

Introducción a R y Rstudio

Dr. Samuel D. Gamboa Tuz

¿Qué es R?



- **R** es un lenguaje y entorno de programación utilizado mayormente para análisis estadísticos y creación de gráficos para publicación.
- Lenguaje interpretado.
- Multiparadigma: programación funcional y orientada a objetos.
- Código abierto, libre y gratis (*GNU General Public License*).
- Existe una gran comunidad de usuarios de R. Algunos ejemplos en latinoamérica:
 - https://comunidadbioinfo.github.io/post/user_latin_american_community/

Un poquito de historia...

- En 1976, John Chambers comenzó a desarrollar el language de programación S en *Bell Labs*, EUA. El diseño de este language está enfocado a análisis estadísticos.
- En 1991, **R** fue creado por **Ross Ihaka** y **Robert Gentleman** de la Universidad de Auckland, Nueva Zelanda como una implementación de S.
- En 1995, R se convirtió en un software libre bajo la *GNU General Public License*.
- En 1997, se creó el *R-core Team*. Sitio oficial <https://www.r-project.org>.
- En 2000, la versión 1.0.0 de R estuvo disponible para el público en general.
- En 2020, salió la versión 4.0.3 - "Bunny-Wunnies Freak Out."
- <https://bookdown.org/rdpeng/rprogdatascience/history-and-overview-of-r.html>

