Métodos Cuantitativos en R

Enero – Febrero 2022

Introducción a R y manejo de datos
Temas generales

Sesión I

¿Qué es R?

R es un conjunto integrado de programas para manipulación de datos, cálculo y gráficos. Entre otras características dispone de:

• almacenamiento y manipulación efectiva de datos,

• una amplia, coherente e integrada colección de herramientas para análisis de datos,

• posibilidades graficas para análisis de datos, y

• un lenguaje de programación bien desarrollado, simple y efectivo, que incluye condicionales, ciclos, funciones recursivas y posibilidad de entradas y salidas.

R es un lenguaje Orientado a Objetos: bajo este complejo término se esconde la simplicidad y flexibilidad de R. Orientado a Objetos significa que las variables, datos, funciones, resultados, etc., se guardan en la memoria activa del computador en forma de objetos con un nombre específico. El usuario puede modificar o manipular estos objetos con operadores (aritméticos, lógicos, y comparativos) y funciones (que a su vez son objetos).

RStudio

RStudio es un entorno de desarrollo integrado (IDE) para *R*. Es software libre con licencia GPLv3 y se puede ejecutar sobre distintas plataformas (Windows, Mac, o Linux).

Entre otras cosas encontramos que RStudio :

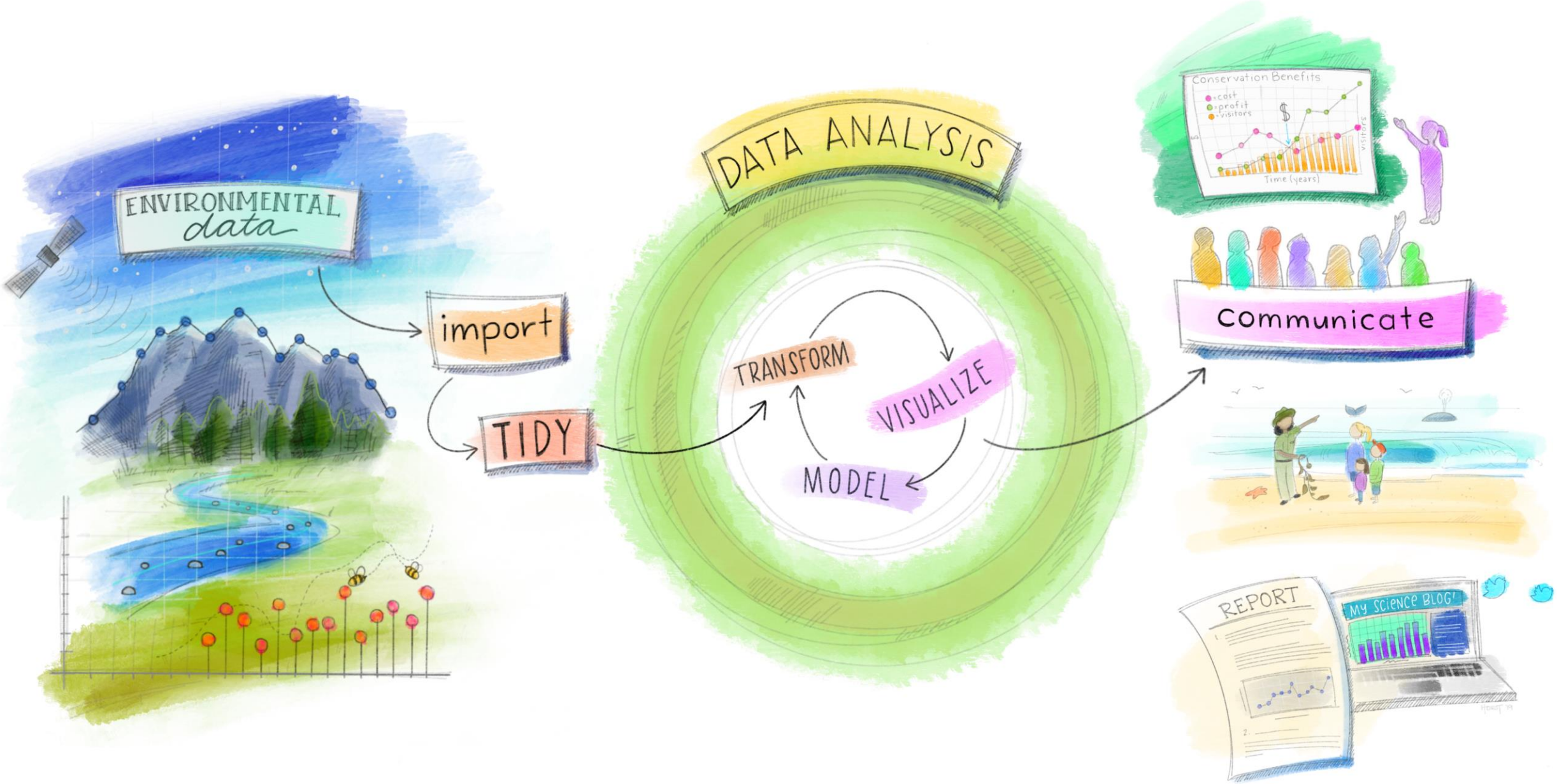
1. Nos permite abrir varios scripts a la vez

