BIBLIOGRAFÍA:

UNDERSTANDING ROBUST AND EXPLORATORY DATA ANALYSIS. Hoaglin, Mosteller, Tukey. Wiley.

MODERN APPLIED STATISTICS WITH S-PLUS. Venables, Ripley.

SOFTWARE: R
Página:

http://www.dm.uba.ar/materias/analisis_de_datos/2011/2/

EVALUACIONES

Tres evaluaciones. Dos son parciales y una es global.

Las evaluaciones constan de dos partes:
i) entrega de un trabajo,
ii) preguntas teórico-prácticas.

El trabajo y las preguntas teórico prácticas son calificados en una escala de 1 a 10. Para aprobar la evaluación es necesario que estén aprobadas las dos partes (puntaje mayor a 6).

La nota de una evaluación aprobada resultará del promedio pesado de cada parte (0.6 para el trabajo y 0.4 para las preguntas teórico-prácticas).

Criterio de aprobación: Aprobar **una** evaluación parcial y la evaluación global.

Las preguntas teórico-prácticas parciales no se recuperan.
Las preguntas teórico-prácticas correspondientes a la evaluación global pueden recuperarse una vez.
Si un trabajo no está aprobado, es devuelto para su corrección una vez.

¿POR QUÉ ANÁLISIS DE DATOS?

técnicas estadísticas clásicas

- **óptimas**- condiciones restrictivas
- **inadecuadas**- situación real alejamiento de los supuestos

técnicas robustas y exploratorias más recientes han ampliado la efectividad de los análisis estadísticos.

técnicas del análisis exploratorio de datos

*permiten dar un tratamiento informal a un conjunto de datos

*dan énfasis al estudio flexible de los datos antes de compararlos con cualquier modelo probabilístico.

¿Por qué R?

Es un lenguaje de distribución libre que tiene

- un entorno flexible para el análisis de datos.
- una colección extensa y coherente de herramientas estadísticas para análisis de datos,
- un lenguaje para expresar modelos estadísticos y herramientas para utilizar modelos estadísticos lineales y no lineales.
- facilidades para el análisis de datos y su presentación tanto en la computadora como en papel,

- un lenguaje de programación orientado a objetos que puede ser fácilmente extendido.

En ***R*** el programa pregunta si se quiere guardar el espacio de trabajo - workspace - cada vez que se cierra la sesión.

Al **guardar** el espacio de trabajo, los objetos creados durante la sesión, quedan en forma **permanente** hasta que se los borre.

Un poco de historia

R es una implementación libre, independiente, "open-source" del lenguaje de programación S que actualmente es un producto comercial llamado S-PLUS y es distribuido por Insightful Corporation.

El lenguaje S, que fue escrito a mediados de los años 70 en Bell Labs (de AT&T y actualmente Lucent Technologies).

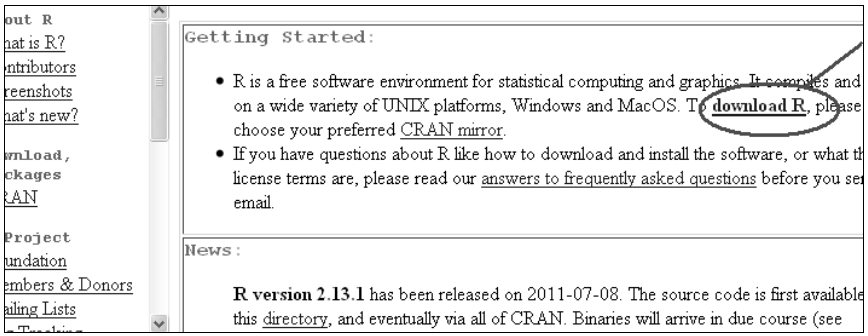
Originalmente un programa para el sistema operativo Unix, **R** ahora puede obtenerse también en versiones para Windows y Macintosh y Linux.

A pesar de que hay diferencias menores entre **R** y S-PLUS (la mayoría en la interfase gráfica), son esencialmente idénticos.

El proyecto R fue iniciado por Robert Gentleman y Ross Ihaka (de donde se deriva "R") del Statistics Department in the University of Auckland en 1995.

Actualmente R es mantenido por un grupo internacional de desarrolladores *voluntarios*: Core development team.

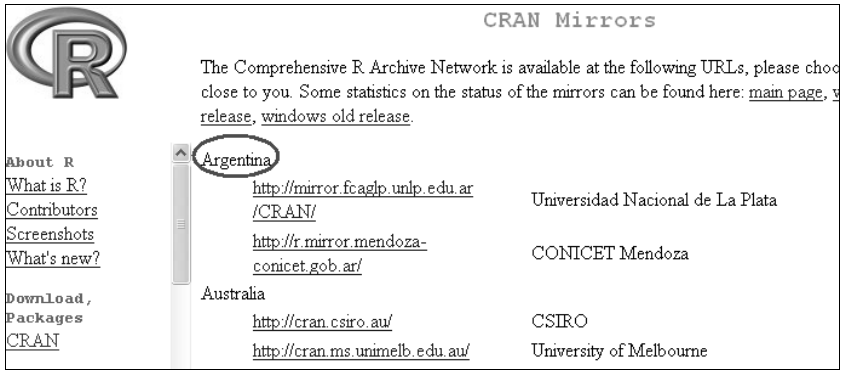
La página web del proyecto R es
`http://www.r-project.org/.`

