fichero llamado **.Rhistory**, en la carpeta del proyecto.
- También se pueden pasar a la consola con el comando **To Console** para ser ejecutados, o a un script directamente con el botón **To Source**.
- Por último, se pueden eliminar con la tecla **Supr (Del)** o con el botón correspondiente, o borrar completamente todo el historial con el botón de la escoba.

# Primer script R en RStudio

- Una vez cargados los datos, vamos a escribir el primer script en lenguaje R para hacer un primer análisis.
- Hacemos click en el icono + verde del editor de scripts (izquierdo superior), y elegimos **R Script**.



# Primer script R en RStudio

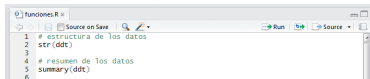
- Las acciones más interesantes sobre un script de R son:
  - **Disco**: salvar el fichero.
  - **Lupa**: buscar y reemplazar.
  - **Varita mágica**: herramientas útiles (algunas las veremos luego).
  - **Run**: ejecuta el código seleccionado (**Ctrl+Enter**).
  - **Re-Run**: ejecuta el último código que seleccionamos.
  - **Cuaderno**: compila el script R a un fichero HTML (lo veremos luego).
- Las opciones del tipo **Source** sirven para cargar el código fuente al espacio de trabajo de R (lo veremos luego).
- Vamos a introducir los primeros comandos R en el script:

```
# estructura de los datos
str(ddt)

# resumen de los datos
summary(ddt)
```

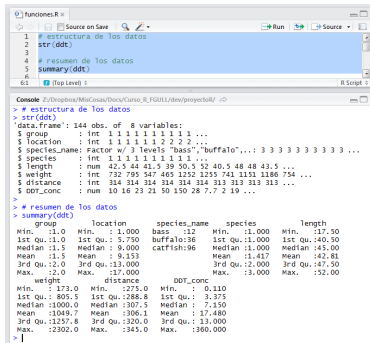
# Primer script R en RStudio

- Guardamos este script pulsando el botón del disco (arriba), o en el menú **File > Save**, con el nombre `funciones.R` en el directorio `R` de nuestro proyecto.



```
1 # estructura de los datos
2 str(dtt)
3
4 # resumen de los datos
5 summary(dtt)
6
```

- Una vez guardado, podemos ejecutar el script de varias formas:
  - Sobre la línea en la que estamos, pulsamos el botón **Run** o **Ctrl+Enter**.
  - O seleccionamos todo el script, y hacemos lo mismo que en el punto anterior.



```
> # estructura de los datos
> str(dtt)
'data.frame': 144 obs. of  8 variables:
 $ group      : int  1 1 1 1 1 1 1 1 1 ...
 $ location   : int  1 1 1 1 1 2 2 2 2 ...
 $ species_name: factor w/ 3 levels "bass","buffalo",...: 3 3 3 3 3 3 3 3 3 ...
 $ species    : int  1 1 1 1 1 1 1 1 1 ...
 $ length     : num  42.5 44 41.5 39 50.5 52 40.5 48 43.5 ...
 $ weight     : int  732 795 547 465 1252 1255 741 1151 1186 754 ...
 $ distance   : int  314 314 314 314 314 314 313 313 313 ...
 $ ddt_conc   : num  10 16 23 21 50 150 28 7 7 2 19 ...

> # resumen de los datos
> summary(dtt)
      group      location  species_name species      length
Min.   :1.0    Min.   : 1.000    bass :12    Min.   :1.000   Min.   :17.50
1st Qu.:1.0    1st Qu.: 5.750    buffalo:36 1st Qu.:11.000 1st Qu.:40.50
Median :1.5    Median : 9.000    catfish:96 Median :11.000  Median :45.00
Mean   :1.5    Mean   : 9.153      Mean :1.417   Mean   :42.81
3rd Qu.:2.0    3rd Qu.:13.000      3rd Qu.:12.000 3rd Qu.:47.50
Max.   :2.0    Max.   :37.000      Max.   :13.000   Max.   :52.00

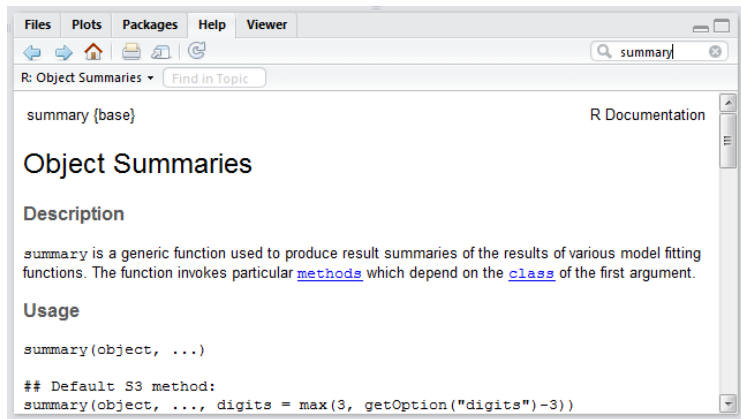
      weight     distance      ddt_conc
Min.   :173.0    Min.   :275.0    Min.   : 0.110
1st Qu.: 805.5    1st Qu.:288.8    1st Qu.: 3.375
Median :1000.0    Median :307.5    Median : 7.150
Mean   :1049.7    Mean   :306.1    Mean   :17.480
3rd Qu.:1257.8    3rd Qu.:320.0    3rd Qu.:13.000
Max.   :2302.0    Max.   :345.0    Max.   :360.000

> |
```



