INTRODUCCIÓN A R

Christian Camilo Urcuqui López, MSc

GitHub: urcuqui
https://github.com/urcuqui

UNIVERSIDAD ICESI | A OTRO NIVEL

PRESENTACIÓN

Christian Camilo Urcuqui López

Ing. Sistemas, Magister en Informática y Telecomunicaciones

Big Data Professional

Big Data Scientist

Deep Learning Specialization

Grupo de investigación i2t

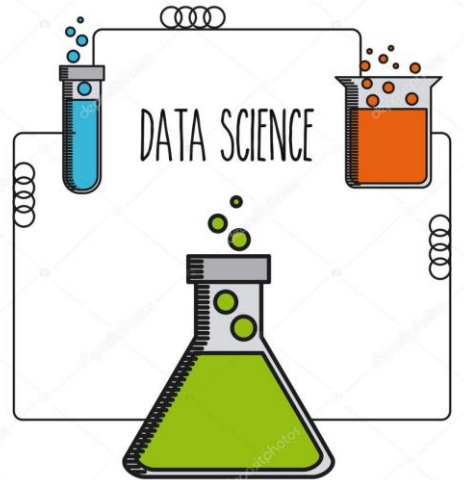
Líder de investigación y desarrollo

Ciberseguridad y ciencia de datos aplicada

ccurcuqui@icesi.edu.co

COMPETENCIAS

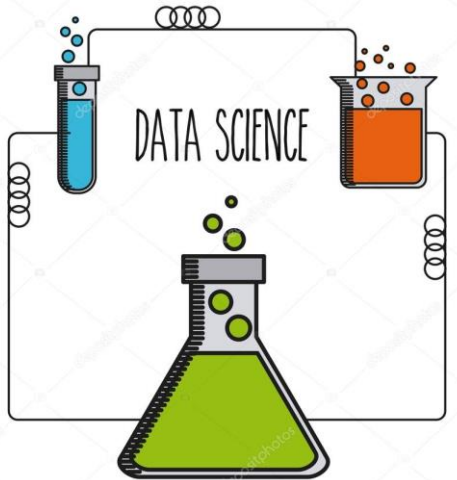
- Describir el lenguaje de programación R y su aplicación en proyectos de ciencia de datos.
- Utilizar el entorno de trabajo RStudio
- Aplicar los conceptos básicos de codificación en R.
- Explorar un *data.frame* en R



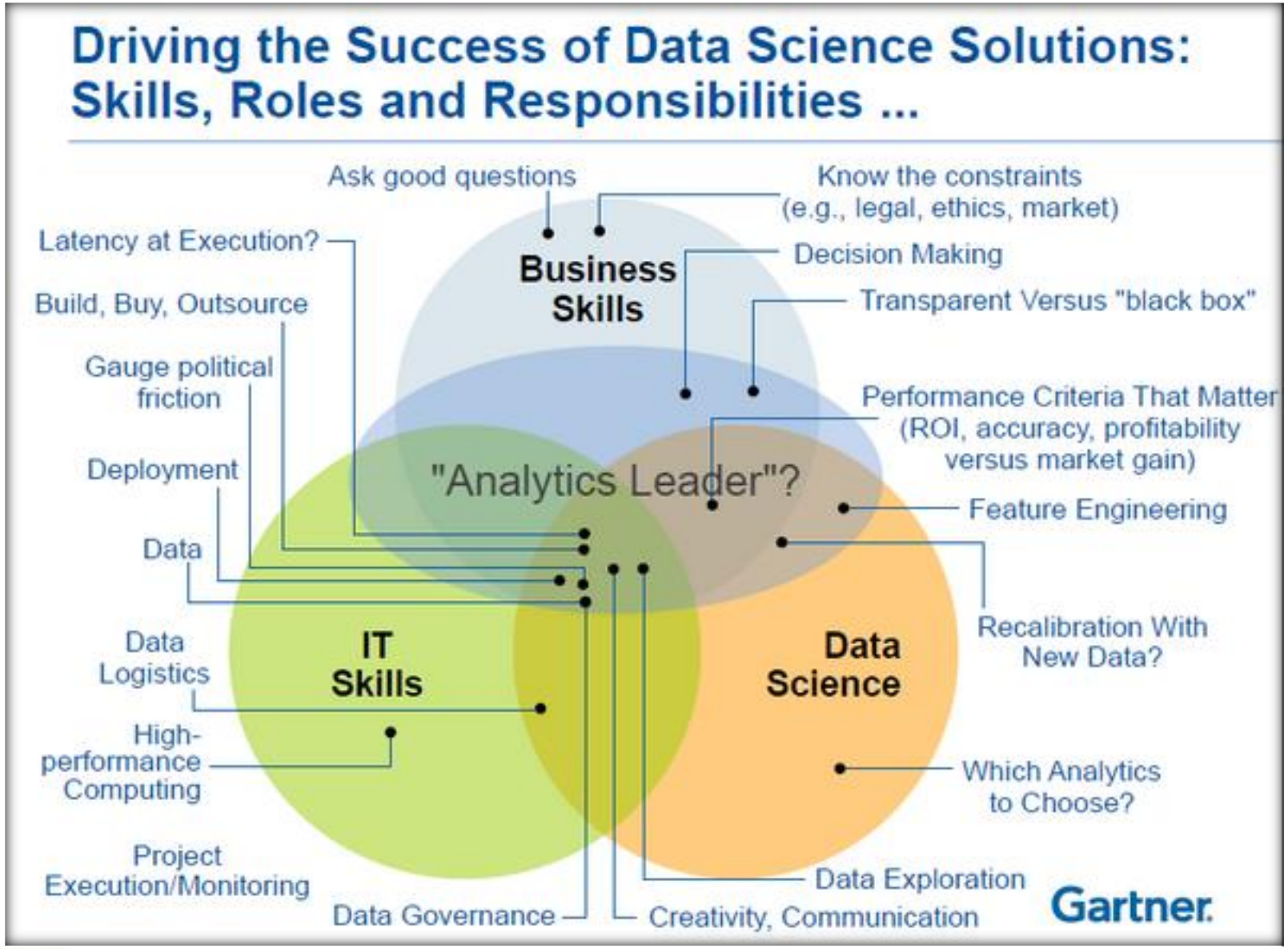
UNIVERSIDAD ICESI | A OTRO NIVEL

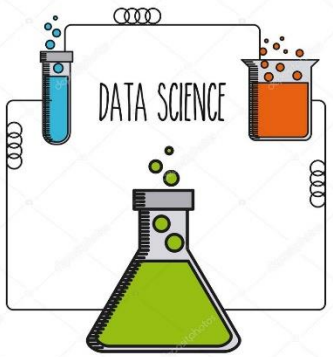
CONTENIDO

1. Introducción al lenguaje de programación R
2. Flujo de trabajo en R - Nombres de objetos, llamada a funciones
3. Explorando las estructuras de datos en R
4. Importando y trabajando con datos

