```
library(dplyr)
rladies_global %>%
filter(city=="Barranquilla" |city== "Galápagos")
```

Manipulación de datos con dplyr y visualización con ggplot2

¡Conoce zoom!

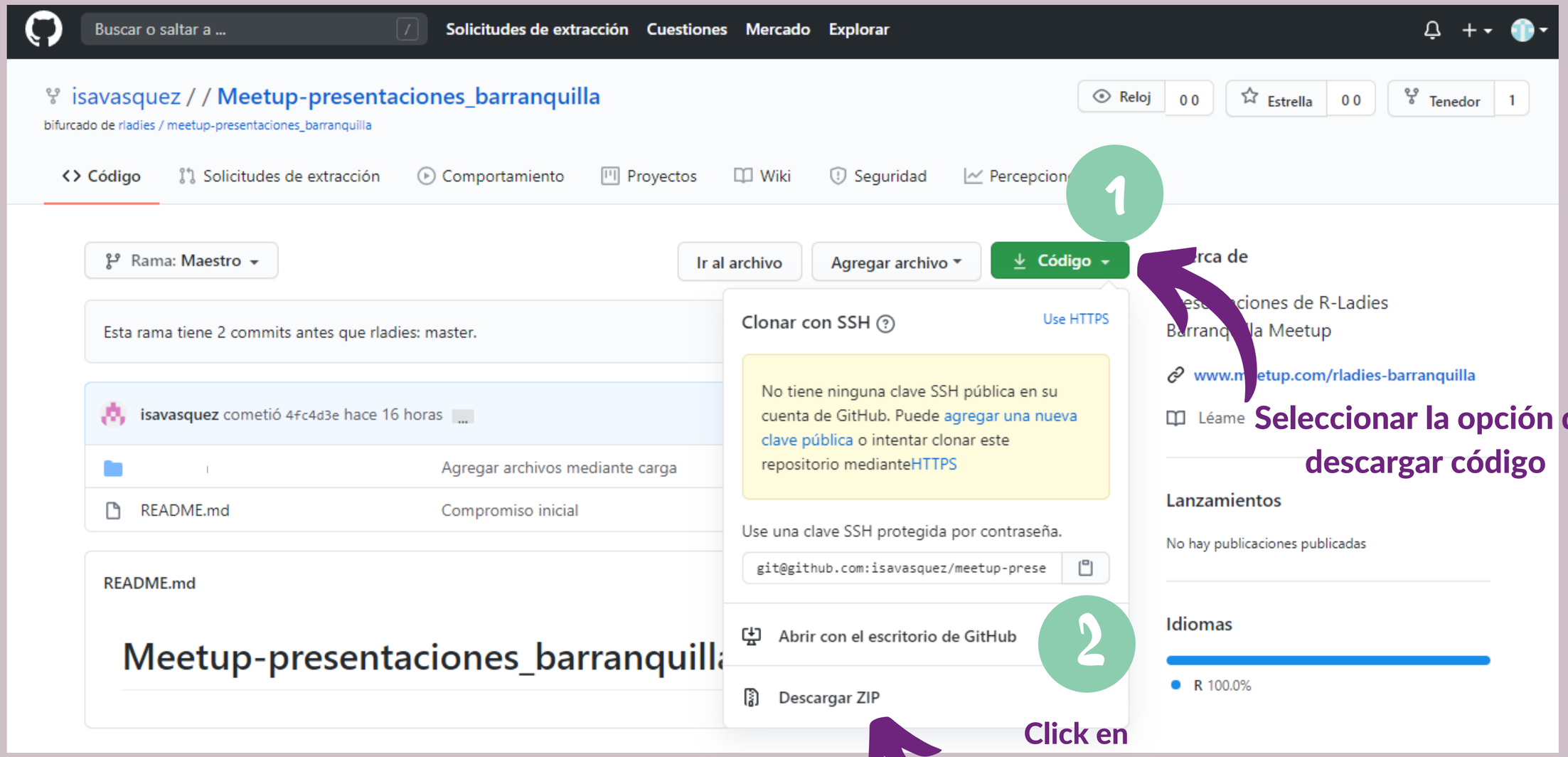
R-Ladies

¡Conoce zoom!

R-Ladies

R-Ladies

GitHub



Seleccionar la opción de descargar código

Click en "Descargar ZIP"

</> Paquetes </>

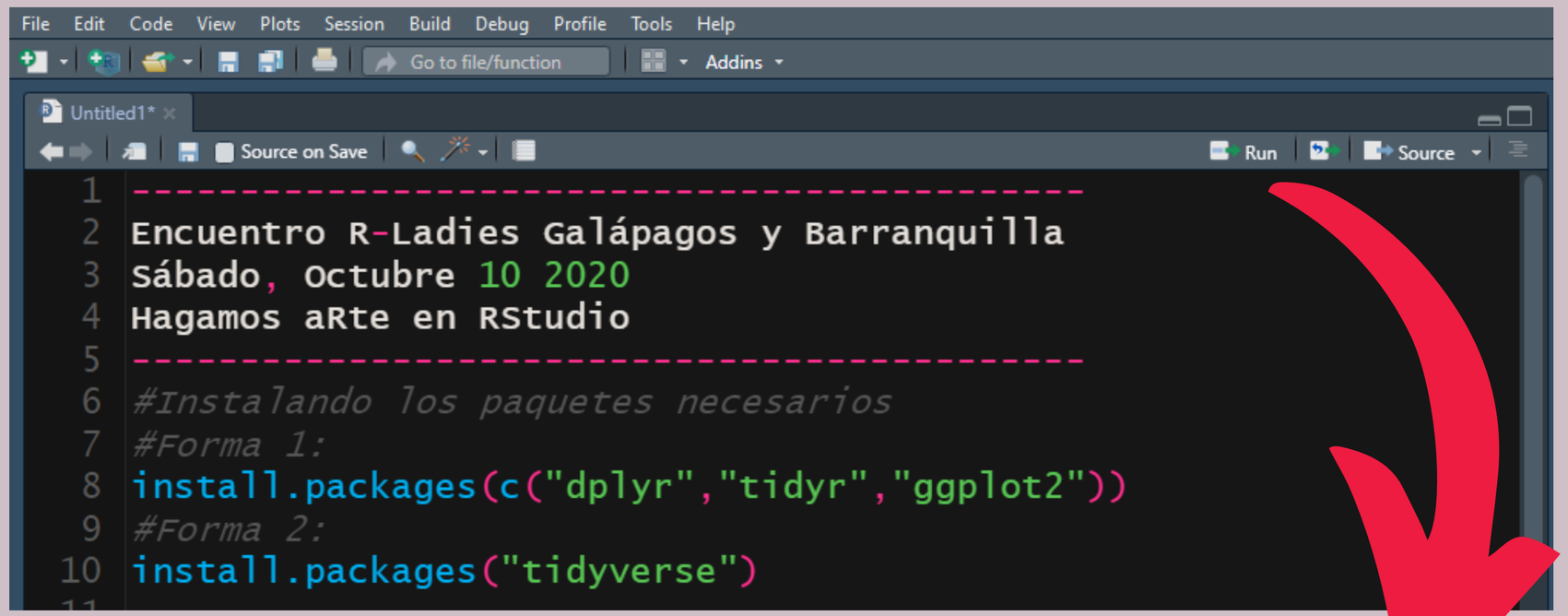
dplyr: es un paquete con distintas funciones que permiten realizar diferentes acciones sobre una base de datos: filtrar, seleccionar columnas, ordenar, añadir variables nuevas, etc.

