

# INTRODUCCIÓN

## Fundamentos de Data Science con R

---

Arturo Chian



Un presentación BEST <http://besteamperu.org/>

# Objetivos

- ➡ Conocer el término Data Science, en qué se diferencia de estadística y qué habilidades requiere.
- ➡ Conocer el rol de R en el mundo de la ciencia de datos.
- ➡ Distinguir entre R y RStudio.
- ➡ Entender funciones básicas de R.
- ➡ Saber cómo crear tu primer producto con R: un reporte en HTML.
- ➡ Conocer un poco acerca del universo tidyverse.
- ➡ Contar con un nivel básico de dplyr.



“

*For a long time I have thought I was a statistician, interested in inferences from the particular to the general. But as I have watched mathematical statistics evolve, I have had cause to wonder and to doubt. ... All in all I have come to feel that my central interest is in data analysis, which I take to include, among other things: procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.*

**Tukey (1962). *The future of Data Analysis, The Annals of Mathematical Statistics***



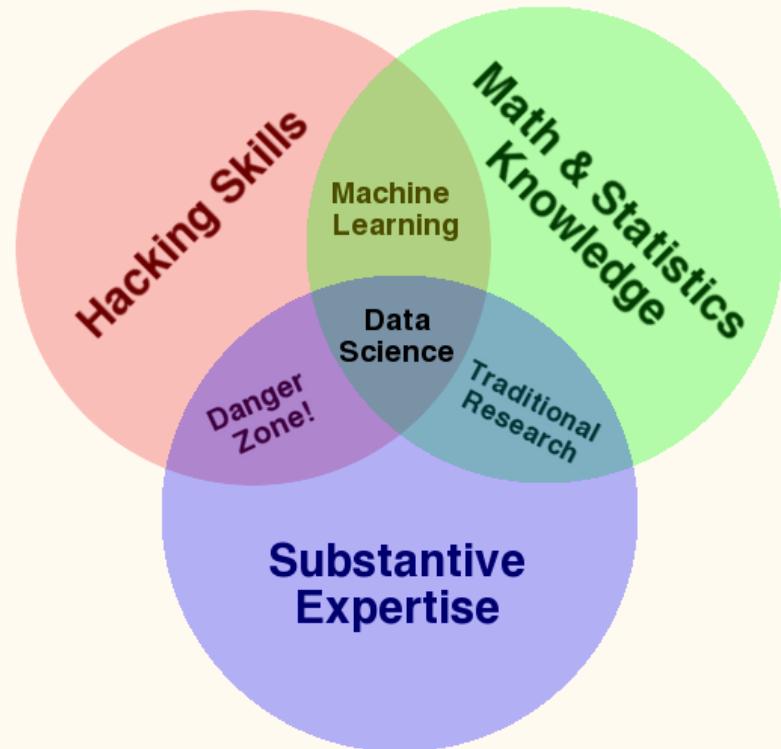
‘

Hace 50 años, John Tukey llamó a una reforma académica en estadística, a través de uno de los más importantes papers de esa época, llamado “The Future of Data Analysis”, donde señalaba la necesidad futura de una ciencia cuyo interés sea aprender de la data o análisis de datos. Hace unos 20 a 10 años, John Chamber, Jeff Wu, Bill Cleveland y Leo Breiman, dieron una serie de argumentos, de forma independiente sobre expandir los límites de la estadística teórica: Chambers enfatizaba la importancia de la preparación de datos, más que el modelaje estadístico; Breiman, prefería enfatizar la predicción antes que la inferencia; y Cleveland y Wu sugerían llamar a este nuevo campo Data Science por su estrecha relación a la data.

**Arturo Chian (2018). A propósito de los 25 años de R y 50 años de Data Science (Parte 1), Blog de Behavioral Economics & Data Science Team**