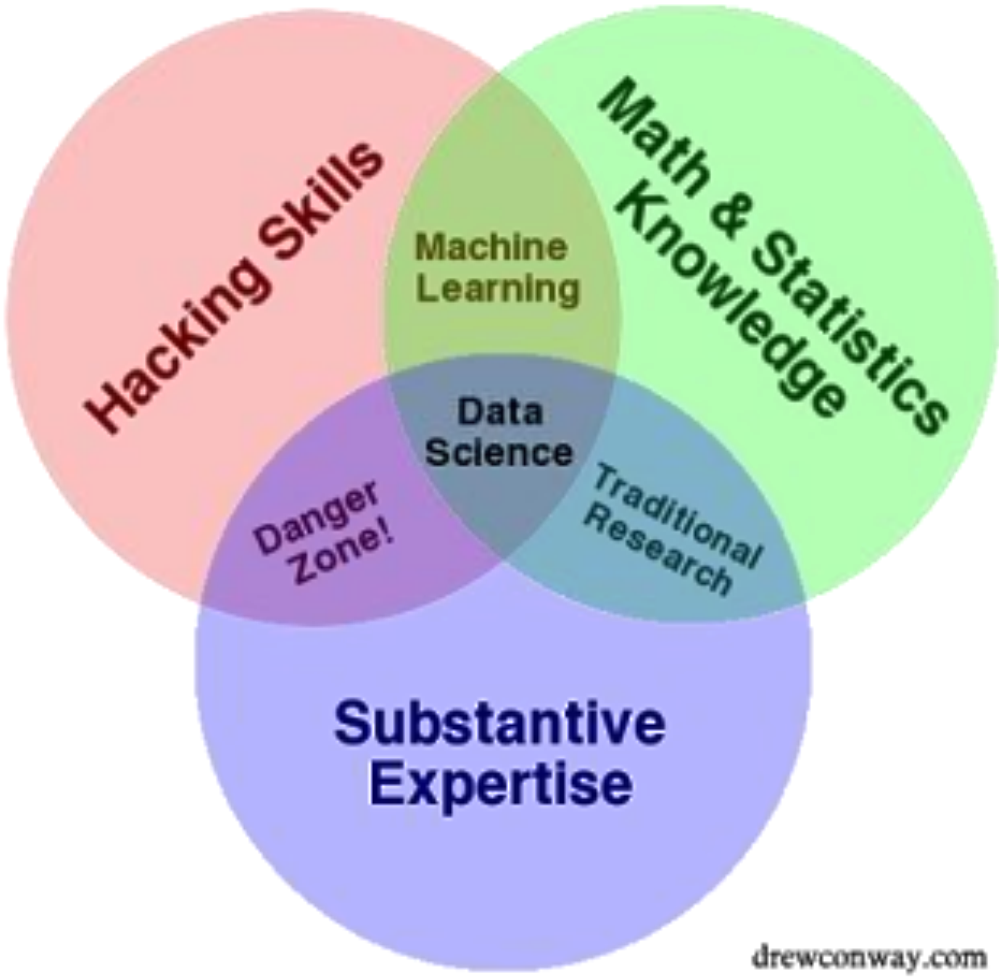


UNIVERSIDAD ICESI | A OTRO NIVEL

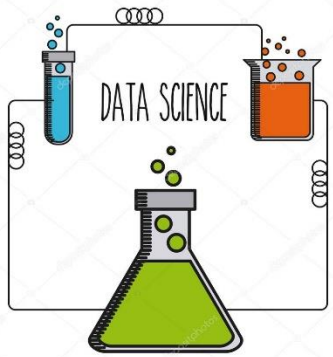




https://www.datasciencecentral.com/profiles/blogs/difference-between-machine-learning-data-science-ai-deep-learning



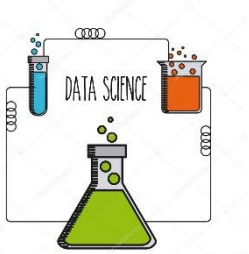
Drew Conway in 2010



UNIVERSIDAD ICESI | A OTRO NIVEL

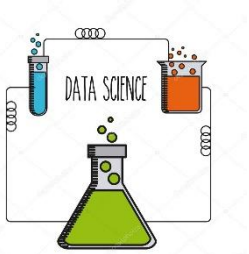
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- La forma de analizar los datos ha estado cambiando durante los últimos años.
- "Los datos son el petróleo del siglo xxi".
- La ciencia de datos a partir de sus técnicas (por ejemplo, estadísticas, visuales, econométricas y de aprendizaje de máquina) han permitido descubrir y explotar la información.

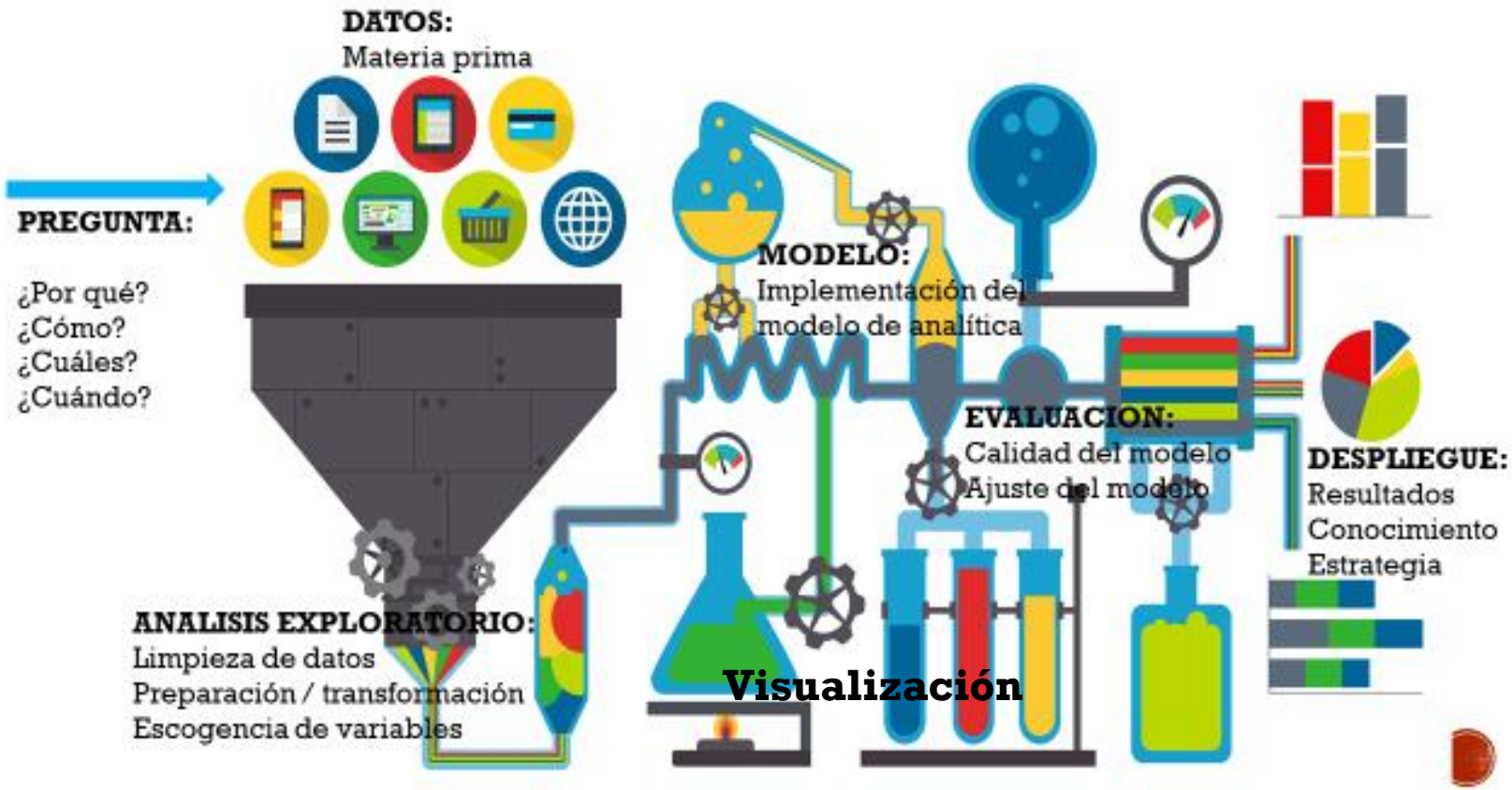


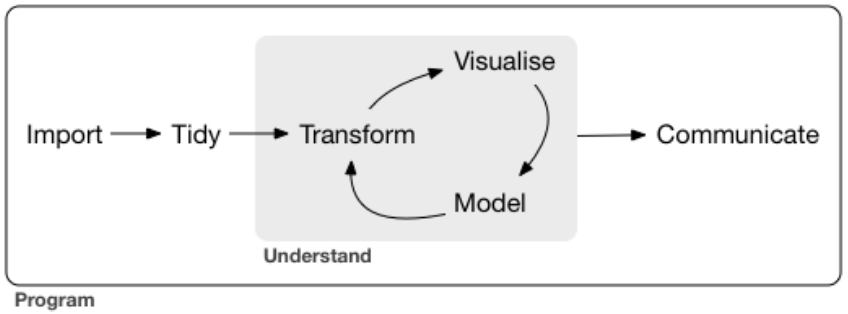
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Antiguamente, los investigadores solían publicar sus resultados en prestigiosas revistas y la implementación de sus descubrimientos en software tomaba mucho tiempo.
- Actualmente, los investigadores y la industria han mejorado sus métodos en conjunto con la implementación de software, estos resultados ahora se encuentran en sitios web de fácil acceso (en muchos casos con licencia ***open source***).

