

Master en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Sesión 1: Presentación. Instalación de Software y Primeros Pasos

Fernando San Segundo

Curso 2021-22. Última actualización: 2021-07-19



- 1 Presentación del curso.
- 2 Instalación de software
- 3 La interfaz de RStudio
- 4 Variables y asignaciones en R.
- 5 El editor de RStudio. Ficheros de comandos de R.
- 6 Estructuras de datos básicas en R.
- 7 Librerías de R
- 8 Rmarkdown para la creación de documentos
- 9 Introducción al uso de GitHub para el curso

Sección 1

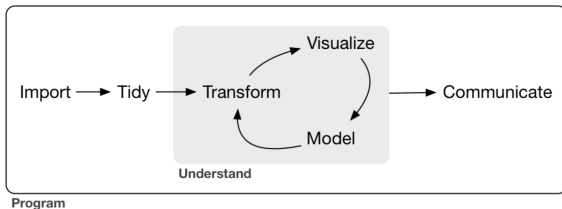
Presentación del curso.

- **Profesores:**

- ▶ Fernando San Segundo, Contacto: fsansegundo@comillas.edu
- ▶ Santiago Cano, Contacto: PENDIENTE

- La Guía Docente de la asignatura, así como el resto del material está accesible desde [Moodle](#). PENDIENTE
- La asignatura se plantea con un enfoque práctico y conceptual. Se trata de presentar las ideas fundamentales de la Estadística y el Análisis de Datos y hacerlo mediante la práctica computacional, usando el ecosistema de R como herramienta.

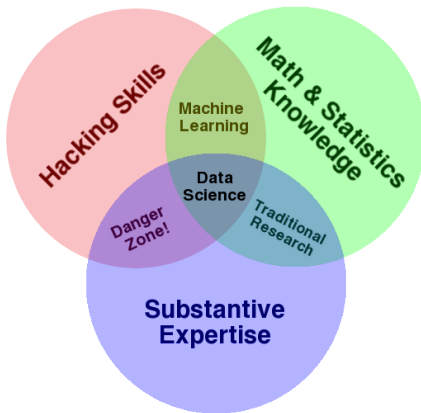
- La estructura básica de un Análisis de Datos es, según (Wickham and Grolemund 2016), esta:



Vamos a conocer las herramientas que R nos aporta para facilitar cada uno de los pasos de este proceso.

El diagrama de Venn del Análisis de Datos.

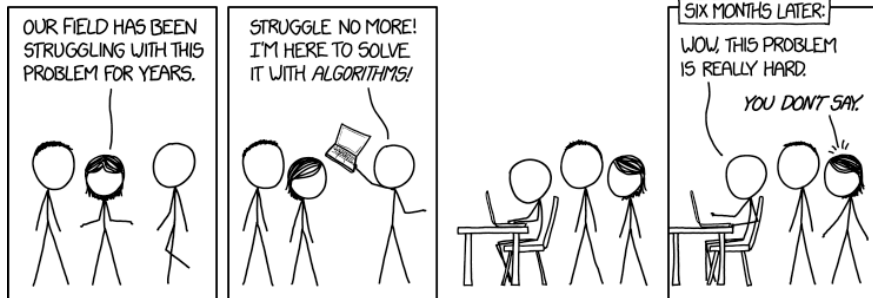
- Este diagrama creado por [Drew Conway](#) es uno de los más usados para describir el Análisis de Datos.



Nuestro curso se va a mover en la parte superior del diagrama, para proporcionaros las competencias necesarias para otras asignaturas del máster.

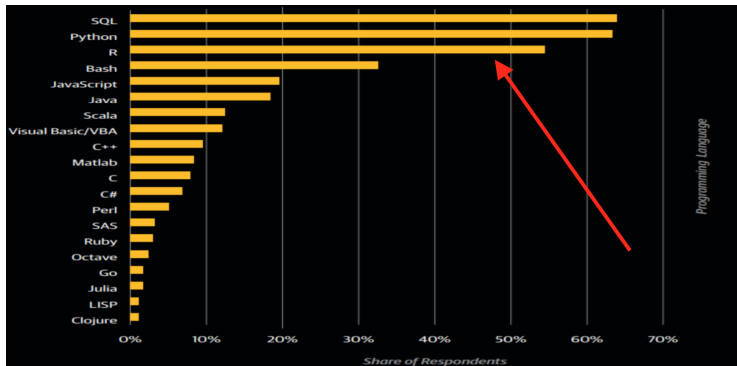
Danger zone number 2!

- El diagrama de Conway avisa de una zona de peligro para quienes rehuyen el uso correcto de la Estadística. Pero hay que tener cuidado también con otra zona de peligro para los practicantes de Machine Learning, que ilustra bien esta tira de [XKCD](#).



El ecosistema de R.

- R es uno de los lenguajes de programación más usados para el análisis de datos y uno de los que han experimentado un crecimiento más rápido **en los últimos años**.



- La comunidad de usuarios de R es uno de los pilares que fundamentan el éxito del lenguaje. La documentación disponible hace extremadamente fácil encontrar respuestas a casi todas las preguntas que vas a plantearte como usuario de R. Además hay un amplísimo catálogo bibliográfico y cientos de cursos disponibles para dominar cualquier aspecto de R. Aquí os enseñaremos algunos de nuestros favoritos.

Sección 2

Instalación de software

Instalación de R



Cran
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributing

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages. **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) [R 3.6.1 tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

- La página oficial de R es

www.r-project.org

Para que te sirva de referencia, la versión de R que estamos usando al escribir estas notas es: R version 4.1.0 (2021-05-18)

- Para descargar el instalador de R para tu sistema usa directamente [este enlace](#). Si vas a instalar en Mac o Linux y tienes alguna duda habla con tu profesor (especialmente si usas un Mac con procesador M1). La instalación es bastante sencilla, pero si quieres hacerte una idea previa aquí tienes un vídeo reciente sobre el [proceso de instalación en Windows](#). Y también puedes consultar [este enlace](#).
- ¡Instala R antes de continuar!