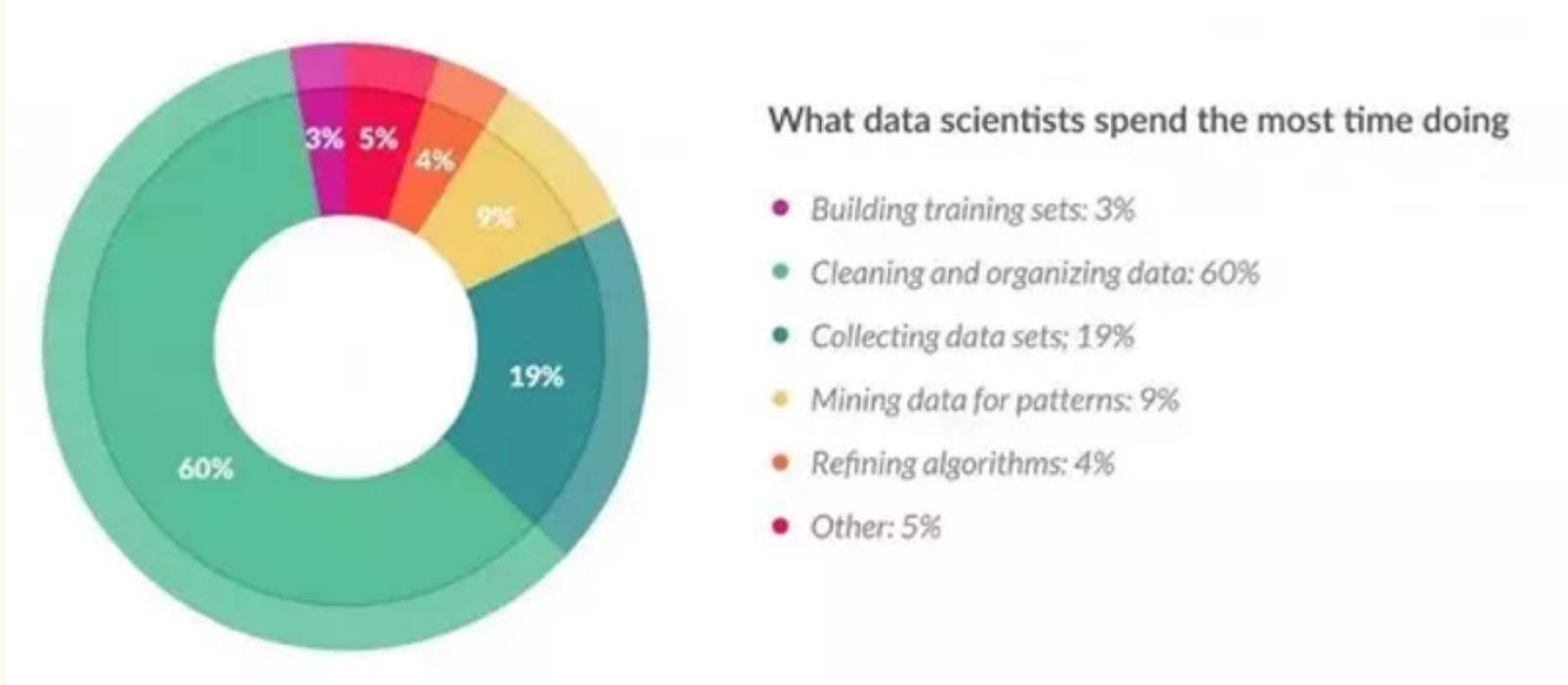


## Diagrama de Venn - Drew Conway



1. **Hacking Skills:** Capacidad de resolver problemas programando.
2. **Math & Statistics knowledge:** Aplicar de forma correcta estadística.
3. **Conocimiento de experto:** Comprender la data en su campo de investigación (economía, biología, psicología, derecho, etc).

# ¿Qué hace un Data Scientist en el día a día? 🧠



# tips y visión general a R



# ¿Cómo muchos hemos aprendido a usar R?



# Diferenciando entre R y RStudio

## R Studio

Es uno de los más IDE más usados.  
Facilita al usuario para programar R y  
otros lenguajes.

