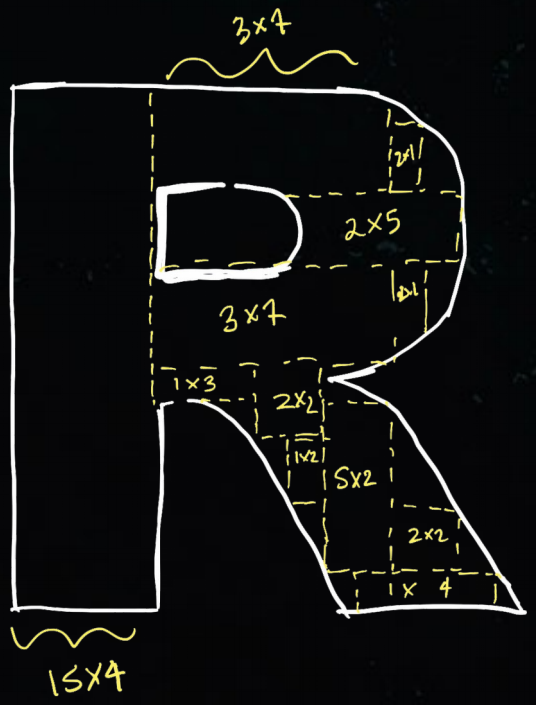
Funciones y paquetes

Pepi Amodeo
Instituto Argentino de Oceanografía
CONICET-UNS
CCT Bahía Blanca



Funciones

Funciones

f(argumento1 = ... , argumento 2 = ...)

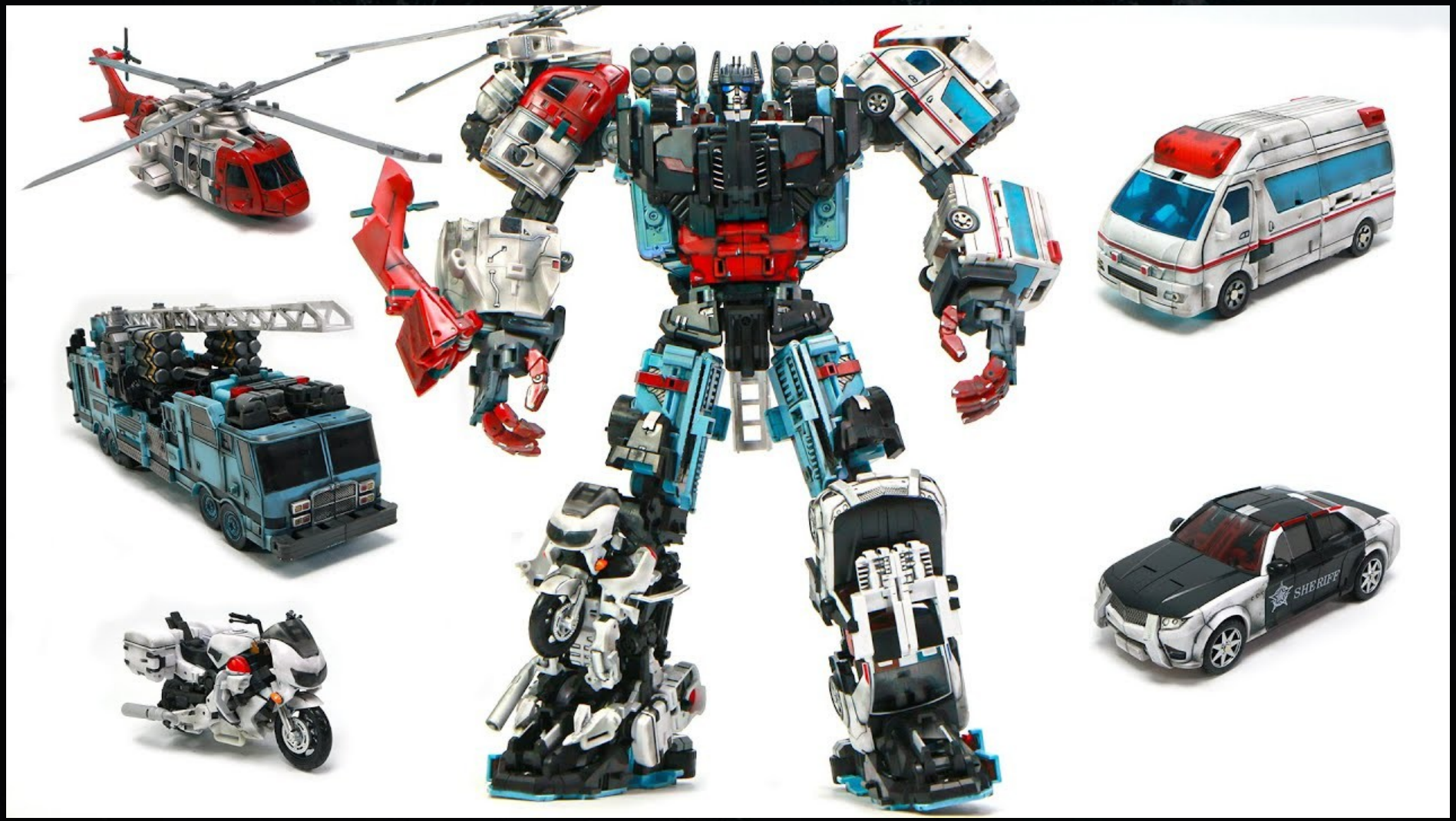
Funciones

```
log(2)
log10(2)

log(x=2)
log10(x=2)
```

