

DEPARTAMENTO DE CIENCIAS DE LA ATMÓSFERA Y LOS OCÉANOS

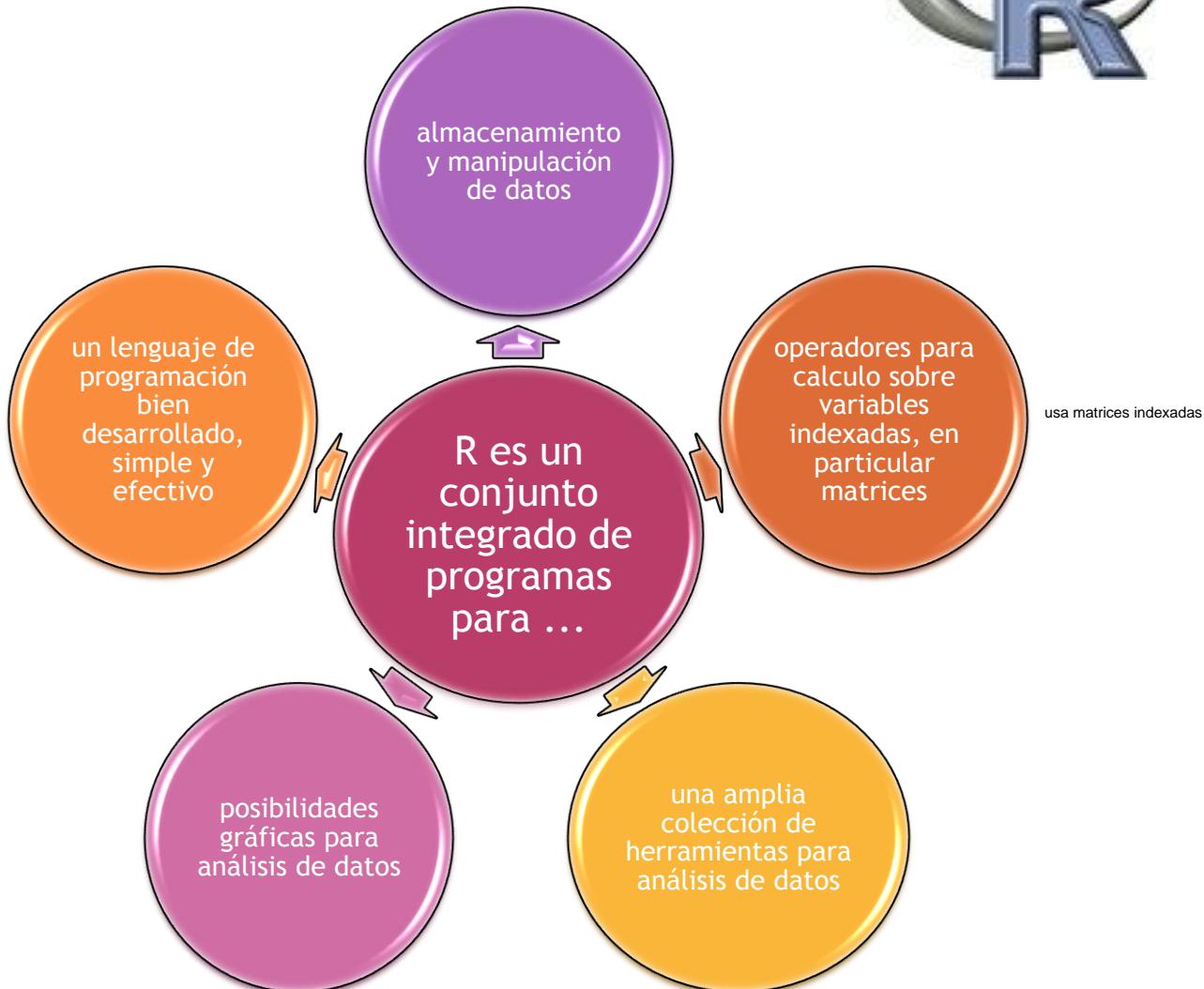
LABORATORIO DE
PROCESAMIENTO DE
INFORMACIÓN
METEOROLÓGICA/
OCEANOGRÁFICA

Lenguaje de Programación R

ELEMENTOS BÁSICOS

- Introducción
- Familiarización con el ambiente de trabajo
- Concepto de script
- Variables de tipo numéricas
- Precisión y formato
- Concepto de Not-a-Number
- Variables de tipo carácter
- Operadores aritméticos
- Funciones matemáticas básicas

¿QUÉ ES R?



¿QUÉ ES R?



R es un entorno de software libre para **cálculos estadísticos** y gráficos.

Se lo puede compilar y correr en una variedad de plataformas UNIX, Windows y MacOS

En la instalación de R se incluyen aproximadamente 25 paquetes llamados “standard” o “recomendados” pero hay más de 1500 librerías disponibles

<https://www.r-project.org/>

¿CÓMO PODEMOS TRABAJAR EN R?



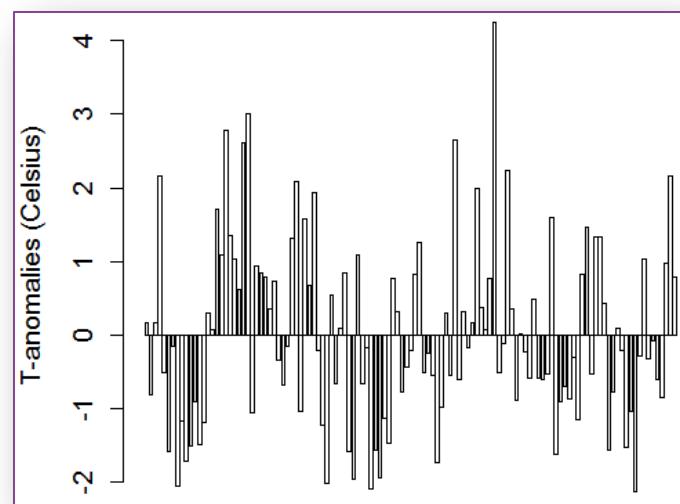
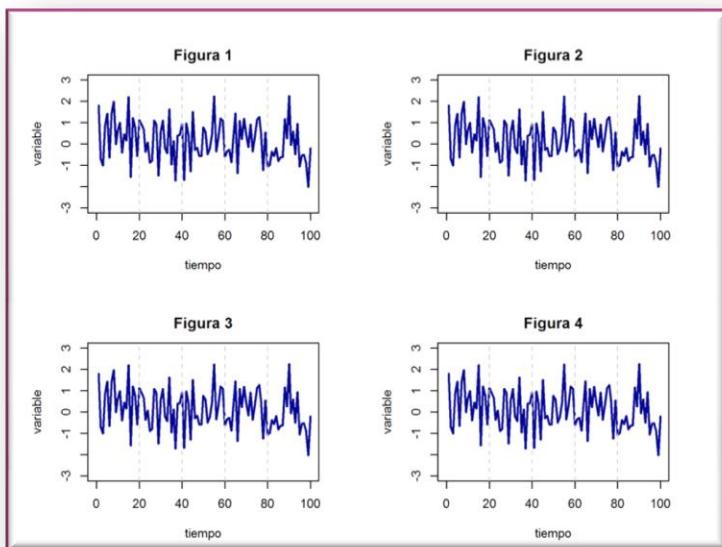
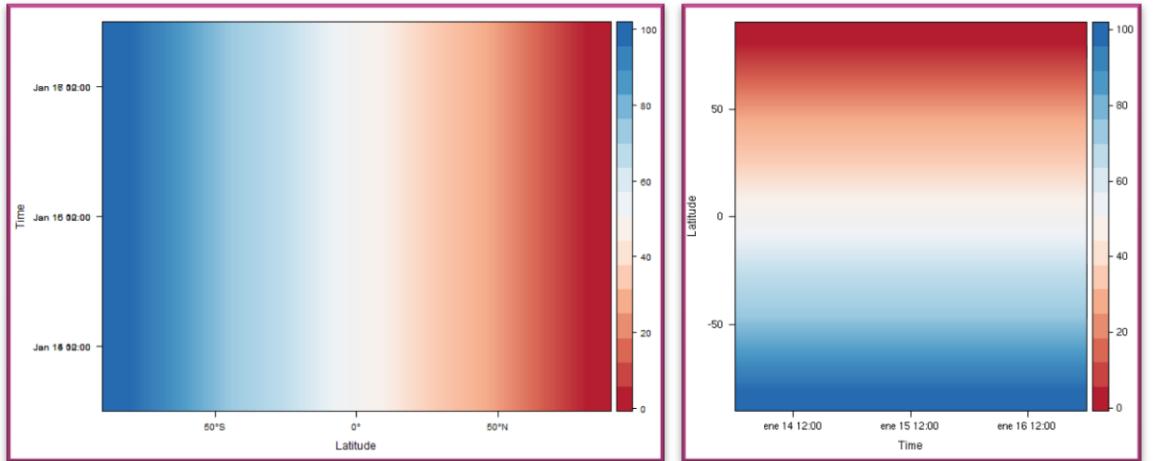
Observaciones