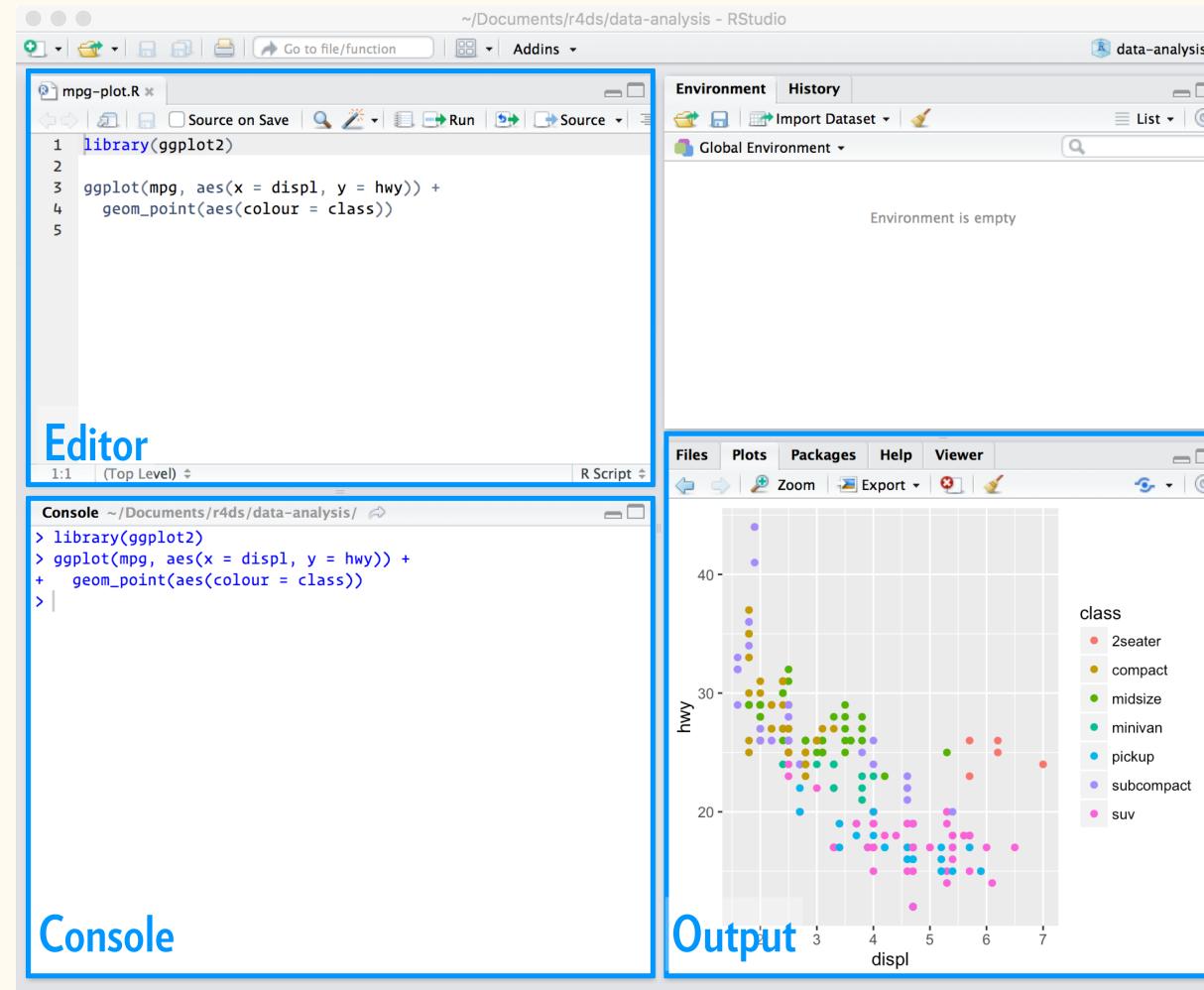


## R

Es el software estadístico por  
excelencia. ***La lingua franca*** en  
estadística.



# Veamos primero la terminología



# Tip 1: Trabaja de forma organizada.

‘

*One day you will need to quit R, go do something else and return to your analysis later.*

*One day you will have multiple analyses going that use R and you want to keep them separate.*

*One day you will need to bring data from the outside world into R and send numerical results and figures from R back out into the world.*

*To handle these real life situations, you need to make two decisions:*

- 1. What about your analysis is “real”, i.e. you will save it as your lasting record of what happened?*
- 2. Where does your analysis “live”?*

**Jenny Bryan**

# ¿Qué hace que el análisis sea real?

Se refiere al proceso de guardar tus archivos con un nombre. Para ello deben ingresar a file, save as...

Es muy común manejar diversos untitled pero podría perderse todos tus avances un día, por lo que no se recomienda trabajar ahí, salvo por temas temporales y cortos.