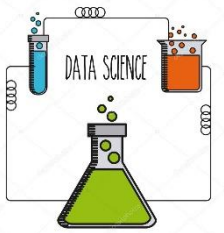


CICLO DE TRABAJO

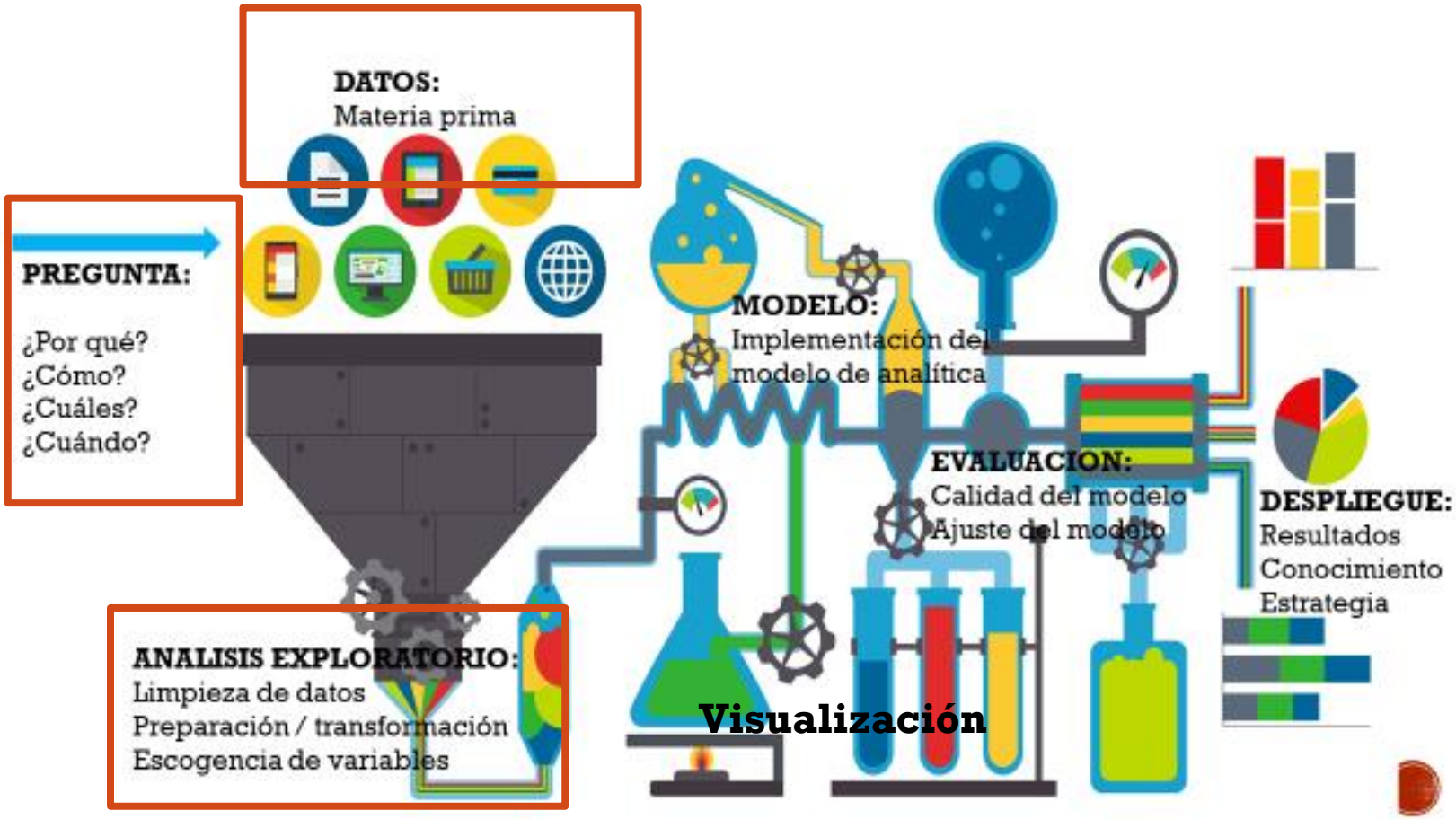


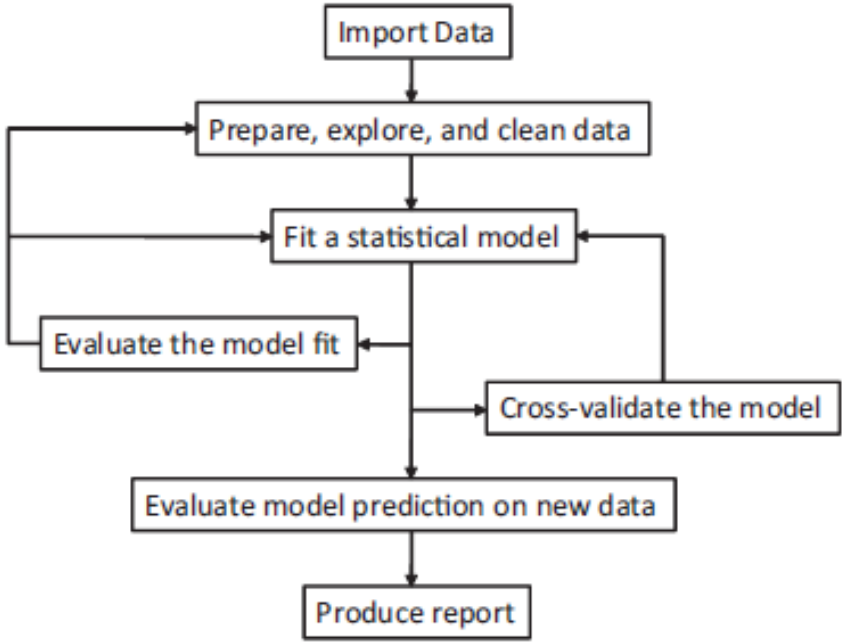


Marco de trabajo típico de un proyecto de ciencia de datos.
R for Data Science



CICLO DE TRABAJO



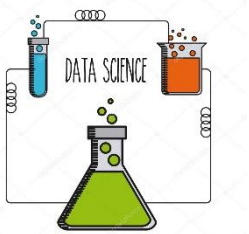


Pasos típicos de un proyecto de ciencia de datos.

R IN ACTION: Data analysis and graphics with R

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

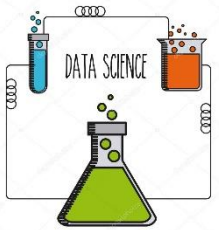
- R es un entorno y un lenguaje para computación estadística y gráfica desarrollado en Bell Labs, un proyecto que nace del software libre S.
- Una solución *open source* para análisis de datos soportado por varias comunidades científicas en todo el mundo.
- ¿Por qué utilizar R en vez de otras soluciones populares para estadística y gráficos (por ejemplo, Microsoft Excel, SAS, IBM SPSS, Stata, y Minitab)?



INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

Excel

- **Cálculo básicos.** Excel provee una interfaz (entorno) más amigable para cálculos simples (por ejemplo, estadísticas descriptivas) o algunas manipulaciones sencillas (por ejemplo, filtros y búsquedas).
- **Ver los datos continuamente**. Excel es una herramienta que nos permite constantemente ver la estructura y el contenido de los datos.
- **Presentación de datos y resumen**. Excel nos da un contenido estético más agradable de las hojas de cálculo.
- **Menor curva de aprendizaje**. Requiere un menos tiempo para llegar a manejar gran parte de sus funcionalidades. n