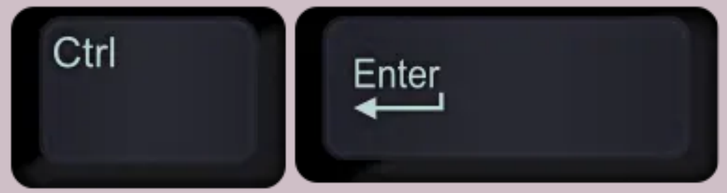
ggplot2: permite construir distintos gráficos a partir de un conjunto de datos

tidyr: facilita la manipulación y organización de un dataset

R-Ladies

¿CÓMO INSTALAR UN PAQUETE EN RStudio ?





R-Ladies

Orden del día

- ✓ R-Ladies.
- ✓ ¿Cómo iniciar sesión en RStudio Cloud?
- ✓ ¿Cómo subir archivos a RStudio Cloud?
- ✓ Ejercicio 1 dplyr
- ✓ Ejercicio 2 dplyr y ggplot2

R-Ladies Global

La comunidad R sufre de una representación insuficiente de los géneros minoritarios (incluidas, entre otras, mujeres cis / trans, hombres trans, no binarios, queer, a-género) en todos los roles y áreas de participación, ya sea como líderes, desarrolladores de paquetes, conferenciantes, participantes de la conferencia, educadores o usuarios (ver estadísticas recientes).

Gabriela de Queiroz

Fundadora R-Ladies Global.

Código de conducta

Espacio libre de acoso

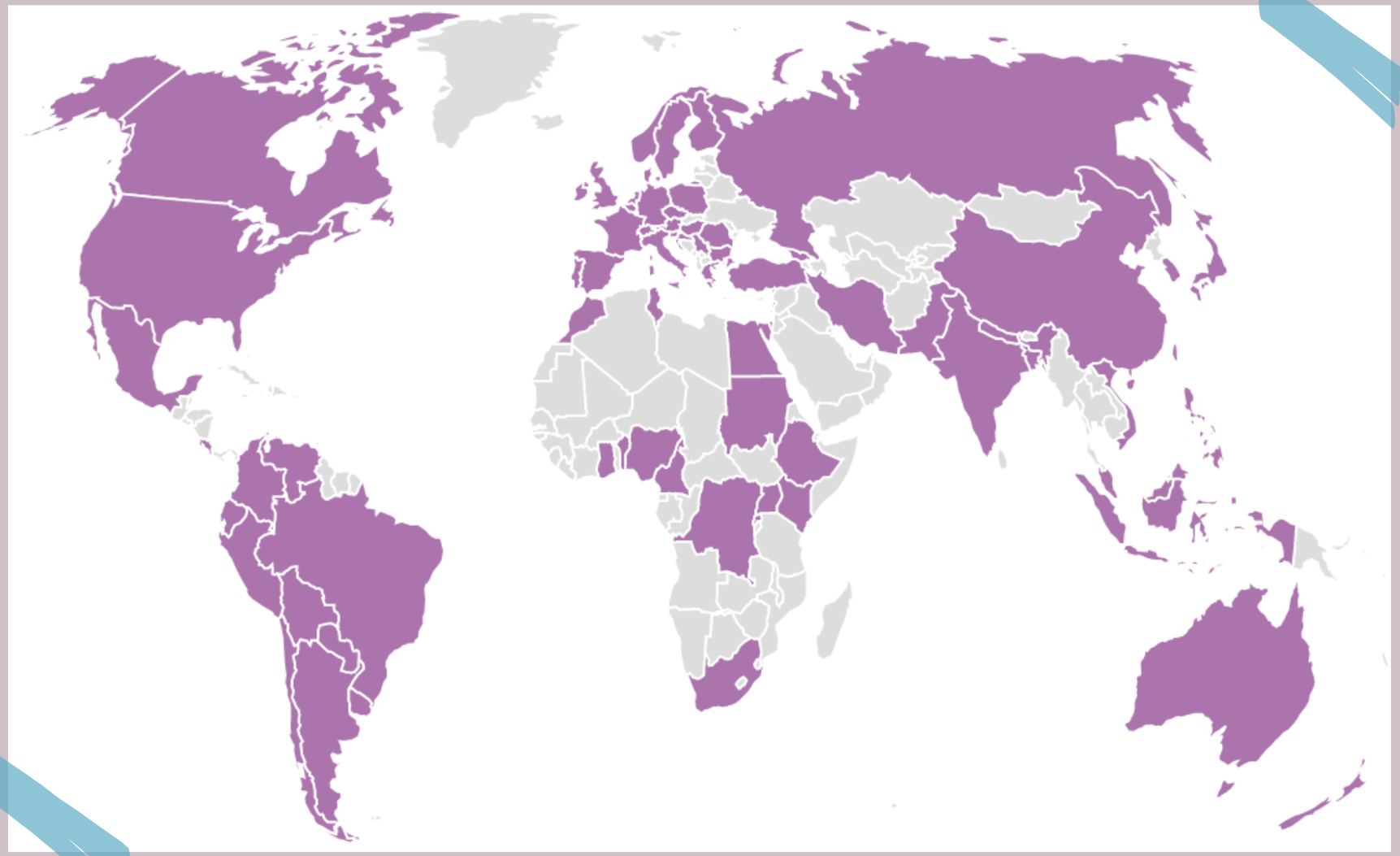
Todas(os) debemos ser tratados con respeto