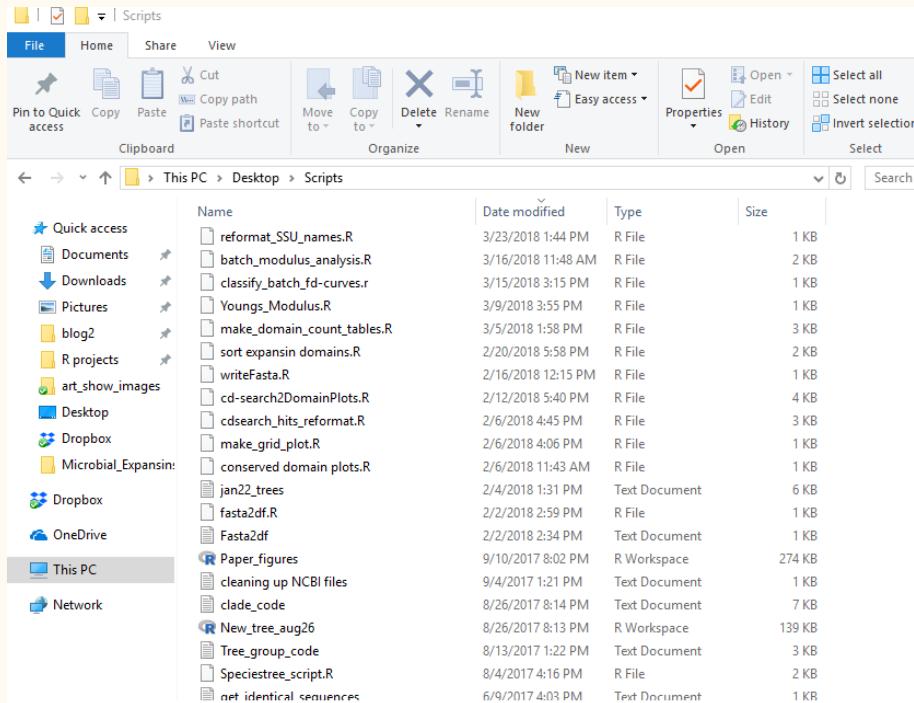
# ¿Dónde vive el análisis?

Es super importante reconocer el directorio de trabajo. Para ello es necesario utilizar la función *setwd()*.

Un tip para los que inician es ingresar a session/ set working directory / to source file location

# ¿Dónde vive el análisis?

## Mala organización



## ¿Resultados?



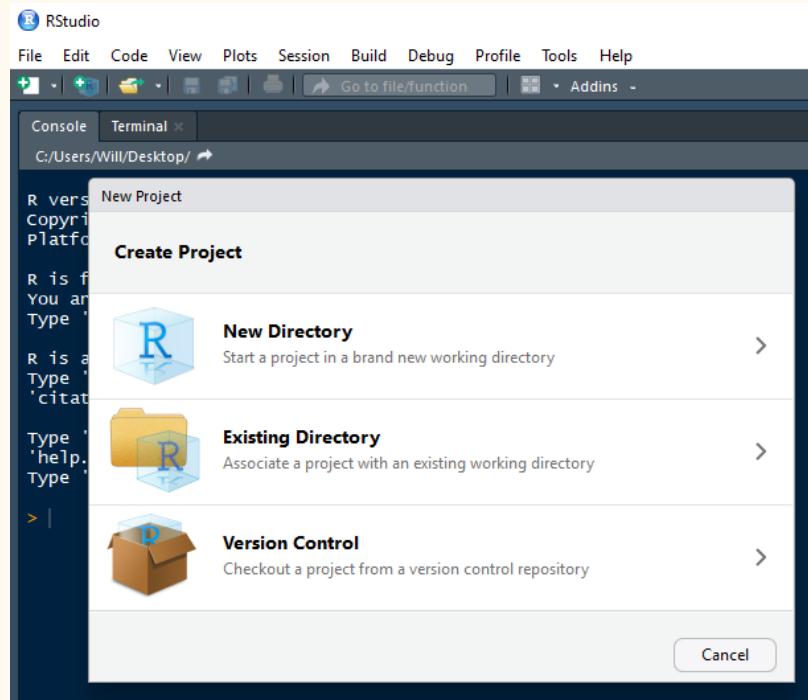
```
setwd("will/folder-that-only-exists-on-this-computer")
```