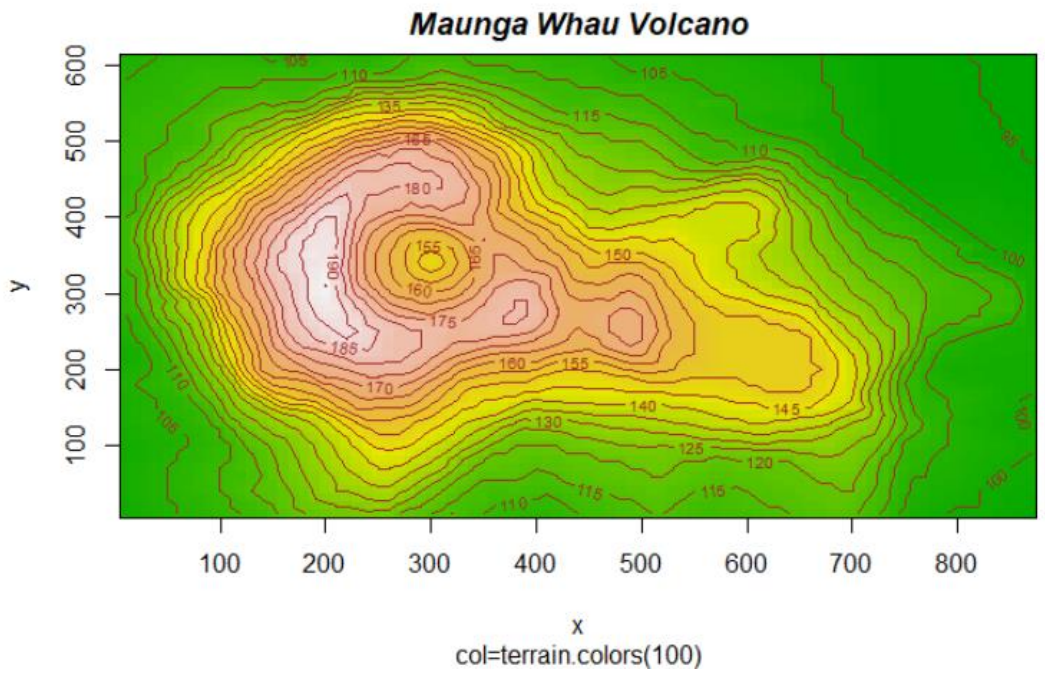


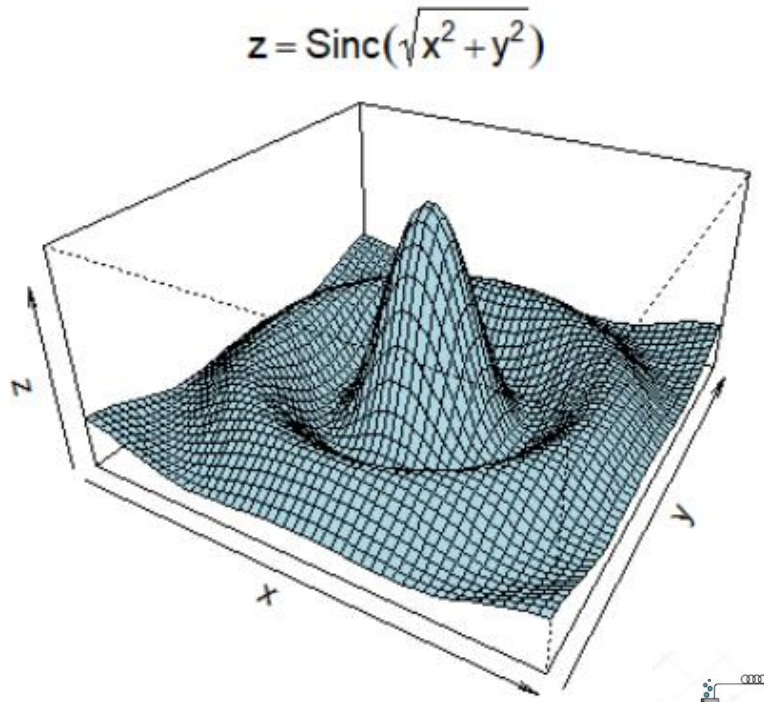
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

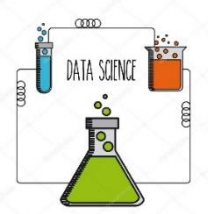
R

- Es una tecnología gratuita a diferencia de otros software comerciales de altos costos.
- Tiene comunidades muy activas, casi semanalmente se proponen nuevos paquetes estadísticos y actualizaciones, lo cual a llevado a los sistemas comerciales a integrar R.
- Es más fácil la automatización ya que se pueden desarrollar *scripts (líneas de código con un propósito especifico)* que permiten ejecutar el análisis varias veces.
- Leer casi cualquier tipo de datos (.txt, .csv, .dat), también, existen paquetes que permiten leer información de archivos JSON, Excel, STATA, SAS. E incluso utilizar datos de sitios web y de sistemas de base de datos (Por ejemplo, MySQL y, PostgresSQL)

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R



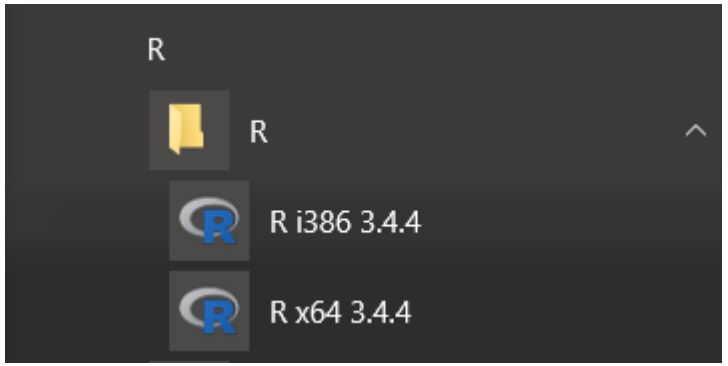


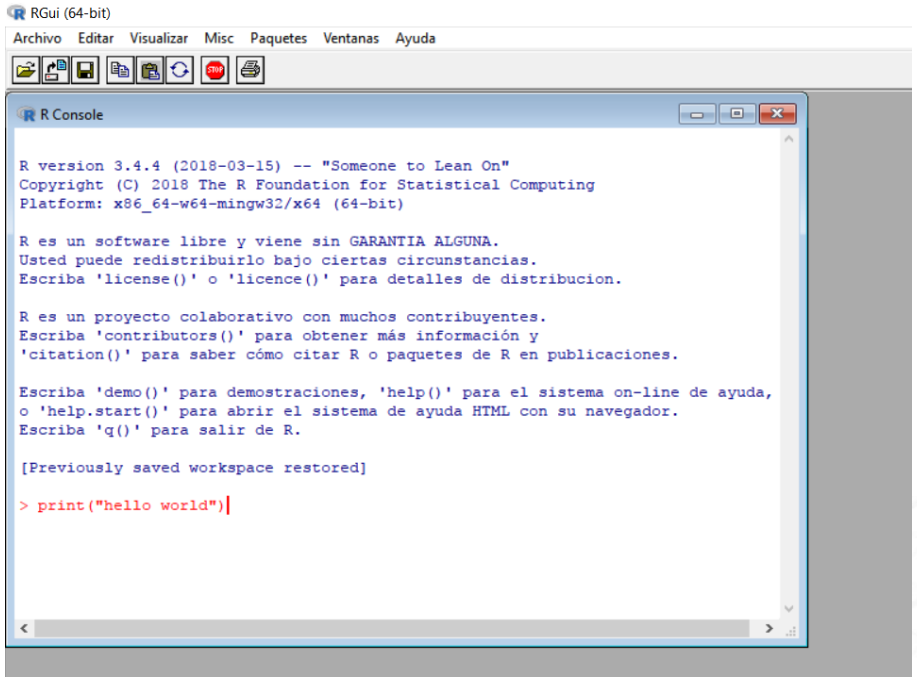


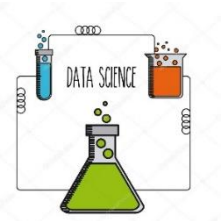
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

Nuestro primer hello world

- Una vez instalado R en nuestro equipo, nosotros podemos acceder a la consola de R para que podamos escribir los códigos. Procedamos ha abrir una consola y digitemos el comando **print("hello world")**







