Curso básico de lenguaje R aplicado a las Ciencias Sociales

Curso básico de lenguaje R aplicado a las Ciencias Sociales

Curso organizado por la Oficina de Apoyo al Investigador de la Universidad Internacional de la Rioja

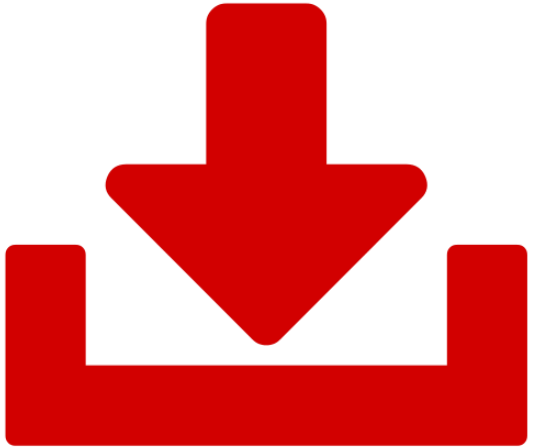
Versión 2.0
Abril 2019

Nicolás Robinson-Garcia

http://nrobinsongarcia.com

10 de abril 2019
Curso de 10 horas

Todo el material del curso así como recursos adicionales están disponibles



Nicolas Robinson-Garcia

I am a social scientist specialized in bibliometrics. I work as a postdoctoral researcher at Delft Institute of Applied Mathematics, TU Delft.

BIO CV

PUBLICATIONS

¿Quién soy yo y cuánto sé de este tema?

- ✓ Licenciado en Documentación
- ✓ Doctor en Ciencias Sociales
- ✓ Investigador Marie Sklodowska-Curie

Agenda

PARTE 1

1. Presentación
2. ¿Qué es y por qué R?
3. Instalación y toma de contacto
4. Importación de datos

PARTE 2

1. El uso de paquetes y librerías
2. Intro al paquete *ggplot2*
3. Análisis descriptivo y visualización
4. Puesta en común

Parte 1

- Presentación
- ¿Qué es y por qué R?

- Instalación y toma de contacto
- Importación y exploración de datos

Presentación

PARTE I-I

¿Qué aprenderéis en este curso?

Lenguaje estadístico R

- Qué es y por qué merece la pena aprenderlo
- Los principios básicos para aprender a programar en R
- Paquetes de interés para la visualización de datos y para el análisis estadístico
- Recursos y consejos para buscar ayuda y seguir con la formación en R

¿Qué NO aprenderéis en este curso?

Análisis estadístico

- Conocimiento pormenorizado de las técnicas de análisis
- La aplicación de técnicas de análisis avanzadas

¿Qué NO aprenderéis en este curso?

Lenguaje estadístico R

- Poder realizar cualquier tipo de análisis estadístico con R
- Conocer todas las características del lenguaje
- Programar en R


© 1993 Watterson/Distributed by Universal Press Syndicate
THIS IS A BIG, FAT WASTE OF MY TIME!

Contenidos

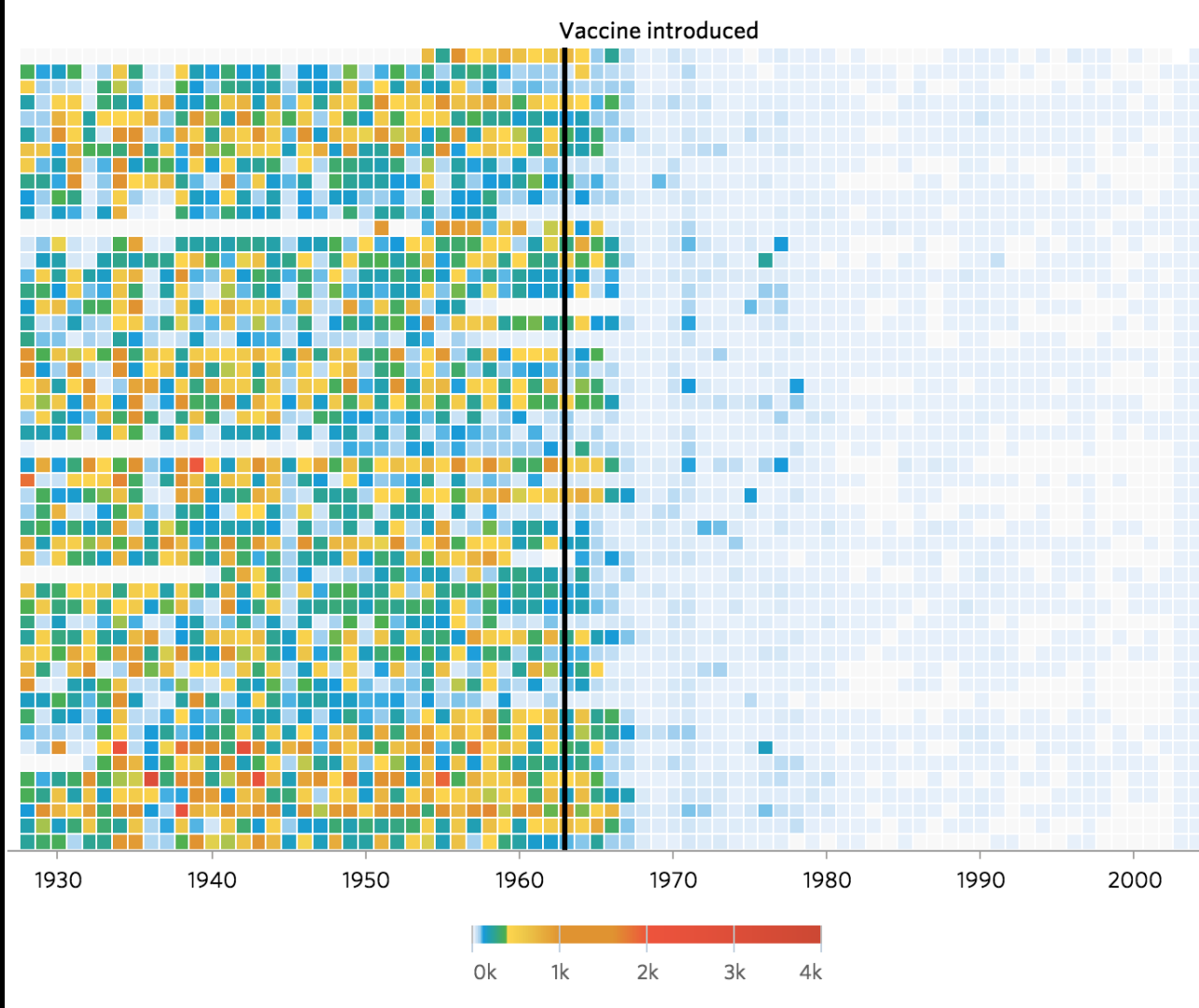
Programación en R

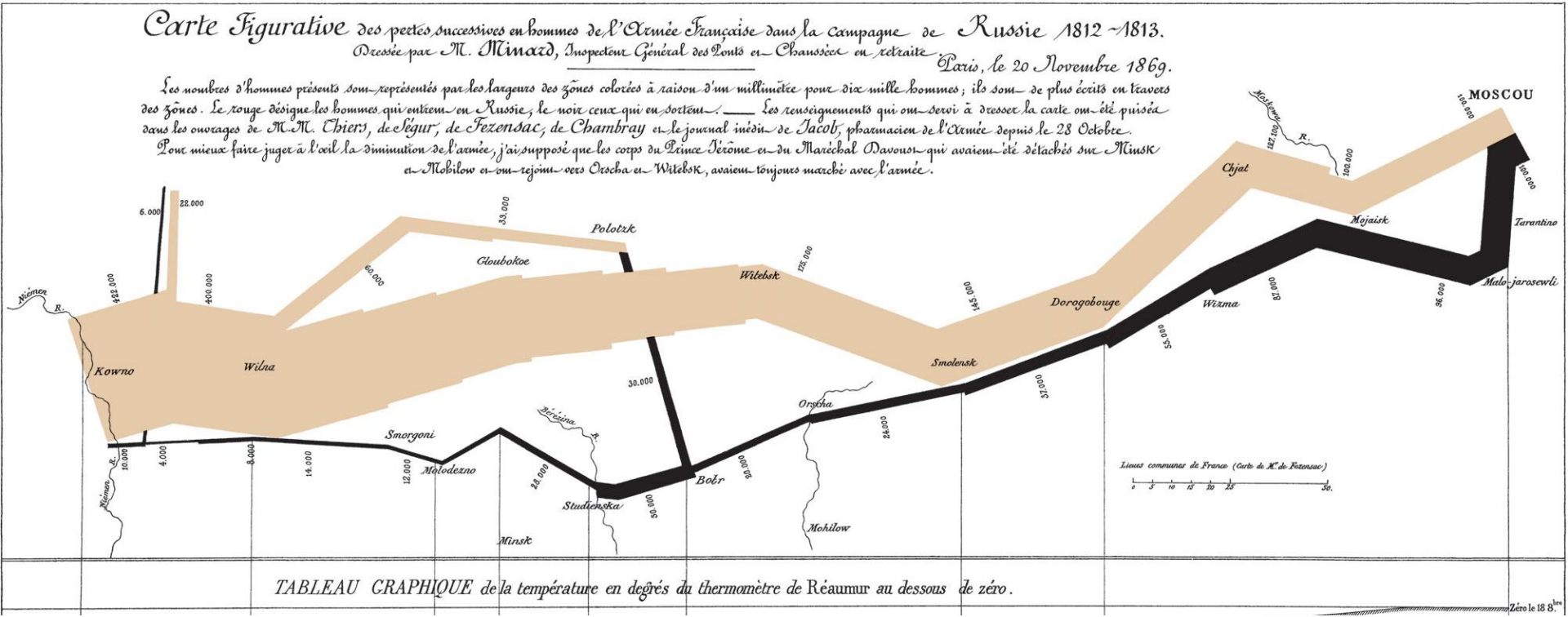
- Introducción al lenguaje de R
- Directrices básicas de comandos, software adicional y acceso a formación
- Paquetes básicos para la visualización y el análisis de datos
- Dinámicas de análisis descriptivo de datos: Importación, descriptivos y visualización de datos

¿Qué es R y por qué?

1. Por qué R
2. Un poco de historia
3. R y RStudio
4. Comandos básicos

PARTE I-2

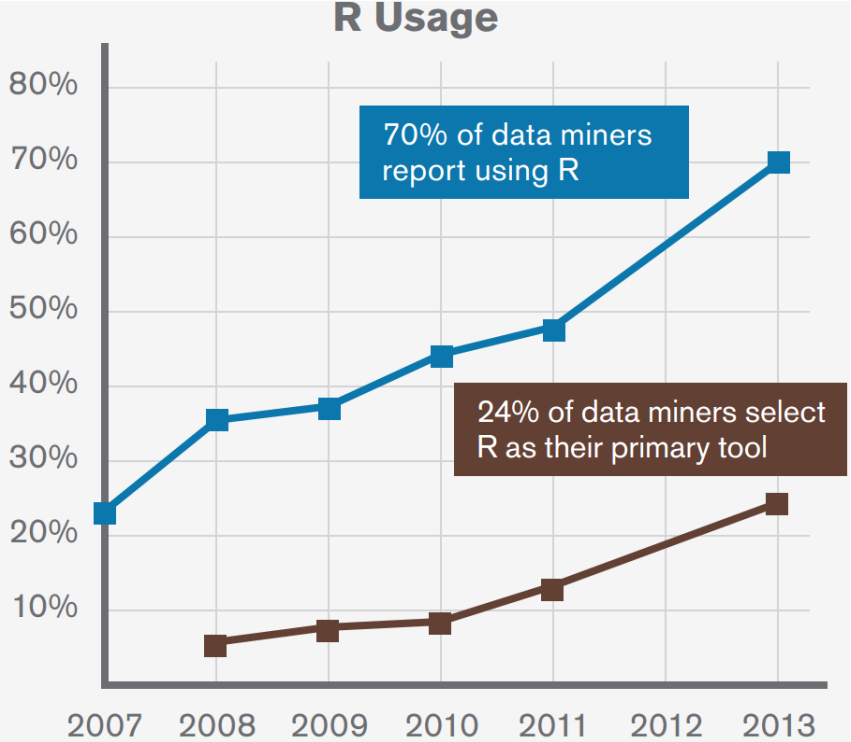




Razones por las que vale la pena aprender R

1	Gratuito y abierto
2	Compatible con distintas plataformas
3	Reproducible
4	Universal y colaborativo
5	Flexible y en continuo crecimiento

Fuente: https://rafalab.github.io/dsbook/getting-started.html



Alternativas:

- X SPSS
- X SAS
- X Stata
- X MatLab

1	Gratuito y abierto

2	Compatible con distintas plataformas

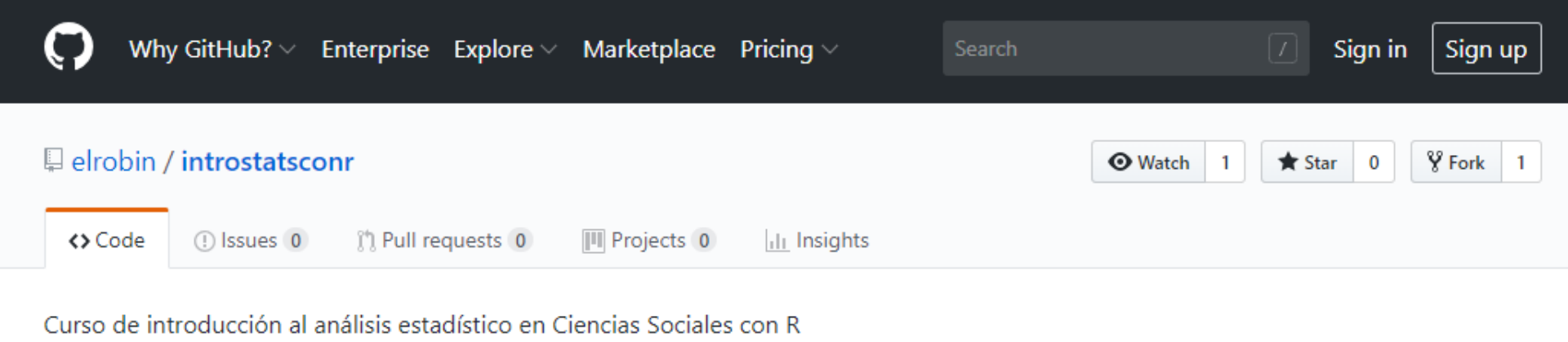
R es reproducible

Se dice que R es reproducible ya que al ser necesario escribir en código todos y cada uno de los pasos que se siguen en el tratamiento de los datos desde el momento en que se importan, cualquier persona con el mismo set de datos y el mismo código debería obtener los mismos resultados. Por ejemplo, todos tenemos el set de datos `mtcars` por defecto en `R`. Por tanto, si correlacionamos las cuatro primeras columnas del set de datos, todos deberíamos obtener los mismos resultados.

```
cor(mtcars[,1:4])
```

```
##             mpg        cyl       disp         hp
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475
## disp -0.8475514  0.9020329  1.0000000  0.7909486
## hp   -0.7761684  0.8324475  0.7909486  1.0000000
```

3	Reproducible



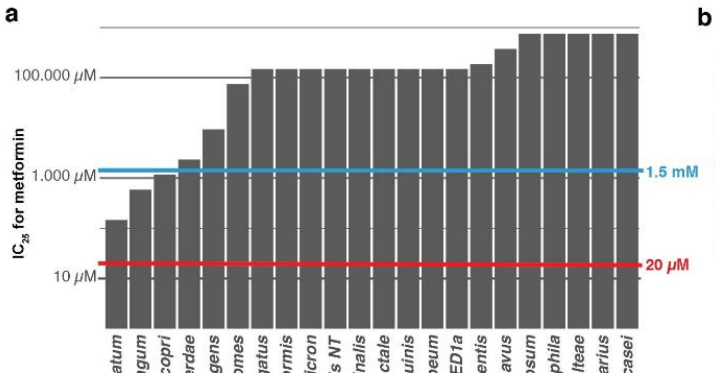
4	Universal y colaborativo

The tidyverse

Components

5	Flexible y en continuo crecimiento

Algunos ejemplos



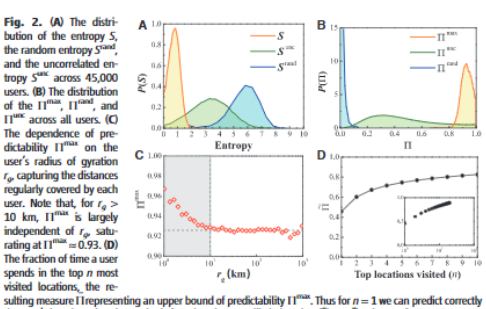


Fig. 2. (**A**) The distribution of the entropy S, the random entropy S^{rand}, and the uncorrelated entropy S^{unc} across 45,000 users. (**B**) The distribution of the Π^{max}, Π^{rand}, and Π^{unc} across all users. (**C**) The dependence of predictability Π^{max} on the user's radius of gyration r_g, capturing the distances regularly covered by each user. Note that, for $r_g >$ 10 km, Π^{max} is largely independent of r_g, saturating at $\Pi^{max} = 0.93$. (**D**) The fraction of time a user spends in the top n most visited locations, the resulting measure Π representing an upper bound of predictability Π^{max}. Thus for $n = 1$ we can predict correctly the user's location when he or she is found at the most likely location ("home"), whereas for $n = 2$, we can predict correctly the user's whereabouts when he is in one of his top two locations ("home" or "office"). (Inset) Π appears to grow approximately logarithmically with n.

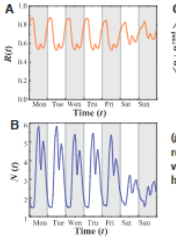


Fig. 3. (**A**) The hourly regularity $R(t)$ over a week-long time period, measuring the fraction of instances when the user is found in his or her most visited location during the corresponding hour-long period. (**B**) The average number of visited locations $N(t)$ during each hourly time frame within a week, revealing that high regularity $R(t)$ correlates with small $N(t)$. (**C**) The averaged $R|R^{rand}$ versus the radius of gyration r_g, showing that the users with large r_g have high relative regularity.

mobility pattern of the users hides an unex
pectedly high degree of potential predictabilit
We have also determined the maximal predict
ability Π^{unc} and the random predictability Π^{rand}
extracted from S^{unc} and S^{rand}. As Fig. 2B shows
the result is strikingly different—$P(\Pi^{unc})$ is ex
tremely widely distributed and peaked at Π^{unc} =
0.3, which indicates that, if we rely only on th
heterogeneous spatial distribution, the predict
ability across the whole population is insignif
icant and varies widely from person to person
Similarly, $P(\Pi^{rand})$ has a peak at $\Pi^{rand} = 0$, whic
suggests not only that Π^{rand} and Π^{unc} are inef
fective as predictive tools, but also that a sig
nificant share of predictability is encoded in th
temporal order of the visitation pattern.