Alertar cuando se perciba alguna situación peligrosa

Participantes infractores serán sancionados

R-Ladies

R-Ladies en el mundo



consortium

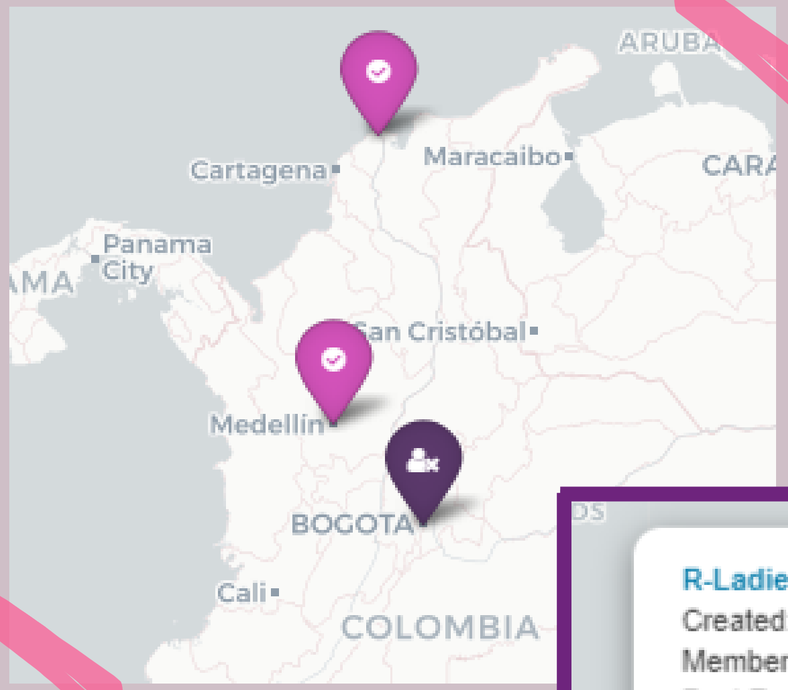
193
Chapters

75333
Members

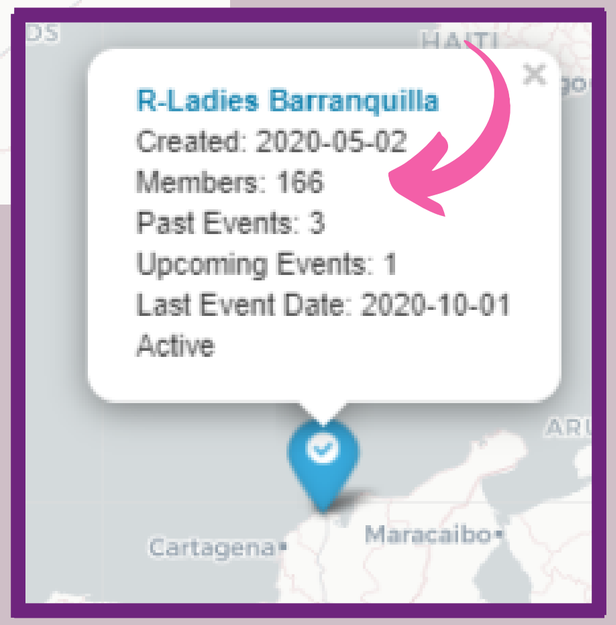
54
Countries

R-Ladies

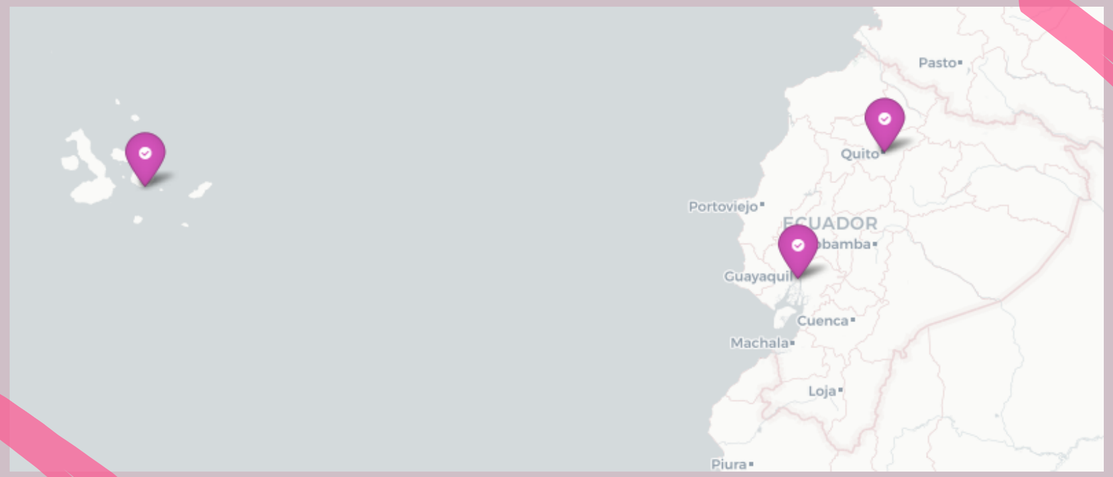
R-Ladies en



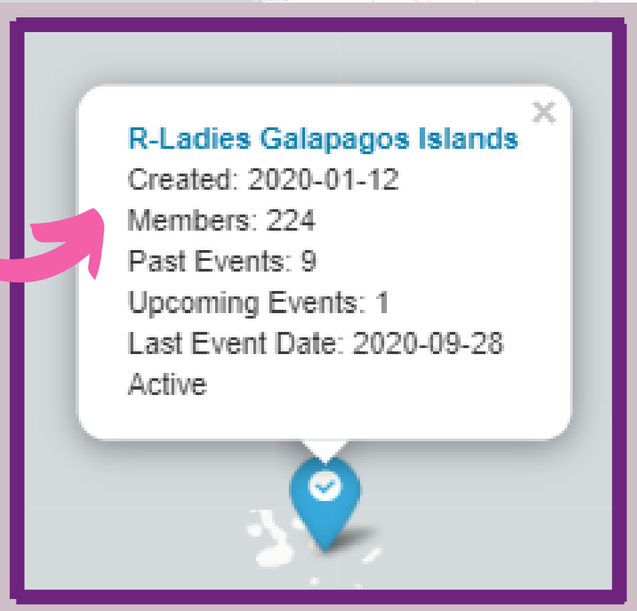
3
Chapters



R-Ladies en



3
Chapters



R-Ladies

R-Ladies Barranquilla

@rladiesbquilla

@RLadies Bquilla

https://github.com/rladies/meetup-presentations_barranquilla