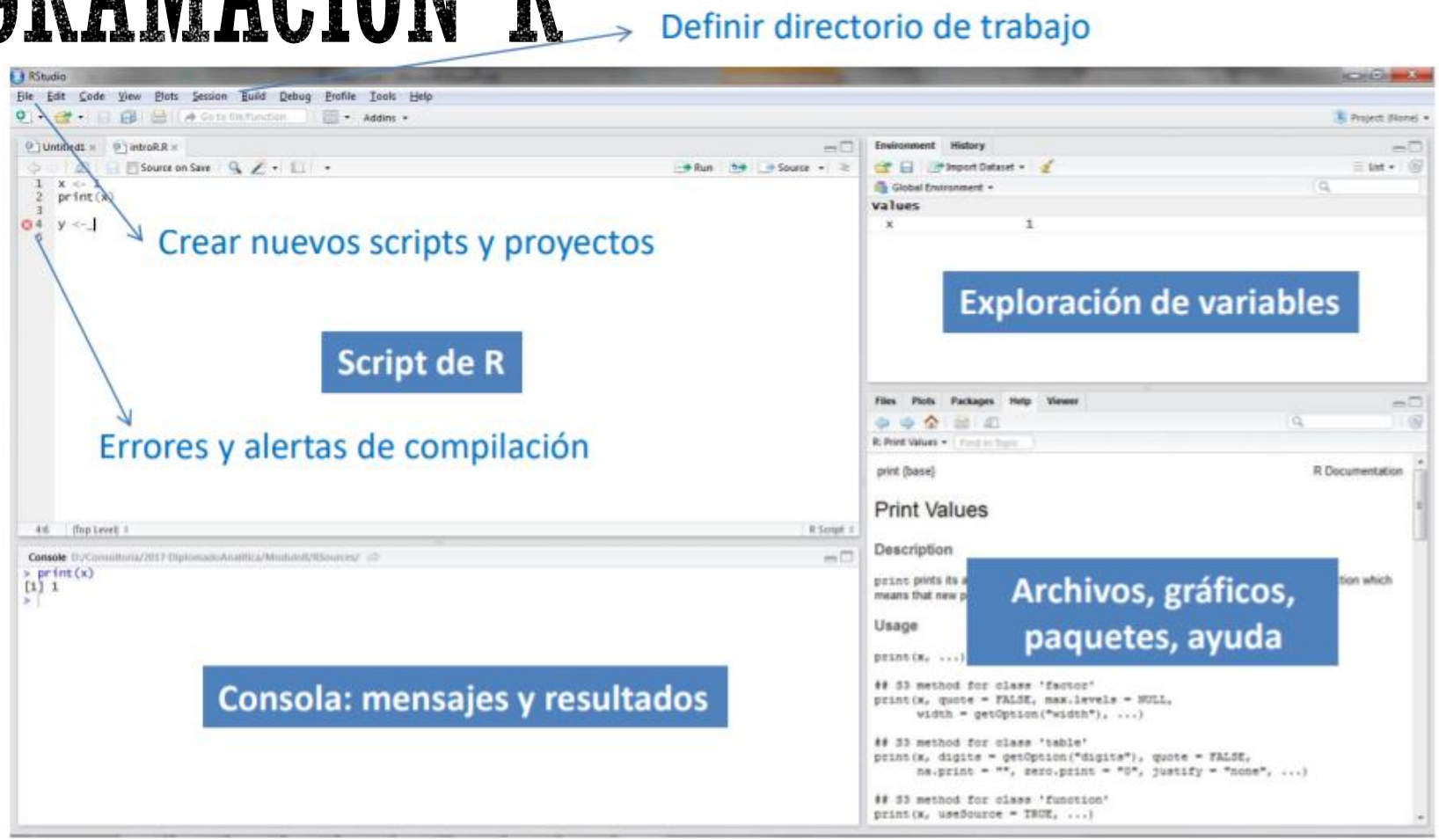
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

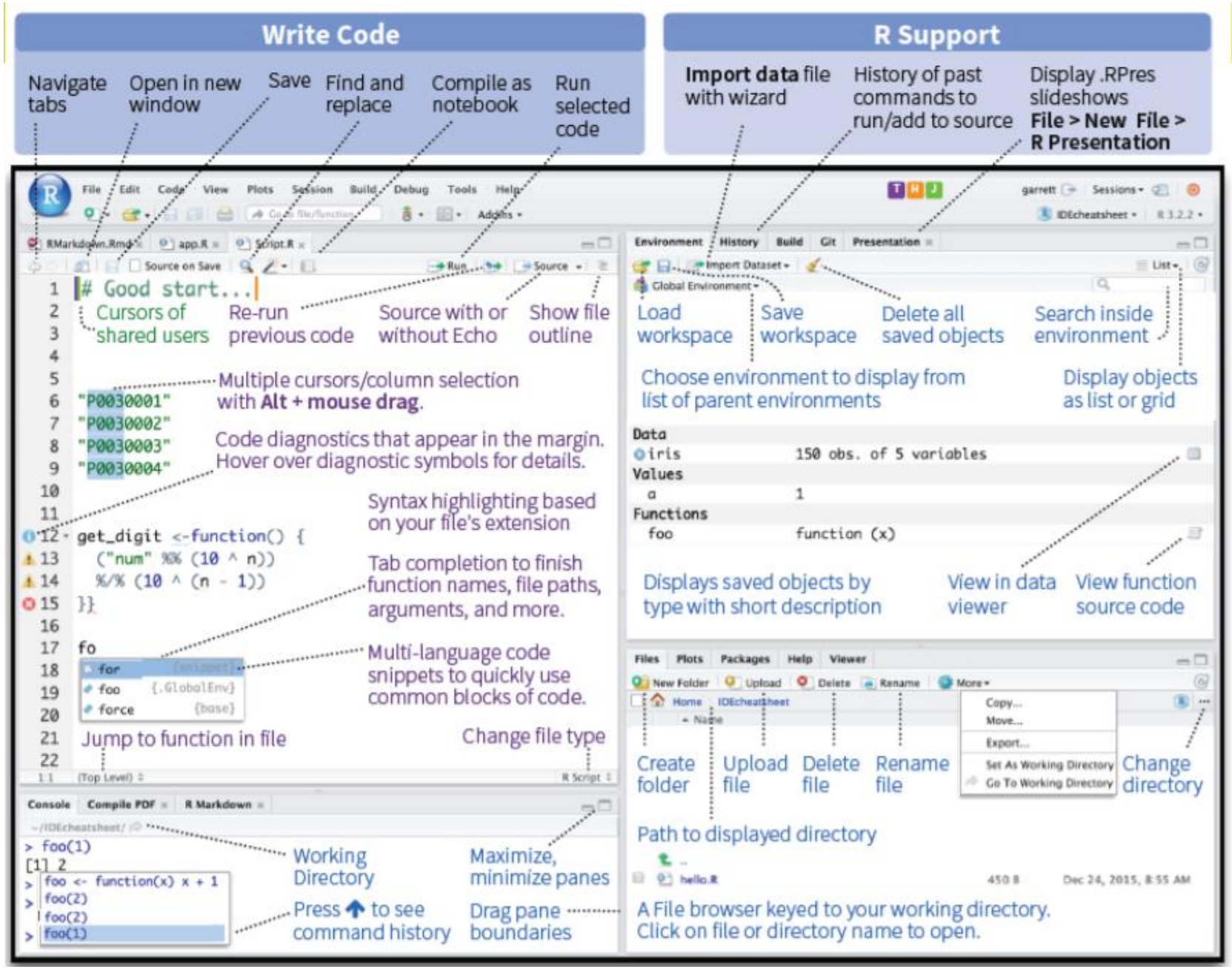
- Existen ambientes de programación más amigables y con una serie de utilidades que nos facilitan el desarrollo de proyectos de software, para nuestro caso utilizaremos **RStudio**.
- RStudio puede descargarse de la siguiente página web:

https://www.rstudio.com/products/rstudio/download/

- RStudio nos facilita el trabajo con R
  - Editor de código
  - Depurador (permite probar y depurar errores en tiempo de ejecución)
  - Herramientas de visualización
- Algunas de sus versiones son de uso libre, otras son licenciadas.

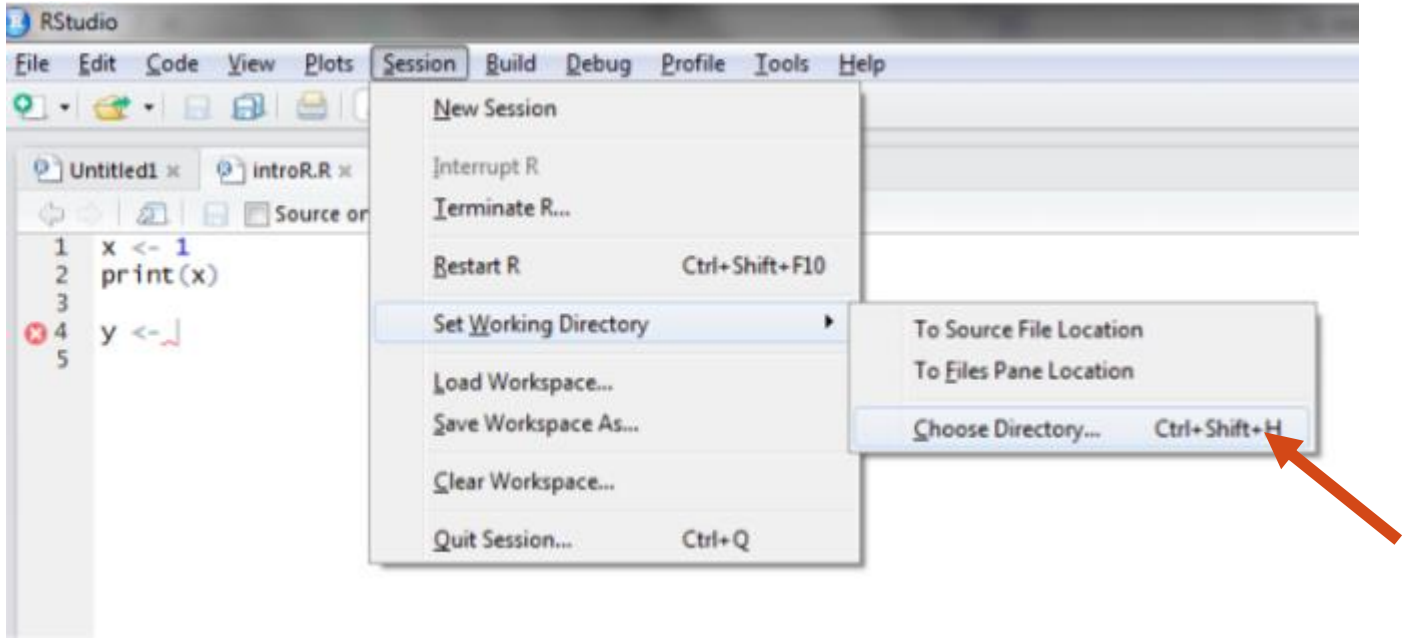
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R