# ¿Qué es un R Project?

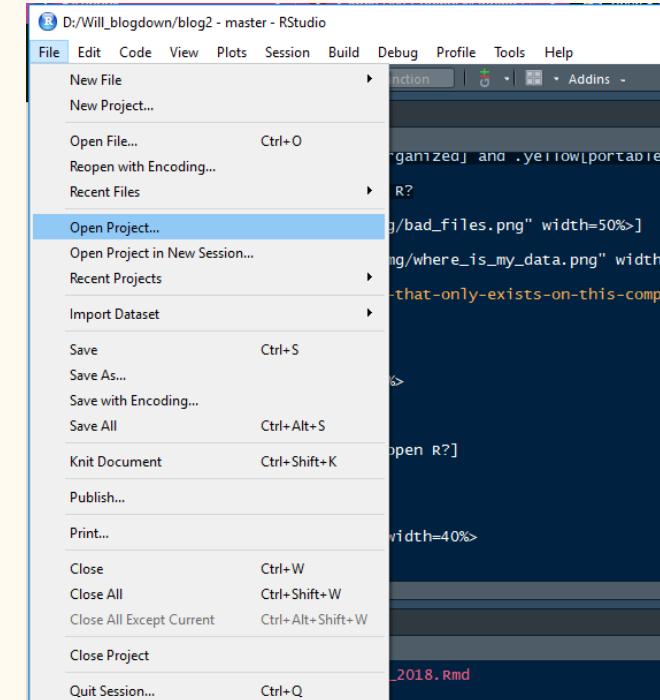
- ➡ Una manera de mantener todos tus archivos (scripts, gráficos, reportes) asociados con un proyecto de forma organizada en un sólo lugar.
- ➡ Permite diversas opciones para guardar tu R History o enviroment.
- ➡ Mantiene todos tus archivos centralizados
- ➡ Permite utilizar funciones de versión de control (por ejemplo Github) de forma más sencilla y útil.

# Veamos primero la terminología

## Creando un nuevo proyecto



## Abriendo un proyecto anterior



## Tip 2: Usa control de versión.

Es crítico para proyectos de mayor envergadura. *Esto lo podrán aprender en nuestro curso Gestión de Proyectos de Data Science.*



## Tip 2: Usa control de versión.

Así evitamos los clásicos, final, final final, final final esta es, final final esta es 2, etc.



## Tip 3: Redacta tus códigos de forma lecturable

‘

*Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.*

***John Woods***

***Spoiler: El psicópata eres tú*** 

# Hora de usar



# Paso 1: Instala los paquetes necesarios

Hay 2 tipos de instalaciones: CRAN y otros.

- ↗ para las instalaciones con el CRAN, `install.packages()`.
- ↗ para las instalaciones con github, `devtools::install_github()`. El paquete es devtools y la función `install_github()`.

Revisa el archivo instalador.R para instalar los paquetes que más vamos a utilizar. Ten en consideración que los paquetes del CRAN son paquetes oficiales y cuentan con una revisión exhaustiva mientras que Github y otros repositorios, cuentan con versiones de paquetes en desarrollo que pueden ser muy geniales!

## Paso 2: Carga los paquetes necesarios

Hay 2 formas de cargar.

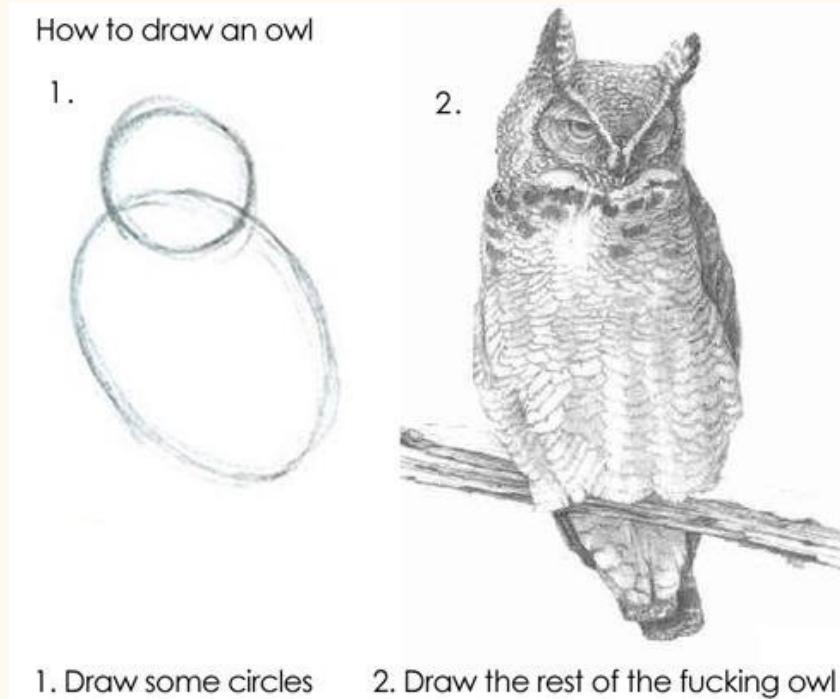
 *library()*: carga el paquete.