R-Ladies Barranquilla

barranquilla@rladies.org

R-Ladies Barranquilla

R-Ladies Galápagos

https://github.com/rladies/meetup-presentations_galapagos-islands

@rladiesgps

galapagos@rladies.org

R-Ladies Galápagos

R-Ladies Ecuador

Equipo organizador

María Isabel Arrieta

Denisse Fierro

Isabel Vasquez

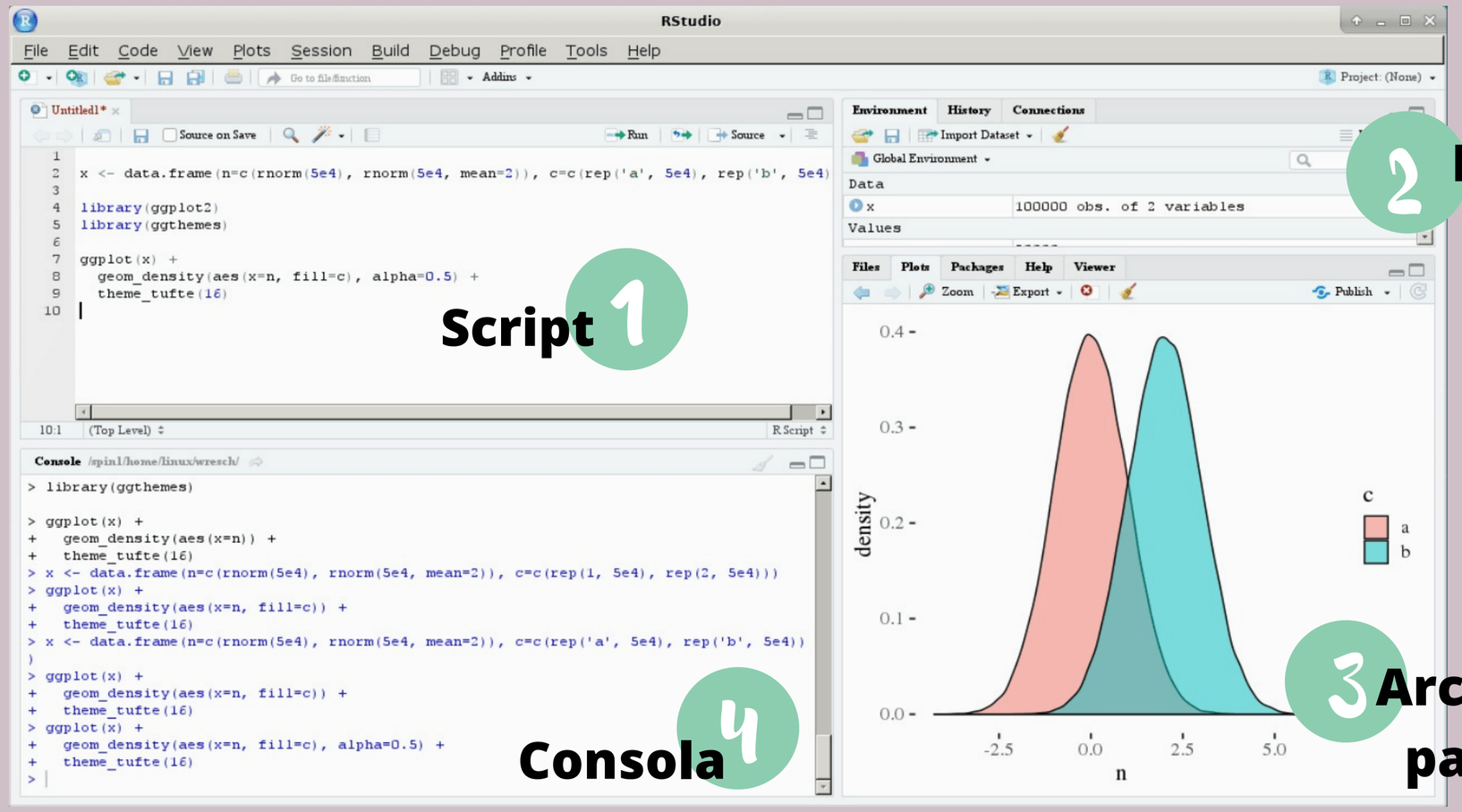
Viviana Flórez

Danisse Carrascal

Mary Jane Rivero

El entorno de RStudio®

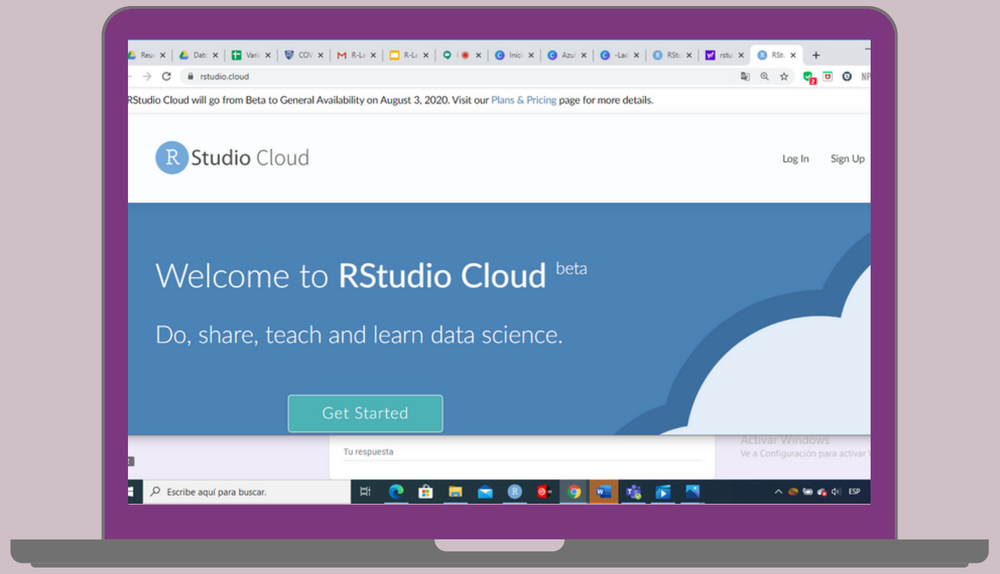
La pantalla se divide principalmente en 4 paneles o ventanas.