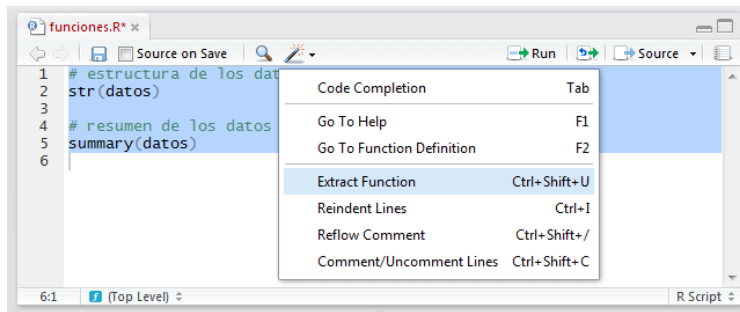
# Usando la ayuda

- Ahora es buen momento para empezar a usar la ayuda para, por ejemplo, el comando `summary`:



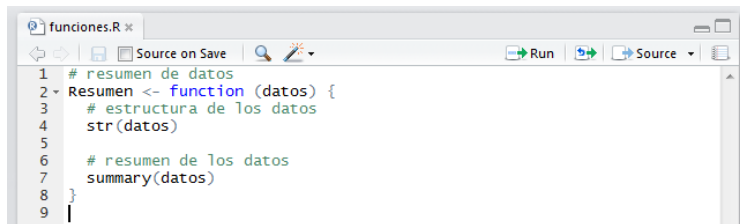
## Extracción de funciones

- Una de las características más interesantes de RStudio es la posibilidad de crear funciones a partir de trozos de código.
- Por ejemplo, vamos a crear una función llamada `Resumen`, que recibe un parámetro `datos`, y ejecuta los dos comandos que acabamos de escribir.
- Para ello, cambiamos `ddt` por `datos` en ambos comandos.
- Seleccionamos las líneas del código.
- Pulsamos sobre la “varita mágica” en la opción **Extract Function**, y le damos el nombre `Resumen`.