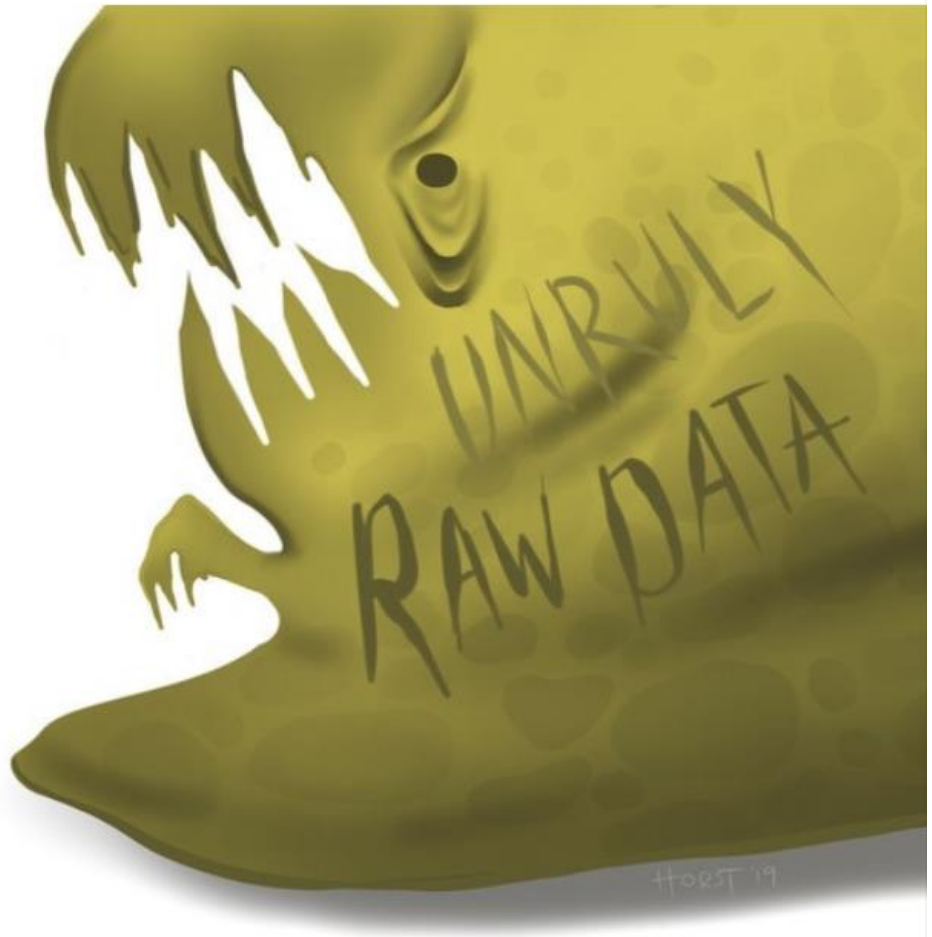
2. Nos muestra el workspace

3. Nos muestra el historial

4. Nos muestra los objetos del workspace

5. Integra la gestión de librerías

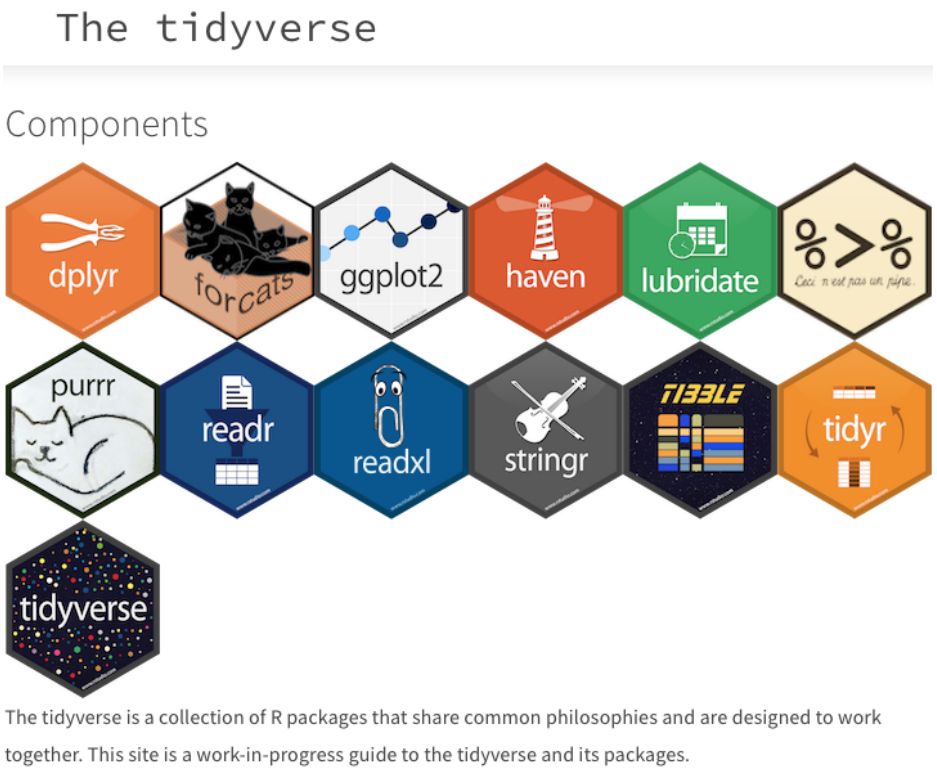

ENVIRONMENTAL data
import
TIDY
DATA ANALYSIS
TRANSFORM
VISUALIZE
MODEL
Conservation Benefits
cost
profit
visitors
Time (years)
Communicate
REPORT
My Science BLOG!



@allison_horst

Tidyverse

El tidyverse es un conjunto de paquetes de R diseñado para su uso en ciencia de datos. Todos estos paquetes comparten un diseño y una filosofía, gramática y estructura de datos común. Estos paquetes son:



Tidyverse

El tidyverse es un conjunto de paquetes de R diseñado para su uso en ciencia de datos.

Todos estos paquetes comparten un diseño y una filosofía, gramática y estructura de datos común. Estos paquetes son:

•**dplyr**, para el manejo de bases de datos.: Incluye:

filter(), select(), mutate(), group_by(), summarise() y arrange()

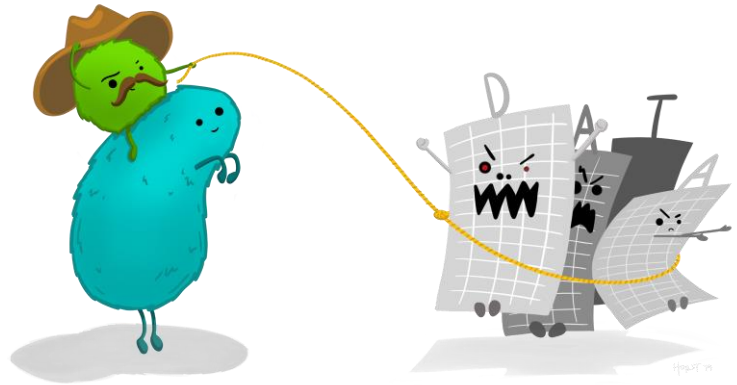
•**ggplot2**, para la elaboración de gráficas a partir de la *gramática de las gráficas*.

•**readr,** para leer archivos tabulares de manera sencilla.

•**tibble,** formato mejorado para manejar data.frames y tablas.

Operador pipa (%>%)

La pipa es un elemento central en tidyverse y su misión es la de concatenar funciones. Opera vinculando los objetos del lado izquierdo con las funciones del lado derecho.



Para escribirlo se utiliza:

MAC: command + shift + m

Windows: control + shif + m

De manera coloquial, el operador pipa se debe leer como un "y después".