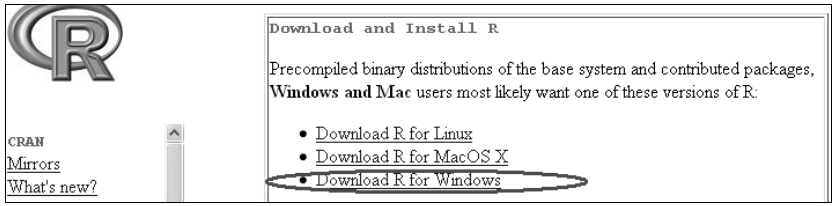


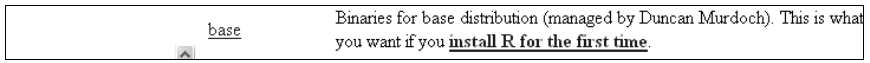
Este es el sitio principal sobre in formación de R

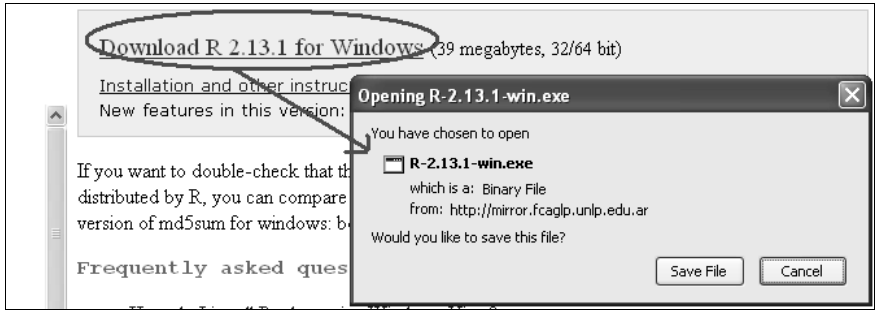
Para bajar el software directamente se sugiere utilizar una página "mirror" (espejo) en argentina por ejemplo

`http://mirror.fcaglp.unlp.edu.ar/CRAN/`



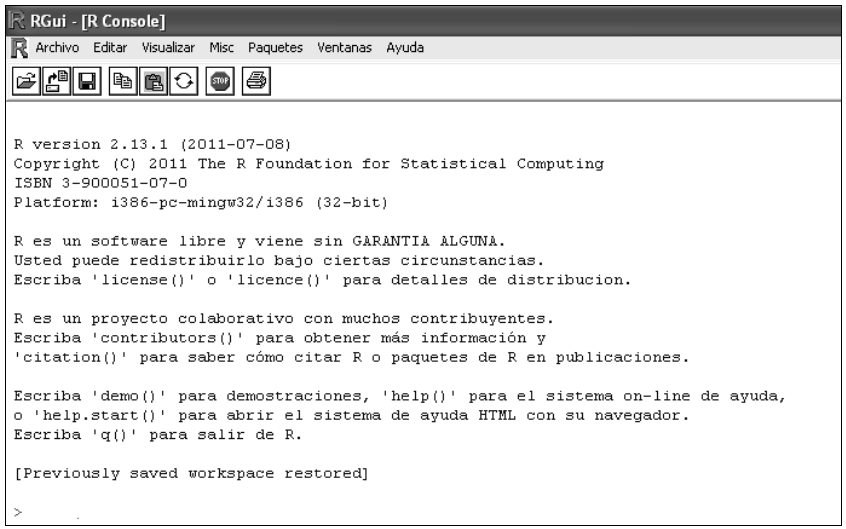






Iniciando R

La forma más fácil de usar R es en forma interactiva mediante la línea de comandos. Una vez instalado hay que hacer un doble click en el ícono de R (en Unix/Linux, se escribe R desde el símbolo de comandos (command prompt)). Cuando R se inicia, aparece la ventana del programa "*Gui*" (graphical user interface) con un mensaje de apertura



Debajo del mensaje de apertura en la Consola de R se encuentra el "prompt" que es el símbolo **>** ("mayor").

>

La mayoría de las expresiones en R se escriben directamente a continuación del "prompt" en la Consola de R.

También es posible generar un **archivo "script"** con los comandos que se quieren ejecutar. Veremos esto luego.

Conceptos básicos del lenguaje de R

expresiones, asignaciones, *funciones, tipos de datos*

expresión simple

```
> 2+3   ↵  (enter) #expresión
[1] 5       #evaluación
```

expresión un poco más compleja

```
> sqrt(3/4)/(1/3 - 2/pi^2)
[1] 6.626513
```

El símbolo **>** (prompt) indica la línea de comandos y el **[1]** que la respuesta comienza en el primer elemento de un vector.