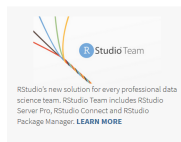
Instalación de RStudio



[Products](#) [Resources](#) [Pricing](#) [About Us](#) [Blogs](#) [Q](#)

Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace. [Learn More about RStudio features.](#)



| RStudio Desktop Open Source License | RStudio Desktop Commercial License | RStudio Server Open Source License | RStudio Server Pro Commercial License |
|--|---------------------------------------|---------------------------------------|--|
| FREE | \$995 per year | FREE | \$4,975 per year (5 Named Users) |
| DOWNLOAD | BUY | DOWNLOAD | BUY |

- Trabajar con R directamente no es muy cómodo, así que usaremos un entorno de trabajo llamado **RStudio Desktop**. Descarga la versión gratuita (free) para tu sistema de RStudio Desktop usando [este enlace](#). Denuedo, para ver el proceso por adelantado, aquí tienes un vídeo de una [instalación reciente de RStudio en Windows](#).
- **¡Instala RStudio antes de continuar!**
- Versión disponible en la fecha de publicación de estas notas: 1.4.1725

- \LaTeX : es necesario para producir ficheros en formato pdf a partir de nuestros análisis de datos. Aunque es posible que nos baste con instalar *tinytex* [tal y como se describe aquí](#), por si acaso incluimos aquí enlaces a las páginas de descarga de [MikTeX](#) (para Windows) y [MacTeX](#) (para Mac).
- Como lector de documentos pdf puedes usar [Adobe Reader](#), aunque en Windows te recomendamos [Sumatra Pdf](#) y en Mac recomendamos *Vista Previa* o [Skim](#).
- En algún momento puede ser necesario usar un editor de texto. El que incluye RStudio sirve para tareas sencillas, pero en Windows recomendamos [Notepad++](#) y en Mac [BBEdit](#).
- Para abrir algunos ficheros de datos es conveniente disponer de una hoja de cálculo como Excel de Microsoft o, alternativamente, Calc (de [Open Office](#) o de [Libre Office](#)). **Ten cuidado** si abres ficheros de datos (csv o txt) del curso con Excel: es fácil *romper* uno de esos ficheros inadvertidamente.

- Vamos a usar R de forma casi exclusiva en esta asignatura, pero es obligado señalar que Python es también extremadamente popular en este momento y es el lenguaje predominante en campos como el Aprendizaje Profundo (*Deep Learning*). En particular veremos en clase ejemplos que ilustran la forma de combinar ambos lenguajes en nuestro trabajo para aprovechar lo mejor de ambos.
- Si usas Mac o Linux ya tienes una instalación de Python en tu sistema. En Windows tienes que instalar una distribución de Python antes de usarla. **ATENCIÓN:** la instalación de Python es una fuente habitual de quebraderos de cabeza. Si dudas es *mucho mejor* que preguntes a tus profesores. En el Curso 0 tendrás una introducción a Python. Pero en cualquier caso es importante que te informes bien antes de hacer cambios que pueden *dañar tu instalación de Python gravemente*.

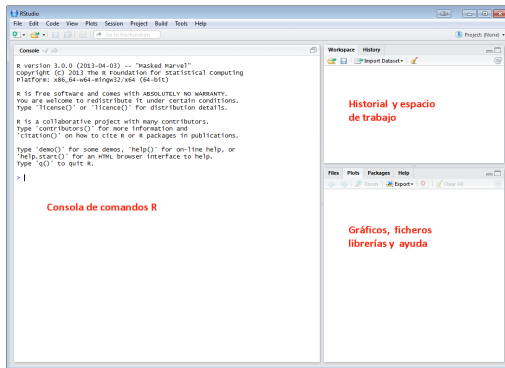
WSL 2.

- Si eres usuario de una versión reciente de Windows 10 te puede interesar conocer el [Windows Subsystem for Linux](#) y usarlo para acceder a muchas de las herramientas de las que vamos a hablar. Y en particular puedes usarlo para disponer de un entorno de desarrollo en Python. Pregunta a tu profesor antes de empezar si te interesa este tema: la situación de WSL es muy fluida y a menudo se producen cambios sustanciales en esa herramienta.

Sección 3


La interfaz de RStudio

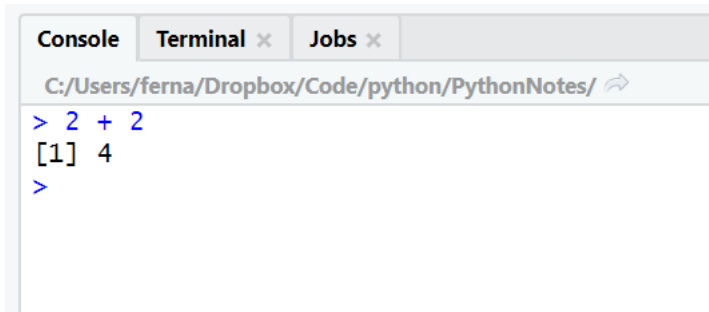
Primer Contacto



- Abre RStudio. Inicialmente verás que hay tres paneles principales, que vamos a ir conociendo:

- 1 Consola.
- 2 Historial / Entorno / Git.
- 3 Gráficos / Ficheros / Ayuda / Paquetes

- Para empezar, haz clic en la consola (el panel de la izquierda) en la línea con el prompt (símbolo >). Verás el cursor parpadeando. Escribe 3^2 y pulsa 



```
> 2 + 2
[1] 4
>
```

- El uno entre corchetes es la forma en la que R nos indica que esta es la primera línea de su respuesta. Enseguida veremos respuestas que ocupan varias líneas. Debajo hay un nuevo prompt listo para seguir trabajando.

Ejercicio 1: R como una calculadora

- Copia o teclea cada línea en el prompt (¡practica las dos maneras!), una por una. Trata de adivinar el resultado de cada operación antes de ejecutar el código:.

```
2 + 3  
15 - 7  
4 * 6  
13/5  
1/3 + 1/5  
sqrt(25)  
sqrt(26)  
sin(pi)  
sin(3.14)
```

Comentarios sobre el Ejercicio 1

- Usa paréntesis para controlar mejor el orden de las operaciones.
- Los espacios a menudo son irrelevantes pero a veces son esenciales. Por ejemplo es igual escribir

$$3 + 5$$

que $3+5$. Usa espacios extra para ganar claridad. Pero **no escribas**

`sqrt (25)`



en lugar de `sqrt(25)`. Funciona pero ¡es feo y difícil de leer!