## Extracción de funciones

- Añadimos finalmente un comentario para describir el objetivo de la función.
- Por último, salvamos el script.



```
funciones.R *
1 # resumen de datos
2 Resumen <- function (datos) {
3   # estructura de los datos
4   str(datos)
5
6   # resumen de los datos
7   summary(datos)
8 }
9
```

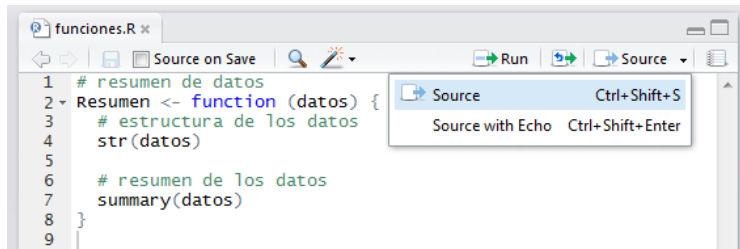
## Ejecutando funciones propias

- Si tecleamos la siguiente línea en la consola de R, nos dará un error:

```
Resumen(ddt)
```

```
## Error in eval(expr, envir, enclos): could not find function "Resumen"
```

- Esto sucede porque la función `Resumen` no ha sido cargada (*sourced*) al entorno o espacio de trabajo de nuestro proyecto.
- Para poder usar la función `Resumen(datos)` tenemos que hacer lo siguiente:



- Si hubiéramos elegido la opción **Source with Echo**, obtendríamos el mismo resultado además de mostrar todo el código cargado.
- Las funciones cargadas aparecen, al igual que los datos, en el panel derecho superior de **Environment**, en la sección de **Functions**.

## Ejecutando funciones propias

- Ya podemos usar nuestra nueva función `Resumen` pasándole como parámetro los datos de `ddt`:

```
Console Z:/Dropbox/MisCosas/Docs/Curso_R_FGULL/dev/proyectoR/
> Resumen(ddt)
'data.frame': 144 obs. of  8 variables:
 $ group      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ location   : int  1 1 1 1 1 1 2 2 2 2 ...
 $ species_name: Factor w/ 3 levels "bass","buffalo",...: 3 3 3 3 3 3 3 3 3 3 ...
 $ species    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ length     : num 42.5 44 41.5 39 50.5 52 40.5 48 48 43.5 ...
 $ weight     : int 732 795 547 465 1252 1255 741 1151 1186 754 ...
 $ distance   : int 314 314 314 314 314 314 313 313 313 313 ...
 $ DDT_conc   : num 10 16 23 21 50 150 28 7.7 2 19 ...

  group      location  species_name  species
Min.   :1.0    Min.   : 1.000    bass :12    Min.   :1.000
1st Qu.:1.0    1st Qu.: 5.750    buffalo:36  1st Qu.:1.000
Median :1.5    Median : 9.000    catfish:96 Median :1.000
Mean   :1.5    Mean   : 9.153                Mean   :1.417
3rd Qu.:2.0    3rd Qu.:13.000                3rd Qu.:2.000
Max.   :2.0    Max.   :17.000                Max.   :3.000

  length  weight  distance  DDT_conc
Min.   :17.50  Min.   :173.0  Min.   :275.0  Min.   : 0.110
1st Qu.:40.50  1st Qu.: 805.5  1st Qu.:288.8  1st Qu.: 3.375
Median :45.00  Median :1000.0  Median :307.5  Median : 7.150
Mean   :42.81  Mean   :1049.7  Mean   :306.1  Mean   :17.480
3rd Qu.:47.50  3rd Qu.:1257.8  3rd Qu.:320.0  3rd Qu.:13.000
Max.   :52.00  Max.   :2302.0  Max.   :345.0  Max.   :360.000
> |
```

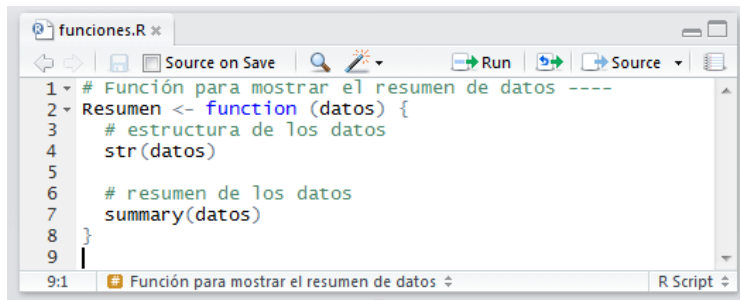
- Si vamos a estar cambiando la función continuamente, podemos activar la opción **Source on Save** para que la cargue al entorno automáticamente después de salvar el script.