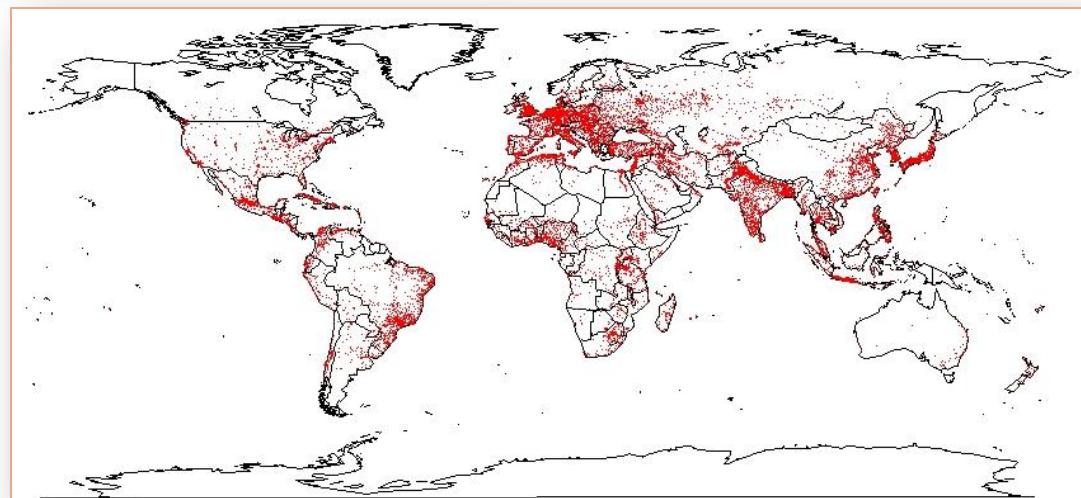
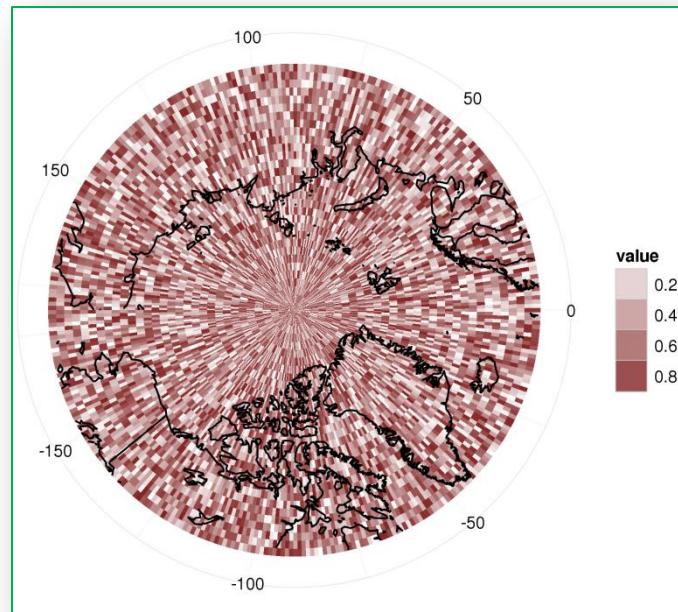
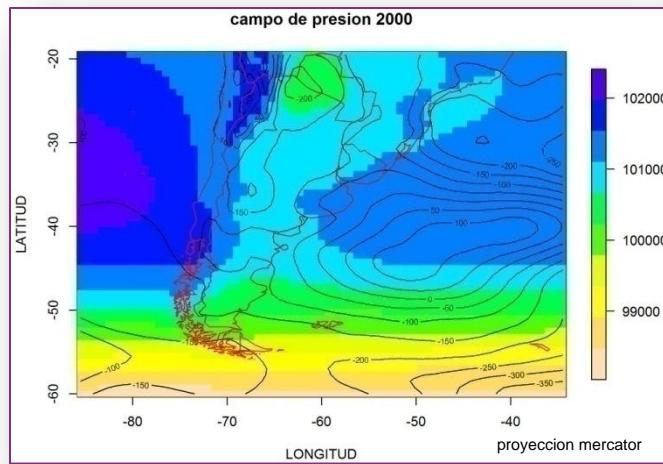
Salidas de Modelos
Numéricos



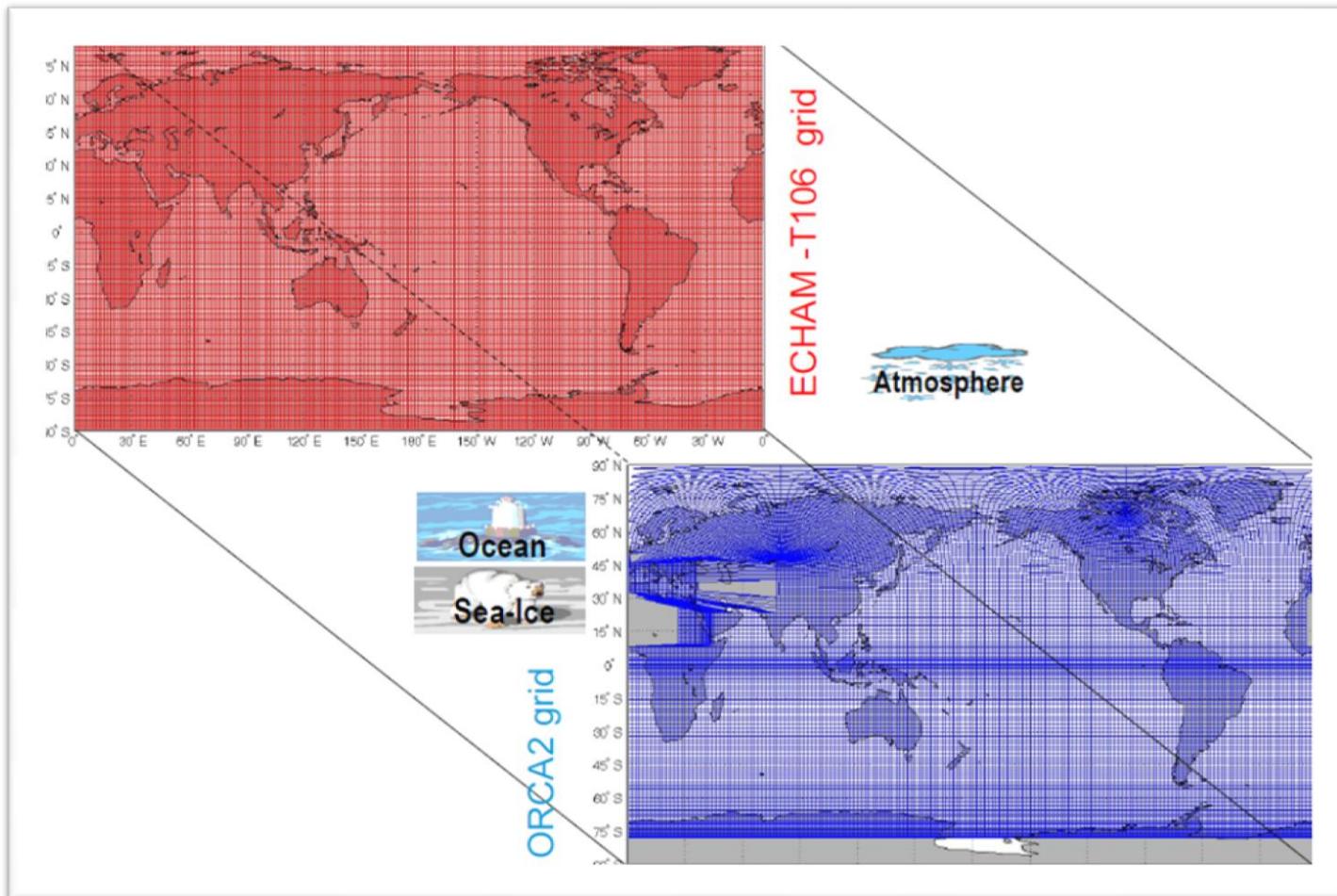
VISUALIZACIÓN SERIES TEMPORALES



VISUALIZACIÓN CAMPOS

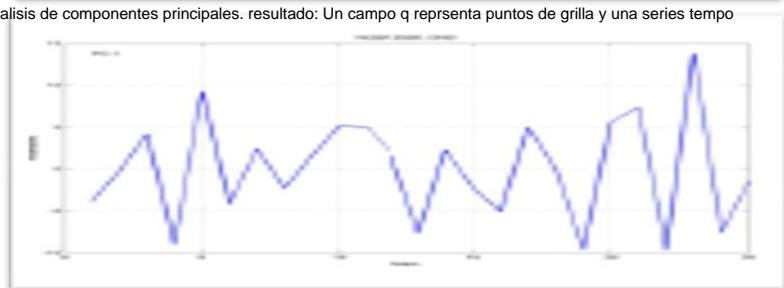
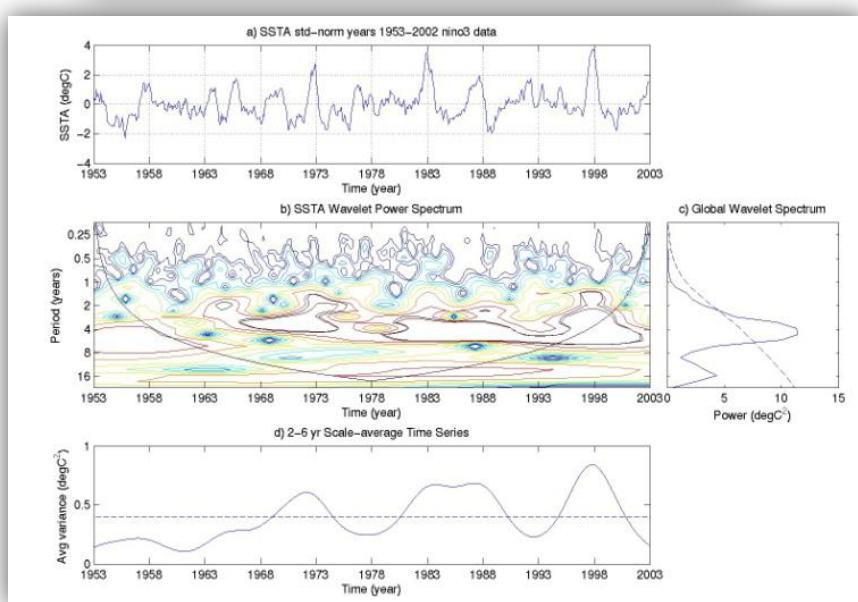
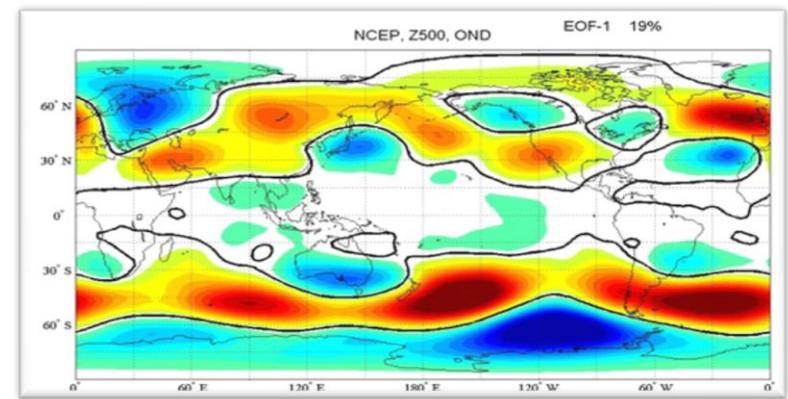
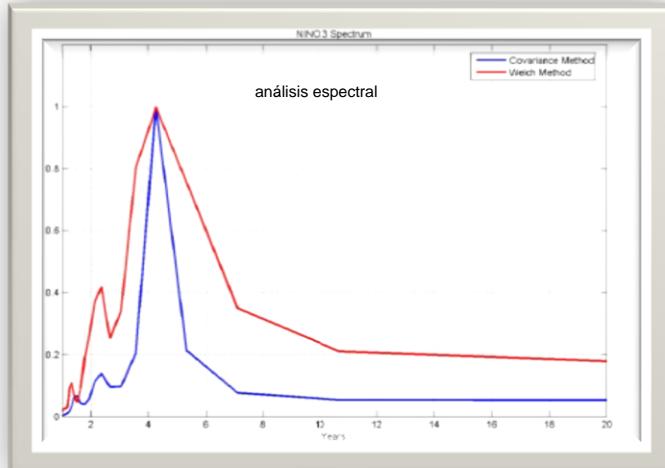
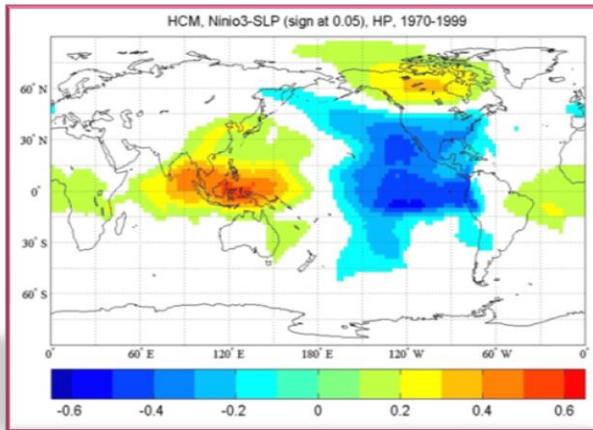