Los argumentos se pueden explicitar siempre
Pero para agilizar se pueden indicar directamente (sin =)... ojo! R asume...

Funciones



Paquetes

Agregan funciones al R base

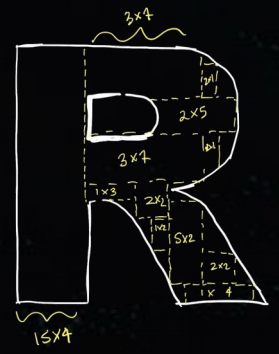
Paquetes

1997: 12 paquetes
2022: 18728 paquetes

Paquetes

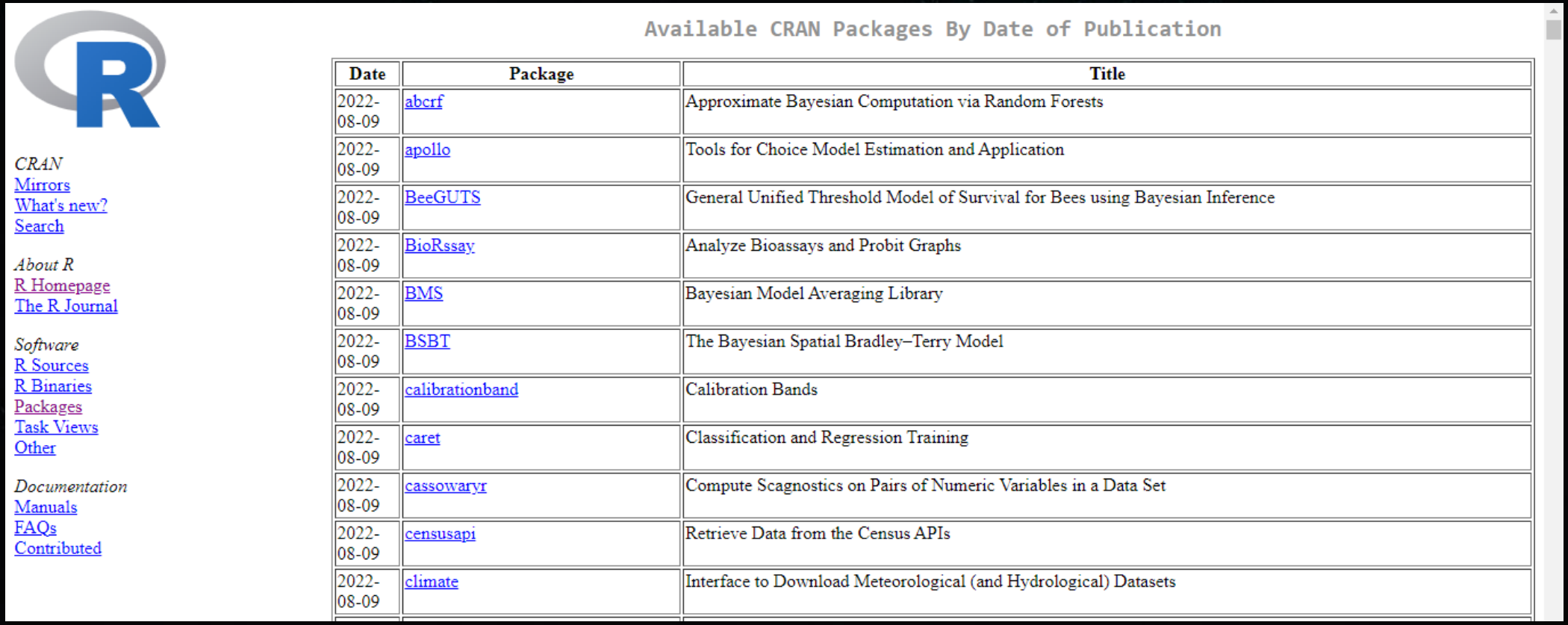
- Open-source. Los generan los mismos usuarios (libre)
- Incorporan nuevas funcionalidades al R base
- Hay redundancias
- CRAN unifica y organiza los paquetes oficiales
- Existen paquetes en desarrollo o desarrollos “no oficiales” (github, dropbox, de mano en mano)