- `sin` es la función *seno* y `pi` se refiere a la constante matemática $\pi \approx 3.141593$. Enseguida hablaremos de otras funciones de R.
- R es un lenguaje **numérico** (no es *simbólico*). No hemos obtenido 0 (que es la respuesta exacta), sino un valor aproximado muy pequeño (en notación científica):
`1.224606e-16`

Ejercicio 2: Errores.

- Prueba ahora a escribir esta expresión incompleta:

```
3 +
```

y pulsa . La respuesta + es la forma que tiene R de decirnos *"necesito algo más"* (no tiene nada que ver con la suma). Lo mejor es que uses la tecla  y completes la expresión.

- Escribe y ejecuta una a una estas expresiones para ver distintos tipos de errores:

```
4/*3  
2/0  
sqrt(-4)
```

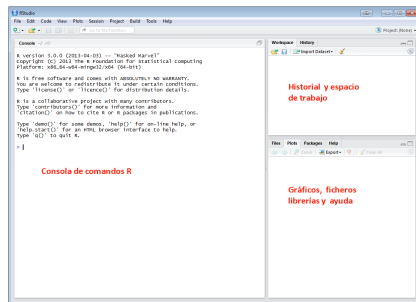
- Escribe y ejecuta





```
Sqrt(4)
```

Error in Sqrt(4): no se pudo encontrar la función "Sqrt"

y verás que se produce un error. **Es muy importante recordar que R siempre distingue mayúsculas de minúsculas.**

Detalles adicionales sobre RStudio.

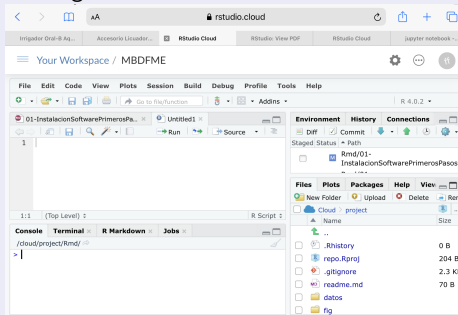


- **Historial de comandos:** Cuando estás trabajando en la consola puedes pulsar la flecha  repetidamente, e irás viendo pasar todos los comandos previos. Con  recorres la lista en sentido contrario. Además en el panel History (arriba a la derecha) puedes ver esa lista y copiar/pegar los comandos o guardarlos como fichero de texto.
- **Limpieza de la consola:** Pulsa  + . Esto no afecta al historial de comandos.
- **Ayuda;** El panel Help (abajo a la derecha) nos servirá, cuando aprendamos más, para acceder al sistema de ayuda de R.

Otras opciones de trabajo.

RStudio Cloud

- Los creadores de RStudio ponen a nuestra disposición una versión web gratuita de la aplicación, llamada **RStudio Cloud**, que puede ser muy útil. Aunque se accede a través del navegador la interfaz es muy similar e incluso los atajos de teclado coinciden mayoritariamente. Esta opción puede resultar especialmente útil cuando se combina con el uso de repositorios en GitHub; por ejemplo para trabajar desde una tablet como se está haciendo en la figura:

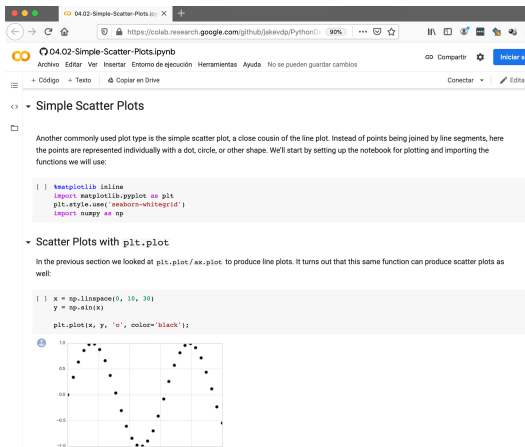


- Recomendamos crear una cuenta gratuita de RStudio Cloud para poder experimentar con la herramienta y para disponer de una forma de acceder a R en caso de necesidad. Haremos algunos experimentos con RStudio Cloud en las próximas sesiones del curso.

Jupyter

- En el trabajo con Python es frecuente utilizar como herramienta de desarrollo los [Jupyter Notebooks](#). Los Notebooks son un tipo de documento que permite combinar código ejecutable con documentación (texto, figuras, ecuaciones, enlaces). El Notebook puede ejecutarse desde un navegador y los resultados de la ejecución del código se incorporan al propio documento. Aunque los Notebooks de Jupyter se usan habitualmente como una herramienta para código Python, también ofrecen soporte para R y otros lenguajes.
- Se pueden instalar los Jupyter Notebooks en local como parte de una instalación estándar de Python ([si es que tal cosa existe](#)). Pero también existen opciones de acceso remoto a servidores Jupyter públicos y gratuitos (hay asimismo servidores de pago). En particular, si disponéis de una cuenta en Google os recomiendo [Google Colab](#), que permite incluso acceder a GPUs para aplicaciones de Deep Learning. Además usando [este enlace](#) podéis utilizar Google Colab para crear Notebooks del lenguaje R.

- La figura muestra un Jupyter Notebook abierto en Google Colab.



Fíjate en la combinación de texto, código y resultados de la ejecución del código (en este caso figuras). Este ejemplo forma parte del [Python Data Science Handbook](#) de Jake VanderPlas, que está disponible gratuitamente en ese enlace.

Sección 4

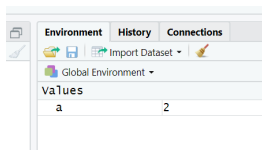
Variables y asignaciones en R.