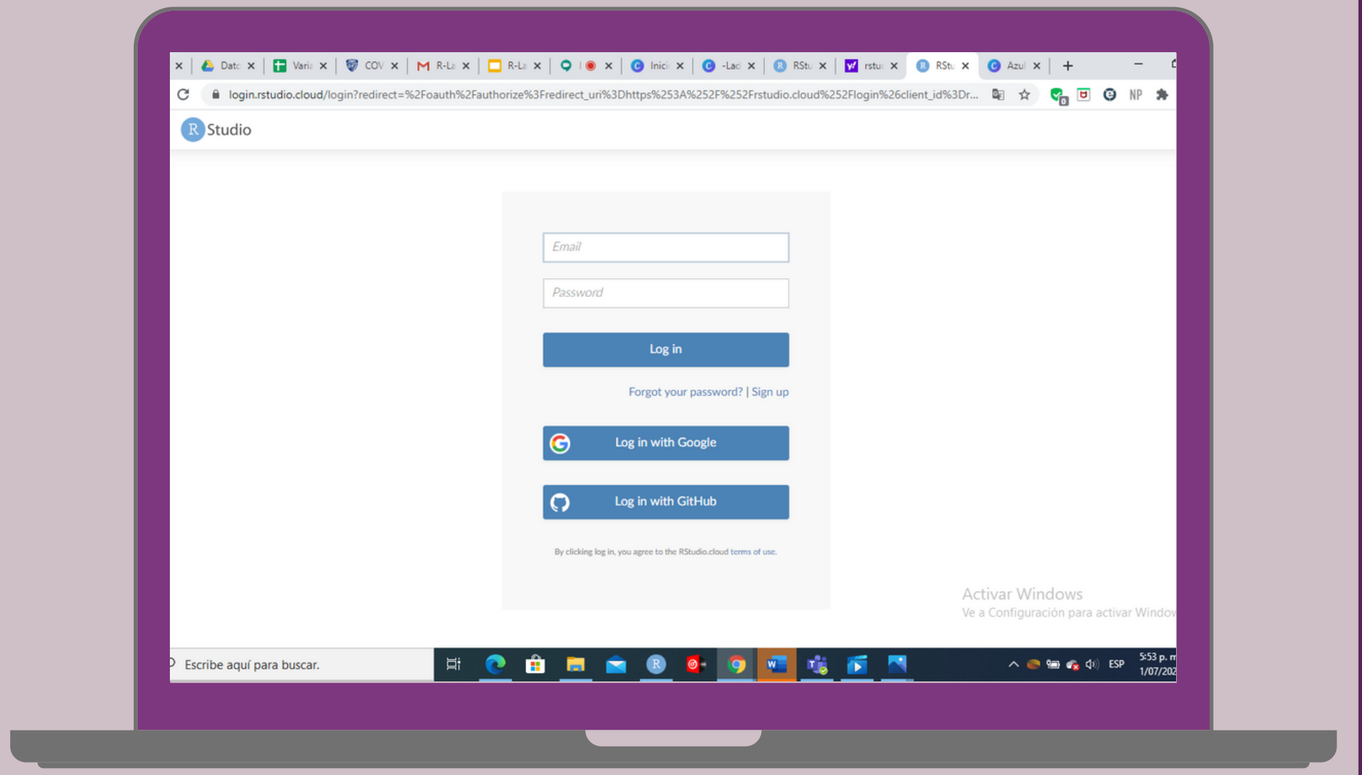
R-Ladies

¿Cómo iniciar sesión RStudio Cloud?

Paso 1: Ingresa al link https://rstudio.cloud/



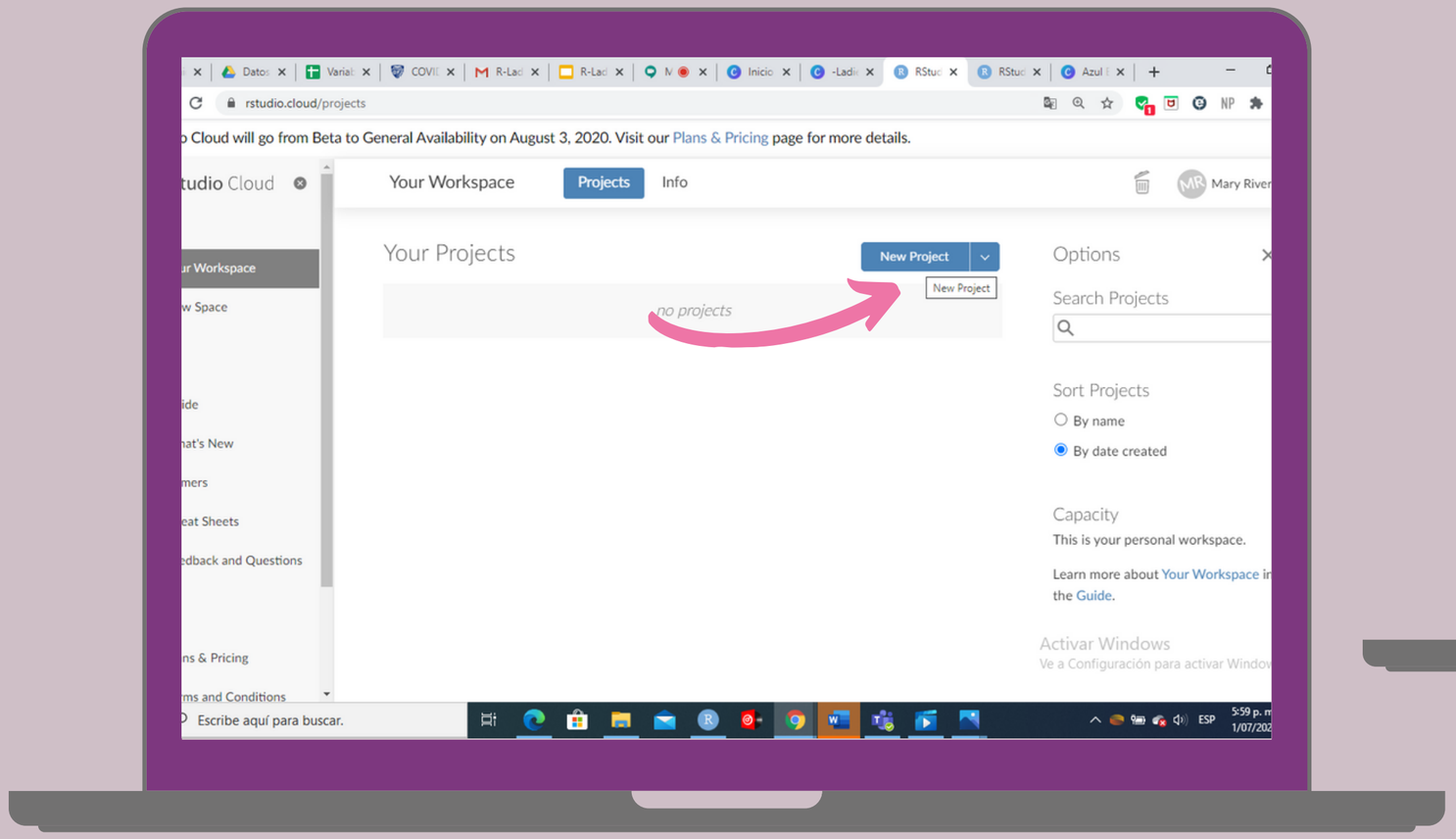
Paso 2: Regístrate o inicia sesión con una cuenta ya existente.