 *require()*: trata de cargar el paquete.

Recuerda no poner comillas en estas funciones y escribir correctamente, respetando mayúsculas y minúsculas.

# Paso 3: Programa

Sólo hay una forma.



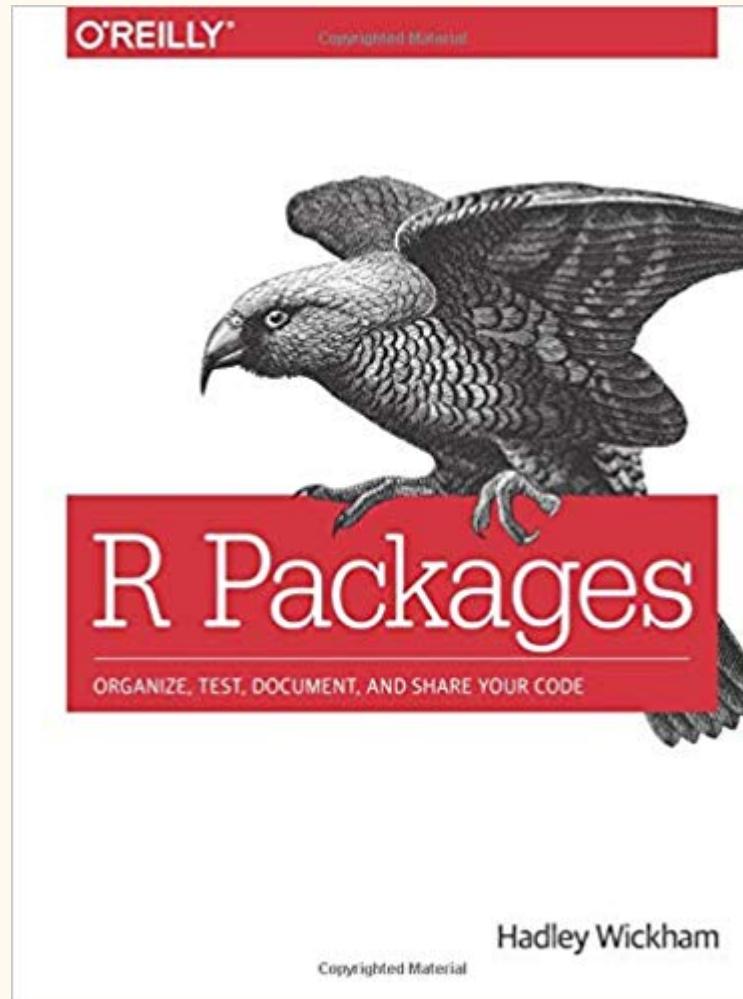
# Práctica 1: Lee un archivo de excel

1. Instala el paquete readxl
2. Descarga el archivo SBS.
3. Crea una carpeta en Mis Documentos y llámala practicas
4. Crea un R Project.
5. Crea un R Script usando la función *readxl::read\_xlsx()*
6. Utiliza el operador "<-" y ponle el nombre dataSBS.
7. Utiliza la función View() para ver la data.

# Práctica 1: Lee un archivo de excel

```
library(readxl) # carga el paquete  
setwd("~/practicas") # Configura tu propio directorio de trabajo  
dataSBS<-read_xlsx("B-2401-en2018.xls") # lees el archivo  
View(dataSBS) # veamos cómo está cargada la data!
```

# Conceptos: Paquetes



# Conceptos: Paquetes

‘

*Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data.*

***Hadley (2015). R packages***

# Conceptos: Paquetes

‘

*In R, the fundamental unit of shareable code is the package. A package bundles together code, data, documentation, and tests, and is easy to share with others. As of January 2015, there were over 6,000 packages available on the Comprehensive R Archive Network, or CRAN, the public clearing house for R packages. This huge variety of packages is one of the reasons that R is so successful: the chances are that someone has already solved a problem that you’re working on, and you can benefit from their work by downloading their package.*

**Hadley (2015). R packages**

# Conceptos: Paquetes

Hay paquetes para todo!