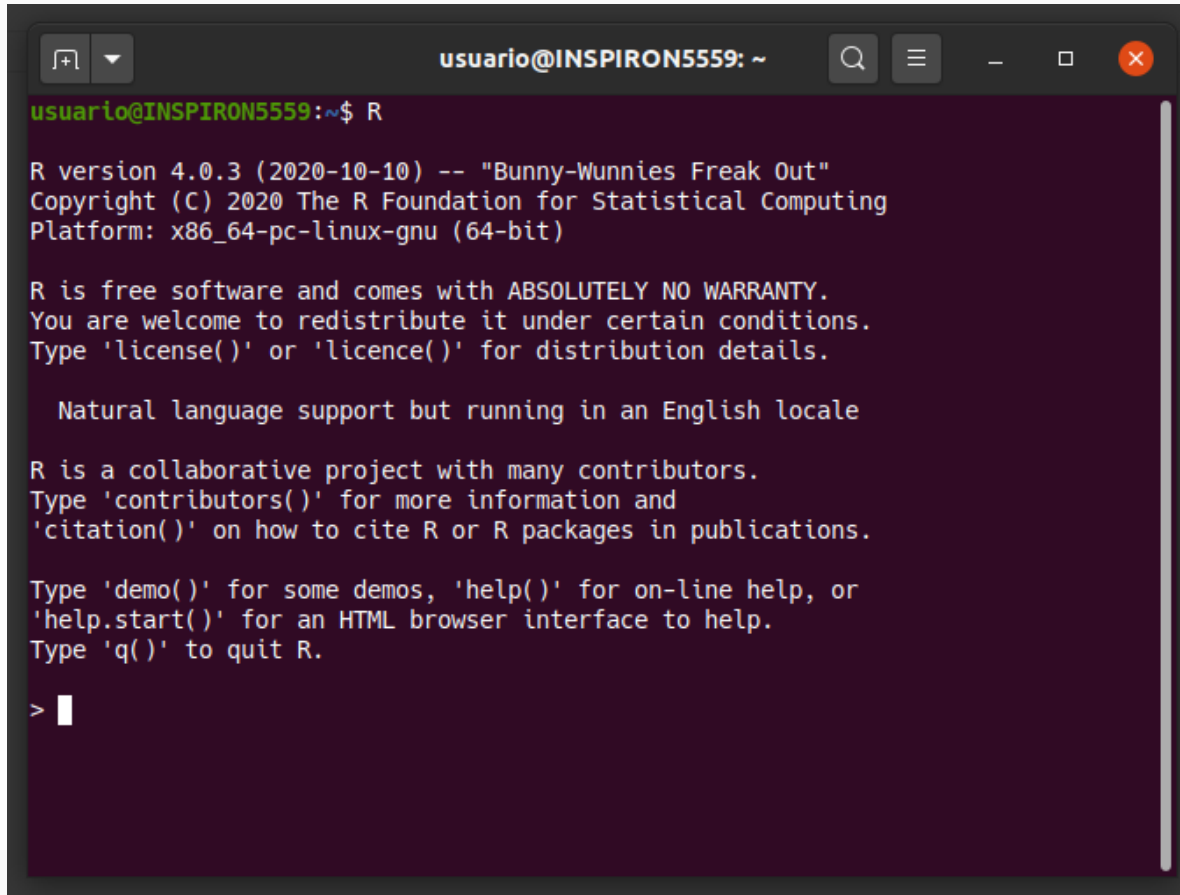
Cómo instalar R

- Entrar a la *Comprehensive R Archive Network*: <https://cran.r-project.org/mirrors.html>.
- Elegir el *mirror*; por ejemplo, <https://cran.itam.mx/>.
- Elegir el sistema operativo y seguir las instrucciones.
- Por ejemplo, para Ubuntu 20.04.1 LTS:
 1. Abrir el archivo **/etc/apt/sources.list** con un editor de texto (con permiso de superusuario).
 2. Agregar la línea: `deb https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/`
 3. Ejecutar en terminal:

```
sudo apt-get update  
sudo apt-get install r-base  
sudo apt-get install r-base-dev
```

Iniciar sesión de R (en Linux)

- Abrir terminal y ejecutar `$ R`.



The screenshot shows a terminal window titled 'usuario@INSPIRON5559: ~'. The prompt is 'usuario@INSPIRON5559:~\$' and the command 'R' has been entered. The output displays the R version (4.0.3), copyright information (© 2020 The R Foundation), platform (x86_64-pc-linux-gnu), and a welcome message. It also provides instructions on how to use R, including running demos, getting help, and quitting. The prompt is now '>'.

```
usuario@INSPIRON5559:~$ R

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Expresiones y asignaciones

- Ejemplo de **expresión**: el comando es **evaluado**, se imprime en pantalla (por lo general) y desaparece.

```
10 * 2 # Expresión / expression
```

```
## [1] 20
```

- Ejemplo de **asignación**: el comando (expresión) es **evaluado**, pero no se imprime. El resultado se almacena en un **objeto** definido por un **nombre** utilizando el **operador** `<-` (*assignment operator*).

```
x <- 3 + 2 # Asignación / assignment