Write Code
Navigate tabs
Open in new window
Save
Find and replace
Compile as notebook
Run selected code
R Support
Import data file with wizard
History of past commands to run/add to source
Display .RPres slideshows
File > New File > R Presentation
Cursors of shared users
Re-run previous code
Source with or without Echo
Show file outline
Multiple cursors/column selection with Alt + mouse drag.
Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
Syntax highlighting based on your file's extension
Tab completion to finish function names, file paths, arguments, and more.
Multi-language code snippets to quickly use common blocks of code.
Jump to function in file
Change file type
Load workspace
Save workspace
Delete all saved objects
Search inside environment
Choose environment to display from list of parent environments
Display objects as list or grid
Displays saved objects by type with short description
View in data viewer
View function source code
Create folder
Upload file
Delete file
Rename file
Change directory
Path to displayed directory
A File browser keyed to your working directory. Click on file or directory name to open.
Working Directory
Maximize, minimize panes
Press ↑ to see command history
Drag pane boundaries

ESTABLECER UN DIRECTORIO DE TRABAJO

- Cree un directorio en el disco local en el que quiera almacenar los scripts y datasets que usaremos hoy.
- En RStudio realice la siguiente acción para definir el directorio de trabajo



DIRECTORIO DE TRABAJO Y WORKSPACE

- Directorio de trabajo: Es la carpeta donde RStudio buscará los archivos de trabajo para leerlos y sobrescribirlos.
- Workspace: es el ambiente de ejecución actual de R. En ese espacio se almacena en memoria todos los objetos y paquetes definidos/cargados por el usuario.
- **Se recomienda salvaguardar la sesión** para futuras actividades sobre los mismos objetos, ya que una vez finalizada la sesión, si el workspace no se guarda, la memoria con su contenido se vaciara.

DIRECTORIO DE TRABAJO Y WORKSPACE

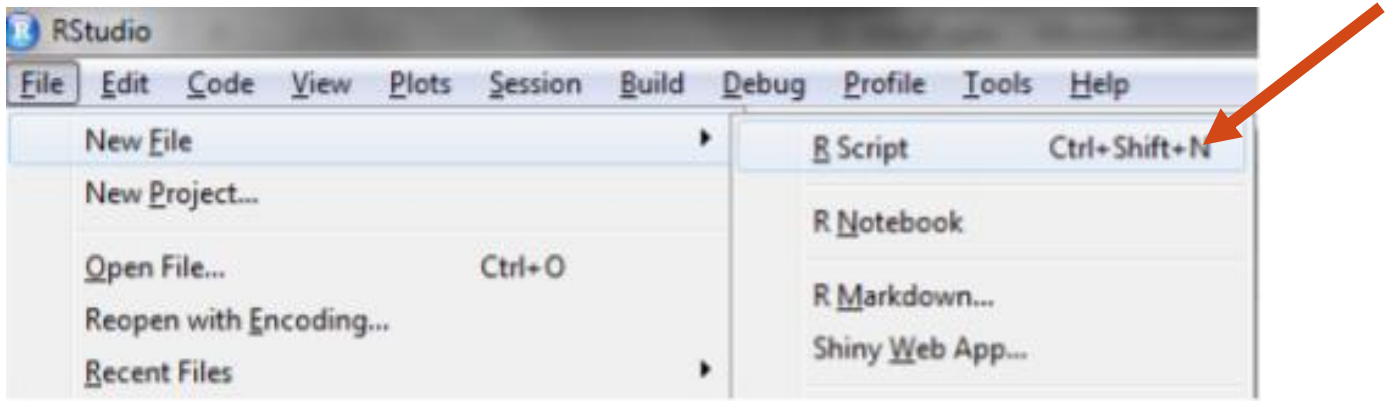
Table 1.3 Functions for managing the R workspace

Function	Action
`getwd()`	List the current working directory.
`setwd("mydirectory")`	Change the current working directory to *mydirectory*.
`ls()`	List the objects in the current workspace.
`rm(objectlist)`	Remove (delete) one or more objects.
`help(options)`	Learn about available options.
`options()`	View or set current options.
`history(#)`	Display your last # commands (default = 25).
`savehistory("myfile")`	Save the commands history to *myfile* (default = `.Rhistory`).
`loadhistory("myfile")`	Reload a command's history (default = `.Rhistory`).
`save.image("myfile")`	Save the workspace to myfile (default = `.RData`).
`save(objectlist, file="myfile")`	Save specific objects to a file.
`load("myfile")`	Load a workspace into the current session (default = `.RData`).
`q()`	Quit R. You'll be prompted to save the workspace.

Robert I. Kabacoff: R in Action. Manning Publications, 2015

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Vamos a crear un nuevo archivo de código en R



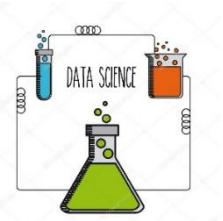
- Guarde el archivo con el nombre deseado, por ejemplo "introR".
  - Si hemos configurado bien el workspace, el archivo debería estar en esta carpeta.

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- R es un lenguaje de programación interpretado, es decir, el código será ejecutado instrucción por instrucción.
- Muchos de los datos y variables son almacenadas en memoria durante una sesión. Nosotros podemos guardar una sesión con la finalidad de conservar nuestro trabajo para futuras actividades.
- R utiliza el símbolo **<-** para detonar una asignación, a diferencia del típico = utilizado en muchos otros lenguajes de programación.
- ¡Veamos el ejemplo en el script example.R!

OBJETOS EN R

- Todo lo que puede ser asignado a una variable es considerado como un objeto en R.
- 6 clases de objetos básicos son:
  - character
  - numeric (números reales)
  - integer (números enteros)
  - logical (TRUE / FALSE). Valores de verdadero o falso.
  - factor. Sus valores son categóricos.



INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

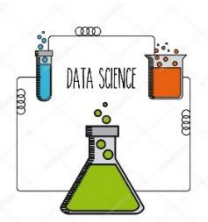
- R por defecto inicializa su trabajo obteniendo las entradas desde el teclado y despliega las salidas por la consola. Pero, es posible cambiar estas opciones, en los siguientes dos ejemplos ejecutaremos scripts con contenido de código en R llamándolos desde nuestro directorio script principal, luego procederemos a cambiar las salidas de los resultados a dos tipos de formatos.
- Descargue el archivo example2.R al directorio de trabajo, en el archivo actual de R que se encuentra trabajando utilice la función *source()* y dentro del paréntesis escriba el nombre del archivo descargado, es decir, *source("example2.R).* Finalmente, ejecute la anterior instrucción y observará que hará un llamado a un archivo externo con líneas de código.
- ¡Veamos el segundo ejemplo en el script example2.R!
- ¡Veamos el tercer ejemplo en el script example3.R!