usuario/a



desarrollador/a

CRAN



CRAN
Mirrors
What's new?
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

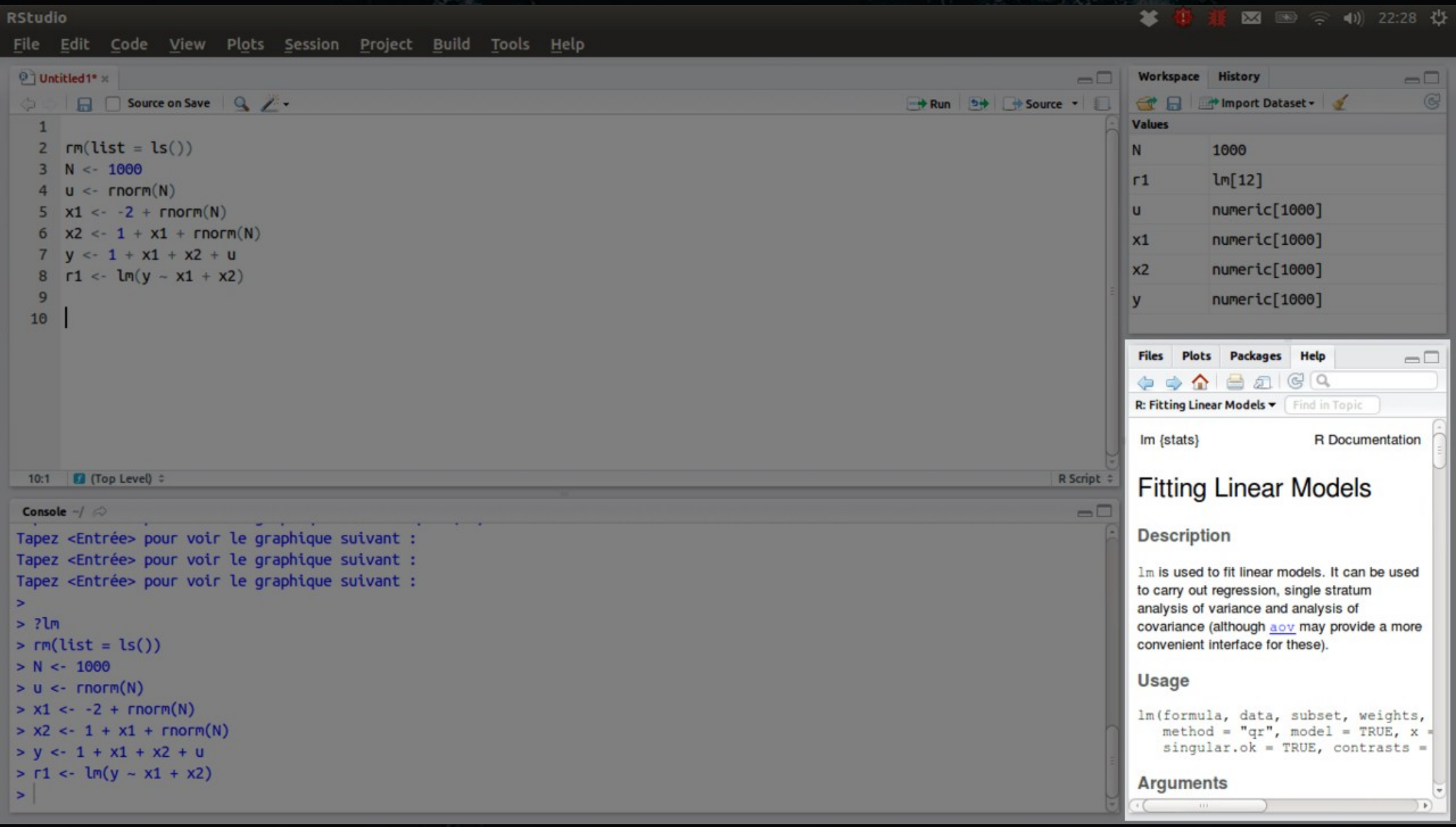
Documentation
Manuals
FAQs
Contributed

Available CRAN Packages By Date of Publication

Date	Package	Title
2022-08-09	abcrf	Approximate Bayesian Computation via Random Forests
2022-08-09	apollo	Tools for Choice Model Estimation and Application
2022-08-09	BeeGUTS	General Unified Threshold Model of Survival for Bees using Bayesian Inference
2022-08-09	BioRssay	Analyze Bioassays and Probit Graphs
2022-08-09	BMS	Bayesian Model Averaging Library
2022-08-09	BSBT	The Bayesian Spatial Bradley–Terry Model
2022-08-09	calibrationband	Calibration Bands
2022-08-09	caret	Classification and Regression Training
