

MANUAL DE USUARIO:

Análisis estadístico de datos espaciales con R y QGis

Unidad I

Introducción a R y QGis e Instalación, configuración y uso básico de R y sus interfaces



Autor: Ligdamis A. Gutiérrez E. PhD.

Versión 1.0

ÍNDICE

Unidad I: Introducción a R y QGis e Instalación, configuración y uso básico de R y sus interfaces

1)	Introducción a R	3
•	Breve definición e historia de R	4
○	Historia	4
○	Conceptos básicos	4
○	Paquetes y tipos de paquetes de R	6
○	Documentación de R	7
2)	Introducción a QGis	7
•	Breve definición e historia de QGis	7
○	Historia	7
○	Sistemas de Información Geográfica SIG/GIS	8
○	Datos SIG (Vectorial y Raster)	9
○	Documentación de QGis	11
3)	Instalación de R	12
•	Breve descripción de los pasos de la instalación en sistemas 32 y 64 bits	12
○	Instalación en Windows de R	12
○	La interfaz gráfica de R, comandos y operaciones básicas	20
a)	Comandos generales de R	20
b)	Barra de Herramientas de menú y sus secciones	21
c)	Operaciones y Funciones básicas con R.....	26
4)	Instalación de RStudio	30
○	Introducción e Instalación básica de RStudio	30
○	Interfaz gráfica de RStudio	34
○	Partes de las ventanas y menús de RStudio	36
5)	Instalación de RCommander (RCmdr)	44
○	Introducción e Instalación básica de RCmdr	44
○	Interfaz gráfica de RCmdr	46
○	Partes de la ventana y menús de RCmdr	46
6)	Primeros pasos con datos en R, RStudio y RCmdr	47
•	Conjuntos de datos	47
○	Creación y manejo de variables	47
○	Vectores	48
○	Factores	56
○	Matrices	58
○	Estructuras de datos (Data frames)	62
○	Listas	69
7)	Gráficos con los datos	75
○	Comandos y tipos de gráficos	75
8)	Uso de Script en R, RStudio y RCmdr	94
9)	Guardar y visualizar datos	95
10)	Bibliografía y Enlaces de Interés	97

1) Introducción a

Comenzaremos indicando que R es un lenguaje de programación orientado al cálculo estadístico y generación de gráficos. A la vez, se puede expresar que R es una herramienta de trabajo que permite de forma sencilla el tratamiento y análisis de datos, de ahí que R pueda ser adaptable a una gran variedad de tareas.

Aunque R no es precisamente un lenguaje estadístico, si es un entorno en el que se han implementado una gran variedad de técnicas estadísticas, clásicas y modernas, abarcando desde modelos lineales a las más avanzadas técnicas de clasificación, pasando por test clásicos y análisis de series temporales, convirtiéndolo así para muchos en un sistema estadístico. Algunas de dichas técnicas están incorporadas en su entorno base y otras muchas acompañan al lenguaje, son creadas y perfeccionadas por una gran comunidad de desarrolladores y que conforman el entorno de programación, son las denominadas librerías, también llamadas en R paquetes (packages)¹, que al igual que R, pueden ser descargadas de forma gratuita, lo que le concede al lenguaje potencia y versatilidad convirtiéndolo en uno de los mayores entornos utilizado por la comunidad científica. El principal repositorio de paquetes estables en R se denomina “CRAN (*Comprehensive R Archive Network*)”, y actualmente existen más de 10,000 paquetes solo en el CRAN, aparte de los miles que se encuentran disponibles por parte de los colaboradores y la comunidad en general, incluso existe documentación para crear² y distribuir paquetes si se desea contribuir.

En la actualidad, existen distintos paquetes de software estadístico propietario (que se tienen que registrar o poseer una licencia y son de pago), como: SPSS, Stata, Minitab, Statistica, SAS, SPlus, etc. Que perfectamente pueden cubrir las necesidades de cualquier usuario de técnicas estadísticas ya sea básicas o avanzadas. Sin embargo, lo que diferencia a R de estos paquetes es que R es una alternativa de software libre, es decir gratuito apoyado como se ha indicado, en una gran comunidad de desarrolladores de software, que continuamente están aumentando su capacidad y riqueza y que proporcionan una herramienta sencilla y potente para diversos ambientes de docencia, desarrollo e investigación.

Entre sus principales características se encuentran:

- R es Open Source (libre, abierto). Además de gratuito, indica que se puede modificar.
- R es Multiplataforma, puede ser implementado en diversos sistemas operativos sin ningún problema (Windows, Unix, MacOs y GNU/Linux entre otros).
- Dispone de un elevado número de paquetes.
- Posee un ciclo completo de trabajo: Implementación de algoritmos, almacenamiento, preparación y manipulación de datos, análisis de resultados y la generación de documentos.
- Es un lenguaje orientado a objetos.
- Operadores para cálculo sobre variables indexadas que son los arreglos (Arrays), en particular matrices.
- Posee muy buena documentación.
- Facilidad de uso e instrucciones mínimas para presentar resultados a comparación de otros lenguajes.
- Crea gráficos de gran calidad de manera muy sencilla.

Básicamente, la interacción con R, se produce a través de una interfaz de línea o ventana de comandos denominada “Consola”, aunque R posee interfaces gráficas de usuario (GUI), que permiten un uso amigable y sencillo al usuario.

¹ Como referencia, habría que indicar que, a finales de marzo del 2013, existían más de 4,400 paquetes disponibles para solucionar diversos problemas estadísticos.

² <https://oscarperpinan.github.io/R/Paquetes.html> (crear paquetes en R)

- Breve definición e historia de R

- Historia

R es conocido también como “GNU S”, debido a que es un proyecto GNU (conjunto de programas auspiciados por la Free Software Foundation de [Richard Stallman](#)), es decir, de uso libre y similar al lenguaje S, desarrollado por los laboratorios Bell de AT&T inicialmente diseñado para uso interno, por esta razón la mayoría de los códigos de S también corren bajo R. El término GNU corresponde al acrónimo recursivo *“GNU's Not Unix”* (*en español*: GNU no es Unix).

El nombre de R se debe a sus creadores en 1993, “Ross Ihaka y Robert Gentleman”, cuyos nombres comienzan con la letra “R”, que trabajaban en el departamento de estadística de la universidad de Auckland en Nueva Zelanda, inicialmente lo crearon para tareas docentes. Fue en 1995, cuando Martin Maecheler les convenció para que el software tuviera su distribución gratuita. De esta forma, las primeras versiones pilotos, con un cero en su denominación en el primer dígito estuvieron disponibles para 1990. Actualmente con la gran cantidad de aportaciones que recibe cada año R las versiones han crecido de forma exponencial, todas de libre distribución y que pueden ser descargadas en la página Web de R. La versión que utilizamos en el presente módulo es la 3.6.2 del 2019-12-12 (“Dark and Stormy Night”), al inicial R, también se indica la plataforma en la que se encuentra trabajado, por ejemplo “Platform: i386-w64-mingw32/i386 (32-bit)”.

- Conceptos básicos

Cuando se completa la instalación de R, se encuentran varios archivos ejecutables en la ruta por defecto, que en el caso de Windows es:

C:\Program Files\R\R-3.6.2\bin\i386

Al ejecutar R, se presenta la pantalla inicial del software que normalmente es la interfaz gráfica de usuario GUI, de tipo MDI (Multiple Document Interface) a través de RGui. Dicha interfaz, visualiza la consola de comando de R, a través de la cual se indicarán la mayoría de las instrucciones. En R además se pueden encontrar otras GUI para mayor facilidad de usuario, como RStudio y RCommander (RCdr), pero para poder utilizarlas, previamente ha de ser instalado R, pues dichas interfaces trabajan bajo R como base. La imagen es similar a la figura siguiente.

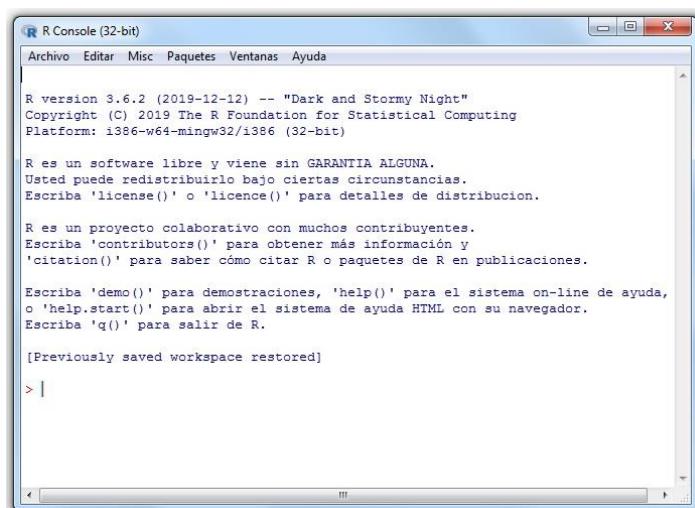


Fig. 1. Pantalla Inicial de RGui

La ventana que se observa en la figura es la consola o ventana de comandos (R Console), en la parte inferior se observa, la línea de comandos, que comienza con el símbolo (prompt); “>”, aquí será desde donde se comenzarán a teclear las instrucciones o comandos que serán ejecutadas por R. R al igual que todos los software estadísticos, utiliza un lenguaje propio. Adicionalmente, si se desea entrar únicamente en la ventana clásica de la línea de comandos tipo MSDOS, en el directorio de instalación se encuentra el ejecutable “**R.exe**”, que da acceso a la siguiente ventana.



Fig. 2. Pantalla Inicial de la ventana de comandos de R

R puede utilizarse como una calculadora matemática muy potente. Por ejemplo, una instrucción básica si queremos realizar una raíz cuadrada de la multiplicación de dos números

```
> sqrt ( 8 * 2) # Aquí se multiplica 2 por 8 y se calcula la raíz cuadrada
```

```
[1] 4
```

La instrucción “sqrt” es una función que realiza el cálculo de la raíz cuadrada. El símbolo # a la derecha de la operación se utiliza para colocar comentarios. En la respuesta que se observa abajo, el número entre corchetes que aparece [1], solamente indica el lugar que ocupa el primer valor del resultado de la expresión. Hay que tener cuidado con el uso de mayúsculas o minúsculas en R.

Las funciones en R se constituyen de un nombre seguido de un paréntesis (como en el ejemplo anterior), en el que, dentro del paréntesis, se pueden incluir diversos argumentos dependiendo de la función a utilizar. R es un lenguaje funcional, en el sentido de que los programas en R se presentan como funciones escritas en su lenguaje.

Las instrucciones que se ejecutan desde la línea de comandos se denominan “expresiones”. De esta forma, las expresiones se ejecutan al pulsar la tecla “Enter”. Las expresiones pueden tener más de una línea. Cuando se pulsa “Enter” después de una expresión incompleta, R no envía o produce ningún error, sino que se encuentra esperando a que se complete la expresión en la siguiente línea, para ello, el “prompt”, cambia a “+” al inicio de la siguiente línea, indicando que se está esperando completar la expresión inconclusa.

Al ser R un lenguaje orientado a objetos, los elementos básicos de R, son precisamente los objetos. Por lo tanto, toda expresión de R se referirá a ellos, de tal forma, que los objetos son archivos capaces de ser editados y en su caso, ejecutados. Esto diferencia entre R y S con el resto de software estadístico, debido a que el objeto se maneja como una entidad básica, por eso cualquier expresión evaluada con R da como resultado un objeto. De esta forma, cada objeto pertenece a una clase, así las funciones pueden tener diferentes comportamientos, en función de a qué clase pertenezcan.

Es importante que una vez que se instala R, se debe proceder a organizar el entorno de trabajo. Esto implica seleccionar el directorio o la carpeta en donde se van a almacenar los diversos archivos con los que se estará trabajando. Para ello hay que recordar crear las carpetas en base a su uso, así hay que proceder a crear carpetas para archivos, para gráficas, etc. Al seleccionar el directorio de trabajo, este estará disponible automáticamente para cargar los archivos de datos. Si no se selecciona un directorio de trabajo, se utiliza el que se tiene por defecto que depende de la instalación.

Para seleccionar un directorio de trabajo se utiliza la función “setwd()”, indicando la ruta hacia la carpeta que se desea. Por ejemplo la instrucción:

```
setwd("C:/Users/Lig/Documents/Programas_R")
```

Seleccionará la carpeta “Programas_R”, en la ruta a mis documentos del disco “C”. Hay que hacer notar que las barras invertidas, se encuentran a diferencia de la ruta en la computadora, en sentido inverso. Porque en vez de “C:\User\|Lig”, se debe de escribir “C:/User/Lig”.

Para poder saber cuál es el directorio actual de trabajo, se utiliza la instrucción “getwd()”. Para salir de R, ejecutamos la función “q()”, cuando se abandona R, el software nos pregunta si se desean conservar los cálculos que se han realizado en la sesión, con la pregunta: “¿Guardar imagen de trabajo? (ver figura 3) Al responder “Sí”, cuando se comienza la siguiente sesión, se podrán volver a utilizar los resultados de la sesión finalizada. Lo que es útil en caso de utilizar bibliotecas o datos que ya se han utilizado sin tener que volver a cargarlos.

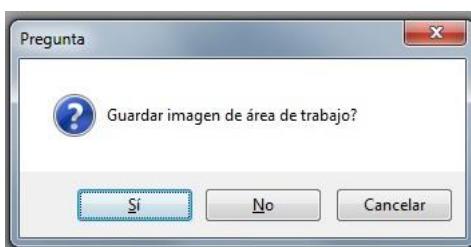


Fig. 3. Pantalla de fin de sesión en R

- Paquetes y tipos de paquetes de R.

R se compone de un sistema base nombrado “CRAN” y de paquetes adicionales que amplían su funcionalidad. Al ejecutar R, cualquier aplicación que se utilice usa lo que se denomina “paquete” que básicamente es un archivo con el código que contiene las funciones a utilizar. Los paquetes son programados por usuarios que los ponen a disposición pública en los repositorios, siendo el mayor el ya mencionado “CRAN” y los que no son parte del sistema base pero son “recommended” o sugeridos.

Para instalar un paquete en la consola de comandos se teclea la función “install.packages()”, con el nombre del paquete entre comillas.

Por ejemplo, para actualizar a la última versión de R, desde R, se teclea la siguiente instrucción desde la ventana de comandos:

```
> install.packages("paquete")           # llama a instalar el paquete llamado paquete
> install.packages("paquete", dependencies = T) # Instala el paquete con sus dependencias
> library("paquete")                  # carga la librería o el paquete para ser utilizado
> updateR()                          # Actualiza la versión de R
```

No solo basta con instalar un paquete, se debe de cargar en memoria, por ello se utiliza la función “library()”, con el nombre del paquete entre el corchete y entre comillas, para que este esté ya disponible para utilizarlo. En instrucciones más complejas de programas se utiliza la función “require()”, debido a que carga el paquete si está disponible o retorna “FALSE”, en caso contrario, evitando la interrupción del script o programa al no encontrar el paquete.

Para ver la lista de los paquetes que se tienen en memoria se utiliza la función “search()” , y para ver la lista de los paquetes o librerías disponibles solo hay que teclear la función, “library()”, si lo que se desea es saber que paquetes se encuentran instalados es: “installed.packages()”.

En Internet se puede visualizar los paquetes disponibles a través de la página Web de R.

<https://www.r-project.org/> Se da clic a “CRAN”, y se busca un servidor espejo (Mirror) o servidor de alojamiento, se puede elegir entre el designado, base o “0-Cloud” (<https://cloud.r-project.org/>), o dar clic en el que se desee de la lista disponible, tanto en español (<https://cran.rediris.es/>) como en otros lenguajes. Una vez dentro del CRAN, se da clic a “Packages”. Aquí se puede desplegar una lista de los paquetes ordenados por fecha de publicación o por nombre. Es útil teclear “update.packages()”, para revisar las versiones de los paquetes instalados en el sistema, esto compara con los disponibles en CRAN. De esta forma, se pueden actualizar los paquetes con las versiones más recientes.

- Documentación de R.

Existe una gran variedad de documentación, ayuda, ejercicios, etc. Sobre R en la red. La principal fuente de documentación es precisamente la página Web de R³. En dicha página en la parte izquierda se encuentra una sección de documentación, en la que se puede encontrar manuales en diversos formatos como PDF y HTML. Además, libros de varios contenidos y autores y las respectivas editoriales desde donde pueden ser adquiridos. De igual forma, los textos recomendados en el sílabo deben ser consultados para ampliar el conocimiento de R.

2) Introducción a QGis

QGis es un sistema de información geográfica GIS o por sus siglas en inglés GIS, anteriormente denominado “Quantum GIS”, pertenece a la categoría de software libre.

- Breve definición e historia de QGis

- Historia

QGis fue desarrollado por QGis Development Team, su lanzamiento fue en julio de 2002 y para 2008 se consolidó en el mercado como uno de los mejores sistemas de información geográfica open source. QGis permite manejar datos raster y vectoriales, así como bases de datos para múltiples plataformas como Windows, Unix, GNU/Linux, Mac OS y Android. QGis fue desarrollado en C++, Python y un framework multiplataforma orientado a objetos llamado Qt. QGis es muy versátil, debido a la facilidad que brinda al interconectar múltiples bases de datos espaciales como PostgreSQL/PostGIS, GeoPackage, Oral Database y SpatialLite entre otras. La versión actual es QGIS 3.10.1 “A Coruña”, que fue lanzada el 06 del 12 de 2019, para 64 y 32

³ <https://www.r-project.org/>

bits en Windows, así como para otras plataformas. Adicionalmente, se pueden descargar diversas versiones del software en la página Web.⁴

- Sistemas de Información Geográfica SIG/GIS

Prácticamente todo el gran volumen de información que se maneja hoy en día, como pueden ser, la información que se brinda por los medios de comunicación, a que manejan los gobiernos a través de sus entidades y ministerios como el de educación, o las bases de datos de grandes superficies o almacenes y que están interesados en determinados productos, o escuelas en una determinada zona geográfica, o el seguimiento que se puede realizar a través de Internet etc. Toda dicha información va resultar mucho más útil si está georreferenciada, es decir, si en dicha información se incluyen las coordenadas geográficas donde se producen.

Por este motivo, los sistemas de información geográfica SIG o por sus siglas en inglés GIS (*Geographic Information System*), son herramientas que se han desarrollado para poder gestionar la información que se obtiene en un territorio determinado y que debido a su gran potencia de cálculo y análisis se pueda trabajar con un elevado volumen de datos como hemos indicado anteriormente.

La definición clásica de un SIG, proviene de Burrough y McDonell⁵ en 1998 que indica que un SIG es “*un potente conjunto de herramientas para recopilar, almacenar, recuperar a voluntad, trasformar y mostrar datos espaciales del mundo real para un conjunto particular de propósitos*”.

Dicho de otro modo, se podría indicar que un SIG es un conjunto de herramientas específicas que integra y relaciona diversos componentes, que permiten a los usuarios finales organizar, almacenar, manipular, modelizar, crear consultas, integrar, analizar y representar de una forma eficiente grandes cantidades de datos de cualquier tipo de información geográfica referenciada asociada a un territorio, facilitando la toma de decisiones de una forma mucho más eficaz.

En el mercado existen una gran variedad de software SIG, con licencia comercial como ArcGis, IDRISI, ESRI, MapInfo, Autodesk, ABACO DbMap o con licencia libre GNU como gvSIG, Capawer o QGis. Entre todos, QGis aparte de ser gratuito, brinda la potencia y capacidad para convertirse en uno de los mejores softwares de información geográfica desarrollados y en constante expansión y crecimiento, además QGis de que tienen una mayor interconexión con el paquete estadístico R (también gratuito). De hecho, sigue progresando y actualmente existen dos nuevos productos denominados: QGIS Browser y QGIS Server con diferentes interfaces de usuario.

Los datos utilizados en los SIG poseen dos propiedades básicas que son:

- ✓ Geométricas: Los datos se localizan en un lugar determinado, o lo que es lo mismo, se encuentran georreferenciados con una determinada coordenada que permiten localizar puntos, líneas y polígonos.
- ✓ Información estática (o descriptiva): Aquí cada dato o “geodato” tiene asociada una matriz de datos con información de variables, que se recoge en tablas de atributos, como, por ejemplo, el número de escuelas, número de docentes, productividad, población de alumnos por municipio, uso de recursos, etc. Asociados a unas determinadas coordenadas.

En la mayoría de los sistemas GIS como por ejemplo el ArcGis, la arquitectura básica con la que están desarrollados y funcionan es mediante “toolbox” o sea, una caja de herramientas

⁴ <https://www.qgis.org/es/site/forusers/download.html>

⁵ Burrough, P.A y McDonnell R.A. (1998) Principles of Geographical Information System. Oxford University Press.

informáticas que componen un entorno de desarrollo integrado con toda la información, por ejemplo, los datos, contenida en dichos programas y dependiendo de cada computadora.

En la actualidad, con la enorme cantidad de datos disponible en medios móviles y digitales como Google Earth o Google Maps, la tendencia es a utilizar GIS con arquitectura de servicios, en donde la información utilizada (los datos), puede provenir también de Internet, del espacio, de imágenes de satélites a través de meta-datos, o imágenes del terreno que forman los mapas y que se obtienen con facilidad mediante los GPS (Global Positioning System). Son estos GIS como el QGis que poseen ventaja, ya que, aunque muchos de los datos que se obtienen no son gratuitos, sus visualizaciones si, además que no hay que estar pendiente de las actualizaciones, debido a que constantemente se encuentra realizando por parte de la comunidad de desarrolladores.

Algo importante a mencionar de los Gis es que se almacenan en “capas”, las cuales pueden combinarse, o sea, sobreponer una capa sobre otra, para de esta forma, crear mapas muy diversos, a través de los cuales se puedan realizar consultas diferentes, tanto por su geometría como por su información descriptiva. Por ejemplo, analizar una determinada área, mapa o zona geográfica de acuerdo a su relieve, hidrología, núcleos urbanos, población, centros educativos, recursos, comunicaciones, etc. Con un Gis se puede individualizar cada capa, o solo seleccionar aquellas que se deseen trabajar para resaltar un determinado aspecto, incorporar nueva información y crear así nuevos mapas.

Cada una de las capas, con las que se trabaja poseen información geográfica (mapa digital) sobre el que se añaden datos alfanuméricos generando dos tipos de archivos: cartográficos (mapas digitales) y datos (tablas de atributos). Con los GIS se pueden localizar objetos en el espacio o sea, georreferenciar lo que permite calcular áreas, distancias, etc. Y relacionarlos con áreas topográficas, lo que permite conocer zonas conectadas.

Un aspecto útil de los GIS, aparte de capturar y almacenar información, es realizar consultas de forma muy rápida teniendo en cuenta de que se está manipulando una gran cantidad o miles de datos. Así se pueden, por ejemplo, localizar zonas con necesidades de infraestructuras como escuelas, hospitalares, caminos, etc. Hacer un seguimiento y monitorización de una determinada área o territorio que permita ver su evolución, cambio a través del tiempo, población, etc.

- Datos SIG (Vectorial y Raster)

El trabajo con un GIS se realiza de acuerdo con una parte de la realidad elegida de acuerdo a una necesidad específica. Esto lleva a modelizar dicha realidad de dos formas principalmente: Vectorial y Raster. De esta forma, se nombra un GIS Vectorial o un GIS Raster cuando predomina una u otra forma.

✓ **GIS Vectorial:** Está formado por datos geográficos, es decir, datos espaciales, representados a través de coordenadas. Los datos pueden ser de tres tipos:

- Puntos: Representan objetos espaciales que solo están localizados, no tienen dimensiones, es decir, ni largo (longitud), ni ancho (anchura). La posición de cada objeto queda fijada a través de las coordenadas de los sistemas de referencia, o sea, georreferenciados, que es lo mismo, que dotados de coordenadas (x,y). De esta forma, pueden representar escuelas, hospitalares, iglesias, recursos, etc.
- Líneas (segmentos que une dos puntos): Son una sucesión de puntos y representan objetos espaciales con una dimensión, longitud. Su posición se fija con dos pares de coordenadas. Mediante este objeto se representan, caminos, carreteras, ríos, líneas de ferrocarril, etc.
- Polígonos (unión de varias líneas): Son una sucesión de líneas cerradas y representan objetos espaciales con dos dimensiones, longitud y anchura. La

posición de cada objeto se fija con dos o más líneas cuyas coordenadas inicial y final coinciden. A través de estos objetos se representan, lagos, pueblos, etc.

Dependiendo de la escala que se utilice, una misma entidad puede ser representada por diversos objetos. De este modo, una ciudad, por ejemplo, puede aparecer como un punto, pero también como un polígono. El Gis vectorial posee una componente cartográfica (mapa geográfico sobre el que se trabaja) y una base de datos que compone lo que se denomina “la tabla de atributos”. En dichas tablas, que básicamente son lo que en estadística se denomina “Matrices de Datos”, se registran las características de cada dato Gis vectorial y sirven para realizar consultas, para unir datos de otras tablas, para realizar cálculos de forma automática, etc. Los Gis vectoriales, permiten representar información tanto de datos cualitativos como cuantitativos de diversas formas: designando símbolos, tamaños, o colores en mapas de puntos; o eligiendo grosor o colores diferentes en mapas de líneas, creando mapas de isolíneas a partir de mapas de puntos por interpolación (TIN) o también utilizando tramas o colores diferentes en mapas de polígonos, etc.

Los formatos de datos vectoriales son diversos y dependen del software GIS que se utilice. Los más comunes son archivos con extensiones: shp, svg, mif, mid, E00, mdb, dgn, dwg, dxf y dxn. Los archivos shp (shapefile) son los que utiliza generalmente QGis. Por decir algunas características, hay que mencionar que los archivos con extensión shp, contienen la geometría. O sea, los puntos o vértices que definen la forma de los elementos geográficos. Los de extensión dbf son archivos del conocido programa DBase y son archivos que contienen una tabla de atributos o descripciones de cada uno de los elementos. Los archivos shx contienen un índice para el manejo de tablas entre archivos y permite facilitar las búsquedas. Los prj contienen la definición del sistema de coordenadas, proyección cartográfica y unidades que utiliza el shapefile para registrar los elementos geográficos. Los de extensión xml, contienen metadatos (es decir, descripción de los geodatos) en un formato estandarizado.

- ✓ **GIS Raster:** En matemáticas raster se ha designado a las mallas o redes (grids), que vienen a conformarse en matrices de unos y ceros que se asocian con pixeles. Por tal motivo, suelo asociarse a las fotos con GIS raster. En caso de que además de unos y ceros se encuentren otros dígitos, se obtendrá un raster a color, en lugar de un raster blanco y negro.

Los Gis raster se basan en análisis y representación de localizaciones espaciales. Se utilizan para representar datos continuos, sin límites marcados, zonas de transición y datos que cambian (espacios naturales, altitudes, precipitaciones, etc.) aunque, de forma más imprecisa, también permiten representar entidades.

En un GIS raster se trabaja con mapas, fotografías de satélites, ortofotografías, en donde la realidad se representa como una retícula rectangular dividida en cuadrículas, celdillas o pixeles de igual tamaño donde las celdas no se solapan: los datos se definen por la posición en una fila y columna, localización relativa, (coordenadas (x,y)).

Dependiendo del tamaño del pixel, la resolución será mejor o peor (a mayor tamaño de pixel, menor resolución). Para poder determinar el tamaño correcto, se puede seguir la norma de utilizar como tamaño del pixel, la mitad de la longitud más pequeña a representar. En caso de que dos valores queden representados en la misma celda, se asignará como valor de la celda, el valor que tenga una mayor presencia o el que quede en el centro. Sobre un mapa se puede crear una retícula para incorporar los valores en formato raster. De esta forma, cada celda o pixel representa una parte del espacio geográfico y tiene unas características propias, valores relacionados con la variable que representa.

Al igual que en los caos de GIS vectorial, también se trabaja utilizando capas con cada grupo de datos y utilizando la misma retícula, de esta forma, se pueden hacer análisis mucho más complejos al relacionar unas capas con otras.

En cuanto a los formatos de archivos utilizados, los datos raster utilizan ASCII y archivos con extensiones: bil, bin, bsq y grid. Los archivos de dibujos, mapas y fotos que normalmente se utilizan en el GIS raster tienen extensiones tiff, jpeg, gif, png y eps.

Cada capa raster produce dos tipos de archivos, en el primero se almacenan los valores de las celdas en el formato ASCII y en el segundo se almacena la información general de la retícula, de la leyenda, orientación, resolución, número de filas y columnas, tipo de variable, etc.

En la siguiente figura, se puede observar gráficamente la diferencia entre modelos de Vector y Raster.

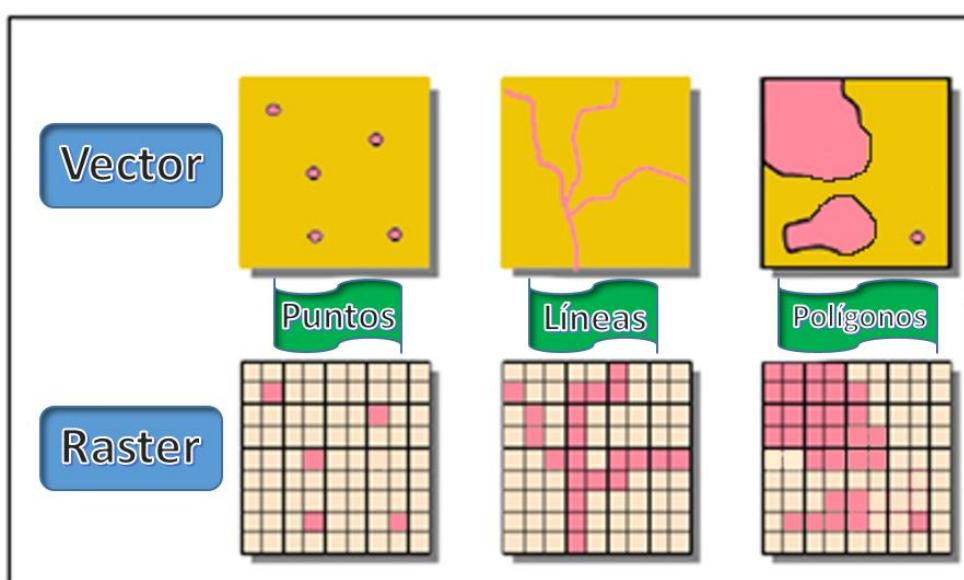


Fig. 4. Diferencia entre modelos de Vector y Raster

○ Documentación de QGis

Al igual que en R, en QGis existe una gran variedad de documentación en la Web sobre el software, sus características y utilidades. En la página Web oficial de QGis⁶ se puede descargar además del software, diversos manuales tanto para usuarios, como para quienes deseen escribir documentos o para desarrolladores. Asimismo, se encuentra en la red, diversos cursos, manuales, prácticas, y foros en los que se pueden consultar una gran variedad de tópicos relacionados con QGis. Asimismo, los textos recomendados en el sílabo y en los que se basa gran parte de la documentación de este curso pueden ser consultados para ampliar y tener una mayor comprensión de las capacidades de QGis en el análisis de datos espaciales.

⁶ <https://www.qgis.org/es/site/forusers/download.html>

3) Instalación de R

Para instalar R el proceso es muy sencillo, al ser un software de libre distribución, los pasos necesarios se presentan a continuación.

- Breve descripción de los pasos de la instalación en sistemas 32 y 64 bits.

- Instalación en Windows de R

Al teclear “r Project”, en un buscador como Google, se presenta en primer lugar la página oficial de R “[R: The R Project for Statistical Computing](#)”. Al dar clic en dicho enlace, se presenta la siguiente imagen:

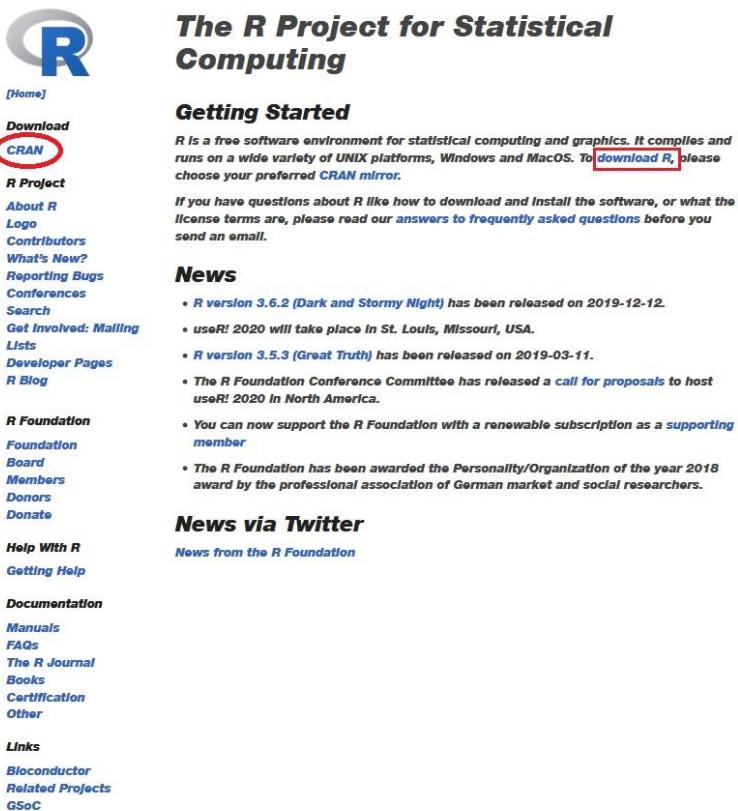


Fig. 5. Página principal de R project

En la parte izquierda bajo “Download”, se observa “[CRAN](#)”, señalado con el círculo rojo, de forma similar en la parte derecha señalado con un cuadro de color rojo se indica “[download R](#)”, ambos enlaces trasladan al dar clic a la siguiente página de descarga de R, a través de distintos servidores espejos (Mirror). Hay que notar que en la parte izquierda también se encuentra la zona desde donde se pueden descargar documentación y ayuda de R.