Variables

- Una *variable*, en R, es un símbolo que usamos para referirnos a un valor. Por ejemplo, sitúate en la Consola de Comandos y teclea

```
a <- 2
```

Ahora ejecuta esa instrucción. Aunque aparentemente no ha sucedido nada (porque no hay respuesta), a partir de ese momento R ha asignado el valor 2 al símbolo a. Fíjate en el panel superior derecho llamado *Environment* (*entorno*), en el que aparece el valor actual de la variable.



- Así que si, por ejemplo, tecleas y ejecutas:

```
a + 1
```

obtendrás como respuesta 3. Comprueba en el panel de entorno que el valor de a no ha cambiado. ¿Qué sucede si ejecutas `a <- a + 1`? ¿Y si ejecutas `a <- A + 1`?

Ejercicio 3. Operaciones, asignaciones y resultados.

- Ejecuta estos comandos, uno detrás de otro. Trata de imaginar el valor de las variables tras cada operación.

```
a <- 2
b <- 3
c <- a + b
a <- b * c
b <- (c - a)^2
c <- a * b
```

Consulta el panel de entorno para ver cuanto valen las variables tras cada operación.

- Una orden como `c <- a + b` en la que guardamos en una variable el resultado de una operación es una **asignación**. Por defecto R no muestra el resultado de las asignaciones al ejecutarlas. Si quieres ver el resultado encierra entre paréntesis *toda la asignación*

```
(c <- a + b)
```

```
[1] 115
```

Como ves, ahora R sí muestra el resultado.

- Es recomendable usar nombres informativos para las variables y jugar con la tipografía para hacerlos más legibles, como `precio_final` o `poblacion1995`. Los nombres de variables no pueden empezar por un número ni contener caracteres especiales.
- Aunque puedes usar el igual `=` para asignaciones, es importante que sepas que en R se usa a menudo el símbolo `<=` para las asignaciones, de manera que

```
a = 2
```

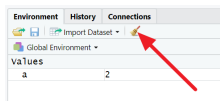
es equivalente a

```
a <= 2
```

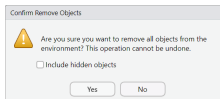
De hecho el uso de `=` para asignaciones [se desaconseja](#) en la comunidad de usuarios de R. En RStudio con Windows puedes usar el atajo de teclado `Alt` + `=`, mientras que en Mac usaremos `⌘` + `=`. Si quieres leer algo más sobre esta discusión (bizantina), [puedes hacerlo aquí](#). En realidad este párrafo es una excusa para introducir enlaces a Stackoverflow, y a una guía de estilo del código de R.

Haciendo limpieza antes de seguir adelante.

- Vamos a aprender a eliminar cualquier resto de nuestro trabajo anterior de la memoria de R. Esto es necesario a menudo cuando empezamos un nuevo análisis de datos, para evitar posibles errores debidos a la permanencia de esos resultados previos.
- Empieza usando el menú `Session` `Restart R`. Verás aparecer el mensaje `Restarting R session...` en la Consola.
- Con esto no hemos acabado. Si miras el panel de entorno verás que esa operación no ha eliminado los valores asignados a variables. Para hacer que R olvide esos valores podemos usar el icono de una escoba que aparece encima de ese panel de entorno.



Al pulsarlo aparecerá un mensaje de confirmación.



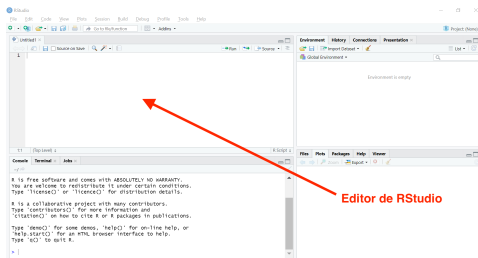
Acepta haciendo clic en `Yes` y estaremos listos para seguir. Veremos más adelante formas alternativas de hacer esto.

Sección 5

El editor de RStudio. Ficheros de comandos de R.

Abriendo el editor

- El trabajo en la Consola de Comandos puede resultar conveniente para algunas operaciones básicas. Pero resulta incómodo para cualquier análisis de datos salvo los más triviales. Vamos a ver otra forma mejor de trabajar.
- Pulsa `ctrl` + `⇧` + `N` (o, en un Mac, `⌘` + `⇧` + `N`). Alternativamente usa el menú `File` » `New File` » `R Script`. Verás aparecer un panel nuevo en la parte superior izquierda de la ventana, el *Editor* de RStudio.



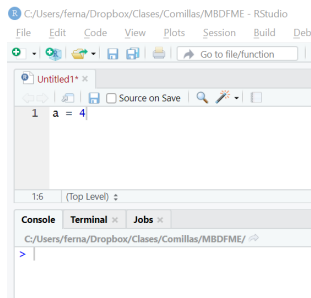
- El editor va a ser el escenario principal de nuestro trabajo con RStudio.




Ejecutando código en el editor

- Empieza escribiendo esta asignación en el editor

```
a <- 4
```

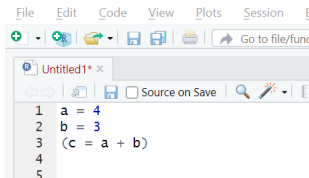
para obtener algo como



Si ahora pulsas la tecla  verás que ese código no se ejecuta (mira la consola y el entorno), simplemente pasas a la siguiente línea del editor. Para ejecutar el código vuelve a hacer clic en la línea a <- 4 y ahora pulsa a la vez  + . Veras que el código aparece copiado en la consola como si lo hubieras ejecutado allí.

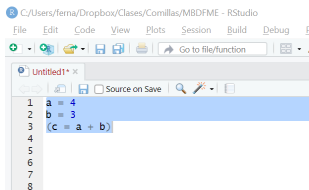
Ejecutando varios comandos a la vez.