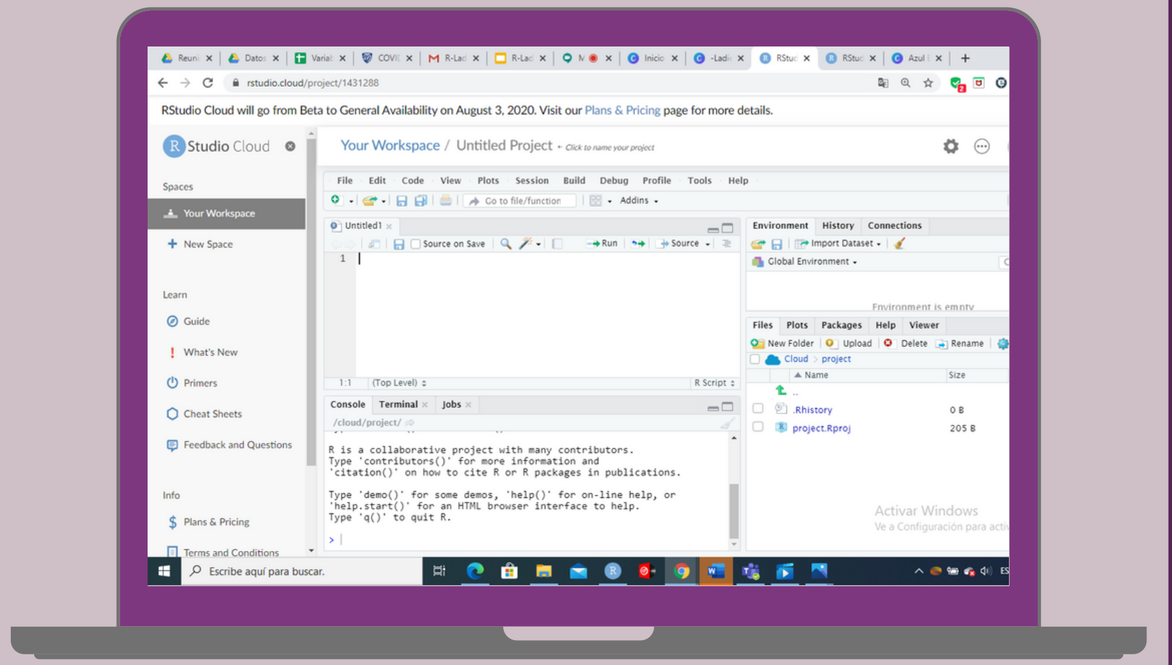
R-Ladies

¿Cómo iniciar sesión RStudio Cloud?

Paso 3: Crea un nuevo proyecto.

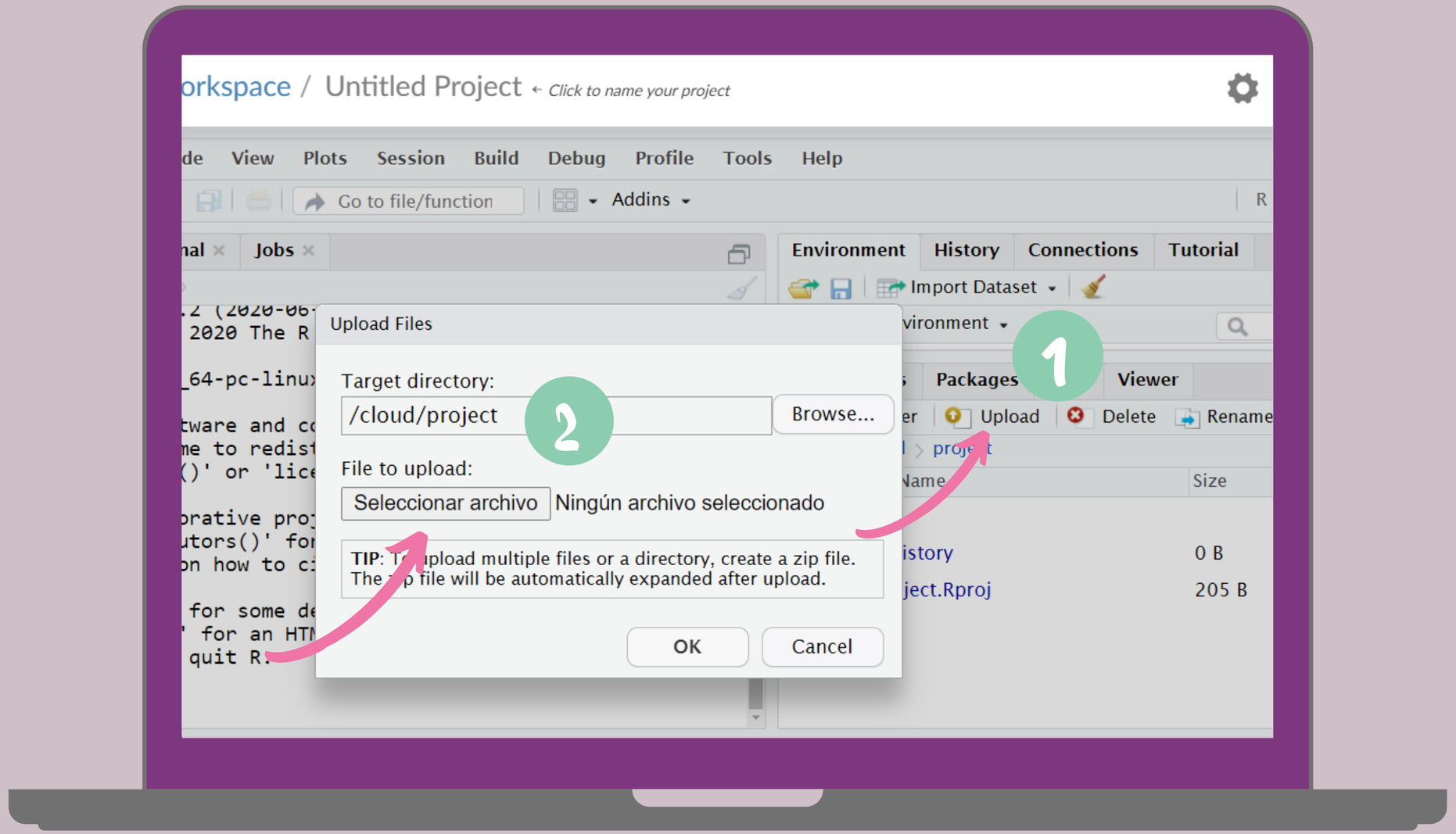


Paso 4: Empieza a disfrutar de RStudio Cloud.



R-Ladies

¿Cómo subir archivos en RStudio® Cloud?



R-Ladies

Hadley Wickham

- Proporciona una "gramática“ para la manipulación de datos y para operar en los marcos de datos.
- Opera de forma rápida

¿Base de datos o informe?

Fecha	Producto	Precio	Cantidad
20/02/2017	Manzana	10	1.0
20/02/2017	Pera	15	2.0
21/03/2017	Ciruela	13	0.5
22/03/2017	Naranja	9	3.0

Mes	Ventas
Febrero	40.0
Marzo	33.5

Datos rectangulares

Ventas				
Frutería “La desprolija”	NA	NA	NA	NA
Primer Trimestre	NA	NA	NA	NA
Manzanas	100	NA	NA	NA
Peras	120	NA	Ventas de fruta de carozo:	614
Ciruelas	85	NA	NA	NA
Segundo Trimestre	NA	NA	Ventas de frutas de pepita:	166
Manzanas	110	NA	NA	NA
Peras	40	NA	NA	NA
Ciruelas	25	NA	NA	NA
Tercer Trimestre	NA	NA	NA	NA
Manzanas	60	NA	NA	NA
Peras	45	NA	NA	NA
Ciruelas	22	NA	NA	NA

Fruta	Primer trimestre	Segundo Trimestre	Tercer Trimestre
Manzanas	100	110	60
Peras	120	40	45
Ciruelas	85	25	22

Ordenar los datos

Cada **columna** es una **variable** y cada **fila** una **observación**.