En la sección “NEWS”, se puede observar la versión más reciente de R, que es este caso es la “3.6.2”. Adicionalmente al dar clic en el enlace “[R version 3.6.2 \(Dark and Stormy Night\)](#)” se nos remite a una página en donde se pueden descargar las diferentes versiones de R en el tiempo.

Una vez que se ha dado clic en el CRAN o la alternativa en la parte derecha, se presenta la siguiente pantalla de los espejos o servidores desde donde se puede descargar la versión de R.

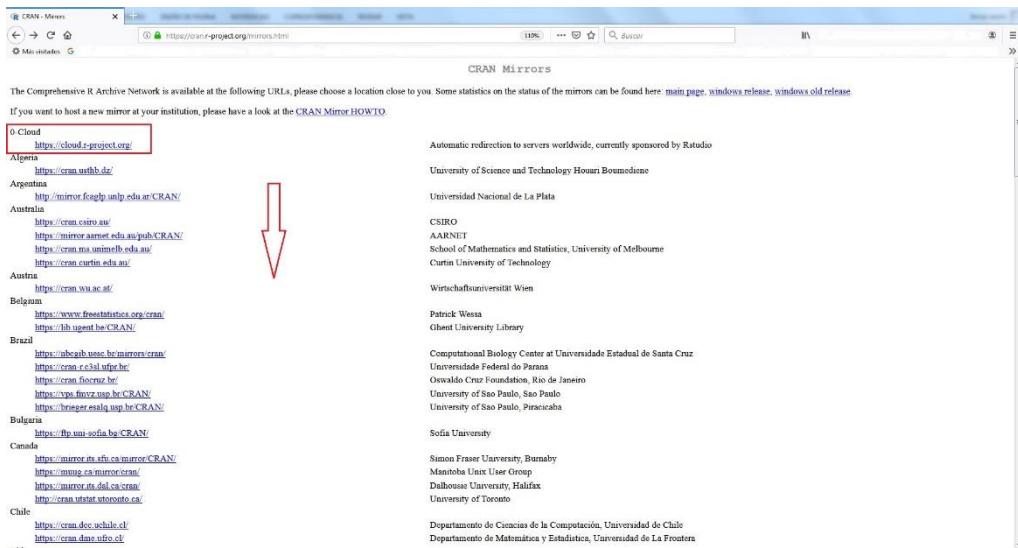


Fig. 6. Página de descarga de CRAN

En esta página se pueden observar los diversos sitios desde donde se puede descargar el software, el primero y que se encuentra señalado mediante el cuadro en rojo, designa a es “[0-Cloud](#)”, es el principal y realiza una redirección automática a la página de descarga de R. Hacia abajo la flecha indica los servidores de los diferentes países desde donde pueden redirigir la descarga. Por ejemplo, México, España, Chile, etc. Al dar clic en ya sea en la página 0-Cloud, o cualquiera de los servidores, se presenta la siguiente imagen.

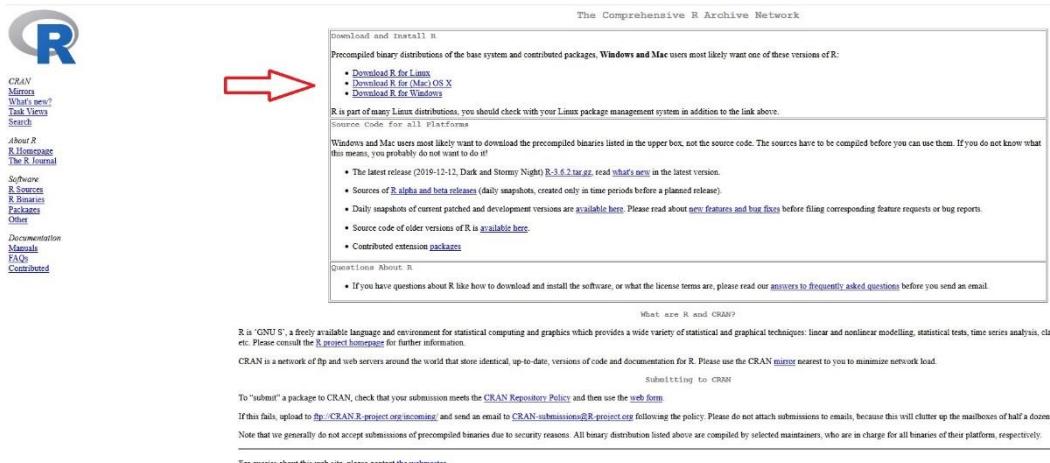


Fig. 7. Página de descarga de O-Cloud

La flecha en rojo en la figura, indica la descarga del software R en las diversas plataformas. Windows, Mac OS y Linux. También si se desea, al ser un software de código libre, se puede descargar el código fuente para que pueda ser editado o mejorado por los usuarios avanzados.

En el caso de sistemas operativos basados en Windows, al dar clic en “[Download R for Windows](#)”, se presenta la siguiente pantalla.

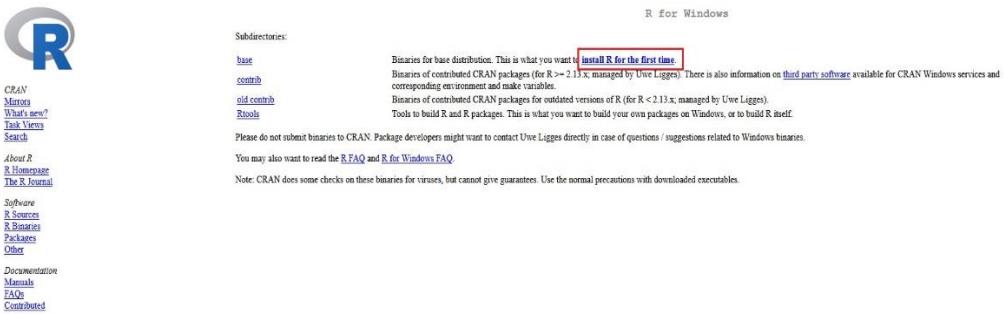


Fig. 8. Enlace para descarga en Windows

En esta pantalla se puede observar, en el cuadro rojo realizar la instalación por primera vez “[install R for the first time](#).” Al dar clic en dicho enlace se presenta la pantalla siguiente:



Fig. 9. Enlace para descargar la última versión de R bajo Windows

En esta pantalla se puede observar en el cuadro rojo señalado con la flecha, que se desea descargar la última versión de R bajo Windows, que en este caso es la 3.6.2. Dicha versión es para máquinas de 32 y 64 bits. Al dar clic en el enlace se presenta la pantalla de descarga automática del software, que es la siguiente.



Fig. 10. Pantalla de descarga de la versión 3.6.2. de R

Se debe dar clic en “Guardar archivo”, para descargar el ejecutable de R (con extensión exe) en el sitio elegido o predestinado de la computadora.



Fig. 11. Archivo ejecutable para instalar R en Windows

Una vez descargado el software, hay que ejecutarlo como administrador, al realizarlo se verá la siguiente secuencia.

- Seleccionar idioma de instalación.



Fig. 12. Seleccionar el idioma de instalación

Para nuestro caso dejamos seleccionado el idioma “Español” y damos clic a aceptar. La pantalla siguiente es el acuerdo de licencia, se debe de leer

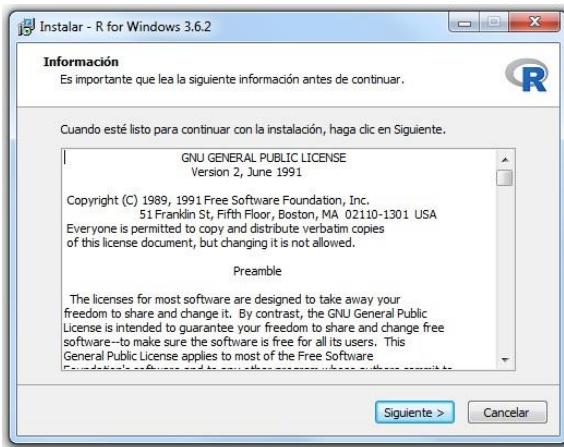


Fig. 13. Acuerdo de licencia del software R

Una vez leído los términos de la licencia, la siguiente pantalla permite seleccionar la carpeta de destino, que es donde se va a instalar el software. Por defecto, la ruta o el camino es: “C:\Program Files\R\R-3.6.2”.



Fig. 14. Selección de la carpeta de destino

Generalmente, se deja establecida esta ruta para la carpeta de destino que será creada, por lo que se da clic a siguiente, lo que presenta la pantalla de selección de los componentes.



Fig. 15. Selección de los componentes

En dicha pantalla como se observa, se encuentran seleccionados los componentes del núcleo (core), los mensajes de traducción y los de 32 bits, al ser la máquina o sistema operativo de 32 bits, en caso de que se necesite instalar para 64 bits, se selecciona dicho componente. El mensaje en la parte inferior indica que se van a necesitar 146,1 MB de espacio en disco. También se puede realizar una instalación a medida o personalizada. En este caso se deja tal y como se muestra la pantalla y se da clic a siguiente. La siguiente pantalla pregunta si se desea utilizar las opciones de configuración.



Fig. 16. Pantalla de selección de opciones de configuración

Se selecciona que “SI”, para facilitar la configuración y se da clic a siguiente. En la siguiente pantalla se selecciona el modo “display”, o desplegar la información.



Fig. 17. Pantalla de modo display

En esta pantalla se selecciona el modo SDI (ventanas separadas), para poder facilitar el uso de RComander. Una vez seleccionado este modo se da clic a siguiente. La siguiente pantalla proporciona el estilo de ayuda de R.



Fig. 18. Pantalla de estilo de ayuda

Se puede optar para que R presente la ayuda en texto simple o como una página Web en formato HTML. En este caso, se deja seleccionado en formato HTML, porque se obtiene una mejor visualización de la información de ayuda y se da clic a siguiente. La pantalla siguiente permite crear la carpeta de menú de inicio.



Fig. 19. Pantalla de selección de la carpeta del menú de inicio

En dicha carpeta es donde se crearán los accesos al programa. Por defecto se encuentra indicada como “R”, por lo que se deja así y se da clic a siguiente. La siguiente pantalla indica la selección de tareas adicionales.

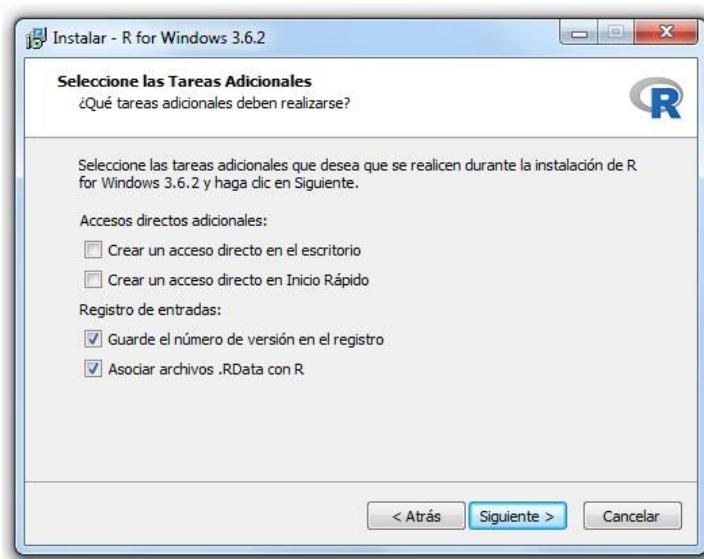


Fig. 20. Pantalla de selección de tareas adicionales

Las tareas adicionales se refieren a accesos directos que se deseen crear en el escritorio y en inicio rápido y el registro de entradas. Si se desean crear se debe de activar las casillas y a continuación dar clic a siguiente. A continuación, comenzará la instalación de los archivos del software en el sistema de la computadora.

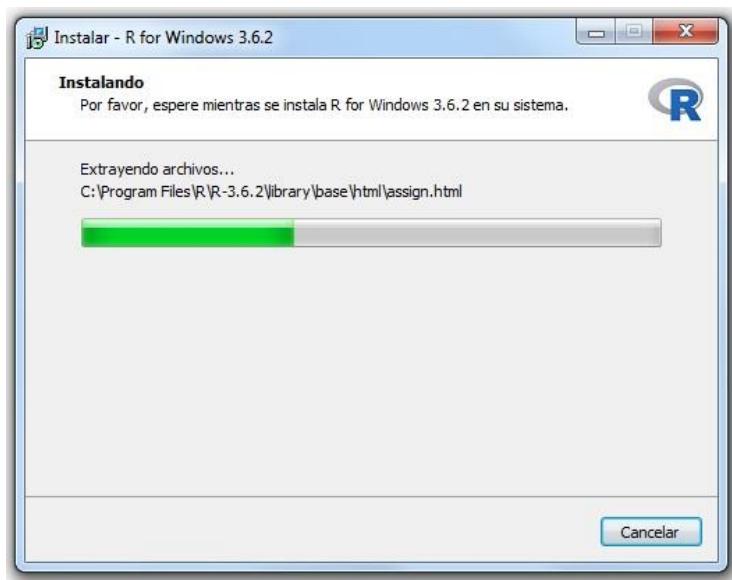


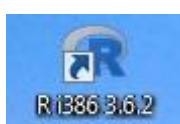
Fig. 21. Pantalla de instalación de archivos en el sistema

Una vez finalizada la instalación de archivos (lo que toma pocos minutos), se presenta la pantalla de finalización de la instalación.



Fig. 22. Pantalla de finalización de la instalación de R en el sistema

Se da clic a “Finalizar”, con lo que termina el proceso de instalación de R en la computadora. En archivos de programa del disco C, debe de estar ya creada la carpeta R y dentro de ella una carpeta denominada “R-3.6.2” en donde se encuentran todos los archivos para utilizar R. Adicionalmente tanto en el escritorio, como en el menú de inicio deben de haberse creado los accesos directos en caso de que se eligiera incluirlos. Ya se puede ejecutar R en la computadora.



- La interfaz gráfica de  , comandos y operaciones básicas

Al ejecutar R, se presenta la interfaz gráfica, que muestra la consola.

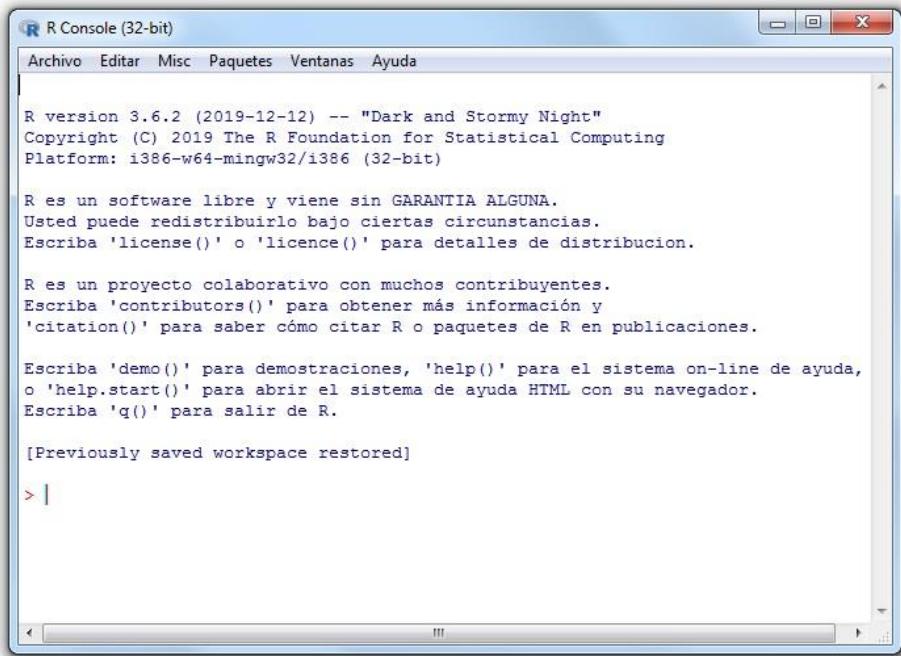


Fig. 23. Pantalla de la interfaz gráfica de R

a) Comandos generales de R

En la imagen anterior se puede observar en la consola el prompt de R “>” en la consola, a partir de este prompt, se escribirán todas las instrucciones en R. Una vez instalado R en el sistema, hay que recordar que una de las primeras actividades a realizar es designar el sitio en donde se deseen que se van a almacenar los datos, programas, etc., que se van a utilizar. Esto es muy conveniente, ya que de otra forma se almacenarán en la ruta por defecto que R indica y dicha ruta es mucho más difícil de cargar que la que nosotros mismos proporcionamos a R. Para realizar esto utilizamos el comando o función “setwd()”, indicando la ruta hacia la carpeta que se desea entre comillas y utilizando la barra “/” invertida y no “\” como Windows la maneja. Por ejemplo, la instrucción que ya se había indicado.

```
> setwd("C:/Users/Lig/Documents/Programas_R")
```

Al dar “Enter”, después de la instrucción el prompt vuelve a estar normal “>”, aparentemente no ha ocurrido nada. Sin embargo, el directorio de trabajo ha cambiado al sitio que se ha indicado. Para comprobarlo, se utiliza la función “> getwd()” y comprobamos el resultado.

```
> getwd()
> [1] "C:/Users/Lig/Documents/Programas_R"
```

R nos muestra el directorio de trabajo actual. Hay que recordar que el número entre corchetes [1] solo indica el número de la instrucción en el resultado.

Para obtener ayuda en R, basta invocar la función “> help()”, lo que abrirá una página Web (como se indicó en la instalación) presentando la documentación interna de R, que se encuentra alojada de manera local en el sistema bajo la dirección: <http://127.0.0.1:18726/library/utils/html/help.html>.

Para buscar información sobre una determinada función, se coloca el nombre de la función entre los paréntesis, sin comillas. Por ejemplo, para buscar información sobre la función “plot”, para graficar, se teclea. “> help(sqrt)”. Lo que abrirá la página Web, en este caso con la documentación de R sobre la función matemática. Cada vez que se utilice “help()” se abrirá una nueva página Web mostrando la documentación requerida.

Hay que mencionar que solo se proporciona la información sobre las funciones y paquetes que se encuentran instalados, en caso contrario, se indicará que R no posee información sobre lo que se encuentra buscando. Para búsqueda que contenga una palabra determinada, se utiliza la función “help.search()”, con la palabra entre comillas, por ejemplo: [help.search(“factor”)], lo que buscará en la documentación todo lo relacionado con la palabra “factor” o factor y lo presentará mediante una página Web.

Otra manera de buscar información sobre determinada función es teclear un signo de interrogación antes de la función que se deseé consultar. Por ejemplo, lo siguiente.

```
> ?sqrt #Presentará la documentación al igual que el comando help
```

Otros comandos de utilidad a tomar en cuenta en R son los siguientes:

```
> ls () # lista los objetos que se encuentran en memoria  
> rm (objeto) # elimina el objeto de la memoria.  
> Ctrl + L # Limpia la consola de trabajo  
> dir () # Presenta el contenido del directorio de trabajo
```

Las funciones en R vienen contenidas en librerías o paquetes que han de ser instalados. Para saber en qué paquete está contenida una función se utiliza “find()”

```
> find("median") # El nombre de la función entre comillas  
[1] "package:stats" # R indica que el paquete es "stats": Paquete de funciones estadísticas
```

A medida que el presente curso avance, se irán presentando muchos más comandos e instrucciones de utilidad para los objetivos que se requieran.

b) Barra de herramientas de menú y sus secciones

Uno de los principales objetos es la barra de Menú principal de R, presenta los elementos o secciones principales de R, y que se observan en la siguiente figura.

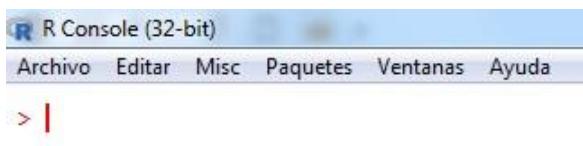


Fig. 24. Pantalla de barra de herramientas del menú de R

Las secciones como se observa son las siguientes: **Archivo**, **Editar**, **Misc**, **Paquetes** y **Ayuda**.

NOTA: No es el objetivo de este curso profundizar en cada una de las opciones o elementos que componen las secciones, sin embargo, aquí se mostrarán con sus principales características. Algunas de ellas se verán durante el desarrollo del curso más adelante.

✓ Sección de archivo

Esta sección se refiere a todo lo referente a los archivos y la salida de R.



Fig. 25. Pantalla de sección de Archivo

Los elementos de esta sección son los siguientes:

- i. **Interpretar código fuente R....:** Con esta opción se puede ejecutar un programa en la consola siempre y cuando, la extensión de guardado sea del tipo **.R**.
- ii. **Nuevo script:** Al ejecutar, se abre un editor de lenguaje R para crear un script que posteriormente se llama desde la consola, el tipo de archivo que se genera es **R o S**.
- iii. **Abrir script:** Los scripts generados y almacenados en la carpeta destinada para los archivos del tipo R o S se abren en un editor para poder modificarlos.
- iv. **Mostrar archivo(s)...:** Con esta instrucción se abre, visualizar o edita cualquier archivo. La diferencia con la opción *Abrir script* de que éste no puede editarlos directamente, solamente se visualizan. También, pueden abrirse varios archivos a la vez.
- v. **Cargar área de trabajo....:** Esta instrucción carga un área de trabajo que previamente se ha configurado y guardado, como por ejemplo se ha definido un tipo de letra, color, el espacio de trabajo, etc., y que se va a utilizar. La extensión por defecto es **.RData** Sin embargo, también pueden cargarse entornos de trabajo con el formato anterior que tiene la extensión “**.rda**”.
- vi. **Guardar área de trabajo....:** Al realizar cambios en el entorno de trabajo, generalmente R no guarda la configuración, pero este elemento sirve para guardar dicha configuración, tipos de letras, colores, etc, para poder ser utilizado posteriormente. El formato del entorno de trabajo es **.RData**.
- vii. **Cargar Histórico....:** Con esta instrucción se carga el archivo histórico de comandos que se hayan ejecutado en una sesión previamente guardada. El formato de salida es **.history**.
- viii. **Guardar Histórico....:** Mediante esta instrucción se guarda el histórico de comandos introducidos una sesión. El formato de guardado es **.history**.
- ix. **Cambiar dir....:** A través de esta instrucción se modifica el directorio de trabajo, ya sea el que se guardó por defecto durante la instalación o el que se tiene actualmente designado.
- x. **Imprimir....:** Por medio de esta instrucción se puede configurar e imprimir el entorno de trabajo de R, la consola.
- xi. **Guardar en archivo....:** Esta instrucción guarda el contenido escrito por la consola formato **.txt** que posteriormente se puede recuperar.
- xii. **Salir:** Instrucción para salir del programa R, antes preguntará si se desea guardar el área de trabajo. Para salir también se puede teclear en la consola **q()**.

✓ Sección de Editar

En esta sección se configura el entorno de trabajo y se realiza la edición de la consola de R.

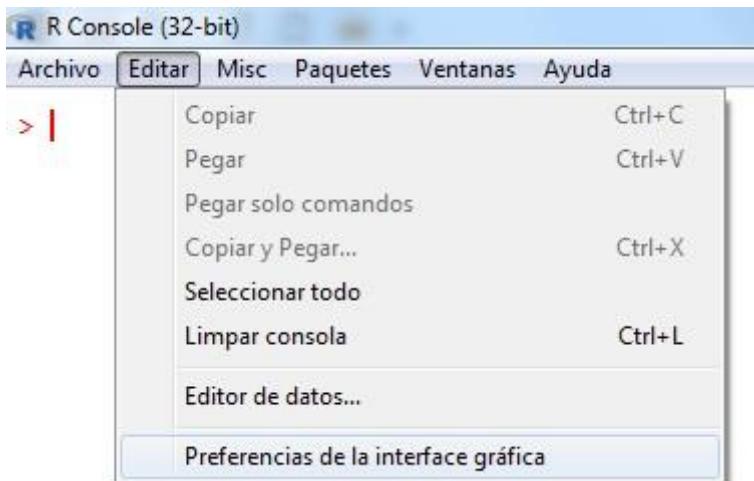


Fig. 26. Pantalla de sección de Editar

Los elementos de esta sección son los siguientes:

- i. **Copiar:** Por medio de esta instrucción se copia al portapapeles el comando, sentencia o cualquier instrucción que se necesite de la consola. También se puede realizar por medio de: **Ctrl + C**.
- ii. **Pegar:** Realiza una copia en la consola del contenido del portapapeles. Igualmente, al pulsar: **Ctrl + V**.
- iii. **Pegar solo comandos:** Como su nombre lo indica solo acepta comandos para pegar en la consola para ejecutarse.
- iv. **Copiar y Pegar:** Con esta instrucción, todo lo que se copie se pegará de forma automática, en la consola. Por medio del teclado se debe de pulsar: **Ctrl + X**.
- v. **Seleccionar todo:** Esta instrucción selecciona todo lo que se encuentre en la consola.
- vi. **Limpiar consola:** Con esta instrucción se borra el contenido de la consola, es igual a limpiar el escritorio de trabajo. Se borrará todo lo que esté presente en la consola. Sin embargo, no se borran las variables y estructuras definidas. A través del teclado se debe de pulsar: **Ctrl + L**.
- vii. **Editor de datos...:** Se utiliza para definir datos (vectores, estructuras, funciones, etc) y almacenarlos para posteriormente llamarlos en la consola.



Fig. 27. Pantalla del Editor de datos

- viii. **Preferencias de la interfaz gráfica:** A través de esta opción se configura el entorno de trabajo de la consola: tamaño de letra, colores, tipo de letra, etc. Funciona también para múltiples ventanas o para una única ventana.

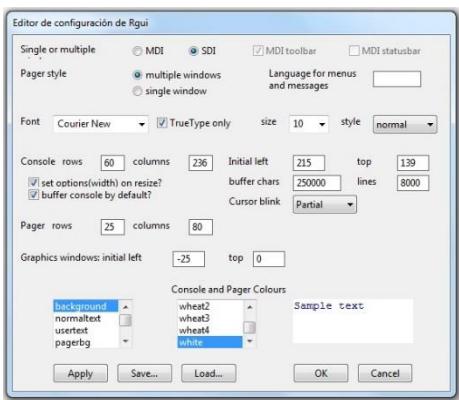


Fig. 28. Pantalla de Preferencias de la Interfaz Gráfica

✓ Sección de Miscelánea (Misc)

Dispone de varios elementos para manejo de archivos y objetos

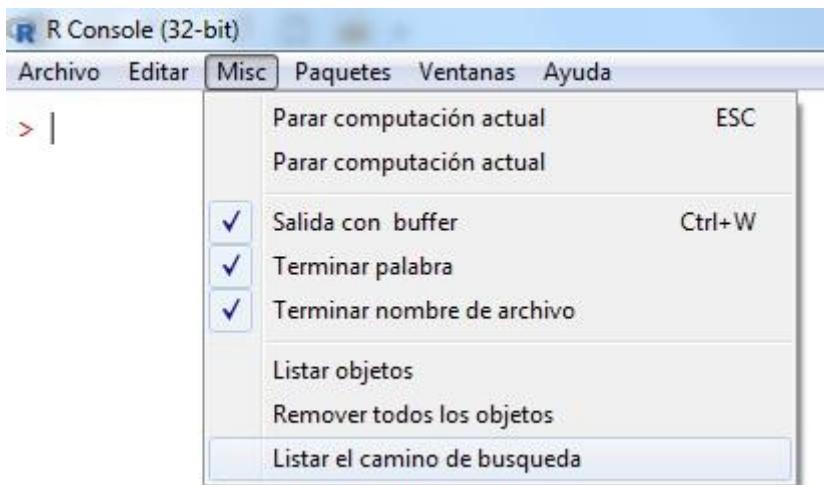


Fig. 28. Pantalla de sección de miscelánea (Misc)

Los elementos de esta sección son los siguientes:

- Parar computación actual:** Mediante esta instrucción se puede detener o parar cualquier archivo, sentencia o código que esté ejecutando en ese momento la consola de R. También se puede realizar mediante: **ESC**.
- Salida con buffer:** Realiza una salida con buffer, también mediante las teclas: **Ctrl + W**.
- Terminar palabra:** Esta instrucción permite una ayuda interactiva, que permite completar las palabras mientras se escriben, siempre y cuando la consola las reconozca.
- Terminar nombre de archivo:** Funciona igual que la opción “Terminar palabra”, solo que con archivos.
- Listar objetos:** Esta opción muestra en la consola todos los objetos que se han definido por la consola.
- Remover todos los objetos:** Esta opción debe de tratarse con cuidado debido a que elimina de memoria todos los objetos que se hayan definido (datos, variables, matrices, vectores, etc) en la consola de R. Al seleccionar esta opción, el programa, nos preguntará si realmente se desea borrar los objetos.
- Listar el camino de búsqueda:** Presenta por consola las librerías y complementos que están instalados en R. Funciona de la misma manera que la función “> search()”

Ejemplo

```
> search()
[1] ".GlobalEnv"      "package:stats"    "package:graphics" "package:grDevices" "package:utils"
"package:datasets"   "package:methods"   "Autoloads"        "package:base"
```

✓ Sección de Paquetes.

Esta sección se encarga del manejo de las librerías que utiliza R

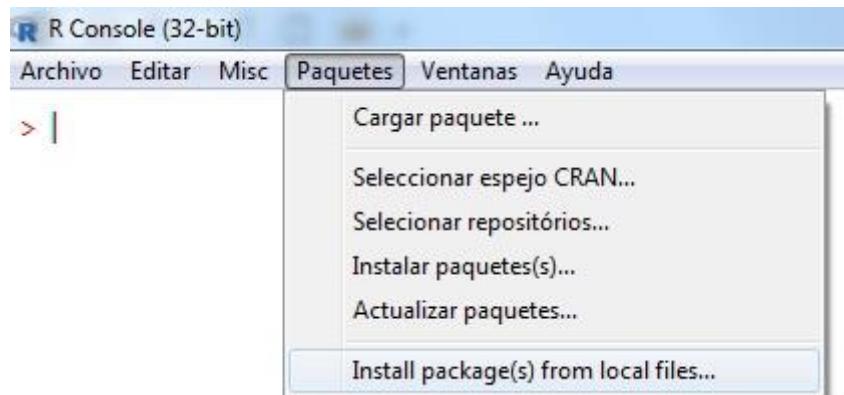


Fig. 29. Pantalla de sección de Paquetes

Los elementos de esta sección son los siguientes:

- i. **Cargar paquete...**: A través de esta opción se cargan las librerías o paquetes de R para ser utilizados. Se presenta una lista de paquetes disponibles y se selecciona el que se desea cargar en el sistema.
- ii. **Seleccionar espejo CRAN...**: Esta opción permite configurar el servidor de librerías. Funciona igual que cuando se instaló R, presenta una lista de los servidores O-cloud y los países para elegir desde el cual se desea descargar las librerías.
- iii. **Seleccionar repositorios....**: Esta opción muestra en consola los repositorios instalados y pedirá cual se desea utilizar para la sesión activa.
- iv. **Instalar paquetes(s)...**: Esta opción brinda la posibilidad de actualizar o instalar librerías nuevas en red, para ello, se debe de elegir un servidor de la lista que se presenta.
- v. **Actualizar paquetes(s)...**: Esta opción permite actualizar librerías instaladas en caso de haber una actualización reciente por red, previamente se debe seleccionar un servidor de la lista que se muestra.
- vi. **Instalar paquetes(s) a partir de archivos zip locales...**: Con esta opción se instalan paquetes que previamente han sido descargados del CRAN y almacenados en la computadora.

✓ Sección de Ventanas

Organiza las ventanas de Windows de la forma que el usuario mejor le convenga.



Fig. 30. Pantalla de sección de Ventanas

Funciona organizando las ventanas de Windows, para presentar la información y que así le sea de una mayor utilidad al usuario.

✓ Sección de Ayuda

Presenta una serie de elementos para presentar la ayuda y manuales que R dispone para el usuario.