2022-08-09	cassowaryr	Compute Scagnostics on Pairs of Numeric Variables in a Data Set
2022-08-09	censusapi	Retrieve Data from the Census APIs
2022-08-09	climate	Interface to Download Meteorological (and Hydrological) Datasets

Repositorio donde se alojan todos los desarrollos "oficiales" relacionados con el proyecto R

Instalación paquetes



Instalación vs Activación

- install..packages("ggplot2")
- Instalado en la pc
- Queda disponible pero inactivo
- Única vez

- library("ggplot2")
- En cada sesión
- Tiene que estar instalado previamente
- Cada vez que lo quiero usar

Instalación paquetes

- Forma reproducible (en el script)

  install..packages("ggplot2")

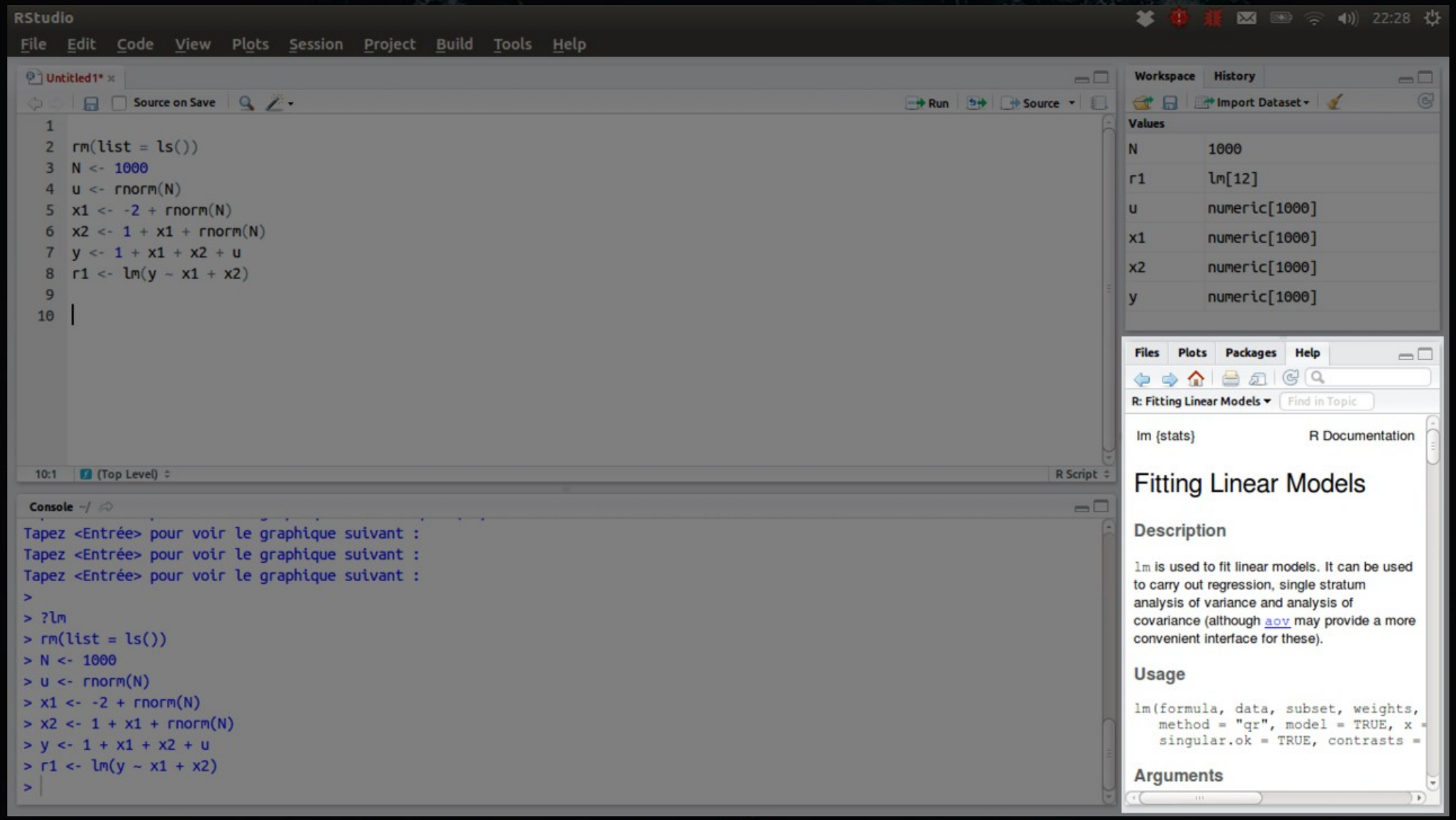
- Forma NO reproducible (en el panel)