```

- Hay que indicarle a R que imprima el resultado almacenado en el objeto en pantalla.

```
x # Se le ordena a R imprimir x
```

```
## [1] 5
```

- Todo lo que va después de un `#` es un **comentario** y **no es evaluado** por R (a menos que esté entre comillas "dobles" o 'simples').

Funciones

- Las **funciones** son "comandos" que realizan alguna acción cuando se les pasan **argumentos** (algunos comandos pueden usarse sin argumentos).
- La función `ls()` puede usarse sin argumentos para listar todos los objetos presentes en el área de trabajo (**workspace**):

```
ls() # objects() cumple la misma función
```

```
## [1] "x"
```

- A la función `rm()` se le puede pasar como argumentos los objetos que se desean borrar del *workspace*.

```
rm(x)
```

- Comprobamos:

```
ls()
```

```
## character(0)
```

- Por ejemplo, a la función `seq()`, que genera una secuencia de números, le podemos pasar tres argumentos posicionales:

```
seq(1, 10, 2)
```

```
## [1] 1 3 5 7 9
```

- Pero si indicamos los argumentos de manera explícita, éstos pueden ser pasados a la función en cualquier orden. La siguiente **function call** da el mismo resultado que el ejemplo anterior:

```
seq(by = 2, to = 10, from = 1)
```

```
## [1] 1 3 5 7 9
```

- Puedes saber qué argumentos lleva una función con `?foo`, donde `foo` es el nombre de la función. Por ejemplo, `?seq`.

- Puedes **definir** tus propias funciones con la función `function()`.
- Por ejemplo, una función que eleva un número (argumento *data*) a la potencia 2 por default, pero puede elevar a la potencia indicada por un segundo argumento (*power*):