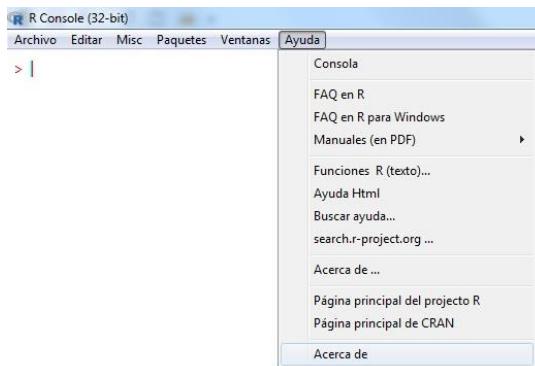


Fig. 31. Pantalla de sección de Ayuda

Los elementos de esta sección son los siguientes:

- i. **Console:** Presenta en una ventana todos los atajos que por teclado se pueden realizar en la consola.
- ii. **FAQ en R:** Presenta una ayuda de documentación por medio de una página Web sobre las preguntas frecuentes los usuarios hacen al utilizar R.
- iii. **FAQ en R para Windows:** Preguntas frecuentes para los usuarios de Windows.
- iv. **Manuales en PDF:** Presenta una lista en PDF de los manuales para el manejo de R.
- v. **Funciones R(texto)...:** Por medio de esta opción se puede introducir sentencias de comando para que se presenten ayudas sobre dichas instrucciones.
- vi. **Ayuda Html:** Presenta una página Web con una ayuda interactiva.
- vii. **Buscar ayuda...** Presenta una ventana en la que se escribe la instrucción sobre la que se desea buscar la ayuda.
- viii. **Html search page:** Presenta una ventana en la que se escriben palabras para buscar en listas de archivos de ayuda y documentación. Se presenta por medio de una página Web.
- ix. **Acerca de....** Una ventana en la que se escribe una función, la cual presenta en consola todo lo referente a dicha función. Es igual a teclear la función “>apropos("funcion")”.
- x. **Página principal R:** Enlace directo a la página Web de R.
- xi. **Página principal de CRAN:** Enlace directo al directorio principal (CRAN) de librerías de R.
- xii. **Acerca de:** Presenta la información sobre la versión que se tenga instalada de R.

c) Operaciones y funciones básicas con R

R maneja operadores básicos al igual que otro lenguaje de programación, lo que le da una gran potencia de cálculo y permite realizar cualquier análisis estadístico.

Operadores Básicos en R			
Aritméticos		Comparativos	
		Lógicos	
+	Adición	==	Igual a
-	Substracción	!=	Diferente de
*	Multiplicación	<	Menor que
/	División	>	Mayor que
^	Potencia (Alt+94)	<=	Menor o Igual que
% / %	División Entera	>=	Mayor o Igual que
			is.na(x) Ausente?

Fig. 32. Operadores básicos en R

Como se observa en la tabla anterior, se manejan los operadores aritméticos elementales, al igual que los comparativos y lógicos. Mediante ellos se pueden construir todas las fórmulas

matemáticas y estadísticas que se deseen para el análisis de datos. Se puede visualizar una ayuda de las funciones mediante la función “demo(plotmath)”. Veamos ejemplos de esto a continuación:

- | | |
|-------------------------|--|
| a) > 123 + 435 | # Ejemplo de suma |
| [1] 558 | |
| b) > 435 - 123 | # Ejemplo de resta |
| [1] 312 | |
| c) >(123 + 435) * 8 | # Ejemplo de suma y multiplicación |
| [1] 4464 | |
| d) >((123+435)*8)/4 | # Ejemplo de suma, multiplicación y división |
| [1] 1116 | |
| e) >(((123+435)*8)/4)^3 | # Ejemplo de suma, multiplicación, división y potencia |
| [1] 1389928896 | |
| f) > 435% 123 | # Ejemplo de división entera |
| [1] 66 | |

Es importante tener en cuenta el uso del paréntesis, debido a que los resultados obtenidos no serán los mismos. Por ejemplo

- | | |
|-------------------------|--|
| g) >(11/6)+(9/4) | |
| [1] 4.083333 | |
| h) >11 / ((6 + 9) / 4) | |
| [1] 2.933333 | |

Un listado de las funciones básicas tanto matemáticas como aritméticas en R se puede observar en la siguiente figura.

Funciones Básicas en R	
Matemáticas	Estadísticas
sqrt(x)	Raíz de x
exp(x)	Exponencial de x
log(x)	Logaritmo natural de x
log10(x)	Logaritmo base 10
length(x)	Número de elementos
sum(x)	Suma los elementos de x
prod(x)	Producto de los elementos
sin(x)	Seno
cos(x)	Coseno
tan(x)	Tangente
round(x,n)	redondea a n dígitos
cumsum(x)	calcula las sumas acumuladas $x_1, x_1 + x_2, x_1 + x_2 + x_3,$ $x_1 + x_2 + \dots + x_n$
	mean(x)
	sd(x)
	var(x)
	median(x)
	quantile(x,p)
	cor(x,y)
	max(x)
	min(x)
	range(x)
	sort(x)
	which(condición)
	summary
	IQR(x)
	choose(n, k)
	Media
	Desviación típica
	Varianza
	Mediana
	Quantiles
	Correlación
	El máximo
	El mínimo
	Retorna el máximo y mínimo
	Ordena las componentes de x
	los índices que cumplen la condición
	Resumen de las variables
	Rango Intercuartílico: $IQR = Q_3 - Q_1$
	número combinatorio de n sobre k

Fig. 33. Funciones básicas en R

Para la raíz cuadrada se utiliza la función “sqrt (en inglés square root)”. Recordando que R distingue entre mayúsculas y minúsculas.

- | | |
|------------------------------|----------------------------|
| i) > sqrt(((123+435)*8)/4)^3 | # Ejemplo de raíz cuadrada |
| [1] 37281.75 | |

Algunos ejemplos con ejercicios de las funciones matemáticas básicas son los siguientes:

- | | |
|--------------|------------------------------------|
| j) > exp(7) | # Ejemplo de exponencial (e^x) |
| [1] 1096.633 | |

```

k) > log(exp(7))          # Ejemplo de logaritmo natural (función inversa)
[1] 7
l) > log10(7)             # Ejemplo de logaritmo base 10
[1] 0.845098
m) > length(7)           # Ejemplo de número de elementos
[1] 1
n) > sum(1:5)             # Suma los elementos de 1 a 5
[1] 15
o) > prod(1:5)            # Producto de los elementos de 1 a 5
[1] 120
p) > sin(45)              # Cálculo del seno de 45
[1] 0.8509035
q) > cos(45)              # Cálculo del coseno de 45
[1] 0.525322
r) > tan(45)               # Cálculo de la tangente de 45
[1] 1.61977

```

Algunos ejemplos con ejercicios de las funciones estadísticas básicas son los siguientes:

Primero se procederá a introducir un conjunto de datos {dato1, dato2, dato3,...}, correspondientes a un grupo de 24 personas. A este conjunto de datos se le asignará un nombre, denominado “edad” (objeto), ya que corresponden a la edad de los mismos, de la siguiente manera.

```
(edad = c(18, 19, 24, 17, 20, 17, 21, 23, 19, 25, 22, 16, 19, 17, 21, 29, 22, 18, 21, 20, 21, 23, 22, 19))
```

Como se observa, en R se construye un conjunto de datos (más adelante se verá el tema de vectores), utilizando la letra “c” y a ese conjunto se le ha asignado el título “edad” (más adelante se verá el tema de las variables). La función “**c()**” se utiliza para concatenar objetos, aunque resulta útil para crear vectores.

En R las asignaciones se realizan utilizando el símbolo “=” o bien mediante el operador de asignación “**<-**”. De esta forma, la misma expresión podría escribirse de la siguiente manera

```
(edad <- c(18, 19, 24, 17, 20, 17, 21, 23, 19, 25, 22, 16, 19, 17, 21, 29, 22, 18, 21, 20, 21, 23, 22, 19))
```

```
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

Para saber lo que contienen los nombres o variables, solo hay que nombrarlas en la consola de R. El programa presentará el contenido del nombre o expresión indicada.

```
> edad
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

Una vez ingresado el conjunto de datos, se procederá a realizar los cálculos con las funciones estadísticas básicas. Hay que tomar en cuenta que las funciones **var** y **sd** no calculan la varianza y la desviación típica de una variable, sino su cuasi-varianza y su cuasi-desviación típica. En caso de necesitar la varianza o la desviación típica, basta con multiplicar el resultado de las funciones **var** y **sd** por $(n - 1)/n$, siendo n el número total de datos con el que se está trabajando.

```

a) > mean(edad)           # Cálculo de la media de la edad
[1] 20.54167
b) > median (edad)        # Cálculo de la mediana de la edad
[1] 20.5
c) > sd(edad)             # Cálculo de la cuasi-desviación Típica

```

```

[1] 2.977877
d) > round(sd(edad),2)          # Redondea la desviación estándar a 2 decimales
[1] 2.98
e) > var(edad)                 # Cálculo de la cuasi-varianza
[1] 8.867754
f) > Varianza_Edad <- 23/24 * var(edad)  # Cálculo de la varianza
[1] 8.498264
g) > Desvt_Edad <- sqrt(Varianza_Edad) # Cálculo de la Desviación Estándar
[1] 2.915178
h) > max(edad)                 # Cálculo del valor máximo
[1] 29
i) > min(edad)                 # Cálculo del valor mínimo
[1] 16
j) > range(edad)               # Cálculo del rango [min –max]
[1] 16 29
k) > sort(edad)                # Ordena los componentes de edad
[1] 16 17 17 17 18 18 19 19 19 19 20 20 21 21 21 21 22 22 22 23 23 23 24 25 29
i) > quantile(edad, 0.25)      # Cálculo del Cuantil Q1
25%
18.75
j) > quantile(edad, 0.75)      # Cálculo del Cuantil Q3
75%
22
k) > quantile(edad, probs = c(0.25, 0.75)) # Cálculo de los cuantiles indicando probabilidad
25% 75%
18.75 22.00
l) > quantile(edad, probs = seq(0, 1, 0.25), na.rm = FALSE)
0% 25% 50% 75% 100%
16.00 18.75 20.50 22.00 29.00
m) > summary(edad)            # Resumen estadístico del objeto “edad”
Min. 1st Qu. Median Mean 3rd Qu. Max.
16.00 18.75 20.50 20.54 22.00 29.00

```

NOTA: **na.rm**: Es un argumento lógico, que indica si hay que eliminar los valores faltantes del conjunto de datos.

n) > IQR(edad) # Cálculo del Rango Intercuartílico: $IQR = Q_3 - Q_1$
[1] 3.25

NOTA: Recordemos que los **cuantiles** son aquellos valores de la variable, que ordenados de menor a mayor, dividen a la distribución en partes, de tal manera que cada una de ellas contiene el mismo número de frecuencias. A su vez, en **estadística** descriptiva, se le llama **rango intercuartílico** o **rango intercuartil**, a la diferencia entre el tercer y el primer cuartil de una distribución. Es una medida de la dispersión **estadística**: $IQR = Q_3 - Q_1$.

Ejercicio propuesto:

Un centro de estudio, ha determinado que las horas laboradas por la plantilla docente, compuesta por 20 profesores durante el pasado año han sido las siguientes:

1235,1925,1850,1500,2015,1925,1750,1967,925,1500,1714,955,1800,1645,1992,1985,1555,1956,1962,2015

- Proceda a construir un conjunto de datos, asignarle un nombre e ingresarlos en la consola de R.
- Realice en base a los datos, los cálculos con las funciones estadísticas básicas y anote los resultados

4) Instalación de R Studio

Como ya se ha mencionado, RStudio es un entorno de desarrollo integrado (IDE), pensado para R en el que al igual que la interfaz GUI de R se pueden desarrollar programas, análisis estadísticos y gráficos. Al igual que R, el modelo de desarrollo es de software libre, por lo que es gratuito tanto en la versión para escritorio (PC) o la versión para servidor. Sin embargo, existen y se ofrecen dentro de RStudio versiones comerciales de pago, tanto para escritorio como para servidor con características adicionales. En este curso se trabajará con la versión gratuita para escritorio.

o Introducción e Instalación básica de RStudio

Una vez que se ha instalado R, se puede descargar e instalar RStudio, ya que este funciona sobre la base de R. Para descargarlo se debe de acceder a la página Web de [RStudio](#).



Fig. 34. Pantalla principal de RStudio

Como se observa, hay que dirigirse a la sección de descargas (download) marcada con la flecha y el cuadrado rojo. Al dar clic en este enlace se abre una página en la que se elige la versión que se desea descargar. Aquí hay que elegir la versión gratuita para escritorio.

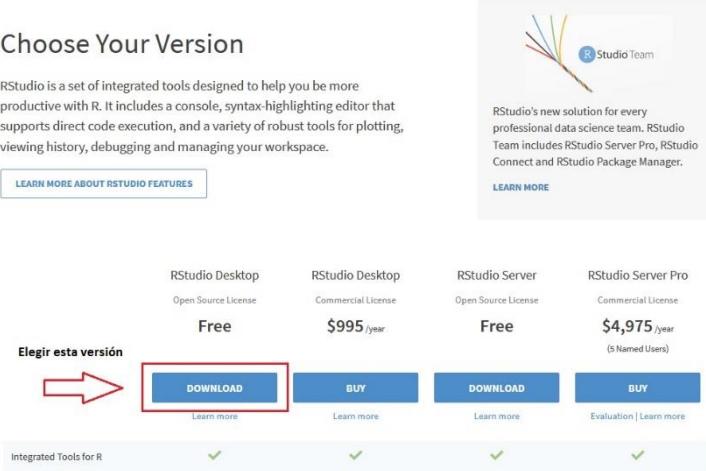


Fig. 35. Pantalla de selección de la versión

Una vez en la página Web de descarga, se debe de elegir el software de acuerdo a si la versión del sistema es de 32 o 64 bits. La versión más actual funciona con Windows 7,8 y 10 de 64 bits. Para máquinas con 32 bits se debe de seleccionar descargar el software en versiones anteriores.

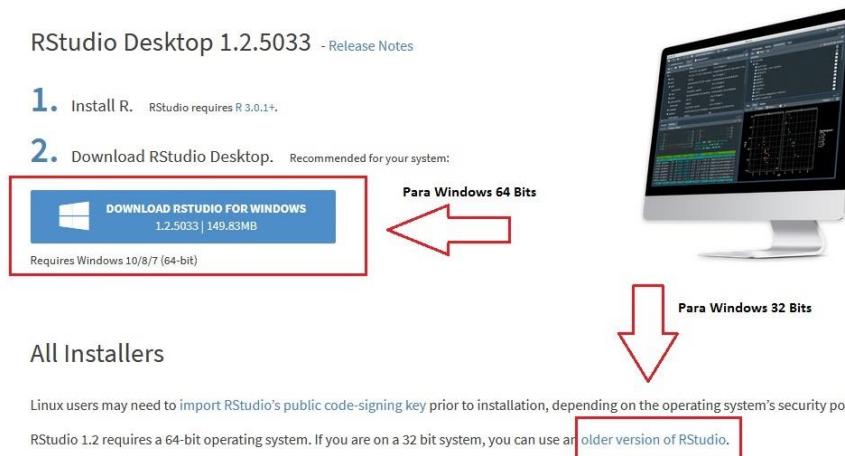


Fig. 36. Pantalla de descarga para RStudio de 32 y 64bits

La descarga para sistemas de 64 bits se realiza automáticamente, para la de 32 bit se despliega una pantalla que permite elegir tanto el sistema operativo, como si se desea descargar el archivo ejecutable (EXE) o un archivo ZIP que contiene todo lo necesario. Preferentemente elegir el ejecutable.



Fig. 37. Archivos ejecutables de RStudio para 32 y 64 bits respectivamente

Una vez descargado el ejecutable, se ejecuta como administrador y se presenta la pantalla de bienvenida del asistente de instalación de RStudio.



Fig. 38. Pantalla de inicio del asistente de instalación de RStudio

En esta pantalla para continuar la instalación, se debe de dar clic a siguiente, lo que presentará la pantalla de selección del sitio en donde se desea instalar RStudio.

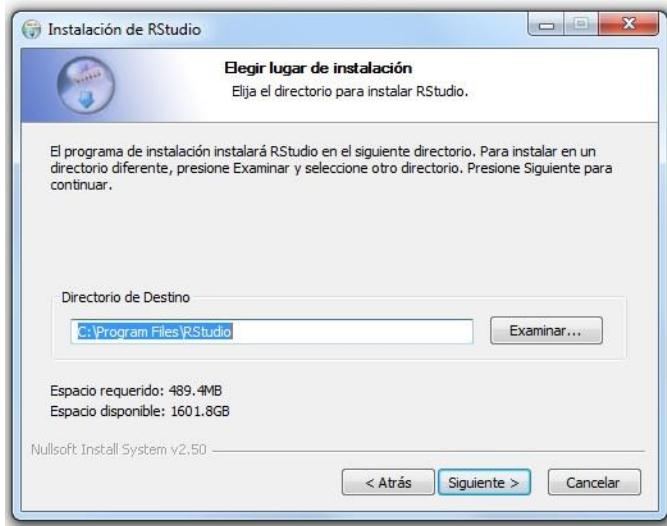


Fig. 39. Pantalla de selección del lugar de instalación

Por defecto, RStudio se instalará en la dirección “C:\Program Files\RStudio”, en este caso se va a dejar que RStudio instale en el sitio por defecto. RStudio ocupará 489.4 MB. Para continuar la instalación se da clic a siguiente. Lo que presentará la pantalla de elegir la carpeta del menú de inicio.



Fig. 40. Pantalla de elegir la carpeta del menú de inicio

Por defecto, se indica que la carpeta a crear va a ser “RStudio”, aunque se puede modificar o crear una nueva carpeta. En este caso se deja tal y como lo sugiere el software y se da clic a Instalar. Se presenta la pantalla del proceso de instalación de los archivos en el sistema.

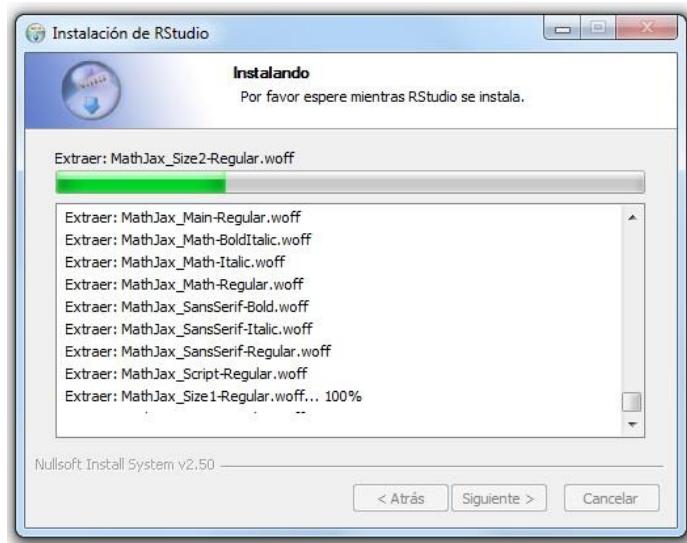


Fig. 41. Pantalla de instalación de los archivos de RStudio en el sistema

En la pantalla anterior, se ha dado clic a “Ver detalles”, para observar el proceso de instalación. Este proceso es bastante rápido y una vez finalizado se presenta la pantalla del asistente indicando que el proceso de instalación de los archivos ha finalizado.



Fig. 42. Pantalla de finalización del asistente de instalación de RStudio

Una vez completado el proceso de instalación, se da clic a “Finalizar” lo que cierra el asistente de instalación de RStudio. Los archivos han sido instalados en la ruta especificada y se cuenta con un acceso directo en el menú de inicio de Windows. Es conveniente hacer un acceso directo al escritorio o a la barra de tareas, para tener acceso rápido al software.

Al iniciar RStudio, se presenta la interfaz gráfica de RStudio, que es la siguiente.

○ Interfaz gráfica de R Studio

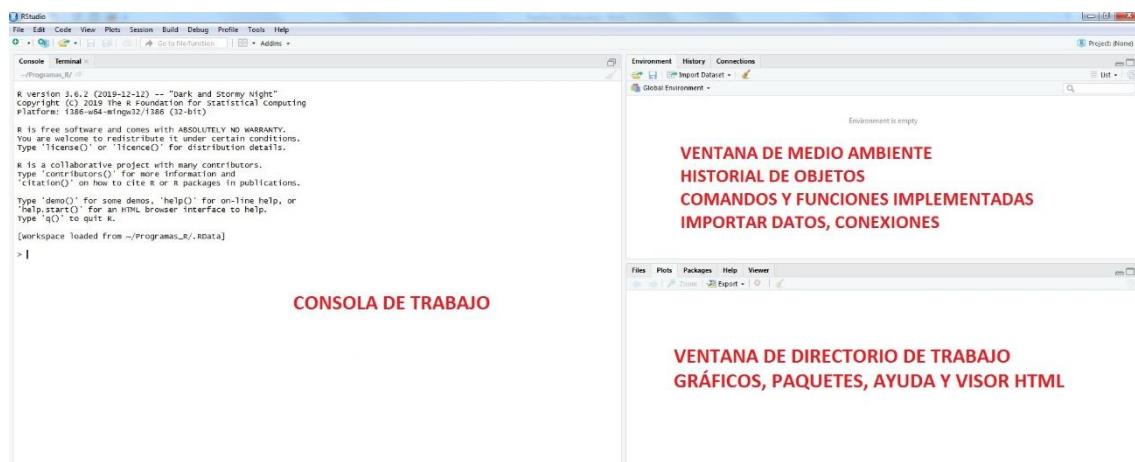


Fig. 43. Pantalla de la interfaz gráfica de RStudio

La interfaz gráfica presenta en un inicio tres partes que se observan en la pantalla:

- La consola/Terminal: Es donde se encuentra al igual que en R, el prompt “>”. En esta pantalla se puede ver la versión de R bajo la cual está funcionando RStudio que es la 3.6.2. El terminal es similar a ejecutar R nativo, como el presentado en la figura 2.
- En la parte superior derecha se observa la ventana de medio ambiente. Aquí se encuentra el historial de objetos guardados en memoria. También, aquí se puede realizar una limpieza del historial, importar datos y mostrar en el historial los comandos y funciones implementados.
- En la parte inferior derecha, RStudio muestra el directorio de trabajo, los gráficos que se van creando, los paquetes para cargarlos e instalarlos directamente, ayuda y un visor HTML.

Nota: Cuando se abre RStudio y previamente se ha estado trabajando con R, hay que recordar que RStudio trabaja con base en R, por lo que se observará en la parte de la ventana de historial, todos los objetos que hasta ahora se han trabajado en R. Si se desea limpiar la pantalla del historial se da clic en la herramienta “escobilla”, situada en la barra de herramientas de la ventana de medio ambiente que se observa en la siguiente figura marcada con la flecha y el círculo en color rojo.

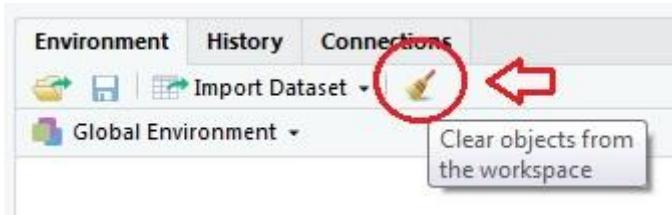


Fig. 44. Herramienta para limpiar los objetos de la pantalla de medio ambiente

Al pulsar dicha herramienta, se presenta una ventana de advertencia, en la que se pide la confirmación en la que se indica que se van a borrar del historial todos los objetos (no podrá revertirse y los objetos no podrán ser utilizados a menos que se vuelvan a cargar), por lo que pregunta si se está seguro de realizar dicha operación. La siguiente figura presenta dicha pregunta.



Fig. 45. Pantalla de advertencia para confirmar el borrado de los objetos

Hay que recalcar que si en la consola se vuelve a teclear alguno de los comandos o funciones que ya se habían trabajado, aparecería un error, como por ejemplo el siguiente:

```
> mean(edad)
Error in mean(edad) : object 'edad' not found
```

Indica que el objeto “edad” no se encuentra, ya que ha sido borrado del historial. Por lo mismo, se debe de tener precaución al utilizar el borrado de objetos, si se van a seguir utilizando los objetos que previamente se han trabajado.

TRUCO: Una opción que se puede realizar para rescatar los objetos perdidos, consiste en que se debe de cerrar RStudio y al aparecer si se desea guardar los cambios efectuados decir que “NO”, con lo que se conservará la configuración anterior, es decir la que se tenía antes de borrar el historial, con esto al abrir nuevamente RStudio, se presentará el historial de objetos trabajados y podrán utilizarse las funciones anteriores. Ahora después de abrir nuevamente RStudio, al ejecutar la anterior expresión, se presentará lo siguiente.

```
> mean(edad)
[1] 20.54167
```

Uno de los motivos por lo que es importante conservar el historial, es por si cualesquiera de las expresiones o funciones utilizadas se necesitan ejecutar nuevamente. Por ejemplo, al dar clic en la parte “History” o historial de la ventana de trabajo de medio ambiente, se observan las expresiones acumuladas como se muestra en la figura.

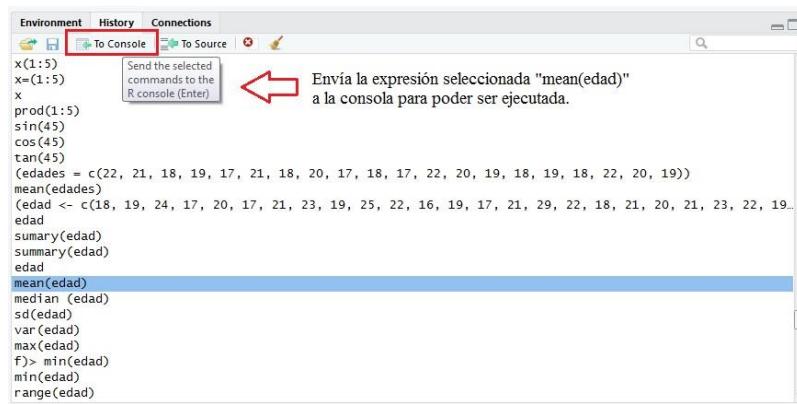


Fig. 46. Pantalla del historial de funciones y comandos en RStudio

En este ejemplo se ha seleccionado la expresión “mean(edad)” y mediante el botón “To Console”, señalado en el cuadro rojo, se envía dicho comando a la consola para poder ser ejecutado.

A lo largo de este curso se van a analizar algunas de las partes en que RStudio está constituido. Sin embargo, en el alcance de este curso no se podrá desarrollar al completo todos los puntos y

servicios que tanto R, como las interfaces de usuario como RStudio y RCmdr ofrecen. Esto implicaría realizar un curso más avanzado de R. Queda, por tanto, de parte del alumnado, profundizar a futuro sobre los servicios y elementos que se deseen analizar. En este curso, se tratarán los más puntos más elementales y básicos a fin de lograr los objetivos del análisis estadístico de datos propuestos.

- Partes de la ventana y menús de 

El acceso a los elementos y ventanas del menú se realiza desde la barra de herramientas de menú de RStudio

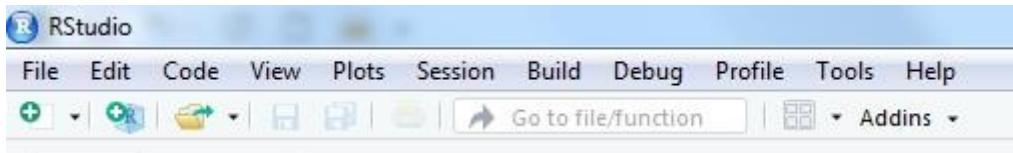


Fig. 47. Barra de herramienta de menú de RStudio

Las secciones como se observa son las siguientes: **File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools y Help**.

A continuación, de manera general se presentarán los diferentes elementos de las secciones del menú de opciones de RStudio.

- ✓ Sección File

En esta sección se presentan las opciones generales para el manejo de archivos.

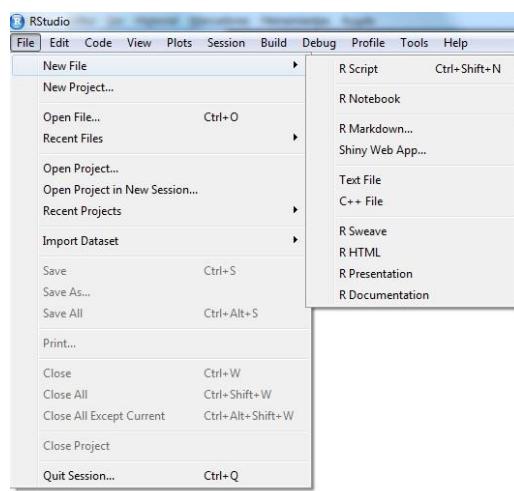


Fig. 48. Pantalla de sección File

Como se observa en la figura, esta sección básicamente se refiere a archivos y proyectos. Los proyectos son de gran utilidad, pues RStudio dispone de un soporte integrado para asociar y almacenar en un mismo sitio tanto datos de entrada, programas o scripts, gráficos y resultados, a todo ello se le conoce como proyectos. De esta forma se puede organizar mucho mejor el trabajo. Más adelante se tratará el tema de los proyectos. Por ahora el punto a tratar será con los archivos.

En la pantalla anterior, se observa la opción “New File” o nuevo archivo cuando se selecciona se presenta la ventana emergente de la derecha, desde donde se pueden crear diferentes formas de archivos.

Una de las grandes características al trabajar con R y RStudio es la creación de programas o scripts. Un scripts es un archivo que va a contener todo el código, ya sea expresiones, funciones, gráficas, etc., que se deseen utilizar para el análisis de datos que se necesite.

De esta forma, los elementos de esta sección de New File son los siguientes:

- i. **New File->R Script:** Crea una nueva ventana en la interfaz gráfica de RStudio, denominada la ventana de **Edición**, el cual es el entorno de trabajo o el **Editor de R**, donde se pueden editar, manipular, borrar , etc. Todos los datos o expresiones que contengan. Hasta ahora se ha tenido como entorno de trabajo la consola, sin embargo, no es la forma más eficiente de trabajar en RStudio, por lo que es conveniente realizar las operaciones a través de un editor donde se almacenan todos los comandos necesarios para el análisis de datos. Cuando se tenga listo un scripts para ejecutar una instrucción se señala y se pulsa (**Ctrl+Intro**); (**Ctrl+r**) o dando clic en el icono “**Run**”, que se sitúa en la parte superior derecha de la ventana de edición.
- ii. **New File->R Notebook:** Crea un Nuevo archive tipo block de notas. Para utilizar esta función RStudio indica que se debe de descargar un paquete.

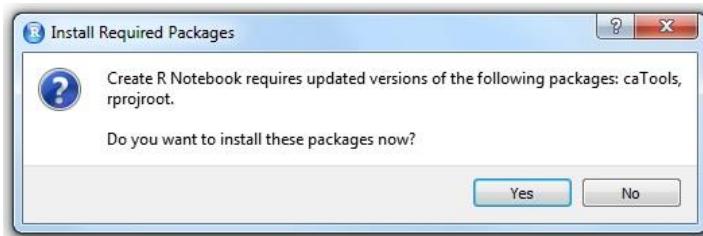


Fig. 49. Pantalla de requerimiento de instalación de paquete para R Notebook

- iii. **New File->R Markdown:** Permite crear archivos de tipo RMarkdown. R Markdown es un formato que permite la creación de documentos, presentaciones dinámicas e informes de R. Markdown es un formato de sintaxis simple en el que también se pueden crear documentos en HTML, PDF, y Word.
- iv. **New File->Text File:** Al igual que el comando anterior, permite crear una ventana de texto en lugar de un script. La diferencia de los archivos tipo texto es que no se pueden ejecutar ninguna expresión o función como en los scripts, a menos de que se copien y se peguen en el espacio de trabajo.
- v. **New File->C++ File:** Permite compilar funciones de C++ en R.
- vi. **New File->R Sweave:** Permite crear un archivo que trabaja con documentos de tipo científico en LaTe
- vii. **New File->R HTML:** Permite la creación de informes web en formato HTML
- viii. **New File->R Presentation.** Permite realizar presentaciones sencillas tipo PowerPoint. Las presentaciones se componen de diapositivas que utilizan código R y ecuaciones LaTeX de manera simple y básica.
- ix. **New File->R Documentation.** Permite crear una documentación básica sobre un determinado tema referente a funciones o conjunto de datos.