Si se escribe una expresión incompleta > 2* ↵
R responde con un + que indica continuar

```
> 2* ↵
+ 5
[1] 10

> sqrt(3/4  ↵
+ )
[1] 0.8660254
```


La expresión más común es el *llamado a una función*, se escribe el nombre de la función seguida de sus argumentos entre paréntesis.

```
> sqrt(3/4)
[1] 0.8660254

> Sqrt(3/4)
Error: no se pudo encontrar la función "Sqrt"
```

```
> sqrt(3/4)
[1] 0.8660254
```

Diferencia mayúsculas de minúsculas.

El software "reconoce" al número pi

```
> pi
[1] 3.141593
```

Si se escribe una cadena de caracteres seguidos por un par de paréntesis R lo interpreta como el **nombre** de una **función**.

```
> pi()

Error: no se pudo encontrar la función "pi"
```

Si la función existe y no requiere argumentos se ejecutará la función, como

```
q()
```

para irse de la sesión**.**

Si la función existe y requiere argumentos dará un mensaje de error

> sqrt()
Error en sqrt() : 0 arguments passed to 'sqrt' which requires 1

Algunos mensajes de error aparecen en Castellano y otros en Inglés!

Asignaciones

Hay varios operadores con los que es posible realizar asignaciones

"<-" signo menor seguido del signo menos, sin espacios

"=",

Enteros consecutivos

```
> a <- 2:6        # crea el vector (2,3,4,5,6)

> a
[1] 2 3 4 5 6
```

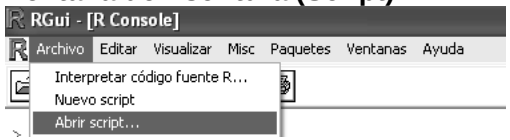
Aritmética

```
> b <- 2*a+1
```



```
> b
[1]  5  7  9 11 13

> b <- a/2          # división
> b
[1] 1.0 1.5 2.0 2.5 3.0

> b <- a^3.7    # eleva  a la potencia 3.7 cada componente de a

> b
[1]  12.99604  58.25707 168.89701
[4] 385.64616 757.11112

> b <- log(a)      #  logaritmo natural
> b
[1] 0.6931472 1.0986123 1.3862944
[4] 1.6094379 1.7917595

> b <- log10(a)   # asignación
> log10(a)        # evaluación
[1] 0.3010300 0.4771213 0.6020600
[4] 0.6989700 0.7781513
> b <- logb(a,base=2) # logaritmo base 2

> b <- logb(a,2)      # idem

> help(logb)   # se abre una ventana de
               #ayuda
```

Ventana de Escritura (Script)



Archivo -> Nuevo script

Es útil acomodar las dos ventanas, la de comandos y la del editor, para poder verlas simultáneamente

Ventanas -> Divida Verticalmente

Tipos de datos

5 Tipos de objetos datos básicos:

data frames, matrices, vectores, listas y funciones.

Data frame (marco de datos)

Ejemplo de un conjunto de datos incorporado en el R

```
> library(rpart)
> kyphosis
   Kyphosis Age Number Start
1    absent  71      3     5
2    absent 158      3    14
```



```
3   present 128      4     5
4    absent   2      5     1
5    absent   1      4    15
6    absent   1      2    16
7    absent  61      2    17
8    absent  37      3    16
9 ........................
```

en cada columna se guardan los valores de una variable.

> class(kyphosis)

[1] "data.frame"

> ?kyphosis

data frame, 81 filas (casos, niños sometidos a una cirugía espinal) Variables: Kyphosis - factor - con 2 niveles presencia o ausencia de una deformidad post operatoria, Age, Number, Start son vectores numéricos.

Todas las columnas de un data frame deben tener la misma longitud: intentamos generar un data frame con columnas de diferente longitud

```
> data.frame( x=1:4,y=1:3)
Error en data.frame(x = 1:4, y = 1:3) :
  arguments imply differing number of rows: 4, 3
```

Corregimos el error

```
 > data.frame( x=1:3,y=1:3)
  x y
1 1 1
2 2 2
3 3 3
```

¿Pero si necesitamos variables con diferente longitud? Corregimos el error de otra forma:

```
> data.frame( x=1:4,y=c(1:3,NA))
  x  y
1 1  1
2 2  2
3 3  3
4 4 NA
```

¿Cual es el resultado de aplicar la función c?

¿Que significa “NA”?

Para ver todos los datos incorporados en R:

```
> data(package = .packages(all.available = TRUE))
```

Aparecen unos mensajes de advertencias (warnings) que pasamos por alto y se abre una ventana con el listado de todos los conjuntos de datos incorporados al R

Veremos también:

Matrices: son similares a los data frames, salvo que sus elementos deben tener datos con el mismo modo (carácter, numérico, lógico). Las filas y las columnas pueden tener nombres.

Vectores: es un conjunto ordenado de elementos que tienen el mismo modo. Los elementos de un vector pueden tener nombres.

Listas: son colecciones de otros objetos. Sus componentes pueden ser data frames, matrices, vectores, otras listas, cualquier objeto.

Funciones: existen gran cantidad de funciones incorporadas También es posible agregar funciones definidas por el usuario.