How can we reconcile the wide variability i
the observed travel distances, as captured by th
fat-tailed $P(r_g)$, with the highly bounded predict
ability observed across the user population? T
answer this, we measured the dependency o
Π^{max} on r_g, and found that, for $r_g \geq 10$ km, pre
dictability becomes largely independent of r_g
saturating at $\Pi^{max} \approx 0.93$ (Fig. 2C). Therefore
Fig. 2C explains the failure of our earlier hy
pothesis: Individuals with $r_g \geq 100$ km, coverin
hundreds of kilometers on a regular basis, are ju
as predictable as those whose life is constraine
to a $r_g \approx 10$-km neighborhood, a saturation th
lies behind the high predictability observed acros
the whole user base.

To determine how much of our predictabilit
is really rooted in the visitation patterns of the to
locations, we calculated the probability Π that, i
a given moment, the user is in one of the top
most visited locations, where $n = 2$ typicall
captures home and work. Thus, Π represents a
upper bound for Π^{max}, as, even if our predictiv
algorithm is 100% accurate, it can foresee th
future location only when the user is found in on
of the top n locations monitored by the algorithm
As Fig. 2D shows, the top two locations ($n = 2$

Algunos ejemplos

Nature

Software

Policy information about availability of computer code

7. Software

Describe the software used to analyze the data in this study.	Data was processed using R (version 3.4.2) using scripts deposited at https://git.embl.de/mkuhn/drug_impact_gut_bacteria, Iris (doi:10.1038/nmicrobiol.2017.14)

For manuscripts utilizing custom algorithms or software that are central to the paper but not yet described in the published literature, software must be made available to editors and reviewers upon request. We strongly encourage code deposition in a community repository (e.g. GitHub). *Nature Methods* guidance for providing algorithms and software for publication provides further information on this topic.

Recursos

- **Cursos Data Science** - Johns Hopkins University
- **R for Data Science** - Hadley Wickham
- **Introduction to Data Science** - Rafael A. Irizarry
- **R Journal** **- Q2 JCR**
- **Journal of Statistical Software** **- Q1 JCR**

Un poco de historia

1976 John Chambers desarrolla el lenguaje de programación S en Bell Labs.

1992 R. Ihaka y R. Gentleman comienzan a desarrollar R basado en S.

1995 M. Mächler se une y les anima a liberar el código en abierto

2000 Se publica la primera versión beta estable de R.

2009 New York Times ***se hace eco*** del éxito y crecimiento en el uso de R.

2011 Lanzamiento oficial de la primera versión de ***RStudio***.

Características de R

Lenguaje de programación sin interfaz

Curva de aprendizaje alta

Actualizaciones frecuentes

Estructura modular

https://cloud.r-project.org/

RStudio

Interfaz para R libre y gratuita

Una amplia comunidad detrás

Actualizaciones constantes y recursos propios

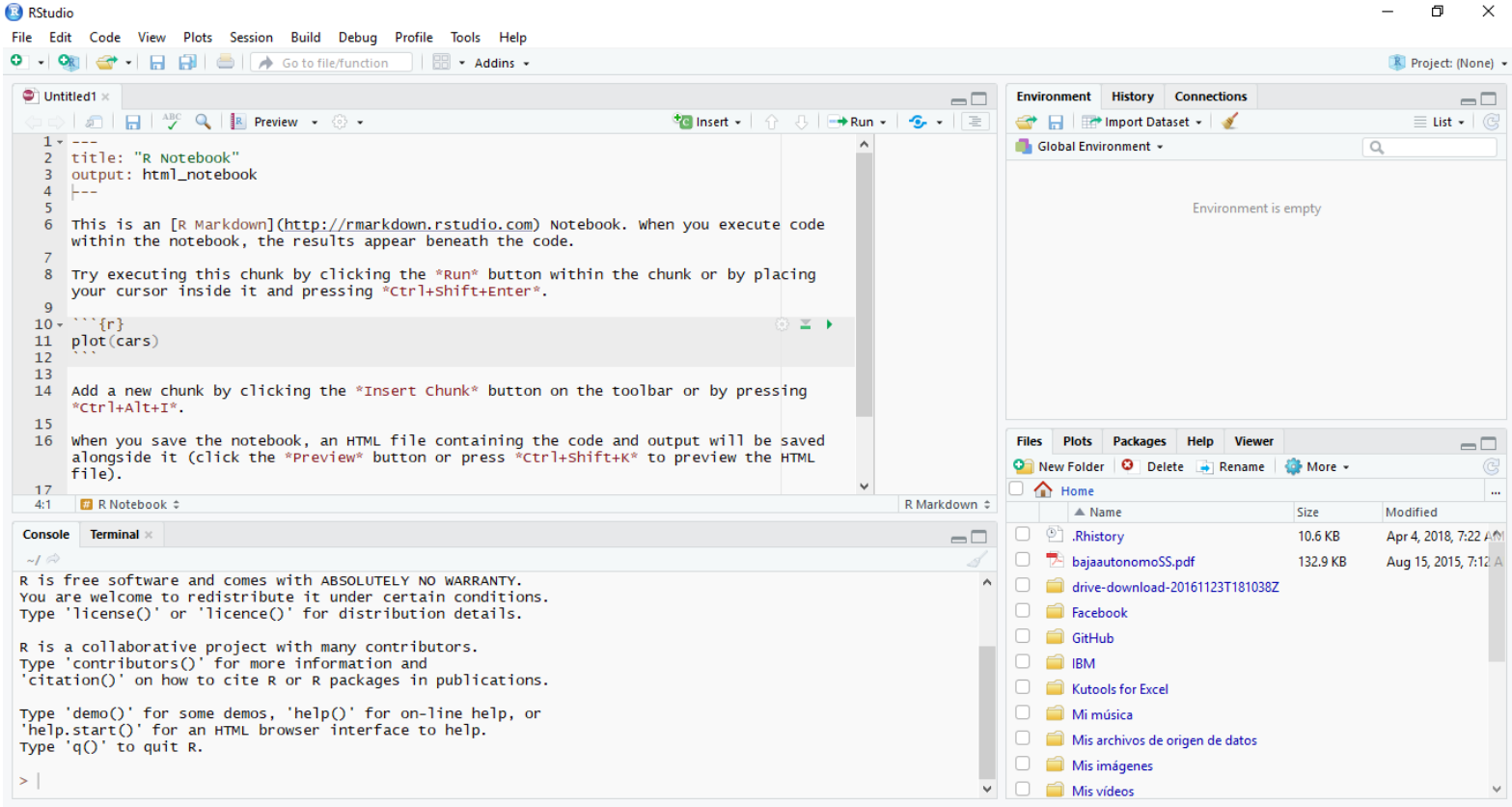
Ofrece un entorno integrado

https://www.rstudio.com

- ✓ Publicación de documentos en PDF
- ✓ Cuadernos de notas RMarkdown
- ✓ Creación de presentaciones y páginas web
- ✓ Creación de aplicaciones
- ✓ Control de versiones - GIT

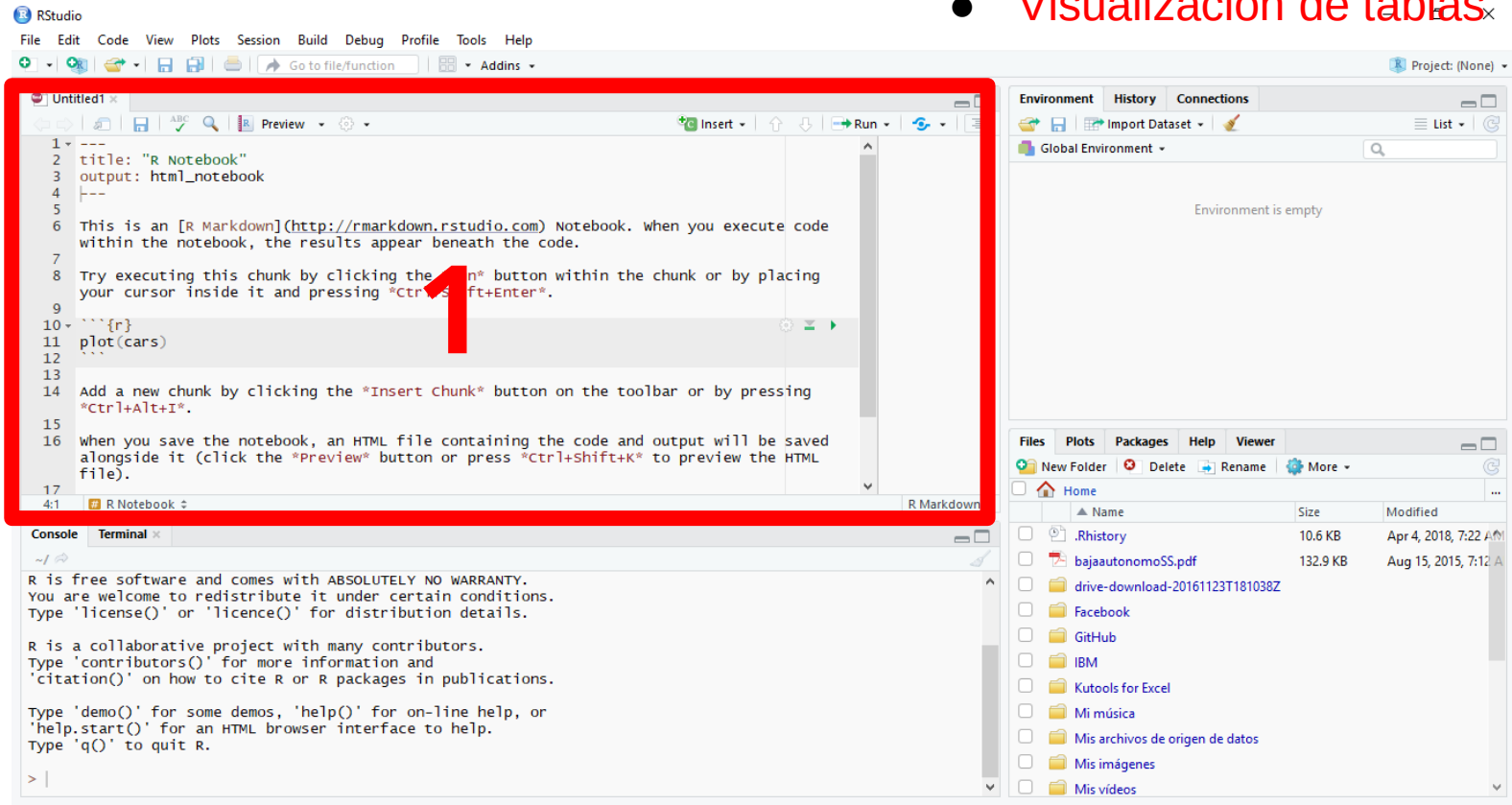
No es solo una interfaz para programar

¿Cómo es RStudio?