  Panel / packages / install

- A mano (zip)

  Alternativa cuando falla CRAN

Ayuda



- ?mifuncion()
- help()
- En el Panel / Help / Buscar

ó F1

```
help("+")
```

Arithmetic Operators

Description

These unary and binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %% y
x %/% y
```

Arguments

`x, y`	numeric or complex vectors or objects which can be coerced to such, or other objects for which methods have been written.

Details

The unary and binary arithmetic operators are generic functions: methods can be written for them individually or via the `Ops` group generic function. (See `Ops` for how dispatch is computed.)

If applied to arrays the result will be an array if this is sensible (for example it will not if the recycling rule has been invoked).

Logical vectors will be coerced to integer or numeric vectors, `FALSE` having value zero and `TRUE` having value one.

`1 ^ y` and `y ^ 0` are `1`, *always*. `x ^ y` should also give the proper limit result when either (numeric) argument is infinite (one of `Inf` or `-Inf`).

Objects such as arrays or time-series can be operated on this way provided they are conformable.

For double arguments, `%%` can be subject to catastrophic loss of accuracy if `x` is much larger than `y`, and a warning is given if this is detected.

`%%` and `x %/% y` can be used for non-integer `y`, e.g. `1 %/% 0.2`, but the results are subject to representation error and so may be platform-dependent. Because the IEC 60559 representation of `0.2` is a binary fraction slightly larger than `0.2`, the answer to `1 %/% 0.2` should be `4` but most platforms give `5`.

Users are sometimes surprised by the value returned, for example why `(-8)^(1/3)` is `NaN`. For double inputs, `R` makes use of IEC 60559 arithmetic on all platforms, together with the C system function `pow` for the `^` operator. The relevant standards define the result in many corner cases. In particular, the result in the example above is mandated by the C99 standard. On many Unix-alike systems the command `man pow` gives details of the values in a large number of corner cases.

Arithmetic on type double in `R` is supposed to be done in 'round to nearest, ties to even' mode, but this does depend on the compiler and FPU being set up correctly.

Value

Unary + and unary − return a numeric or complex vector. All attributes (including class) are preserved if there is no coercion: logical `x` is coerced to integer and names, dims and dimnames are preserved.

S4 methods

These operators are members of the S4 `Arith` group generic, and so methods can be written for them individually as well as for the group generic (or the `Ops` group generic), with arguments `c(e1, e2)` (with `e2` missing for a unary operator).

Implementation limits

`R` is dependent on OS services (and they on FPUs) for floating-point arithmetic. On all current `R` platforms IEC 60559 (also known as IEEE 754) arithmetic is used, but some things in those standards are optional. In particular, the support for *denormal* aka *subnormal* numbers (those outside the range given by `.Machine`) may differ between platforms and even between calculations on a single platform.

Another potential issue is signed zeroes: on IEC 60559 platforms there are two zeroes with internal representations differing by sign. Where possible `R` treats them as the same, but for example direct output from C code often does not do so and may output `-0.0` (and on Windows whether it does so or not depends on the version of Windows). One place in `R` where the difference might be seen is in division by zero: `1/x` is `Inf` or `-Inf` depending on the sign of zero `x`. Another place is `identical(0, -0, num.eq = FALSE)`.

Note

All logical operations involving a zero-length vector have a zero-length result.

The binary operators are sometimes called as functions as e.g. `` `&`(x, y) ``: see the description of how argument-matching is done in `Ops`.

`**` is translated in the parser to `^`, but this was undocumented for many years. It appears as an index entry in Becker *et al* (1988), pointing to the help for `Deprecated` but is not actually mentioned on that page. Even though it had been deprecated in S for 20 years, it was still accepted in `R` in 2008.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

D. Goldberg (1991). What Every Computer Scientist Should Know about Floating-Point Arithmetic. *ACM Computing Surveys*, **23**(1), 5–48. doi: 10.1145/103162.103163.
Also available at https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.

For the IEC 60559 (aka IEEE 754) standard: https://www.iso.org/standard/57469.html and https://en.wikipedia.org/wiki/IEEE_754.

See Also

`sqrt` for miscellaneous and `Special` for special mathematical functions.

`Syntax` for operator precedence.

`%*%` for matrix multiplication.

Examples