## Secciones del código

- Otra característica interesante de RStudio (y no de R) es la posibilidad de estructurar el código en secciones.
- Las secciones se pueden crear desde el menú **Code > Insert Section**, o simplemente poniendo un comentario (#) con un nombre de sección y acabado en 4 guiones (----):

```
# <NombreDeLaSección> ----
```

- Podemos aprovechar el comentario de la función Resumen para hacer nuestra primera sección:

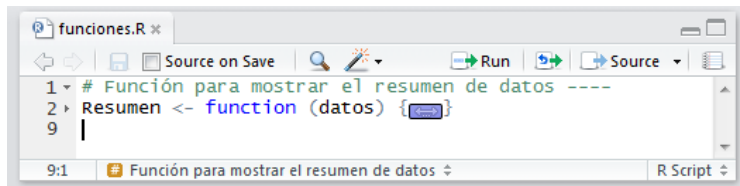


```
1 # Función para mostrar el resumen de datos ----
2 Resumen <- function(datos) {
3   # estructura de los datos
4   str(datos)
5
6   # resumen de los datos
7   summary(datos)
8 }
9
```

- Fíjense que dicha sección aparece en la parte inferior del editor como **navegación del código**, que luego veremos.

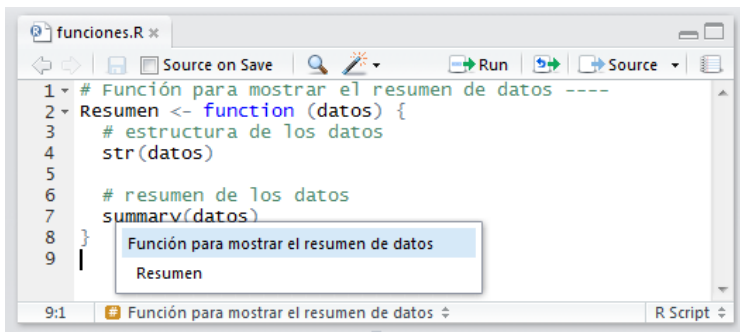
## Plegado de código/secciones

- Otra característica muy útil de RStudio es el plegado de las secciones o de partes del código que estén rodeadas por llaves `{ }`.
- Al plegarse aparecerá un pequeño triángulo que permite colapsar o expandir el bloque de código.



## Navegación de código

- La navegación de código en RStudio es una utilidad que permite editar el código de forma más rápida.
- Se puede acceder a una línea concreta pulsando **Alt+Shift+G**, o en el menú **Edit > Go to Line...**
- Con la opción **Code > Jump To...** (**Alt+Shift+J**) se puede saltar directamente a funciones o secciones del código.



- Otra opción muy útil es la de ir a un fichero/función determinado usando **Code > Go To File/Function (Ctrl+.)**. RStudio mostrará todos los ficheros o funciones dentro del directorio de trabajo que empiecen con los caracteres tecleados.



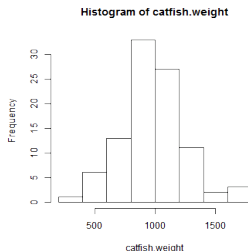
# Visualización de datos

- La visualización (o dibujo de gráficos) es una parte esencial del análisis de datos. RStudio tiene un panel específico para la visualización de datos (**Plots**) abajo a la derecha.
- Para ilustrar el manejo de las instrucciones de visualización, vamos a usar escribir el siguiente código dentro de nuestro script de R debajo de la función `Resumen`:

```
# cargamos los datos del fichero
ddt <- read.csv("ddt.csv")

# pesos de la especie "catfish"
catfish.weight <- ddt$weight[ddt$species_name == "catfish"]

# histograma del peso de los "catfish"
hist(catfish.weight)
```



- Al teclear `ddt$` podemos pulsar la tecla `Tab` para que nos muestre la lista de variables de `ddt`. Esto se denomina **completado de objetos**.