- ➡ **Temas académicos:** tesis, investigación reproducible, libros, papers, manejar bibliografías, etc.
- ➡ **Producción:** Desarrollo de aplicaciones, servidores, etc.
- ➡ **Ciencias:** Economía, Psicología, Biología, Medicina, Finanzas, etc.
- ➡ **Acceso a APIs:** Banco Mundial, FMI, Bloomberg, etc.

# Conceptos: Paquetes

‘

*But packages are useful even if you never share your code. As Hilary Parker says in her introduction to packages: “Seriously, it doesn’t have to be about sharing your code (although that is an added benefit!). It is about saving yourself time.” Organising code in a package makes your life easier because packages come with conventions.*

***Hadley (2015). R packages***

# Conceptos: Paquetes

Ella es Hillary Parker!



# Conceptos: Funciones

Las funciones son objetos básicos de R que ayudan a programar. Tiene 3 componentes:

\*`the body()`: El código dentro de la función.

\*`the formals()`: la lista de argumentos dentro de la función.

\*`the environment()`: El "mapa" de localización de la función de las variables.

Las 2 primeras son básicas para los que están conociendo R, en especial la segunda. En caso de funciones dentro de paquetes, es esencial leer la documentación. Para ello utiliza la función "?" después del nombre de la función. Ejm `?read.csv`. Usualmente no se usa todos los argumentos, sin embargo, es importante leerlos porque podría permitir cambios que podrían servirnos.

# Práctica 2: Lee un archivo de excel

Evita leer las primeras 5 líneas del excel de la práctica 1.

1. Instala el paquete `readxl`
2. Descarga el archivo SBS.
3. Crea una carpeta en Mis Documentos y llámala `practicas`
4. Crea un R Project.
5. Crea un R Script usando la función `readxl::read_xlsx()`
6. Evita (`skip`) leer las primera 5 filas. Usa `?read_xlsx`.
7. Utiliza el operador "`<-`" y ponle el nombre `dataSBS`.
8. Utiliza la función `View()` para ver la data.

## Práctica 2: Lee un archivo de excel

```
library(readxl) # carga el paquete
setwd("~/practicas") # Configura tu propio directorio de trabajo
dataSBS<-read_xlsx("B-2401-en2018.xls",skip = 5) # lees el archivo
View(dataSBS) # veamos cómo está cargada la data!
```

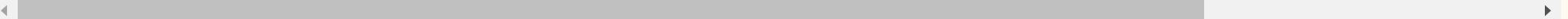
# Conceptos: Dataframes

En el ejemplo anterior dataSBS es una dataframe, un tipo de objeto especial de R, el cual está compuesto por diversos vectores.

```
class(dataSBS) # función para saber qué clase de objeto es  
dim(dataSBS) # función para saber las dimensiones del objeto  
head(dataSBS) # función para ver sus primeras observaciones (filas)
```

# Explorando data

```
library(datasets) # paquete con diversas bases de datos  
data(iris) # cargar una data de un paquete  
dim(iris) # para saber cuántas dimensiones tiene  
class(iris) # para saber la clase del objeto  
summary(iris) # para ver los percentiles de cada base de datos  
head(iris) # para ver las primeras observaciones  
sapply(iris,class) # para ver los tipos de vectores que contiene numéricos, f.  
# El uso de las funciones apply lo veremos más adelante!
```



# Conceptos: Objetos

Hay paquetes para todo!

- ➡ **Numeric:** Sirve para operaciones matemáticas.
- ➡ **Character:** Textos como nombres.
- ➡ **Factor:** Esto es para variables categóricas: Sexo, país, ciudad, etc.
- ➡ **Integer:** Número entero, sirve para ahorrar memoria si la data es pesada.
- ➡ **Dates:** Objeto para operar o graficar series de tiempo.
- ➡ **Muchos otros:** tibble, TS, XTS, data.table, etc. Se pueden crear nuevos objetos.

# Rmarkdown



# El inicio: knitr

El paquete que dio inicio a todo se llama knitr.



# El creador de knitr: Yihui Xie

Creador de diversos paquetes de R y uno de los más relevantes Data Scientist del mundo de R. Actualmente cuenta con un PhD y trabaja en RStudio.