INTERPOLACIÓN DE CAMPOS



simples: correlación entre dos variables

ESTADÍSTICAS SIMPLES A COMPLEJAS



AMBIENTE DE TRABAJO



R Rterm (32-bit)

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-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.

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.

> -
```

DEMO ()



Demos in package 'base':

error.catching	More examples on catching and handling errors
is.things	Explore some properties of R objects and is.FOO() functions. Not for newbies!
recursion	Using recursion for adaptive integration
scoping	An illustration of lexical scoping.

Demos in package 'graphics':

Hershey	Tables of the characters in the Hershey vector fonts
Japanese	Tables of the Japanese characters in the Hershey vector fonts
graphics	A show of some of R's graphics capabilities
image	The image-like graphics builtins of R
persp	Extended persp() examples
plotmath	Examples of the use of mathematics annotation

Demos in package 'grDevices':

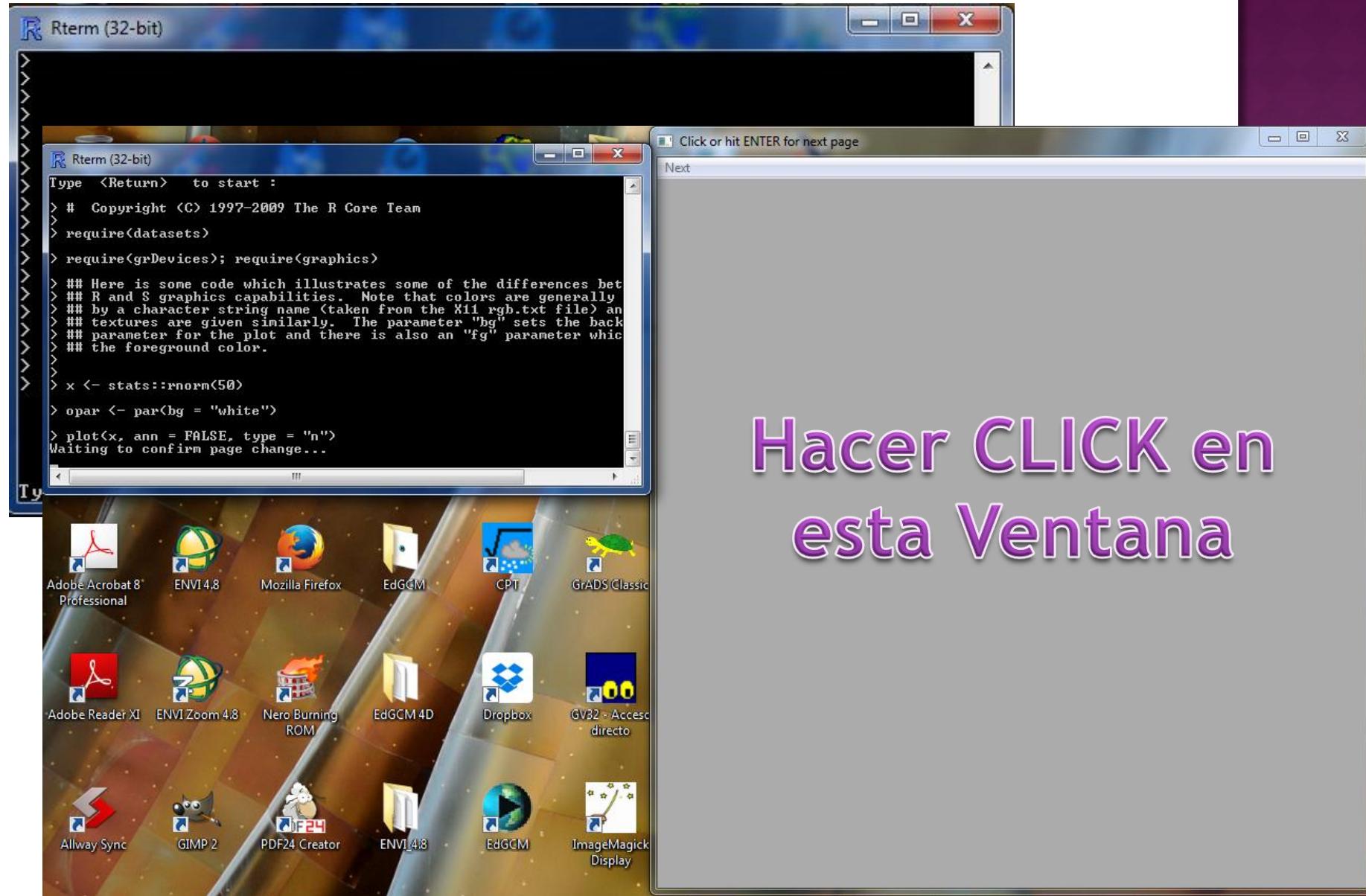
colors	A show of R's predefined colors()
hclColors	Exploration of hcl() space

Demos in package 'stats':

glm.vr	Some glm() examples from V&R with several predictors
lm.glm	Some linear and generalized linear modelling examples from 'An Introduction to Statistical Modelling' by Annette Dobson
nlm	Nonlinear least-squares using nlm()
smooth	'Visualize' steps in Tukey's smoothers

Use 'demo(package = .packages(all.available = TRUE))'
to list the demos in all *available* packages.

DEMO (GRAPHICS)



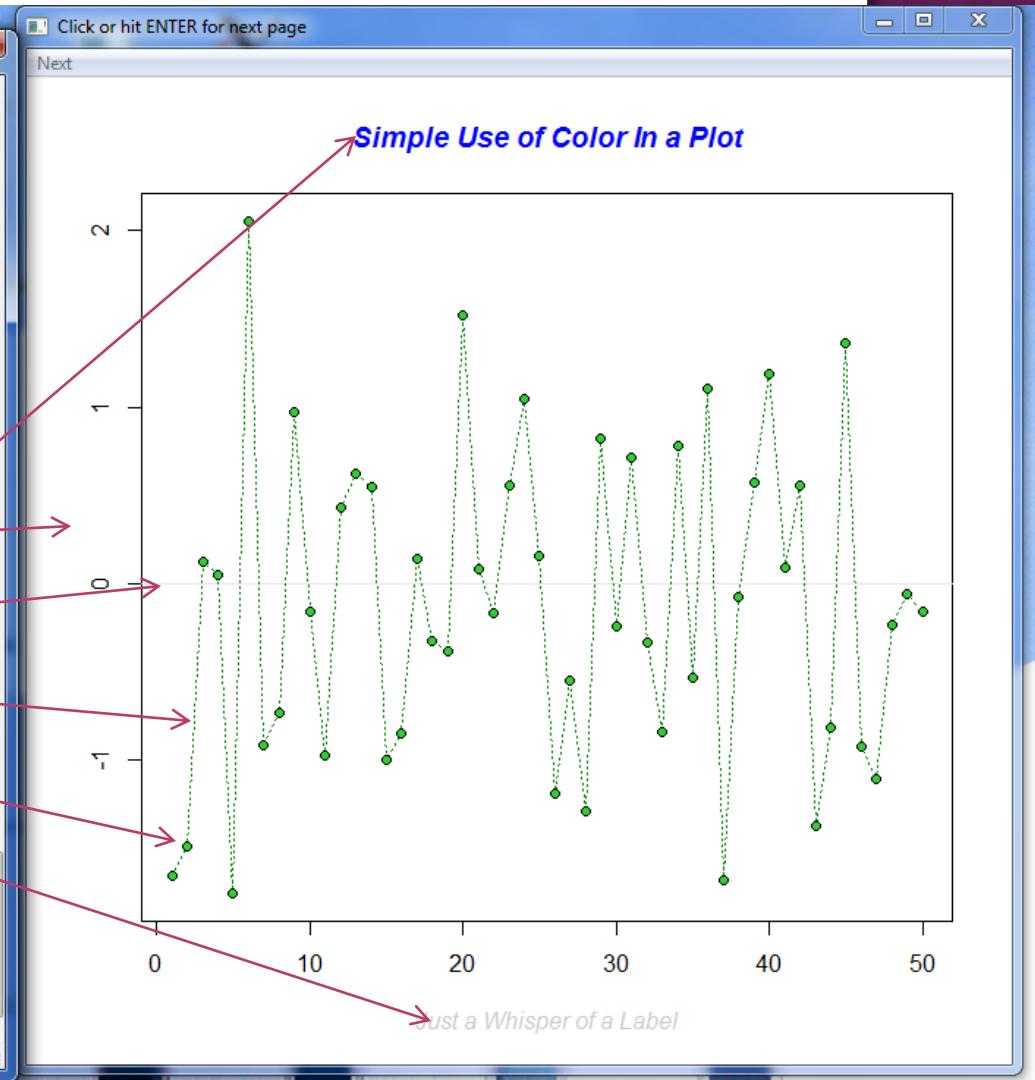
DEMO (GRAPHICS) CONT.