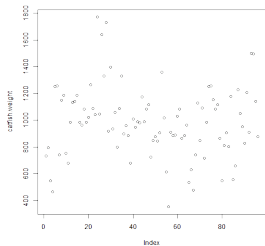
## Opciones de la pestaña **Plot**

- Con la opción **Zoom** se abre una nueva ventana con una versión más grande del gráfico.
- El botón **Export** nos permite guardar el gráfico como una imagen en varios formatos (PNG, JPEG, TIFF, etc) o como un fichero PDF.
- También podemos copiar el gráfico actual al portapapeles del sistema.
- En caso de haber generado varios gráficos, las flechas permiten avanzar o retroceder en la visualización de dichos gráficos.

## El comando `plot`

- Una vez vista la forma de trabajar con gráficos dentro de RStudio vamos a ver opciones más avanzadas de los mismos.
- Uno de los comandos más útiles para dibujar gráficos en R es `plot`.
- El gráfico más simple es dibujar simplemente los pesos de la especie `catfish`:

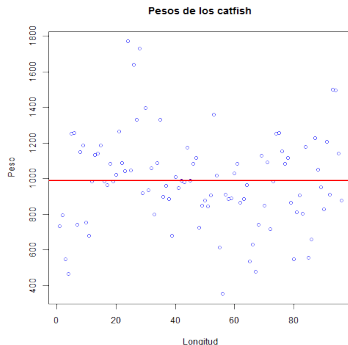
```
# gráfico de los pesos de todos los catfish  
plot(catfish.weight)
```



## El comando `plot`

- En primer lugar, vamos a añadir color, unas nuevas etiquetas en los ejes (`xlab`, `ylab`) y el título principal (`main`), además de una línea en el eje Y que indica la media de los valores.

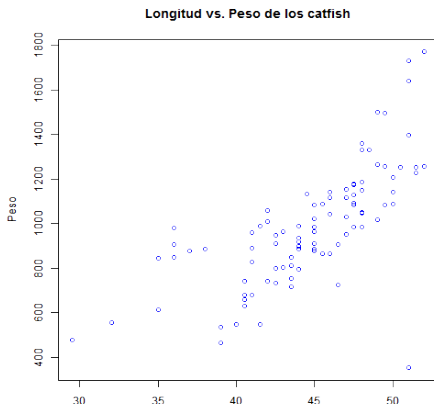
```
plot(catfish.weight, col="blue", xlab="Longitud", ylab="Peso", main="Pesos de los catfish")  
  
# línea horizontal que marca la media de los pesos  
abline(mean(catfish.weight), 0, col="red", lwd=2)
```



## El comando `plot`

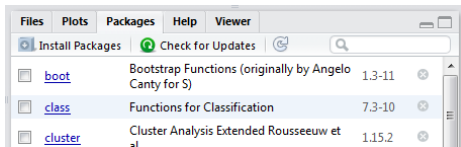
- Ahora, podemos dibujar un gráfico que muestre la longitud (`length`) frente al peso (`weight`) de la especie `catfish`.

```
# longitudes de la especie "catfish"  
catfish.length <- ddt$length[ddt$species_name == "catfish"]  
  
# gráfico de comparación de la longitud vs. peso de los "catfish"  
plot(x=catfish.length, y=catfish.weight, col="blue", xlab="Longitud", ylab="Peso",  
      main="Longitud vs. Peso de los catfish")
```