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

Table 1.4 Functions for saving graphic output

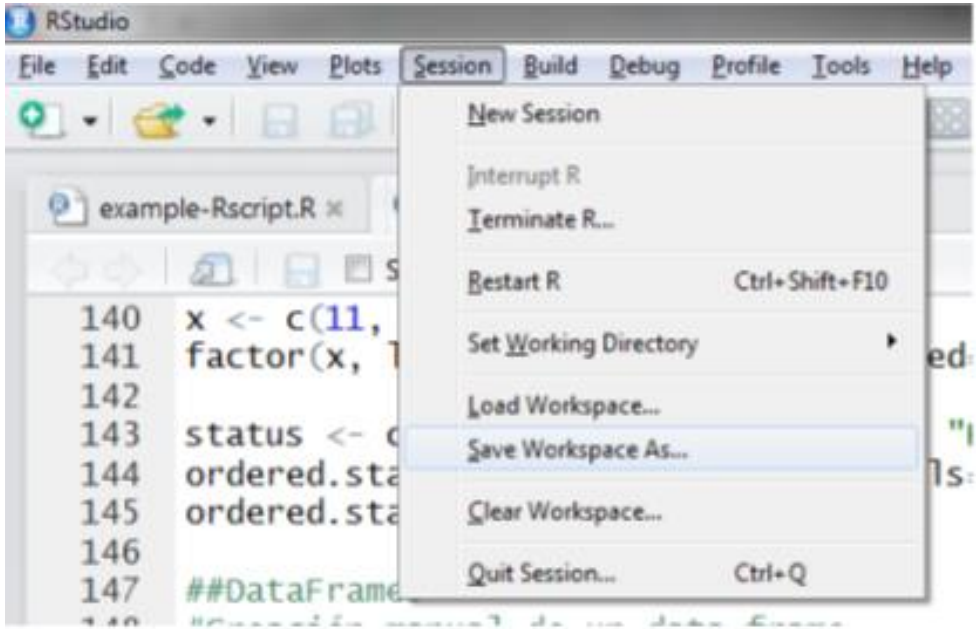
Function	Output
`pdf("filename.pdf")`	PDF file
`win.metafile("filename.wmf")`	Windows metafile
`png("filename.png")`	PBG file
`jpeg("filename.jpg")`	JPEG file
`bmp("filename.bmp")`	BMP file
`postscript("filename.ps")`	PostScript file

Robert I. Kabacoff: R in Action. Manning Publications, 2015



GUARDAR WORKSPACE

- El workspace es almacenado como un archivo .Rdata.
- Antes de cerrar Rstudio, el programa pregunta si desea guardar el workspace.
- Es una opción que puede guardarse en cualquier momento con la finalidad de salvaguardar el trabajo realizado.

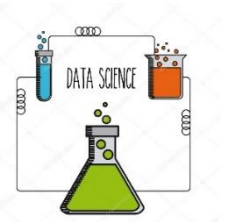


INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Markdown es un lenguaje de marcado ligero para formar archivos de texto, al utilizar este medio podrás producir archivos de texto plano (legibles por gran parte de los dispositivos).
- Gran parte de las plataformas permiten alojar sitios web estáticos que aceptan Markdown (un ejemplo de estos es GitHub) y traducen estos archivos como elementos HTML para su visualización en la web.
- Los archivos R Markdown(.Rmd) permiten crear archivos con codificación enriquecida, es decir, contienen el código, los resultados de este (por ejemplo, operaciones y visualizaciones) más texto y elementos desde Markdown. Este tipo de archivo permite crear reportes o documentos en documentos .html, .pdf, entre otros.
- Ahora veamos un nuevo archivo tipo R Markdown (RMD)

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Como hemos mencionado, la aplicación de la función rnorm nos dio un vector de cinco valores, ahora si deseamos crear un vector y asignarlo a una variable, debemos proceder a digitar la función c() seguido de los valores que queremos tener.
- ¡Veamos el ejemplo en el notebook!
- Ahora, si deseamos obtener mayor información sobre una función podemos utilizar la opción de ayuda con R a través de **?**,
- ¡Veamos el ejemplo en el notebook!

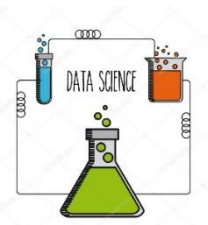


USO DE LA AYUDA

Table 1.2 R help functions

Function	Action
`help.start()`	General help
`help("foo")` or `?foo`	Help on function `foo` (quotation marks optional)
`help.search("foo")` or `??foo`	Searches the help system for instances of the string `foo`
`example("foo")`	Examples of function `foo` (quotation marks optional)
`RSiteSearch("foo")`	Searches for the string `foo` in online help manuals and archived mailing lists
`apropos("foo", mode="function")`	Lists all available functions with `foo` in their name
`data()`	Lists all available example datasets contained in currently loaded packages
`vignette()`	Lists all available vignettes for currently installed packages
`vignette("foo")`	Displays specific vignettes for topic `foo`

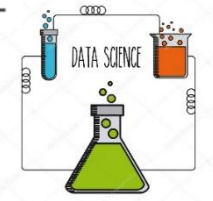
Robert I. Kabacoff: R in Action. Manning Publications, 2015



INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Los comentarios en el código podemos agregarlos con #
- Procedamos al siguiente ejercicio, vamos a crear una tabla a través de la asignación de vectores, utilizaremos unas funciones predefinidas en R para estadística descriptiva y vamos a crear un gráfico (conocido en ingles como *plot*).

Age (mo.)	Weight (kg.)	Age (mo.)	Weight (kg.)
01	4.4	09	7.3
03	5.3	03	6.0
05	7.2	09	10.4
02	5.2	12	10.2
11	8.5	03	6.1



INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN R

- Existen numerosas funciones de R en distintos paquetes que nos pueden facilitar la vida. La instalación de un paquete se realiza a través de:

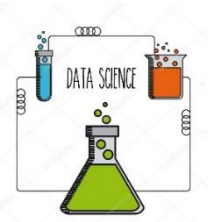
  install.packages("nombre del paquete")

- Como hemos mencionado, constantemente las comunidades lanzan nuevas versiones de sus paquetes ya sea para incorporar nuevas funcionalidades o para corregir algún error; si deseamos estar en la ultima versión podemos utilizar el siguiente comando:

  update.packages()

- Para cargar el paquete que necesitamos utilizar en nuestro entorno de trabajo se debe utilizar el siguiente comando en conjunto con el nombre del paquete

  library(gclus)

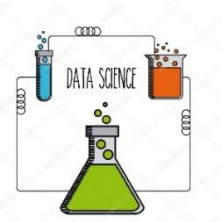


FUNCIONES

- Cada paquete trae consigo una serie de funciones (procedimientos) y la única forma de acceder a estas utilidades es cargando paquete a paquete a nuestro entorno de trabajo, la estructura general de una función puede interpretarse de la siguiente manera:

```
    nombre_funcion(parametro1 = valor1, parametro2 = valor2, ...)
```

- El objetivo de una función es abstraer una funcionalidad que puede ser utilizada en varias extensiones del código o en otros proyectos como una librería de funciones.
- Una función puede o no recibir argumentos (parámetros), por ejemplo, abs(x)
- Las funciones también pueden ser definidas por el programador



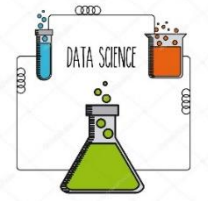
FUNCIONES

Function	Description
abs(*x*)	absolute value
sqrt(*x*)	square root
ceiling(*x*)	ceiling(3.475) is 4
floor(*x*)	floor(3.475) is 3
trunc(*x*)	trunc(5.99) is 5
round(*x*, **digits=***n*)	round(3.475, digits=2) is 3.48
signif(*x*, **digits=***n*)	signif(3.475, digits=2) is 3.5
cos(*x*), **sin**(*x*), **tan**(*x*)	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x*)	natural logarithm
log10(*x*)	common logarithm
exp(*x*)	e^*x*

Function	Description
substr(*x*, **start=***n1*, **stop=***n2*)	Extract or replace substrings in a character vector. x <- "abcdef" substr(x, 2, 4) is "bcd" substr(x, 2, 4) <- "22222" is "a222ef"
grep(*pattern*, *x* , **ignore.case=**FALSE, **fixed=**FALSE)	Search for *pattern* in *x*. If fixed =FALSE then *pattern* is a regular expression. If fixed=TRUE then *pattern* is a text string. Returns matching indices. grep("A", c("b","A","c"), fixed=TRUE) returns 2
sub(*pattern*, *replacement*, *x*, **ignore.case** =FALSE, **fixed=**FALSE)	Find *pattern* in *x* and replace with *replacement* text. If fixed=FALSE then *pattern* is a regular expression. If fixed = T then *pattern* is a text string. sub("\\s",".","Hello There") returns "Hello.There"
strsplit(*x*, *split*)	Split the elements of character vector *x* at *split*. strsplit("abc", "") returns 3 element vector "a","b","c"
paste(..., **sep=""**)	Concatenate strings after using *sep* string to seperate them. paste("x",1:3,sep="") returns c("x1","x2" "x3") paste("x",1:3,sep="M") returns c("xM1","xM2" "xM3") paste("Today is", date())
toupper(*x*)	Uppercase
tolower(*x*)	Lowercase