```
my_function <- function(data, power = 2) {  
  data ^ power  
}  
my_function(2) # Sin otro argumento
```

```
## [1] 4
```

```
my_function(2, 3) # Argumentos posicionales
```

```
## [1] 8
```

```
my_function(power = 3, data = 2) # Argumentos explícitos
```

```
## [1] 8
```

- Dos o más funciones pueden estar **anidadas**; es decir, el resultado de evaluar una función puede pasarse como argumento a una segunda función en una sola llamada (*call*):

```
seq(1, 20)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```
head(seq(1, 20))
```

```
## [1] 1 2 3 4 5 6
```

- En el ejemplo anterior, la función `head()` por default muestra los primeros 6 elementos de un objeto. En este caso, el resultado de `seq()` se pasó a `head()`, por lo que solo se ven los primeros seis números y no los 20 completos.

Algunas funciones que te pueden ser útiles:

Función	Acción
<code>getwd()</code>	Obtener el directorio de trabajo.
<code>setwd()</code>	Ubicarte en un nuevo directorio de trabajo.
<code>mean()</code> , <code>median()</code> , <code>sd()</code> , <code>var()</code>	Media, mediana, desviación estándar, varianza.
<code>std_error()</code> *	Error estándar (<i>custom</i>).
<code>min()</code> , <code>max()</code> , <code>range()</code>	Valor mínimo, máximo y valores min. y máx.
<code>seq()</code> , <code>rep()</code>	Crear secuencia de números, repetir valores.
<code>paste()</code> , <code>paste0()</code>	Unir caracteres.
<code>grep()</code> , <code>grepl()</code>	Buscar patrones (<i>regex</i>) en un vector de caracteres.
<code>print()</code>	Imprimir en pantalla.
<code>head()</code> , <code>tail()</code>	Mostrar los primeros y últimos elementos de un vector.
<code>sample()</code>	Generar un vector con elementos al azar.

- `std_error <- function(x) sqrt(var(x)/length(x))`

Reglas para los nombres de objetos

- Los nombres admiten letras, números, " _ " y " . ".
- Recomendable iniciar nombre con alguna letra.
- Hay nombres especiales que son reservados para R, por ejemplo: **if**, **for**, etc. Si lo intentas, R arroja un error.
- Ejemplo de estilos de nombres:

```
camelCase <- 1
PascalCase <- "a"
snake_case <- TRUE # Recomendando
dot.case <- c(1,2,3)
```

- Cada comando puede estar separado por un *enter* (**newline**) o un ";".
- Los espacios y sangría (*indentation*) no son tomados en cuenta por R para evaluar, pero son útiles para poder leer mejor el código.

```
ls();rm(camelCase,PascalCase,snake_case,dot.case);ls()
```

```
## [1] "camelCase"      "dot.case"       "my_function"    "PascalCase"
## [5] "snake_case"

## [1] "my_function"
```

Operadores

Operador	Descripción	Asociatividad
\wedge	Exponenciación	dcha. a izda.
$-x, +x$	Unarios	izda. a dcha.
$\%\%$	Módulo	izda. a dcha.
$*, /$	Multiplicación, división	izda. a dcha.
$+, -$	Suma, resta	izda. a dcha.
$<, >, <=, >=, ==, !=$	Comparaciones	izda. a dcha.
$!$	Negación (<i>NOT</i>)	izda. a dcha.
$\&, \&\&$	<i>AND</i>	izda. a dcha.
$, $	<i>OR</i>	izda. a dcha.

- Los operadores se pueden checar en R con `?Syntax()`

- A veces los operadores no se comportan como se esperaría...
- Si pruebas si un objeto contiene un `NA` con `==` siempre dará `NA`, aún cuando sí contenga un valor:

```
x <- 1; y <- NA  
x == NA; y == NA
```

```
## [1] NA
```

```
## [1] NA
```

- En este caso es mejor usar la función `is.na()`:

```
is.na(x); is.na(y)
```

```
## [1] FALSE
```

```
## [1] TRUE
```

```
!is.na(x); !is.na(y)
```

```
## [1] TRUE
```

```
## [1] FALSE
```

Paquetes

- Los **paquetes** contienen las funciones y sets de datos en un *namespace* determinado.
- Algunos paquetes vienen pre-cargados en R, los podemos checar con `(.packages())`.