# Instalación de paquetes en RStudio

- Antes de ver el comando `qplot`, tenemos que aprender como instalar nuevos paquetes de R.
- Una de las pestañas más interesantes en el lado derecho es **Packages** (paquetes):



- **Check for Updates:** permite actualizar los paquetes a sus últimas versiones.
- **Install Packages:** permite instalar paquetes desde **CRAN** (repositorio).
- Se puede conseguir el mismo efecto con el siguiente comando:

```
install.packages("<Nombre_de_la_librería>")
```

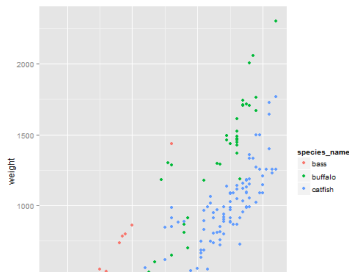
## La función `qplot`

- El comando `qplot` es una versión “*quick*” del comando `plot`, y permite crear gráficos complejos y avanzados de forma simple y rápida.
- Este comando está en la librería `ggplot2`. Para usar dicha librería podemos ejecutar el siguiente comando:

```
library(ggplot2)
```

- En caso de que no esté instalada, habría que seguir los pasos descritos en la diapositiva de Instalación de paquetes en RStudio.
- Por ejemplo, podemos hacer un gráfico rápido que permite visualizar la longitud (`length`) frente al peso (`weight`) del conjunto de datos `ddt`.

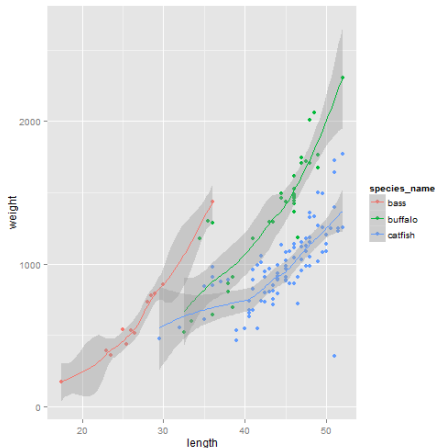
```
# longitud vs. peso de todos los peces  
qplot(length, weight, data=ddt, col=species_name)
```



## La función `qplot`

- Incluso podemos añadir una línea de tendencia (con un margen de error) a cada especie, simplemente añadiendo (sumando) la función `geom_smooth()`:

```
qplot(length, weight, data=ddt, col=species_name) + geom_smooth()
```

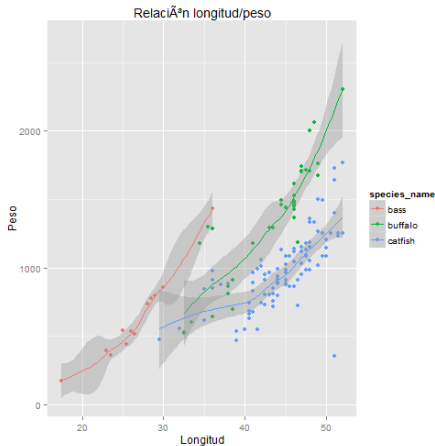




## La función `qplot`

- Podemos usar las opciones que ya conocemos del comando `plot` para personalizar este gráfico:

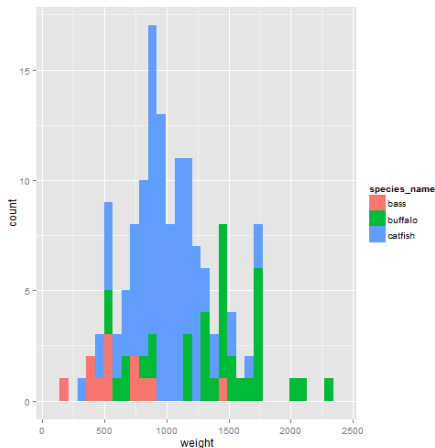
```
qplot(length, weight, data=ddt, col=species_name, xlab="Longitud", ylab="Peso",  
      main="Relación longitud/peso") + geom_smooth()
```