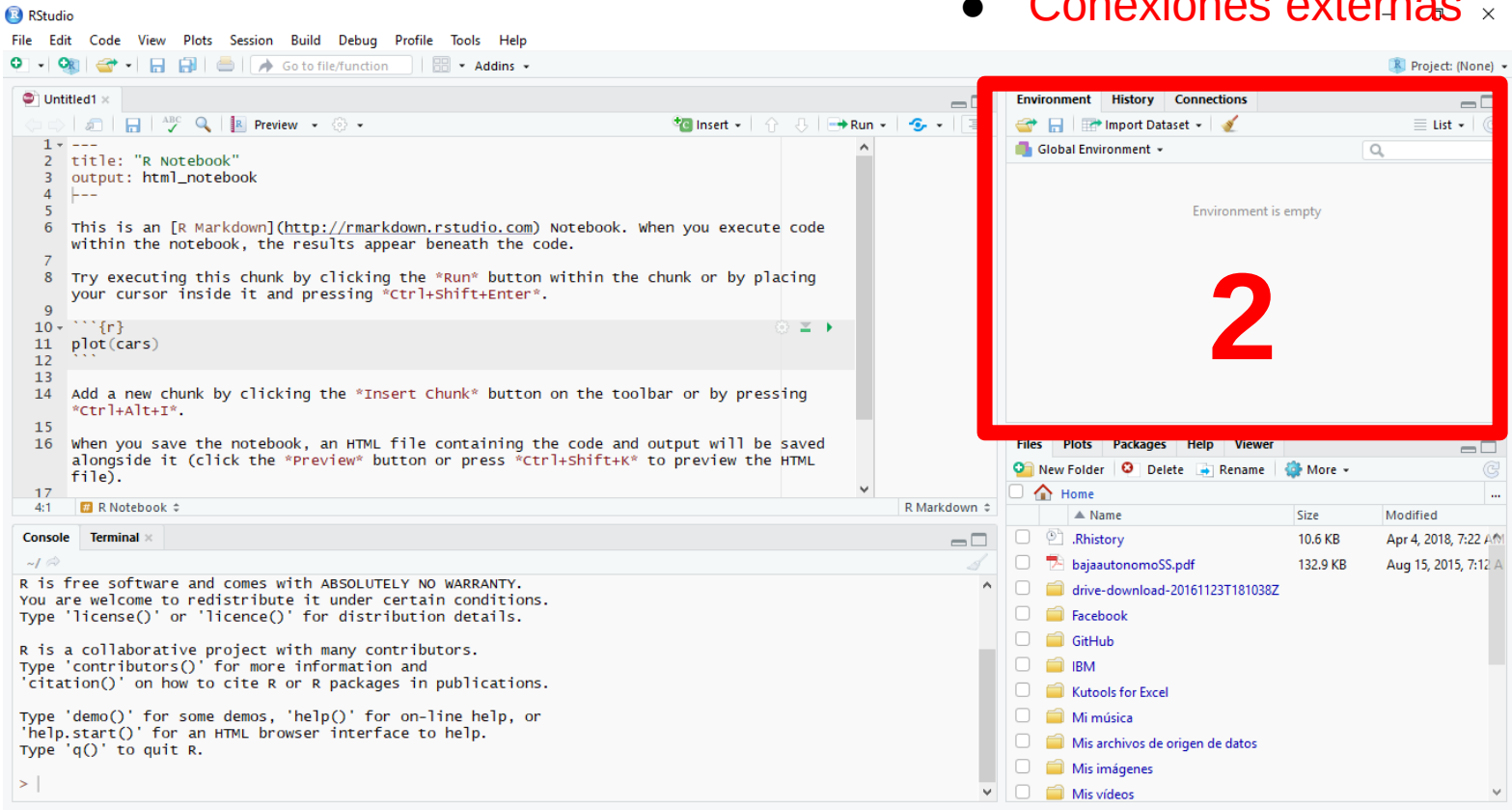
Zona de trabajo

- Scripts
- Notas
- Visualización de tablas



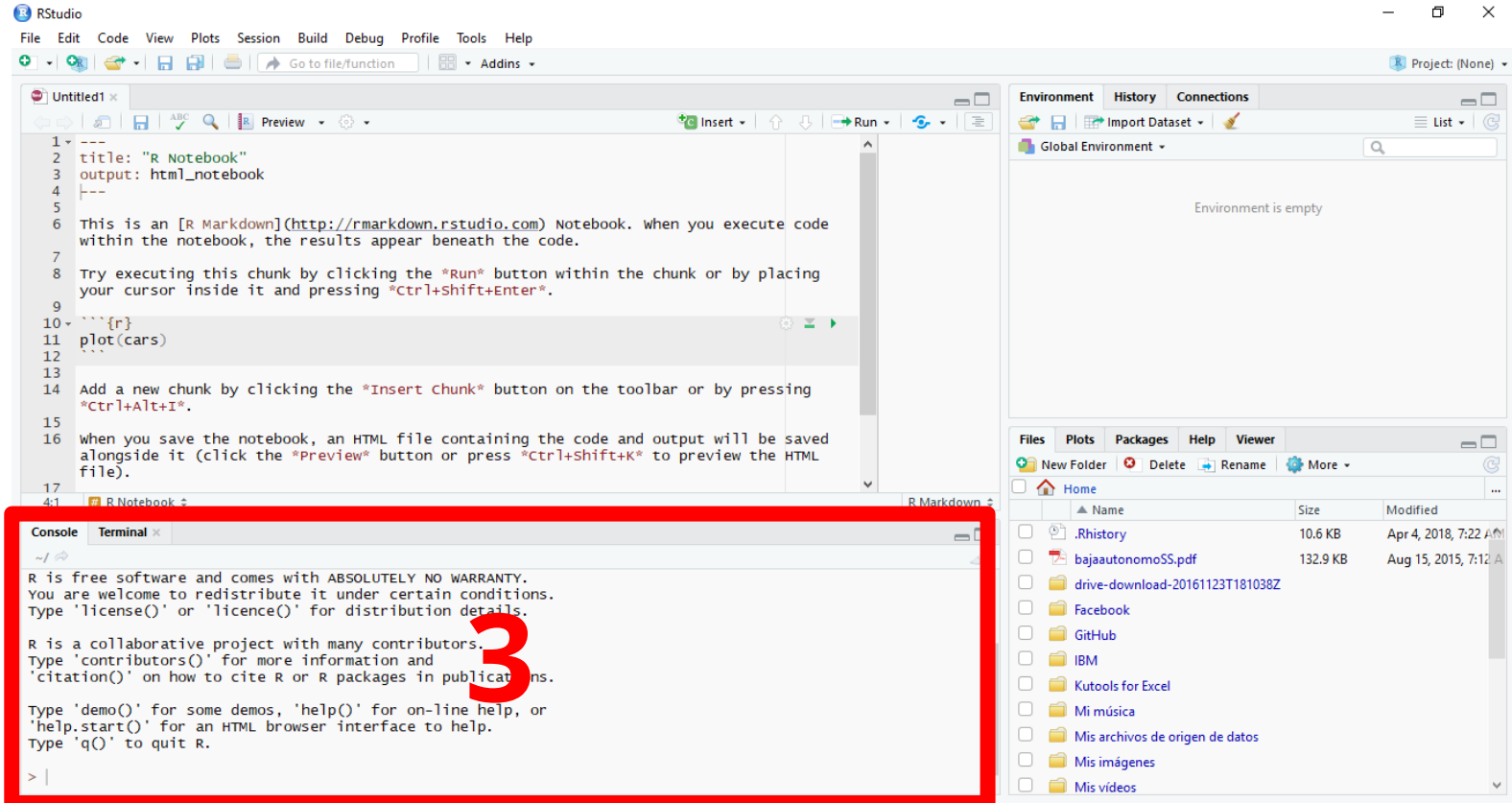
Zona de medio

- Objetos cargados
- Historial
- Conexiones externas



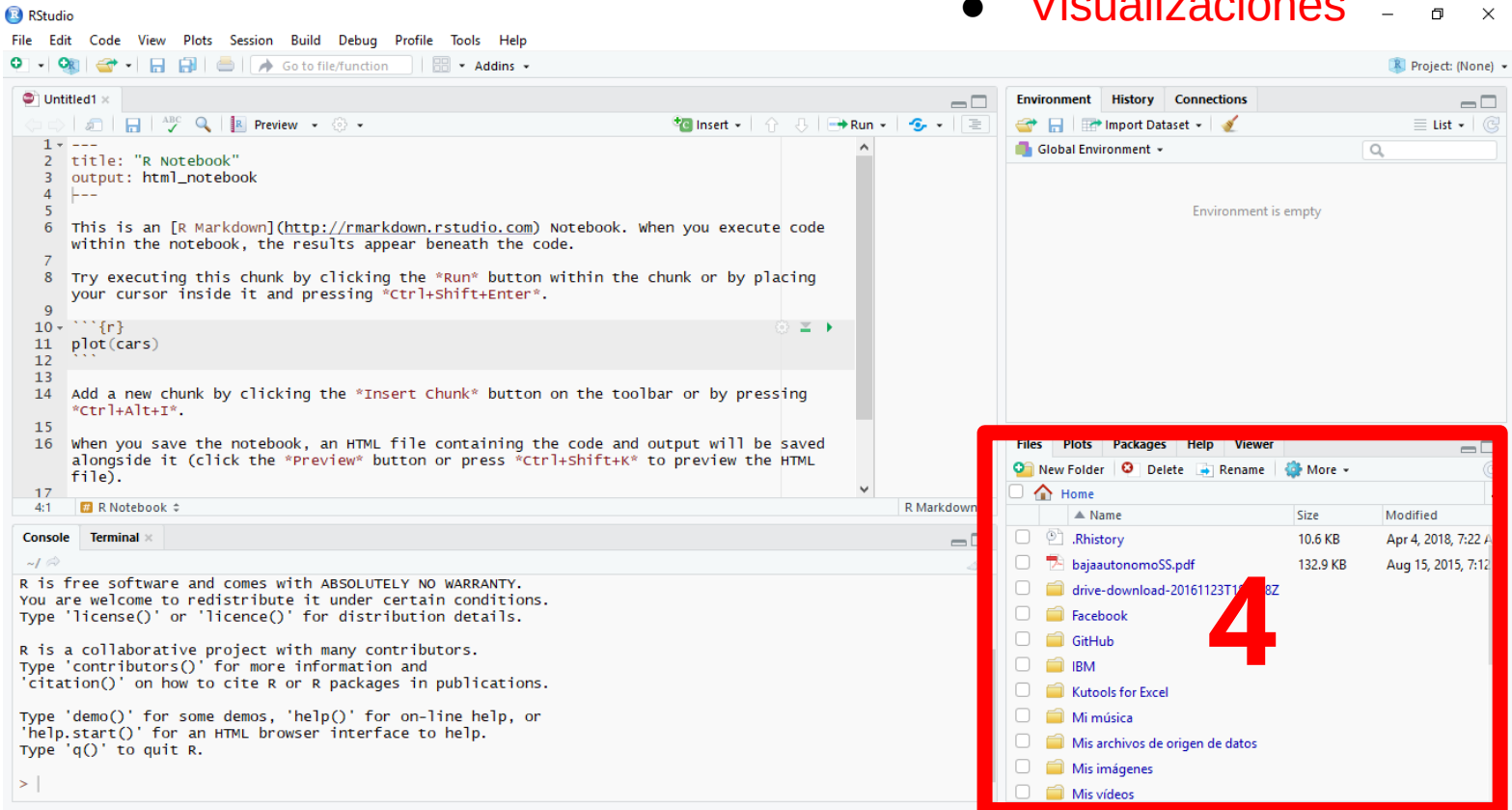
Zona de ejecución

- Consola



Zona de archivos

- Archivos externos
- Ayuda
- Visualizaciones



Conoce bien las reglas, para poder romperlas de forma efectiva

COMANDOS BÁSICOS

Todo es un objeto...

```
> presentacion <- "Buenos días mundo"
> presentacion
[1] "Buenos días mundo"
> 
```

... una frase

COMANDOS BÁSICOS

Todo es un objeto...

```
> suma <- 5+8
> suma
[1] 13
>
```

... una operación matemática

COMANDOS BÁSICOS

ELEMENTOS BÁSICOS

Todo es un objeto...

```
[1] 13
> lista_de_compra <- c("tomates", "lechuga", "pimientos", "patatas", "manzanas")
> lista_de_compra
[1] "tomates"   "lechuga"   "pimientos" "patatas"   "manzanas"
>
```

... una lista

COMANDOS BÁSICOS

ELEMENTOS BÁSICOS

Todo es un objeto...

```
> vector <- c(3, 1, 2, 5, 10)
> vector
[1]  3  1  2  5 10
>
```

... un vector

COMANDOS BÁSICOS

ELEMENTOS BÁSICOS

Todo es un objeto...

```
> matriz <- matrix(1:6, 2, 2)
> matriz
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> 
```

... una matriz

COMANDOS BÁSICOS

ELEMENTOS BÁSICOS

Todo es un objeto...

```
> data <- as.data.frame(cbind(vector, lista_de_compra))
> data
  vector lista_de_compra
1      3          tomates
2      1          lechuga
3      2        pimientos
4      5          patatas
5     10         manzanas
```

... un data frame

Diferencias entre data frame y matriz

En una matriz todas las columnas tienen el mismo tipo de datos, mientras que en un data frame se pueden incluir distintos tipos de datos (e.g., numérico, ordinal, etc.)

Normalmente trabajaremos con data frames

COMANDOS BÁSICOS

```
[1] 13
> lista_de_compra <- c("tomates", "lechuga", "pimientos", "patatas", "manzanas")
> lista_de_compra
[1] "tomates"   "lechuga"   "pimientos" "patatas"   "manzanas"
> 
```

```
> presentacion <- "Buenos días mundo"
> presentacion
[1] "Buenos días mundo"
> 
```

COMANDOS BÁSICOS

Los elementos entrecomillados se consideran valores de tipo textual.

Si no se trata de números, y no está entrecomillado, se considera un objeto o función

```
> data <- as.data.frame(cbind(vector, lista_de_compra))
> data
```

```
> vector <- c(3, 1, 2, 5, 10)
> vector
```

Algunas funciones

```
> data <- as.data.frame(cbind(vector, lista_de_compra))
> data
  vector lista_de_compra
1      3          tomates
2      1          lechuga
3      2        pimientos
4      5          patatas
5     10         manzanas
```

Algunas funciones

```
> data <- as.data.frame(cbind(vector, lista_de_compra))
> data
  vector lista_de_compra
1      3          tomates
2      1          lechuga
3      2        pimientos
4      5          patatas
5     10         manzanas
```

Algunas funciones

```
> data <- as.data.frame(cbind(vector, lista_de_compra))
> data
  vector lista_de_compra
1      3          tomates
2      1          lechuga
3      2         pimientos
4      5          patatas
5     10         manzanas
```

1. **cbind() - Une una serie de objetos en columnas**
rbind() - Une filas

2. **as.data.frame() - Convierte el objeto en un data frame**

Kit de ayuda

1. Información sobre funciones y ayuda dentro de RStudio

>?cbind()

>??cbind

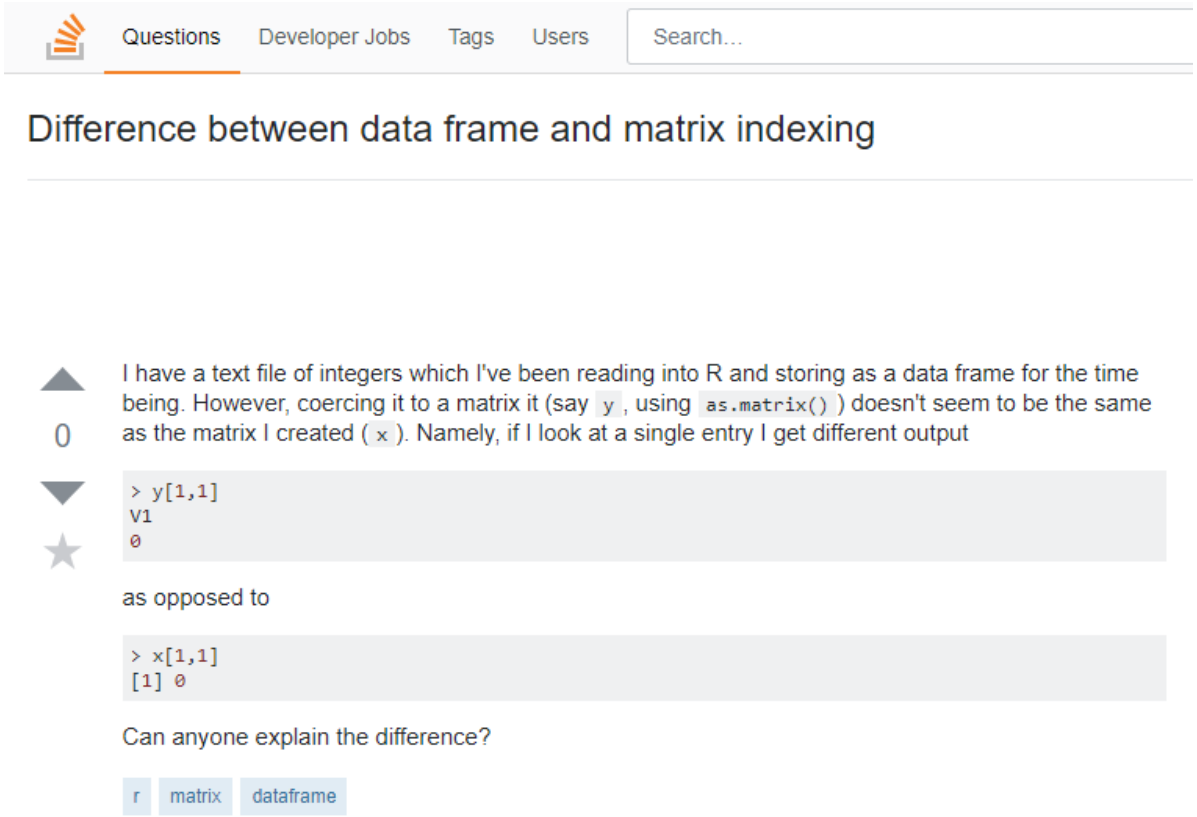
1. Ayuda en la web

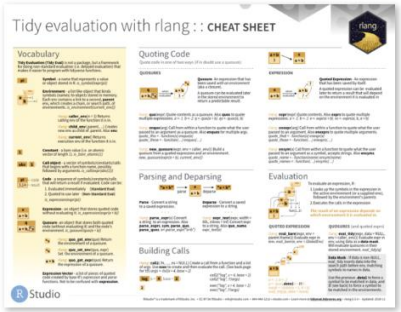
r how to *[función a realizar]*

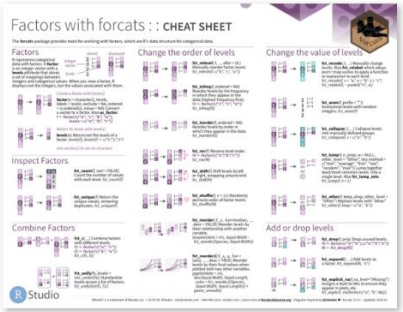
Kit de ayuda

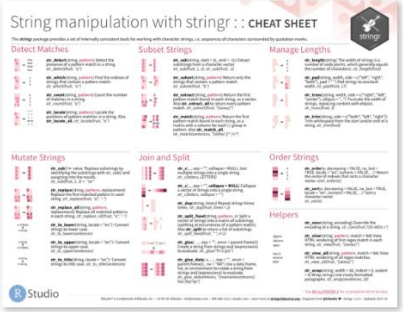
Stackoverflow

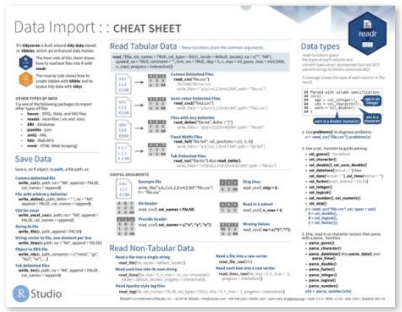
El lugar para preguntarle a la comunidad











https://www.rstudio.com/resources/cheatsheets/

Ayuda en español

- Lista de correos muy activa
- Grupos locales de apoyo
- Anuncio de cursos formativos

http://r-es.org

Instalación y toma de contacto

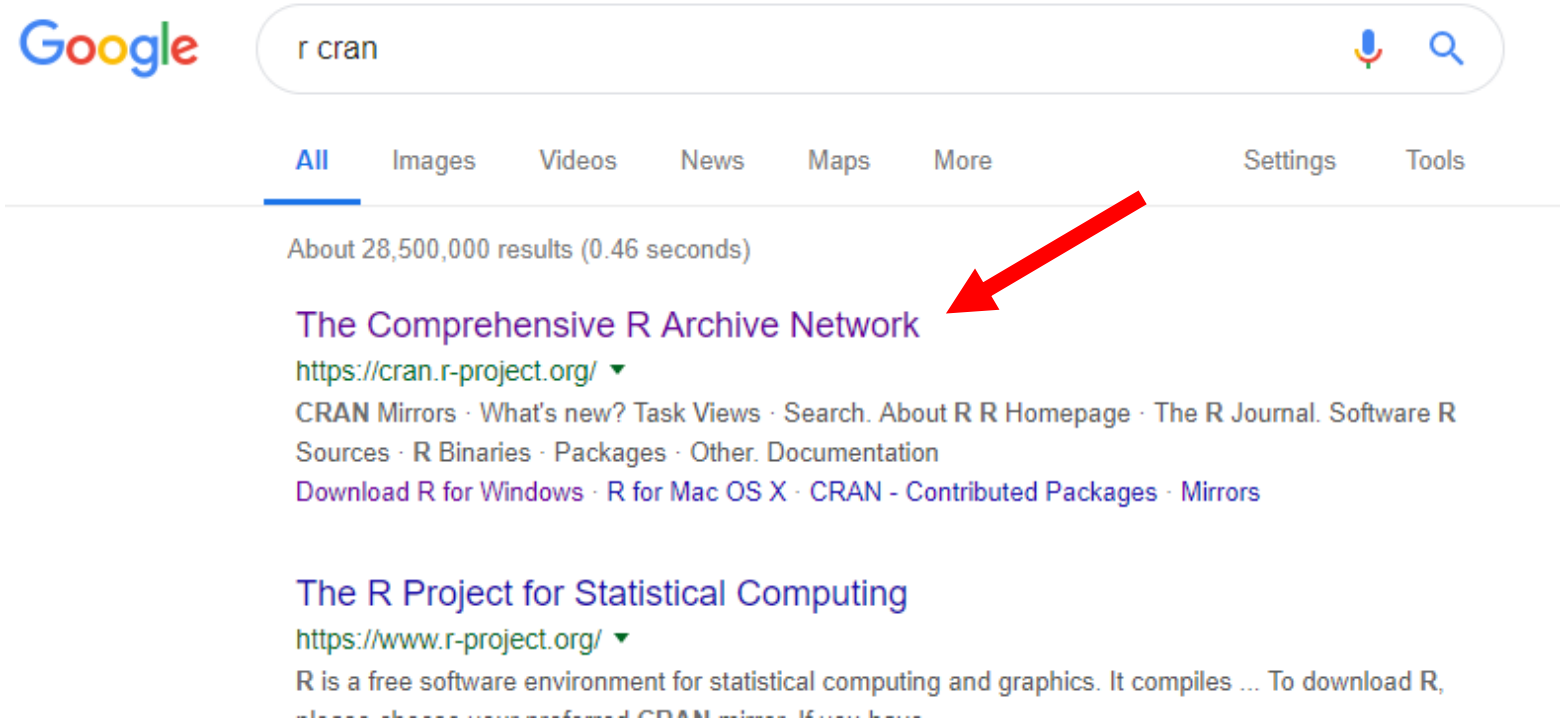
1. R CRAN y versiones

2. RStudio: proyectos y tipos de ficheros

3. Primeros pasos en el análisis de datos

PARTE I-3

Para instalar R debemos introducir en nuestro buscador *r cran*



Descarga la opción correspondiente con vuestro sistema operativo

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
Contributed

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 (2018-03-15, Someone to Lean On) R-3.4.4.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

Incluso si se trata de una actualización, pincha en la opción de instalar R por primera vez

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

R-3.4.4 for Windows (32/64 bit)

Download R 3.4.4 for Windows (62 megabytes, 32/64 bit)

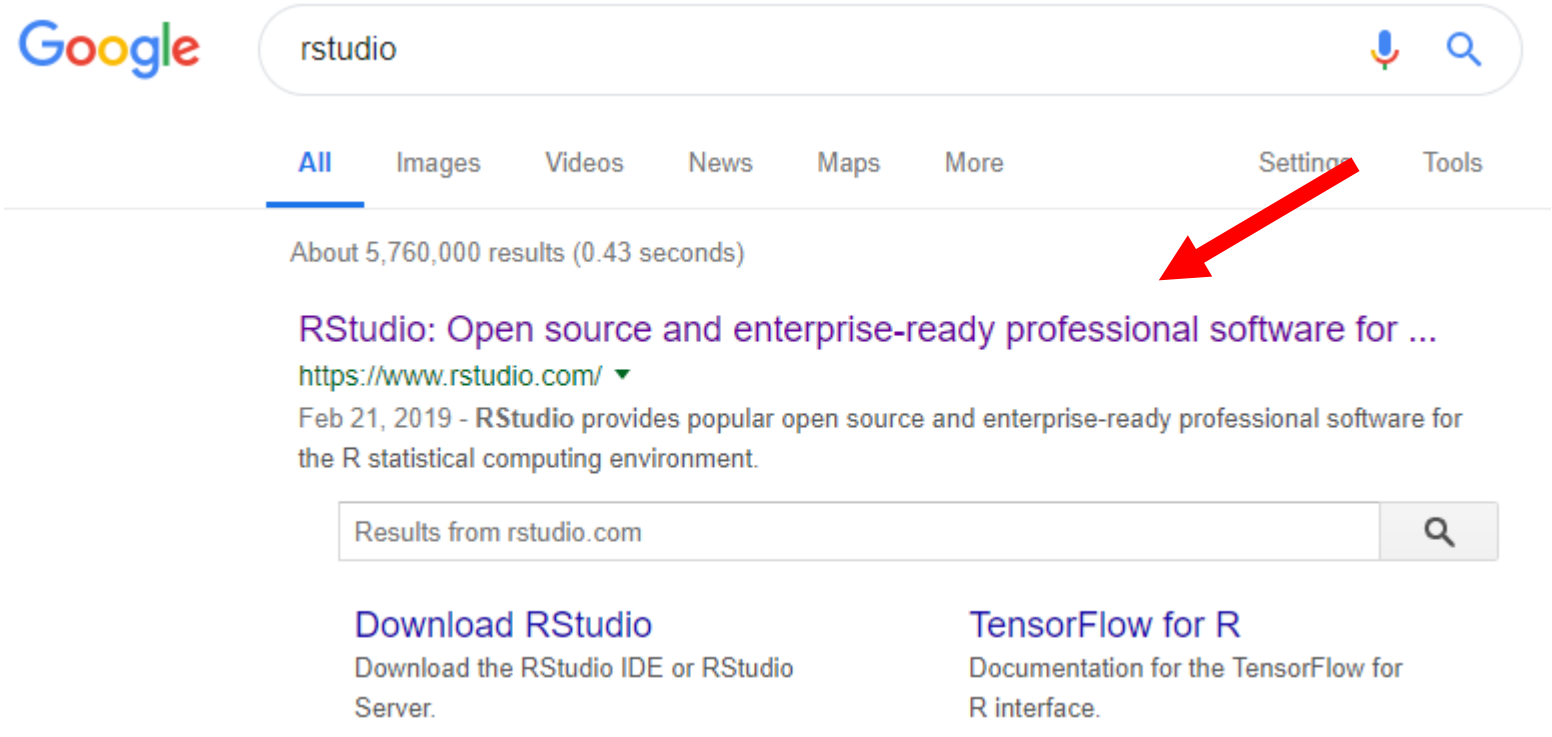
Installation and other instructions
New features in this version

Importante saber que….