Function	Description
mean(*x*, **trim=**0, **na.rm=**FALSE)	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)
sd(*x*)	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(*x*)	median
quantile(*x*, *probs*)	quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
range(*x*)	range
sum(*x*)	sum
diff(*x*, **lag=***1*)	lagged differences, with lag indicating which lag to use
min(*x*)	minimum
max(*x*)	maximum
scale(*x*, **center=**TRUE, **scale=**TRUE)	column center or standardize a matrix.

http://www.statmethods.net/management/functions.html

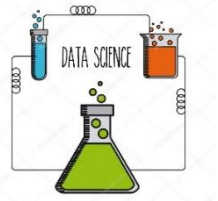


ERRORES COMUNES DE PROGRAMACIÓN

Common mistakes in R programming

Some common mistakes are made frequently by both beginning and experienced R programmers. If your program generates an error, be sure to check for the following:

- *Using the wrong case*—`help()`, `Help()`, and `HELP()` are three different functions (only the first will work).
- *Forgetting to use quotation marks when they're needed*—`install.packages("gclus")` works, whereas `install.packages(gclus)` generates an error.
- *Forgetting to include the parentheses in a function call*—For example, `help()` works, but `help` doesn't. Even if there are no options, you still need the `()`.
- *Using the \ in a pathname on Windows*—R sees the backslash character as an escape character. `setwd("c:\mydata")` generates an error. Use `setwd("c:/mydata")` or `setwd("c:\\mydata")` instead.
- *Using a function from a package that's not loaded*—The function `order.clusters()` is contained in the `gclus` package. If you try to use it before loading the package, you'll get an error.



CREANDO UN DATASET

- Un dataset es usualmente un arreglo rectangular de datos con filas que representan las observaciones y las filas las variables que lo presentan.

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	flight
1	2013	1	1	517	515	2	830	819	11	UA	1545
2	2013	1	1	533	529	4	850	830	20	UA	1714
3	2013	1	1	542	540	2	923	850	33	AA	1141
4	2013	1	1	544	545	-1	1004	1022	-18	B6	725
5	2013	1	1	554	600	-6	812	837	-25	DL	461
6	2013	1	1	554	558	-4	740	728	12	UA	1696
7	2013	1	1	555	600	-5	913	854	19	B6	507
8	2013	1	1	557	600	-3	709	723	-14	EV	5708
9	2013	1	1	557	600	-3	838	846	-8	B6	79
10	2013	1	1	558	600	-2	753	745	8	AA	301
11	2013	1	1	558	600	-2	849	851	-2	B6	49
12	2013	1	1	558	600	-2	853	856	-3	B6	71
13	2013	1	1	558	600	-2	924	917	7	UA	

ESTRUCTURAS DE DATOS EN R

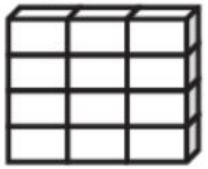
La más básica. Una dimensión
Todos los objetos deben ser de la misma clase

Arreglo de dos dimensiones, todos los objetos de la misma clase

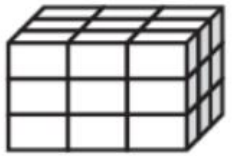
Arreglo que puede tener más de dos dimensiones, todos los objetos de la misma clase

(a) Vector

(b) Matrix

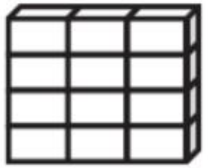


(c) Array



La más común, dos dimensiones, objetos pueden ser de diferente tipo

(d) Data frame



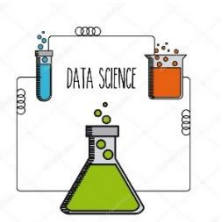
Columns can be different modes

(e) List

- Vectors
- Arrays
- Data frames
- Lists

Un tipo especial de vector que puede contener objetos de distintas clases. Es el tipo de dato más complejo

Robert I. Kabacoff: R in Action. Manning Publications, 2015

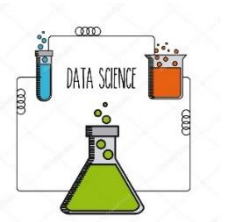


VECTORES

- Son de una sola dimensión
- Un vector solo puede tener datos de un solo tipo, por ejemplo, datos numéricos, de tipo carácter, o lógicos.

```{R vector}
x <-c(0.5, 0.6) # numérico
typeof(x)
x <-c(TRUE, FALSE) #logico
typeof(x)
x <-c("a", "b", "c") #caracter
typeof(x)
x <-c(10:20) # integer/entero
typeof(x)
x <-c(1+0i, 2+4i) #compleja
typeof(x)
```

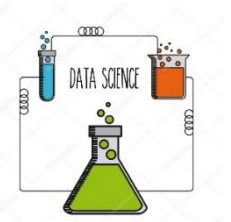
```
[1] "double"
[1] "logical"
[1] "character"
[1] "integer"
[1] "complex"
```



MATRICES

- Son arreglos de dos dimensiones con todos los registros del mismo tipo

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
```

FACTORES

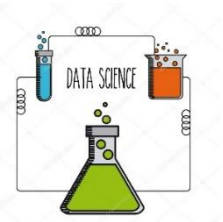
- Son variables categóricas

Nominales

- No implican un orden ni representan una cantidad.
  - Ej: Diabetes(Tipo1, Tipo2)
  - Incluso si los tipos fueran 1 y 2, no implicarían un orden

Ordinales

- Implican un orden, pero no cantidad
  - Ej: PatientStatus(Poor, Improved, Excellent)



FACTORES

- Creación de factores nominales
  - diabetes <- c("Type1", "Type2", "Type1", "Type1")
  - diabetesFactor <- factor(diabetes)
- Esta última sentencia almacena el vector internamente como (1,2,1,1) y a cada número le asigna la categoría correspondiente (la asignación es alfabética). La categoría en formato character es lo que se despliega al imprimir el factor
- Visualicemos el contenido del factor diabetes en el explorador del ambiente de ejecución
- Los niveles de un factor corresponden con las distintas categorías que conforman el factor

A note for programmers

Experienced programmers typically find several aspects of the R language unusual. Here are some features of the language you should be aware of:

- The period (.) has no special significance in object names. But the dollar sign ($) has a somewhat analogous meaning, identifying the parts of an object. For example, `A$x` refers to variable `x` in data frame A.
- R doesn't provide multiline or block comments. You must start each line of a multiline comment with `#`. For debugging purposes, you can also surround code that you want the interpreter to ignore with the statement `if(FALSE){…}`. Changing the `FALSE` to `TRUE` allows the code to be executed.
- Assigning a value to a nonexistent element of a vector, matrix, array, or list will expand that structure to accommodate the new value. For example, consider the following:

```
> x <- c(8, 6, 4)
> x[7] <- 10
> x
[1]  8  6  4 NA NA NA 10
```

  The vector `x` has expanded from three to seven elements through the assignment.
  `x <- x[1:3]` would shrink it back to three elements again.
- R doesn't have scalar values. Scalars are represented as one-element vectors.
- Indices in R start at 1, not at 0. In the vector earlier, `x[1]` is 8.
- Variables can't be declared. They come into existence on first assignment.

To learn more, see John Cook's excellent blog post, *R programming for those coming from other languages* (www.johndcook.com/R_language_for_programmers.html).

Programmers looking for stylistic guidance may also want to check out *Google's R Style Guide* (http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html).

Robert I. Kabacoff: R in Action. Manning Publications, 2015

Function	Purpose
`length(object)`	Number of elements/components.
`dim(object)`	Dimensions of an object.
`str(object)`	Structure of an object.
`class(object)`	Class or type of an object.
`mode(object)`	How an object is stored.
`names(object)`	Names of components in an object.
`c(object, object,...)`	Combines objects into a vector.
`cbind(object, object, ...)`	Combines objects as columns.
`rbind(object, object, ...)`	Combines objects as rows.
`object`	Prints the object.
`head(object)`	Lists the first part of the object.
`tail(object)`	Lists the last part of the object.
`ls()`	Lists current objects.
`rm(object, object, ...)`	Deletes one or more objects. The statement `rm(list = ls())` will remove most objects from the working environment.
`newobject <- edit(object)`	Edits object and saves as newobject.
`fix(object)`	Edits in place.

Robert I. Kabacoff: R in Action. Manning Publications, 2015

REFERENCIAS

- Kabacoff, R. (2015). R IN ACTION: Data analysis and graphics with R.
- Wickham, H., & Grolemund, G. (2016). R for Data Science.