- Para empezar a ver las ventajas de esta forma de trabajar, teclea en el editor dos líneas más de código:



```
File Edit Code View Plots Session E
+ [New File] [New Plots Window] [Save] [Run] [Go to file/function]
Untitled1* x
[Previous] [Next] [Source on Save] [Find] [Run] [Debug]
1 a = 4
2 b = 3
3 (c = a + b)
4
5
```

y ahora selecciona esas líneas (usando el ratón o el teclado). Asegúrate de que las tres líneas aparecen completamente seleccionadas, como se muestra aquí:

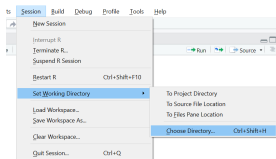


```
C:/Users/ferna/Dropbox/Clases/Comillas/MBDFME - RStudio
File Edit Code View Plots Session Build Debug P
+ [New File] [New Plots Window] [Save] [Run] [Go to file/function] [Full Screen]
Untitled1* x
[Previous] [Next] [Source on Save] [Find] [Run] [Debug]
1 a = 4
2 b = 3
3 (c = a + b)
4
5
6
7
8
```

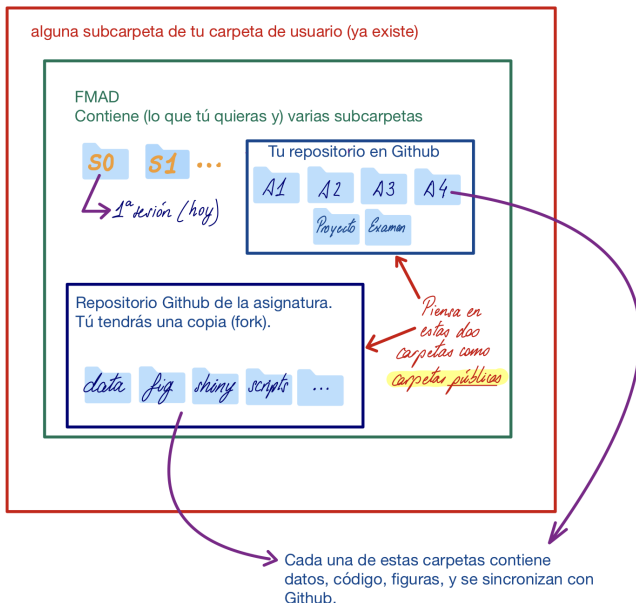
- Si ahora usas a la vez **ctrl** + **↵** podrás comprobar en la consola que las tres líneas de código se han ejecutado todas una tras otra.



Ficheros de comandos de R. Directorio de trabajo.

- Vamos a aprender a guardar en un fichero de comandos (*script*) el trabajo que hacemos en el Editor de RStudio.
- Cada sesión de trabajo con R utiliza una carpeta de nuestro ordenador llamada *Working Directory* (*Directorio de Trabajo*) para almacenar datos, ficheros de comandos, etc. Es necesario que elijas un directorio de trabajo para este curso y que te acostumbres, al principio de cada sesión, a indicarle a R cuál es ese directorio.
- Dentro de tu carpeta de usuario crea una carpeta para los ficheros de esta asignatura (llámala por ejemplo FMAD o Fundamentos, aunque el nombre no es importante). Si tu nombre de usuario contiene acentos, espacios, la letra ñ, etc. y experimentas algún problema al usar R, habla con tu profesor. Finalmente dentro de esta crea una carpeta llamada S0 para la sesión de hoy. La figura de la siguiente página ilustra la estructura de carpetas que vamos a usar en la asignatura.
- Ahora usa el menú **Session > Set Working Directory > Choose Directory**:



Estructura de carpetas.



- A continuación usa el menú   y guarda el fichero de instrucciones de R con el nombre `sesion01-00` en el directorio de trabajo. Al guardarlo RStudio le añadirá la extensión `.R`, que identifica a los ficheros de comandos de R.
- Es recomendable que uses el *Explorador de Archivos* de Windows (o el *Finder* de Mac, etc.) para navegar a la carpeta en la que has guardado ese fichero y lo abras con un *editor de texto* (como el Bloc de Notas). Habla con tu profesor si tienes dudas de como hacer esto. Comprueba que se trata de un fichero de texto que contiene simplemente los comandos que hemos escrito en el Editor de RStudio.
- También puedes abrir un fichero de comandos escrito por otra persona. De hecho en el curso vamos a usar a menudo ficheros de comandos prediseñados para dirigir nuestro trabajo. Empieza descargando este fichero de comandos [sesion01-01.R](#) y guárdalo en tu directorio de trabajo. En la próxima sección lo abriremos y usaremos para seguir avanzando con R. **IMPORTANTE:** para llevar a cabo la descarga lee la siguiente página.

Instrucciones de descarga de ficheros.

- **IMPORTANTE:** En muchas ocasiones te vamos a pedir que descargues ficheros de datos, código, etc. Dependiendo del navegador, sistema operativo y servidor que aloje esos ficheros, la descarga puede presentar problemas que complicarán nuestro trabajo. Para evitarlos vamos a usar una serie de comandos de R que te servirán con cualquiera de esos ficheros simplemente **copiando su enlace de descarga dentro de las comillas de la variable file_url y ejecutando estos comandos.**

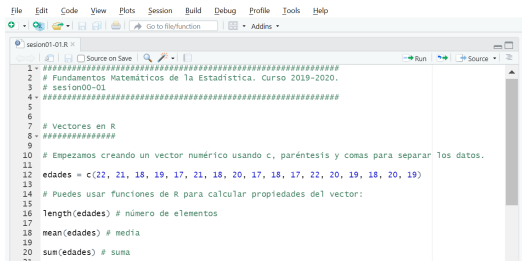
```
# Comprobamos el directorio de trabajo
getwd()
# Datos del fichero. La url, incluyendo el nombre del fichero.
file_url <- ""
# Cambia la siguiente fila si quieres usar una subcarpeta del wd.
local_folder <- "/"
# No cambies nada a partir de aquí
file_name <- tail(unlist(strsplit(file_url, split = "/")), 1)
localFile <- paste0(local_folder, file_name)
if(!file.exists(localFile)){
  download.file(url = file_url, destfile = localFile)
} else {
  warning(paste0("Cuidado: el fichero de datos", localFile, " ya existe."))
}
```

- Te recomiendo que abras un nuevo script de R y que copies estos comandos en ese script. Llámalo herramientas.R o algo parecido y guárdalo en la carpeta de la asignatura, donde puedas localizarlo fácilmente. Si lo dejas abierto en la primera pestaña del editor de RStudio siempre podrás volver a usar estos comandos fácilmente.