- Al instalar R, es recomendable indicar que recuerde el número de versión
- Todos los años se lanza una nueva versión de R, normalmente en los meses de verano
- Para tener la última versión hay que volver a descargar e instalar R, al recordar el número de versión, el ordenador ignorará y eliminará las más antiguas

Para instalar RStudio es esencial haber instalado previamente R

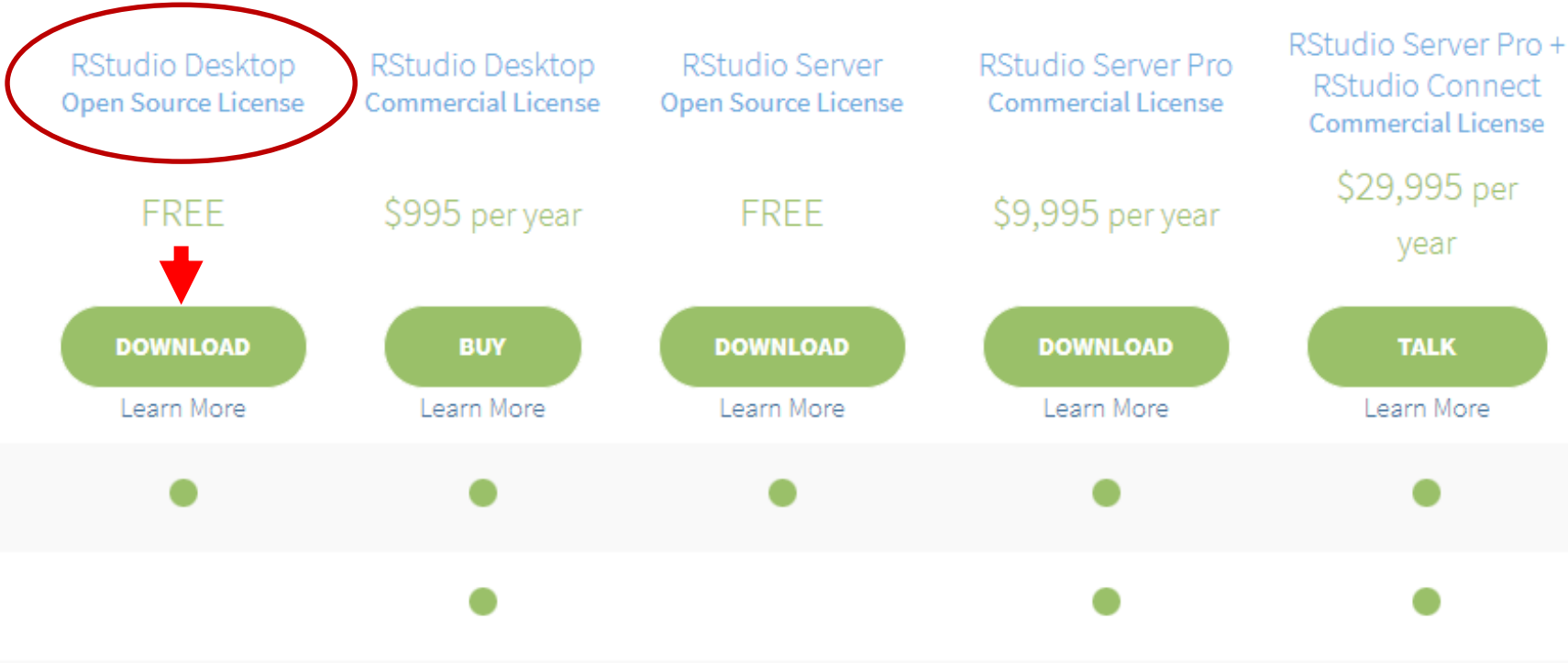
En primer lugar buscamos la web de RStudio desde nuestro buscador



RStudio es un software libre, aunque cuenta con una versión de pago, la versión libre cuenta con todas las características necesarias, la única diferencia es que no cuenta con servicio de mantenimiento.

AT YOUR OWN RISK!

Descarga la versión gratuita para escritorio de RStudio e instálala



Ejercicio 1. Conociendo las herramientas

— — —

- **Objetivo:** Familiarizarse con la interfaz de RStudio y realizar operaciones básicas en la consola de R.

- **Tarea:** Inicia un proyecto en RStudio llamado curso-iniciacion. Utilizando la consola de RStudio, deberás crear tres objetos: un vector de números, un vector de nombres de personas y un data frame.

- **Planteamiento:** Utiliza la función str() para cada uno de los tres objetos creados. ¿Para qué sirve esta función? ¿Qué información ofrece sobre cada uno de los tres objetos creados?

Ejercicio 2. Conociendo las herramientas

— — —

- **Objetivo:** Conocer los tipos de ficheros que crea RStudio

- **Tarea:** Basándote en el ejercicio anterior, crea un documento en R que pueda exportarse a formato html, Word o PDF en el que se explique cómo se ha creado cada objeto así como el análisis de la función str().

- **Planteamiento:** La utilidad de los documentos Rmarkdown radica en que nos permitirán guardar un *historial* de toda nuestra actividad, siendo capaces de integrar en un mismo software texto y comentarios, resultados y código.

Importación de datos

1. Importación de datos

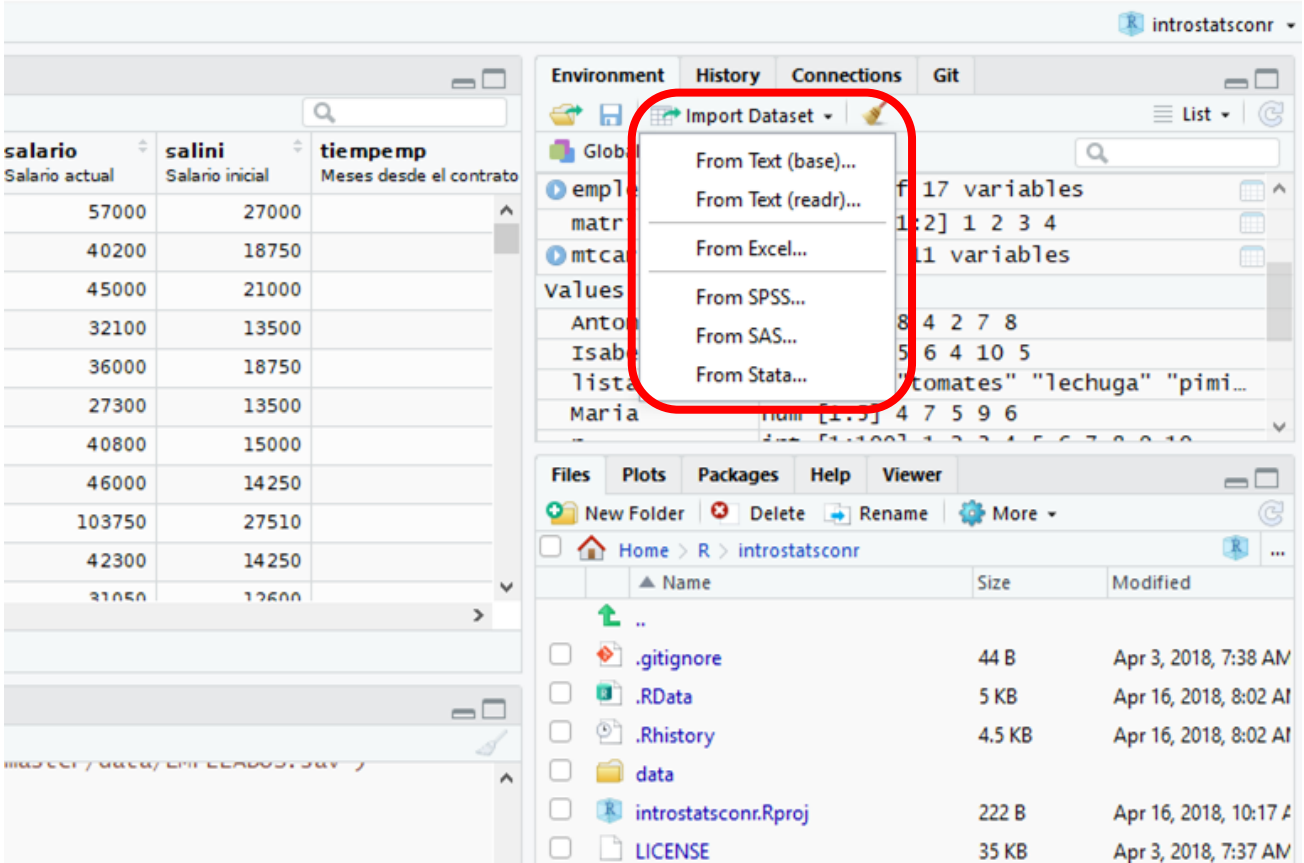
2. Comprobación de importación

3. Estructura de datos

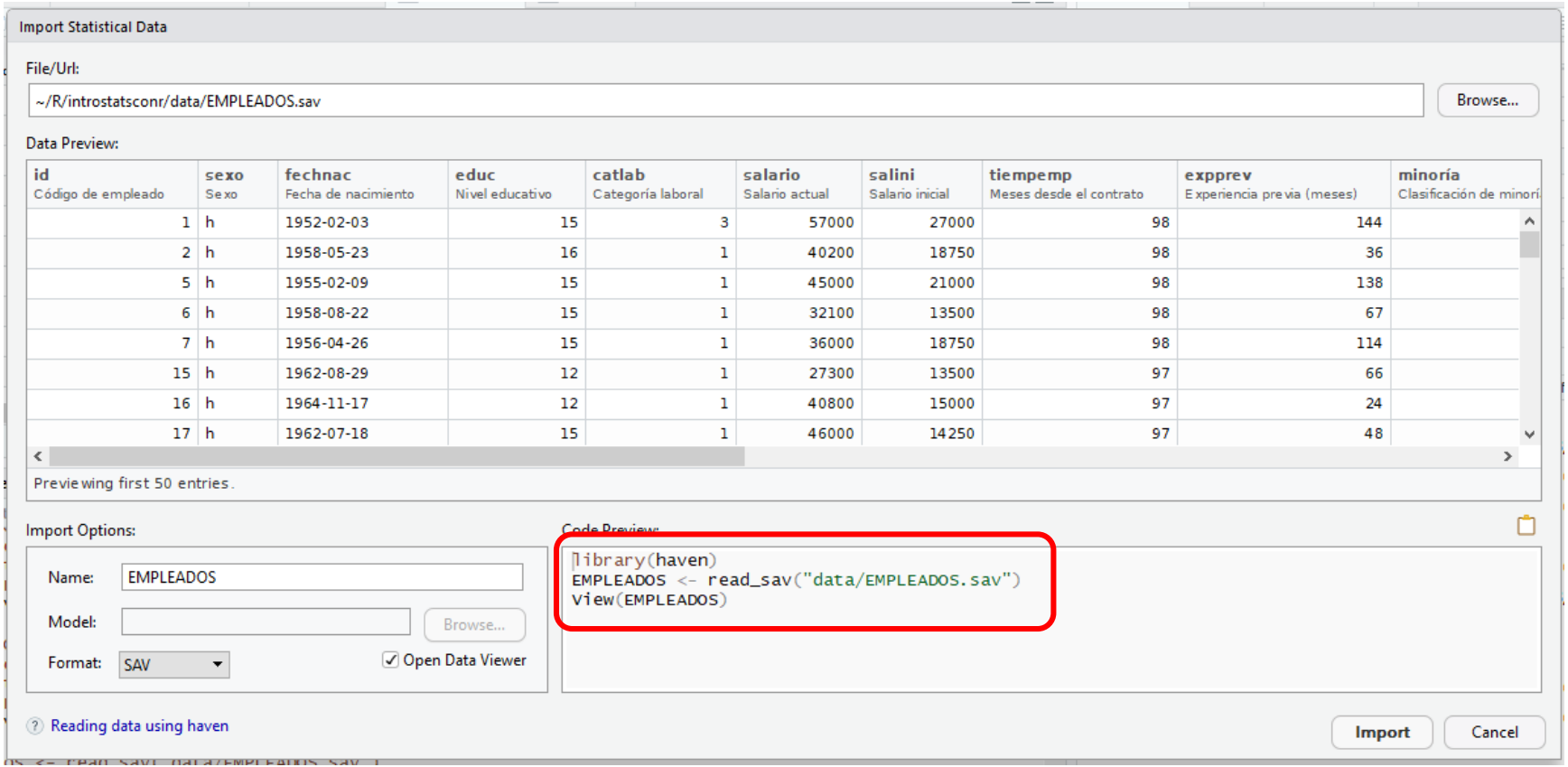
4. Exploración

PARTE I-4

Importación de set de datos



Importación de set de datos



Comandos básicos para explorar los datos

str()	• Muestra para cada variable el tipo de dato que contiene
summary()	• Indicadores descriptivos de cada variable (media, max., mín., etc.)
complete.cases()	• Indica para cada línea si hay datos incompletes o no
head()	• Muestra las seis primers líneas del set de datos
subset()	• Filtrado condicional del set de datos
nrow()	• Cuenta el número de líneas del set de datos
[,] $	• Filtrado de datos de columnas o líneas específicas

str()

```
> str(mtcars)
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

summary()

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat      
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930  
       wt             qsec             vs              am       
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000  
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062  
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000  
      gear            carb      
 Min.   :3.000   Min.   :1.000  
 1st Qu.:3.000   1st Qu.:2.000  
 Median :4.000   Median :2.000  
 Mean   :3.688   Mean   :2.812  
 3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :5.000   Max.   :8.000
```

complete.cases()

```
> complete.cases(mtcars)
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[31] TRUE TRUE
```

head()

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

subset()

subset(x, subset, select, …)

x =	Dataframe sobre el que se va a aplicar el filtro
subset =	Condición necesaria para filtrar
select =	Variables que queremos preservar

subset()

```
> subset(x = mtcars, subset = cyl==6, select = c("mpg", "cyl"))
                  mpg cyl
Mazda RX4        21.0   6
Mazda RX4 Wag    21.0   6
Hornet 4 Drive   21.4   6
Valiant          18.1   6
Merc 280         19.2   6
Merc 280C        17.8   6
Ferrari Dino     19.7   6
```

subset()

```
> subset(mtcars, mpg>20, c("mpg", "cyl"))
                  mpg cyl
Mazda RX4        21.0   6
Mazda RX4 Wag    21.0   6
Datsun 710       22.8   4
Hornet 4 Drive   21.4   6
Merc 240D        24.4   4
Merc 230         22.8   4
Fiat 128         32.4   4
Honda Civic      30.4   4
Toyota Corolla   33.9   4
Toyota Corona    21.5   4
Fiat X1-9        27.3   4
Porsche 914-2    26.0   4
Lotus Europa     30.4   4
Volvo 142E       21.4   4
```

subset()

```
> subset(mtcars, mpg>10 & mpg<20)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood 10.4  8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

nrow()

```
> nrow(mtcars)
[1] 32
> ncol(mtcars)
[1] 11
> length(mtcars)
[1] 11
```

[,] $

[filas,columnas]

```
> mtcars[1:4,]
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4  6  258 110 3.08 3.215 19.44  1  0    3    1
> mtcars[,1:4]
                   mpg cyl  disp  hp
Mazda RX4         21.0   6 160.0 110
Mazda RX4 Wag     21.0   6 160.0 110
Datsun 710        22.8   4 108.0  93
Hornet 4 Drive    21.4   6 258.0 110
Hornet Sportabout 18.7   8 360.0 175
Valiant           18.1   6 225.0 105
Duster 360        14.3   8 360.0 245
Merc 240D         24.4   4 146.7  62
Merc 230          22.8   4 140.8  95
Merc 280          19.2   6 167.6 123
Merc 280C         17.8   6 167.6 123
Merc 450SE        16.4   8 275.8 180
Merc 450SL        17.3   8 275.8 180
```

[,] $

dataframe$variable

```
> mtcars$cyl
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Explorando los datos con visualizaciones

plot()

- Gráfico de dispersion o lineal

hist()

- Histograma

barplot()