```
Rterm (32-bit)
>
> demo(graphics)

  demo(graphics)
  ---- ~~~~~~
Type <Return> to start :
> # Copyright <C> 1997–2009 The R Core Team
> require(datasets)
> require(grDevices); require(graphics)
> ## Here is some code which illustrates some of the differences bet
> ## R and S graphics capabilities. Note that colors are generally
> ## by a character string name (taken from the X11 rgb.txt file) an
> ## textures are given similarly. The parameter "bg" sets the back
> ## parameter for the plot and there is also an "fg" parameter which
> ## the foreground color
>
> x <- stats::rnorm(50)
> opar <- par(bg = "white")
> plot(x, ann = FALSE, type = "n")
Waiting to confirm page change...
> abline(h = 0, col = gray(.90))
> lines(x, col = "green4", lty = "dotted")
> points(x, bg = "limegreen", pch = 21)
> title(main = "Simple Use of Color In a Plot",
+       xlab = "Just a Whisper of a Label",
+       col.main = "blue", col.lab = gray(.8),
+       cex.main = 1.2, cex.lab = 1.0, font.main = 4, font.lab = 3)
> ## A little color wheel. This code just plots equally spaced
> ## a pie chart. If you have a cheap SUGA monitor (like me) you
> ## probably find that numerically equispaced does not mean visually
> ## equispaced. On my display at home, these colors tend to cluster
> ## the RGB primaries. On the other hand on the SGI Indy at work the
> ## effect is near perfect.
>
> par(bg = "gray")
> pie(rep(1,24), col = rainbow(24), radius = 0.9)
Waiting to confirm page change...
```

Genera los datos a graficar

Grafico



DEMO (GRAPHICS) CONT.

Seguir el demo en R

¿.... Y AHORAAAAA?



The screenshot shows the RStudio interface with several panels:

- Scripts_Curso.R** (Script Editor): A code editor window containing R script code. The code reads multiple NetCDF files, processes them, and generates a postscript file. It includes comments explaining the steps.
- Environment** (Data View): A table showing the global environment variables and their values. It lists objects like dd, x, x1, x2, x3, a, b, b1, c, c1, and c2 with their respective characteristics.
- Console**: The R command-line interface showing the R version, copyright information, and standard welcome message.

INSTALACION

- **Instalar primero R**
[\(https://cran.r-project.org/\)](https://cran.r-project.org/)
- **Instalar Rstudio**
[\(https://www.rstudio.com/products/rstudio/download/\)](https://www.rstudio.com/products/rstudio/download/)
- **Instalar LIBRERIAS necesarias desde Rstudio**

INSTALACION LIBRERIAS

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help.
- Project:** Project: (None)
- Code Editor:** Script titled "Scripts_Curso.R" containing R code for reading netCDF files and calculating monthly mean temperatures. A specific line, `> install.packages("nombre de la libreria")`, is highlighted with a red oval.
- Environment Tab:** Shows objects in the Global Environment, including `dd`, `x`, `x1`, `x2`, `x3`, and values `a` through `c2`.
- Packages Tab:** Shows a list of installed packages with their descriptions and versions. The "Plots" tab is also circled in purple.
- Console:** Displays the R startup message, license information, and the command `> install.packages("nombre de la libreria")`.
- Taskbar:** Includes icons for Windows, Internet Explorer, File Explorer, Firefox, and R.
- System Tray:** Shows the date (04/01/2016) and time (12:56).

ALGUNAS LIBRERÍAS

- `ggplot2`
- `ncdf4`
- `Fields`
- `Mapdata`
- `Akima`
- `MASS`
- `lmomco`

LUGAR DE TRABAJO

Para encontrar el directorio de trabajo en que nos encontramos se utiliza el comando:

```
>getwd()
```

El comando:

```
>setwd("Mi directorio de trabajo")
```

Permite en cambio cambiar el directorio

Para ser más ordenado puedo crear distintas carpetas

```
WORKDIR <- "D:/Moira/PCFacultad/R_course/"  
SOURCES <- "D:/Moira/PCFacultad/R_course/Sources/"  
DATA   <- "D:/Moira/PCFacultad/R_course/Data/"  
OUTPUTS <- "D:/Moira/PCFacultad/R_course/Outputs/"
```



HELP

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Scripts_Curso.R * Go to file/function

Project: (None)

Environment History Import Dataset Global Environment

Data

dd	36525 obs. of 6 variables
x	4440 obs. of 1874 variables
x1	4440 obs. of 1872 variables
x2	4440 obs. of 1272 variables
x3	Large matrix (5647680 elements, 43.1 Mb)

values

a	-56.75
b	-20.25
b1	601L
c	280.25
c1	1872L
c2	1272

Files Plots Packages Help Viewer

The R Language Find in Topic

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export

The R Language Definition
R Installation and Administration
R Internals

Reference

Packages Search Engine & Keywords

Miscellaneous Material

About R Authors Resources
License Frequently Asked Questions Thanks
NEWS User Manuals Technical papers

Material specific to the Windows port

CHANGES up to R 2.15.0 Windows FAQ

Icons: Windows, Internet Explorer, File, Print, Firefox, Notepad, R

2do Cuatrimestre 2023 - Moira E. Doyle

HELP (CONT)

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help.
- Left Panel:** Scripts_Curso.R (R Script), Console (R version 3.2.2 output).
- Right Panel:** Environment tab (Data and values), Global Environment, Help tab (Generic X-Y Plotting), Viewer tab (plot [graphics]).
- Bottom Bar:** Taskbar icons (Windows, Internet Explorer, File Explorer, Firefox, Notepad, R), system tray (Speaker, Battery, Date/Time).

In the console, the command `> help("plot")` is shown in red, with a yellow arrow pointing from the text to the opening double quotes of the help command. The output of the command is the help documentation for the `plot` function.

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for statistical computing
Platform: i386-w64-mingw32/i386 (32-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.

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.

[workspace loaded from ~/RData]
> help("plot")
"plot"
```

Help Documentation for plot:

Description
Generic function for plotting of R objects. For more details about the graphical parameter arguments, see [par](#).

For simple scatter plots, [plot.default](#) will be used. However, there are plot methods for many R objects, including [functions](#), [data.frames](#), [density](#) objects, etc. Use [methods\(plot\)](#) and the documentation for these.

Usage
`plot(x, y, ...)`

Arguments

- x the coordinates of points in the plot. Alternatively, a single plotting structure, function or any R object with a [plot method](#) can be provided.
- y the y coordinates of points in the plot, optional if x is an appropriate structure.
- ... Arguments to be passed to methods, such as [graphical parameters](#) (see [par](#)). Many methods will accept the following arguments:
 - type

HELP (CONT)

- `help.search("plot")`
- `vignette()` lista de ar
- `vignette("maplevels")`

The image shows two windows side-by-side. On the left is a PDF viewer window titled "mapLevels.pdf - Adobe Reader" displaying the "Mapping levels of a factor" vignette from the gdata package. The vignette includes a title, author information, introduction, and a "Description with examples" section containing R code. On the right is an "R: Search Results" window showing the search results for "plot". Both windows have toolbars and status bars.

R: Search Results

Search Results

MAPPING LEVELS OF A FACTOR

Mapping levels of a factor

The `gdata` package
by Gregor Gorjanc

Introduction

Factors use levels attribute to store information on mapping between internal integer codes and character values i.e. levels. First level is mapped to internal integer code 1 and so on. Although some users do not like factors, their use is more efficient in terms of storage than for character vectors. Additionally, there are many functions in base R that provide additional value for factors. Some of these need to work with internal integer codes and mapping levels back to factor, especially when interfacing external programs. Mapping information is also of interest if there are many factors that should have the same set of levels. This note describes `mapLevels` function, which is an utility function for mapping the levels of a factor in `gdata` package (Warnes, 2006).

Description with examples

Function `mapLevels()` is an (S3) generic function and works on factor and character atomic classes. It also works on list and `data.frame` objects with previously mentioned atomic classes. Function `mapLevels` produces a list so that it can work with names and values. Names are levels, while values can be internal integer codes or (possibly other) levels. This will be clarified later on. Class of this "map" is `levelsMap`, if `x` in `mapLevels()` was atomic or `listLevelsMap` otherwise - for list and `data.frame` classes. The following example shows the creation and printout of such a "map".

```
> library(gdata)
> (fac <- factor(c("B", "A", "Z", "D")))
[1] B A Z D
Levels: A B D Z
> (map <- mapLevels(x=fac))
A B D Z
1 2 3 4
```

If we have to work with internal integer codes, we can transform factor to integer and still get "back the original factor" with "map" used as argument in `mapLevels<-` function as shown below. `mapLevels<-` is also an (S3) generic function and works on same classes as `mapLevels` plus integer atomic class.

¹from version 2.3.1

Up to now examples showed "map" with internal integer codes for values and levels for names. I call this invalid because the "map" function does "map" levels for values and (possibly other) levels for names. This feature is a bit odd at first sight, but can be used to easily unify levels and internal integer codes across several factors. Imagine you have a factor that is for some reason split into two factors `f1` and `f2` and that each factor does not have all levels. This is not uncommon situation.

HTML [source](#) [R code](#)

HTML [source](#) [R code](#)

PDF [source](#) [R code](#)

PDF [source](#) [R code](#)

HELP (CONT)

example (“plot”)

The screenshot shows the RStudio interface with the following components:

- Top Bar:** View, Plots, Session, Build, Debug, Tools, Help.
- File Explorer:** Scripts_Curso.R, R vignettes (multiple tabs).
- Console:** Displays R code related to plotting and statistical functions.
- Environment:** Shows the global environment with variables like stdt, tas, tas_1960_2010, time, title1, title2, title3, threshold, ul, uu, winsize, WORKDIR, and x.
- Plots:** A plot of the sine function $\sin(x)$ against x , showing a periodic wave from -2 to 6 on the x-axis and -1 to 1 on the y-axis.
- Bottom Bar:** Taskbar with icons for various applications.