Verbos del tidyverse

Las principales funciones del tidyverse para tratar con datos son los siguientes:

filter(), para filtrar renglones/observaciones

select(), para seleccionar columnas

mutate(), para generar nuevas columnas

arrange(), para ordenar los renglones en función de una variable

group_by(), para generar grupos o clusters de renglones

Instalar librerías/paquetes

Al instalar R y RStudio, tenemos precargadas las librerías:

base, datasets, graphics, grDevices, methods, stats y utils.

En R, utilizamos librerías externas para añadir más funcionalidades a nuestros códigos:

Si queremos instalar una libreria nueva, tenemos las siguientes opciones:

a) **Descargarlas con el instalador de paquetes de RStudio**

b) **Utilizar la función install.packages()** si el paquete esta en el CRAN

c) **Usar la librería** devtools **y la función** devtools::install_github() para descargar paquetes que no estan en el CRAN desde Github u otra ubicación externa

Activar o llamar librerias

Para llamar librerias, utilizaremos la función library(), la cual nos sirve para cargar paquetes de R previamente instalados en nuestra computadora.

Nota: Alternativa usando la librería pacman

pacman es un paquete que nos permite llamar otros paquetes utilizando la función pacman::p_load(), la cual **llama a los paquetes de la misma manera que library() y require() si estan instalados, y los instala si no lo estan**. La función recibe como argumentos la lista de los nombres de las librerias, pudiendo cargar multiples librerias en una sola línea de código.

Repaso de operadores $ y <-

El operador $ nos sirve para acceder a los componentes de un objeto, mientras que el operador <- nos permite asignarle un nombre a un objeto y guardarlo en memoria dentro del ambiente. Usados conjutamente, el operador $ y <- nos permiten crear nuevos objetos y variables.

Subseteo con [ ]

```
# Guardamos en otro objeto
bd <- palmerpenguins::penguins

# Nos quedamos con el primer renglon
bd[1,]

# Nos quedamos con la primer columna
bd[,1]

# Nos quedamos con la columna species
bd[,"species"]

# Nos quedamos con los renglones 1,2,3 y 64
bd[c(1,2,3,64),]

# Nos quedamos con los renglones de los pinguinos
# de la isla Torgersen
bd[bd$island == "Torgersen",]
```

Archivos nativos de R

Al trabajar con R y RStudio, generamos cuatro tipos de archivos nativos de R, estos son los siguientes:

*.Rproj,

*.RData,

*.R y

*.rds.

*.Rproj, proyectos de R

Los archivos de proyectos de R mantienen nuestro trabajo dividido en múltiples contextos, teniendo asociados un **único directorio de trabajo**, **espacio de trabajo**, **historial** y **documento fuente**.

Podemos crear Proyectos en nuevos directorios desde RStudio (*File > New Project...*).

*.RData, datos de R

Los archivos *.RData nos permiten almacenar múltiples objetos dentro de un sólo archivo. Estos archivos son específicos de R, y no son compatibles con ningún otro lenguaje de programación.

Para desempaquetar los objetos de un archivo *.RData, utilizamos la función load(). Otra opción es que arrastremos el archivo hasta nuestra ventana de RStudio y el programa entenderá que queremos desempaquetarlo.

```
# Abriendo los archivos
load(file = "study1.RData")
```

```
save(c1.df, c2.df, c1.htest,
file = "study1.RData")
```

c1.df

p	x	y
1	a	25
2	b	12
3	c	31

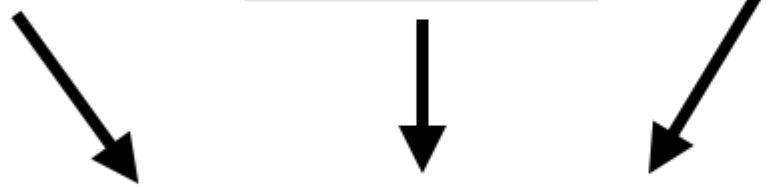
c2.df

p	x	y
1	x	23
2	y	8
3	z	9

c1.htest

```
        Welch Two Sample t-test

data:  score by sex
t = 1.8385, df = 2.9431, p-value = 0.1651
alternative hypothesis: true difference in
95 percent confidence interval:
 -16.37630  60.04297
sample estimates:
mean in group f mean in group m
       59.50000        37.66667
```



study1.RData

*.R, scripts de R

Scripts escritos en R. Estos archivos contienen los comandos o instrucciones necesarios para que el lenguaje de programación R lleve a cabo acciones específicas.