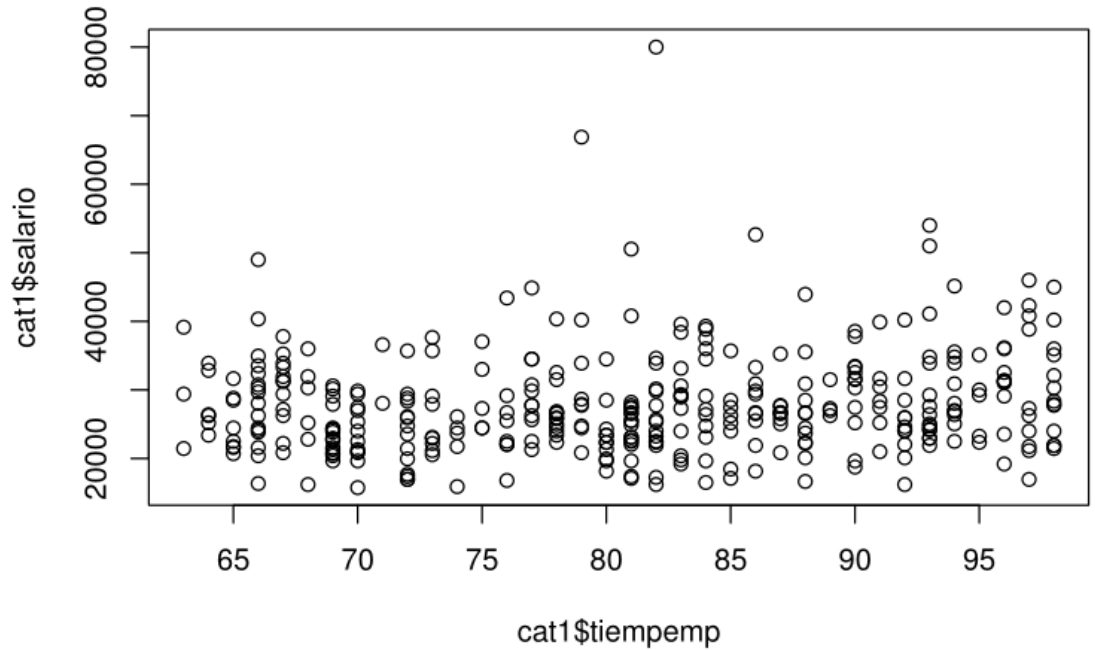
- Gráfico de barras

boxplot()

- Gráfico de caja y bigotes

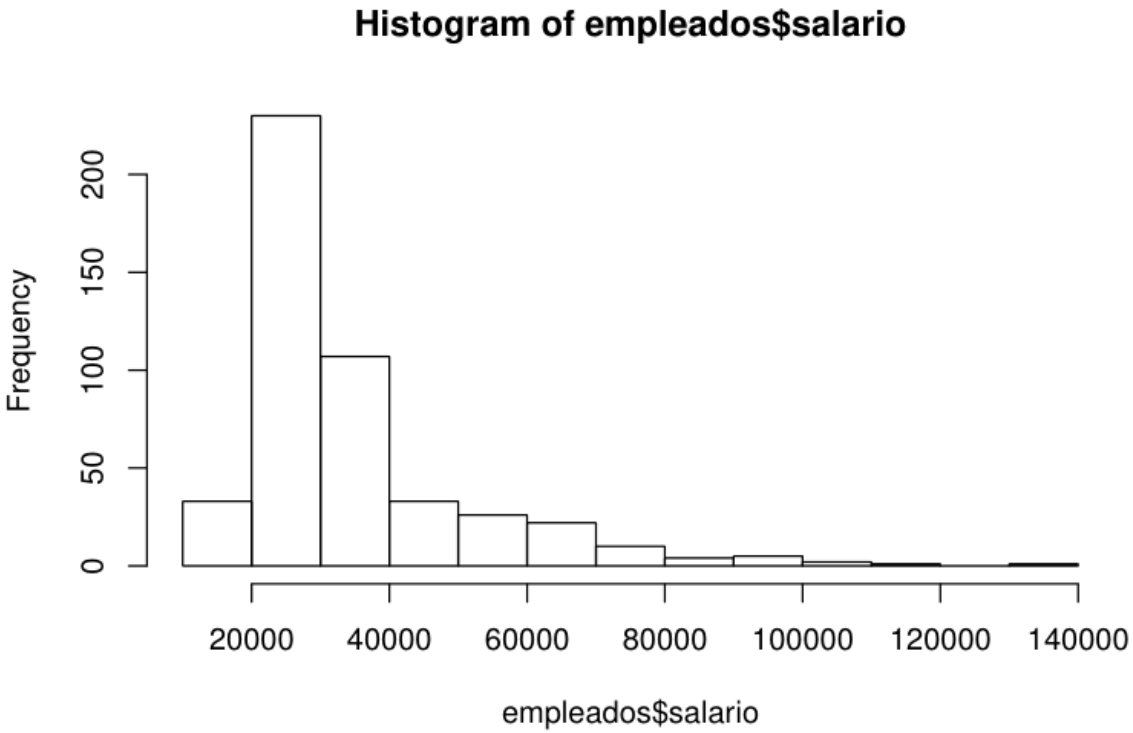
plot()

```
cat1 <- subset(empleados, catlab==1)
plot(cat1$tiempemp, cat1$salario)
```



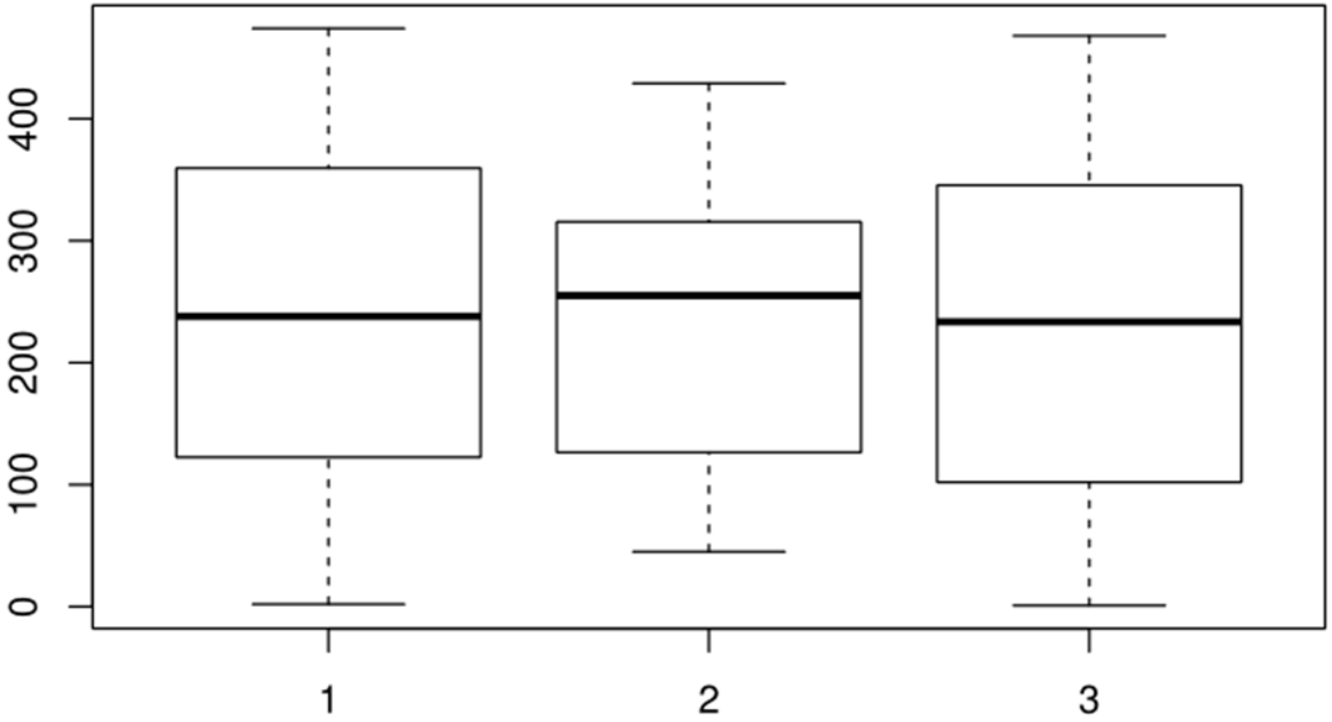
hist()

```
hist(empleados$salario)
```



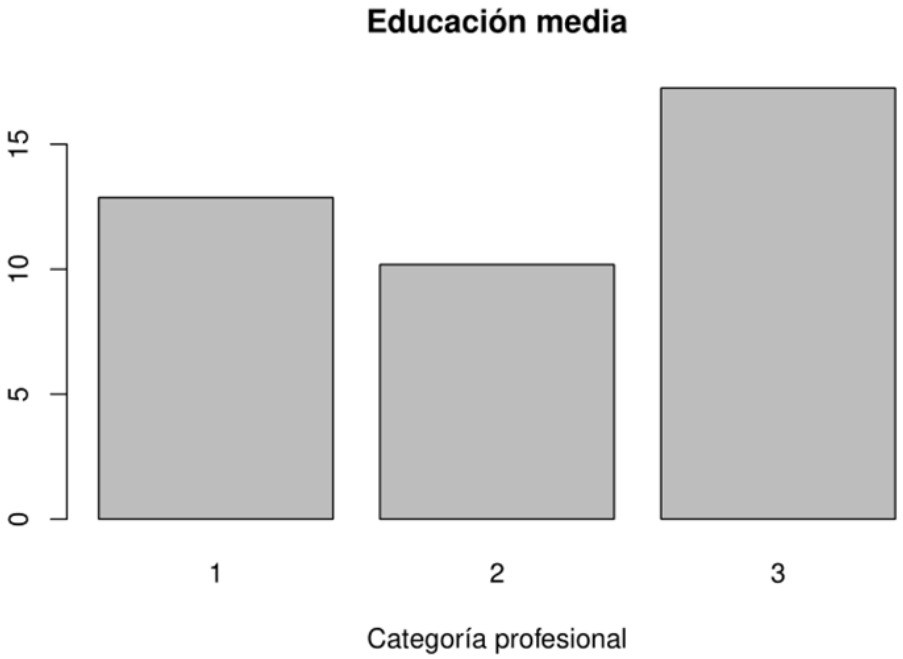
boxplot()

```
boxplot(id~catlab, data = empleados)
```



barplot()

```r
mean.educ <- aggregate(educ~catlab, data = empleados, mean)
mean.educ <- mean.educ[,2]
names(mean.educ) <- c("1", "2", "3")
barplot(mean.educ, main="Educación media",
        xlab = "Categoría profesional")
```



Ejercicio 3. Explorando datos

— — —

- **Objetivo:** Comienza a emplear las funciones básicas de R en un set de datos. Crea un documento donde toda la información esté disponible.

- **Tarea:** Utiliza el set de datos incluido por defecto en R llamado *USArrests,* aplica las funciones básicas tanto de exploración como de visualización para conocer el dataset.

- **Planteamiento:** Es evidente que no conseguiremos asimilar todas las funciones de R, pero combinando la información ofrecida aquí, la ayuda de R y ayuda en Internet, deberemos ser resolutivos y ser capaces de explorar de manera básica un set de datos.