HELP (CONT)

`apropos("plot")`

```
[1] ".__c__recordedplot" "assocplot" "barplot"  
[4] "barplot.default" "biplot" "boxplot"  
[7] "boxplot.default" "boxplot.matrix" "boxplot.stats"  
[10] "cdplot" "coplot" "fourfoldplot"  
[13] "interaction.plot" "lag.plot" "matplot"  
[16] "monthplot" "mosaicplot" "plot"  
[19] "plot.default" "plot.design" "plot.ecdf"  
[22] "plot.function" "plot.new" "plot.spec.coherency"  
[25] "plot.spec.phase" "plot.stepfun" "plot.ts"  
[28] "plot.window" "plot.xy" "plotx"  
[31] "pplots" "preplot" "qqplot"  
[34] "recordPlot" "replayPlot" "savePlot"  
[37] "screeplot" "spineplot" "sunflowerplot"  
[40] "termplot" "ts.plot"
```

Busca en R funciones cuyo nombre contengan lo escrito entre paréntesis y comillas, en este caso, la palabra “*plot*”

HELP - RSEEK.ORG (MOZILLA)

Inbox (98) - mail... (13) CIMA Web... 10 horas de ... help in R - Busc... r faq - How to g... Getting help on ... Pagina de inform... Rseek.org

rseek.org lecture 5 trevor hastings Más visitados Comenzar a usar Firefox Galería de Web Slice Sitios sugeridos Suggested Sites Web Slice Gallery

plot

All Articles Books Support Packages For Beginners

About 84,700,000 results (0.39 seconds) Sort by: Relevance powered by Google™ Custom Search

Quick-R: Density Plots
www.statmethods.net/graphs/density.html
The option freq=FALSE plots probability densities instead of frequencies. The option breaks= controls the number of bins. # Simple Histogram hist(mtcars\$mpg).
Labeled [For Beginners](#) [Support](#)

Quick-R: Line Charts
www.statmethods.net/graphs/line.html
By default, plot() plots the (x,y) points. Use the type="n" option in the plot() command, to create the graph with axes, titles, etc., but without plotting the points.
Labeled [For Beginners](#) [Support](#)

Quick-R: Mosaic Plots
www.statmethods.net/advgraphs/mosaic.html
Extended mosaic and association plots are described here. Each provides a method of visualizing complex data and evaluating deviations from a specified ...
Labeled [For Beginners](#) [Support](#)

Quick-R: Combining Plots
www.statmethods.net/advgraphs/layout.html
Combining Plots. R makes it easy to combine multiple plots into one overall graph, using either the par() or layout() function. With the par() function, you can ...
Labeled [For Beginners](#) [Support](#)

Quick-R: Bar Plots
www.statmethods.net/graphs/bar.html
Bar Plots. Create barplots with the barplot(height) function, where height is a vector or matrix. If height is a vector, the values determine the heights of the bars in ...
Labeled [For Beginners](#) [Support](#)

Quick-R: Dot Plots
www.statmethods.net/graphs/dot.html
Dot Plots. Create dotplots with the dotchart(x, labels=) function, where x is a numeric vector and labels is a vector of labels for each point. You can add a groups= ...
Labeled [For Beginners](#) [Support](#)

Quick-R: Probability Plots
www.statmethods.net/graphs/probability.html

Windows Taskbar: 14:44 01/01/2016

RSiteSearch()

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Scripts_Curso.R * R vignettes * R vignettes * R vignettes *

Vignettes in package 'sp':

```
csdacm      Customising spatial data classes and methods  
intro_sp     sp: classes and methods for spatial data (source, pdf)  
over         sp: overlay and aggregation (source, pdf)
```

Vignettes in package 'SparseM':

```
SparseM      An Introduction to the SparseM Package for Sparse Linear Algebra (source, pdf)
```

Vignettes in package 'stringr':

```
stringr     Introduction to stringr (source, html)
```

Vignettes in package 'survival':

```
adjcurve    Adjusted Survival Curves (source, pdf)  
tests       Cox models and ``type 3'' Tests (source, pdf)  
compete     Multi-state models and competing risks (source, pdf)  
splines     Splines, plots, and interactions (source, pdf)  
timedep    Using Time Dependent Covariates (source, pdf)
```

Vignettes in package 'utils':

```
Sweave      Sweave User Manual (source, pdf)
```

Console ~ /

```
[1] ".c_recordeuplot" "assocplot" "parplot"  
[4] "barplot.default" "biplot"    "boxplot"  
[7] "boxplot.default" "boxplot.matrix" "boxplot.stats"  
[10] "cdplot"        "coplot"    "fourfoldplot"  
[13] "interaction.plot" "lag.plot" "matplot"  
[16] "monthplot"    "mosaicplot" "plot"  
[19] "plot.default"  "plot.design" "plot.ecdf"  
[22] "plot.function" "plot.new"   "plot.spec.coherency"  
[25] "plot.spec.phase" "plot.stepfun" "plot.ts"  
[28] "plot.window"   "plot.xy"    "plotx"  
[31] "pplots"        "preplot"   "qqplot"  
[34] "recordPlot"   "replayPlot" "savePlot"  
[37] "screeplot"    "spineplot" "sunflowerplot"  
[40] "termplot"     "ts.plot"   > rseek.org  
> Error: object 'rseek.org' not found  
> RSitesearch("plot")  
A search query has been submitted to http://search.r-project.org  
The results page should open in your browser shortly  
> RSitesearch("plot")  
A search query has been submitted to http://search.r-project.org  
The results page should open in your browser shortly  
>
```

Environment History

Global Environment

- stdt List of 1
- tas Large array (31334400 elements, 239.1 Mb)
- tas_1960_2010 Large array (31334400 elements, 239.1 Mb)
- time num [1:1956(1d)] 15.5 45 74.5 105 135.5 ...
- title1 "Plot of Ind143"
- title2 "Ind143"
- title3 "Years"
- threshold 5
- u1 0
- uu 25
- winsize 5
- WORKDIR "F:/Moira/R_course/"
- x num [1:47] -2.43 -1.57 -1.56 -1.35 -1.17 ...

Files Plots Packages Help Viewer

sin

x

RSITESEARCH

The screenshot shows a web browser window with the address bar containing 'search.r-project.org: plot'. This address is circled in green. The page displays search results for the term 'plot'. The search interface includes a search bar with 'plot', a 'Sort by: relevance' dropdown, a 'Results per Page: 20' dropdown, and search filters for 'R Manuals', 'Base Packages Help Pages', 'CRAN Task Views', 'CRAN Packages', 'Description', 'Help Pages', 'News', 'Readme', and 'Vignettes'. Below these filters are two radio buttons: 'Matching any words' (unchecked) and 'Matching all words' (checked), followed by the text '1-20 of about 120,000 matches'.

Term frequencies: **plot:** 124,996
Search took 0.009630 seconds

R: Plot a Tornado Plot object
Modified: 2023-03-18
Size: 2.7K
[/CRAN/refmans/tornado/html/plot.tornado.html](#)
cran-help matching: *plot, plot and plot*

R: Plot an Importance Plot object
Modified: 2023-03-18
Size: 2.9K
[/CRAN/refmans/tornado/html/plot.importance.html](#)
cran-help matching: *plot, plot and plot*

How to Present Tables in Plot Devices
Modified: 2022-10-12
Size: 210.0K
...to Present Tables in Plot Devices Peter Carl Chicago R User Group Meetup: R Output Peter Carl (PerformanceAnalytics, etc.) textplot June 2011 1 / 15 Outline 1 Overview 2 Example 3 Potential Solutions...
[/CRAN/packages/PerformanceAnalytics/vignettes/textplotPresentation-CRUG-2011.pdf](#)
cran-vignettes matching: *plot, plot and plot*

At the bottom, the taskbar shows various application icons and the system tray indicates the date and time as 09:19 a.m.

COMO EJECUTAR COMANDOS???

A través de un programa
Secuencia de comandos
Su extensión es .R
(ej. pirulo.R)

Se ejecuta como
un comando
source('pirulo.R')

En modo interactivo

Línea de comando
a=1
b=2
c=a+b

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help, and a search bar. The left pane contains a script editor titled 'Scripts_Curso.R' with R code. The right pane has tabs for Environment and History, showing global variables like stdt, tas, tas_1960_2010, time, title1, title2, title3, treshold, ul, uu, winsize, WORKDIR, and x. Below these are tabs for Files, Plots, Packages, Help, and Viewer. The bottom pane is the Console, which displays the output of running the script. A green arrow points from the 'En modo interactivo' text to the console, and an orange arrow points from the 'Línea de comando' text to the same area.

```
library(ncdf)
## Este comando setea el directorio de trabajo #####
setwd("/Users/gustavo/Desktop/curso_R_SMN/R_course/")
#####
open.ncdf("data/t2max_021_1950-2100_day_Debilt.nc")->nc
get.var.ncdf(nc,"t2max")->tday
open.ncdf("data/t2max_debilt_021-037_1950-2100.nc")->ncmax
get.var.ncdf(ncmax,"t2max")->tmax
open.ncdf("data/t2monthly_021_1950_Debilt.nc")->ncmonthly
get.var.ncdf(ncmonthly,"t2max")->tmonmean
open.ncdf("data/multi-year-mean-1950-1959.nc")->ncmulti
get.var.ncdf(ncmulti,"t2max")->multimean
multimean<- (multimean-273.15)
tmax<-tmax[1:151,1]
tday<-(tday-273.15)
tmax<-tmax-273.15
tmonmean<-(tmonmean-273.15)
time<-c(1:length(tday))
time<-time/366+1949
month<-c(1:length(tmonmean))/12+1949
year=c(1950:2100)
days<-c(1:length(tday))/365+1949
#skip postscript and dev.off everywhere
postscript("Debilit_daily_temp.ps")

```

Console ~ / ~

```
[1] > rseek.org
Error: object 'rseek.org' not found
> RSitesearch("plot")
A search query has been submitted to http://search.r-project.org
The results page should open in your browser shortly
> RSitesearch("plot")
A search query has been submitted to http://search.r-project.org
The results page should open in your browser shortly
> help(operators)
No documentation for 'operators' in specified packages and libraries:
you could try ??operators'
warning messages:
1: In .HTMLSearch(query) : Unrecognized search field: title
2: In .HTMLSearch(query) : Unrecognized search field: keyword
3: In .HTMLSearch(query) : Unrecognized search field: alias
> help("Arithmetic")
> library()
```

PROGRAMACIÓN (SCRIPT)

Ver Ejemplo1.R y Ejemplo2.R en Clases teóricas

El símbolo

Se utiliza
para escribir
comentarios

The screenshot shows the RStudio interface with the following components highlighted:

- Code Editor:** Shows the script file `ejemplo1.R` containing R code. The last line of code, `c`, is highlighted with an orange box and labeled "Ultima instrucción".
- Run Buttons:** A group of three buttons in the toolbar labeled "Run", "Source", and "Print" is highlighted with a purple oval.
- Environment View:** Shows the global environment with variables `a`, `b`, and `c` defined. Variable `c` has a value of 3, which is also highlighted with an orange box.
- Console View:** Shows the command `> source('C:/PCMoira/materias/SeminarioComputacion/Teoricas 2016/Ejemplos Teorica/ejemplo1.R')` and the resulting output `> c`. The variable `c` is highlighted with an orange box and labeled "Resultado ????".
- Help View:** A tooltip for the `S3 Group Generic Functions` is shown, detailing the description, usage, and arguments of the function.