¿Los datos están ordenados o desordenados?

Raza	Indicador	2018	2019
Beagle	Precio	300.000	350.000
Pastor Alemán	Precio	400.000	340.000
Bulldog	Precio	350.000	400.000

¿Y ahora?

Raza	Año	Precio
Beagle	2018	300.000
Beagle	2019	350.000
Pastor Alemán	2018	400.000
Pastor Alemán	2019	340.000
Bulldog	2018	350.000
Bulldog	2019	400.000

Operador pipe %>%

Útil para concatenar múltiples operaciones de dplyr

El siguiente ejemplo muestra que cada vez que queremos aplicar mas de una función, la instrucción es una secuencia de llamadas a funciones de forma anidada y que resulta ilegible:

```
third(second(first(x)))
```

Este anidamiento no es una forma natural de expresar un secuencia de operaciones. El operador %>% nos permite escribir una secuencia de operaciones de izquierda a derecha:

```
first(x) %>% second(x) %>% third(x)
```

Pipe o pipeline.

Argumento.

Función.

Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

dplyr::select

Selecciona las variables que s·a·t·i·s·f·a·c·e·n tu interés

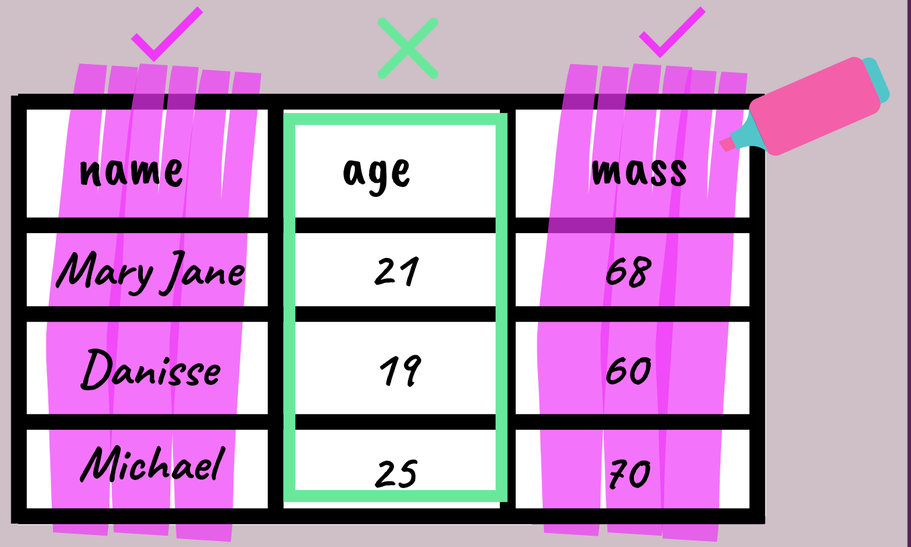
```
library(dplyr)
select(dataset,name,mass)
```

Selecciona las variables.

Conjunto de datos

No olvides separar los argumentos con coma.

Variables de interés sin comillas.



name	age	mass
Mary Jane	21	68
Danisse	19	60
Michael	25	70

dplyr::glimpse

Conoce y explora las características de tus variables

```
library(dplyr)
glimpse(x = dataset, width = NULL,
        ...)
```

Explora la estructura de tus datos.

Conjunto de datos.

Otros argumentos pasados a métodos individuales.

Ancho de salida.

¡No olvides las comas!

```
Console  Terminal  Jobs
/cloud/project/
>glimpse(x = dataset, width = NULL)
Rows: 3
Columns: 2
$ name   <chr> "Mary Jane", "Danisse", "Michael"
$ mass   <dbl>  68, 60, 70
```

dplyr::arrange

```
library(dplyr)
arrange(x = dataset,mass)
```

Ordena las filas ascendentemente por defecto.

Conjunto de datos.

¡La coma!

Variable a ordenar

name	mass
Mary Jane	68
Danisse	60
Michael	70

name	mass
Danisse	60
Mary Jane	68
Michael	70

Ordena las filas de tus variables

```
library(dplyr)
arrange(x = dataset,des(age))
```

Ordena las filas.

Conjunto de datos.

¡La coma!

Variable a ordenar.

Ordena los datos de forma descendente.

name	mass
Mary Jane	68
Danisse	60
Michael	70

name	mass
Michael	70
Mary Jane	68
Danisse	60

dplyr::rename

Renombra las variables de tu dataframe

Conjunto de datos

```
library(dplyr)
rename(Storms, tormenta =
storm, viento = wind, presion
  = pressure, fecha = date)
```

Cambia el nombre de tus columnas

¡La coma!

Nuevo nombre

Antiguo nombre

```
Console  Terminal  Jobs
/cloud/project/
>glimpse(x = dataset, width = NULL)
  tormenta viento presion      fecha
     (chr)  (int)   (int)     (date)
1  Alberto    110    1007 2000-08-03
2     Alex     45    1009 1998-07-27
3  Allison     65    1005 1995-06-03
4      Ana     40    1013 1997-06-30
5   Arlene     50    1010 1999-06-11
6   Arthur     45    1010 1996-06-17
```

Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

dplyr::mutate

Crea una nueva v·a·r·i·a·b·l·e a partir de variables existentes

```
library(dplyr)
mutate(Storms,ratio = pressure/wind)
```

Genera una nueva columna o variable.

Conjunto de datos.

Nueva variable.

Variables existentes.

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

```
.after = "wind" |.before = "wind"
```

Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

dplyr::filter

```
library(dplyr)
filter(Storms,wind >= 50 &
         pressure <= 1010)
```

Condición 1

Selecciona tus filas.

Conjunto de datos

Condición 2

Operador lógico

Storms

storm	wind	pressure	date	
Alberto	110	1007	2000-08-12	✓
Alex	45	1009	1998-07-30	✗
Allison	65	1005	1995-06-04	✓
Ana	40	1013	1997-07-01	✗
Arlene	50	1010	1999-06-13	✓
Arthur	45	1010	1996-06-21	✗

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

Conserva las filas que s·a·t·i·s·f·a·c·e·n tu interés

Las condiciones pueden ser expresiones lógicas construidas mediante los operadores relacionales y lógicos:

Relacionales

<	Menor que
>	Mayor que
==	Igual que
<=	Menor igual que
>=	Mayor igual que
!=	Diferente que
%in%	Pertenece al conjunto
is.na	Es NA
!is.na	No es NA

Lógicos

&	Booleano y
|	Booleano o
xor	O inclusivo
!	No
any	Cualquier verdadero
all	Todos los verdaderos

Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

dplyr::summarise()

Realiza operaciones con variables existentes y crea un nuevo data frame

```
library(dplyr)
summarise(pollution, mediana = median(amount), variance = var(amount))
```

Realiza operaciones y crea un nuevo data frame.