✓ Sección Edit

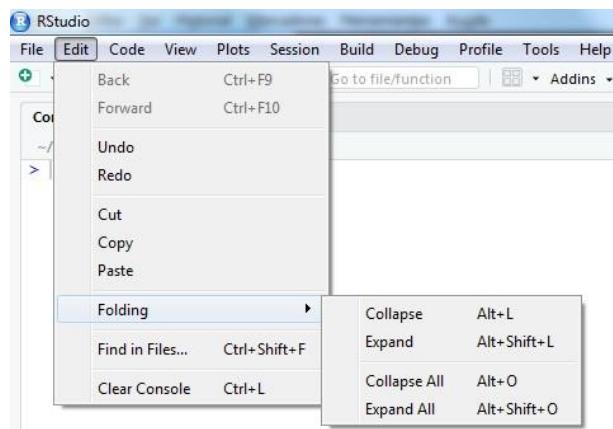


Fig. 50. Pantalla del menú Edit

- i. **Back/Forward:** Atrás/Adelante va una instrucción hacia atrás o adelante en la ventana del editor de comandos.
- ii. **Undo/Redo:** Deshace/Rehace la última acción realizada/rechazada en la ventana del editor de comandos.
- iii. **Cut/Copy/Paste:** Corta/ copia/pega las expresiones de la ventana del editor de comandos.
- iv. **Folding:** Permite mostrar y ocultar fácilmente los bloques de código para que sea más fácil navegar por el archivo del código fuente.
- v. **Folding/Collapse:** Permite duplicar una selección arbitraria de código. Las regiones plegadas se conservan durante la edición de un documento, sin embargo cuando un archivo se cierra y se vuelve a abrir todas las regiones plegables son por defecto regiones no plegables.
- vi. **Go to line...:** Permite ir rápidamente a una línea concreta del texto que se esté utilizando en la ventana del editor de comandos.
- vii. **Find....:** Permite buscar alguna palabra o conjunto de palabras del texto que se esté utilizando en la ventana del editor de comandos.
- viii. **Find Next/Find Previous:** Encuentra el siguiente/anterior conjunto de letras idéntico al buscado anteriormente en la misma ventana del editor de comandos.
- ix. **Replace and Find:** Busca alguna palabra o conjunto de palabras del texto que se esté utilizando en la ventana del editor de comandos. También reemplaza el conjunto de texto buscado por otro que se elija.
- x. **Find in Files...:** Permite buscar de forma recursiva todos los archivos para cada ocurrencia de una cadena dada en un directorio específico.
- xi. **Clear Console:** Limpia la consola, sin borrar los objetos que se hayan almacenado previamente en la memoria.

✓ Sección Code

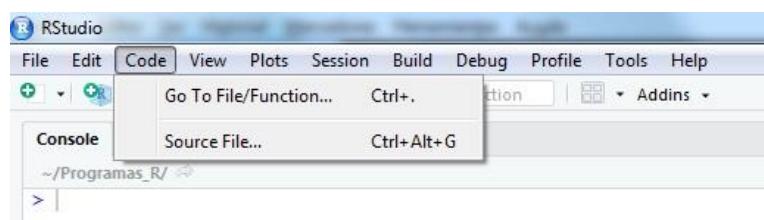


Fig. 51. Pantalla del menú Code

- i. **Go To File Function:** Permite acceder rápidamente a cualquier archivo o función creada con RStudio.
- ii. **Source File:** Almacena en la memoria los objetos definidos de cualquier archivo creado con RStudio aunque no estén abiertos.

✓ Sección View

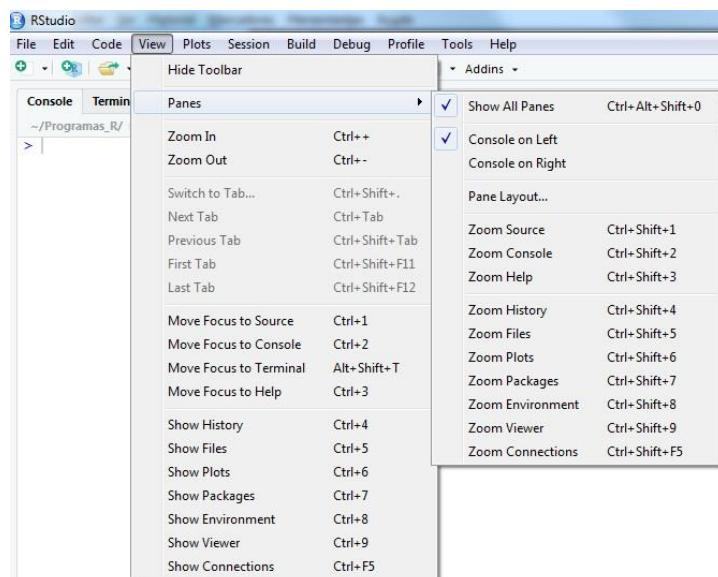


Fig. 52. Pantalla del menú View

- i. **Hide/Show Toolbar:** Muestra/Oculta la barra de herramientas.
- ii. **Zoom In/Zoom Out:** Realiza un zoom sobre cada una de las ventanas aumentando/disminuyendo el tamaño de su contenido.
- iii. **Switch To Tab:** Permite cambiar de pestaña para visualizar cualquier hoja de edición.
- iv. **Next Tab/Previous Tab:** Permite cambiar a la pestaña siguiente/anterior para visualizar la hoja de edición siguiente/anterior.
- v. **First Tab/Last Tab:** Permite cambiar a la primera/última pestaña para visualizar la primera/última hoja de edición.
- vi. **Move Focus To Source/Move Focus To Console:** Mueve el cursor a la ventana de edición/consola de trabajo desde cualquier otra ventana.
- vii. **Move Focus To Help:** Mueve el cursor a la pestaña de ayuda que por defecto se muestra en la ventana auxiliar número 2.
- viii. **Show History:** Muestra todo el código que se ha ejecutado en la consola desde la última vez que se eliminó el historial. Por defecto, el historial se muestra en la ventana auxiliar número 1.
- ix. **Show Files/Show Plots/ Show Packages:** Muestra el conjunto de archivos existentes en el directorio de trabajo/conjunto de gráficos que se han generado/conjunto de paquetes que el programa tiene instalados hasta el momento. Por defecto, se muestra en la ventana auxiliar número 2.
- x. **Show Environment:** Muestra el conjunto de objetos que se han guardado en la memoria del programa. Por defecto se muestra en la ventana auxiliar número 1.
- xi. **Show Viewer:** Muestra la zona viewer del escritorio de trabajo.
- xii. **Show Connections:** Muestra la zona de conexiones de la ventana de medio ambiente.

✓ Sección Plots

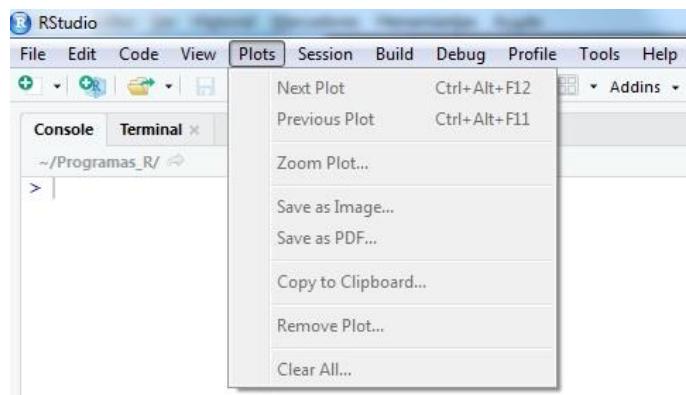


Fig. 53. Pantalla del menú Plot

- i. **Next Plot/ Previous Plot:** Muestra el gráfico siguiente/anterior.
- ii. **Zoom Plot...:** Abre una nueva ventana en la que se muestra el gráfico seleccionado.
- iii. **Save Plot as Image.../Save Plot as PDF...:** Guarda el gráfico seleccionado como una imagen (.png, .jpg, .tiff, .bmp, .metafile, .svg, .eps)/o en formato PDF
- iv. **Copy Plot to Clipboard:** Copia el gráfico en un portapapeles.
- v. **Remove Plot...:** Elimina el gráfico seleccionado
- vi. **Clear All...:** Elimina todos los gráficos creados.

✓ Sección Session

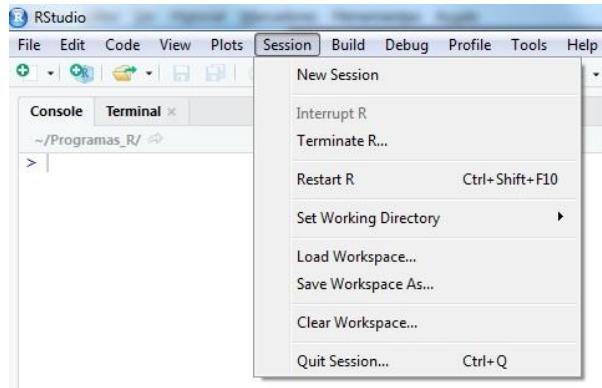


Fig. 54. Pantalla del menú Session

- i. **New Session:** Permite iniciar una nueva sesión de trabajo.
- ii. **Interrupt R:** Permite interrumpir algún proceso interno que no queremos que finalice.
- iii. **Restart R:** Permite actualizar la sesión en la que estemos trabajando.
- iv. **Terminate R:** Permite eliminar toda la información creada en una sesión, pero sin eliminar lo escrito en la ventana de edición.
- v. **Set Working Directory:** Permite configurar el directorio de trabajo.
- vi. **Load Workspace/Save Workspace As/Clear Workspace:** Permite cargar/guardar/eliminar un determinado espacio de trabajo. Por defecto los objetos contenidos en un espacio de trabajo se visualizan en la ventana auxiliar número 1.

✓ Sección Build

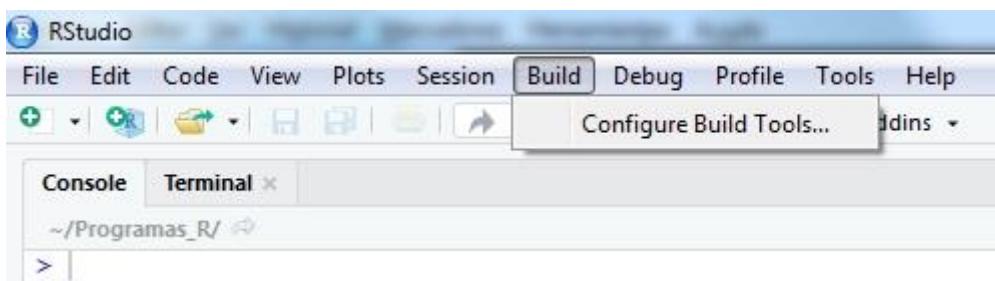


Fig. 55. Pantalla del menú Build

Configure Build Tools...: Permite la construcción de paquetes y herramientas dentro de un proyecto creado por el usuario.

✓ Sesión Debug

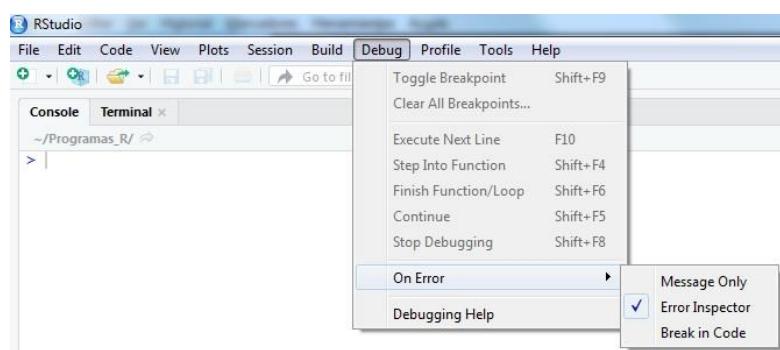


Fig. 56. Pantalla del menú Debug

- i. **Toggle Breakpoint:** Permite introducir un punto de interrupción en el texto de la ventana de edición con la finalidad de averiguar si, hasta la línea donde se coloca dicho punto, la ejecución del texto es correcta.
- ii. **Clear All Breakpoints:** Permite eliminar todos los puntos de interrupción que se hayan utilizado hasta el momento.
- iii. **Execute Next Line:** Permite ejecutar texto colocado después de la línea donde se ha introducido un punto de interrupción.
- iv. **Step Into Function:** Permite realizar paso a paso la revisión dentro de una función.
- v. **Continue:** Permite continuar con la ejecución una vez que se ha detenido dicha ejecución en el punto de interrupción.
- vi. **Stop Debugging:** Detiene la depuración.
- vii. **On Error:** En caso de error, permite elegir entre que sólo salga un mensaje de aviso, que se inspeccione el error o que no ejecute más código a partir del error.
- viii. **Debugging Help:** Muestra la página web del programa con la ayuda sobre la depuración de errores.

✓ Sesión Profile

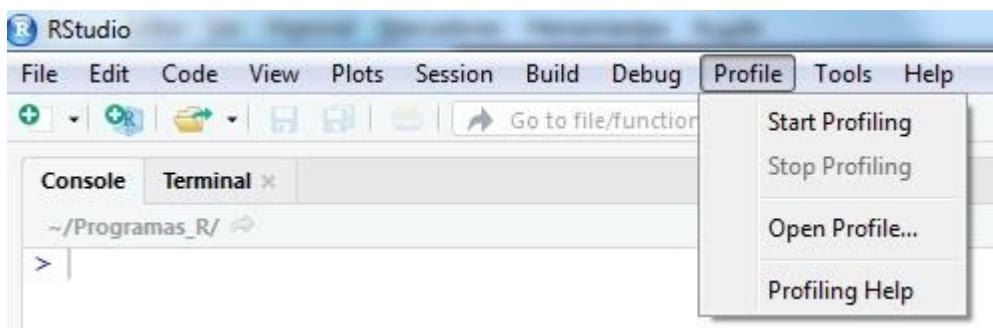


Fig. 57. Pantalla del menú Profile

- i. **Profile:** Permite la creación de perfiles personalizados

✓ Sesión Tools

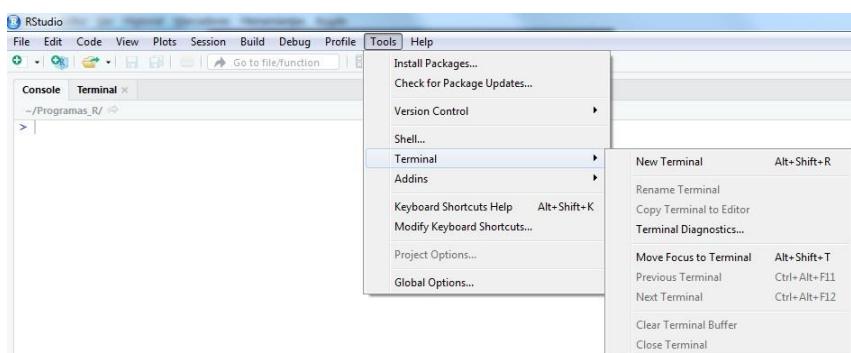


Fig. 58. Pantalla del menú Tools

- i. **Install Packages...:** Permite instalar paquetes. Es importante que dentro de esta opción esté marcada la opción de **Install dependencies**.
- ii. **Check for Packages Updates...:** Permite actualizar los paquetes seleccionados.
- iii. **Version Control:** Permite controlar varios proyectos a la vez, hacer copias de seguridad de los proyectos...
- iv. **Shell...:** Es una interfaz de ventana de comandos para las operaciones más comunes de control de versiones.
- v. **Terminal:** Permite la gestión de terminales (nueva, modificarla, copiar, diagnóstico, limpieza de memoria y cerrar).
- vi. **Addins:** Permite el uso de complementos Addins como calendarios dentro de los proyectos.
- vii. **Keyboard Shortcut Help:** Presenta una pantalla de ayuda que muestra todos los atajos de teclado.
- viii. **Modify Keyboard Shortcut:** Permite realizar una modificación personalizada de los atajos de teclado para las funciones.
- ix. **Project Options:** Muestra las opciones de un Project generado por el usuario.
- x. **Global Options...:** Muestra y configura las opciones generales de RStudio

✓ Sesión Help

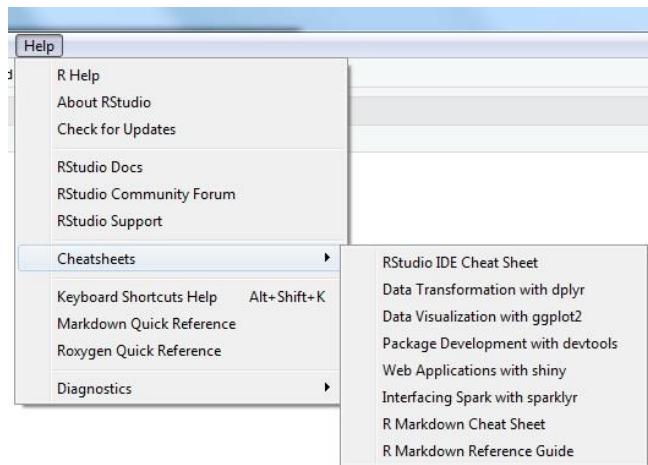


Fig. 59. Pantalla del menú Help

- i. **R Help:** Muestra la ayuda del programa. Por defecto, se puede visualizar en la ventana auxiliar número 2.
- ii. **About RStudio:** Muestra información sobre RStudio.
- iii. **Check For Updates:** Permite realizar una búsqueda en la última versión con la finalidad de obtener la última actualización de RStudio.
- iv. **RStudio Docs:** Muestra la página web de RStudio, en la que se explica la documentación con la que se puede trabajar en RStudio.
- v. **RStudio Support:** Muestra la página web RStudio, en la que hay un soporte de ayuda para cualquier duda sobre RStudio.
- vi. **Keyboard Shortcuts:** Muestra la página web de RStudio, en la que se puede consultar todos los posibles métodos abreviados de teclado (combinaciones de teclas) para ejecutar las expresiones y comandos en RStudio.
- vii. **Diagnostics:** Permite realizar algunas opciones sobre diagnósticos de RStudio.

NOTA: Es importante resaltar que cuando se está escribiendo una expresión en RStudio, el programa va desplegando una sintaxis de ayuda que permite que el usuario visualice mucho mejor tanto la sintaxis como lo que deseé realizar por medio de dicha función.

Ejemplo: Si se está escribiendo la expresión “mean”, al estarla tecleando, el programa despliega la ayuda similar a la siguiente.

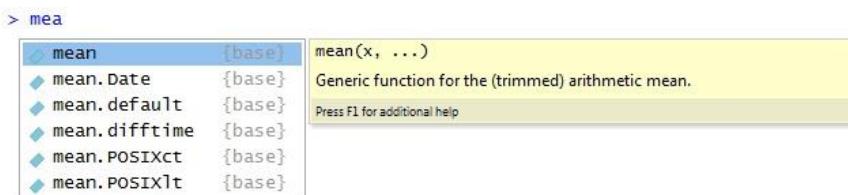


Fig. 60. Pantalla de ayuda de escritura de expresiones de RStudio

En la pantalla anterior, se observa como RStudio despliega los tópicos similares relacionados con la expresión que se quiere escribir. En la parte derecha de cada uno presenta la notación para escribirla. Si se desea más información puede pulsarse la tecla de ayuda F1

5) Instalación de RCommander (RCmdr)

R Commander, también conocido como RCmdr, es una interfaz gráfica de usuario o GUI en inglés para el lenguaje R, de libre distribución bajo licencia GNU GPL. RCmdr fue desarrollado por John Fox del Departamento de Sociología de la Universidad McMaster. Permite acceder a muchas de las capacidades estadísticas siendo la alternativa más viable basada en R a los paquetes estadísticos como SPSS.

- Introducción e Instalación básica de RCmdr

RCmdr permite realizar gran parte de los análisis estadísticos a través de menús, además de la mayoría de las labores de R sin necesidad de utilizar complejas órdenes de programación que se encuentran en la mayoría de los lenguajes. Aun así, RCmdr, escribe el código de las expresiones en una ventana de instrucciones de la misma manera que R. Debido a que funciona bajo R por lo que familiarizarse con el paquete es bastante sencillo.

Para utilizar RCmdr, hay que instalar el paquete “Rcmdr”, el cual no viene instalado por defecto en la base de R, por lo que se debe de descargar desde R, pero la instalación es muy rápida y sencilla.

Existen dos formas de realizar esta operación. En cualquiera de las dos opciones se presentará una ventana indicando que se seleccione el servidor “CRAN” que se desee para realizar la instalación en la opción a por función en la consola o el paquete que se desea descargar buscando el “RCmdr” en la opción b por medio del menú.

- a) A través de función: Por medio de la consola de R, a través de una instrucción mediante la función de instalar paquetes, vista al inicio de este curso.

```
> install.packages("Rcmdr", dependencies = T ))
```

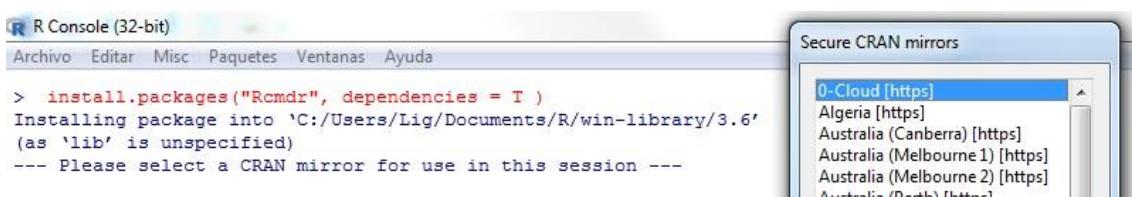
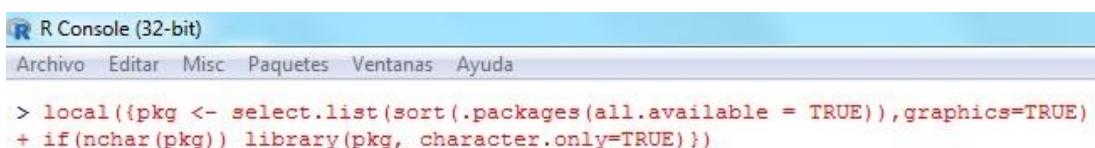


Fig. 61. Pantalla de descarga del paquete “RCmdr” desde la consola

Como se observa en la imagen, se indica el sitio para la instalación

- b) Por medio de menú: Se selecciona el menú “Paquetes/Cargar paquete...” de la barra de herramientas de menú de R.



```
> local(pkg <- select.list(sort(.packages(all.available = TRUE)), graphics=TRUE)
+ if(nchar(pkg)) library(pkg, character.only=TRUE))
```

Fig. 62. Pantalla de consola al indicar “Paquetes/Cargar paquete..”

- A continuación, se abre la ventana de la lista de paquetes, para que se seleccione uno y se busca el paquete, en este caso “Rcmdr”.

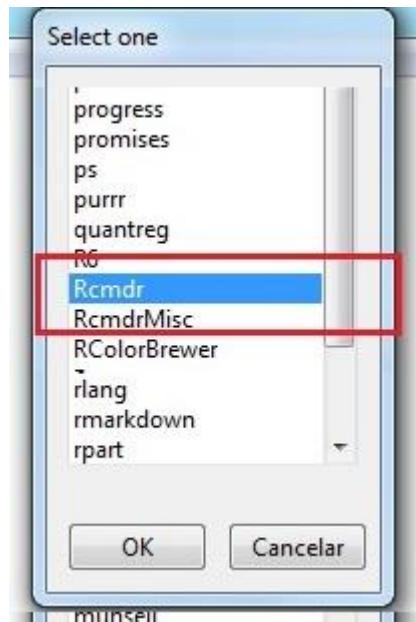


Fig. 63. Pantalla de selección del paquete a descargar de la lista de paquetes de R

- Una vez seleccionado el paquete y pulsado el botón “OK”, el paquete se instala automáticamente. En la consola se presenta la información general de la instalación

```
R Console (32-bit)
Archivo Editar Misc Paquetes Ventanas Ayuda

> local(pkg <- select.list(sort(.packages(all.available = TRUE)),graphics=TRUE)
+ if(nchar(pkg)) library(pkg, character.only=TRUE))
Loading required package: splines
Loading required package: RcmdrMisc
Loading required package: car
Loading required package: carData
Loading required package: sandwich
Loading required package: effects
Registered S3 methods overwritten by 'lme4':
  method           from
  cooks.distance.influence.merMod car
  influence.merMod      car
  dfbeta.influence.merMod    car
  dfbetas.influence.merMod   car
lattice theme set by effectsTheme()
See ?effectsTheme for details.

Versión del Rcmdr 2.6-1

Attaching package: 'Rcmdr'

The following object is masked from 'package:base':
  errorCondition

> |
```

Fig. 64. Pantalla con la información de la instalación de RCdr en la consola de R

- Interfaz gráfica de RCmdr

Una vez descargado e instalado el paquete, se presenta la pantalla de la interfaz de RCmdr.

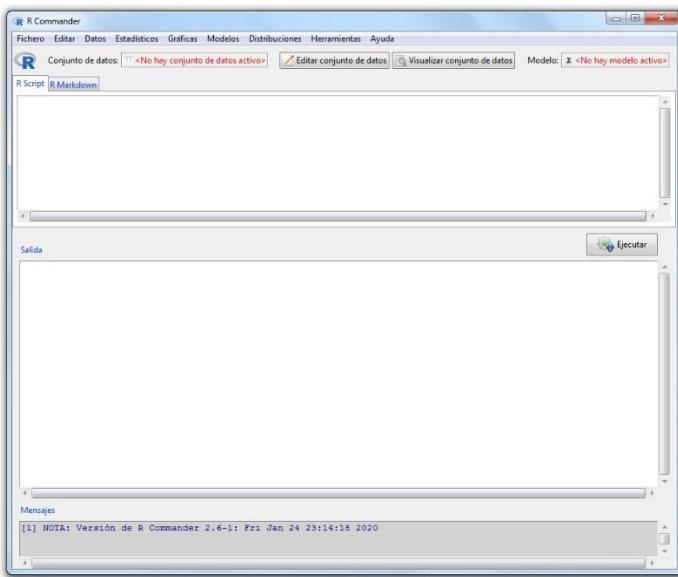


Fig. 65. Pantalla principal de la interfaz de RCmdr

NOTA: En caso de que en una sesión se cierre esta interfaz, solo hay que teclear en la consola de R la instrucción.” > [Commander\(\)](#)”

De esta forma, se abrirá de nuevo la interfaz de RCmdr. Hay que indicar que RCmdr funciona solo a través de R, si se intenta acceder desde RStudio no se puede. Por último, si existe algún problema con la carga y ejecución de RCmdr, basta con utilizar la función “library()” con “Comander” como argumento de la siguiente manera “> [library\(Rcmdr\)](#) ”. Así R, procede a cargar RCmdr en el sistema.

- Partes de la ventana y menús de RCmdr

En la interfaz de RCmdr se pueden observar cuatro secciones delimitadas

1. El menú de ventanas desplegables, con las opciones: Fichero, Editar, Datos, Estadísticos, Gráficas, Modelos, Distribuciones, Herramientas y Ayuda.
2. La ventana R Script/ R Markdown. Es la ventana de instrucciones donde se mostrarán las órdenes que se están ejecutando con RCmdr. Por cada acción ejecutada del menú, RCmdr traducirá dicha acción a código R y lo presentará en esta ventana. Hay que recordar que previamente se ha mencionado que R Markdown es un formato que permite la creación de documentos, presentaciones dinámicas e informes de R. Markdown es un formato de sintaxis simple en el que también se pueden crear documentos en HTML, PDF, y Word.
3. La ventana de Salida. En esta ventana se visualizan las expresiones y los resultados. También, se pueden introducir manualmente las órdenes del lenguaje R, para ello se selecciona dicha orden, dejando el cursor sobre ella, y se pulsa el botón Ejecutar. La salida de la orden se mostrará en la ventana de Salida.
4. La ventana de Mensajes. En esta ventana RCmdr presenta diversos mensajes, como errores o información al usuario sobre acciones. Por ejemplo inicialmente se presenta como se observa, el mensaje de la versión actual de RCmdr.

NOTA: No es el propósito describir cada una de las secciones de RCmdr, por ser muy extensas para el objetivo de esta unidad. De tal forma, que a medida que se utilice RCmdr, se irán definiendo los elementos que sean necesarios.

6) Primeros pasos con datos en R, RStudio y RCmdr

• Conjuntos de datos

En R, el concepto de dato es mucho más amplio que el que se maneja en estadística, puesto que en R lo que se denomina “dato” es el resultado de ejecutar una expresión R, y hay que recordar que las expresiones en R al ser un lenguaje orientado a objetos, son un tipo de objetos.

En este caso, un dato puede ser una matriz en donde la primera columna va a presentar los nombres de individuos en los que se han observado las variables cuyos valores aparecen en el resto de columnas.

Cada tipo de dato tiene asociado determinados atributos; siendo el más importante su “modo”. De esta forma se deben de considerar cuatro clases de modos.

- a) Lógico (logical): Es un modo binario en donde los posibles valores son T (True) o verdadero y F (False) o Falso.
- b) Numérico (Numeric): Es el modo en donde los valores posibles representan números reales.
- c) Complejo (Complex): Determina el modo en donde los valores posibles son números complejos.
- d) Carácter (Character): Es el modo en donde los valores posibles son caracteres separados por comillas.

Asimismo, existen cinco diferentes tipos de datos a considerar que son:

- a) Vector (vector): Es el conjunto de elementos en un orden específico. Todos los elementos de un vector deben de ser del mismo modo. Normalmente, los más utilizados son los vectores numéricos, es decir, vectores en el que los números son los elementos.
- b) Matriz (Matrix): Es la disposición bidimensional de elementos de un mismo modo.
- c) Factor (Factor): Es un vector cuyos elementos son valores procedentes de un número finito de categorías.
- d) Estructura de Datos (Data frame): Es la disposición bidimensional de elementos cuyas columnas pueden estar formadas por elementos de distinto modo.
- e) Lista (List): Es una expresión mucho más generalizada de dato, la cual puede contener colecciones arbitrarias de datos.

NOTA: Con cada tema, existen una gran cantidad de funciones útiles para manejar los objetos, en este curso solo vamos a mencionar algunos de los más importantes.

- Creación y manejo de variables

Una variable en R es un símbolo que se utiliza para referirse a un valor. Proporciona un nombre a almacenar que a través de los programas se pueden manipular.

Un nombre válido de una variable consiste en letras, números y los caracteres de punto o subrayado. El nombre de la variable puede comenzar con una letra o con un punto pero no seguido de un número sino de una letra. Ejemplo de variables válidas pueden ser:

(Variable_nombre1), (variable_nombre%), (.variable_nombre)

A las variables se les asignan valores, utilizando como ya hemos visto el operador “< - “, hacia la izquierda o hacia la derecha, y también mediante el operador de igualdad “=“.

Los valores de las variables se pueden imprimir utilizando las funciones “print()” o “cat()”, esta última función combina múltiples elementos en una salida de impresión continua.

```
> print(edad)
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

```
> cat(edad)
18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

De la misma forma, se puede visualizar el contenido de una variable solo con teclearla en la consola de R y pulsar “Enter”. Por ejemplo con teclear solo “edad”

```
> edad
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

Por último, se debe de tener cuidado a la hora de asignar nombres a las variables, pues si se repite un nombre se borra el contenido anterior por el nuevo.

- Vectores

El vector es la concatenación o sucesión de datos, es el tipo de dato más empleado en R, que se utiliza generalmente como un argumento de funciones. Hay que recordar que los argumentos de un vector deben de ser del mismo modo. Otro atributo tomar en cuenta del vector es su longitud.

La forma más sencilla y habitual para construir un vector es utilizando la función “c()”, que indica “concatenar”. Por ejemplo, el vector visto anteriormente “edad”.

```
(edad <- c(18, 19, 24, 17, 20, 17, 21, 23, 19, 25, 22, 16, 19, 17, 21, 29, 22, 18, 21, 20, 21, 23, 22, 19))
```

En donde el vector “edad” está formado por 24 números y es de modo numérico.

Para conocer tanto el modo como la longitud de un vector se utilizan las funciones “mode()” y “length()”. También se puede utilizar la función “**class()**” para determinar la clase del vector

```
> mode(edad)           > class(edad)
[1] "numeric"          [1] "numeric"

> length(edad)
[1] 24
```

En caso de que los elementos de un vector sean del modo “carácter”, se deben escribir los elementos no numéricos entre comillas (Las comillas se encuentran sobre el número 2 en el tecleado). Por ejemplo

```
> nombre.docentes <- c("María Molina", "Juan Pérez", "Pedro Maldonado", "Asunción Ramírez", "Elena Ruiz", "José González", "María José Zambrana", "Juan Calderón", "Sebastián Olmedo", "Clementina Rodríguez", "Clara Molina", "Fausto Flores", "Martín Espinoza", "Roberto Duarte", "Carmen Gómez", "Julian Soto", "Domitila Navas", "Eliud Zúñiga", "Imelda Ortiz", "Mireya Fonseca", "Francisco Heredia", "Tomas Castro", "Jose Maria González", "María Concepción Montero")
```

Para comprobar el modo y la longitud de este vector llamado “nombre.docente”, utilizamos las funciones antes mencionadas

```
> mode(nombre.docentes)
[1] "character"
```

```
> length(nombre.docentes)
[1] 24
```

Para acceder a los elementos de un vector se utilizan los corchetes. Por ejemplo

Del vector “edad”

```
> edad
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23 22 19
```

```
> edad[3]           # El tercer elemento del vector
[1] 24
```

```
> edad[3:5]         # Del tercero al quinto elemento del vector
[1] 24 17 20
```

También se pueden realizar operaciones con los elementos individuales de un vector, por ejemplo:

```
> sumaV = (edad[3] + edad[6])      # suma el tercer y el sexto elemento del vector edad
> sumaV                          # Se presenta el contenido de sumaV
[1] 41
```

Para obtener varios datos no secuenciales

```
> edad[c(2,8,12)]      # se colocan los elementos a acceder entre paréntesis
[1] 19 23 16
```

```
> nombre.docentes[c(2,8,12)]
[1] "Juan Pérez"  "Juan Calderón" "Fausto Flores"
```

Para conocer el número de elementos de un vector por su valor se utiliza la función “table()”, es muy útil a la hora de examinar vectores categóricos.