## La función `qplot`

- Finalmente, podemos usar `qplot` para generar también un histograma de los pesos de las tres especies juntas de forma automática:

```
# histograma usando "qplot"  
qplot(weight, data=ddt, fill=species_name)
```



## El comando `manipulate`

- RStudio permite controlar de forma dinámica los gráficos generados con R.
- El comando que permite la interactividad entre el usuario y los gráficos generados.
- Las opciones de controles que permite `manipulate`:
  - `slider`: control para un rango (min, max) numérico.
  - `picker`: control sobre un conjunto de opciones fijas.
  - `checkboxbox`: control de casilla de verificación.
  - `button`: control de botón.
- Vamos a ver un pequeño ejemplo:

```
# control de selección para elegir la especie de pez
library(manipulate)
manipulate(
  hist(ddt$weight[ddt$species_name == fish], xlab="Peso", ylab="Frecuencia",
    main=paste("Histograma del peso de los", fish)),
  fish = picker("bass", "buffalo", "catfish")
)
```

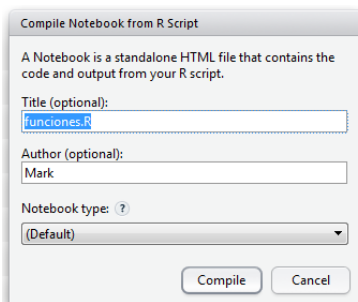


## Generando informes HTML a partir de scripts de R

- RStudio posee una opción que permite compilar un fichero script de R en HTML directamente (**Compile Notebook**).
- Dicha opción se encuentra, en cualquier script de R, al final de la barra de herramientas del panel de edición, con forma de cuaderno, o en la opción del menú **File > Compile Notebook...**
- Al pulsar el icono, nos aparece la siguiente ventana:
- Pulsamos el boton **Compile** y obtenemos la vista previa del fichero HTML generado:

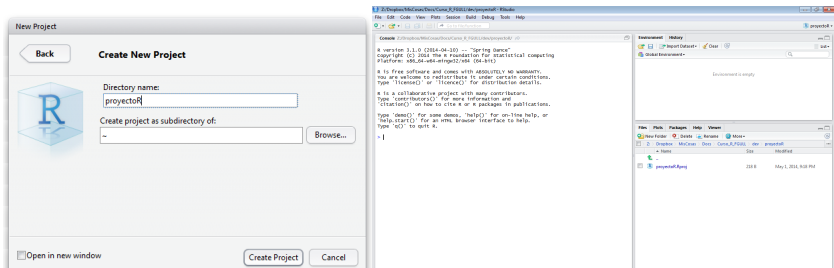
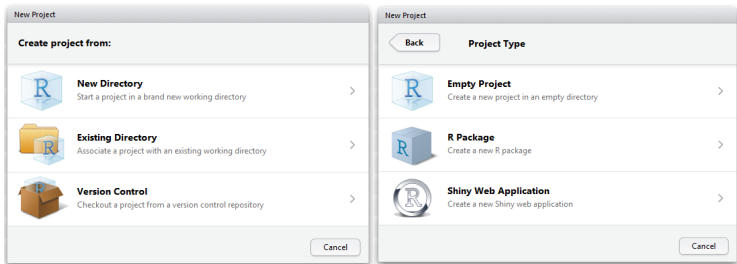
### Proyecto dentro de RStudio

- Para crear un proyecto, vamos a la esquina derecha de la barra de herramientas:



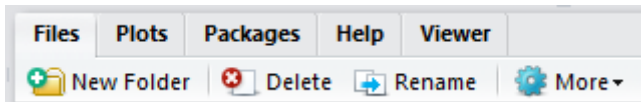
## Otros aspectos de R

- También podemos hacer lo mismo desde **File > New Project...**
- Le vamos a poner el nombre **proyectoR**.