Sección 6

Estructuras de datos básicas en R.

- Ahora usa el Menú **File** ➤ **Open File** y abre el fichero `sesion01-01.R` que acabas de descargar. Se abrirá en el Editor.



The screenshot shows the RStudio interface with the 'sesion01-01.R' file open in the editor. The menu bar at the top includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar below the menu bar contains icons for opening files, saving, and running code. The editor window displays the following R code:

```
1 - #####
2 # Fundamentos Matemáticos de la Estadística, Curso 2019-2020.
3 # sesion00-01
4 - #####
5
6
7 # Vectores en R
8 #####
9
10 # Empezamos creando un vector numérico usando c, paréntesis y comas para separar los datos.
11
12 edades = c(22, 21, 18, 19, 17, 21, 18, 20, 17, 18, 17, 22, 20, 19, 18, 20, 19)
13
14 # Puedes usar funciones de R para calcular propiedades del vector:
15
16 length(edades) # número de elementos
17
18 mean(edades) # media
19
20 sum(edades) # suma
21
```

Si tienes problemas con los acentos al abrir el fichero usa el menú

File ➤ **Reopen with Encoding**

y si los problemas persisten pregunta a tu profesor.

Nosotros usamos (y recomendamos) UTF-8 como encoding por defecto en todos los ficheros.

- Fíjate en que R usa el símbolo `#` para introducir comentarios en un fichero de comandos e ignorará el resto de la línea a partir del `#`. Vamos a usar los comentarios de ese fichero como guía para aprender las propiedades más básicas de los vectores en R. Volveremos aquí después de recorrerlo.

Tablas (Data Frames)

- Junto con los vectores, las tablas y listas constituyen las estructuras de datos básicos de R. Vamos a ver ahora el primero de estos dos tipos de estructuras. En el sistema base de R se denominan `data.frames`. Las tablas de datos son muchas veces el punto de partida de nuestros análisis. Pronto aprenderemos a abrir tablas almacenadas en ficheros externos de datos y a usar funciones relacionadas del tidyverse. Pero para empezar vamos a usar una tabla de datos de ejemplo que R trae incorporada al instalarlo.

- Ejecuta este código:

```
head(iris)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

- La tabla contiene datos sobre 150 flores de 3 especies de iris. La función `head` hace que R solo nos muestre el encabezamiento y las primeras 6 filas de la tabla (con `tail` verías las 6 últimas). Prueba a ejecutar `head(iris, 12)`.
- Si ejecutas `View(iris)` la tabla completa se abrirá en otra pestaña del editor.

- Fíjate en la tabla

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

y observa que:

- ▶ cada columna corresponde a una *variable* (una propiedad observable). Todos los elementos de una columna son del mismo tipo (homogéneos).
- ▶ cada fila corresponde a una *observación* o *individuo* (una flor en este ejemplo). Los elementos de una misma fila pueden ser de tipos distintos (heterogéneos).

- Podemos obtener las dimensiones (número de filas y columnas) de la tabla con

```
dim(iris)
```

```
[1] 150  5
```

También podemos obtener los números de filas y columnas directamente con `nrow` y `ncol`. ¡Compruébalo!

Selección de elementos de una tabla

- Para acceder al elemento en la fila 2, columna 3 podemos usar la notación de corchetes:

```
iris[2, 3]
```

```
[1] 1.4
```

Ese acceso permite cambiar los elementos de la tabla directamente:

```
iris[2, 3] <- 7  
head(iris)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 7.0 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

Selección de filas y columnas de una tabla

- Puesto que las columnas están formadas por elementos homogéneos podemos extraer una columna y obtenemos un vector de R. Hay dos formas de hacer esto:
 - 1 usando la notación de corchetes con el número de columna y dejando el número de fila en blanco (solo se muestra el principio de la columna).

```
iris[ , 3]
```

```
[1] 1.4 7.0 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1  
[15] 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5  
[29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4
```

- 2 Usando el nombre de la columna y el símbolo \$ así:

```
iris$Petal.Length
```

```
[1] 1.4 7.0 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1  
[15] 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5  
[29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4
```

Esta segunda opción es preferible cuando es cómoda. Veremos formas de hacer esto con dplyr más adelante.

- Para seleccionar una fila podemos usar corchetes, pero al ser heterogénea el resultado es una nueva tabla:

```
iris[2, ]
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 2 | 4.9 | 3 | 7 | 0.2 | setosa |

- También podemos seleccionar más de una fila o columna. Aquí vamos a seleccionar las filas de la 49 a la 52 y las columnas 1, 3 y 5:

```
iris[49:52, c(1, 3, 5)]
```

| | Sepal.Length | Petal.Length | Species |
|----|--------------|--------------|------------|
| 49 | 5.3 | 1.5 | setosa |
| 50 | 5.0 | 1.4 | setosa |
| 51 | 7.0 | 4.7 | versicolor |
| 52 | 6.4 | 4.5 | versicolor |

- Al igual que con los vectores también podemos seleccionar elementos de una tabla usando condiciones lógicas (típicamente comparaciones). Por ejemplo, para seleccionar las filas que corresponden a flores con longitud de sépalo mayor que 2 usamos:

```
iris[iris$Sepal.Width > 2, ]
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 7.0 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

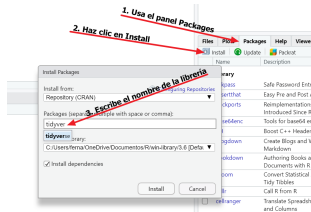
(sólo se muestran las primeras líneas del resultado). Fíjate en que hemos dejado en blanco el número de columna después de la coma para seleccionar todas las columnas. También podríamos haber combinado una condición sobre las filas con otra para seleccionar columnas.