```
(.packages())
```

```
## [1] "stats"      "graphics"  "grDevices" "utils"     "datasets"  
## [6] "methods"   "base"
```

- Podemos ver todos los paquetes instalados con `installed.packages()` o `library()`.
- Prioridad:
 - **base:** `rownames(installed.packages(priority = "base"))`.
 - **recommended:** `rownames(installed.packages(priority = "recommended"))`.
 - **high:** *base + recommended*.

Cargar paquetes

- Los paquetes se pueden cargar con `library()` o `require()`:

```
library(MASS)
(.packages()) # Ver paquetes cargados.
```

```
## [1] "MASS"      "stats"      "graphics"   "grDevices"  "utils"
## [6] "datasets"  "methods"    "base"
```

- Podemos quitar un paquete cargado del *workspace* con `detach()`:

```
detach(package:MASS)
search() # También se puede usar para ver qué paquetes están cargados.
```

```
## [1] ".GlobalEnv"      "tools:rstudio"    "package:stats"
## [4] "package:graphics" "package:grDevices" "package:utils"
## [7] "package:datasets" "package:methods"   "Autoloads"
## [10] "package:base"
```

- Puedes acceder a una función o set de datos de un paquete sin cargarlo usando el operador `::`:

```
head(MASS::abbey)
```

```
## [1] 5.2 6.5 6.9 7.0 7.0 7.0
```


Instalar y desinstalar paquetes

Nuevos paquetes se instalan con la función `install.packages("packagename")` (*packagename* es el nombre del paquete entre comillas).