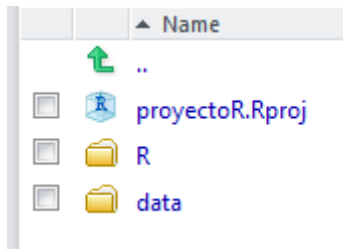
## Directorios del proyecto

- Normalmente, para proyectos simples, podemos poner todos los ficheros (scripts de R, datos, docs, informes, etc) en la misma carpeta del proyecto.
- Sin embargo, es una buena práctica, crear los siguientes directorios en cada proyecto:
  - `R`: contendrá los scripts de R que se vayan desarrollando.
  - `data`: almacenará los datos necesarios para realizar los análisis.
  - `doc`: contendrá toda la documentación necesario para los análisis.
  - `informes`: carpeta que guardará los informes generados por los análisis.
- En principio, solo vamos a crear las carpetas `R` y `data`.
- Vamos al panel de ficheros:



## Directorios del proyecto

- Hacemos clic en **New Folder**, y le damos el nombre `R`.
- Haremos lo mismo para crear la carpeta `data`.
- Por tanto, la carpeta del proyecto quedaría como:





# Atajos de teclado

- Como se ha visto a lo largo de este curso sobre RStudio, existen múltiples formas de realizar la misma acción dentro del entorno, a saber, desde los botones, el menú o los atajos de teclado.
- En cada apartado, se han ido comentado algunos atajos de teclado que pueden resultar interesantes.
- La tabla que contiene todos los atajos se puede obtener en el menú **Help > Keyboard Shortcuts**.

## Rstudio

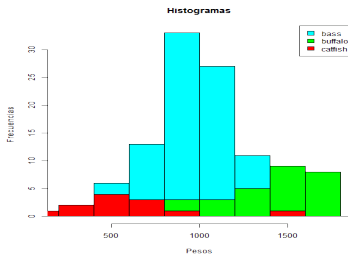
### Keyboard Shortcuts

#### Console

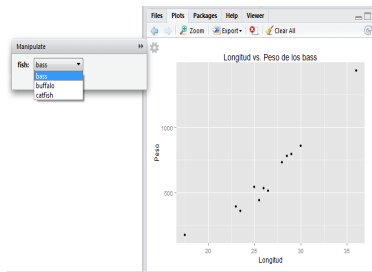
Description	Windows & Linux	Mac
Move cursor to Console	Ctrl+2	Ctrl+2
Clear console	Ctrl+L	Command+L
Move cursor to beginning of line	Home	Command+Left
Move cursor to end of line	End	Command+Right
Navigate command history	Up/Down	Up/Down
Popup command history	Ctrl+Up	Command+Up
Interrupt currently executing command	Esc	Esc
Change working directory	Ctrl+Shift+K	Ctrl+Shift+K

#### Source

- 1 Hacer una función en R que muestre los histogramas de frecuencias de los pesos de las tres especies (`catfish`, `buffalo` y `bass`) en uno solo usando el comando `hist`. (**Pista:** empezar por la especie `catfish` y usar `add=TRUE`).



- 2 Hacer una función que use el comando `manipulate` sobre el comando `qplot` para que el usuario elija la especie sobre la cual quiere comparar la longitud (`length`) contra el peso (`weight`).



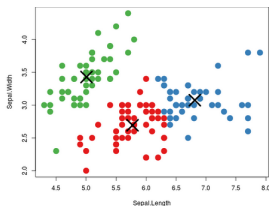
- Shiny by RStudio: *A web application framework for R.*

## Iris k-means clustering

X Variable  
Sepal.Length

Y Variable  
Sepal.Width

Cluster count  
3



server.R ui.R

show below

```
palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
"#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

shinyServer(function(input, output, session) {
  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
          col = clusters()$cluster,
          pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })
})
```

- RStudio Support: web de soporte de RStudio.
- RStudio Training: página de formación de RStudio.
- Google's R Style Guide: guía de estilo de programación en R según Google.