Sección 7

Librerías de R

Instalación y carga de librerías

- La instalación básica de R contiene muchas herramientas pero hay además miles de librerías adicionales disponibles, muchas de ellas muy especializadas para temas concretos (por ejemplo *Caret* para Machine Learning, *BioConductor* para Bioinformática). Nosotros vamos a utilizar mucho una librería de herramientas llamada *tidyverse* de H. Wickham.
- Para usar una librería de R debemos instalarla (una única vez) y cargarla (en cada sesión en la que vayamos a usarla). Para instalar una librería sigue los pasos que aparecen en la figura;



R descargará e instalará la librería. Si tienes problemas, pide ayuda al profesor. Y por si sirve de ayuda aquí tienes un [vídeo ilustrando el proceso](#).

- Cuando la instalación concluya puedes cargar la librería para usarla ejecutando

```
library(tidyverse)
```

- **Ejercicio** Además de la anterior instala las librerías llamadas `gapminder` y `nycflights13`.

Carga la segunda de estas librerías con

```
library(nycflights13)
```

Esta librería contiene datos de todos los vuelos que despegaron de aeropuertos de Nueva York en 2013.

- **Ejercicio** ¿Cuántas filas y columnas tiene la tabla? ¿Cuántos vuelos despegaron del aeropuerto JFK? Indicación: usa la función `str` para empezar. ¡¡Es una de las funciones esenciales de R!!

Un primer encuentro con dplyr

- La librería dplyr es uno de los componentes básicos del *tidyverse* y permite simplificar mucho nuestro trabajo con tablas como la que acabamos de examinar. En concreto dplyr proporciona un conjunto de funciones que incluye entre otras:
 - `filter`: para seleccionar algunas filas según sus valores.
 - `arrange`: para reordenar las filas.
 - `select`: para seleccionar columnas (variables)
 - `mutate`: para añadir nuevas columnas calculadas a partir de las existentes.
 - `summarize`: para obtener estimadores estadísticos, por ejemplo medias.
- Además con dplyr vamos a empezar a usar el operador pipe, que en R es `%>%`. Este operador pasa el resultado de una operación como primer argumento de la siguiente. El uso de `%>%` normalmente resulta más claro que la sintaxis básica de R.
- Por ejemplo usando `iris` vamos a buscar, en las columnas que se refieren a pétalos, cuáles son las flores con anchura de pétalo mayor que 2.3.

```
iris %>%  
  select(c('Petal.Length', 'Petal.Width')) %>%  
  filter(Petal.Width > 2.3)
```

| | Petal.Length | Petal.Width |
|---|--------------|-------------|
| 1 | 6.0 | 2.5 |
| 2 | 6.1 | 2.5 |
| 3 | 5.1 | 2.4 |
| 4 | 5.6 | 2.4 |
| 5 | 5.6 | 2.4 |
| 6 | 5.7 | 2.5 |

Y un primer encuentro con ggplot

- La librería `gapminder` que hemos instalado antes contiene una tabla de datos con algunas características socioeconómicas de casi todos los países del mundo a lo largo de varios lustros (proceden de www.gapminder.org). Carga la tabla y examínala con

```
library(gapminder)
View(gapminder)
```

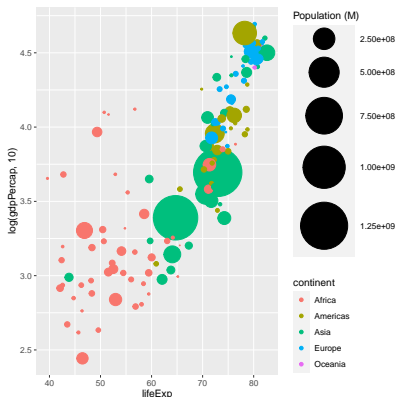
- La librería `ggplot` es otro de los componentes del `tidyverse` y proporciona herramientas para construir gráficas muy útiles. Su uso puede resultar un poco complejo al principio, pero pronto te acostumbraras. La plantilla básica de un gráfico con `ggplot` es

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Vamos a usar `dplyr` y `ggplot` para dibujar un gráfico de la relación entre esperanza de vida y renta per cápita para el año 2007.

- Los colores corresponden a los distintos continentes y el tamaño de cada punto indica la población del país. La escala vertical (renta per cápita) es logarítmica de base 10. El código es:

```
gapminder %>%  
  filter(year == 2007) %>%  
  ggplot() +  
  geom_point(mapping = aes(x = lifeExp, log(gdpPercap, 10),  
                           color = continent, size = pop)) +  
  scale_size(range = c(.1, 24), name="Population (M)")
```



¿Qué preguntas te haces a la vista de este gráfico?

Sección 8

Rmarkdown para la creación de documentos

- En este curso vamos a iniciarnos en el manejo de Rmarkdown y, progresivamente, lo usaremos cada vez más. El objetivo es que el trabajo personal y el examen de la asignatura se realicen en este formato.
- Rmarkdown, creado por Yihui Xie, es una herramienta muy general para la creación de documentos, en especial documentos relacionados con el Análisis de Datos. A partir de los ficheros escritos con Rmarkdown se pueden obtener fácilmente salidas en formato pdf o HTML, pero también presentaciones, artículos e informes técnicos, entradas de blog, libros, dashboards con Shiny y la lista sigue creciendo. Aprovecharemos las secciones finales de nuestras sesiones para presentar algunas de estas posibilidades.
- Recomendamos empezar leyendo una [introducción básica a RMarkdon](#) de T. Goicoa (Univ. Pública de Navarra).
- Hay dos fuentes de información online básicas para adentrarse en las posibilidades de RMarkdown:
 - ▶ Del propio Yihui Xie: [R Markdown: The Definitive Guide](#).
 - ▶ De RStudio: [R Markdown: Get Started](#).

¿Qué es un documento RMarkdown?

- Un **documento Rmarkdown** es un fichero de texto con extensión *.Rmd* que contiene una mezcla de texto, comandos de formato (secciones, listas, negritas) y para inclusión de figuras, enlaces, etc. En particular se pueden usar comandos $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ para crear ecuaciones.
- Pero además, y de forma característica, el fichero contiene bloques (en inglés, *chunks*) con **comandos de código R**. Esos comandos de código R se evalúan como parte del procesamiento del documento para crear *documentos dinámicos*, similares a los notebooks de Jupyter, pero aún más flexibles.
- Usaremos como ejemplo un fichero Rmarkdown sencillo, llamado [rmarkdownDemo.Rmd](#) disponible en el repositorio. A partir de ese documento Rmd obtendremos salidas en formatos pdf, HTML y docx (MS Word). En todas estas opciones de salida se mantienen los formatos de texto, ecuaciones, figuras, enlaces y, por supuesto, el resultado del código en R.