PROGRAMACIÓN (SCRIPT)

Ver ejemplo1.R y ejemplo2.R en teóricas/ejemplos

The screenshot shows the RStudio interface with the following components:

- File Explorer:** Shows files "ejemplo1.R" and "ejemplo2.R".
- Source Editor:** Displays the content of "ejemplo2.R". The code uses `print` and `scan` functions to interact with the user for input values a and b, calculates their sum c = a + b, and prints the result.
- Environment View:** Shows the global environment with variables a, b, and c having values 7, 5, and 12 respectively.
- Console:** Shows the command `source('F:/Moira/materias/SeminarioComputacion/Teoricas 2016/Ejemplos Teorica/ejemplo2.R', echo=TRUE)` and its output, which includes the user interaction and the final result.
- Search Results:** A modal window titled "Search Results" with the search string "crt". It lists the help page for "graphics::par".

```
1 # ejemplo2
2
3 # idem ejemplo 1 pero usando comando print y scan
4
5
6 print('entre un valor para a (presione enter para terminar): '); a=scan()
7 print('entre otro valor para b: '); b=scan()
8 c=a+b;
9 print('la suma de a+b es')
10 print(c)
11
```

```
> source('F:/Moira/materias/SeminarioComputacion/Teoricas 2016/Ejemplos Teorica/ejemplo2.R', echo=TRUE)

> # ejemplo2
>
> # idem ejemplo 1 pero usando comando print y scan
>
> print('entre un valor para a (presione enter para terminar): '); a=scan()
[1] "entre un valor para a (presione enter para terminar)."
1: 7
2:
Read 1 item

> print('entre otro valor para b: '); b=scan()
[1] "entre otro valor para b: "
1: 5
2:
Read 1 item

> c=a+b;

> print('la suma de a+b es')
[1] "la suma de a+b es"

> print(c)
[1] 12
>
```

The "Search Results" window displays the following content:

The search string was "crt"

Help pages:

[graphics::par](#) Set or Query Graphical Parameters

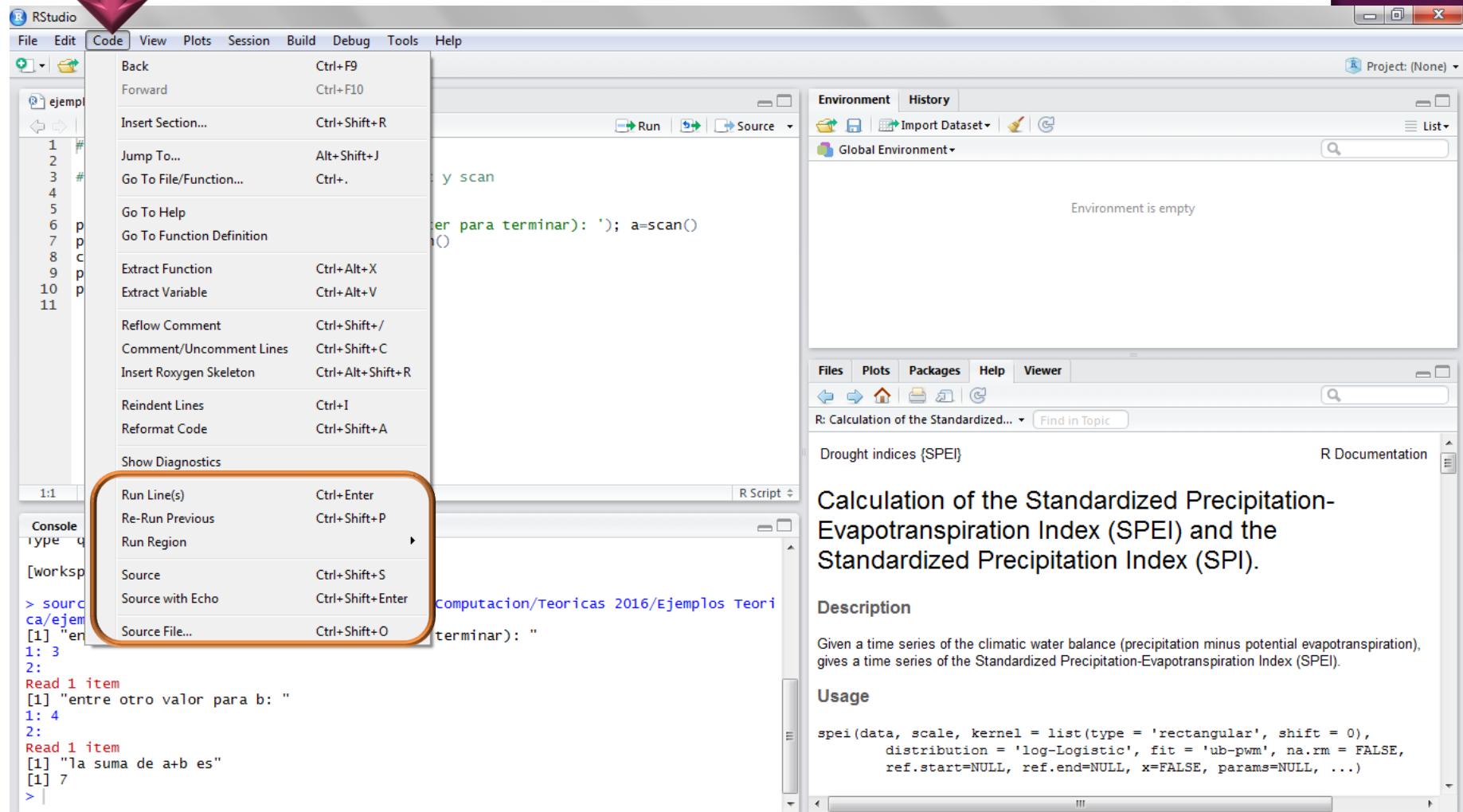
PROGRAMACIÓN (SCRIPT)

Ver ejemplo1.R y ejemplo2.R en teóricas/ejemplos

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays two files: "ejemplo1.R" and "ejemplo2.R". "ejemplo1.R" contains a simple script to calculate the sum of two numbers. "ejemplo2.R" is shown with its code highlighted.
- Environment:** Shows the global environment with variables a=6, b=4, and c=10.
- Console:** Shows the execution of "ejemplo2.R" with echo=FALSE, displaying the user input and the calculated sum.
- Help:** A help page for the "R" function is open, explaining how it reads code from a file or connection.
- Viewer:** Shows the help content for the "R" function, including sections on encodings and references.

COMANDOS DE EJECUCION (SCRIPT)



A screenshot of the RStudio interface. A large red arrow points down to the 'Code' menu item in the top navigation bar. The 'Code' menu is open, showing various options for running code. The 'Run Line(s)' option is highlighted with a red box. The main workspace shows some R code and its output. The bottom right corner displays the documentation for the 'spei' function.

RStudio

File Edit View Plots Session Build Debug Tools Help

Code

Back Ctrl+F9
Forward Ctrl+F10
Insert Section... Ctrl+Shift+R
Jump To... Alt+Shift+J
Go To File/Function... Ctrl+.
Go To Help
Go To Function Definition
Extract Function Ctrl+Alt+X
Extract Variable Ctrl+Alt+V
Reflow Comment Ctrl+Shift+/
Comment/Uncomment Lines Ctrl+Shift+C
Insert Roxygen Skeleton Ctrl+Alt+Shift+R
Reindent Lines Ctrl+I
Reformat Code Ctrl+Shift+A
Show Diagnostics

Run Line(s) Ctrl+Enter
Re-Run Previous Ctrl+Shift+P
Run Region
Source Ctrl+Shift+S
Source with Echo Ctrl+Shift+Enter
Source File... Ctrl+Shift+O

Project: (None)

Environment History

Global Environment

Environment is empty

Files Plots Packages Help Viewer

R: Calculation of the Standardized... Find in Topic

Drought indices (SPEI)

R Documentation

Calculation of the Standardized Precipitation-Evapotranspiration Index (SPEI) and the Standardized Precipitation Index (SPI).

Description

Given a time series of the climatic water balance (precipitation minus potential evapotranspiration), gives a time series of the Standardized Precipitation-Evapotranspiration Index (SPEI).

Usage

```
spei(data, scale, kernel = list(type = 'rectangular', shift = 0),
      distribution = 'log-Logistic', fit = 'ub-pwm', na.rm = FALSE,
      ref.start=NULL, ref.end=NULL, x=FALSE, params=NULL, ...)
```

COMANDOS DE EJECUCION (SCRIPT)

COMANDO	EJECUCION RAPIDA	DESCRIPCION
Run Line(s)	CTRL+ENTER	Ejecuta la linea donde estoy parado en el script o el conjunto de lineas seleccionadas (pintar) OJO puede dar error. Ejemplo 2
Re-Run-Previous	CTRL+SHIFT+P	Si estoy ejecutando por lineas, vuelve a ejecutar la ultima linea ejecutada
Run from Beginning to Line	CTRL+ALT+B	Ejecuta desde el inicio hasta la linea donde estoy parado
Run from Line to End	CTRL+ALT+E	Ejecuta desde donde estoy parado hasta el final OJO puede dar error. Ejemplo 2
Run Function Definition	CTRL+ALT+F	Carga en la memoria una funcion para usar
Run Code Section	CTRL+ALT+T	Ejecuta una seccion Ejemplo 1
Run All	CTRL+ALT+R	Ejecuta todo el script OJO puede dar error. Ejemplo 2

COMANDOS DE EJECUCION (SCRIPT)

The screenshot shows the RStudio interface with several annotations:

- A red arrow points from the "Code" menu in the top navigation bar to a callout box labeled "Defino una Sección o región del programa".
- Two red boxes highlight specific sections of code in the script editor:
 - Line 5: `#####`
 - Line 10: `#####`
- A callout box labeled "Ejecuto usando Run Code Section" has arrows pointing to both of the highlighted code blocks.
- In the bottom left, the R Console shows the execution of the script, resulting in the value 5.
- A callout box labeled "Resultado???" is positioned over the console output.
- The right side of the interface includes the Environment pane showing variables like a=3, at.y, b=2, c=5, Distance, e.y, and entrada; and the Help pane for the apropos function.

COMANDOS DE EJECUCION (SCRIPT)

COMANDO	EJECUCION RAPIDA	DESCRIPCION
Source	CTRL+SHIFT+S	Ejectuta todo el Script SIN mostrar las lineas de programa
Source with Echo	CTRL+SHIFT+ENTER	Ejectuta todo el Script mostrando en la consola las lineas de programa
Source File	CTRL+SHIFT+O	Abre una ventana para seleccionar el archivo que quiero correr sin abrirlo

OJO puede dar error.