```
> table(edad)          # suma cuantas veces se repite cada valor del vector
edad
16 17 18 19 20 21 22 23 24 25 29
1 3 2 4 2 4 3 2 1 1 1
```

Ahora bien, se han construido dos vectores con el mismo número de datos, en el primero se encuentran almacenadas las edades y el segundo los nombres de docentes. De esos elementos se desean conocer algunos como por ejemplo los docentes que tienen 21 años. Esto se realiza de la siguiente manera.

```
> nombre.docentes[edad==21]
[1] "María José Zambrana" "Carmen Gómez"     "Imelda Ortiz"     "Francisco Heredia"
```

Se coloca primero el vector que contiene el dato a conocer, esto es el nombre del docente y entre corchetes el vector con el argumento a preguntar, es decir, que tenga o sea igual a 21, para ello se utiliza el operador comparativo “igual a ==”. R responde presentando el resultado que son cuatro docentes.

- Unión de los elementos de dos vectores: Para unir los elementos de dos vectores se crea un nuevo vector que va a concatenar los dos vectores individuales. Por ejemplo:

```
> Edades_2 = c( edad, nombre.docentes)      # Vector concatenado simple
```

El resultado es el siguiente

> Edades_2

[1] "18"	"19"	"24"	"17"
[5] "20"	"17"	"21"	"23"
[9] "19"	"25"	"22"	"16"
[13] "19"	"17"	"21"	"29"
[17] "22"	"18"	"21"	"20"
[21] "21"	"23"	"22"	"19"
[25] "María Molina"	"Juan Pérez"	"Pedro Maldonado"	"Asunción Ramírez"
[29] "Elena Ruiz"	"José González"	"María José Zambrana"	"Juan Calderón"
[33] "Sebastián Olmedo"	"Clementina Rodríguez"	"Clara Molina"	"Fausto Flores"
[37] "Martín Espinoza"	"Roberto Duarte"	"Carmen Gómez"	"Julián Soto"
[41] "Domitila Navas"	"Eliud Zúñiga"	"Imelda Ortiz"	"Mireya Fonseca"
[45] "Francisco Heredia"	"Tomas Castro"	"Jose Maria González"	"María Concepción Montero"

En este caso, primero se colocan los elementos del primer vector y seguidamente los del segundo. Sin embargo, si lo que se desea es unir en uno solo los elementos de ambos vectores, es decir unir la edad con el nombre, se debe de utilizar la función “paste()”, “pegar” y se teclea lo siguiente.

Edades_1 <- paste(edad, nombre.docentes, sep=" ") # vector concatenado mediante paste()

La función “paste()”, une todos los vectores de caracteres que se le suministran y construye una sola cadena de caracteres. También admite argumentos numéricos, que convierte inmediatamente en cadenas de caracteres. En su forma predeterminada, en la cadena final, cada argumento original se separa del siguiente por un espacio en blanco, aunque ello puede cambiarse utilizando el argumento (sep="cadena"), que sustituye el espacio en blanco por cadena, la cual podría ser incluso vacía, en el ejemplo se ha separado las comillas para dar más espacio. El resultado es el siguiente

> Edades_1

[1] "18 María Molina"	"19 Juan Pérez"	"24 Pedro Maldonado"
[4] "17 Asunción Ramírez"	"20 Elena Ruiz"	"17 José González"
[7] "21 María José Zambrana"	"23 Juan Calderón"	"19 Sebastián Olmedo"
[10] "25 Clementina Rodríguez"	"22 Clara Molina"	"16 Fausto Flores"
[13] "19 Martín Espinoza"	"17 Roberto Duarte"	"21 Carmen Gómez"
[16] "29 Julián Soto"	"22 Domitila Navas"	"18 Eliud Zúñiga"
[19] "21 Imelda Ortiz"	"20 Mireya Fonseca"	"21 Francisco Heredia"
[22] "23 Tomas Castro"	"22 Jose Maria González"	"19 María Concepción Montero"

Como se observa, el resultado es diferente en ambos vectores, la longitud es diferente (con las funciones “modey length” se observa mejor). Pero hay que puntualizar que al concatenar, los vectores numéricos se convierten en vectores alfanuméricos, el modo es igual en el resultado.

> mode(Edades_1) # Vector de datos de modo carácter
[1] "character"

> length(Edades_1) # Vector de datos de tamaño 24
[1] 24

> length(Edades_2) # Vector de datos de tamaño 48
[1] 48

Esto se ha realizado con dos vectores diferentes concatenándolos, sin embargo, si lo que se desea es asociar a las edades los nombres de los docentes, se utiliza la función “names()”, de la forma siguiente.

```
> names(edad) <- c("María Molina", "Juan Pérez", "Pedro Maldonado", "Asunción Ramírez", "Elena Ruiz", "José González", "María José Zambrana", "Juan Calderón", "Sebastián Olmedo", "Clementina Rodríguez", "Clara Molina", "Fausto Flores", "Martín Espinoza", "Roberto Duarte", "Carmen Gómez", "Julián Soto", "Domitila Navas", "Eliud Zúñiga", "Imelda Ortiz", "Mireya Fonseca", "Francisco Heredia", "Tomas Castro", "José María González", "María Concepción Montero")
```

Si ahora se muestra el contenido del vector “edad” se presenta lo siguiente.

```
> edad
   María Molina      Juan Pérez      Pedro Maldonado      Asunción Ramírez
   18                  19                  24                  17
   Elena Ruiz        José González     María José Zambrana    Juan Calderón
   20                  17                  21                  23
   Sebastián Olmedo Clementina Rodríguez    Clara Molina     Fausto Flores
   19                  25                  22                  16
   Martín Espinoza    Roberto Duarte     Carmen Gómez     Julián Soto
   19                  17                  21                  29
   Domitila Navas     Eliud Zúñiga       Imelda Ortiz     Mireya Fonseca
   22                  18                  21                  20
   Francisco Heredia  Tomas Castro      José María González María Concepción Montero
   21                  23                  22                  19
```

El vector ha cambiado, a cada edad se le ha asignado el nombre de un docente.

Ahora, se puede asignar a un nuevo vector este resultado de la forma siguiente:

```
> Nuevos.nombres <- edad      # El vector Nuevos.nombres, contendrá los datos asociados
```

También, se puede realizar de manera directa, asignando en la creación del vector, el valor que se dese por ejemplo

```
> Nombre3 <- c("María Molina" = 18, "Juan Pérez" = 19, "Pedro Maldonado" = 24, "Asunción Ramírez" = 17)
```

Lo que da como resultado:

```
> Nombre3
  María Molina   Juan Pérez  Pedro Maldonado  Asunción Ramírez
  18              19          24            17
```

Los operadores lógicos también pueden ser utilizados con los vectores. Por ejemplo para saber cuáles de los elementos, del vector “edad” son menores o iguales a 18 años, simplemente se teclea lo siguiente.

```
> edad <= 18
  María Molina      Juan Pérez      Pedro Maldonado      Asunción Ramírez
  TRUE             FALSE           FALSE           TRUE
  Elena Ruiz        José González     María José Zambrana    Juan Calderón
  FALSE            TRUE            FALSE           FALSE
  Sebastián Olmedo Clementina Rodríguez    Clara Molina     Fausto Flores
  FALSE            FALSE           FALSE           TRUE
  Martín Espinoza    Roberto Duarte     Carmen Gómez     Julián Soto
  FALSE            TRUE            FALSE           FALSE
  Domitila Navas     Eliud Zúñiga       Imelda Ortiz     Mireya Fonseca
  FALSE            TRUE            FALSE           FALSE
  Francisco Heredia  Tomas Castro      José María González María Concepción Montero
  FALSE            FALSE           FALSE           FALSE
```

Los valores verdaderos (TRUE), corresponden a cinco docentes.

Y si lo que se desea saber, es cuáles de los valores del vector tienen una determinada característica. Por ejemplo, ¿Quiénes son los docentes que tienen 18 años exactamente? Para esto se utiliza la función “which()” de la forma siguiente.

```
> which(edad==18)          # Se utiliza el operador lógico “==”
María Molina    Eliud Zúñiga
      1           18
```

Son: María Molina que es el primer valor del vector y Eliud Zúñiga el valor número 18. Recordemos que con la función “table()”, se sabía que eran 2. Esta misma función indicaba que existen tres docentes con 22 años, para saber quiénes son y en qué posición del vector se encuentran, volvemos a utilizar “which()”

```
> which(edad==22)
Clara Molina   Domitila Navas   José María González
      11          17            23
```

En términos estadísticos, esto puede ser de mucha utilidad, porque de la misma manera se manipulan grandes volúmenes de información, como se verá más adelante, para obtener determinados resultados.

Otra forma de obtener el mismo resultado es a través de lo siguiente:

```
> nombre.docentes[edad==22]
[1] "Clara Molina"       "Domitila Navas"       "José María González"
```

Aquí se le pregunta al vector “nombre.docente”, quienes de ellos tienen una edad igual (mediante el operador de igualdad ==) de 22 años.

Ahora bien, para seleccionar muestras de un vector se utiliza la función “sample()”, por ejemplo

```
> sample(edad,5)
José María González   Juan Calderón   Sebastián Olmedo   María Molina   Francisco Heredia
      22             23              19                18              21
```

La función “sample()” ha seleccionado 5 muestras del vector. Cada vez que se ejecute seleccionará diferentes muestras.

En ocasiones particulares, se necesita crear vectores a partir de secuencias, para crear un vector con una secuencia simple solo hay que definir el inicio y final mediante el uso de dos puntos. Por ejemplo

```
> X <- c(3:10)
```

Lo que crea un vector “X” con una secuencia desde 3 hasta 10. Al llamar al vector en la consola se obtendrá el contenido del vector.

```
> X
[1] 3 4 5 6 7 8 9 10
```

Pero si lo que se desea es un vector de una longitud determinada, se utiliza la función “seq()”.

```
Vector.Secuencia <- seq(from = 1, by = 3, length.out = 24)
```

En este caso se crea un vector llamado “Vector.Secuencia” con la función seq(), indicando que se va a tener una secuencia de 24 números, iniciar desde el valor 1 con un aumento de tres en tres. Al llamar al vector, se observa su contenido.

Vector.Secuencia

```
[1] 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64  
67 70
```

En algunos casos, es necesario repetir el número de secuencias varias veces, para conseguir un vector deseado. Esto se realiza utilizando la función “rep()” de la siguiente manera.

Con el vector “X” anterior se va a repetir tres veces dicha secuencia mediante la instrucción.

```
> w <- rep(x, times = 3)  
> w  
[1] 3 4 5 6 7 8 9 10 3 4 5 6 7 8 9 10 3 4 5 6 7 8 9 10
```

Como se puede observar al mostrar el contenido de “w”, se ha repetido 3 veces la secuencia.

Lectura y escritura de vectores a archivos de datos

Una vez que se tienen vectores de datos creados, lo más probable es que se deseen que sean guardados en archivos ya sea de tipo texto ASCII (txt) o con formato “csv” (comma separated values o datos separados por comas), que es un formato sin delimitadores para tablas de Excel que además puede ser abierto en editores de texto.

Primero es comprobar el directorio de trabajo a través de “getwd()”

```
> getwd()  
[1] "C:/Users/Lig/Documents/Programas_R"
```

Para almacenar los datos vamos a crear una carpeta dentro de Programas_R, llamada “datos”, dentro de datos es donde se almacenarán los archivos de vectores.

Para almacenar los datos de los vectores en un determinado formato se utiliza la función: “write.table()”. Hay que recordar que para saber más de la sintaxis de una función R dispone de la ayuda y documentación, tecleando “?write.table”, o bien la función “help(write.table)”.

En este caso se van a utilizar los siguientes argumentos:

- i. El nombre del vector que se desea almacenar.
- ii. El nombre del archivo en donde se va a guardar
- iii. La ruta en donde lo vamos a almacenar.
- iv. Se incluye la opción: sep = “,” para separar con comillas
- v. La opción: col.names = TRUE, col.names = FALSE, indican si R va a incluir una línea de cabecera con el nombre de la variable.
- vi. La opción: row.names = TRUE, row.names = FALSE, indican si R va a incluir números de fila en el archivo de salida.

A continuación, se creará un archivo de texto ASCII con extensión “txt”, para el vector edad

```
> write.table(edad,file = "./datos/muestra1.txt", sep = ",", col.names = FALSE, row.names=TRUE)
```

Lo que indica la instrucción es que se almacenará el contenido del vector “edad”, dentro del directorio datos (se coloca “.”, para indicar que va dentro del directorio de trabajo), el archivo a

crear se llama “muestra1” con extensión “txt” de formato ASCII, se ha indicado a R, que no coloque nombres en las columnas, pero si incluya el número de filas, así se asegura de que se muestren los datos. Ahora si vemos la carpeta de datos, se encontrará dicho archivo.

Seguidamente, se va a crear un archivo de tipo “csv”, del vector edad, al que llamaremos “muestra1.csv”

```
> write.table(edad,file = "./datos/muestra1.csv", sep = ",", col.names=FALSE, row.names=TRUE)
```

También se hará lo mismo con los vectores: “Nuevos.nombres” y “nombre.docentes”

```
> write.table(Nuevos.nombres,file = "./datos/muestra2.txt", sep = ",", col.names=FALSE, row.names=TRUE)
```

```
> write.table(Nuevos.nombres,file = "./datos/muestra2.csv", sep = ",", col.names=FALSE, row.names=TRUE)
```

```
> > write.table(nombre.docentes,file = "./datos/muestra3.txt", sep = ",", col.names=FALSE, row.names=TRUE)
```

```
> write.table(Nuevos.nombres,file = "./datos/muestra3.csv", sep = ",", col.names=FALSE, row.names=TRUE)
```

De esta forma, los archivos han sido creados y almacenados en la carpeta datos.

Si abrimos los anteriores archivos, podrán observarse con datos concatenados y asociados. Si se desea guardar un archivo con solo los datos numéricos de la edad

```
> write.table(edad,file = "./datos/muestra4.txt", col.names=FALSE, row.names=FALSE)
```

Los nombres de columnas y filas se colocan en FALSE. Recordad que el vector “edad” es numérico.

Ahora bien, para poder leer datos desde una carpeta, se utiliza la función “read.table()”

Por ejemplo, para importar el archivo “muestra1.txt”

```
muestra1 <- read.table(file = "./datos/muestra1.txt", header = FALSE, sep = ",")
```

La opción header = TRUE hace referencia a las cabeceras de los archivos.

El resultado de este procedimiento es el siguiente en RStudio

En la figura siguiente, se observa en la ventana de medio ambiente en “Data”, el objeto “muestra1”, que corresponde al conjunto de datos

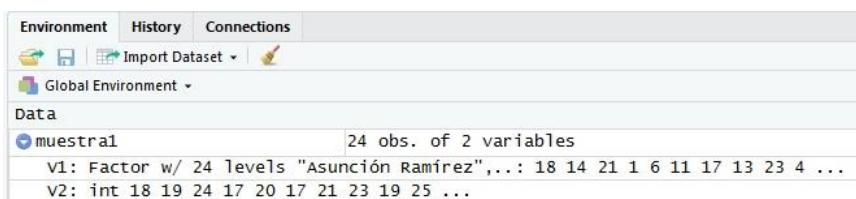


Fig. 66. Pantalla de medio ambiente con el archivo “muestra1” en Data

Al dar clic en Data al archivo, se muestra en la ventana de “archivo”, situada a la izquierda en RStudio, que es la que presenta como se ha mencionado todo lo referente a los archivos. Así la imagen es la siguiente.

	V1	V2
1	Maria Molina	18
2	Juan Pérez	19
3	Pedro Maldonado	24
4	Asunción Ramírez	17
5	Elena Ruiz	20
6	José González	17
7	Maria José Zambrana	21
8	Juan Calderón	23
9	Sebastián Olmedo	19
10	Clementina Rodríguez	25
11	Clara Molina	22
12	Fausto Flores	16
13	Martín Espinoza	19
14	Roberto Duarte	17
15	Carmen Gómez	21
16	Julián Soto	29
17	Domitila Navas	22
18	Eliud Zúñiga	18
19	Imelda Ortiz	21
20	Mireya Fonseca	20
21	Francisco Heredia	21
22	Tomas Castro	23
23	José María González	22
24	María Concepción Montero	19

Fig. 67. Pantalla del archivo muestra1.txt

Los datos de este archivo han sido cargados en el programa y se encuentran listos para ser manipulados y analizados. De forma análoga, se pueden cargar y leer todos los archivos, ya sean creados o que se tengan disponibles, para que sus datos sean analizados.

En este sentido, es importante conocer que R-base ya viene con varios datos precargados que se pueden utilizar para manipular y presentar resultados como práctica y uso para propios análisis. Estos datos vienen en el paquete denominado “datasets”. De la misma forma, muchos paquetes de R también contienen conjuntos de datos. Se puede ver una lista de los datos que R tiene disponible al teclear en la consola la función “data()”. Esto funciona tanto en R como en RStudio. En ese grupo de datos por ejemplo se encuentra el conjunto flor Iris de Edgar Anderson (iris).

Si se quiere ver un conjunto de dichos datos, por ejemplo, los datos almacenados en “women”, solo se teclea en la consola (“> women” y R presenta en la consola dichos datos). Durante este curso, en algunos ejemplos de ejercicios se utilizarán algunos conjuntos de datos ya pregrabados en R-base.

○ Factores

Un factor es un vector de datos no numéricos que está formado por datos procedentes de categorías; Dicho de otra forma, el factor es una estructura de datos que maneja variables categóricas, entendiéndose por variable categórica un dato que toma una cantidad finita de valores, o sea, que puede ser parte de una o varias categorías. Por ejemplo, datos sobre rango de edades, datos obtenidos al anotar si la persona es hombre o mujer, datos de tipos de titulaciones escolares, dividir a una población en grupos, etc.

Los vectores que hasta ahora se han descrito, pertenecen al rango de variables continuas, numéricas o cuantitativas, los factores permiten utilizar modelos y gráficas de variables categóricas o variables cualitativas.

Los datos categóricos tienen asociado una descripción, una cadena de caracteres y a su vez, contar con un número limitado de valores posibles. Así, un factor internamente se almacenará como un número, conteniendo lo que se denomina etiquetas asociadas a cada valor nombradas como niveles. Los niveles identifican los elementos del vector sin repetición, determinando datos únicos, que pueden ser muy útil a la hora de analizar datos. De esta forma, los niveles indican los valores numéricos posibles.

Para crear los factores, se utiliza la función “factor()” y también por “ordered()”. El número de niveles de un factor se obtiene a través de “nlevels()”, y el conjunto de niveles se conoce a través de “levels()”.

Los niveles de los factores se guardan en orden alfabético o en el orden especificado en la función factor. A los factores creados por medio de la función “factor()” se le denominan **factores nominales**, son simples factores si no existe posibilidad de confusión. En cambio los factores creados con la función “ordered()”, se denominan factores ordinales, aquí se distingue un orden en los niveles del factor a tomar en cuenta para análisis estadísticos.

Por ejemplo, vamos a recordar que se realizó una concatenación con el vector “edad”, sin embargo, el vector “edad”, sigue siendo un vector de tipo numérico. Esto se puede comprobar utilizando la función “mode()” o la función “class()” .

```
> mode(edad)
[1] "numeric"
```

Vamos a crear un factor con este vector al que llamaremos “Prof”.

```
> Prof <- factor(edad)      # Creación del factor "Prof"
> Prof                      # Impresión del contenido del factor
  María Molina    Juan Pérez    Pedro Maldonado   Asunción Ramírez
  18              19            24             17
  Elena Ruiz     José González  María José Zambrana Juan Calderón
  20              17            21             23
  Sebastián Olmedo Clementina Rodríguez  Clara Molina   Fausto Flores
  19              25            22             16
  Martín Espinoza Roberto Duarte  Carmen Gómez   Julián Soto
  19              17            21             29
  Domitila Navas Eliud Zúñiga    Imelda Ortiz   Mireya Fonseca
  22              18            21             20
  Francisco Heredia Tomas Castro  José María González María Concepción Montero
  21              23            22             19
Levels: 16 17 18 19 20 21 22 23 24 25 29
```

Como se puede observar, el factor contiene la información, pero además presenta los niveles, que en este caso son las edades sin repetición.

El número de niveles de este factor:

```
> nlevels(Prof)
[1] 11
```

Y el conjunto de niveles del factor

```
> levels(Prof)
[1] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "29"
```

Ahora, vamos a realizar lo mismo, con el vector “nombre.docentes”, que se sabe que es de tipo “carácter”

```
> mode(nombre.docentes)
[1] "character"
```

Se va a crear un factor de este vector llamado “Prof2”

```
> Prof2 <- factor(nombre.docentes)      # Creación del factor "Prof2"
> Prof2                                # Impresión del contenido del factor
[1] María Molina      Juan Pérez       Pedro Maldonado     Asunción Ramírez
[5] Elena Ruiz        José González    María José Zambrana Juan Calderón
[9] Sebastián Olmedo Clementina Rodríguez Clara Molina      Fausto Flores
[13] Martín Espinoza   Roberto Duarte   Carmen Gómez      Julián Soto
[17] Domitila Navas   Eliud Zúñiga     Imelda Ortiz     Mireya Fonseca
[21] Francisco Heredia Tomas Castro     Jose Maria González María Concepción Montero
24 Levels: Asunción Ramírez Carmen Gómez Clara Molina Clementina Rodríguez Domitila Navas ... Tomás Castro
```

Presentamos el número de niveles de este factor y el conjunto de niveles.

```
> nlevels(Prof2)
[1] 24
```

Los niveles van a ser 24

```
> levels(Prof2)
[1] "Asunción Ramírez"      "Carmen Gómez"      "Clara Molina"
[4] "Clementina Rodríguez"  "Domitila Navas"    "Elena Ruiz"
[7] "Eliud Zúñiga"         "Fausto Flores"    "Francisco Heredia"
[10] "Imelda Ortiz"        "José González"    "Jose Maria González"
[13] "Juan Calderón"       "Juan Pérez"      "Julián Soto"
[16] "María Concepción Montero" "María José Zambrana" "María Molina"
[19] "Martín Espinoza"      "Mireya Fonseca"  "Pedro Maldonado"
[22] "Roberto Duarte"       "Sebastián Olmedo" "Tomas Castro"
```

El conjunto de niveles es conformado por cada uno de los nombres de los docentes que se encuentran en la información del factor. Más adelante en este curso, se verá con mayor claridad la utilidad de los factores.

- Matrices

Una matriz es una disposición bidimensional o, dicho de otra forma, una matriz es un vector con un atributo adicional, su dimensión (dim), el cual es un vector entero compuesto de dos elementos, el número de renglones o filas y el número de columnas, en donde al igual que con los vectores, todos los elementos deben de ser del mismo tipo.

Para crear una matriz se utiliza la función “matrix()”, utilizando dos argumentos, la función “c()”, que a su vez tendrá como argumentos los datos a introducir, y el número de columna que contiene la matriz.

Un ejemplo sencillo de la creación de una matriz es el siguiente:

```
> Matriz1 <- matrix(c(48, 23, 12, 6, 18, 25, 31, 18), ncol=4)
> Matriz1
 [,1] [,2] [,3] [,4]
[1,] 48   12   18   31
[2,] 23   6    25   18
```

Aquí se ha creado una matriz llamada “Matriz1” que contiene 4 columnas (col=4) y dos filas

Para verificar el modo y la clase de la matriz, recordemos que hay que utilizar la función “mode()” y “class()”

```
> mode(Matriz1)
[1] "numeric"

> class(Matriz1)
[1] "matrix"
```

El modo es numérico y la clase pertenece a “matrix”, lo que indica que es una matriz.

Hay que recordar, que las matrices deben de ser cuadradas, su número de elementos debe de corresponder al de filas y columnas. Por ejemplo, si se elimina un elemento de la matriz anterior, y se quiere construir la matriz, R responderá de la siguiente manera:

```
> Matriz2 <- matrix(c(48, 23, 12, 6, 18, 25, 31), ncol=4)
Warning message:
In matrix(c(48, 23, 12, 6, 18, 25, 31), ncol = 4) :
  data length [7] is not a sub-multiple or multiple of the number of rows [2]
```

Se envía un mensaje de alerta indicando que el tamaño de datos de 7, no es un múltiplo del número de filas que es 2.

La sintaxis completa de la función “matrix()” para crear una matriz es la siguiente:

```
> matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Donde:

- a) data: Lo determina de forma opcional un vector de datos.
- b) nrow: Representa al número de filas
- c) ncol: Representa al número de columnas
- d) byrow: valor lógico, por defecto es FALSE o falso en este caso la matriz se llena con columnas, si es TRUE o verdadero se llena con filas.
- e) dinnames: Se utiliza para indicar el nombre de las filas y columnas.

Una matriz puede ser de caracteres. Por ejemplo, vamos a utilizar los datos de los nombres de los docentes del vector “nombre.docentes”, para crear una matriz llamada “Matriz2” con cuatro columnas.

```
> Matriz2 <- matrix(nombre.docentes, ncol = 4)
```

Para presentarla:

```
> Matriz2
 [,1]      [,2]      [,3]      [,4]
 [1,] "María Molina"  "María José Zambrana"  "Martín Espinoza"  "Imelda Ortiz"
 [2,] "Juan Pérez"    "Juan Calderón"        "Roberto Duarte"   "Mireya Fonseca"
 [3,] "Pedro Maldonado" "Sebastián Olmedo"  "Carmen Gómez"     "Francisco Heredia"
 [4,] "Asunción Ramírez" "Clementina Rodríguez" "Julián Soto"      "Tomas Castro"
 [5,] "Elena Ruiz"     "Clara Molina"       "Domitila Navas"   "Jose María González"
 [6,] "José González"   "Fausto Flores"     "Eliud Zúñiga"     "María Concepción Montero"
```

Verificamos el modo y la clase de esta matriz.

```
> mode(Matriz2)
[1] "character"

> class(Matriz2)
[1] "matrix"
```

El modo es “carácter” y la clase pertenece a “matrix”

La dimensión de la matriz se puede obtener por medio de la función “dim()”, de esta forma las dos matrices que se han creado son:

```
> dim(Matriz1)          # Matriz de 2 x 4
[1] 2 4

> dim(Matriz2)          # Matriz de 6 x 4
[1] 6 4
```

Al igual que en los vectores, se puede obtener la información de un elemento de la matriz, utilizando los corchetes.

```
> Matriz1[2, 3]          # Elemento fila 2, columna 3
[1] 25

> Matriz2[5, 3]          # Elemento fila 5, columna 3
[1] "Domitila Navas"

> Matriz2[5, ]            # Se selecciona la fila 5
[1] "Elena Ruiz"         "Clara Molina"       "Domitila Navas"   "José María González"

> Matriz2[ , 2]           # Se selecciona la columna 2
[1] "María José Zambrana" "Juan Calderón"      "Sebastián Olmedo"
[2] "Clementina Rodríguez" "Clara Molina"
[6] "Fausto Flores"
```

También se puede realizar el acceso a un elemento como vector

```
> Matriz2[3]  
[1] "Pedro Maldonado"
```

>En este caso, “Pedro Maldonado”, es el tercer elemento del vector “nombre.docentes”, que constituye la matriz.

También a las matrices se les puede asignar nombres a las filas y columnas una vez creadas con la función: “dimnames()”.

Por ejemplo se le va a asignar nombres a las columnas y listas de la “Matriz1” ya creada.

```
> Matriz1  
[,1] [,2] [,3] [,4]  
[1,] 48   12   18   31  
[2,] 23   6    25   18
```

Suponiendo que los datos representan la tasa de escolaridad, el nombre de las columnas representa los años de 2015 a 2018 y los nombres de las filas representan a los departamentos de Managua y León. La instrucción sería la siguiente:

```
> dimnames(Matriz1) <- list(c("Managua", "León"), c("2015", "20016", "20017", "2018"))
```

Conjuntamente con “dimnames()”, se utiliza la función “list()”, que establece los nombres de los objetos. Así list construye vectores alfanuméricos, el primero corresponde a las filas y el segundo a las columnas.

El resultado es el siguiente:

```
> Matriz1  
      2015 20016 20017 2018  
Managua 48     12     18     31  
León     23      6     25     18
```

Esto también se podría hacer utilizando a la creación de la matriz, incluyendo la función “dimnames()” dentro de la función “matrix()”.

Por ejemplo, vamos a crear una matriz que indique la tasa neta de escolarización de jóvenes de 16 a 19 años durante 2018 por 3 departamentos (Managua, León y Chontales).

```
> Matriz3 <- matrix(c(31.7, 37.6, 41.4, 32.5, 37.8, 45.6, 28.6, 36.7, 29.8, 34.6, 26.5, 24.4  
, ncol=4, dimnames=list(c("Managua", "León", "Chontales"), c("16 años", "17 años", "1  
8 años", "19 años"))))
```

El resultado es el siguiente:

```
> Matriz3  
      16 años 17 años 18 años 19 años  
Managua     31.7     32.5     28.6     34.6  
León        37.6     37.8     36.7     26.5  
Chontales   41.4     45.6     29.8     24.4
```

También se puede crear una matriz a partir de los datos almacenados en un archivo, utilizando la función de lectura “scan()”, que se utiliza para leer valores de datos. Por ejemplo, vamos a crear la matriz “Matriz4” con dos columnas a partir de los datos del archivo de texto “Muestra4.txt” creado anteriormente.

```
> Matriz4 <- matrix(scan("C:\\\\Users\\\\Lig\\\\Documents\\\\Programas_R\\\\datos\\\\muestra4.txt"), ncol=2)
Read 24 items
```

En la función “`scan()`”, se debe de colocar el nombre de la ruta donde se encuentra el archivo de datos y mediante doble barra “`\`”, realizar las separaciones. R como se observa indica que ha leído 24 elementos. Ahora, vamos a visualizar el resultado de la matriz.

```
> Matriz4
[,1] [,2]
[1,] 18 19
[2,] 19 17
[3,] 24 21
[4,] 17 29
[5,] 20 22
[6,] 17 18
[7,] 21 21
[8,] 23 20
[9,] 19 21
[10,] 25 23
[11,] 22 22
[12,] 16 19
```

Se observa que la matriz se compone de 12 filas y 2 columnas.

```
> dim(Matriz4)           # Matriz de 12 x 2
[1] 12 2
```

Por último para crear matrices diagonales, se utiliza la función “`diag()`”. Por ejemplo, vamos a crear un vector, llamado “Vector1” que contiene 8 datos:

```
> Vector1<-c(10,15,20,45,68,12,32,23)
```

Visualizamos el vector

```
> Vector1
[1] 10 15 20 45 68 12 32 23
```

Ahora se crea la matriz diagonal del Vector1

```
> diag(Vector1)
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] 10 0 0 0 0 0 0 0
[2,] 0 15 0 0 0 0 0 0
[3,] 0 0 20 0 0 0 0 0
[4,] 0 0 0 45 0 0 0 0
[5,] 0 0 0 0 68 0 0 0
[6,] 0 0 0 0 0 12 0 0
[7,] 0 0 0 0 0 0 32 0
[8,] 0 0 0 0 0 0 0 23
```

- Estructuras de datos (Data Frame)

Una limitante de los objetos-matrices es que todos deben de ser del mismo modo. Es decir, se pueden tener matrices numéricas, matrices de caracteres o matrices lógicas. Sin embargo, en la mayoría de situaciones, la matriz de datos asociado a un análisis, contendrá datos de varios modos en diferentes columnas. De esta forma, no se podrán utilizar los objetos-matrices, para ello se tendrá que utilizar objetos-estructuras de datos. Una estructura de datos o data frame, está compuesto por múltiples filas y columnas, pero a diferencia de la matriz, las columnas pueden ser de un tipo distinto.

Las estructuras de datos se pueden crear de dos formas. La primera y más básica es utilizar la función “`data.frame()`”, la cual une al igual que la función “`matrix()`”, objetos de varias clases, también por columnas. El resultado es un objeto de la clase “`data.frame`”, en el que cada columna aparecerá como una variable y cada fila como una observación. En los “`data.frame`”, los objetos pueden ser vectores, factores, matrices, listas e incluso hojas de datos.

En este punto, vamos a repasar todos los aspectos ya vistos.