```
x <- -1:12
x + 1
2 * x + 3
x %% 2 #-- is periodic
x %/% 5
x %% Inf # now is defined by limit (gave NaN in earlier versions of R)
```

Operadores básicos

Operadores aritméticos

Arithmetic {base} R Documentation

Arithmetic Operators

Description

These unary and binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %% y
x %/% y
```

help("+")
Panel/Help: Artihmetic operators

Operadores básicos

Funciones matemáticas

```
sin(1)  # trigonometry functions
```

```
[1] 0.841471
```

```
log(1)  # natural logarithm
```

```
[1] 0
```

```
log10(10) # base-10 logarithm
```

```
[1] 1
```

```
exp(0.5) # e^(1/2)
```

```
[1] 1.648721
```

Salida de web blog

Operadores básicos

Operadores relacionales

```
1 == 1  # equality (note two equals signs, read as "is equal to")
```

```
[1] TRUE
```

```
1 != 2  # inequality (read as "is not equal to")
```

```
[1] TRUE
```

```
1 <  2  # less than
```

```
[1] TRUE
```

```
1 <= 1  # less than or equal to
```

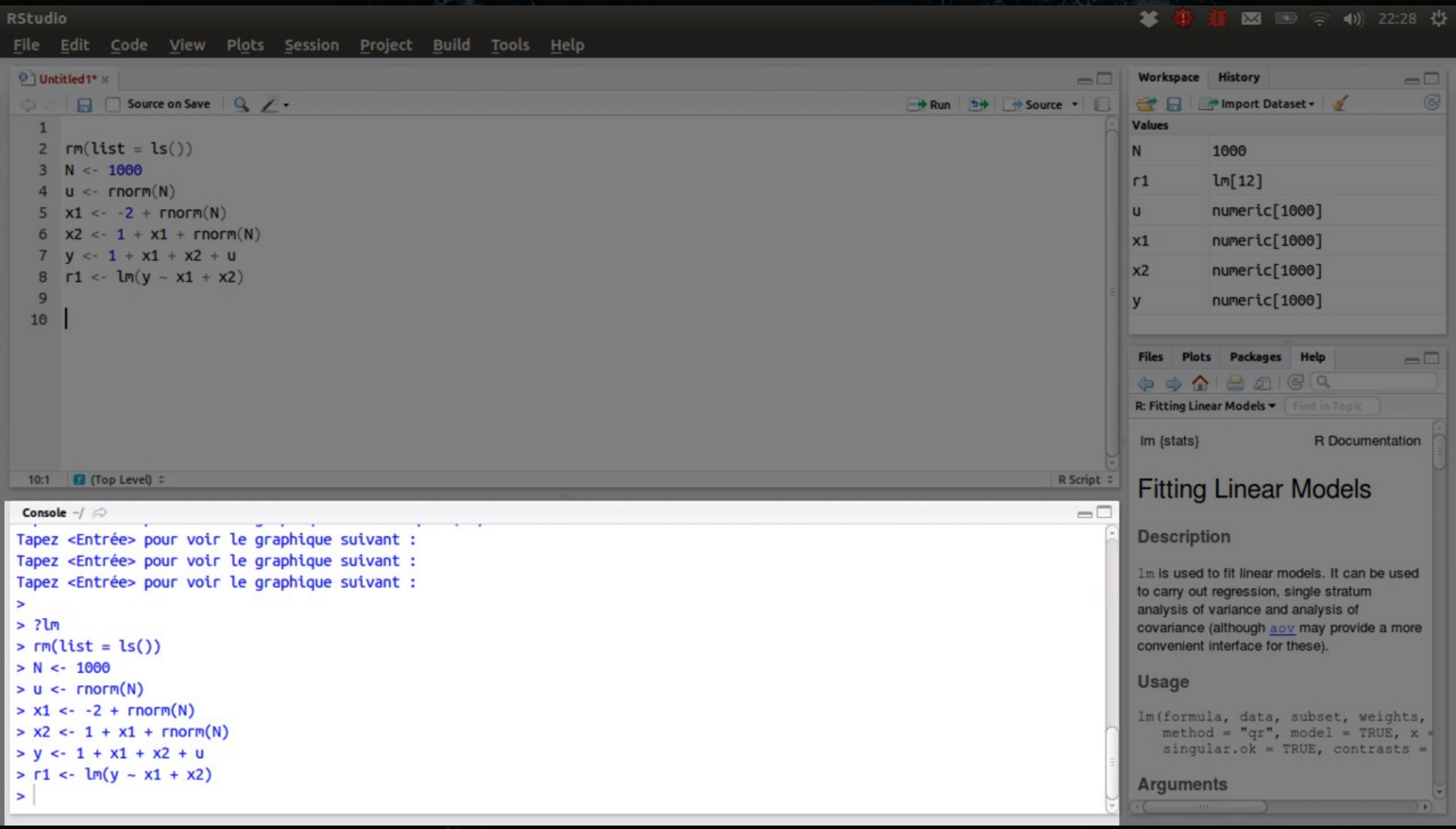
```
[1] TRUE
```

```
1 > 0  # greater than
```

```
[1] TRUE
```