# Rmarkdown vs Latex

¿Qué ventajas tiene markdown sobre Latex?

-  **Rápido de aprender:** En minutos lo aprendes.
-  **Variedad de outputs:** No sólo latex, word, ppt, html, markdown, etc.
-  **flexible a tu medida:** Si necesitas más detalles específicos, puedes usar CSS o incluso Latex.
-  **Combinar lenguajes de programación:** No sólo R, sino Python, Julia, C, etc.

Fuente: [Blog Yihui Xie en inglés](#)

# Estructura de un RMD

- ➡ **YAML:** Lenguaje de serialización. Sirve para meter datos como autor, fecha, opciones avanzadas, etc.
- ➡ **Títulos:** Usando michi para título 1, 2 michis para título 2, así sucesivamente.
- ➡ **Textos:** Tan fácil como tipear normal. En caso de negritas, poner 1, 2 o 3 subrayado o negritas (lo veremos en la práctica!).
- ➡ **Chunks:** Puedes correr código de R y otros lenguajes en este espacio, y puedes configurar de tal forma que sólo sea necesario.
- ➡ **Código en texto:** Aplica a tu paper o reporte un código de tal forma que te salga, por ejemplo, el coeficiente de regresión y no tengas que tipearlo.

# Práctica 3: Crea tu propio RMD

1. Ponle tu nombre.
2. Actualiza la fecha.
3. Pon títulos y texto cualquiera y córrelo!

# Práctica 4: Crea tu propio RMD más cool

1. Usa el formato del Rmd *reporte ejm.*
2. Elige una data de ejemplo de datasets.
3. Describe la data.
4. Usa el cheatsheet de Rstudio para que puedas modificar tu RMD a tu gusto.
5. Cuélgalo en rpub!

# El tidyverse



# ¿Qué es el tidyverse?

‘

*The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.*

[tidyverse.org](https://tidyverse.org)

# ¿Cuáles son los paquetes tidyverse?

Entremos a la página del tidyverse para descubrirlo! [tidyverse.org](https://tidyverse.org)

# Programación funcional

```
arrange(  
  summarize(  
    group_by(  
      filter(mtcars, carb > 1),  
      cyl  
    ),  
    Avg_mpg = mean(mpg)  
  ),  
  desc(Avg_mpg)  
)
```

# Programación basada en objetos

```
a <- filter(mtcars, carb > 1)
b <- group_by(a, cyl)
c <- summarise(b, Avg_mpg = mean(mpg))
d <- arrange(c, desc(Avg_mpg))
print(d)
```

# Programación con pipas

```
library(magrittr)
library(dplyr)

mtcars %>%
  filter(carb > 1) %>%
  group_by(cyl) %>%
  summarise(Avg_mpg = mean(mpg)) %>%
  arrange(desc(Avg_mpg))
```

# Una introducción a Dplyr

- ➡ **filtrar:** `filter()` Utiliza == , <, >, >=, <=, %in% c("elemento1","elemento2").
- ➡ **agrupar por:** `group_by()` Crea grupo para tener resúmenes de datos luego con `summarise`.
- ➡ **crea nuevas variables resumidas:** `summarise()` Crea resúmenes de un grupo. Ejm promedio de notas por salón de clase.
- ➡ **creación de variables sin resumir:** `mutate()` Crea una variable de cualquier tipo.
- ➡ **Ordenar datos:** `arrange()` y `arrange(desc())` Ordena una base de datos de forma ascendente o descendente.
- ➡ Hay muchas más funciones usa `?dplyr`

# Práctica 5: Crea tu propio RMD más cool

1. Usa el formato del Rmd *reporte ejm.*
2. Elige una data de ejemplo de datasets.
3. Describe la data y usa el tidyverse.
4. Usa el cheatsheet de Rstudio para que puedas modificar tu RMD a tu gusto.
5. Cuélgalo en rpub!

# Objetivos: Revisión

- ➡ Conocer el término Data Science, en qué se diferencia de estadística y qué habilidades requiere.
- ➡ Conocer el rol de R en el mundo de la ciencia de datos.
- ➡ Distinguir entre R y RStudio.
- ➡ Entender funciones básicas de R.
- ➡ Saber cómo crear tu primer producto con R: un reporte en HTML.
- ➡ Conocer un poco acerca del universo tidyverse.
- ➡ Contar con un nivel básico de dplyr.

# Fundamentos de Data Science con R