- a) Primero vamos a crear un vector con los datos de 20 alumnos de un curso de secundaria, al cual se le va a realizar el seguimiento de un análisis nutricional. Para ello se va a registrar su edad, peso en kilos y altura en metros.

```
Secundaria1= c (14, 17, 16, 15,13,14,17,14,16,15,14,16,13,17,14,16,17,  
16,17, 16, 34.6,45.8,51.4,36.7,31.3,29.9,41.5,42.8,45.6,37.3,29.8,51.4  
,28.8,36.7,35.5,30.3,40.6,29.9,31.7,45.7,1.63,1.63,1.85,1.62,1.60,1.63  
,1.70,1.65,1.78,1.70,1.77,1.83,1.74,1.65,1.43,1.38,1.45,1.56,1.41,1.51  
)
```

Al presentar el vector tenemos:

```
> Secundaria1  
[1] 14.00 17.00 16.00 15.00 13.00 14.00 17.00 14.00 16.00 15.00 14.00  
16.00 13.00 17.00 14.00 16.00 17.00 16.00 17.00  
[20] 16.00 34.60 45.80 51.40 36.70 31.30 29.90 41.50 42.80 45.60 37.30  
29.80 51.40 28.80 36.70 35.50 30.30 40.60 29.90  
[39] 31.70 45.70 1.63 1.63 1.85 1.62 1.60 1.63 1.70 1.65 1.78  
1.70 1.77 1.83 1.74 1.65 1.43 1.38 1.45  
[58] 1.56 1.41 1.51
```

El modo y la clase d este vector son:

```
> mode(Secundaria1)          # Modo numérico  
[1] "numeric"  
  
> class(Secundaria1)         # Clase numérico  
[1] "numeric"
```

- b) Ahora como tenemos tres valores a evaluar, vamos a construir una matriz de 20 x 3, en donde los nombres de las columnas sean las variables “edad”, “peso” y “altura”.

```
> Matriz.Secundaria <- matrix(Secundaria1, 20, 3, dimnames = list(c(),  
c("Edad", "Peso", "Altura")))
```

Al presentar esta matriz tenemos lo siguiente:

```
> Matriz.Secundaria
   Edad Peso Altura
[1,] 14 34.6 1.63
[2,] 17 45.8 1.63
[3,] 16 51.4 1.85
[4,] 15 36.7 1.62
[5,] 13 31.3 1.60
[6,] 14 29.9 1.63
[7,] 17 41.5 1.70
[8,] 14 42.8 1.65
[9,] 16 45.6 1.78
[10,] 15 37.3 1.70
[11,] 14 29.8 1.77
[12,] 16 51.4 1.83
[13,] 13 28.8 1.74
[14,] 17 36.7 1.65
[15,] 14 35.5 1.43
[16,] 16 30.3 1.38
[17,] 17 40.6 1.45
[18,] 16 29.9 1.56
[19,] 17 31.7 1.41
[20,] 16 45.7 1.51
```

Comprobamos el modo y la clase de la matriz que es:

```
> mode(Matriz.Secundaria)      # La matriz es de modo numérico
[1] "numeric"

> class(Matriz.Secundaria)     # La clase es "matrix" o tipo matriz
[1] "matrix"
```

c) Ahora, vamos a convertir esta matriz en una estructura de datos o “data frame”

```
> Dataframe.Secundaria <- data.frame(Matriz.Secundaria)
```

Al visualizar el contenido del “data frame”, se presenta lo siguiente

```
> Dataframe.Secundaria
   Edad Peso Altura
1    14 34.6 1.63
2    17 45.8 1.63
3    16 51.4 1.85
4    15 36.7 1.62
5    13 31.3 1.60
6    14 29.9 1.63
7    17 41.5 1.70
8    14 42.8 1.65
9    16 45.6 1.78
10   15 37.3 1.70
11   14 29.8 1.77
12   16 51.4 1.83
13   13 28.8 1.74
14   17 36.7 1.65
15   14 35.5 1.43
16   16 30.3 1.38
17   17 40.6 1.45
18   16 29.9 1.56
19   17 31.7 1.41
20   16 45.7 1.51
```

Comprobamos el modo y la clase del “data frame”

```
> mode(Dataframe.Secundaria)           # El modo es una lista  
[1] "list"  
  
> class(Dataframe.Secundaria)          # La clase es un data.frame  
[1] "data.frame"
```

Ahora, la matriz de tipo numérico se ha convertido en el “data frame”, en un conjunto de datos con tres variables y cada una de ellas conteniendo 20 observaciones. Esto se puede observar en la ventana de medio ambiente de RStudio

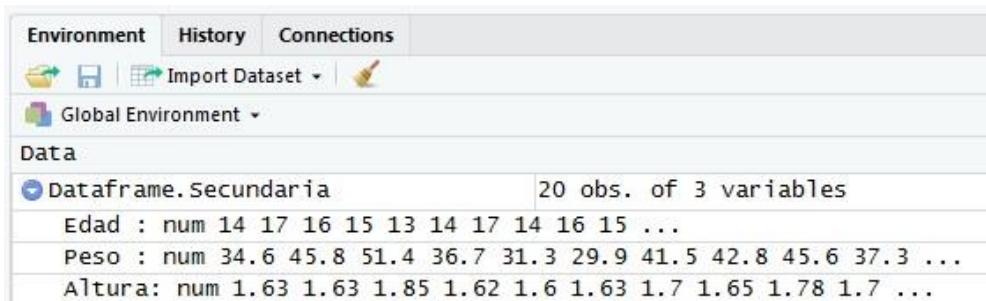


Fig. 68. Pantalla de medio ambiente con el archivo “Dataframe.Secundaria” en Data

Se pueden agregar más datos al “data frame” de varias maneras. Creando un segundo “data frame”, llamado “Dataframe.Secundaria2” y agregando el campo “sexo”, al ya existente de la siguiente manera.

```
> Dataframe.Secundaria2<-data.frame(Dataframe.Secundaria, Sexo=c("Hombre", "Hombre", "Mujer", "Hombre", "Mujer", "Mujer", "Hombre", "Hombre", "Mujer", "Hombre", "Mujer", "Hombre", "Mujer", "Hombre", "Mujer", "Hombre", "Mujer", "Mujer", "Hombre", "Mujer"))
```

Al visualizar el contenido se tiene:

```
> Dataframe.Secundaria2  
   Edad Peso Altura Sexo  
1    14 34.6   1.63 Hombre  
2    17 45.8   1.63 Hombre  
3    16 51.4   1.85 Mujer  
4    15 36.7   1.62 Hombre  
5    13 31.3   1.60 Mujer  
6    14 29.9   1.63 Mujer  
7    17 41.5   1.70 Mujer  
8    14 42.8   1.65 Hombre  
9    16 45.6   1.78 Hombre  
10   15 37.3   1.70 Mujer  
11   14 29.8   1.77 Hombre  
12   16 51.4   1.83 Mujer  
13   13 28.8   1.74 Hombre  
14   17 36.7   1.65 Mujer  
15   14 35.5   1.43 Hombre  
16   16 30.3   1.38 Mujer  
17   17 40.6   1.45 Mujer  
18   16 29.9   1.56 Mujer  
19   17 31.7   1.41 Hombre  
20   16 45.7   1.51 Mujer
```

Hemos mencionado que en los “data frames”, los objetos pueden ser de varios tipos como por ejemplo vectores. Así que vamos a crear un vector que contenga los nombres de los alumnos.

```
> Nombres.Secundaria <- c("Juan José Martínez", "Pedro Almedia", "Maria Molina", "Martín Fuentes", "Lucrecia Campos", "Elisa Perez", "Paola Morales", "Pedro Noriega", "Humberto Alcampo", "Eloisa Fuentes", "Tomas Huescas", "Sofia Campoamor", "Roberto Torres", "Cecilia Gonzalez", "Gustavo Ocampo", "Damaris Dueñas", "Felicia Jimenez", "Dolores delgado", "Feliz Suarez", "Tamara Delgado")
```

Al presentar el vector tenemos:

```
> Nombres.Secundaria
[1] "Juan José Martínez" "Pedro Almedia"      "Maria Molina"       "Martín Fuentes"
[5] "Lucrecia Campos"    "Elisa Perez"        "Paola Morales"
[8] "Pedro Noriega"     "Humberto Alcampo"   "Eloisa Fuentes"    "Tomas Huescas"
[12] "Sofia Campoamor"   "Roberto Torres"     "Cecilia Gonzalez"
[15] "Gustavo Ocampo"    "Damaris Dueñas"     "Felicia Jimenez"   "Dolores delgado"
[19] "Feliz Suarez"       "Tamara Delgado"
```

Ahora procedemos a crear un nuevo “data frame”, llamado “Dataframe.Secundaria3” en el que se incluirá el vector “Nombres.Secundaria” que se ha creado

```
> Dataframe.Secundaria3 <- data.frame(Dataframe.Secundaria2, Nombres_Alumnos=I(Nombres.Secundaria))
```

En la estructura del “data.frame”, se coloca primero el objeto que contiene los datos, en este caso el “Dataframe.Secundaria2”, separado de una coma por el objeto que se va a insertar, en este caso como título será “Nombres_Alumnos” que va a ser igual o asignado al vector creado. Hay que notar en la instrucción anterior que se ha utilizado la función “I()”. Esta función se utiliza para introducir un vector de nombres como tales, sin transformarlo en factores.

El resultado final es el siguiente:

```
> Dataframe.Secundaria3
  Edad Peso Altura Sexo Nombres_Alumnos
1   14 34.6  1.63 Hombre Juan José Martínez
2   17 45.8  1.63 Hombre Pedro Almedia
3   16 51.4  1.85 Mujer Maria Molina
4   15 36.7  1.62 Hombre Martín Fuentes
5   13 31.3  1.60 Mujer Lucrecia Campos
6   14 29.9  1.63 Mujer Elisa Perez
7   17 41.5  1.70 Mujer Paola Morales
8   14 42.8  1.65 Hombre Pedro Noriega
9   16 45.6  1.78 Hombre Humberto Alcampo
10  15 37.3  1.70 Mujer Eloisa Fuentes
11  14 29.8  1.77 Hombre Tomas Huescas
12  16 51.4  1.83 Mujer Sofia Campoamor
13  13 28.8  1.74 Hombre Roberto Torres
14  17 36.7  1.65 Mujer Cecilia Gonzalez
15  14 35.5  1.43 Hombre Gustavo Ocampo
16  16 30.3  1.38 Mujer Damaris Dueñas
17  17 40.6  1.45 Mujer Felicia Jimenez
18  16 29.9  1.56 Mujer Dolores delgado
19  17 31.7  1.41 Hombre Feliz Suarez
20  16 45.7  1.51 Mujer Tamara Delgado
```

Ahora para presentar el modo y la clase de este “data frame” de la siguiente manera:

```

> mode(Dataframe.Secundaria3)           # El modo es una lista
[1] "list"

> class(Dataframe.Secundaria3)          # La clase es un data.frame
[1] "data.frame"

```

Bien, ahora que se ha creado este último “data.frame”, vamos a guardarlo en un formato que podamos luego si deseamos importarlo. Vamos a generar por lo tanto tres archivos, uno en formato ASCII como “txt”, otro en formato “sv” y otro en formato “xlsx”, como hoja de Excel. Para ello se utilizará la función “write()” indicando a qué extensión y formato se desea acceder.

- a) Con formato ASCII, de extensión “txt”

```
> write.table(Dataframe.Secundaria3, file = "./datos/Nutri_Secundaria.txt", row.names = FALSE) # guarda un archivo txt
```

- b) Con formato “csv”

```
> write.csv(Dataframe.Secundaria3, file = "./datos/Nutri_Secundaria.csv", row.names = FALSE) # guarda un archivo csv
```

Para generar un archivo en formato de hoja de cálculo de Excel, se necesita cargar el paquete “xlsx”, se llama a dicho paquete desde el menú de RStudio: “Tools/Install Packages...” que presenta la siguiente ventana

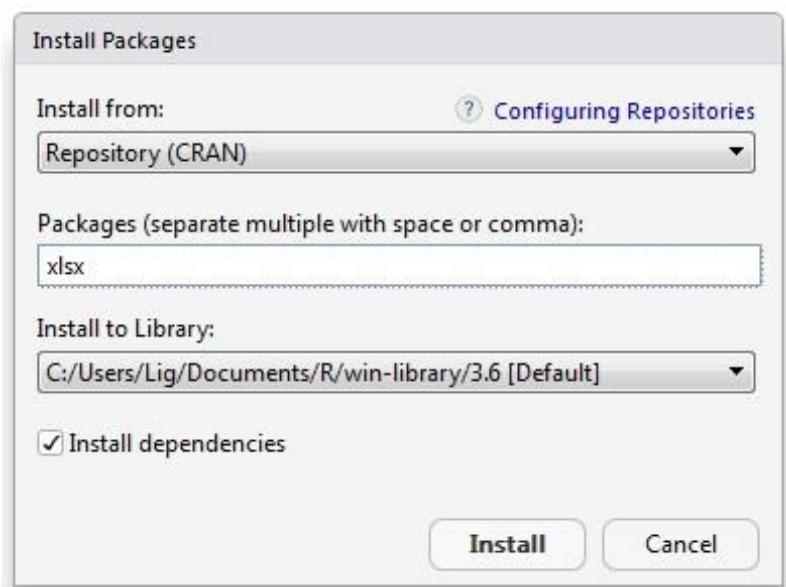


Fig. 69. Pantalla de instalación de paquetes de RStudio

Aquí se instala desde el repositorio “CRAN”, y en “Packages” se escribe el que se desea que es “xlsx” y se da clic a “Install”. La instalación presenta una vez instalado el paquete la información en la consola, similar a la siguiente:

```

> install.packages("xlsx")
Installing package into 'C:/users/Lig/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
also installing the dependencies 'rJava', 'xlsxjars'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/rJava_0.9-11.zip'
Content type 'application/zip' length 832080 bytes (812 KB)
downloaded 812 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/xlsxjars_0.6.1.zip'
Content type 'application/zip' length 9485571 bytes (9.0 MB)
downloaded 9.0 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/xlsx_0.6.1.zip'
Content type 'application/zip' length 460695 bytes (449 KB)
downloaded 449 KB

package 'rJava' successfully unpacked and MD5 sums checked
package 'xlsxjars' successfully unpacked and MD5 sums checked
package 'xlsx' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
      C:\Users\Lik\AppData\Local\Temp\Rtmpg35cAM\downloaded_packages
>

```

Fig. 70. Pantalla de comprobación de la instalación del paquete “xlsx”

Hay que recordar, que para utilizar el paquete, no solo hay que descargarlo, sino cargarlo en el sistema a través de “library()”

```
> library(xlsx)
```

Ahora ya el paquete se encuentra disponible para su uso. Por lo tanto tecleamos la instrucción para generar el archivo de extensión “xlsx” de Excel.

```
write.xlsx(Dataframe.Secundaria3, file = "./datos/Nutri_Secundaria.xlsx", row.names = FALSE) # guarda un archivo Excel
```

Podemos abrir el archivo en Excel y veremos lo siguiente:

	A	B	C	D	E	F	G
1	Edad	Peso	Altura	Sexo	Nombres_Alumnos		
2	14	34,6	1,63	Hombre	Juan José Martínez		
3	17	45,8	1,63	Hombre	Pedro Almedia		
4	16	51,4	1,85	Mujer	Maria Molina		
5	15	36,7	1,62	Hombre	Martín Fuentes		
6	13	31,3	1,6	Mujer	Lucrecia Campos		
7	14	29,9	1,63	Mujer	Elisa Perez		
8	17	41,5	1,7	Mujer	Paola Morales		
9	14	42,8	1,65	Hombre	Pedro Noriega		
10	16	45,6	1,78	Hombre	Humberto Alcampo		
11	15	37,3	1,7	Mujer	Eloisa Fuentes		
12	14	29,8	1,77	Hombre	Tomas Huescas		
13	16	51,4	1,83	Mujer	Sofia Campoamor		
14	13	28,8	1,74	Hombre	Roberto Torres		
15	17	36,7	1,65	Mujer	Cecilia Gonzalez		
16	14	35,5	1,43	Hombre	Gustavo Ocampo		
17	16	30,3	1,38	Mujer	Damaris Dueñas		
18	17	40,6	1,45	Mujer	Felicia Jimenez		
19	16	29,9	1,56	Mujer	Dolores delgado		
20	17	31,7	1,41	Hombre	Feliz Suarez		
21	16	45,7	1,51	Mujer	Tamara Delgado		
22							

Fig. 71. Pantalla del archivo “Nutri_Secundaria” de Excel

- Selección de datos de una estructura de datos o una lista.

Para seleccionar un subconjunto de datos en vectores de una estructura de datos se utiliza la función “subset()”. Esta función es importante ya que permite extraer grupos de subconjuntos de datos de una estructura que pueden ser analizados independientemente de la estructura. Por ejemplo, de la estructura de datos creada “Dataframe.Secundaria3”, vamos a obtener y crear un vector denominado Subconjunto1 de la siguiente forma:

```
> Subconjunto1 <- subset(Dataframe.Secundaria3,select=c(Edad,Nombres_Alumnos))
```

La función “subset()”, primero indica el conjunto de datos a manipular, que en este caso corresponde a una estructura de datos “Dataframe.Secundaria3”, a continuación, mediante la instrucción “select”, se selecciona el subconjunto mediante un vector, determinando que los elementos seleccionados serán, la edad y los Nombres_Alumnos. Ahora, al presentar el resultado del vector “Subconjunto1” creado la respuesta es:

```
> Subconjunto1
   Edad    Nombres_Alumnos
1    14 Juan José Martínez
2    17 Pedro Almedia
3    16 María Molina
4    15 Martín Fuentes
5    13 Lucrecia Campos
6    14 Elisa Perez
7    17 Paola Morales
8    14 Pedro Noriega
9    16 Humberto Alcampo
10   15 Eloisa Fuentes
11   14 Tomás Huescas
12   16 Sofía Campoamor
13   13 Roberto Torres
14   17 Cecilia González
15   14 Gustavo Ocampo
16   16 Damaris Dueñas
17   17 Felicia Jiménez
18   16 Dolores delgado
19   17 Feliz Suárez
20   16 Tamara Delgado
```

Al revisar el modo y la clase de este resultado se obtiene.

```
> mode(Subconjunto1)
[1] "list"

> class(Subconjunto1)
[1] "data.frame"
```

Lo que indica que el subconjunto creado es una lista y la clase es una estructura de datos

- Listas

Una lista es el objeto-dato más flexible con que cuenta R, debido a que puede admitir datos de diferentes modos y de diferentes longitudes, e inclusive otras listas. Así, las listas son, generalmente el objeto-dato final donde se van incorporando los otros tipos de datos.

La mayoría de las funciones de R que realizan análisis estadístico, presentan sus resultados a través de una lista. De esta forma, por ejemplo, una lista puede estar conformada por los datos originales, los valores ajustados, los residuos, los estadísticos de contraste, el p-valor e incluso hasta el método empleado. Todo ello se combina en un solo elemento que es la lista.

Para crear una lista en R se utiliza la función “list()”, en donde cada uno de sus argumentos se convierte en un componente de la lista.

De hecho, ya hemos estado trabajando con lista. Recordemos el dataframe antes creado y como, al preguntar el modo al que pertenecía se indicó que era una lista.

```
> mode(Dataframe.Secundaria3)           # El modo es una lista  
[1] "list"
```

Pero si por ejemplo, con el “dataframe” anterior, se quiere crear otra lista formada por esa estructura de daots “Dataframe.Secundaria3” más un vector formado por los número del 2 al 7, la instrucción sería la siguiente:

```
> Lista1 <- list((Dataframe.Secundaria3), c(2:7))
```

Al presentar el resultado tendríamos:

```
> Lista1  
[[1]]  
   Edad Peso Altura Sexo Nombres_Alumnos  
1    14 34.6  1.63 Hombre Juan José Martínez  
2    17 45.8  1.63 Hombre Pedro Almedia  
3    16 51.4  1.85 Mujer María Molina  
4    15 36.7  1.62 Hombre Martín Fuentes  
5    13 31.3  1.60 Mujer Lucrecia Campos  
6    14 29.9  1.63 Mujer Elisa Pérez  
7    17 41.5  1.70 Mujer Paola Morales  
8    14 42.8  1.65 Hombre Pedro Noriega  
9    16 45.6  1.78 Hombre Humberto Alcampo  
10   15 37.3  1.70 Mujer Eloisa Fuentes  
11   14 29.8  1.77 Hombre Tomás Huescas  
12   16 51.4  1.83 Mujer Sofía Campoamor  
13   13 28.8  1.74 Hombre Roberto Torres  
14   17 36.7  1.65 Mujer Cecilia González  
15   14 35.5  1.43 Hombre Gustavo Ocampo  
16   16 30.3  1.38 Mujer Damaris Dueñas  
17   17 40.6  1.45 Mujer Felicia Jiménez  
18   16 29.9  1.56 Mujer Dolores Delgado  
19   17 31.7  1.41 Hombre Feliz Suárez  
20   16 45.7  1.51 Mujer Tamara Delgado  
  
[[2]]  
[1] 2 3 4 5 6 7
```

Como se observa, la lista contiene tanto una estructura de datos como un vector.

El modo y la clase de “Lista1” son:

```
> mode(Lista1)      # El modo es lista  
[1] "list"  
  
> class(Lista1)     # La clase es lista  
[1] "list"
```

Se pueden realizar muchas acciones con las listas, pero no vamos a detenernos en analizar cada una de ellas, si se quiere tener más información, se puede o bien remitir a la documentación que sobre las listas y otros temas R proporciona, así como de los libros y documentos sugeridos tanto en este curso como los foros y documentos existentes en Internet.

Por ejemplo, vamos ahora a crear una lista llamada “Docentes.Lista1”, en la que vamos a unir el subconjunto “Subconjunto1”, creado a partir de la estructura de datos “DataframeSecundaria3”, además en dicha lista se va a incluir una información de los nombres de los profesores, el proyecto al que se está desarrollando, las materias impartidas a través de este proyecto y las horas por semana que se van a dedicar. La expresión será la siguiente.

```
> Docentes.Lista1 <- list(Subconjunto1, nom.profesores = c("Joaquín Morales", "Ana Rubalcaba", "Felicia Ruiz", "Gonzalo Flores"), ProyectoCreadividad=TRUE, NumeroDeHoras = 2, MateriasImpartidas = c("Matemática", "Geografía", "Física y Química", "Historia"))
```

En la instrucción de la función “list()” que se utilizó para crear el Subconjunto1, como se observa se le ha agregado un vector con los nombres de los profesores, un campo lógico a TRUE, indicando que el proyecto está vigente, una variable numérica indicando el número de horas y un vector que muestra cuales son las materias a impartir.

Al presentar el resultado, se observa lo siguiente:

```
> Docentes.Lista1  
[[1]]  
   Edad    Nombres_Alumnos  
1    14 Juan José Martínez  
2    17 Pedro Almedia  
3    16 María Molina  
4    15 Martín Fuentes  
5    13 Lucrecia Campos  
6    14 Elisa Pérez  
7    17 Paola Morales  
8    14 Pedro Noriega  
9    16 Humberto Alcampo  
10   15 Eloisa Fuentes  
11   14 Tomás Huescas  
12   16 Sofía Campoamor  
13   13 Roberto Torres  
14   17 Cecilia González  
15   14 Gustavo Ocampo  
16   16 Damaris Dueñas  
17   17 Felicia Jiménez  
18   16 Dolores delgado  
19   17 Feliz Suárez  
20   16 Tamara Delgado  
  
$nom.profesores  
[1] "Joaquín Morales" "Ana Rubalcaba" "Felicia Ruiz" "Gonzalo Flores"
```

```
$ProyectoCreatividad
[1] TRUE

$NúmeroDeHoras
[1] 2

$MateriasImpartidas
[1] "Matemática"      "Geografía"      "Física y Química" "Historia"
"
```

Como se puede observar, se han obtenido cinco objetos de esta lista. Hay que notar que cada elemento-objeto de la lista comienza con un símbolo de “\$”, el cual se utiliza para el manejo de listas. El número de elementos de esta lista se comprueba con la función “length()”

```
> length(Docentes.Lista1)      # La lista se compone de 5 elementos
[1] 5
```

De la misma forma que “Lista1”, en esta última, el modo y la clase resultante son y deben ser iguales al ser una lista.

```
> mode(Docentes.Lista1)          # El modo es lista
[1] "list"

> class(Docentes.Lista1)         # La clase es lista
[1] "list"
```

Como se ha mencionado, para seleccionar un elemento de una lista se hace uso del símbolo “\$”. Por ejemplo, para seleccionar el elemento de la lista del nombre de profesores. Primero se coloca el nombre de la lista, seguido del símbolo “\$” y a continuación el elemento-objeto que se desea mostrar de la lista. Ejemplo de esto serían la siguiente línea de código:

```
> Docentes.Lista1$nom.profesores
[1] "Joaquín Morales" "Ana Rubalcaba"   "Felicia Ruiz"    "Gonzalo Flores"
```

En caso de que se desee buscar un elemento en concreto, este se coloca entre corchetes como se muestra en el siguiente ejemplo: (*Donde se indica que se desea obtener el elemento número 2 de la lista*).

```
> Docentes.Lista1$nom.profesores [2]
[1] "Ana Rubalcaba"
```

Ahora bien, se ha mencionado que una lista puede contener diversos objetos. Entonces, vamos a crear una nueva lista, que incluya a “DocentesList1” y a la matriz creada “Matriz2”, que en este ejemplo, indicaría a la restante planta docente de un instituto. Se le va a nombrar “Docente.Lista2”, la expresión sería la siguiente.

```
> Docente.Lista2 <- list(Docentes.Lista1, Matriz2)
```

Como se observa, el proceso de crear listas a partir de vectores, estructuras de datos y matrices es muy sencillo, basta con agregar lo que se desea en la instrucción “list()”

Ahora, la lista “Docentes.Lista1” se comporta como un solo bloque al que se le agrega la matriz. Al presentar la lista resultante se observa lo siguiente

```

> Docente.Lista2
[[1]]
[[1]][[1]]
  Edad Nombres_Alumnos
1    14 Juan José Martínez
2    17 Pedro Almedia
3    16 María Molina
4    15 Martín Fuentes
5    13 Lucrecia Campos
6    14 Elisa Pérez
7    17 Paola Morales
8    14 Pedro Noriega
9    16 Humberto Alcampo
10   15 Eloisa Fuentes
11   14 Tomás Huescas
12   16 Sofía Campoamor
13   13 Roberto Torres
14   17 Cecilia González
15   14 Gustavo Ocampo
16   16 Damaris Dueñas
17   17 Felicia Jiménez
18   16 Dolores delgado
19   17 Feliz Suárez
20   16 Tamara Delgado

[[1]]$nom.profesores
[1] "Joaquín Morales" "Ana Rubalcaba" "Felicia Ruiz" "Gonzalo Flores"

[[1]]$ProyectoCreatividad
[1] TRUE

[[1]]$NumeroDeHoras
[1] 2

[[1]]$MateriasImpartidas
[1] "Matemática" "Geografía" "Física y Química" "Historia"
[[2]]
  [,1] [,2] [,3] [,4]
[1,] "María Molina" "María José Zambrana" "Martín Espinoza" "Imelda Ortiz"
[2,] "Juan Pérez" "Juan Calderón" "Roberto Duarte" "Mireya Fonseca"
[3,] "Pedro Maldonado" "Sebastián Olmedo" "Carmen Gómez" "Francisco Heredia"
[4,] "Asunción Ramírez" "Clementina Rodríguez" "Julián Soto" "Tomás Castro"
[5,] "Elena Ruiz" "Clara Molina" "Domitila Navas" "Jose María González"
[6,] "José González" "Fausto Flores" "Eliud Zúñiga" "María Concepción Montero"
"
```

Si observamos la estructura creada, se pueden visualizar dobles corchetes indicando que existen dos elementos “[1]” y “[2]”, ya que hemos agregado a la anterior estructura de datos una matriz. Esto se puede comprobar preguntando la longitud del objeto con la función “length()”.

```

> length(Docente.Lista2)
[1] 2

```

En este punto si queremos hacer uso de las sentencias para buscar los elementos de la lista, como previamente se había conseguido, antes de incorporar la matriz, habría un problema, ya que si presentamos una búsqueda de un elemento con la notación anterior tendremos lo siguiente.

```
> Docente.Lista2$nom.profesores
NULL
```

RStudio, no encontrará los elementos, porque se está construyendo mal la sentencia, al ser la estructura de datos de un solo bloque. Para este caso, hay que proceder a ponerles nombre a los elementos de la lista, es decir a esos espacios de doble corchete para referirse a ellos, en este caso a la estructura de datos y a la matriz. Para realizar esto se utiliza la función “names()” de la siguiente manera.

```
> names(Docente.Lista2) <- c('Estructura_Datos', 'Matriz')
```

Se está nombrando a los dos elementos de la lista “Docente.Lista2”. Al primero se indica como nombre “Estructura_Datos” y al segundo “Matriz”. Ahora si presentamos el contenido de la lista se observa lo siguiente:

```
> Docente.Lista2
$Estructura_Datos
$Estructura_Datos[[1]]
  Edad    Nombres_Alumnos
1   14 Juan José Martínez
2   17 Pedro Almedia
3   16 María Molina
4   15 Martín Fuentes
5   13 Lucrecia Campos
6   14 Elisa Perez
7   17 Paola Morales
8   14 Pedro Noriega
9   16 Humberto Alcampo
10  15 Eloisa Fuentes
11  14 Tomás Huescas
12  16 Sofía Campoamor
13  13 Roberto Torres
14  17 Cecilia González
15  14 Gustavo Ocampo
16  16 Damaris Dueñas
17  17 Felicia Jiménez
18  16 Dolores delgado
19  17 Feliz Suarez
20  16 Tamara Delgado

$Estructura_Datos$nom.profesores
[1] "Joaquín Morales" "Ana Rubalcaba" "Felicia Ruiz" "Gonzalo Flores"

$Estructura_Datos$ProyectoCreatividad
[1] TRUE

$Estructura_Datos$NumeroDeHoras
[1] 2

$Estructura_Datos$MateriasImpartidas
[1] "Matemática"      "Geografía"       "Física y Química" "Historia"
""

$Matriz
 [,1]          [,2]          [,3]          [,4]
```

```
[1,] "María Molina"      "María José Zambrana"    "Martín Espinoza"   "Imel
da Ortiz"
[2,] "Juan Pérez"        "Juan Calderón"       "Roberto Duarte"   "Mire
ya Fonseca"
[3,] "Pedro Maldonado"   "Sebastián Olmedo"   "Carmen Gómez"     "Fran
cisco Heredia"
[4,] "Asunción Ramírez"  "Clementina Rodríguez" "Julián Soto"      "Toma
s Castro"
[5,] "Elena Ruiz"        "Clara Molina"        "Domitila Navas"   "Jose
Maria González"
[6,] "José González"     "Fausto Flores"       "Eliud Zúñiga"     "Marí
a Concepción Montero"
```

Se observa como los nombres han cambiado. Ahora sí se puede referir a ellos mediante sus nombres con mayor facilidad para buscar o seleccionar elementos.

```
> Docente.Lista2$Estructura_Datos [1]
[[1]]
  Edad Nombres_Alumnos
1   14 Juan José Martínez
2   17 Pedro Almedia
3   16 María Molina
4   15 Martín Fuentes
5   13 Lucrecia Campos
6   14 Elisa Perez
7   17 Paola Morales
8   14 Pedro Noriega
9   16 Humberto Alcampo
10  15 Eloisa Fuentes
11  14 Tomás Huescas
12  16 Sofía Campoamor
13  13 Roberto Torres
14  17 Cecilia Gonzalez
15  14 Gustavo Ocampo
16  16 Damaris Dueñas
17  17 Felicia Jimenez
18  16 Dolores delgado
19  17 Feliz Suarez
20  16 Tamara Delgado

> Docente.Lista2$Estructura_Datos [2] # Colocando en corchetes el número
$nom.profesores
[1] "Joaquín Morales" "Ana Rubalcaba"    "Felicia Ruiz"     "Gonzalo Flo
res"

> Docente.Lista2$Estructura_Datos$nom.profesores # También el nombre
[1] "Joaquín Morales" "Ana Rubalcaba"    "Felicia Ruiz"     "Gonzalo Flo
res"

> Docente.Lista2$Estructura_Datos$nom.profesores [3] # el elemento 3
[1] "Felicia Ruiz"

> Docente.Lista2$Estructura_Datos$MateriasImpartidas [3]
[1] "Física y Química"

> Docente.Lista2[['Matriz']][3,3] # Elemento fila 3 y columna 3
[1] "Carmen Gómez"
```

7) Gráficos con los datos

Tanto en R como en RStudio, la ventana de gráficos se presenta de forma automática al ejecutar alguna función que presente gráficos. Las funciones gráficas se dividen en funciones gráficas de alto nivel y de bajo nivel. Con las funciones de alto nivel se crea un gráfico nuevo, mientras que con las funciones gráficas de bajo nivel se pueden realizar ediciones o modificaciones a los gráficos ya existentes. También se puede controlar aspectos específicos de los gráficos utilizando parámetros adicionales o propiedades de los gráficos.

Cualquier ventana que se cree con gráficos, se presenta en la misma pantalla y sustituye a la anterior, a menos que se le indique mediante alguna instrucción sobreponerla o abrirse en una nueva ventana.