- CRAN (<https://cran.r-project.org/>). Ejemplo: `install.packages("tidyverse")`

Algunos repositorios tienen sus propios instaladores.

- Bioconductor (<https://www.bioconductor.org/install/>). Ejemplo:
`BiocManager::install("ComplexHeatmap")`
- Github (<https://github.com/>). Ejemplo:
`devtools::install_github("jokergoo/ComplexHeatmap")`

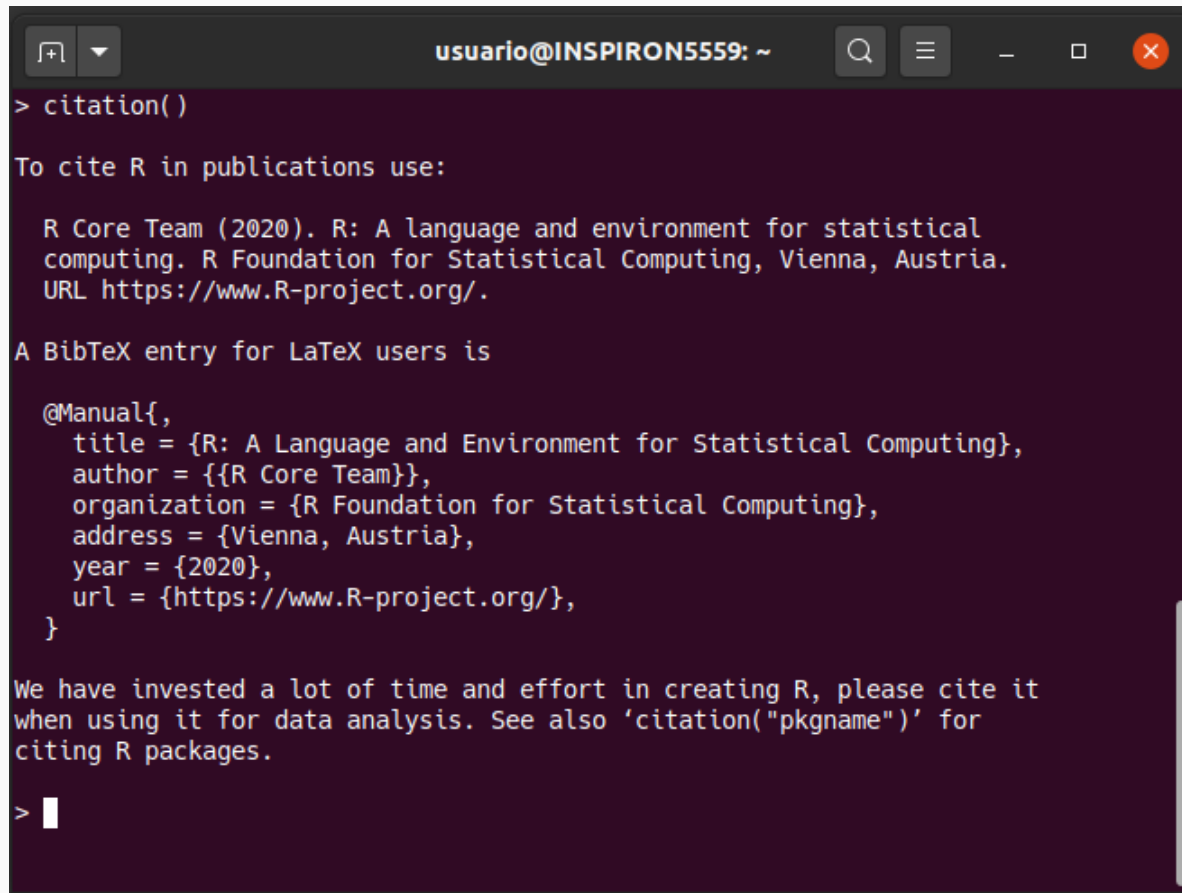
Un paquete se puede desinstalar con `remove.packages("packagename")`.

Cómo conseguir ayuda ?

- Puedes encontrar ayuda en la web con `?help.star()`.
- Puedes encontrar ayuda sobre un paquete o función en específico con la función `help()` o el operador `?`. Por ejemplo:
 - `help("?")`.
 - `?help`.
 - `help(package = "packagename")`.
- Puedes usar la función `help.search()` o `??` para encontrar palabras que se encuentre en el título o palabras claves de la documentación de algún paquete (y acceder a la documentación).
- ¿Necesitas más ayuda?
 - Documentación.
 - Stackoverflow: <https://stackoverflow.com>.
 - Bioconductor Help Forum: <https://support.bioconductor.org/> (Biología).
 - Comunidad R (<https://www.r-statistics.com/>, <https://community.rstudio.com/>, Twitter #rstats, etc.).
 - Google.

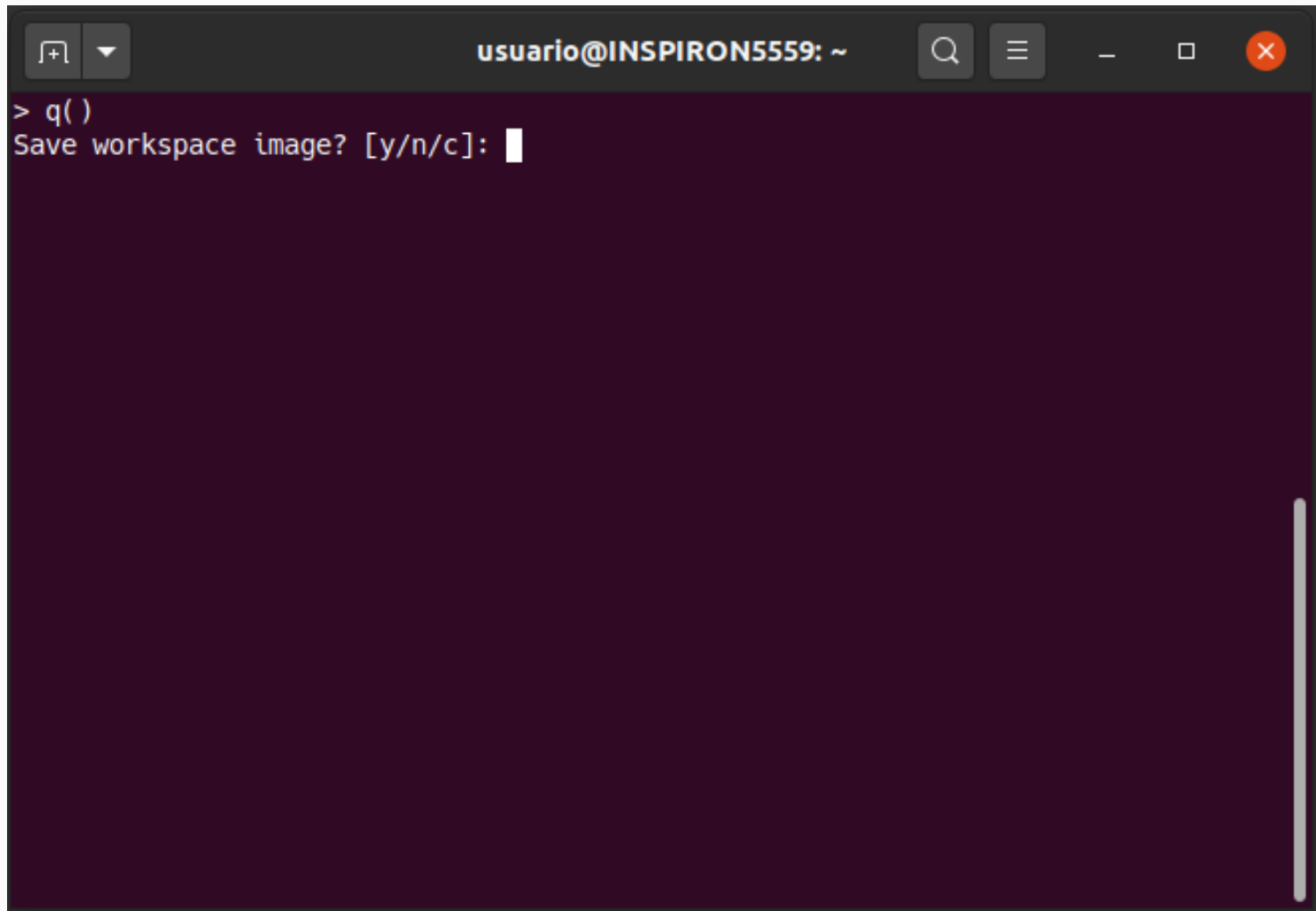
¿Cómo citar R?

- Dentro de R, ejecutar `citation()`.