Estos comandos se pueden ejecutar en la consola y en R

Los anteriores solo se pueden ejecutar en Rstudio y a través del menú o la ejecución rápida

PROGRAMACIÓN (SCRIPT/PROGRAMA)

Un script es un archivo “externo” que contiene una secuencia de comandos

- o Son archivos-R (nombre.R) y se ejecutan tipeando source(path/nombre.R por línea de comando
 - o Son útiles para automatizar cálculos
 - o Operan con datos pre-existentes cargados en el workspace o crean/leen nuevos datos
- o No retornan argumentos; todas las variables quedan cargadas en el workspace luego de su ejecución
 - o Eventualmente producen graficos y/o graban datos
 - o No requieren declaraciones o delimitadores de inicio y fin
- o Pueden contener comentarios (el carácter indicador de comentario es #)

Consejo: escriba todos sus comandos en un script.

PROGRAMACION

Como trabaja R

- Programas
- Funciones



- Ejecución de:
- funciones
 - comandos en línea
 - programas



- Resultados:
- Gráficos
 - Numéricos

Se pueden escribir con cualquier editor de texto

Deben estar en la sintaxis que entiende R

R: las variables se cargan en ***Environment***.

La cantidad de datos a tratar y rapidez de los cálculos dependen de la computadora en que se corre R.

Una de las principales ventajas de usar R es que existen miles de funciones que son compartidas por usuarios de todo el mundo y todo en forma gratuita

FUNCIONES MATEMATICAS

OPERADORES ARITMÉTICOS

HELP S4groupGeneric

*Si escriben en el help **S4groupGeneric** obtendran informacion sobre los distintos operadores que estan definidos en R*

+ x	devuelve el valor positivo de x
- x	devuelve el valor <i>negativo</i> de x
x + y	suma
x - y	resta
x * y	multiplicación
x / y	división
x ^ y	exponenciación
x %% y	x mod y, el resto de dividir x/y
x %/% y	indica división entera

Ejemplo

x=3

y=6.5

Aplicar los distintos operadores definidos

y%/%x Que resultados dan los dos últimos?

[1] 2

y%%x

[1] 0.5

FUNCIONES MATEMATICAS BASICAS

Funciones Hiperbólicas

(x *valor numerico o complejo*)

cosh(x)

sinh(x)

tanh(x)

acosh(x)

asinh(x)

atanh(x)

Trigonométricas (x)

cos(x)

sin(x)

tan(x)

acos(x)

asin(x)

atan(x)

atan2(y, x)

cospi(x) es equivalente cos(pi*x)

sinpi(x)

tanpi(x)

Funciones Matemáticas Varias

abs(x) valor absoluto

sqrt(x) raíz cuadrada

Función sign

sign(x) devuelve un vector con los signos de los correspondientes elementos de x (el signo de un real es 1, 0, or -1 positivo, zero, negativo).

Funciones de Redondeo

ceiling(x) redondea hacia arriba.

floor(x) redondea hacia abajo.

trunc(x) redondea a cero. Toma la parte entera y descarta decimales. saca los decimales

round(x, digits=) redondeo de acuerdo a los decimales de "digits=". 0.5 → 1

signif (x, digits=) redondeo de acuerdo al numero de digitos de "digits="

FUNCIONES MATEMATICAS BASICAS

Funciones Acumulativas

Devuelve un vector cuyos elementos son la suma, el producto, el minimo o el maximo de los elementos del argumento

```
c(1:10)  
[1] 1 2 3 4 5 6 7 8 9 10
```

```
cumsum(1:10)      Suma en forma acumulada los valores de 1 a 10  
[1] 1 3 6 10 15 21 28 36 45 55
```

```
cumprod(1:10)  
[1] 1 2 6 24 120 720 5040 40320 362880 3628800
```

```
c(3:1, 2:0, 4:2)  
[1] 3 2 1 2 1 0 4 3 2
```

```
cummax(c(3:1, 2:0, 4:2))  
[1] 3 3 3 3 3 3 4 4 4
```

```
cummin(c(3:1, 2:0, 4:2))  
[1] 3 2 1 1 1 0 0 0 0
```

comparo
elemento
con
elemento

Logaritmos y Exponenciales

$\log(x)$ es logaritmo natural

$\log(x, \text{base} = \exp(1))$

$\log_b(x, \text{base} = \exp(1))$

es igual a \log

$\log_{10}(x)$

$\log_2(x)$

$\log1p(x)$ es equivalente a $\log(1+x)$

$\exp(x)$

$\expm1(x)$ es equivalente a $e^x - 1$

FUNCIONES MATEMATICAS BASICAS

max(..., na.rm = FALSE)	compara todos los elementos y me devuelve solo un valor
min(..., na.rm = FALSE)	
min(5:1, pi)	devuelve un único valor
[1] 1	
pmax(..., na.rm = FALSE)	
pmin(..., na.rm = FALSE)	
pmin(5:1, pi)	devuelve un vector, aca de 5 componentes
[1] 3.141593 3.141593 3.000000 2.000000 1.000000	
range(5:1)	devuelve el máximo y el mínimo del vector
[1] 1 5	
prod(1:7)	devuelve el producto de todos los miembros del vector
[1] 5040	
sum(1:7)	devuelve la suma de todos los miembros del vector
[1] 28	
any	Dado un conjunto de vectores lógicos ve si alguno de los valores es verdadero
all	Dado un conjunto de vectores lógicos ve si todos los valores son verdaderos

LOS DATOS Y SUS TIPOS

OBJETOS

Todas las cosas que manipula R se llaman **objetos**. En general, éstos se construyen a partir de objetos más simples. Los objetos más simples son de cinco clases a las que se denomina **atómicas** y que son las siguientes:

por default R te lo pone como número real

- **character** (cadenas de caracteres)
 - **numeric** (números reales)
 - **integer** (números enteros)
 - **complex** (números complejos)
 - **logical** (lógicos o booleanos, sólo toman los valores True o False)
- 
- los datos numéricos

Cuando se introduce algo que puede interpretar como un número, su inclinación es tratarlo como un dato de tipo **numeric**, es decir, un número de **tipo real**, a no ser que explícitamente se indique otra cosa.

ASIGNACIÓN

La función principal para definir un objeto es mediante el comando

`<-` como una flechita

Este en **R** es el comando de la asignación.

Las asignaciones pueden realizarse también con una flecha apuntando a la derecha, realizando el cambio obvio en la asignación. tener en cuenta a donde apunta

Ejemplo

`x <- 3`

es equivalente a

`3 -> x`

ASIGNACIÓN

La asignación puede realizarse también mediante la función

assign()

Ejemplo

assign("x", 3)

es equivalente a

x<-3

Notar que el nombre de la variable esta escrito entre “”

Tambien se puede utilizar el simbolo

=

Para asignar un valor a una variable

x=3

ASIGNACIÓN

Resumiendo, podemos asignar valores a un objeto o variable mediante las siguientes funciones

- ✓ **assign()**
- ✓ **<-**
- ✓ **->**
- ✓ **=**

CLASES DE OBJETOS

VECTORES: son el tipo básico de objeto en **R**

MATRICES (en general, *variables indexadas*): son generalizaciones multidimensionales de los vectores. De hecho, son vectores indexados por dos o más índices.

LIST (listas): son una forma generalizada de *vector* en las cuales los elementos no tienen por qué ser del mismo tipo y a menudo son a su vez vectores o listas. Las listas permiten devolver los resultados de los cálculos estadísticos de un modo conveniente.

DATA FRAMES (hojas de datos): son estructuras similares a una matriz, en que cada columna puede ser de un objeto distinto a las otras.

FUNCIONES : son también objetos de **R** que pueden almacenarse en el espacio de trabajo.

CLASS

Todos los objetos en R tienen asociada una *clase*, que puede conocerse a través de la función *class*. Para vectores simples la clase es equivalente al modo *mode*

Ejemplo

"numeric", "logical", "character" or "list"

O también

"matrix", "array", "factor", "data.frame"

Para eliminar la clase de un objeto se utiliza la función *unclass()*

EJEMPLO

Habíamos definido la variable *x*, preguntemos que clase es

```
class(x)
```

```
[1] "numeric"
```

Veamos que ocurre con el modo

```
mode(x)
```

```
[1] "numeric" #en los objetos mas simples coinciden modo y clase
```

NOMBRES DE OBJETOS

R distingue entre mayúsculas y minúsculas:

$$A \neq a$$

se referirán, por tanto, a objetos distintos.

Los nombres de los objetos pueden contener sólo letras mayúsculas o minúsculas, junto con números y puntos

porque es el símbolo para resta

NO utilizar: espacios en blanco,  , %, \$,), []

Durante una sesión de trabajo con R los objetos que se crean se van almacenando por su nombre.

La función *objects()* se puede utilizar para obtener los nombres de los objetos almacenados en R. Es equivalente a la función *ls()*.

Es posible eliminar objetos con el comando *rm (nombre del objeto a eliminar)*.

NOMBRES DE VARIABLES

❑ NOMBRES PERMITIDOS

Combinación de letras y números, comenzando por una letra
Distingue mayúsculas de minúsculas

Ejemplo: Vort_300, geop2press, A2, a2

A2 y a2 están bien escritas pero no me significan nada

❑ NOMBRES NO PERMITIDOS

Ejemplo: 2a, y%, @ta, vort-300

❑ CONSTANTES CONSTRUIDAS EN R

LETTERS letras en mayúscula

letters letras en minúscula

month.abb mes abreviado

month.name mes entero

pi

BUENAS COSTUMBRES

✓ Poner nombres que me ayuden a identificar las variables. Ej: mes

✓ Hacer un diccionario de los nombres de las variables.

Ej: z500 geopotencial en 500 Hpa
vorti_300 vorticidad relativa en 300 HPa

COERCIÓN DE TIPOS

La mayoría de las funciones producen un error cuando el tipo de datos que esperan no coincide con los que ponemos en los argumentos.

Tenemos dos posibilidades:

- comprobar el tipo de datos utilizando funciones **is.algo()**, que nos responde con un valor lógico,
- o forzar al tipo de datos deseados “coercionando”, para lo cual podemos utilizar funciones del tipo **as.algo()**, que fuerzan el tipo de datos.