Por otra parte, el resultado de una función gráfica a diferencia de las demás funciones en R, no puede ser asignada a un objeto, sino que es remitida a un dispositivo gráfico, llámese una ventana gráfica o un archivo. Hay que recordar crear una carpeta de gráficos para almacenar los gráficos generados.

- Comandos y tipos de gráficos

Es conveniente antes de comenzar con los gráficos, instalar un paquete gráfico esencial para el manejo de muchos gráficos. Es el paquete “ggplot2”. Recordemos que se puede instalar tecleando el comando desde la consola o buscando el paquete en el Menú “Tools” de RStudio.

```
> install.packages("ggplot2")
```

Una vez instalado este paquete vamos a comenzar enunciando los primeros gráficos básicos a trabajar que son:

1) Gráficas de una dimensión o dos dimensiones

- a) **plot()**: Se utiliza para diagramas de dispersión
- b) **barplot()**: Para presentar gráficas de barra
- c) **hist()**: Produce un histograma del vector numérico x
- d) **pie()**: Diagramas de pastel o tarta
- e) **boxplot()**: Para diagramas de cajas y bigotes

2) Gráficas para tres o más dimensiones

- a) **persp()**: Para representar superficies
- b) **contour() y filled()** Representa contornos
- c) **images()**: Representa cuadrícula con colores en z
- d) **symbols()**: Diagramas de dispersión con símbolos de tamaños variables

Ahora, vamos primero a construir los gráficos con una secuencia de datos, con las instrucciones aprendidas a manera de repaso, posteriormente vamos a utilizar datos de las tablas que se han generado a lo largo de este curso.

Practica 1

Enunciado. Suponiendo que se tiene los promedios de aprovechamiento de los alumnos de un curso durante el semestre y se han evaluado como: 1 = Reprobado, 2 = Malo, 3 = Aceptable, 4 = Suficiente, 5 = Bueno, 6 = Muy Bueno y 7 = Excelente.

El resultado de los 20 alumnos del curso fue el siguiente: (1,5,4,2,3,6,5,4,2,6,1,3,2,5,5,4,7,4,7,6,4,3,2,6)

Con estos datos vamos a realizar un análisis en R y construir gráficos que presenten los resultados.

- Lo primero es crear el vector de datos.

```
> Aprovechamiento <- c(1,5,4,2,3,6,5,4,2,6,1,3,2,5,5,4,7,4,7,6,4,3,2,6)
> length(Aprovechamiento)
[1] 24
```

- Para poder utilizar una variable cualitativa que indique el nivel de aprovechamiento, necesitamos construir un factor, al cual designaremos “Factor.Aprovechamiento”

```
> FactorAprovechamiento <- factor(Aprovechamiento)
```

- Una vez construido el factor, vamos a asignarle los niveles del aprovechamiento descritos, para ello se construye un vector con la función “levels()”

```
> levels(FactorAprovechamiento) <- c("Reprobado", "Malo", "Aceptable",
  "Suficiente", "Bueno", "Muy Bueno", "Excelente")
```

- El siguiente paso es convertir los datos del factor de aprovechamiento en una tabla para que se puedan utilizar los datos

```
> Datos.Aprovechamiento <- table(FactorAprovechamiento)
```

Una vez que tenemos los datos en una tabla, ahora se procederá a utilizar las funciones gráficas.

I) Función plot()

```
> plot(Datos.Aprovechamiento, main = "Aprovechamiento del curso por Alumnos de 3ro de secundaria", xlab = "Factor Evaluación", ylab = "Número de Alumnos", col = c("blue", "green", "red", "yellow", "brown4", "violet"))
```

Si observamos la función plot(), como primer argumento lleva los datos a presentar, en este caso “Datos.Aprovechamiento”, seguidamente se presenta “main” que se utiliza para poner el título al gráfico, a continuación se colocan los nombres a los ejes (x,y) con “xlab” y “ylab”, y por último se utiliza “col”, para el color de las barras, en este caso se ha designado a un vector, ya que son varias las columnas, indicando una cada color.

Se puede obtener una ayuda de la lista de colores disponibles, tecleando en la consola la función “colors()”, incluida en el paquete básico, que presenta la lista de los 657 colores disponibles para ser utilizados.

El gráfico obtenido es el siguiente:

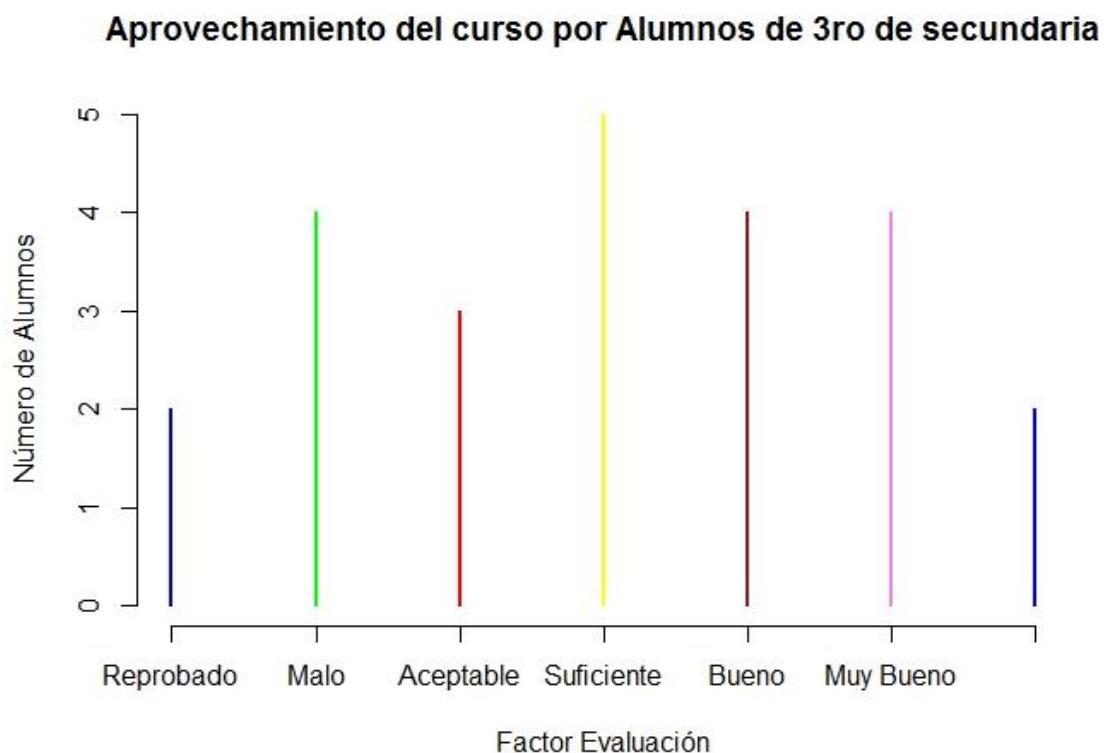


Fig. 72. Gráfico de aprovechamiento con la función “plot()”

Se puede observar que el mayor número de alumnos se encuentra en un nivel suficiente.

- ✓ **Los colores de los Gráficos:** También los colores en los gráficos, pueden ser elegidos mediante el código de un número. Para ver la lista de una codificación básica de colores la función “[colorTable\(\)](#)” del paquete “fBasics” (creado por Wertz, 2011), permite visualizar dicha tabla.

```
> library(fBasics) #Cargar el paquete con library()
Loading required package: timeDate
Loading required package: timeSeries
```

Al teclear `library(fBasics)` se cargan dos paquetes adicionales como se observa en el comando anterior

Ahora al teclear el comando de llamada a la función “colorTable”

```
> colorTable(cex=1)
```

Se abrirá en la ventana de “plots” de RStudio o en una ventana emergente de R la tabla con los colores de referencia similar a la siguiente.

Table of Color Codes

0	10	20	30	40	50	60	70	80	90
■ 1	■ 11	■ 21	■ 31	■ 41	■ 51	■ 61	■ 71	■ 81	■ 91
■ 2	■ 12	■ 22	■ 32	■ 42	■ 52	■ 62	■ 72	■ 82	■ 92
■ 3	■ 13	■ 23	■ 33	■ 43	■ 53	■ 63	■ 73	■ 83	■ 93
■ 4	■ 14	■ 24	■ 34	■ 44	■ 54	■ 64	■ 74	■ 84	■ 94
■ 5	■ 15	■ 25	■ 35	■ 45	■ 55	■ 65	■ 75	■ 85	■ 95
■ 6	■ 16	■ 26	■ 36	■ 46	■ 56	■ 66	■ 76	■ 86	■ 96
■ 7	■ 17	■ 27	■ 37	■ 47	■ 57	■ 67	■ 77	■ 87	■ 97
■ 8	■ 18	■ 28	■ 38	■ 48	■ 58	■ 68	■ 78	■ 88	■ 98
■ 9	■ 19	■ 29	■ 39	■ 49	■ 59	■ 69	■ 79	■ 89	■ 99

Fig. 73. Tabla de código de colores.

De este modo, por ejemplo, en la instrucción del gráfico anterior, se pueden utilizar los números del código, seleccionando por ejemplo (2,3,4,5,6,y 7), de la siguiente manera.

```
> plot(Datos.Aprovechamiento, main = "Aprovechamiento del curso por Alumnos de 3ro de secundaria, Colores 2", xlab = "Factor Evaluación", yl ab = "Número de Alumnos", col = c(2,3,4,5,6,7))
```

Lo que producirá un gráfico similar al siguiente:

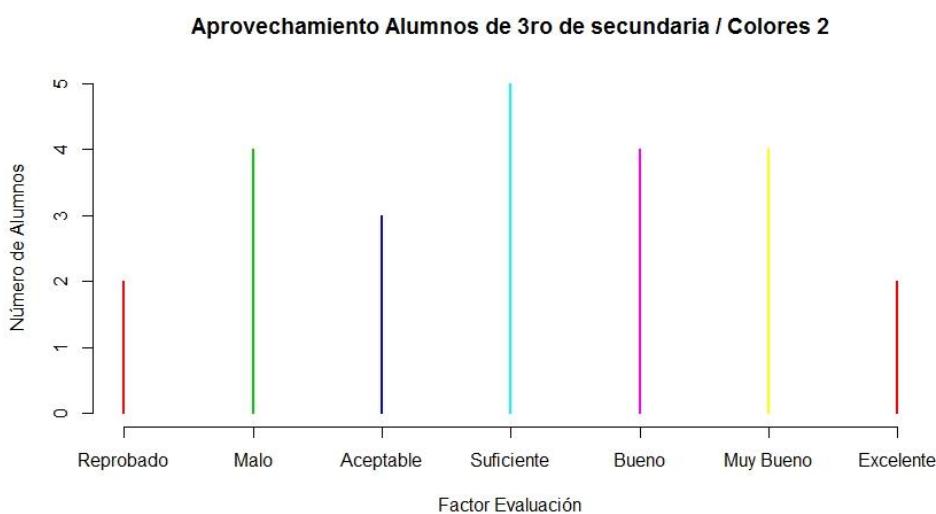


Fig. 74. Gráfico de aprovechamiento con la función “plot()”

La imagen de las gráficas, se han exportado a imágenes “jpeg”, desde el menú “Plots”, al seleccionar “Export”, como se observa en el círculo rojo de la figura siguiente. De esta forma, los gráficos se pueden almacenar como imagen o documentos.



Fig. 75. Pantalla del Menú Plots

La flecha en color azul debajo de “Files”, permite ir hacia atrás en la selección de gráficos una vez generados, para ir observándolos. El “zoom”, presenta un aumento de la pantalla de gráficos y la “X”, elimina el gráfico de la ventana y la escoba, borra todo el historial.

Al seleccionar “Export” se presenta la pantalla para guardar el gráfico similar a la siguiente.

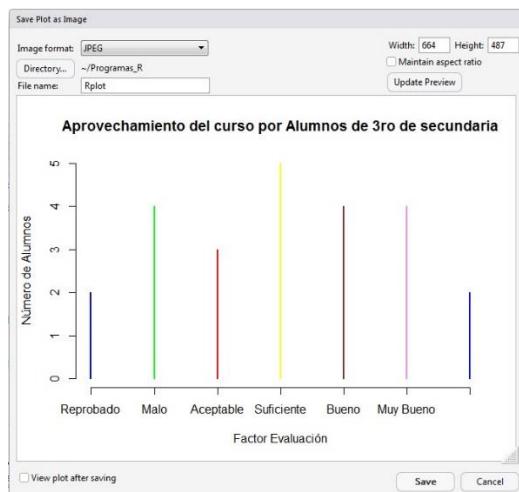


Fig. 76. Pantalla para salvar la imagen del gráfico

El directorio en donde se almacenará el gráfico se modifica a través del botón “Directory”, hay que hacerlo, porque por defecto se encuentra en el directorio de trabajo. También hay que teclear el nombre con el que se va a guardar el gráfico en el área de “File name”, una vez hecho esto, se procede a pulsar el botón de “Save”, con lo que el gráfico se guardará. Se hace esto con todos los gráficos generados. Con la función “plot()”, también se puede construir un diagrama de densidad de los datos, por medio de la función “density()”.

```
> plot(density(Datos.Aprovechamiento), main = "Diagrama de densidad",
ylab = "Densidad", col ="blue")
```

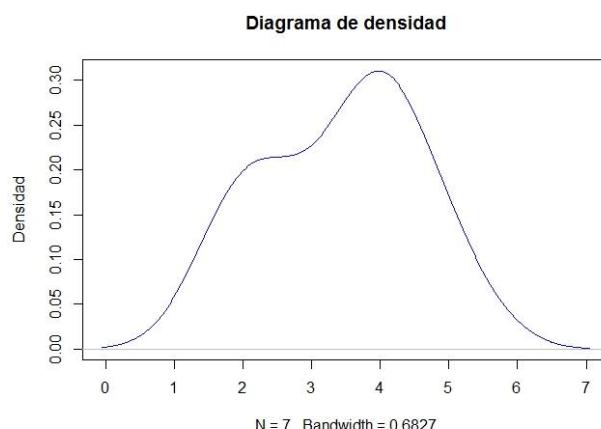


Fig. 77. Gráfico de densidad del aprovechamiento con la función “density()”

Asimismo, se puede dibujar los colores de fondo y borde con la función “polygon()”. Para esto, primero tenemos que convertir la función de densidad a un vector.

```
> densidad <- density(datos.Aprovechamiento)
```

Y ahora, mediante la función polygon(), se dibuja es espacio del vector densidad

```
> polygon(densidad, col="blue", border="red")
```

El resultado es el siguiente:

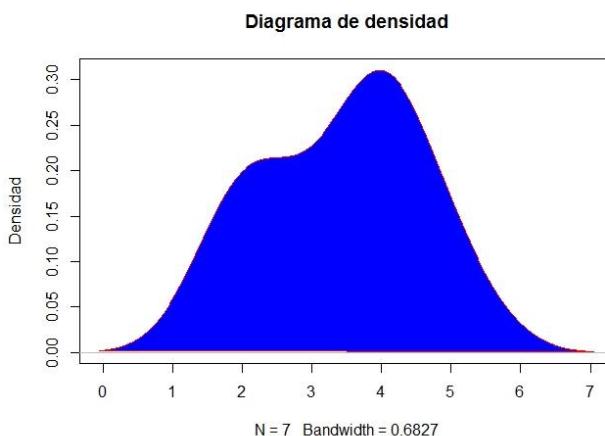


Fig. 78. Gráfico 2 de densidad del aprovechamiento con la función “density()”

- ✓ Trabajar con estructuras de datos creadas:

Para trabajar con estructuras de datos como la creada “Dataframe_Secundaria3”, se utiliza la función “attach()” que permite referenciar los nombres de las columnas de los “data.frames”, sin necesidad de especificar el nombre del “data.frame”, se puede utilizar refiriendo a la variable o por medio del símbolo \$, lo que agiliza su manipulación. La función **detach** lo desactiva.

```
> attach(Dataframe.Secundaria3)
```

Con esta instrucción, la estructura de datos es cargada en memoria para poder utilizar.

Recordemos que la estructura es la siguiente:

```
> Dataframe.Secundaria3
   Edad Peso Altura Sexo   Nombres_Alumnos
1   14 34.6  1.63 Hombre Juan José Martínez
2   17 45.8  1.63 Hombre     Pedro Almedia
3   16 51.4  1.85 Mujer    María Molina
4   15 36.7  1.62 Hombre   Martín Fuentes
5   13 31.3  1.60 Mujer    Lucrecia Campos
6   14 29.9  1.63 Mujer    Elisa Pérez
7   17 41.5  1.70 Mujer    Paola Morales
8   14 42.8  1.65 Hombre   Pedro Noriega
9   16 45.6  1.78 Hombre   Humberto Alcampo
10  15 37.3  1.70 Mujer    Eloisa Fuentes
11  14 29.8  1.77 Hombre   Tomás Huescas
12  16 51.4  1.83 Mujer    Sofía Campoamor
13  13 28.8  1.74 Hombre   Roberto Torres
```

14	17	36.7	1.65	Mujer	Cecilia Gonzalez
15	14	35.5	1.43	Hombre	Gustavo Ocampo
16	16	30.3	1.38	Mujer	Damaris Dueñas
17	17	40.6	1.45	Mujer	Felicia Jimenez
18	16	29.9	1.56	Mujer	Dolores delgado
19	17	31.7	1.41	Hombre	Feliz Suarez
20	16	45.7	1.51	Mujer	Tamara Delgado

Ahora, por ejemplo, se quiere representar un determinado grupo de datos mediante las gráficas. Para ello, se seleccionan como se ha indicado y lo podemos observar mediante la siguiente instrucción, utilizando la función “plot()”. En el siguiente ejemplo se graficarán los datos de Peso y Altura de esta estructura de datos.

```
> plot(Edad,Altura, main = "Gráfica de la Edad y Altura de los Alumno s de Secundaria", xlab ="Edad", ylab = "Altura", col = c(2:8))
```

El color como se observa se ha indicado con un vector desde 2 hasta 8, R decide qué color poner en ese rango. El gráfico resultante es un gráfico de puntos.

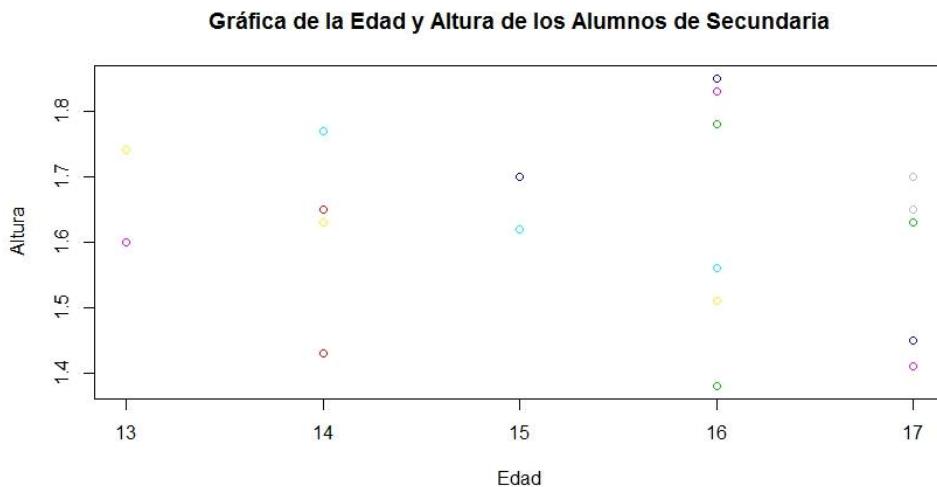


Fig. 79. Gráfico de puntos de las variables Edad y Altura de los estudiantes

- ✓ También se puede trabajar con archivos creados: Hay que recordar, que en Rstudio se pueden cargar los datos almacenados, ya sea leyéndolos a través de la función “read()” para R y RStudio, o mediante la ventana “medio ambiente” para RStudio, al seleccionar “Import Dataset”.

Recordamos la tabla de datos creada: “NutriSecundaria”, con la instrucción de lectura siguiente

```
> Nutri_Secundaria <- read.csv("~/Programas_R/datos/Nutri_Secundaria.c sv")
```

En este caso, funciona utilizando el símbolo “\$”, para utilizar dos variables (x,y) en plot(), así por ejemplo las variables “Peso” y “Altura”, podrían invocarse de la siguiente manera

```
> plot(x = Nutri_Secundaria$Peso, y = Nutri_Secundaria$Altura, main = "Gráfica de Educación: Peso vs Altura", xlab = "Peso de Estudiantes", ylab = "Altura Estudiantes", col = c(2:8))
```

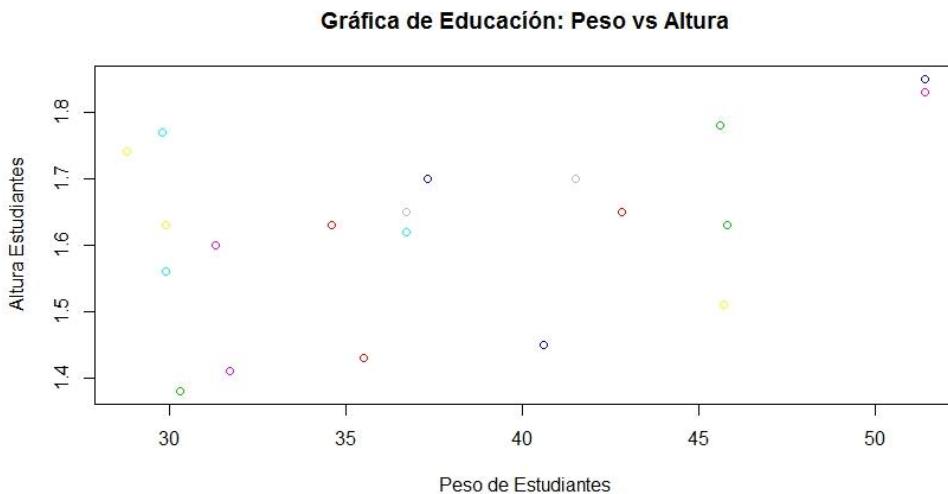


Fig. 80. Gráfico 2 de puntos de las variables de dad y Pesos de los estudiantes

En la gráfica anterior, todos los puntos son circulares, utilizando el parámetro “pch” se designa el tipo de símbolo a utilizar. Los códigos de dichos símbolos se observan en la siguiente imagen.

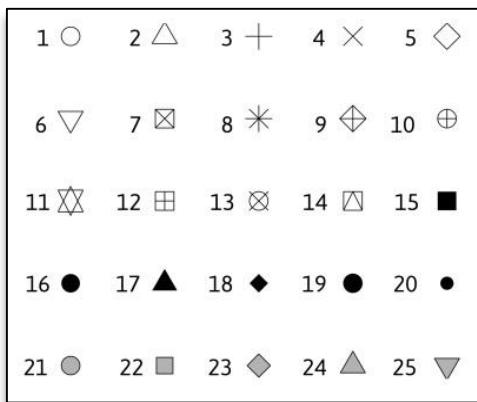


Fig. 81. Tipos de símbolos a utilizar en los gráficos con sus códigos

Vamos a modificar el símbolo de las variables. Para “Peso se utilizará el código 9 y para altura el 15. La expresión para crear la gráfica es la siguiente.

```
> plot(x = Nutri_Secundaria$Peso, y = Nutri_Secundaria$Altura, pch = c(9,15), main = "Gráfica de Educación: Peso vs Altura", xlab = "Peso de Estudiantes", ylab = "Altura Estudiantes", col = c(2,5))
```

También, vamos a presentar una leyenda que indique el nombre de esos símbolos, para eso se utiliza la función “legend()”, la expresión a utilizar para añadir la leyenda, después de construir el gráfico es la siguiente.

```
> legend("topleft", legend=c("Peso","Altura"), pch = c(9,15), col = c(2,5) )
```

Si observamos, se ha designado que la leyenda se encuentre a la izquierda y arriba con “topleft”, la ubicación del argumento “legend()”, puede tener las siguientes opciones.

`"bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"`

Como nota, hay que mencionar que es importante utilizar los mismos parámetros que en la construcción del gráfico, es decir, los símbolos y el color deben de ser iguales. La gráfica resultante, en donde se aprecian mejor los datos es la siguiente.

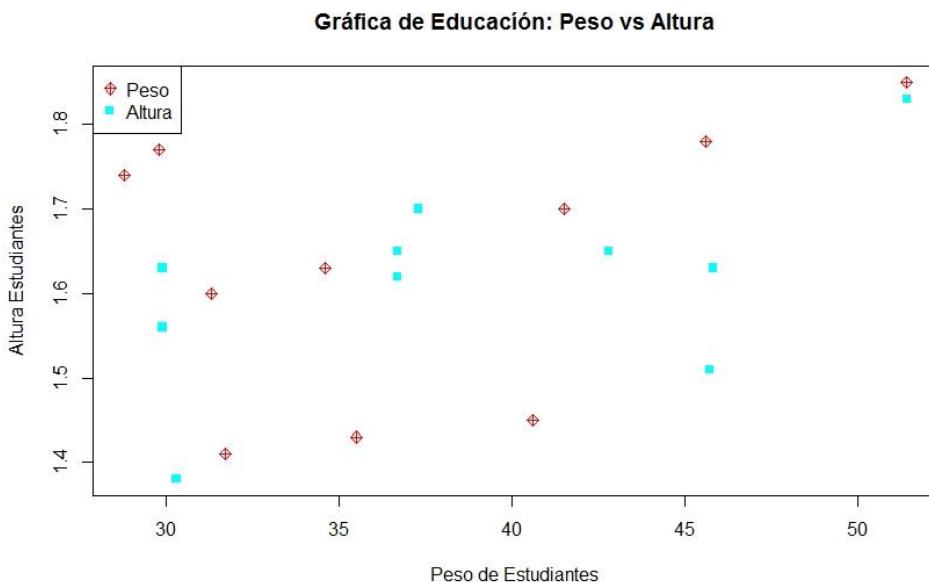


Fig. 82. Gráfica de Peso vs Altura de los estudiantes utilizando símbolos y leyenda

Ahora bien, para el caso de las variables no numéricas, por ejemplo “Sexo”, también con “plot()”, se puede obtener el gráfico. La expresión siguiente:

```
> plot( Dataframe.Secundaria3$Sexo, main = "Gráfica del Sexo de los alumnos", xlab = "Sexo", ylab = "Cantidad", col = c(42,21), las=2, legend = T) # las = 2, hace los textos perpendiculares al eje
```

El parámetro “las= 2”, como se indica, hace que los textos se puedan poner en vertical. El gráfico resultante es el siguiente

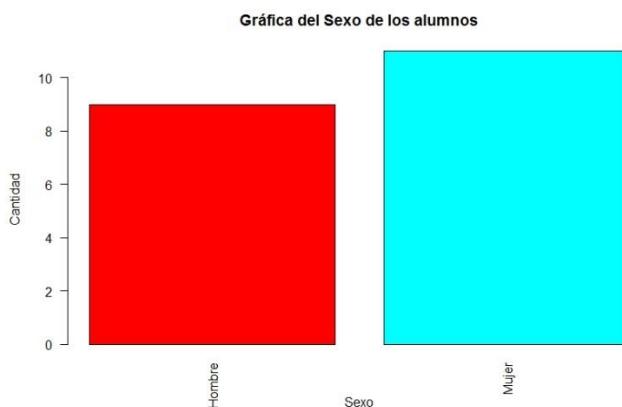


Fig. 83. Gráfica que muestra el sexo de los alumnos.

II) Función barplot()

Cuando se tienen disponibles los datos es más fácil la construcción de las gráficas. Para la función “barplot()”, sería lo siguiente:

```
> barplot(Datos.Aprovechamiento, main = "Aprovechamiento del curso por Alumnos de 3ro de secundaria", xlab = "Factor Evaluación", ylab = "Número de Alumnos", col = c("blue","green", "red", "yellow", "brown4","violet"))
```

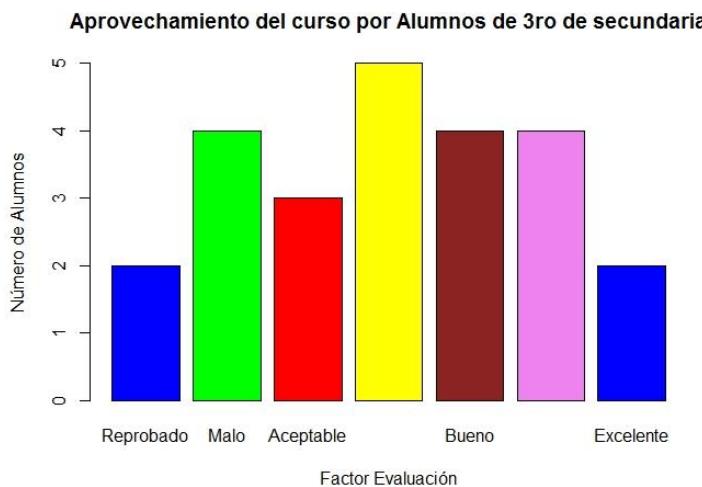


Fig. 84. Gráfico de aprovechamiento con la función “barplot()”

Para incluir nombres de leyenda dentro del gráfico se utiliza la función “legend.text()”, la cual se va a tomar de los nombres de las filas “rownames” de la tabla.

```
> barplot(Datos.Aprovechamiento, main = "Aprovechamiento del curso por Alumnos de 3ro de secundaria", legend.text= rownames(Datos.Aprovechamiento), xlab = "Factor Evaluación", ylab = "Número de Alumnos", col = c("blue","green", "red", "yellow", "brown4","violet"))
```

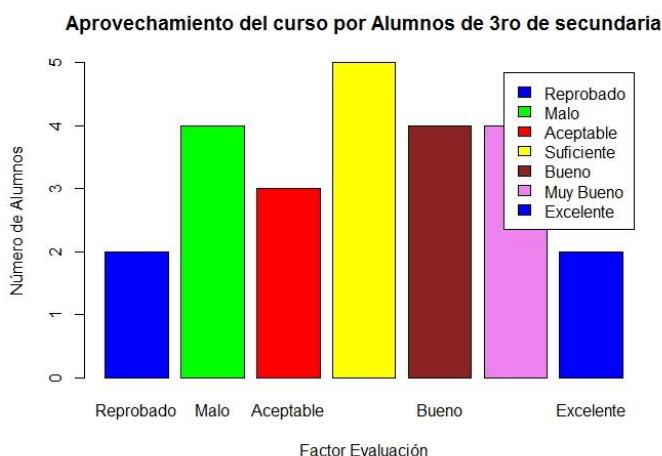


Fig. 85. Gráfico de aprovechamiento con la función “barplot()” con leyendas

Se puede modificar el límite del eje de las coordenadas utilizando la función “ylim(x,y)”

Trabajar con el Dataframe o con los archivos: Se puede utilizar ya sea el anterior dataframe o la estructura de datos para seleccionar un determinado campo, se puede seleccionar por ejemplo el peso de los estudiantes.

```
> barplot(Peso, main = "Gráfica de Educación: Peso de los Estudiantes",
  "xlab = "Número de Estudiantes", ylab = "Peso de los Estudiantes", c
ol = c(2:20))
```

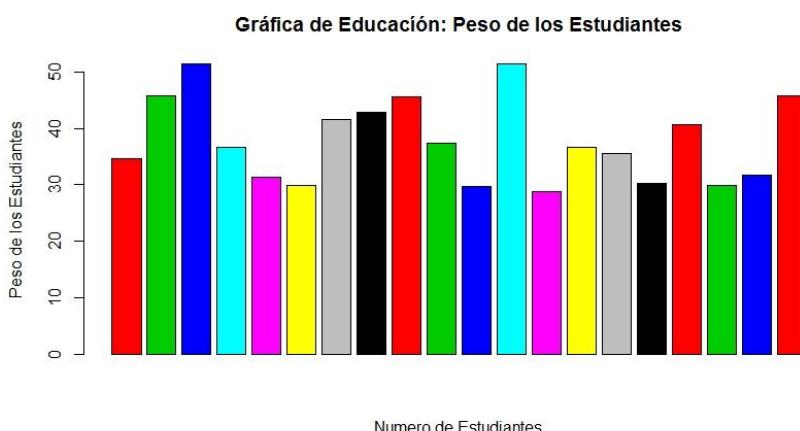


Fig. 86. Gráfico de barras de la variable “Peso” de los estudiantes de secundaria

No se observa ningún valor debajo de las barras, para ello se utiliza el argumento “names.arg”, el cual puede ser un vector. Por ello vamos a crear un vector con los datos del peso de los estudiantes.

```
> li <- c(Nutri_Secundaria$Peso)
```

Ahora lo incorporamos en el argumento de la forma siguiente.

```
> barplot(Peso, main = "Gráfica de Educación: Peso de los Estudiantes",
  "xlab = "Número de Estudiantes", ylab = "Peso de los Estudiantes", c
ol = c(2:20), names.arg = c(li))
```

El resultado es el siguiente.