Parte 2

- El uso de paquetes y librerías
- Intro al paquete *ggplot2*

- Análisis descriptivo y visualización
- Puesta en común

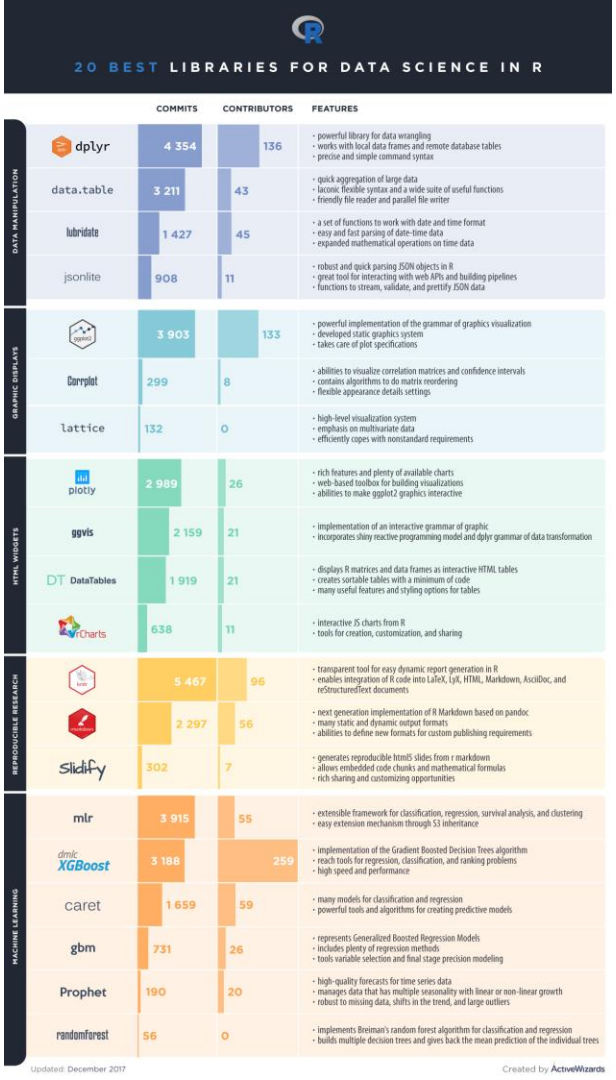
Paquetes y librerías

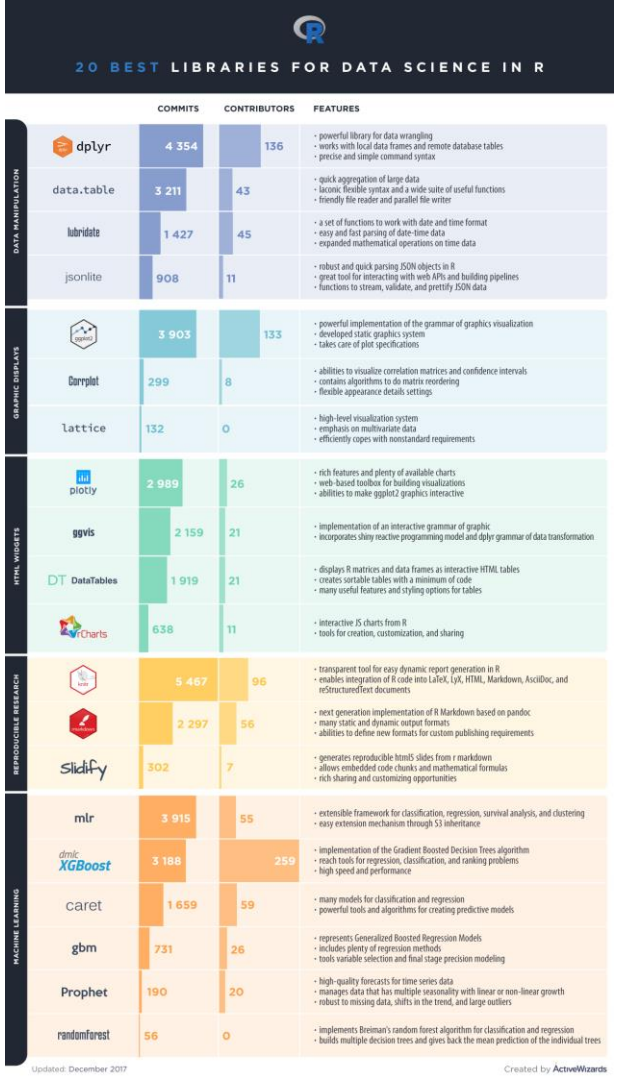
1. Instalación de paquetes

2. Dónde encontrarlos

3. Paquetes recomendados

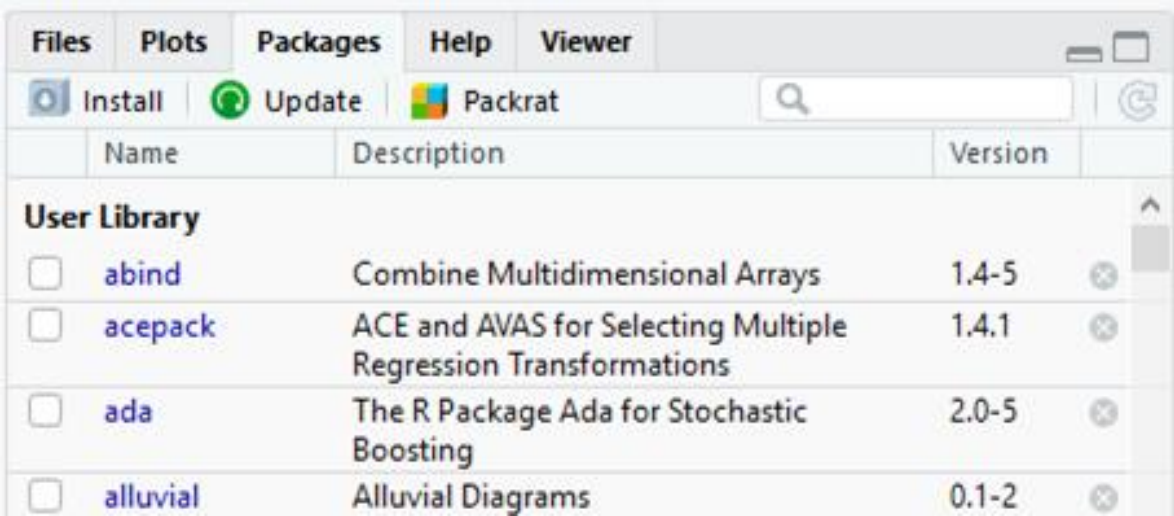
PARTE 2-I



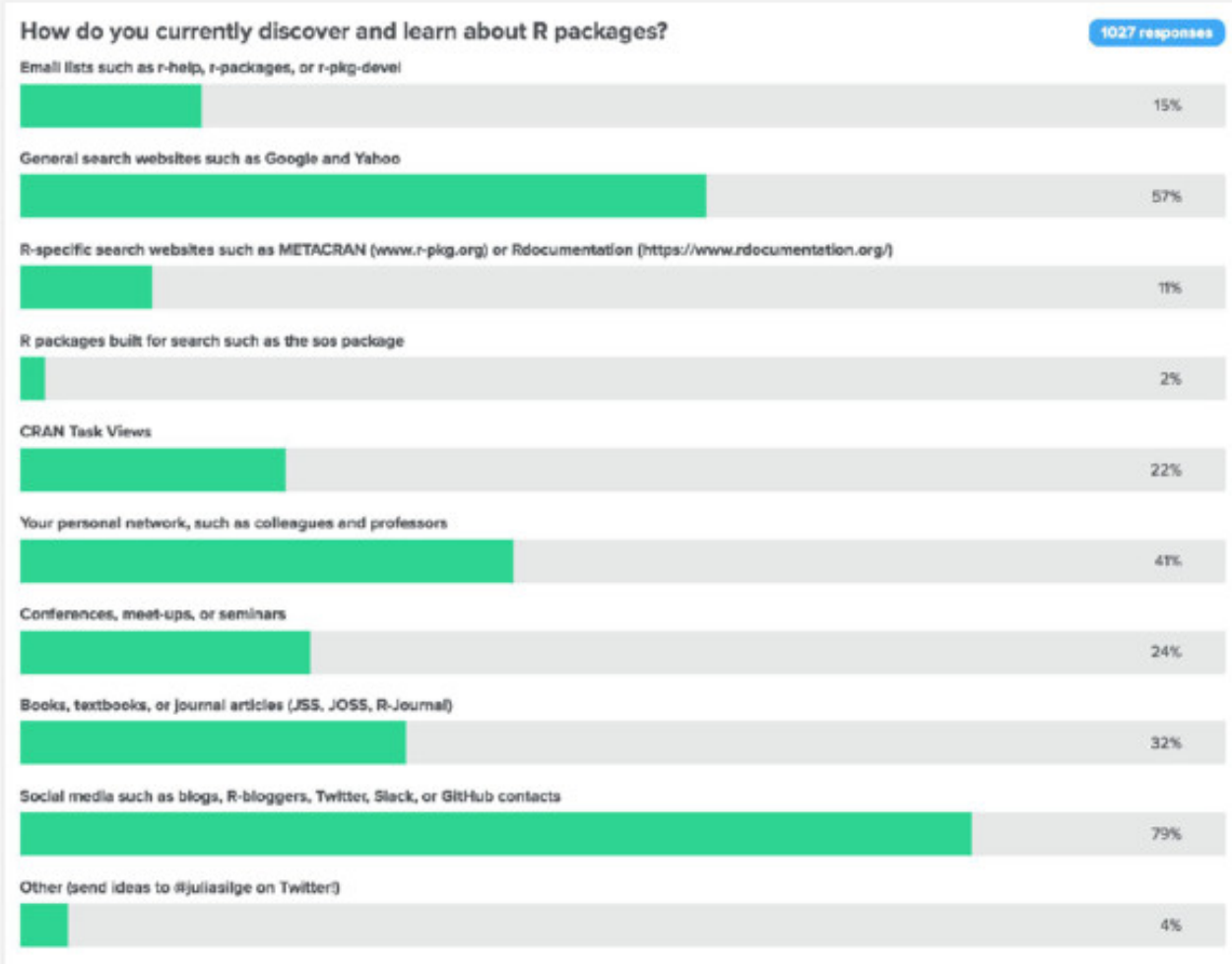


Acceso e instalación de paquetes

- Software descentralizado
- La comunidad crea sus propios paquetes que se van adhiriendo de manera modular
- Cada paquete tiene su propia versión y debe actualizarse



R Package Discovery



Acceso e instalación de paquetes

Normalmente, los paquetes finalizados se suben a **CRAN - https://cran.r-project.org**

Se pueden descargar directamente desde R con el siguiente comando:

```
>install.packages("nombre_del_paquete")
```

Acceso e instalación de paquetes

Sin embargo, muchas veces los paquetes están aún en desarrollo o estamos interesados en acceder a la última versión no testeada.

En muchas ocasiones podemos encontrar paquetes en la plataforma **GitHub**

```
>devtools::install_github("usuario/paquete")
```

Algunos recursos de interés

Listado de paquetes creados por Hadley Wickham, el gurú detrás de RStudio. Incluye paquetes de procesamiento y limpieza de datos, visualización, etc.

Tidyverse

Packages Articles Learn Help Contribute

R packages for data science

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

Algunos recursos de interés

Listado de paquetes mantenidos por la comunidad científica. Desde extracción de datos, procesamiento del lenguaje natural, visualización o de publicación de datos científicos

Algunos recursos de interés

Familia de paquetes de R desarrollados por el Instituto de Estudios Cuantativos de las Ciencias Sociales de la Universidad de Harvard.

Zelig Project

Zelig is an easy-to-use, free, open source, general purpose statistics program for estimating, interpreting, and presenting results from any statistical method. Zelig turns the power of R, with thousands of open source packages — but with free ranging syntax, diverse examples, and documentation written for different audiences — into the same three commands and consistent documentation for every method. Zelig uses R code from many researchers, making it "everyone's statistical software." We hope it becomes everyone's statistical software for applications too, as we designed it so anyone can use it or add their methods to it. We aim for Zelig to be the best way to do analysis, prepare replication files, learn new methods, or teach.

Ejercicio 4. Instalación de librerías

— — —

- **Objetivo:** Comenzar a utilizar funciones de una librería externa a R

- **Tarea:** Instala la librería de análisis de datos *zeligverse.* Una vez instalada, deberás activarla para comenzar a utilizar sus funciones con el comando library()

- **Planteamiento:** Crea un documento Rmarkdown en el que documentes el proceso de instalación. Entra en la web del proyecto Zelig y reproduce el primer ejemplo que muestran.

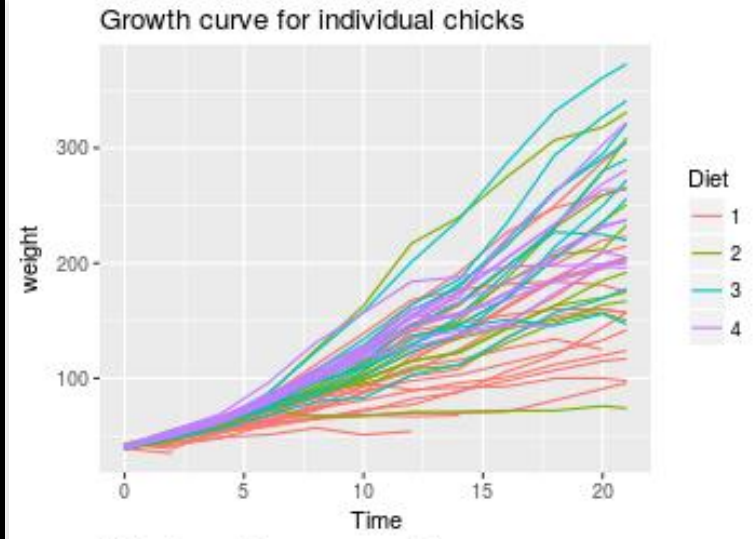
GGPLOT2

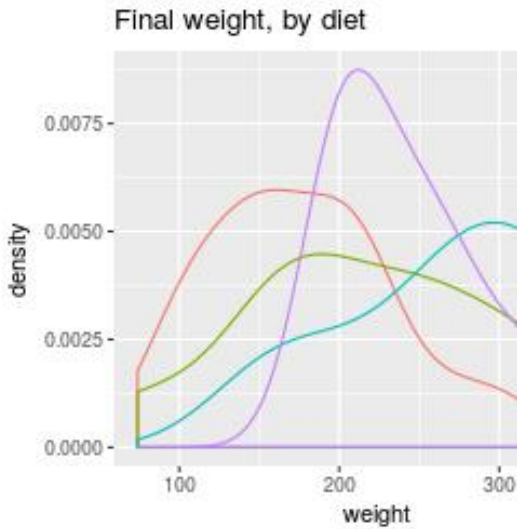
1. Gramática de ggplot2

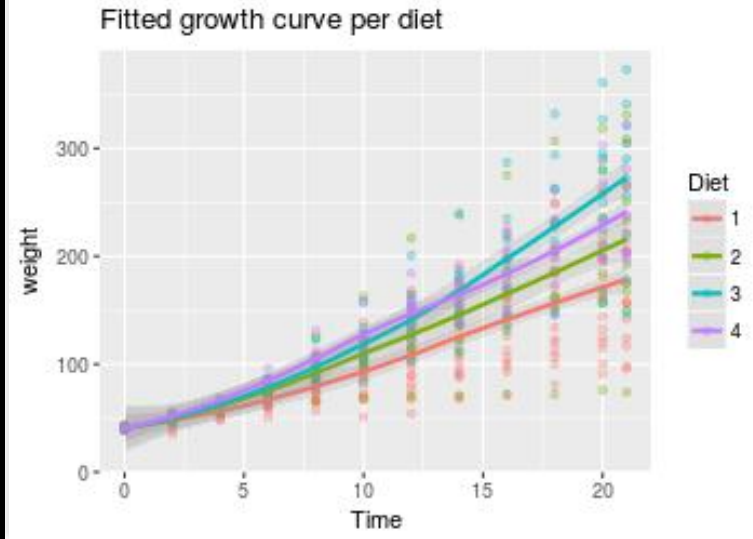
2. Tipos de gráficos

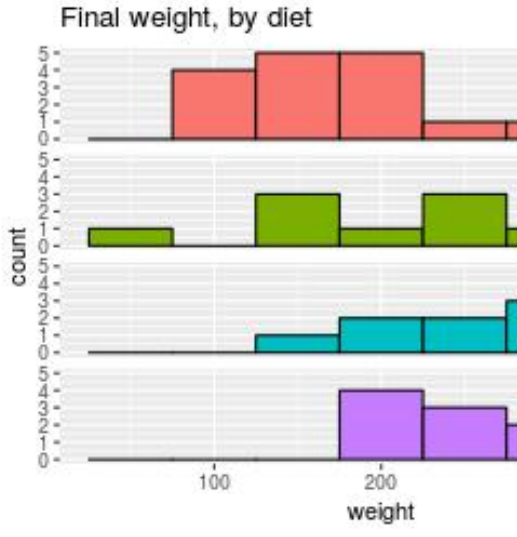
3. Personalización de etiquetas y temas

PARTE 2-2

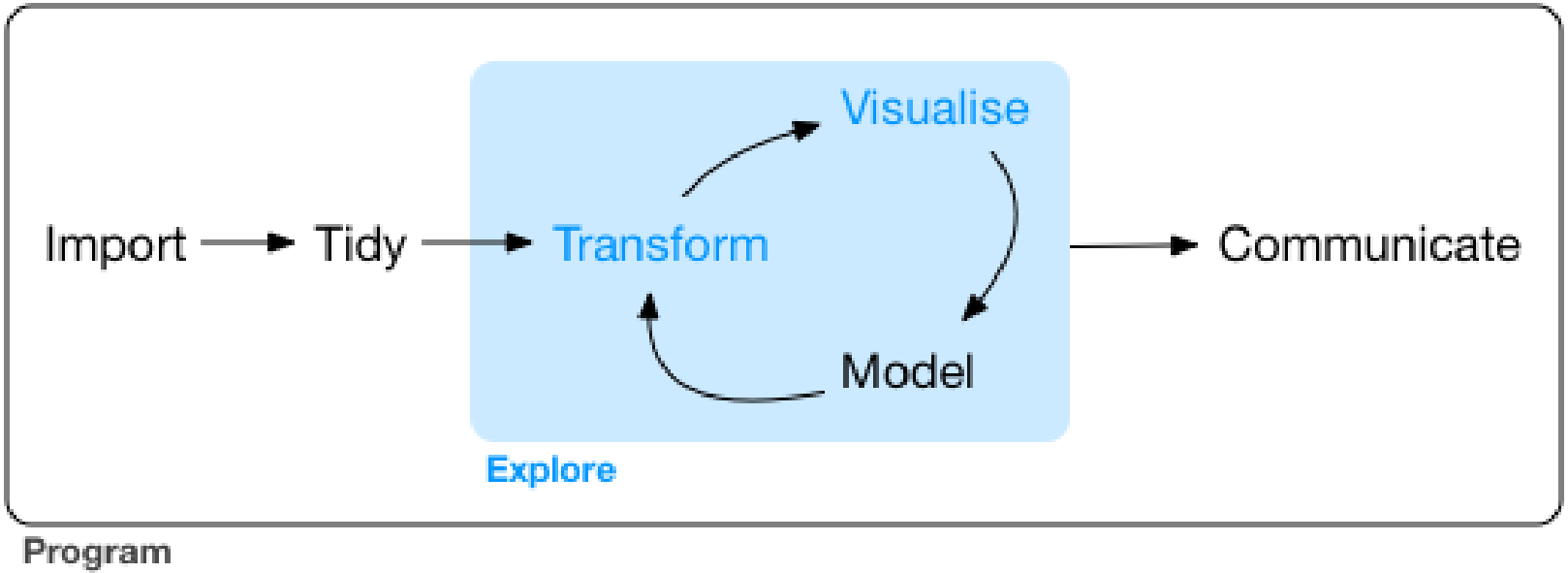






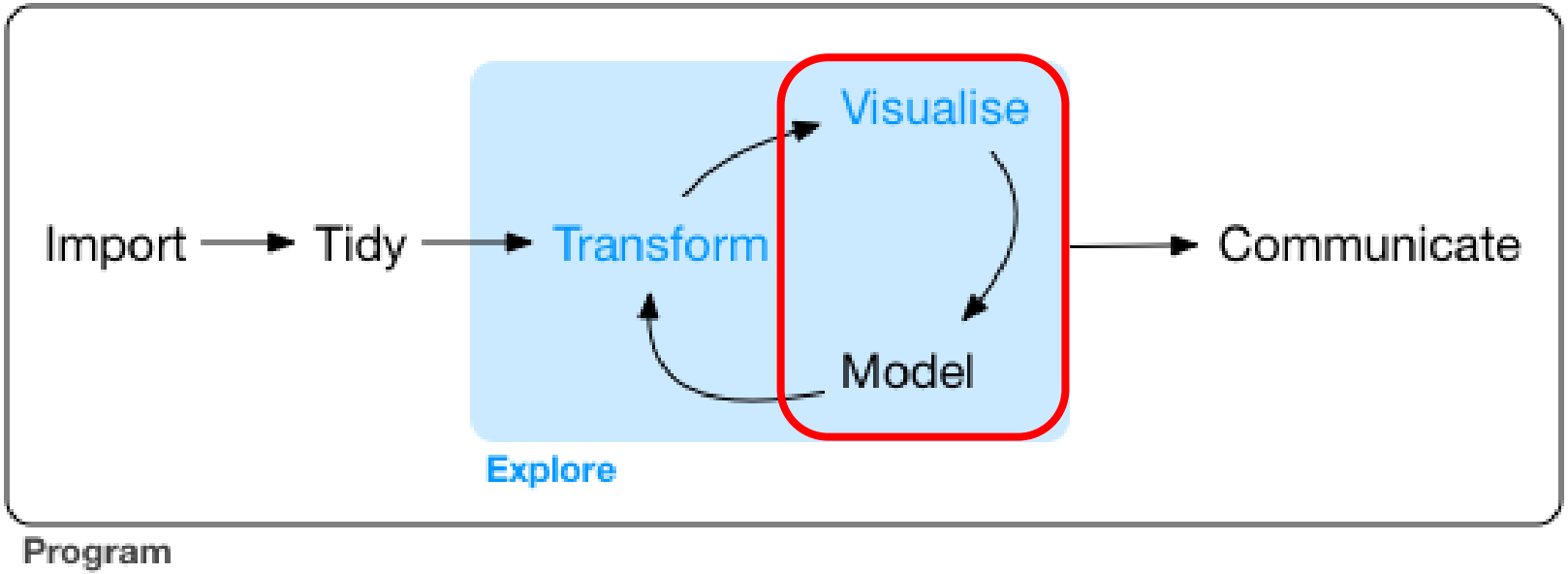


Visualización de datos



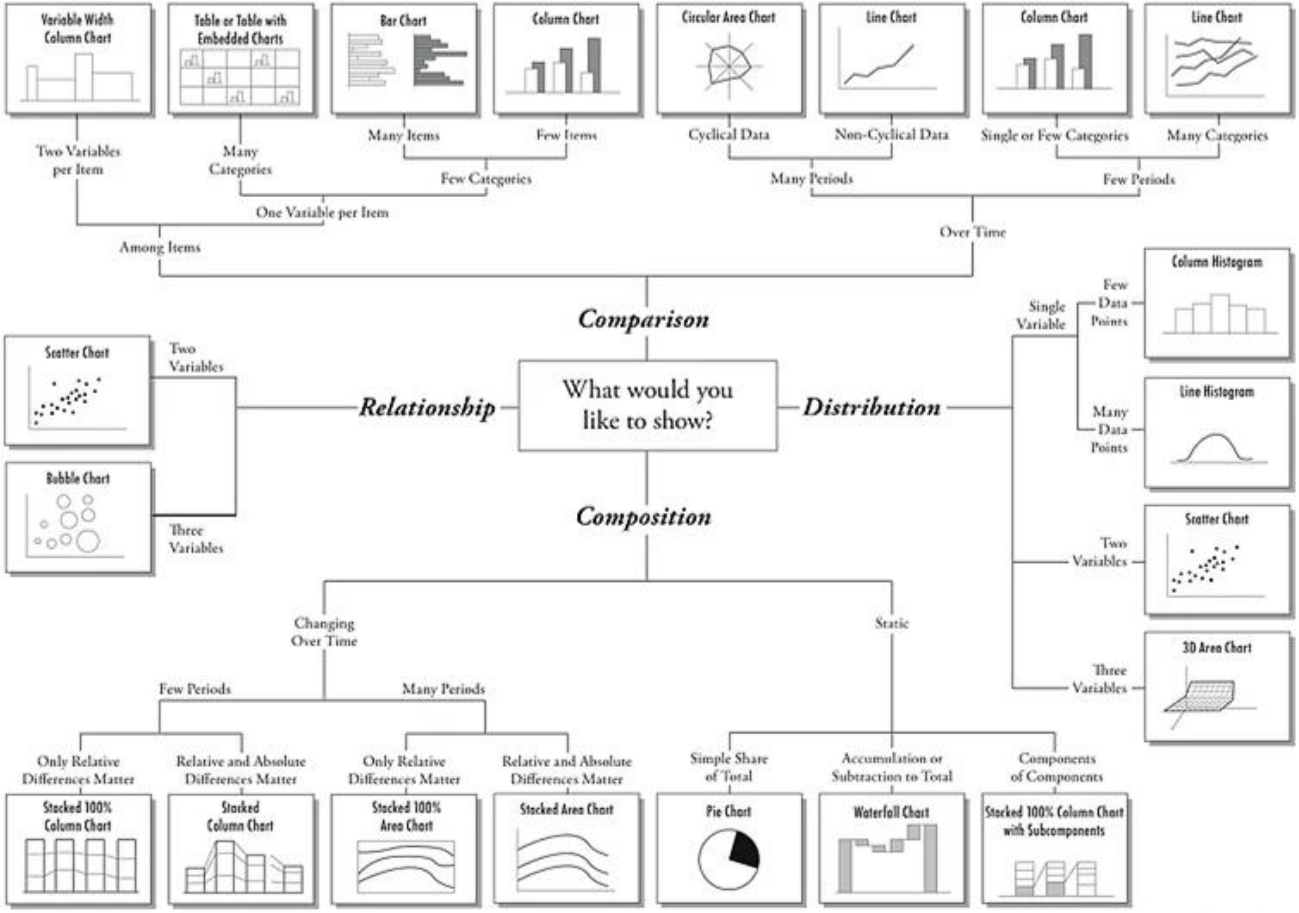
http://r4ds.had.co.nz/explore-intro.html