Conjunto de datos.

Nombre de la variable en el nuevo data frame.

Operación.

Variable a operar.

pollution

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

→

median	variance
22.5	1731.6

A continuación presentamos las funciones que trabajan conjuntamente con la función summarise(). Todas ellas toman como argumento un vector y devuelven un único resultado.

Paquete base

min(), max()

Valores mínimo y máximo.

mean()

Media.

median()

Mediana.

sum()

Suma de los valores.

var(), sd()

Varianza y desviación típica.

Paquete dplyr

first()

Primer valor en un vector.

last()

El último valor en un vector.

n()

El número de valores en un vector.

n_distinct()

El número de valores distintos en un vector.

nth()

Extrae el valor que ocupa la posición n en un vector

Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

dplyr::group_by()

Toma una tbl existente y

c·r·e·a · u·n·a · t·b·l

nueva donde las operaciones se realizan por grupo

```
library(dplyr)
group_by(pollution, city)
```

Agrupa por grupos.

Conjunto de datos.

Variable para agrupar.

pollution

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

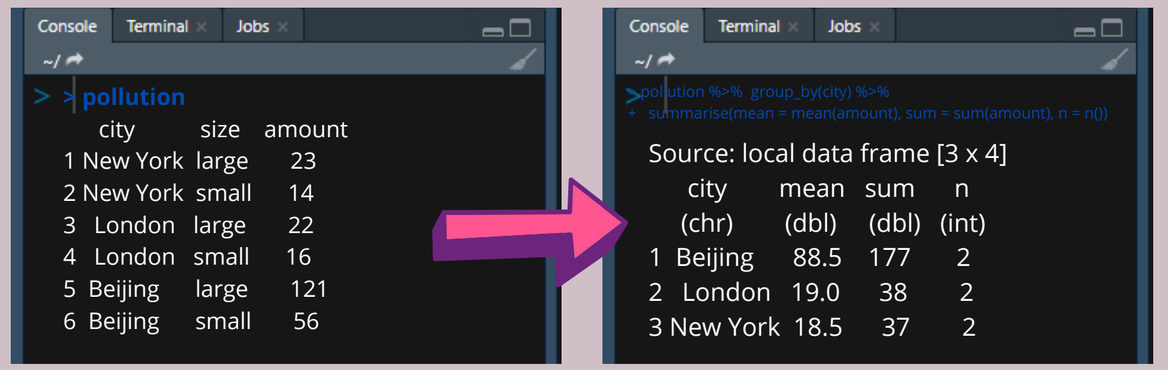
→

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

3 GRUPOS

La función **group_by()** es extremadamente útil trabajando en conjunto con la función **summarise()**:

```
library(dplyr)
pollution %>%  group_by(city) %>%
summarise(mean = mean(amount), sum =
sum(amount), n = n())
```



Fuente: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/filter.html

tidyr::gather()

Toma un dataset existente y

c o n v i e r t e l a s

columnas en valores u observaciones

```
library(dplyr)
messy %>%
  gather(drug, heartrate, a:b)
```

Base de datos.

Re-ordena el dataset

Nombre nuevas columnas

Columnas originales

```
messy
#>      name  a  b
#> 1  Wilbur 67 56
#> 2 Petunia 80 90
#> 3 Gregory 64 50
```

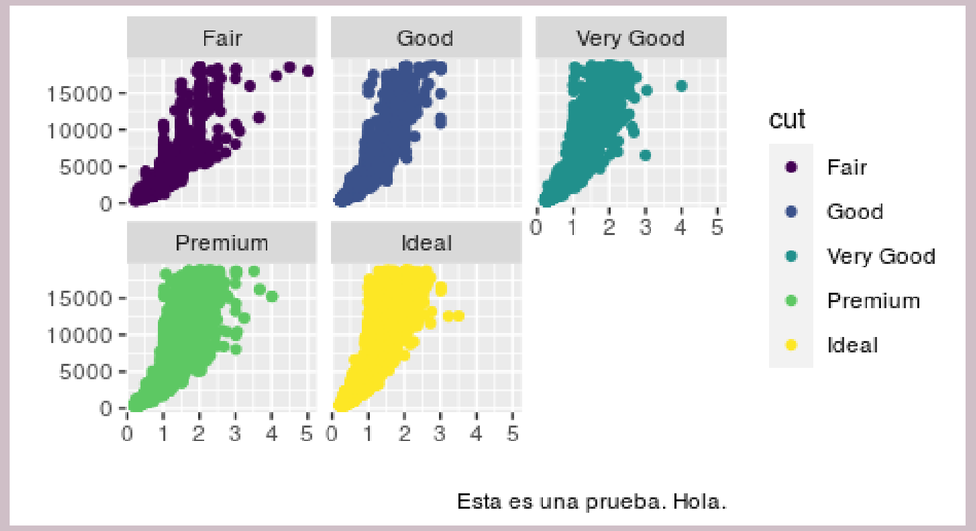
```
messy %>%
  gather(drug, heartrate, a:b)
#>      name drug heartrate
#> 1  Wilbur    a        67
#> 2 Petunia    a        80
#> 3 Gregory    a        64
#> 4  Wilbur    b        56
#> 5 Petunia    b        90
#> 6 Gregory    b        50
```

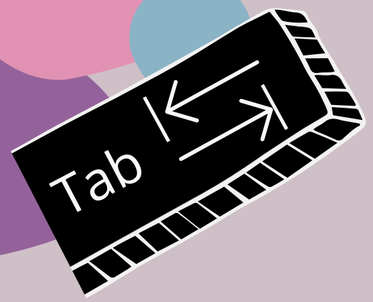
Fuente: https://blog.rstudio.com/2014/07/22/introducing-tidyr/

ggplot2

Construye distintos g·r·á·f·i·c·o·s a partir de un conjunto de datos

```
ggplot(diamonds, mapping = aes(x=carat, y=price, color=cut)) + # es la base, no aparecerá nada
  geom_point() + #IMPORTANTE EL SIGNO "+" ANTES DE EMPEZAR OTRA CAPA
  facet_wrap(~cut)+ #Para que aparezcan separadas
  labs(title = "", subtitle = "",
       y="", x="", caption = "Esta es una prueba. Hola.")
```






```
install.packages()
```

Hagamos un poco de aRte

```
??function
```

```
help()
```

```
datos <- read.csv2("RLadies.csv")
 View(datos)
```

Ctrl + Enter