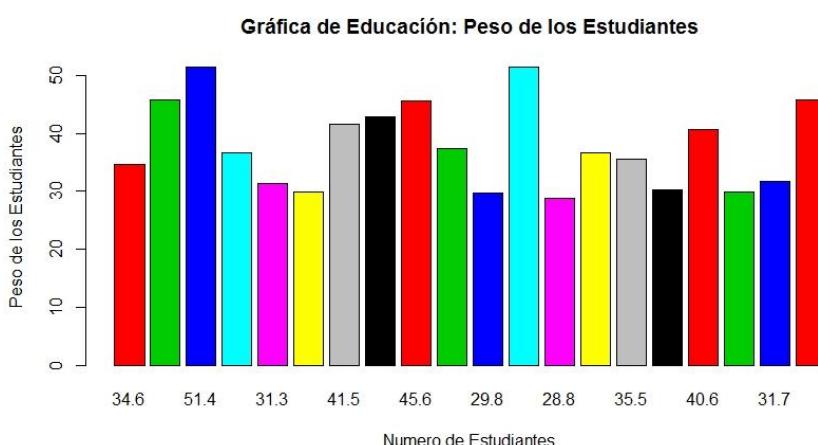


Fig. 87. Gráfico de barras con etiquetas, de la variable “Peso” de los estudiantes de secundaria

III) Función hist()

El histograma representa las frecuencias de una variable continua mediante áreas. Un ejemplo típico podría ser por ejemplo un vector (Altura), que tomamos de la tabla de Nutri_Secundaria. Para crear el vector con la expresión.

```
> Vector_Peso_V3 <- c(Nutri_Secundaria$Altura)
```

Este vector se utiliza para construir el diagrama de frecuencias siguiente.

```
> hist(Vector_Peso_V3, col = (2:7), main = "Datos Nutricion Alumnos 3r o. Secundaria", xlab = " Altura", ylab = "Frecuencia", labels = TRUE)
```

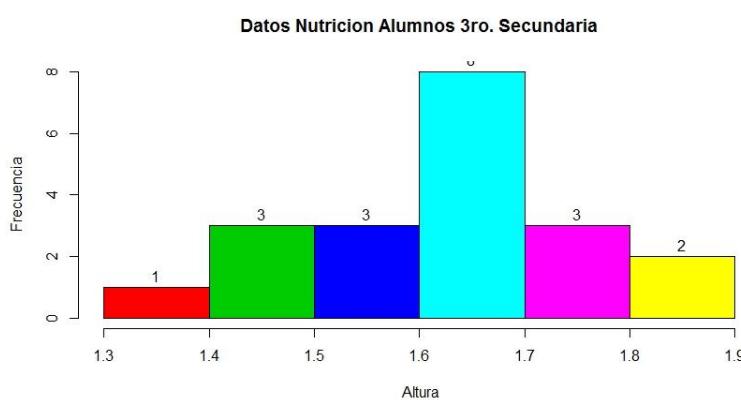


Fig. 88. Gráfico de Frecuencias de la variable Altura de la tabla “Nutri_Secundaria”

Para el caso directo de la lectura de los datos desde el datafram “Nutri_Secundaria”, podemos realizar el análisis de uno de sus elementos que es la edad, de acuerdo a la siguiente expresión.

```
> hist(x = Nutri_Secundaria$Edad, main = "Gráfico de Frecuencia de Edad de los alumnos", xlab = "Edad", ylab="Frecuencia", col = c(2:20), breaks="Sturges", labels = TRUE)
```

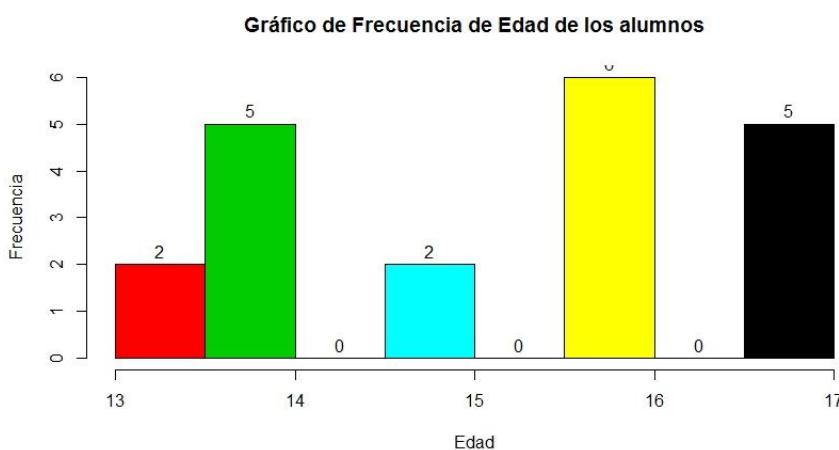


Fig. 89. Gráfico de Frecuencia de la edad

Para el gráfico de dispersión del histograma, se indica la frecuencia de corte por medio de “breaks”, que se establece al tamaño de los datos, lo que permite visualizar mejor los resultados.

Si en lugar de Frecuencia, lo que se desea presentar es la probabilidad, se establece el parámetro “probability = TRUE” a verdadero.

```
> hist(x = Nutri_Secundaria$Edad, main = "Probabilidad en lugar de Frecuencia", xlab = "Edad", col = c(21:28), breaks="Sturges", probability = TRUE, labels = TRUE)
```

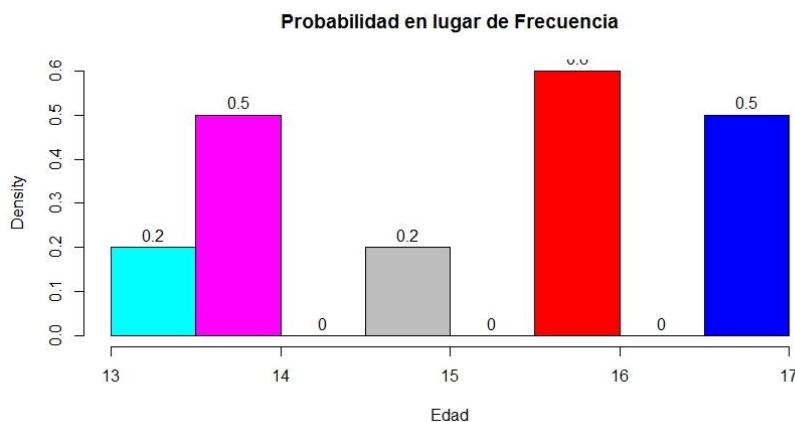


Fig. 90. Gráfico de Frecuencia de la edad con la probabilidad en lugar de frecuencia

Para insertar una línea que estime la densidad de probabilidad, se utiliza la función “lines()”, así para agregar una línea que estime la densidad de probabilidad, de un modo no paramétrico, y teniendo como parámetro “lwd” para definir el grosor de la línea, el parámetro “na.rm”, pertenece al cálculo de la estimación de densidad KERNEL, es un valor lógico, si es verdadero (TRUE), los valores faltantes se eliminan de x. En caso de que sea “FALSE”, cualquier valor faltante causa un error.

```
> lines(density(Nutri_Secundaria$Edad,na.rm = TRUE),lwd=2,col="red")
```

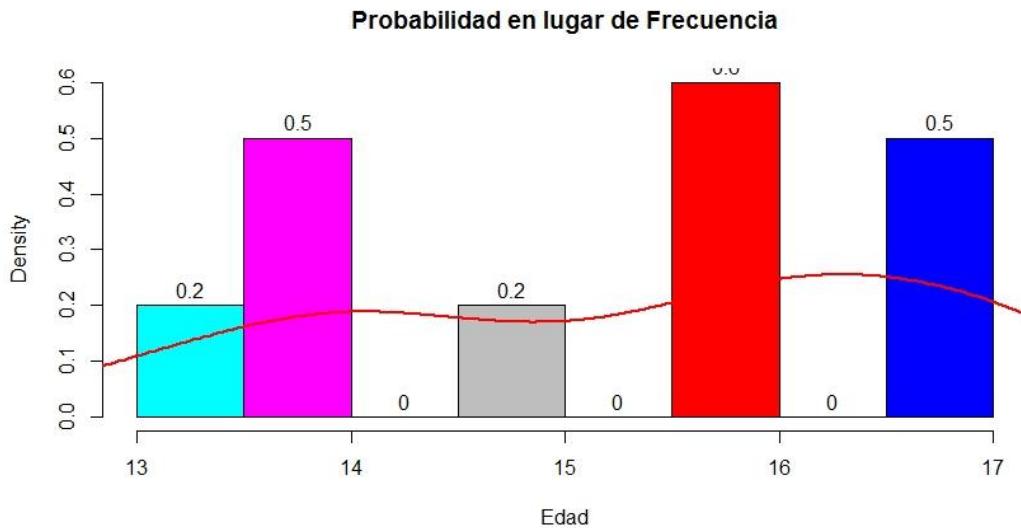


Fig. 91. Gráfico que muestra la probabilidad con la línea de densidad de probabilidad

Y si se desea presentar la línea de manera punteada, fijando el número de clases a 20 se realiza la siguiente modificación en la expresión “lines()”. El parámetro “lty =2” indica puntear la línea.

```
> lines(density(Nutri_Secundaria$Edad, na.rm = TRUE), lwd=2, lty=2, ps=20,
  col="blue")
```

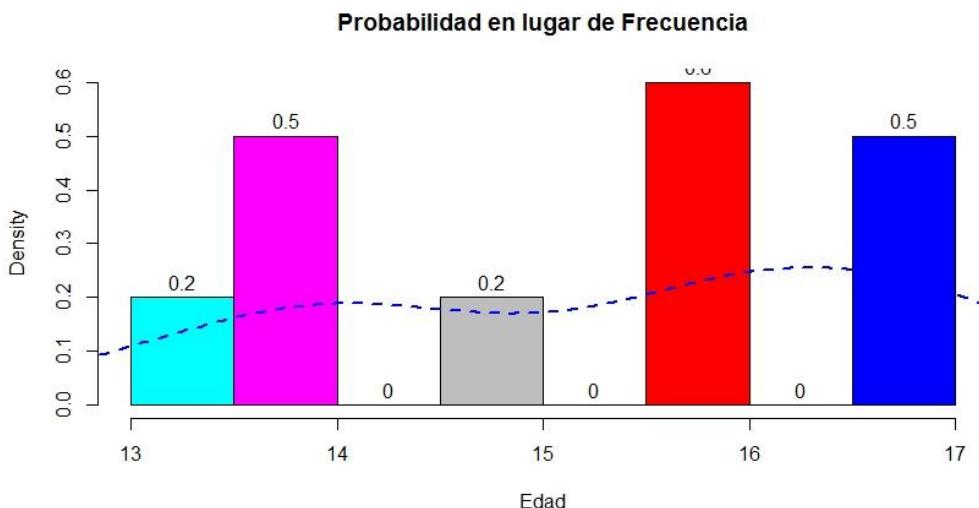


Fig. 92. Gráfico que muestra la probabilidad con la línea punteada de densidad de probabilidad

Para suavizar la curva de la distribución normal, se utiliza el parámetro “adjust” igualándolo al valor deseado. Vamos a probar con un valor de 2. La siguiente expresión dibujará la segunda línea de ese suavizado o ajuste de la línea, que quedará de la siguiente forma:

```
> lines(density(Nutri_Secundaria$Edad, adjust=2 ), lwd=3, col="brown3")
```

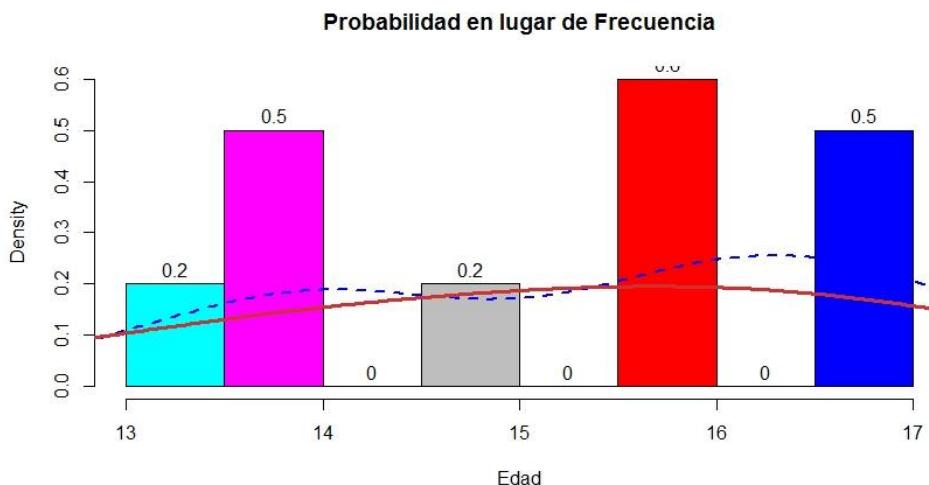


Fig. 93. Gráfico que muestra la probabilidad, con la línea densidad de probabilidad punteada en azul y la suavizada a valor 2 en marrón.

IV) Función pie()

En la función gráfica “pie()”, se pueden añadir los argumentos que se han utilizado, incluyendo además el argumento “clockwise”, que se refiere a rotar a favor de las manecillas del reloj, la instrucción queda de la siguiente manera.

```
> pie(Datos.Aprovechamiento, clockwise = TRUE, main = "Aprovechamiento del curso por Alumnos de 3ro de secundaria", col = c("blue","green", "red", "yellow", "brown4","violet"))
```

Aprovechamiento del curso por Alumnos de 3ro de secundaria



Fig. 94. Gráfico de aprovechamiento con la función “pie()”, con uso de “clockwise”

En cambio, si no se utiliza “clockwise”, el gráfico resultante será el siguiente:

Aprovechamiento 2 del curso por Alumnos de 3ro de secundaria

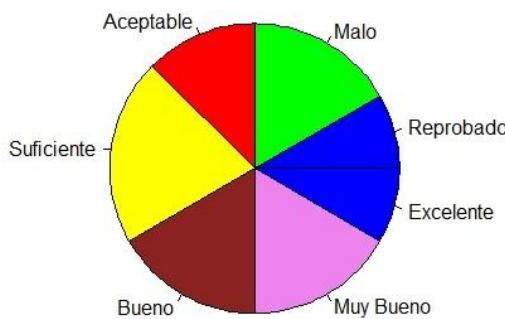


Fig. 95. Gráfico de aprovechamiento con la función “pie()”, sin uso de “clockwise”

Para incluir las cantidades en los sectores, se crea un vector de la tabla al que llamamos “Cantidad”, mediante la función “**round()**” se redondea los números en caso de haber decimales.

```
> Cantidad <- round(Datos.Aprovechamiento)
```

Ahora se crea un vector denominado etiquetas, en el que se juntan, el nombre de las etiquetas con las cantidades, por medio de la función “**paste()**”, se hace uso de “**rownames()**” para obtener el nombre de las filas de la tabla.

```
> Etiquetas <- paste(rownames(Datos.Aprovechamiento), Cantidad)
```

Ahora, se incluye esta etiqueta a través de “labels” en la función de gráfica.

```
> pie(Datos.Aprovechamiento, clockwise = TRUE, main = "Aprovechamiento 2 del curso por Alumnos de 3ro de secundaria", col = c("blue","green", "red", "yellow", "brown4","pink","orange4"), labels = Etiquetas)
```

Aprovechamiento 2 del curso por Alumnos de 3ro de secundaria



Fig. 96. Gráfico de aprovechamiento con la función “pie()” con uso de etiquetas

Si queremos trabajar con un archivo. El archivo de datos “Nutri_Secundaria” y de ahí vamos a tomar los valores de la variable “Peso”, creamos una tabla con la instrucción.

```
> tabla_peso <- table(Nutri_Secundaria$peso)
```

Ahora se crea el gráfico con la instrucción

```
> pie((tabla_peso), col = c(2:20), main = "Diagrama de Pesos de los Alumnos")
```

El resultado es el siguiente

Diagrama de Pesos de los Alumnos

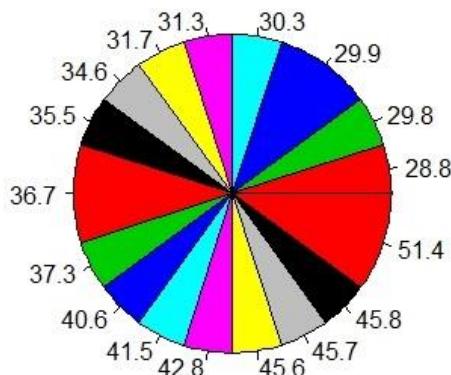


Fig. 97. Gráfico de pastel de la variable “Peso”.

V) Función boxplot()

La función boxplot(), construye el diagrama de cajas, que se utiliza para variables cuantitativas. Se representa la mediana (línea gruesa dentro de la caja) y otros parámetros relacionados con su dispersión. Los bigotes son las líneas (punteadas) con unos segmentos finales (superiores e inferiores). También en los diagramas de cajas, se podrán observar puntos por encima o por debajo de los límites de los bigotes, estos se conocen como valores extremos (valores atípicos) u *outliers* (en inglés).

Por ejemplo, vamos a construir un gráfico sencillo de cajas, con los datos de aprovechamiento de los alumnos de tercero de secundaria. Para ello utilizamos el vector creado “Aprovechamiento”

```
> boxplot(Aprovechamiento, main = "Aprovechamiento Alumnos 3ro Secundaria", col = "orange")
```

El gráfico resultante es el siguiente

Aprovechamiento Alumnos 3ro Secundaria

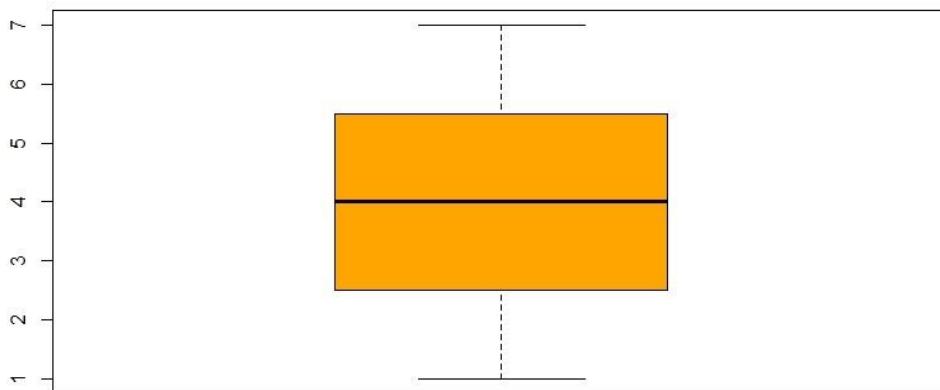


Fig. 98. Gráfico de aprovechamiento con la función “boxplot()”

La mediana de aprovechamiento se encuentra en torno a 4

```
> median(Aprovechamiento)
[1] 4
```

Ahora bien, recordemos las edades de los profesores “edad”, con estos datos del vector “edad”,

```
> edad
[1] 18 19 24 17 20 17 21 23 19 25 22 16 19 17 21 29 22 18 21 20 21 23
22 19
```

Procedemos a crear el siguiente gráfico.

```
> boxplot(edad, main = "Edad de los docentes", col ="tomato3")
```

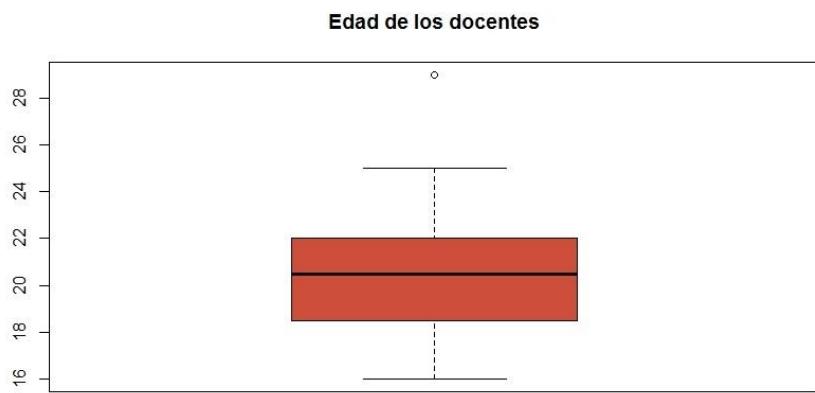


Fig. 99. Gráfico de diagrama de cajas, con la edad de los docentes del curso

El gráfico refleja una anomalía o valor atípico (outliers), en la parte superior (el valor de 29), la mediana se encuentra en torno a 20.

```
> median(edad)
[1] 20.5
```

Si se desea obtener una orientación vertical, basta con incluir el parámetro “” a TRUE

```
> boxplot(edad, main = "Edad de los docentes", horizontal=TRUE, col =
37)
```

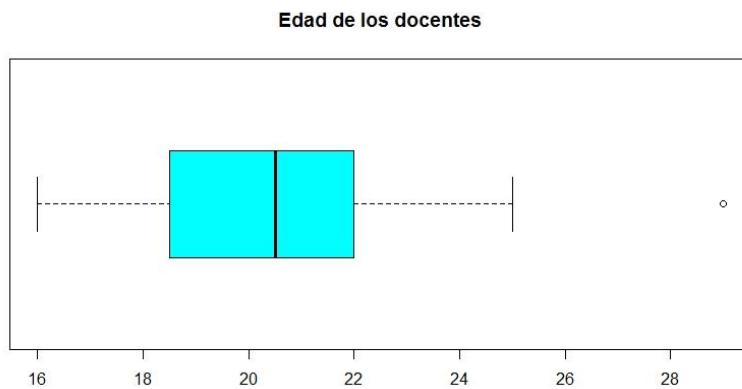


Fig. 100. Gráfico de diagrama de cajas horizontal, con la edad de los docentes del curso

Para incluir o no los valores atípicos (outliers) en el gráfico, se utiliza el argumento “outline”, se asigna igual a TRUE o FALSE (outline = TRUE/FALSE), según se requiera.

Combinar Histogramas y boxplot:

Se pueden combinar diagramas de frecuencia con diagramas de cajas. Para ello se necesita instalar el paquete “sfsmisc”, ya sea a través de la instrucción o por medio de instalación de laquetes en RStudio y teclear el nombre del paquete.

```
> install.packages("sfsmisc")
```

R responderá con la afirmación de que se ha instalado con éxito dicho paquete:

```
package 'sfsmisc' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
```

C:\Users\Lig\AppData\Local\Temp\Rtmp25N0z1\downloaded_packages

Hay que recordar cargar el paquete en memoria

```
> library(sfsmisc)
```

Ahora se puede realizar la creación de un gráfico combinado mediante el uso de la función gráfica “**histBxp**”, por ejemplo, para combinar los histogramas y diagramas de caja de la variable edad se genera la siguiente expresión:

```
> histBxp(edad, main = "Edad de los docentes", xlab = "Edad", ylab = "Frecuencia", labels = TRUE, col = (2:20), probability = F, boxcol = 0, medcol = 1 )
```

Los parámetros **boxcol** y **med col**, hacen alusión al color de la caja y de la línea.

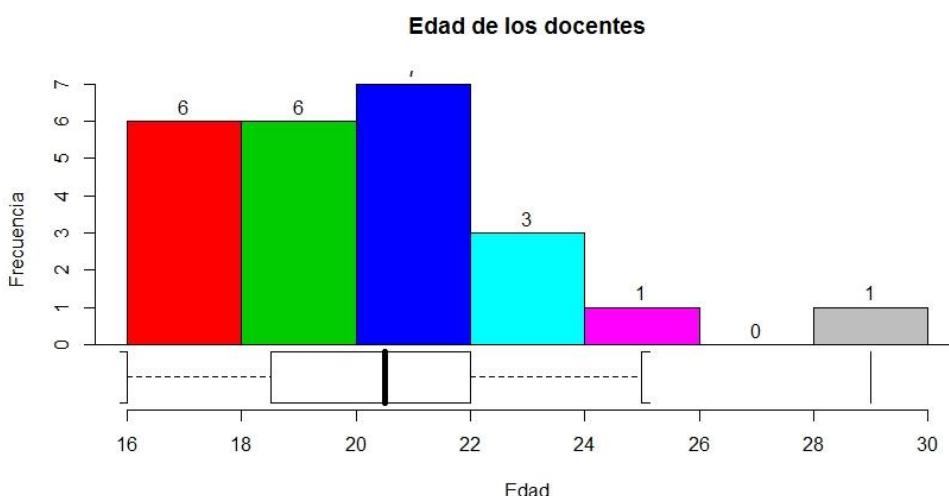


Fig. 101. Gráfico de histograma y diagrama de cajas, con la edad de los docentes del curso

Y para la presentación de la densidad de probabilidad con la línea punteada. La expresión será lo siguiente.

```
> histBxp(x = Nutri_Secundaria$Edad, main = "Edad de los Alumnos", xlab = "Edad", ylab ="Probabilidad", col = c(21:28), breaks= 20,probability = TRUE, labels = TRUE, boxcol = 0, medcol = 1 )  
  
> lines(density(Nutri_Secundaria$Edad,na.rm = TRUE),lwd=2,lty=2,ps=20, col="blue")
```

Se puede observar el valor de la mediana en la siguiente figura en torno a 16

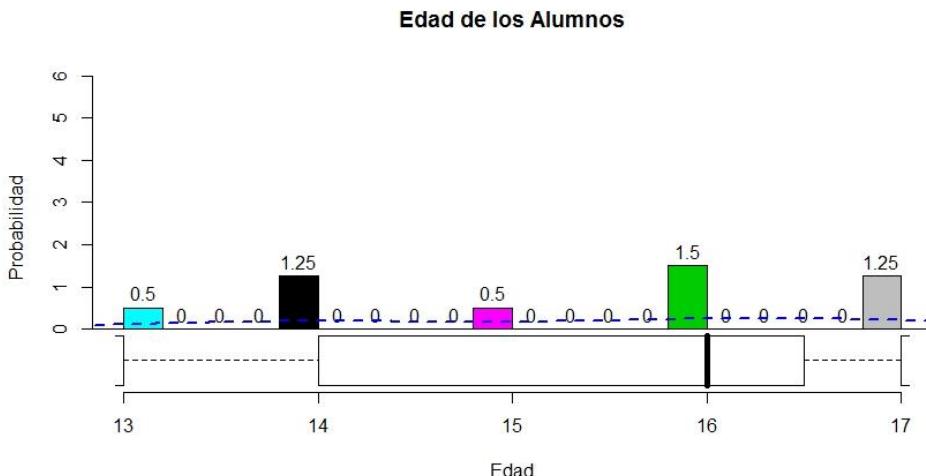


Fig. 102. Gráfico de histograma y diagrama de cajas, de la densidad de probabilidad, de la edad

8) Uso de Script en R, RStudio y RCmdr

Los scripts son los programas en el que se guardan los códigos de R, para posteriormente editarlos, ejecutarlos y realizar diversas acciones con ellos. En realidad, son documentos de texto, es decir, pueden abrirse con un bloc de notas, incluso se pueden crear con él. Sin embargo, los scripts en R tienen la extensión “R”, que indica al programa que puede leer y ejecutar su contenido.

Hasta ahora se han venido utilizando las expresiones de manera interactiva, tecleando la función y ejecutándola en la consola de comandos. Aunque las instrucciones se pueden mantener en el historial, este puede ser borrado, por lo que es útil el mantener el código, sobre todo cuando es complejo en un script, de ahí su importancia.

Para crear un script en R, hay que remitirse al menú “Archivo/Nuevo scripts”, con lo que se abrirá la ventana adicional (no dentro de la interfaz de R) de edición del script, ahí es donde se comenzará a escribir el código o las instrucciones que se deseen incorporar.

Hay que recordar la organización de la que hemos venido hablando, por lo que es recomendable, crear una carpeta “datos” o “scripts”, en el directorio de trabajo, para ahí almacenar todos los scripts generados.

En RStudio, se crea un nuevo “scripts” a través del Menú, “File/New File/ R Scripts”, en la ventana de archivo se presenta la ventana del nuevo script.

A continuación, se observa un ejemplo, con los datos del aprovechamiento que se han venido utilizando en este módulo.

```

1 # Script de Aprovechamiento de un curso por parte de Alumnos de 3ro de secundaria
2 # Datos conteniendo el aprovechamiento durante el ciclo escolar
3
4 Aprovechamiento <- c(1,5,4,2,3,6,5,4,2,6,1,3,2,5,5,4,7,4,7,6,4,3,2,6)    # Datos registrados
5
6 FactorAprovechamiento <- factor(Aprovechamiento)    ## Creación del factor Aprovechamiento
7
8 levels(FactorAprovechamiento) <- c("Reprobado", "Malo", "Aceptable", "Suficiente", "Bueno", "Muy Bueno", "Excelente")
9
10 Datos.Aprovechamiento <- table(FactorAprovechamiento)  # Creación de la tabla de Aprovechamiento de los alumnos
11
12 # Ejemplo de creación de la gráfica
13
14 plot(Datos.Aprovechamiento)      # Gráfica simple de uso de plot

```

Fig. 103. Pantalla del script de Aprovechamiento de R

Como se observa en la pantalla se han escrito los códigos (*o copiado, desde la ventana de comandos mediante Copiar y Pegar*), cada línea tiene una numeración a manera de orden, si se desea ejecutar una o varias líneas se seleccionan y se da clic al botón “Run”, para que sean ejecutadas desde la consola.

9) Guardar y visualizar datos

Los scripts pueden ser cargados al programa desde el menú File, (File/Open File) y buscar el programa que se necesite. Sin embargo, desde la consola también se puede cargar y ejecutar un scripts con la función “source()” de la siguiente manera:

```
source("C:/Users/Lig/Documents/Programas_R/Aprovechamiento.R")
```

En esta línea de código, se indica el camino donde está almacenado y el nombre del script, al dar “Enter”, se ejecutarán las instrucciones. Si hay un gráfico como en el del archivo ejemplo, este se presentará en la ventana “Plots” de RStudio, en R se abrirá una nueva ventana con el gráfico.

Para guardar un script se selecciona “Save as” o “Save” del menú File, o se da clic icono de guardar representado por un diskette. Al guardar un “scripts”, puede que se presente una pantalla indicando la codificación que se desea del archivo, sobre todo si se han utilizado caracteres como apóstrofos u otro especial. Esta ventana es similar a la siguiente

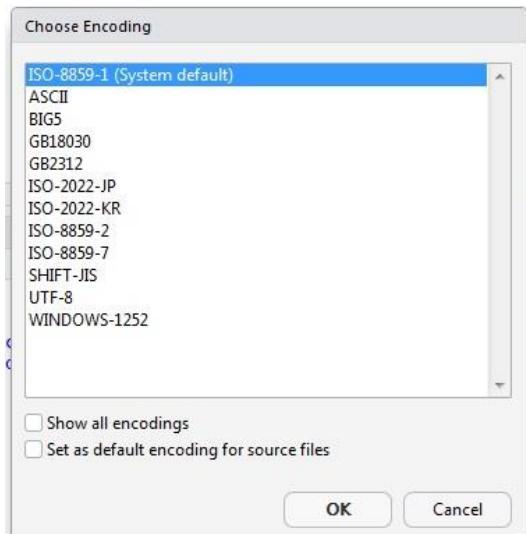


Fig. 104. Pantalla de elección de codificación del archivo

Lo más recomendable es dejar la codificación que presenta el sistema por defecto, tal y como está, es decir “ISO-8859-1”, para no tener problemas al presentar los caracteres.

Al finalizar el trabajo tanto con R, como con RStudio, hay que recordar que se debe de dar “SI”, a guardar la imagen de la sesión, para que el trabajo realizado quede guardado en el historial.

Ejercicio de Actividad:

- Como ejercicio de repaso de lo aprendido, se sugiere crear scripts de los códigos empleados hasta ahora en esta unidad, organizándolos según el tema y almacenarlos en una carpeta de datos.

Avance:

En la siguiente Unidad, se realizará una introducción a la estadística descriptiva e inferencial, utilizando todas las funciones y gráficos que se han contemplado en esta primera unidad. Para ello se hará uso indistintamente tanto de R como de RStudio, con mayor énfasis en este último.

- **Bibliografía:**

1. García Alfonso. (2008) Estadística aplicada con R. Madrid. Librería UNED
2. García Alfonso. (2008) Ejercicios de estadística aplicada. Madrid, Librería UNED
3. Guisande González Castor, Vaamonde Liste Antonio (2012) Gráficos estadísticos y mapas con R, Madrid, Ediciones Díaz de Santos
4. García Alfonso. (2014) La interpretación de los datos Una introducción a la estadística aplicada. Madrid, Librería UNED
5. Cabrero Yolanda, García Alfonso. (2015) Análisis estadístico de datos espaciales con QGis y R. Madrid, Librería UNED.

- **Enlaces de Interés.**

1.- Páginas Web de R

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

2.- Página Web de RStudio

<https://rstudio.com/products/rstudio/>

3.- Página Web de QGis

<https://www.qgis.org/en/site/index.html>

4.- Páginas Web para documentación QGis

https://docs.qgis.org/3.4/es/docs/user_manual/ (Manual de usuario)
https://docs.qgis.org/3.4/es/docs/training_manual/ (Manual de aprendizaje)
https://docs.qgis.org/3.4/es/docs/gentle_gis_introduction/ (Introducción a GIS)