```
usuario@INSPIRON5559: ~  
> citation()  
  
To cite R in publications use:  
  
R Core Team (2020). R: A language and environment for statistical  
computing. R Foundation for Statistical Computing, Vienna, Austria.  
URL https://www.R-project.org/.  
  
A BibTeX entry for LaTeX users is  
  
@Manual{,  
  title = {R: A Language and Environment for Statistical Computing},  
  author = {{R Core Team}},  
  organization = {R Foundation for Statistical Computing},  
  address = {Vienna, Austria},  
  year = {2020},  
  url = {https://www.R-project.org/},  
}  
  
We have invested a lot of time and effort in creating R, please cite it  
when using it for data analysis. See also 'citation("pkgname")' for  
citing R packages.  
  
> █
```

Salir de R

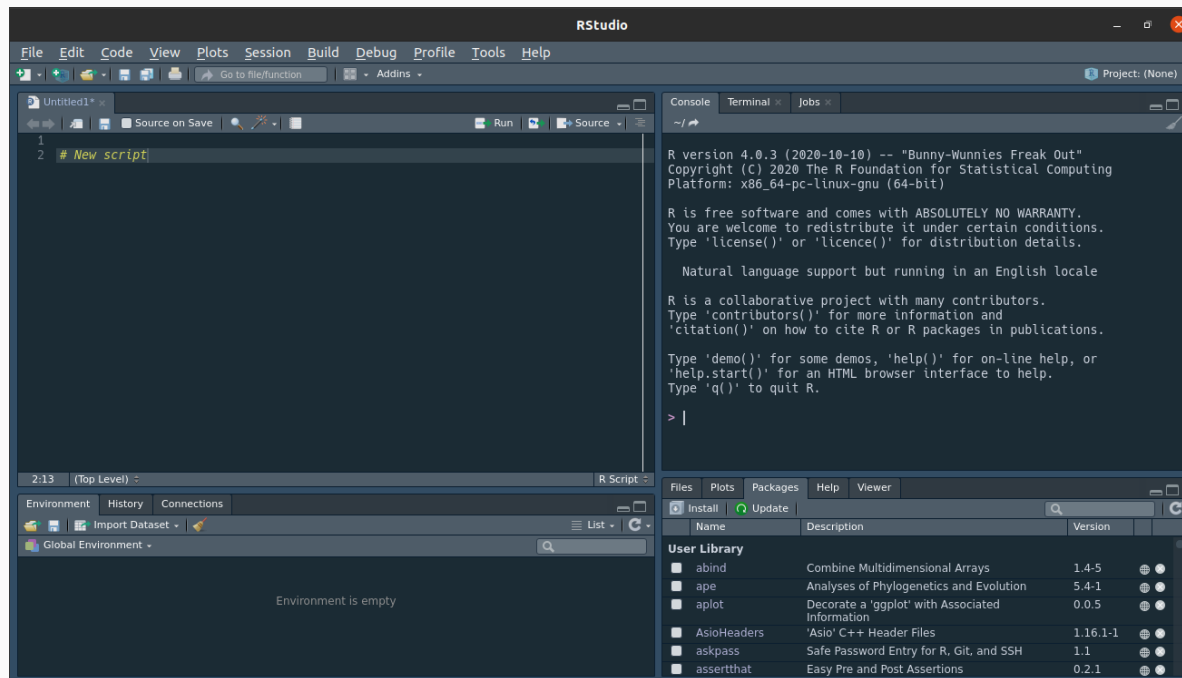


A screenshot of an R terminal window. The window has a dark gray title bar with standard Linux window controls (minimize, maximize, close) and a search icon. The title text reads "usuario@INSPIRON5559: ~". The terminal area has a dark purple background. The prompt ">" is followed by the command "q()", which has been executed. Below the command, a message "Save workspace image? [y/n/c]:" is displayed, followed by a white cursor character. A vertical scrollbar is visible on the right side of the terminal window.

```
> q()  
Save workspace image? [y/n/c]:
```

Rstudio

- <https://rstudio.com>.
- Rstudio tiene cuatro áreas principales:
 1. Scripts.
 2. Consola/Terminal/Jobs.
 3. Environment/History/Connections.
 4. Files/Plots/Packages/Help/Viewer.



Algunos atajos útiles en Rstudio

Shortcut	Descripción
Alt + -	Assignment operator, <-.
Ctrl + Shift + M	Pipe operator, %>%.
Ctrl + L	Limpiar terminal.
Ctrl + Enter	Correr línea, chunk o selección de código.
Ctrl + Shift + F10	Reiniciar sesión de R.
Ctrl + Shift + S	Source (correr todo el script).
Ctrl + Shift + C	Comentar línea.

Además de análisis estadísticos y gráficas...

- Sitios web (<https://bookdown.org/yihui/blogdown/>).
- Libros online (<https://bookdown.org/>).
- Presentaciones (<https://github.com/yihui/xaringan>).
- Aplicaciones web (<https://www.shinyapps.io/>).
- ... y mucho más.

Actividades recomendadas

- Realizar la sesión muestra del apéndice A en el documento introductorio a R en su página oficial: <https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>.
- Instalar el paquete Tidyverse; desde R ejecutar:
`install.packages("tidyverse")`.
- Instalar el paquete *palmerpenguins*:
`(install.packages("palmerpenguins"))`.