Evalúa una condición y devuelve TRUE/FALSE

Consola



A través de ella se da la interacción con "el autómata".
Mandamos órdenes, vuelven mensajes y salidas

Consola

- Símbolo STOP
- Mensajes

Consola

Console | Terminal | Jobs

~/Documents/MarPackProjTest/

```
  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.

> plumber::plumb(file='MarNewPlumbAPIfile/plumber.R')$run()
Starting server to listen on port 8095
Running the swagger UI at http://127.0.0.1:8095/__swagger__/
```

El STOP indica que está procesando.
Pulsar para interrumpir el proceso... a veces es mejor tener paciencia...

Consola

- Message

  Tranqui... no es un error... es solo un autómata tratando de expresarse

```
> message("esto es un mensaje")
esto es un mensaje
```

- Warning

  Tranqui... no es un error... nos avisa algo por las dudas

```
> library(cowsay)
Warning message:
package ‘cowsay’ was built under R version 4.1.3
```

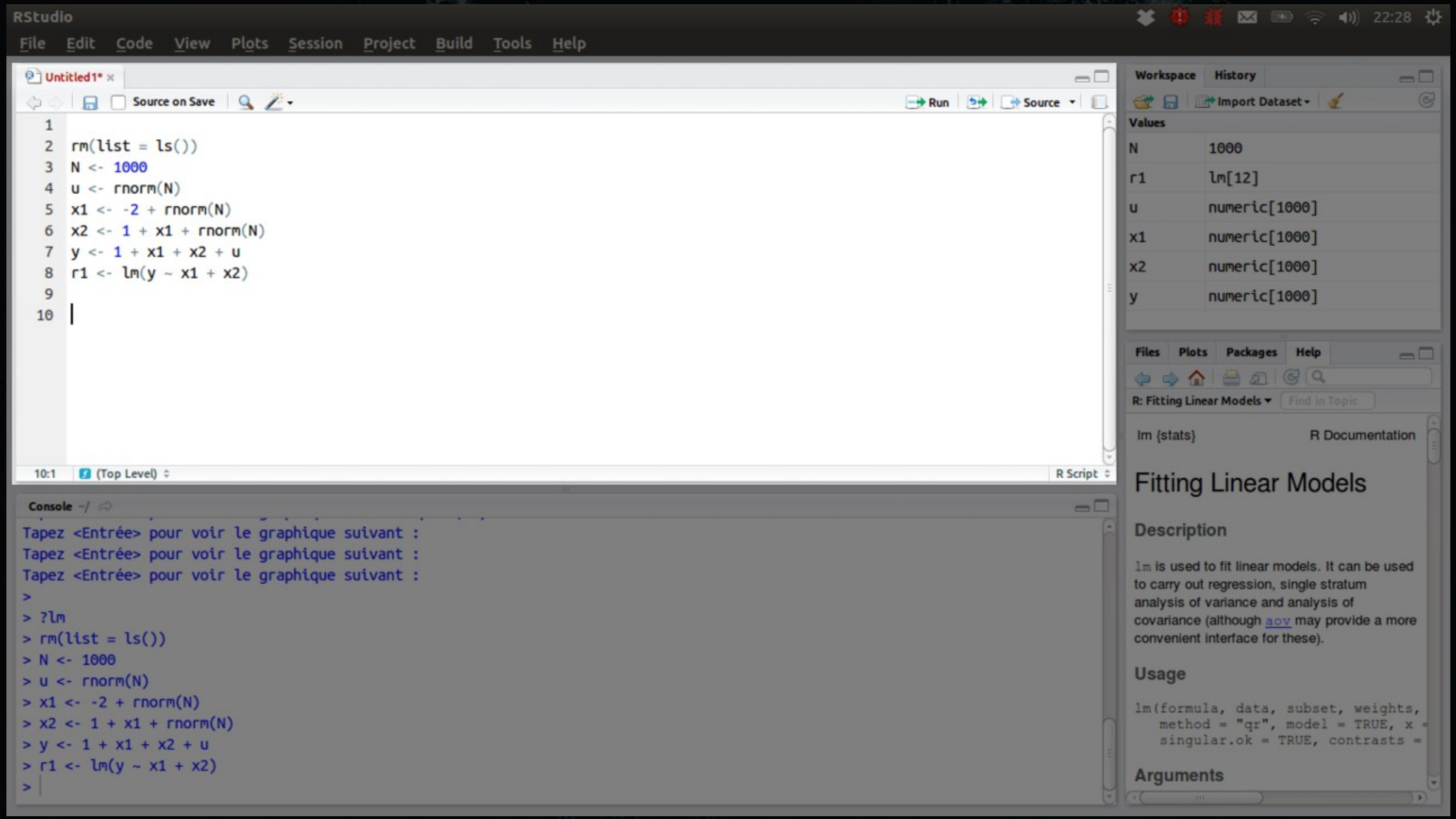
- Error

  Ahora sí se pudrió todo!

```
> say()
Error in say() : could not find function "say"
```

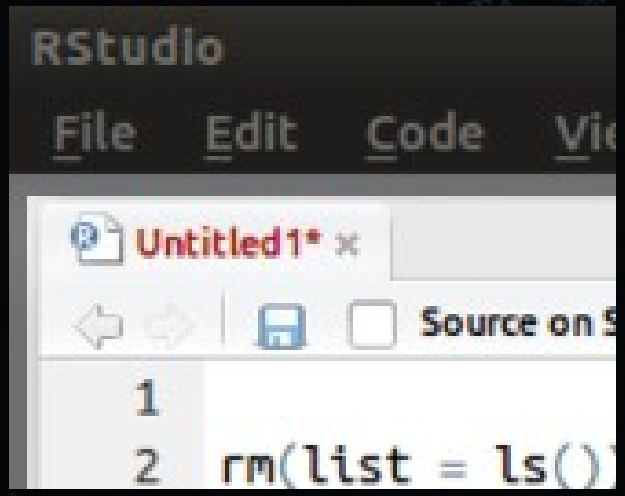
```
> mean(c(uno,2,3))
Error in mean(c(uno, 2, 3)) : object 'uno' not found
```

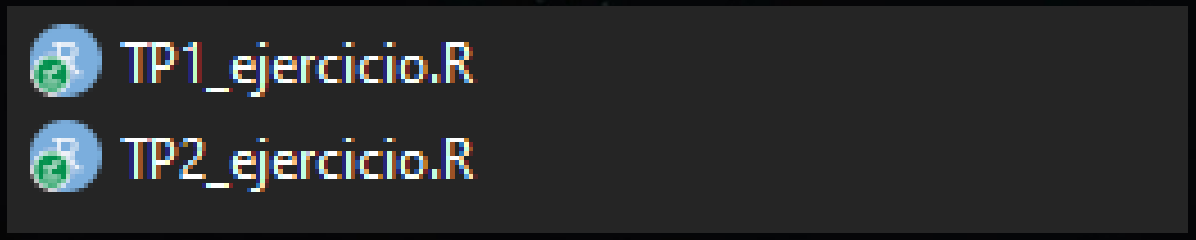
Editor de Scripts



Es donde genero el documento (LO MÁS IMPORTANTE)
Interactúa con la consola enviando línea por línea (CTRL+ENTER)
Con # hago comentarios (lineas que no se ejecutan)

Editor de Scripts





Archivo .R (texto plano)
¡¡¡LO MÁS IMPORTANTE!!!