Visualización de datos



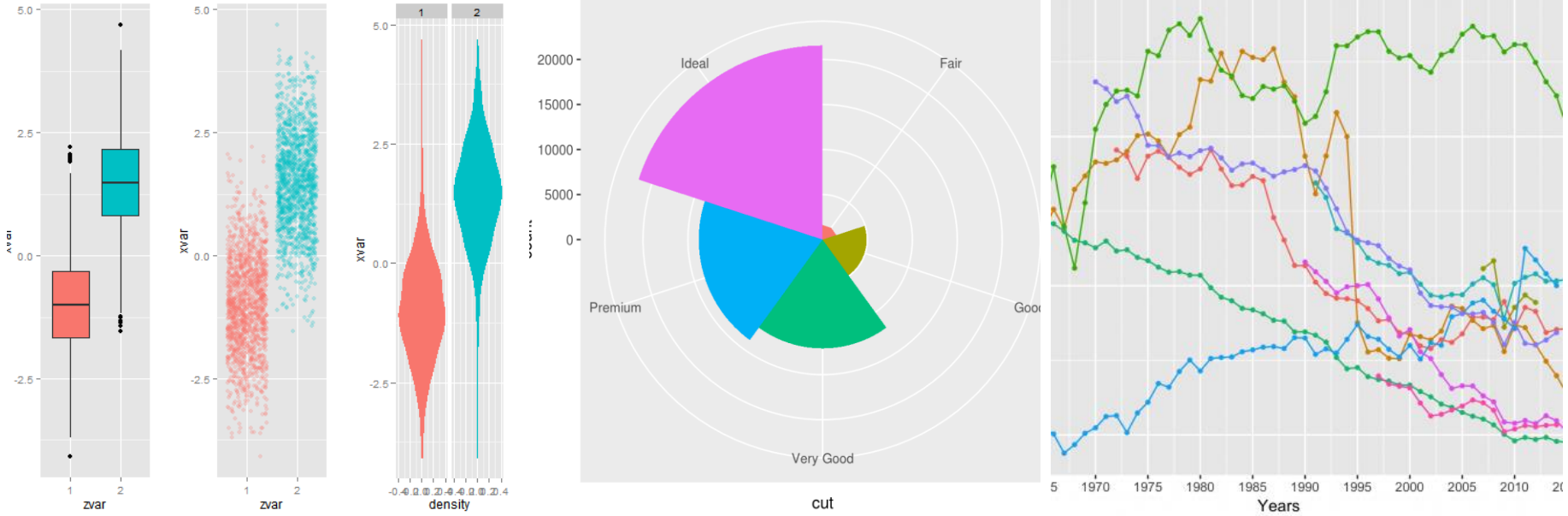
http://r4ds.had.co.nz/explore-intro.html

Chart Suggestions—A Thought-Starter



Visualización de datos - {ggplot2}

- El paquete de gráficos más extendido y empleado actualmente
- Su creador, Hadley Wickham, CEO de RStudio es autor de los paquetes más populares en la actualidad
- Numerosos paquetes de visualización están basados a su vez en ***ggplot2***
- Se caracteriza por emplear la *gramática de gráficos*, una forma intuitiva de crear gráficas

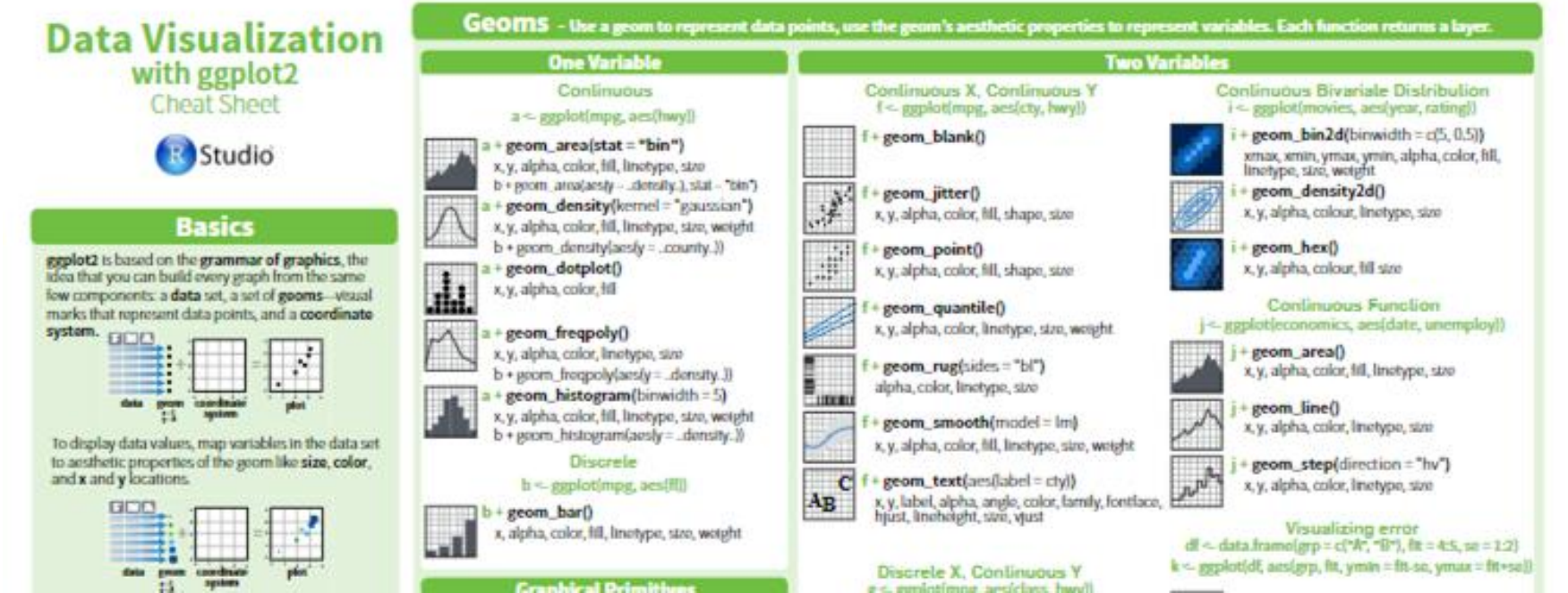


Visualización de datos - {ggplot2}

Algunos ejemplos

Visualización de datos - {ggplot2}

[Descárgala aquí](#) Cheat sheet



Data Visualization with ggplot2 Cheat Sheet

RStudio

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same few components: a data set, a set of geoms—visual marks that represent data points, and a coordinate system.

To display data values, map variables in the data set to aesthetic properties of the geom like size, color, and x and y locations.

Geoms – Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + geom_dotplot()
x, y, alpha, color, fill

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + geom_bar()
x, alpha, color, fill, linetype, size, weight

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + geom_blank()

f + geom_jitter()
x, y, alpha, color, fill, shape, size

f + geom_point()
x, y, alpha, color, fill, shape, size

f + geom_quantile()
x, y, alpha, color, linetype, size, weight

f + geom_rug(sides = "bl")
alpha, color, linetype, size

f + geom_smooth(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + geom_bin2d(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + geom_area()
x, y, alpha, color, fill, linetype, size

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)

k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

Discrete X, Continuous Y

e <- ggplot(mpg, aes(class, hwy))

Visualización de datos - {ggplot2}

- Estructura del código por capas

```
>ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))
```

Visualización de datos - {ggplot2}

- Estructura del código por capas

```
>ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))
```

➔ **Primera capa:** Sistema de coordenadas y definición de set de datos

Visualización de datos - {ggplot2}

- Estructura del código por capas

```
>ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))
```

➔ **Primera capa:** Sistema de coordenadas y definición de set de datos

➔ **Segunda capa:** Tipo de gráfico y variables a mostrar

Visualización de datos - {ggplot2}

- Estructura del código por capas

```
>ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))
```

Para añadir nuevas capas

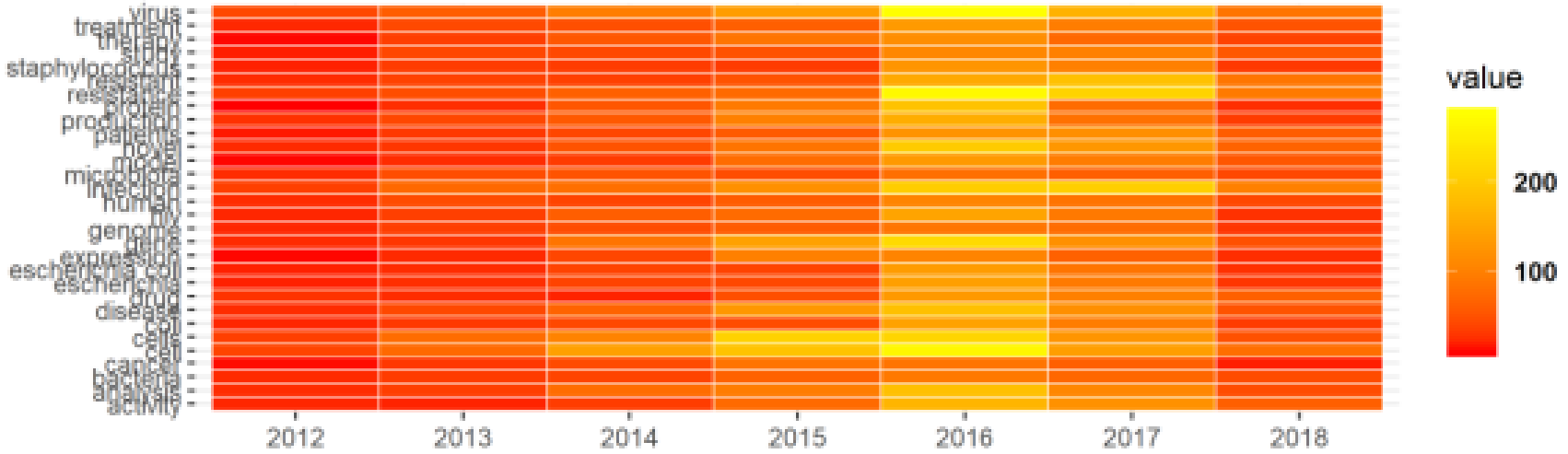
➔ **Primera capa:** Sistema de coordenadas y definición de set de datos