*.rds, almacenamiento de objetos en R

Los archivos *.rds son **archivos que almacenan objetos en un formato compatible sólo con R**.

Estos archivos tienen la ventaja de que se leen más rápido que una base de datos en cualquier formato, y pesan mucho menos.

Ejemplo; testimonio de uso.

"Analizando la base de datos de la Encuesta social europea, esta pesaba 103.5 MB en su versión comprimida. Su versión .DTA (STATA) pesaba aproximadamente 3.16 GB, mientras que la versión .CSV pesaba 59.7 MB. En el caso del .RDS, este archivo pesaba 51.6 MB, ligeramente menor que el archivo .CSV"

Buenas practicas de código

Para escribir en R hay múltiples maneras y estilos.

Escribir código de manera ordenada y documentada tiene las siguientes ventajas:

Hace el código **más sencillo de leer**, y de comprender, tanto como para otras personas como para nuestro *yo* del futuro.

Permite un sencillo mantenimiento, al identificar las partes que lo conforman.

Hace que la detección de errores sea más sencilla.

Algunas de las buenas prácticas de código que yo recomiendo son las siguientes:

De preferencia, siempre usar los *Proyectos de R* (archivos *.Rproj).

Esto nos brinda las siguientes ventajas:

a). Sabemos perfectamente donde está el directorio de trabajo.

b). Nos carga los archivos que necesitamos cargar para un trabajo en especial (en vez de abrirnos los archivos más recientes), lo cual...

c). Nos permite tener espacios de trabajo independientes. De esta manera, el script de un trabajo no se abre cuando abro un proyecto distinto.

Tener una adecuada estructura de carpetas.

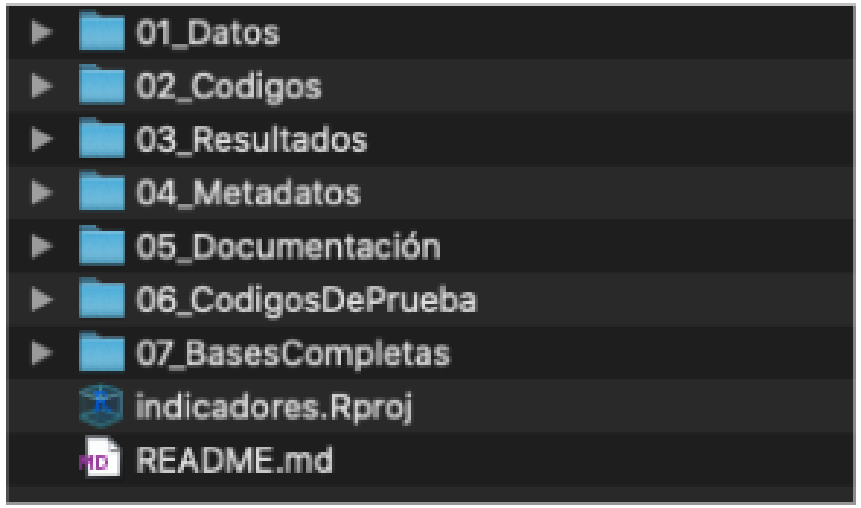
En el **CIDE**, en clases, se utiliza un sistema de carpetas para todos los trabajos que les solicitan a los alumnos, y los cuales son los siguientes:

- 01_Datos
- 02_Codigo
- 03_Graficas

Y por fuera de estas carpetas, el archivo *.Rproj.

Este ordenamiento nos permite **tener bien diferenciados los insumos, los procesos y las salidas de información.**

Esto es una sugerencia. Cada usuario decidirá, de acuerdo a las necesidades de su proyecto, las carpetas necesarias. Por ejemplo, en este caso:



Se montaron 7 carpetas distintas, cada una almacenando un tipo de archivo distinto con funciones distintas.

Comentar el código

Documentar el código es **añadir suficiente información como para explicar lo que hace, punto por punto, de forma que no sólo los ordenadores sepan qué hacer,** sino que además los humanos entiendan qué están haciendo y por qué.

Podemos comentar (y documentar) el código de distintas maneras, las cuales pueden ser complementarias:

- A través de **comentarios** explicando las particularidades de **cada línea o sección de código.**
- A través de **tutoriales** (escritos, en video) donde se explique de manera detallada los procedimientos que realiza tu código.