LISTA DE LOS TIPOS MÁS IMPORTANTES QUE SE PUEDEN COMPROBAR O FORZAR

character	is.character()	as.character()
complex	is.complex()	as.complex()
double	is.double()	as.double()
integer	is.integer()	as.integer()
list	is.list()	as.list()
logical	is.logical()	as.logical()
matrix	is.matrix()	as.matrix()
NA	is.NA()	-
NaN	is.NaN()	-
Null	is.Null()	as.Null()
numeric	is.numeric()	as.numeric()
vector	is.vector()	as.vector()

ATRIBUTOS DE OBJETOS

Los atributos de un objeto suministran información específica sobre el propio objeto.

Todos los objetos tienen dos atributos intrínsecos: el *modo* y su *longitud*.

Las funciones

mode(objeto)

y

length(objeto)

se pueden utilizar para obtener el modo y longitud de cualquier estructura.

ELEMENTOS BASICOS

.Last.value #contiene el resultado del ultimo comando, ya sea que
#fueras asignado a una variable o no.

#Devuelve NULL si no tenemos un valor en la última
#sentencia

object.size() # Devuelve cuantos bytes de la memoria son usados para
almacenar el objeto

Algunas funciones que nos dan información sobre el workspace

ls.str() #Lista larga de las variables definidas

ls() #Lista corta de las variables definidas

str(x) # Devuelve información de la estructura de un objeto

rm() #Elimina la variable que esta entre paréntesis de la memoria de R

rm(list=ls()) #Elimina todas las variables definidas

EJEMPLO

```
y=5  
x=3  
x+y  
[1] 8
```

.Last.value contiene el resultado del ultimo comando, ya sea que fuera asignado a una variable o no

```
[1] 8
```

object.size(x) Devuelve cuantos bytes de la memoria son usados para almacenar el objeto x

```
48 bytes
```

x del ejemplo de arriba

ls.str() Lista *larga* de las variables definidas

```
x : num 3
```

```
y : num 5
```

ls() Lista corta de las variables definidas

```
[1] "x" "y"
```

str(x) Ver información detallada de la estructura de la variable x

```
num 3
```

OTROS ELEMENTOS BASICOS

- Comentarios

Van precedidos por el signo porcentual (#)

- Continuación de línea de comandos

En R, se puede escribir comandos usando multiples lineas siempre que la sintaxis deje claro que la linea no ha sido completada. Ejemplo:

```
x = 3 +
```

```
4
```

funciona, pero

```
x = 3
```

```
+ 4
```

No funciona porque no hay indicio de que la segunda linea sea continuación de la primera.

PRINT

print es una función genérica de R que imprime el argumento de una variable

EJEMPLO

Definimos una variable asignándole un valor

M=25

Quiero ver cual es el valor que quedo guardado en M

print(M)

[1] 25

Si escribo en la consola el nombre de la variable que obtengo?

M

[1] 25 #son dos formas de obtener el valor de la variable

VOLVIENDO A LOS OBJETOS ATOMICOS....

OBJETOS

o R reconoce distinto tipo de números

Enteros	→	usa enteros de 32 bits
Reales	→	doble precisión
Complejos	→	$0 + 1.4142i$

NOTA sobre complejos: si quiero calcular `sqrt(-2)` debo escribirlo como número complejo $-2+0i$ de lo contrario obtengo error

```
sqrt(-2)
[1] NaN
Warning message:
In sqrt(-2) : NaNs produced
sqrt(-2+0i)
[1] 0+1.414214i
```

o Puede usar notación exponencial para números muy grandes/pequeños

a = 187635292012	→	a = 1.8764e+011
b = 0.0000000000123	→	b = 1.2300e-012

o Por defecto, todas las operaciones son hechas en doble precisión reales o complejas

REALES

Double crea objetos de doble precisión

as.double es una función genérica idéntica a *as.numeric*.

is.double test para ver si es del tipo double.

```
is.double(x)  
[1] TRUE
```

R no tiene un tipo de dato de precisión simple. Todos los números reales se almacenan en formato de doble precisión. Las funciones *as.single* y *single* son idénticas a *as.double* y *double* excepto que adjudican el atributo *Csingle* que se usa en la interfaz .C y .Fortran, y la intención es que se usen solo en ese contexto.

EJEMPLO

```
x <- 2          # Se asigna el valor 2 a x  
print(x)       # Se imprime el valor de x  
[1] 2  
class(x)       # Muestra cuál es la clase de x  
[1] "numeric"  
  
x <- 6/2        # Se asigna el valor de la operación dividir 6/2 a x  
print(x)  
[1] 3  
class(x)  
[1] "numeric"
```

En ambos casos parece que asignamos a X números enteros pero vemos que R los toma como reales

Para asignar explícitamente un entero, *integer*, a una variable, se agrega la letra *L* al final del número

```
x <- 23L; print(x)      # Dos expresiones de R en un mismo renglón separadas por ;  
[1] 23  
class(x)  
[1] "integer"
```

REALES Y ENTEROS

Vimos recien:

```
x <- 6/2  
print(x)  
[1] 3  
class(x)  
[1] "numeric" # o double  
is.double(x)  
[1] TRUE  
is.numeric(x)  
[1] TRUE  
is.integer(x)  
[1] FALSE
```

Para que la operación de división $6/2$, arroje como resultado un entero, se hace una conversión usando la función `as.integer`:

```
x <- as.integer(6/2)  
print(x)  
[1] 3  
class(x)  
[1] "integer"  
is.double(x)  
[1] FALSE  
is.numeric(x)  
[1] TRUE  
is.integer(x)  
[1] TRUE
```

Verifica la Clase:
Entero o Real o Doble

Verifica el MODO:
En ambos casos son números

OBJETOS

- Operaciones entre enteros y reales son de clase *numeric*, es decir *reales*

```
x<-23L
```

```
y<-3
```

```
z<- x+y
```

```
[1] 26
```

```
class(z)
```

```
[1] "numeric"
```

```
z<-x*y
```

```
class(z)
```

```
[1] "numeric"
```

- R reconoce otros tipo de valores especiales

Inf

NaN

NA



división por cero

infinito

0/0

no es un número. No es posible este suceso

Dato faltante

me falta un dato. Re inserte

FORMATO

```
format(13.7)
```

```
[1] "13.7"
```

```
format(13.7, nsmall = 3)
```

```
[1] "13.700"
```

```
format(c(6.0, 13.1), digits = 2)
```

```
[1] " 6" "13"
```

```
format(c(6.0, 13.1), digits = 2, nsmall = 1)
```

```
[1] " 6.0" "13.1"
```

«**format**» afecta solo el modo de visualización por pantalla, pero no afecta el modo en el cual R opera y graba

options(digits=6) le dice a R que queremos usar 6 dígitos de precisión en los valores que muestra por pantalla (es sólo una sugerencia y no siempre R lo toma en cuenta) basicamente R hace lo que quiere

«**sprintf** » es la función que podemos utilizar para que escriba respetando el formato

COMPLEJOS

Devuelven:

$\text{Re}(z)$	parte real
$\text{Im}(z)$	parte imaginaria
$\text{Mod}(z)$	el módulo
$\text{Arg}(z)$	argumento
$\text{Conj}(z)$	parte conjugada

VARIABLES TIPO CARACTER

SUBSTR

```
prueba='esto es una cadena de caracteres'      #lo escribí con comas simples  
prueba
```

```
[1] "esto es una cadena de caracteres" #me lo devuelve con comas dobles
```

```
class(prueba)  
[1] "character"
```

me da las posiciones de los caracteres que voy a sustraer

```
prueba2=substr(prueba,6,11)      #Para obtener una sección de la cadena de  
                                caracteres porque me quiero quedar solo con una parte
```

```
prueba2  
[1] "es una"
```

CARACTER - CONT

PASTE

```
nombre='Abril'  
apellido="Acevedo"  
nombre  
[1] "Abril"  
apellido  
[1] "Acevedo"
```

```
completo<-paste(nombre, apellido,sep=" ") # Función interna que concatena caracteres  
completo  
[1] "Abril Acevedo"  
completo<-paste(nombre, apellido,sep="")  
completo  
[1] "AbrilAcevedo"
```

depende si los quiero poner juntos o separados

CARACTER - CONT

PASTE - AS.CHARACTER

Definimos dos variables, una carácter y otra numérica...

```
nombre='Abril'
```

```
nombre =
```

```
Abril
```

```
edad=2
```

```
edad
```

```
[1] 2
```

```
ls.str()
```

```
edad : num 2
```

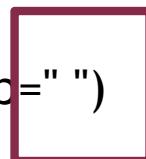
```
nombre : chr "Abril"
```

hay veces q no es necesario poner el as.character

```
completo=paste(nombre, as.character(edad),sep=" ")
```

```
completo
```

```
[1] "Abril 2"
```



deje un espacio poniendo un espacio
entre las comillas en *sep*

CARACTER - CONT AS.NUMERIC

```
rm(list=ls())
```

Definimos variables carácter y necesitamos operar...

```
nombre="Abril"
```

```
edad="02"
```

```
nombre
```

```
[1] "Abril"
```

```
edad
```

```
[1] "02"
```

```
nombre2="Ayelen"
```

```
edad2=1
```

```
nombre2
```

```
[1] "Ayelen"
```

```
edad2
```

```
[1] 1
```

convierto porque como esta entre comillas es un carácter y yo necesito una variable numérica

```
edadmedia=(as.numeric(edad)+edad2)/2   as.numeric: convierte un carácter en un número.
```

```
edadmedia
```

```
[1] 1.5
```

CARACTER - CONT

```
rm(list=ls())
```

Definimos dos variables
y escribimos una cadena de caracteres que incluye variables y tipos ...

```
mes=07
exp="Test01"
mes
[1] 7
exp
[1] "Test01"
```

```
titulo=paste0(exp,", Mes", as.character(mes),", Temp")
titulo
[1] "Test01, Mes7, Temp"
```

Esta estructura puede resultar útil al momento de escribir TITULOS

.MACHINE

precisión, doble precisión. Cosas de la máquina

Información sobre la precisión en la que trabaja R dependiendo la máquina donde este trabajando

el número real positivo más pequeño x tq $(1 + x) \neq 1$ Machine epsilon

.Machine\$double.eps

[1] 2.220446e-16

un número real positivo pequeño x tq $(1 - x) \neq 1$.

.Machine\$double.neg.eps

[1] 1.110223e-16

#el menor número real normalizado distinto de cero

.Machine\$double.xmin

[1] 2.225074e-308

el mayor número real normalizado

.Machine\$double.xmax

[1] 1.797693e+308

#la base para la representación del punto flotante

.Machine\$double.base

[1] 2

#el numero base de digitos en el punto flotante

.Machine\$double.digits

[1] 53

#el mayor entero que puede ser representado

.Machine\$integer.max

[1] 2147483647