➔ **Segunda capa:** Tipo de gráfico y variables a mostrar

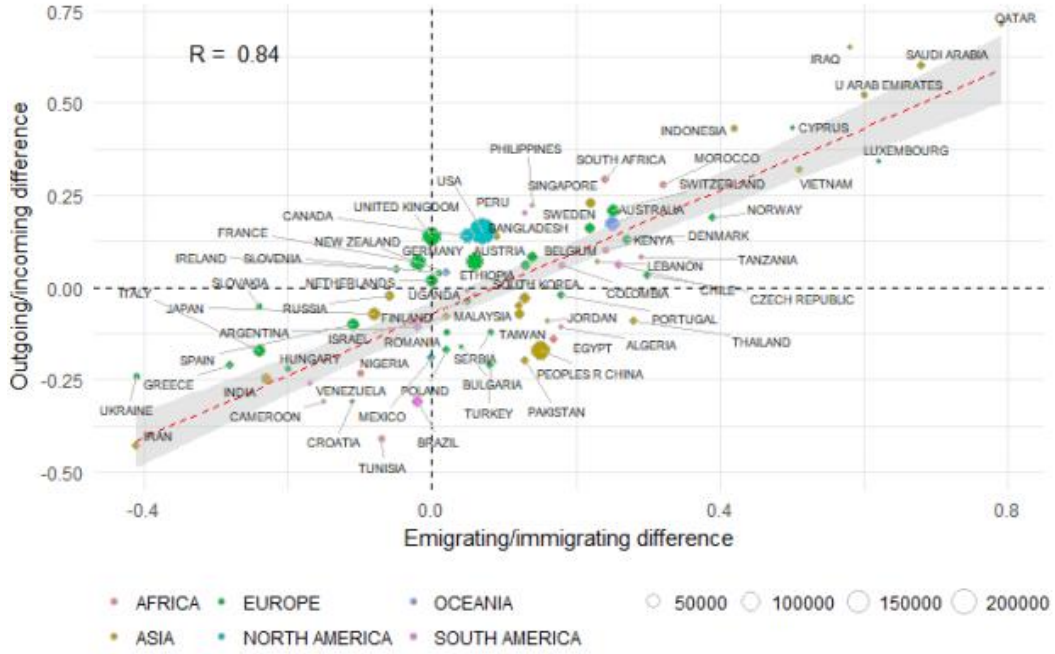
Tipos de gráficos – {ggplot2}

https://ggplot2.tidyverse.org/reference/

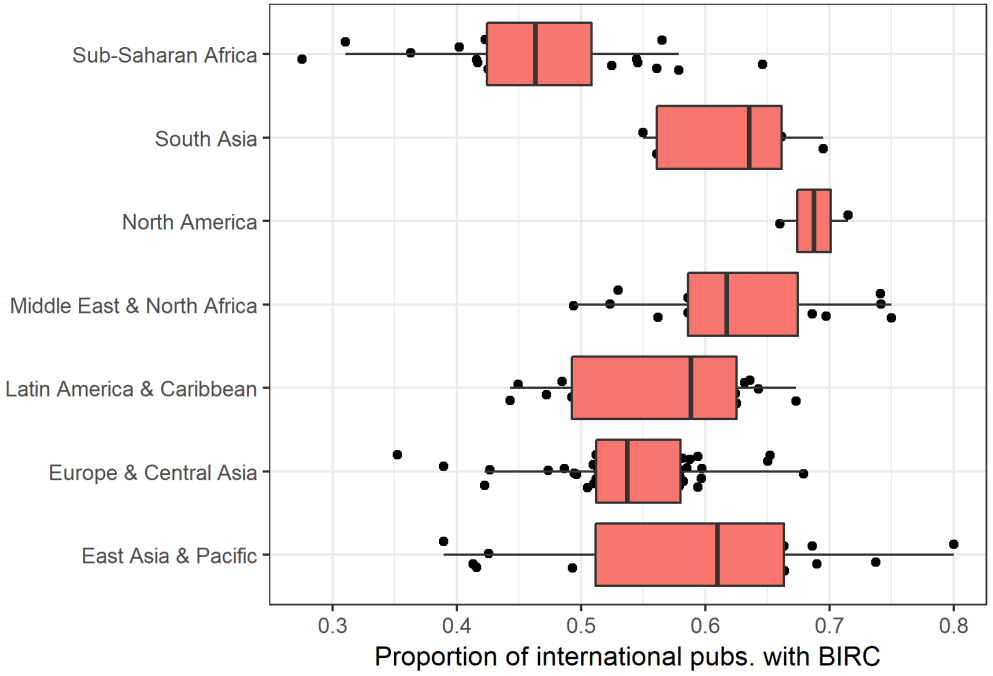
Algunos ejemplos de elaboración propia



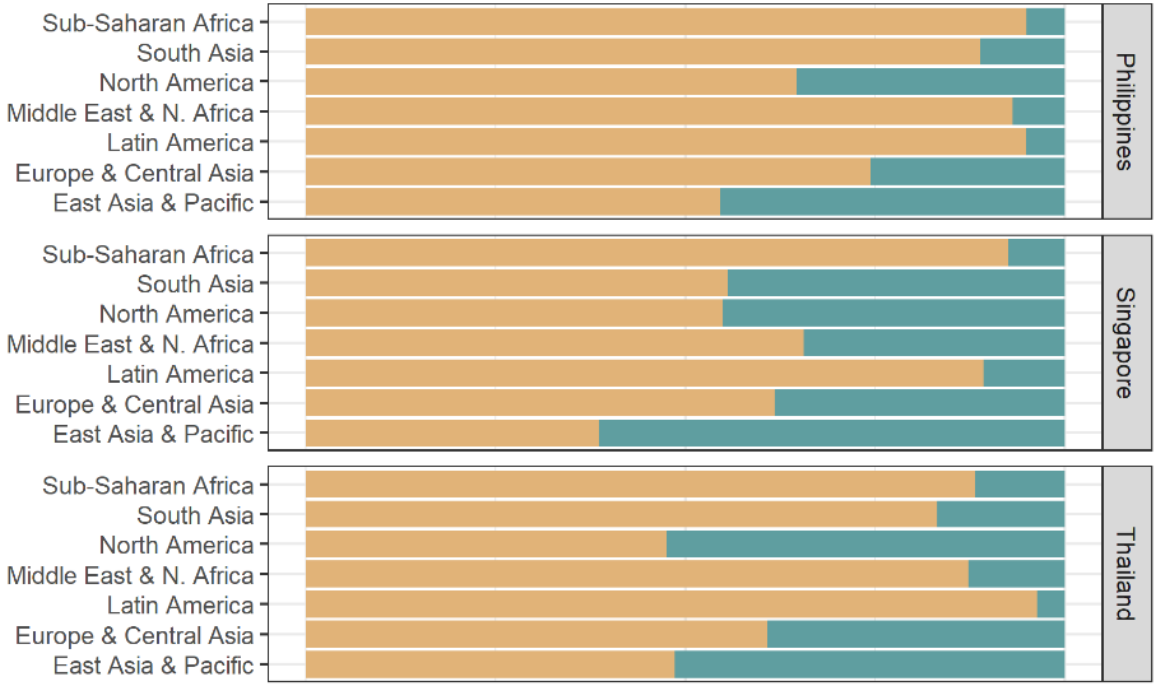
Algunos ejemplos de elaboración propia



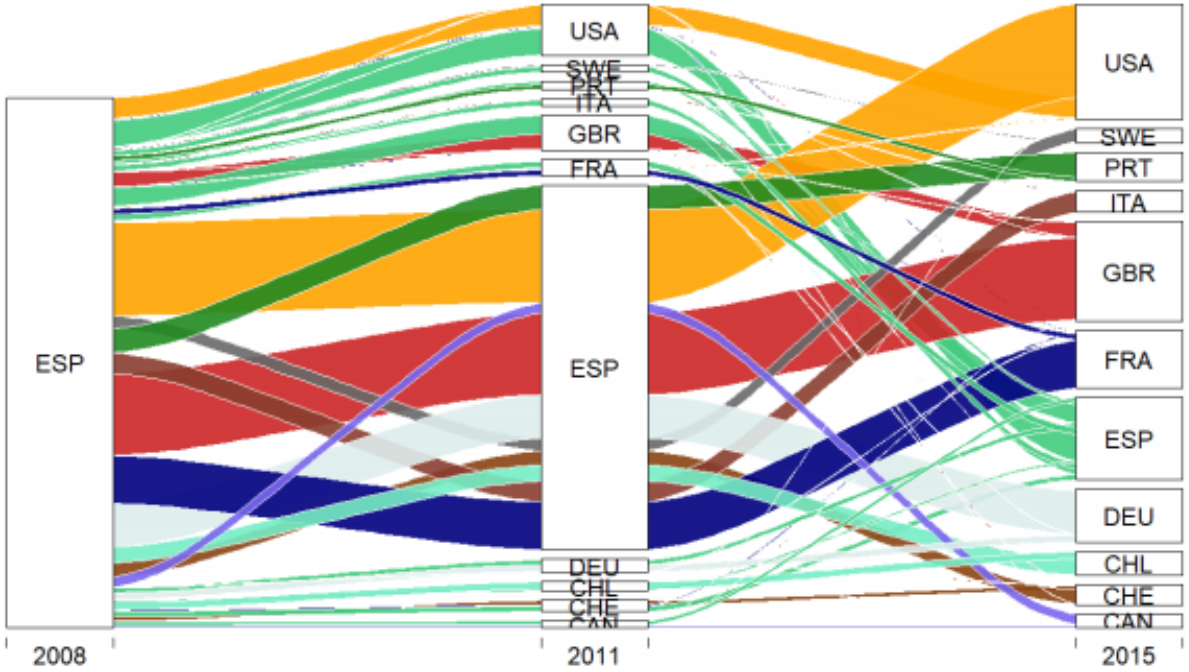
Algunos ejemplos de elaboración propia



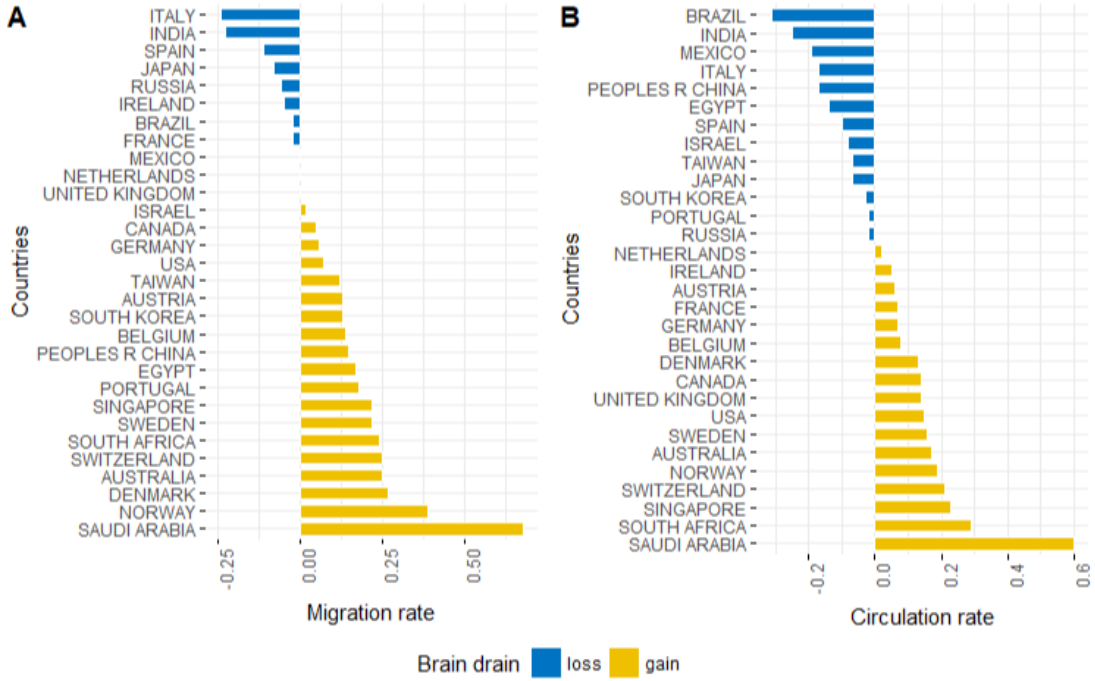
Algunos ejemplos de elaboración propia



En combinación con otros paquetes

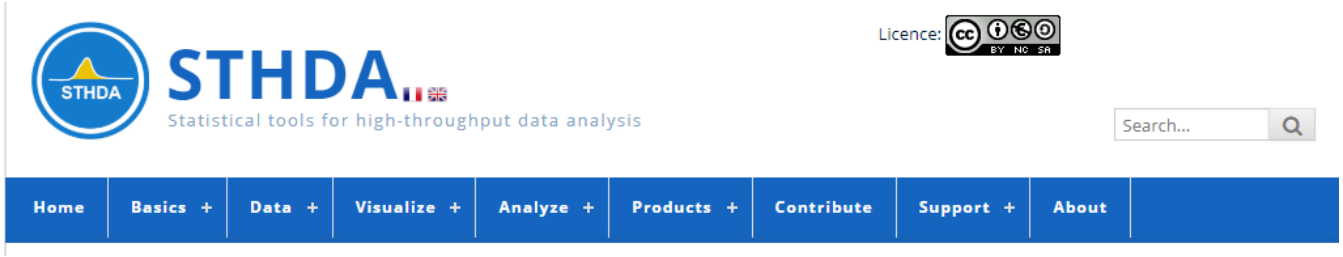


En combinación con otros paquetes

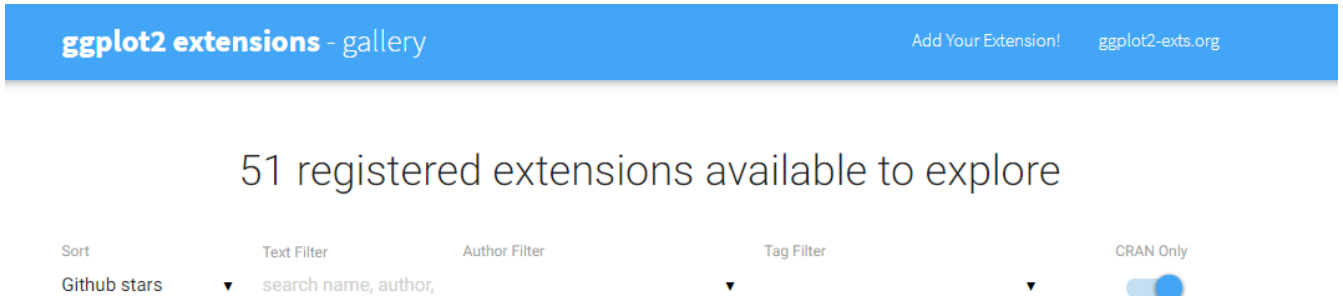


Recursos adicionales

- STHDA – {ggpubr} - http://www.sthda.com/english/



- ggplot2 extensions gallery - http://www.ggplot2-exts.org/gallery/



Ejercicio 5. Gráficos con ggplot2

— — —

- **Objetivo:** Familiarizarse con la librería *ggplot2*

- **Tarea:** Instala la librería de visualización de datos *ggplot2.* Una vez instalada, deberás activarla para comenzar a utilizar sus funciones con el comando library(). Utilizando el set de datos USArrests, comienza a explorar gráficamente los datos

- **Planteamiento:** Crea un documento Rmarkdown en el que documentes las distintas gráficas que crees. Deberás crear un gráfico de dispersión, un gráfico de caja y bigotes y un gráfico de barras.

Material extra

Análisis estadísticos - Test de normalidad

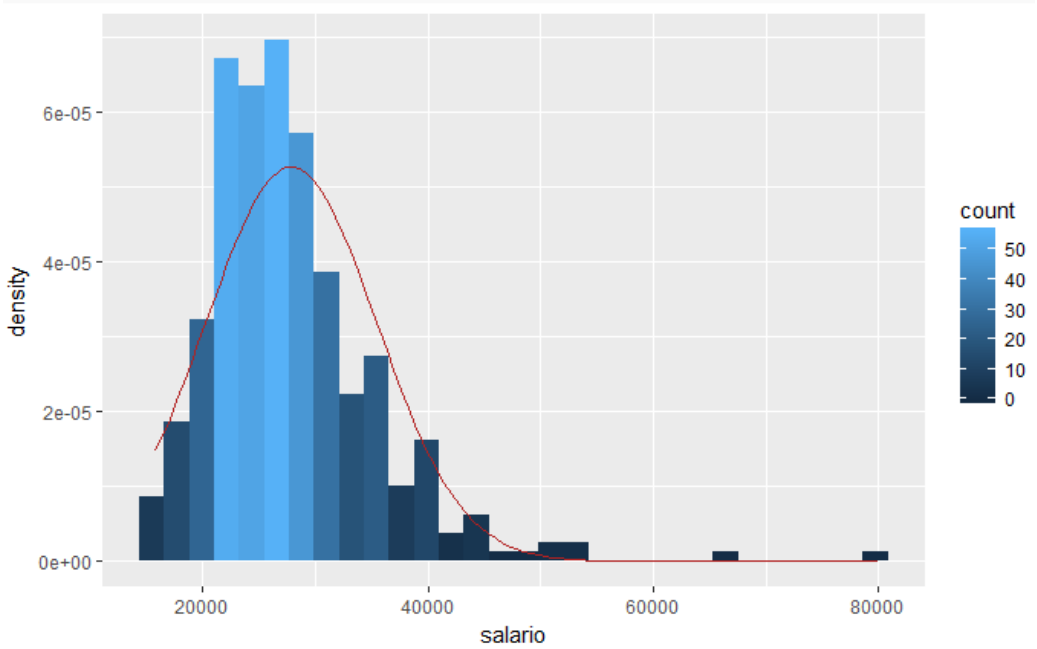
¿Cobran todos los técnicos de la empresa X un salario similar?

```r
aggregate(empleados2$salario, list(empleados2$catlab), FUN= mean)
```

Group.1 <fctr>	x <dbl>
1	27838.54
2	30938.89
3	63977.80

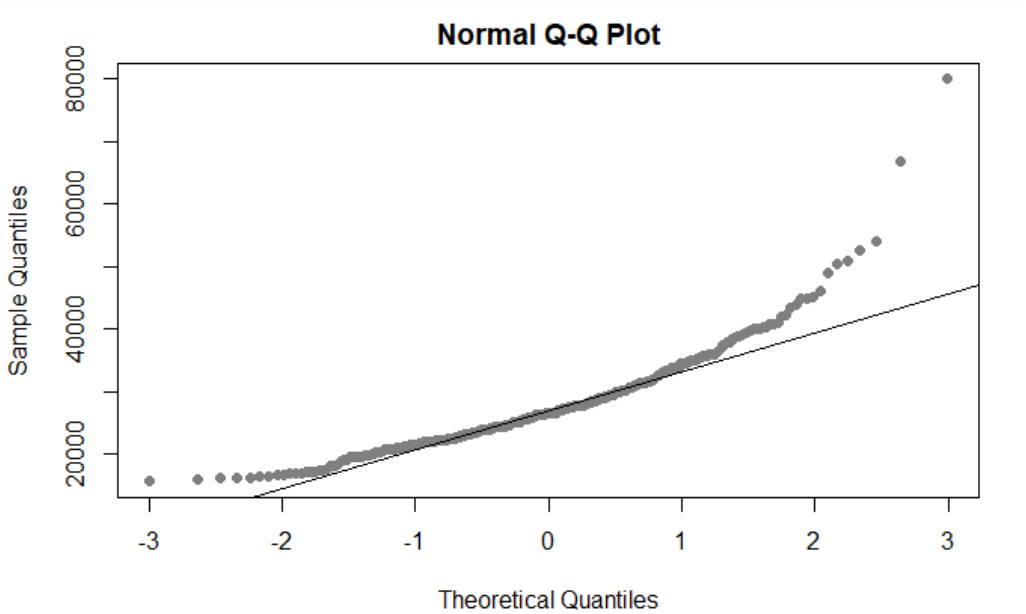
Análisis estadísticos - Test de normalidad

El histograma nos permite ver rápidamente si la distribución es normal



Análisis estadísticos - Test de normalidad

Otra opción es comparar la distribución por cuantiles reales con los teóricos



Análisis estadísticos - Test de normalidad

Test de Shapiro-Wilk de normalidad

> shapiro.test()

```
        Shapiro-Wilk normality test

data:  tecnicos$salario
W = 0.88207, p-value = 4.568e-16
```

Análisis estadísticos - Contraste de hipótesis

Comparación del salario de directivos y técnicos (catlab= 3 y 1)

> t.test()

```
        Welch Two Sample t-test

data:  dire$salario and tec$salario
t = 17.803, df = 89.708, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 32106.31 40172.21
sample estimates:
mean of x mean of y
 63977.80  27838.54
```

Análisis estadísticos - ANOVA

Comparación de salarios entre las tres categorías

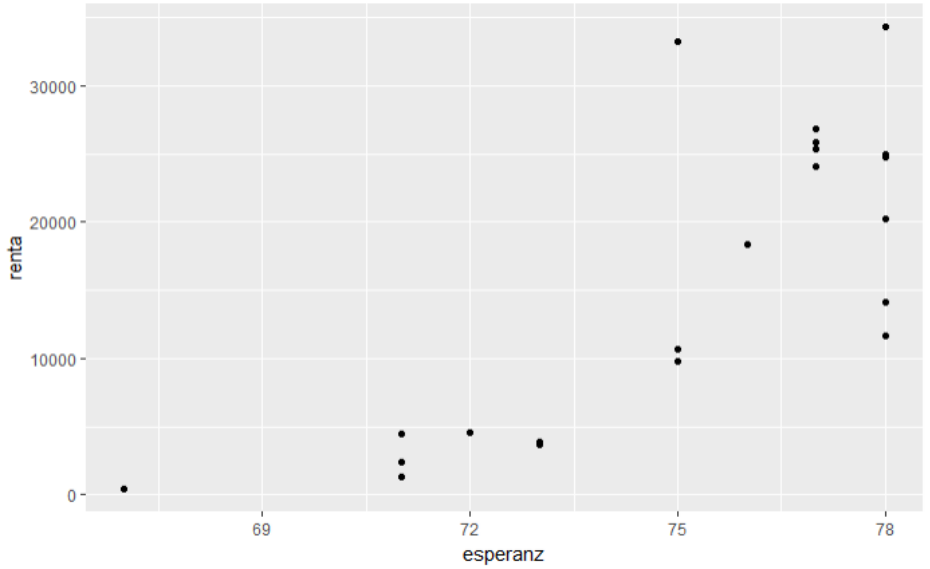
```
> aov(var1 ~ var2)
```

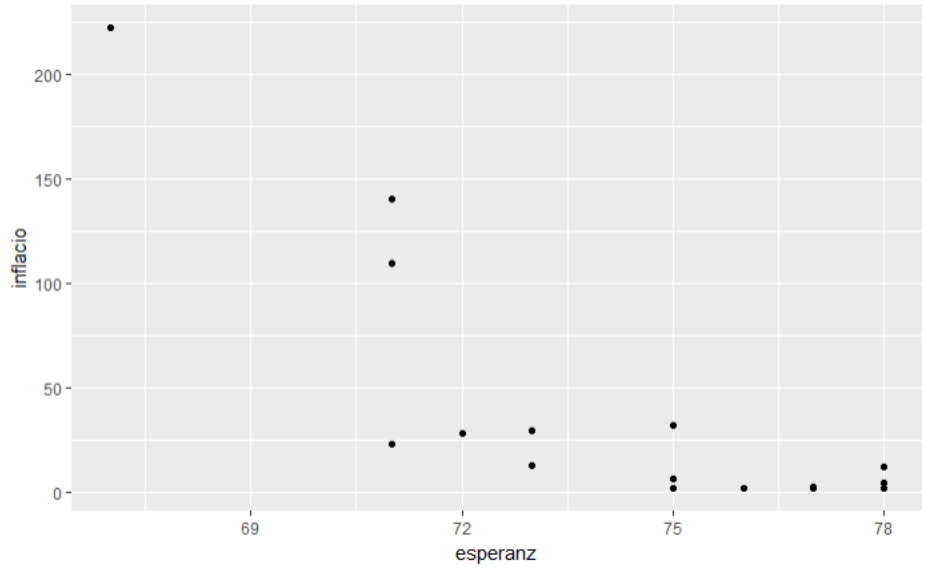
Nota: Esta permite visualizar distintos resultados. Debes asignarle un objeto. Después, prueba resumiendo los resultados o creando un gráfico:

```
> summary()
> plot()
```

Regresiones y correlaciones

¿Influye la renta en la esperanza de vida de los países europeos? ¿Y la inflación?





Regresiones y correlaciones

¿Influye la renta en la esperanza de vida de los países europeos? ¿Y la inflación?

```{r}
cor.test(europa$esperanz, europa$renta)
```

```
        Pearson's product-moment correlation

data:  europa$esperanz and europa$renta
t = 5.4165, df = 19, p-value = 3.163e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5234034 0.9060391
sample estimates:
      cor
0.7790636
```

Regresiones y correlaciones

¿Influye la renta en la esperanza de vida de los países europeos? ¿Y la inflación?

```
Call:
lm(formula = esperanz ~ renta + inflacio, data = europa)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1205 -0.3580 -0.1185  0.9304  2.9355

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.395e+01  8.463e-01  87.378  < 2e-16 ***
renta        1.272e-04  3.936e-05   3.232  0.00463 **
inflacio    -2.985e-02  7.669e-03  -3.892  0.00107 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.54 on 18 degrees of freedom
Multiple R-squared:  0.7866,    Adjusted R-squared:  0.7628
F-statistic: 33.16 on 2 and 18 DF,  p-value: 9.196e-07
```