Otros formatos y capacidades.

- Los tres formatos de salida ilustran las capacidades más básicas de Rmarkdown. Por ejemplo, se puede utilizar Rmarkdown para:
 - ▶ Crear **presentaciones** en Beamer y otros formatos. Por ejemplo las presentaciones de este curso se han escrito íntegramente en Rmarkdown.
 - ▶ Escribir **artículos científicos** y libros (con [bookdown](#)).
 - ▶ Mantener sitios Web y [blogs](#), usando [blogdown](#).
 - ▶ Crear **dashboards** con Shiny para visualizaciones interactivas. Lo mejor es ver los ejemplos de la [galería](#).
- Además los bloques de código no se limitan a R. Se puede usar **otros lenguajes**, entre otros, R, Python y SQL. El soporte para Python en particular es muy bueno como veremos más adelante.

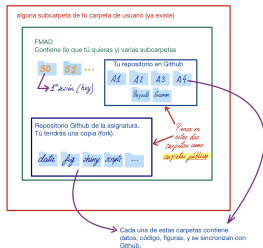
Sección 9

Introducción al uso de GitHub para el curso

- El material de las sesiones del curso está alojado en un [repositorio en GitHub](#).
- Si no estás familiarizado con Git, el curso 0 te proporcionará una introducción y vamos a aprovechar esta asignatura para practicar el uso de una herramienta tan importante. Nuestro recurso favorito para aprender a combinar el uso de R y Git es el libro/web [Happy Git and GitHub for the useR](#). En la primera sesión de prácticas daremos más detalles de la forma en la que vamos a usar Git en este curso, pero para ir adelantando trabajo puedes leer las primeras siete secciones de *Happy Git*. No te llevará demasiado tiempo. En esas secciones se presenta Git y se describe el proceso básico de instalación del software necesario. **Te recomendamos encarecidamente** que primero leas esas siete secciones sin instalar ni hacer nada en tu ordenador, luego vuelvas aquí, termines de leer esta sección y **entonces** empieces a instalar cosas, crear cuentas, etc.
- Como complemento, puedes ver los vídeos 14 y 15 de esta [playlist](#) del curso Data Science Toolbox de la Universidad John Hopkins para Coursera. También puedes encontrar información en este [curso de Introducción a GitHub](#). Pregunta a tu profesor si necesitas más ayuda para empezar.

- 1 Crea una cuenta en [GitHub](#) (es gratuito). **IMPORTANTE:** aunque ya tengas una cuenta en GitHub, para esta debes utilizar tu dirección de correo electrónico de la universidad. Si ya tienes una cuenta y tienes dudas sobre cómo proceder ¡consulta a tu profesor antes de estropear algo!
- 2 **Lee el siguiente enlace antes de elegir un nombre de usuario.**
[Happy Git: username-advice](#)
Es imprescindible que envíes a tu profesor el nombre de usuario que hayas elegido lo antes posible.
- 3 Instala Git en tu ordenador si aún no lo has hecho. Sigue las instrucciones de [este enlace](#).
- 4 Instala las librerías `usethis` y `gitcreds` de R.
- 5 Ve al [repositorio de la asignatura en GitHub](#). Asegúrate de que estás registrado con el usuario que has creado (usa el botón *Sign in* en la esquina superior derecha de la página si no es así).
- 6 Usa el botón verde  y en la ventana que aparece copia la URL del repositorio haciendo clic en el símbolo del portapapeles.

7 En RStudio usa el menú **File** » **New Project** y en la ventana que aparece selecciona *Version Control*. A continuación selecciona *Git* y en el primer campo copia la URL en GitHub del paso anterior. Acepta *fmad2122* como nombre del repositorio y, **muuy importante**, con el botón *browse* selecciona la ubicación del clon local del repositorio. Deberías situarlo dentro de tu carpeta de la asignatura; la que hemos llamado FMAD en esta figura:



Por ejemplo en Windows una ubicación típica podría ser algo como:

```
C:/Users/tu_nombre/.../MasterBD/FMAD/
```

mientras que en Mac sería algo como:

```
/Users/tu_nombre/.../MasterBD/FMAD/
```

En ambos casos los puntos se refieren a subcarpetas intermedias que puedas querer usar (como OneDrive, Dropbox o cualquier otra).

Manteniedo la copia local actualizada

- Aunque Git permite mantener sincronizada tu copia con los cambios que se vayan produciendo en el repositorio de la asignatura, si te has acostumbrado al funcionamiento de soluciones como Dropbox, OneDrive, iCloud y otras similares, al principio te costará un poco de trabajo comprender que **en Git la sincronización no es automática**. Es razonable que sea así puesto que se trata de un sistema pensado para el control de versiones en el desarrollo de software: nadie quiere que los cambios en ese tipo de trabajo sean *demasiado automáticos*, eso causaría muchos más problemas de los que solucionaría.
- Por eso, cuando los profesores del curso hagan cambios en el repositorio *fmad2122* deberás pedir explícitamente a Git que sincronice esos cambios con tu copia local. La forma más sencilla de hacer esto desde RStudio es abrir el proyecto asociado a ese repositorio y en el panel Git usar el botón `pull`. A lo largo del curso te recordaremos varias veces la necesidad de hacer esto con regularidad hasta que lo conviertas en una acto reflejo cada vez que empieces el trabajo.

Enlaces

Junto con el material alojado en Moodle tenéis a vuestra disposición estos recursos:

- [Código de esta sesión](#)
- [R for Data Science \(Wickham\)](#).
- [Repositorio del curso](#) en GitHub.
- [R Markdown: The Definitive Guide \(Yihui Xie\)](#).
- [Happy Git and GitHub for the useR \(Jenny Bryan\)](#)
- Web del libro [PostData](#).
- [Resumen de uso de dplyr](#) elaborado por RStudio.

Bibliografía

Wickham, H., & Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc.