SOURCE STAT LAB

Training, Reporting and Consultancy

CURSO:

Introducción al programa estadístico R

Programa de capacitación para estudiantes universitarios

Quito, Enero 2015

Índice

1. Introducción

El programa R es un software libre distribuido bajo GNU General Public License muy utilizado por la comunidad científica el cual es considerado un lenguaje de comandos de manipulación y análisis estadístico debido a la gran variedad de métodos estadísticos que cubre, así como a las capacidades gráficas que ofrece. R fue desarrollado en el año 1993 por los profesores del Departamento de Estadística **Ross Ihaka** y **Robert Gentleman** de la Universidad de Aukland, se distribuye gratuitamente desde 1995 a través del repositorio *Comprehensive R Archive Network* (CRAN) en `http://www.r-project.org` su mantenimiento se encuentra a cargo del grupo **R Core Team** desde el año 1997 asistido por una gran cantidad de colaboradores internacionales.

Figura 1: Logo del programa R

R es considerado la versión libre del programa comercial S-Plus, el cual fue desarrollado para AT&T Bell Laboratories por John M. Chambers en el año 1988, aunque son evidentes las diferencias entre R y S, la gran mayoría del código escrito para S funciona sin inconvenientes en R.

Dadas su características R tiene gran potencial para ser utilizado en diferentes áreas de la estadística, simulación, reportería dinámica, biomatemática, minería de datos, big data, etc. Al ser un software libre puede ser instalado en diversas plataformas y sistemas operativos: Windows, Linux, Mac OS X y Unix.

El mayor inconveniente para los nuevos usuarios es adaptarse a su interfaz gráfica que para muchos es *poco amigable*, en el sentido que se deben tipear las funciones predefinidas o a su vez programar las funciones nuevas a diferencia de programas clásicos como SPSS, STATA, etc. en los cuales se tienen botones o ventanas que despliegan opciones de análisis para el usuario.

Dada la popularidad que ha adquirido R en los últimos años se han desarrollado varias interfaces gráficas libres GUIS's (Grafical User Interface) con el fin de volver más amigable la interacción con el usuario. Entre las interfaces más populares y utilizadas por la comunidad R se encuentran:

- RStudio
- Rattle
- Deducer
- RKWard
- Eclipse
- Knime
- R Analytic Flow
- JGR
- ESS
- Red -R
- R Commander

R al ser un programa GNU General Public License carece de soporte técnico, sin embargo, en la actualidad existen empresas que proveen varios tipos de soportes para R bajo pago. Entre las más destacadas se encuentran:

- RStudio, Inc.
- Revolution Analytics, Inc.
- XL - Solutions Corporation.

1.1. Funcionamiento

R es un lenguaje orientado a objetos (OOP[1]) diseñado en un entorno auténtico bajo el cual esconde su simplicidad y flexibilidad, lo cual permite a sus usuarios añadir funcionalidad mediante la definición de nuevas *funciones*. El término *orientado a objetos* hace referencia a un paradigma de la programación que emplea objetos en sus interacciones y diseño de aplicaciones. R almacena sus variables, datos, funciones, resultados, etc., en la memoria activa del computador en forma de objetos con un nombre específico y pueden ser modificados o manipulados por el usuario mediante operadores y funciones. El uso y funcionamiento de los operadores en R es bastante intuitivo y se lo verá a detalle más adelante.

El hecho que R sea un lenguaje de programación puede desmotivar a muchos usuarios los cuales piensan que para iniciarse en el programa se necesita *alma de programador* lo cual no es cierto. Primero R es un lenguaje interpretado similar a Java, y segundo no es un lenguaje compilado a diferencia de C, C++, Fortran, Pascal, ect. sino más bien mediante comandos ingresados por teclado los cuales se ejecutan directamente sin necesidad de construir archivos ejecutables.

El programa R incluye 8 bibliotecas o paquetes estándar, sin embargo, las capacidades de R pueden ser ampliadas fácilmente mediante la incorporación de paquetes que se encuentran disponibles en varios repositorios como:

- CRAN
- BioConductor
- Omegahat
- RForge, entre otros.

Los paquetes estándar de R pueden ser visualizados a través del siguiente comando:

```
search()
```

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

En la actualidad[2] existen 6129 paquetes válidos en el repositorio CRAN, los cuáles se encuentran ordenados por fecha de publicación o alfabéticamente agrupadas en diversas líneas de investigación.

[1] La programación OOP está basada en varias técnicas: herencia, clasificación, identidad, polimorfismo y encapsulamiento.

[2] Información obtenida al 21 de Diciembre 2014.

1.2. Ventajas

Entre las principales ventajas que posee el software R podemos anotar lo siguiente:

- Al tratarse de un software libre el costo es nulo.
- Se han implementado una gran cantidad de métodos estadísticos desde los más básicos hasta los más avanzados y modernos. Todos los métodos se encuentran organizados en librerías, las cuales se encuentran en constante crecimiento.
- Capacidad para acceder a datos de múltiples formatos. En la actualidad existen varias librerias para leer datos desde SPSS, SAS, STATA, MySQL, Excel, etc.
- Gran capacidad para la manipulación de datos y funciones, así como para la generación de gráficos de alta calidad.
- Facilidad para enlazarse con LaTeX y generar reportes dinámicos.
- Amplia bibliografía tanto en internet como en libros publicados por prestigiosas editoriales como: Springer, Wiley, O'Reilly, Chapman & Hall/CRC, etc.
- Fácil visualización e interpretación de los algoritmos implementados en R con lo cual el usuario puede conocer exactamente lo que el ordenador ejecuta.
- Permite visualizar los algoritmos en él implementados, modificarlos y ajustarlos a nuestras necesidades (esto no es permitido en los softwares licenciados).

1.3. Desventajas

Los inconvenientes a los cuales se deben enfrentar los usuarios de R son:

- Al ser un programa libre carece de un departamento de atención al cliente al cual se pueda recurrir en caso de que se reporte un inconveniente con el mismo. Sin embargo existe una comunidad en crecimiento de usuarios de R que se encuentran dispuestos a colaborar desinteresadamente en la resolución de problemas.
- El software R como tal no dispone de una interfaz amigable para el usuario, las tareas se llevan a cabo a través de líneas de comando lo cual puede resultar difícil para el usuario común. No obstante con el desarrollo de GUI's se ha facilitado en gran medida la experiencia del programa con el usuario común.
- El código en R es interpretado, no compilado, lo cual puede ocasionar una ejecución lenta en ocasiones en las que se realizan simulaciones intensas. Con el fin de remediar lo anterior el grupo R Core Team a partir de la versión 2.14 ha precompilado todas las funciones y librerías de R con el objetivo de acelerar la ejecución.

2. Instalación y actualización

2.1. Programa R

La instalación del programa R se describe en los siguientes pasos:

1. Accedemos al repositorio de la CRAN a través de un navegador tecleando la dirección: `http://cran.es.r-project.org/`.

2. Nos dirigimos a la parte superior de la página, y nos ubicamos en `Download and install R` seguido elegimos el sistema operativo. Para nuestro caso hacemos click en `Windows`.

3. A continuación elegimos entre los 2 subdirectorios: `base` y `control`. Damos click sobre `base`.

4. Finalmente, en la pantalla damos click sobre `Download R X.Y.Z for Windows`, con lo cual descargamos la versión de R más reciente.

En la actualidad, el software R puede ser instalado incluso sobre la plataforma Android[3], esto permite que muchos usuarios puedan tener el programa al alcance y en todo momento con tan sólo tener un smartphone operativo.

2.2. Entorno de trabajo

Es muy importante para el usuario de R, conocer su entorno de trabajo, y es precisamente lo que trataremos en esta sección. Antes de iniciar, debemos aclarar que el presente documento está enfocado en la plataforma `Windows`, sin embargo en los diferentes sistemas operativos no existe mayor diferencia.

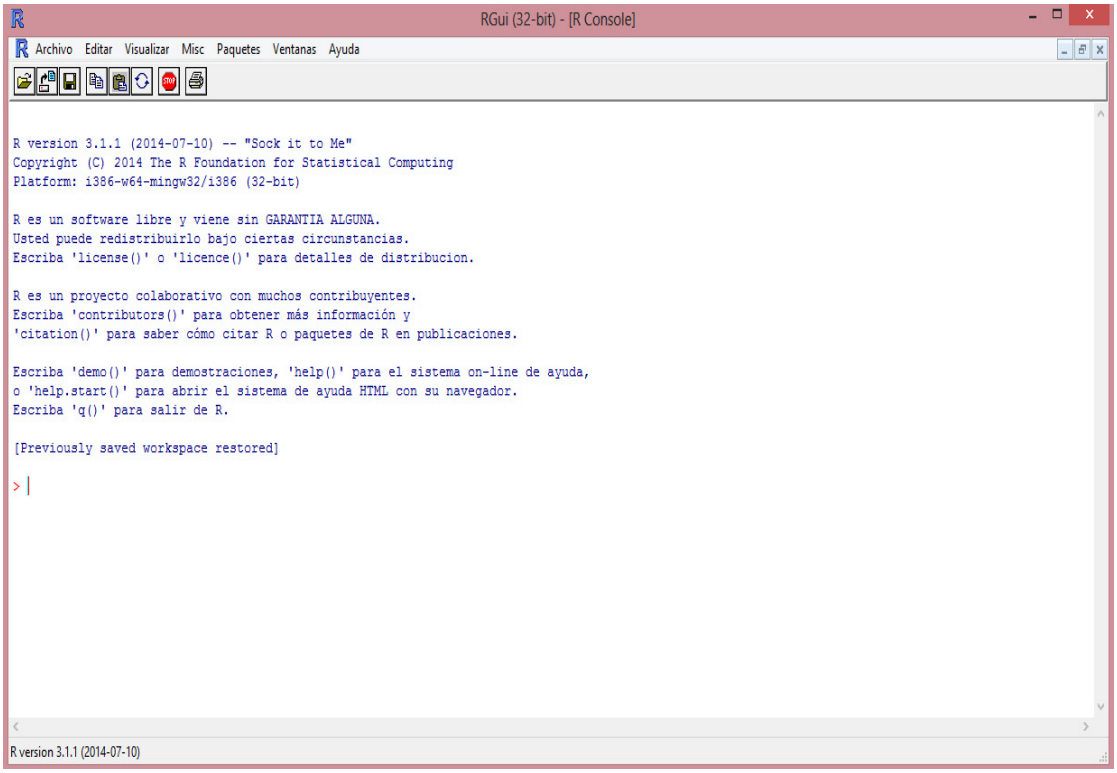

Figura 2: Entorno de trabajo de R

Iniciaremos explicando la barra de herramientas, la misma que consta de las secciones: Archivo, Editar, Visualizar, Misc, Paquetes, Ventanas y Ayuda. Todas serán explicadas con detalle a continuación:

[3] `https://play.google.com/store/apps/details?id=com.appsopensource.R&hl=es_419`

- **Archivo:** En esta sección se podrá tratar todo lo relacionado con el manejo de archivos y salir del programa. Los elementos que lo componen son los siguientes:

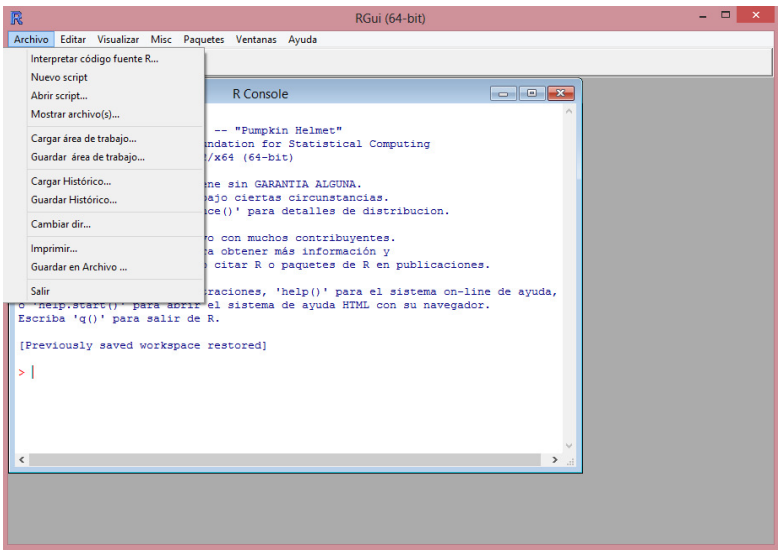

Figura 3: Sección Archivo

  - **Interpretar código fuente R:** Hay que recordar que R es un lenguaje de programación derivado del S, es por ello que podemos realizar programas en un editor externo, con esta opción podemos ejecutar dicho programa en la consola siempre y cuando, la extensión de guardado sea del tipo .R.
  - **Nuevo script:** Si lo ejecutamos, se nos abrirá un editor de lenguaje R para crear script que luegos podemos llamar desde la consola, el tipo de archivo que podemos generar son R o S.
  - **Abrir script:** Los script que hemos generados o que tengamos del tipo R o S podemos abrirlos en un editor para poder editarlos.
  - **Mostrar archivo(s):** Siver para abrir y poder visualizar o editar cualquier archivo, es interesante para los que tengan relación con el lenguaje R o S. Se diferencia con la opción Abrir script de que éste no puede editarlos directamente, simplemente visualizarlos. Se pueden abrir varios archivos a la vez.
  - **Cargar área de trabajo:** Pues como su nombre indica, sirve para cargar un área de trabajo que hayamos configurado y guardado previamente, es útil por ejemplo, cuando hemos definido el tipo de letra, el espacio de trabajo, los colores, etc, y queremos usarlo. La extensión común es .RData aunque también puede cargarse entornos de trabajo con formato antiguo tipo .rda.
  - **Guardar área de trabajo:** Cuando se haya configurado el entorno de consola de R, colores, tipo de letra, tamaño, etc, podemos guardarlo para cargarlo posteriormente en futuras aplicaciones ya que el propio programa R no guarda, de momento, dicha configuración. El formato del entorno de trabajo es .RData.
  - **Cargar Histórico:** Podemos cargar el archivo de comandos que se hayan ejecutado en una sesión previamente guardada. El formato de salida es .history.
  - **Guardar Histórico:** Con él, podemos guardar los comandos que hayamos instroducido por consola en una sesión. El formato de guardado es .history.

- **Cambiar dir:** Podemos configurar el directorio de trabajo que está definido por defecto cuando se instalo el programa.
- **Imprimir:** Podemos configurar e imprimir el entorno de trabajo de R, la consola.
- **Guardar en archivo:** Se guardará todo lo que se haya escrito por la consola en formato .txt que después podremos recuperar.
- **Salir:** Sirve para salir del programa R, antes nos preguntará si queremos guardar el área de trabajo.

- **Editar:** Es la sección encargada de editar la consola de R, así como de configurar el entorno de trabajo.

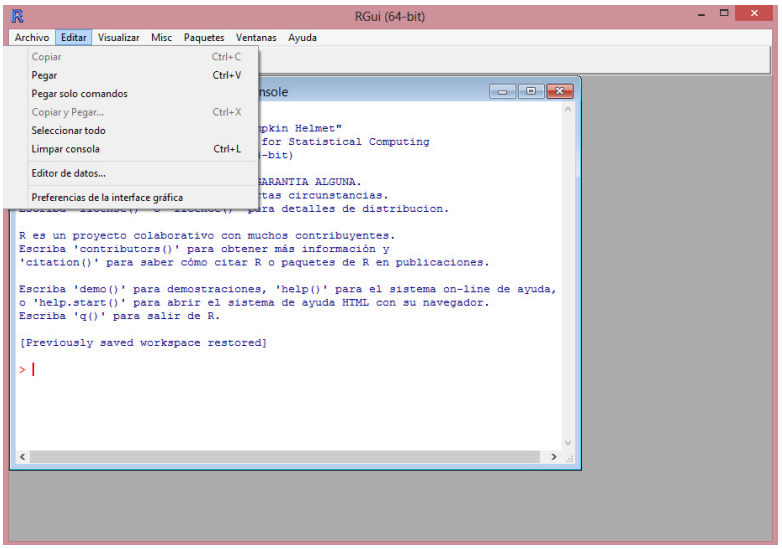

Figura 4: Sección Editar

- **Copiar:** Podemos copiar al portapapeles el comando o sentencia o lo que queramos de la consola. Para acceder de forma rápida por teclado se debe pulsar: Ctrl + C.
- **Pegar:** Podemos pegar en la consola aquello que tengamos en el portapapeles. Para acceder de forma rápida por teclado se debe pulsar: Ctrl + V.
- **Pegar solo comandos:** La diferencia básica con la opción Pegar es que con esta opción, sólo se pegarán en la consola aquello que sea comandos para ejecutarse.
- **Copiar y Pegar:** Todo aquello que copiemos, directamente se pegará, de forma inmediata, en la consola. Para acceder de forma rápida por teclado se debe pulsar: Ctrl + X.
- **Seleccionar todo:** Como su nombre indica, se selecciona todo lo que esté presente en la consola.
- **Limpiar consola:** Se borrarán todo lo que esté presente en la consola, pero ojo, no se borrarán las variables y estructuras definidas. Es útil cuando tenemos en la consola mucha información que ya no es útil. Para acceder de forma rápida por teclado se debe pulsar: Ctrl + L.

- **Editor de datos:** Podemos definir datos (vectores, estructuras, funciones, etc) y guardarlos para posteriormente llamarlos en la consola. Estos datos estarán definidos en la consola cuando se guarde.
- **Preferencias de la interfaz gráfica:** En esta opción es donde podremos configurar todo lo relacionado a la visualización de texto reflejado en la consola: tamaño de letra, colores, tipo de letra, etc. También, se puede configurar el entorno de trabajo, para multiventana o una única ventana.

- **Visualizar:** Activa la visualización de algunos iconos en la parte superior tales como: Abrir script, Cargar área de trabajo, etc., mismas que ya se detallaron anteriormente.

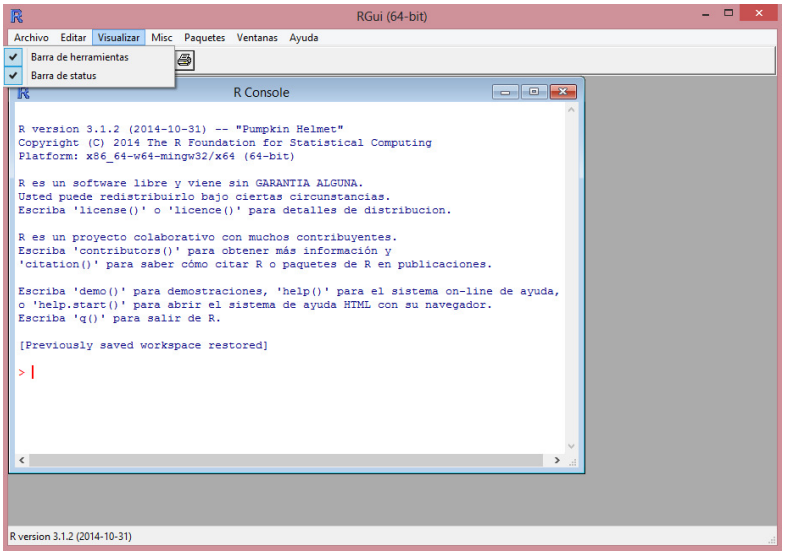

Figura 5: Sección Visualizar

- **Misc:** Esta es la sección denominada misceláneas, donde hay más de un control interesante.

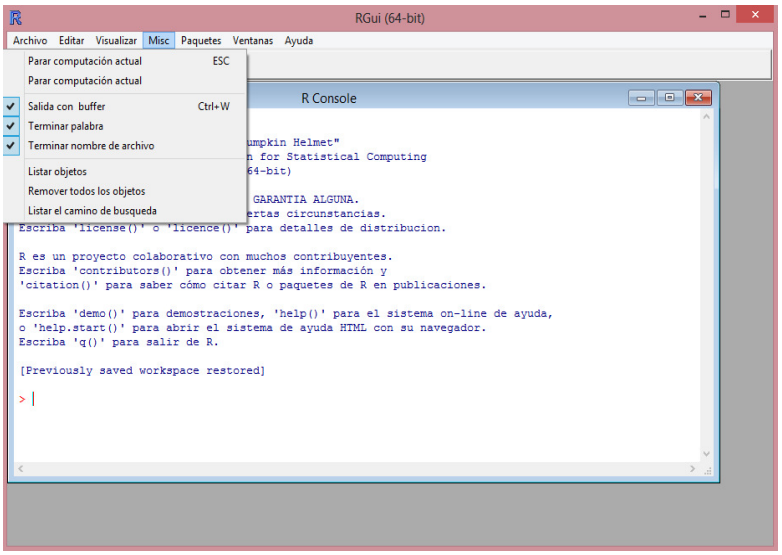

Figura 6: Sección Misc

- **Parar computación actual:** Este control es muy interesante y útil, con él, podremos parar cualquier archivo, sentencia o código que esté ejecutando la consola R. Para acceder de forma rápida por teclado se debe pulsar: ESC.
- **Salida con buffer:** Para acceder de forma rápida por teclado se debe pulsar: Ctrl + W.
- **Terminar palabra:** Es una ayuda interactiva, que nos ayuda a completar las palabras mientras estamos escribiendo en caso que la consola las reconózca.
- **Terminar nombre de archivo:** Realiza la misma función que la opción Terminar palabra pero en archivos.
- **Listar objetos:** Se nos mostrará por consola los objetos que hasta en este momento hemos definido en la consola.
- **Remover todos los objetos:** Como su nombre indica, elimina de memoria todos los objetos que hayamos definido (datos, variables, matrices, vectores, etc) en la consola de R. Cuando pulsemos sobre dicha opción, el programa, nos preguntará si realmente queremos borrarlos.
- **Listar el camino de búsqueda:** Nos ofrece por consola las librerias y complementos que tenemos instalados en R.

- **Paquetes:** Dedicado al manejo de las librerias que posee el programa R.

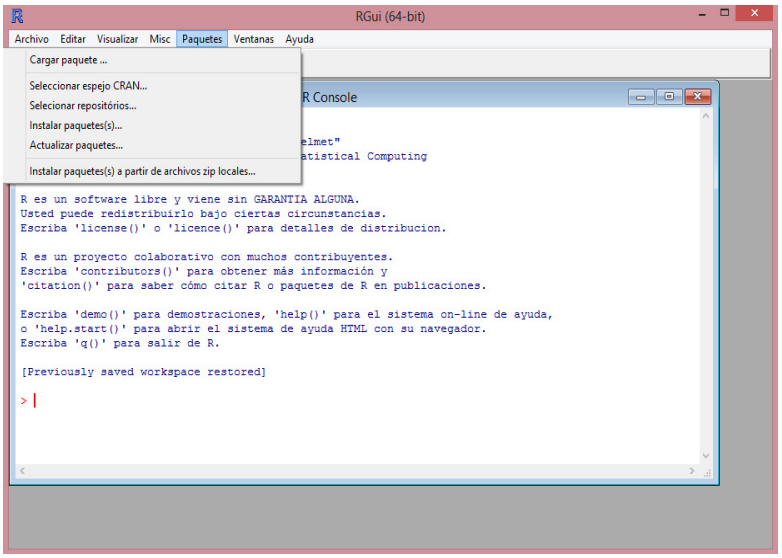

Figura 7: Sección Paquetes

- **Cargar paquete:** Con él, podemos cargar en la consola cualquier libreria que tengamos instalada.
- **Seleccionar espejo CRAN:** Sirve para configurar el servidor de librerias.
- **Seleccionar repositórios:** En consola nos mostrará los repositorios que tenemos instalados y nos pedirá cual usar para la sesión activa.
- **Instalar paquetes(s):** Podremos actualizar o instalar librerias nuevas en red, para ello, debemos elegir un servidor.

- **Actualizar paquetes(s):** Podemos actualizar las librerias que tengamos instaladas en caso de haber una actualización reciente por red, para ello, debemos seleccionar un servidor
- **Instalar paquetes(s) a partir de archivos zip locales:** En caso de haber descargado una libreria y tenerla en nuestro ordenador, podemos instalarlo mediante esta opción.

- **Ventanas:** Permite configurar la visualización de las ventanas: consola, área de gráficos y editor de script en modo vertical, horizontal, etc.

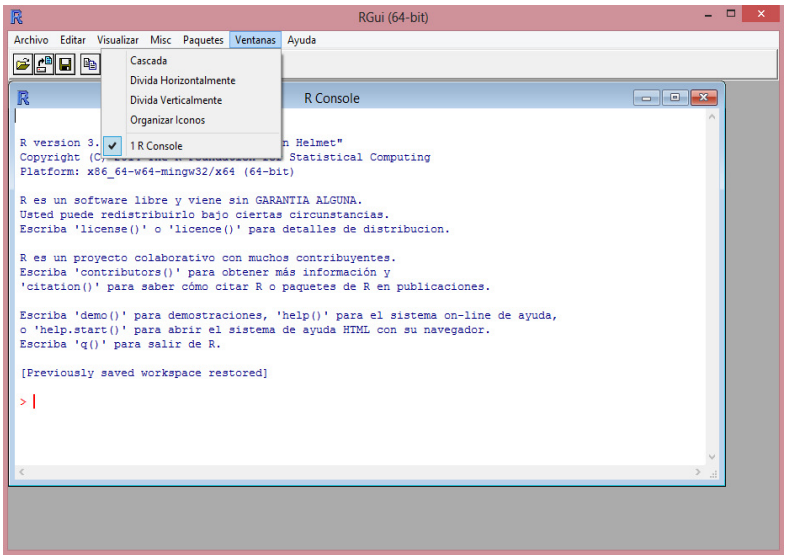

Figura 8: Sección Ventanas

- **Ayuda:** Es un apartado que debemos tenerlo siempre presente, ya que en él se dispone al usuario, los manuales de utilización y específicos de R.

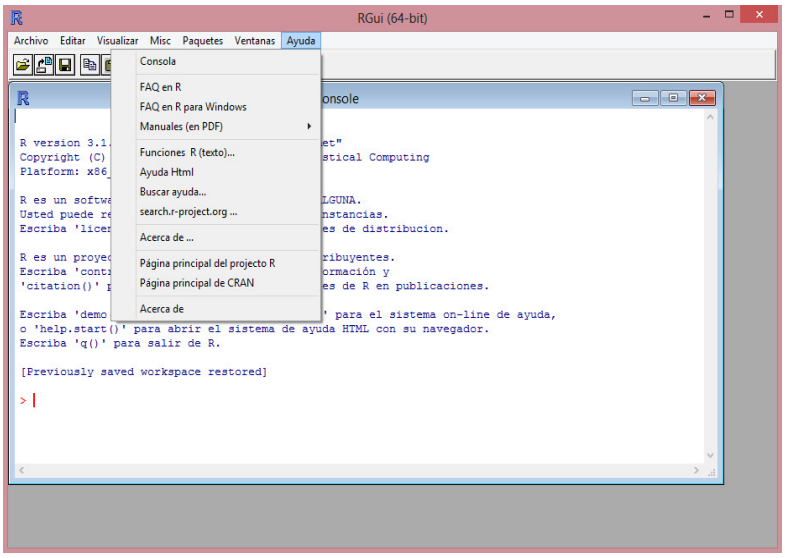

Figura 9: Sección Ayuda

- **Console:** Nos mostrará en una ventana los atajos por teclado que posee la consola.
- **FAQ en R:** Ayuda en html que se carga desde nuestro ordenador, donde nos ofrece las preguntas frecuentes que se suelen hacer los usuarios al utilizar R. No hace falta estar conectado a Internet.
- **FAQ en R para Windows:** Preguntas frecuentes para los usuarios de Windows.
- **Manuales en PDF:** Una recopilación de los manuales para el manejo del programa, esta opción es bastante interesante.
- **Funciones R(texto):** Es una opción donde podremos introducir sentencias de comando para obtener ayuda sobre ellas, es bastante útil.
- **Ayuda Html:** Se nos abrirá en el navegador una ayuda interactiva, no es necesario estar conectado a Internet.
- **Html search page:** Consiste en una página html donde podremos buscar las instrucciones que queramos consultar.
- **Página principal R:** Es un enlace directo a la web del proyecto R.
- **Página principal de CRAN:** Enlace directo al directorio de librerias de R.
- **Sobre:** Indica información de la compilación que se tenga instalada de R.

2.3. Instalación de paquetes

Como habíamos señalado en la sección anterior, una de las principales ventajas de R es su capacidad para incrementar su funcionalidad mediante la incorporación de nuevos paquetes o librerías.

Para entender de mejor manera la necesidad de emplear paquetes en R usaremos la siguiente metáfora: Imaginemos que tener instalado el programa en el computador es como haber adquirido un auto nuevo el cual cumple con ciertas funciones principales y no trae contratiempos, una vez que lo empezamos a rodar nos vamos dando cuenta de la necesidad de mejorar el rendimiento de ciertos mecanismos ya existentes del auto debido que lo vamos exponiendo a diferentes ambientes y condiciones. Lo anterior pasa exactamente con R, cuando descargamos e instalamos el sotfware contamos con ciertas características y paquetes básicos pero existen miles de paquetes adicionales que pueden ser agregados para lograr mejorar su funcionalidad y a la vez realizar cosas estupendas.

2.3.1. Repositorio CRAN

La instalación de paquetes en R desde el repositorio CRAN es sencillo, una vez conocido el nombre del paquete ha ser instalado basta con introducir el siguiente comando en la consola:

```
install.packages("nombre_paquete")
```

Ciertos paquetes de R requieren la instalación de otros paquetes adicionales debido que entre estos comparten algunas funciones (*paquetes sugeridos*) para evitar tener inconvenientes con la instalación de un paquete se aconseja adicionar el parametro siguiente:

```
install.packages("nombre_paquete", dependencies=TRUE)
```

En el caso que se desee instalar un lista de n paquetes planteamos la siguiente solución:

```
paquetes <- c("pckg_1","pckg_2", ... , "pckg_n")
lapply(seq_along(paquetes), function(i){
      install.packages(paquetes[[i]], dependencies=TRUE)
      })
```

Una segunda alternativa al momento de instalar los paquetes de R consiste en obtener el archivo zipeado (.zip) desde el repositorio CRAN e instalarlo manualmente a través del menú del programa.

Una vez instalado un paquete el paso siguiente consiste en cargar las funciones y datos del mismo dentro del área de trabajo, para esto tenemos dos comandos útiles: `library()` y `require()`, sin embargo lo más recomendable es utilizar el primero de ellos como se detalla a continuación:

```
library('translate2R')

## Error in library("translate2R") :
## there is no package called <U+2018>translate2R<U+2019>

require('translate2R')

## Loading required package: translate2R
## Warning message:
## In library(package, lib.loc=lib.loc, character.only=TRUE,
##          logical.return=TRUE,
##         : there is no package called <U+2018>translate2R<U+2019>
```

Para mostrar la diferencia entre de los comandos anteriores trataremos de cargar el paquete `translate2R`, el mismo que no ha sido descargado previamente. Al emplear `library` observamos que nos arroja un error, mientras que al emplear `require` únicamente tenemos una advertencia. En el caso puntual que nos encontremos trabajando en un script la segunda opción nos causaría serios problemas dado que permite continuar con la ejecución del proceso sin que el usuario se percate, mientras que la primera opción alerta al usuario. Por todo lo antes mencionado la gran mayoría de usuarios R prefieren usar el comando `library`.

Si deseamos visualizar el listado completo de los paquetes instalados podemos recurrir al comando `library()`. Además, si deseamos conocer el lugar en el cual se instalan los paquetes podemos hacer lo siguiente:

```
.libPaths()

## [1] "/Library/Frameworks/R.framework/Versions/3.1/Resources/library"
```

En algunas ocasiones los usuarios desean descargar los paquetes de R en una carpeta específica (C:/pckg_down) e instalar los paquetes en otra carpeta diferente (D:/pckg_inst), a continuación mostramos una posible solución al problema:

```
install.packages("pckgs", destdir="C:/pckg_down", lib="D:/pckg_inst")
library("pckgs",lib.loc="D:/pckg_inst")
```

Conocer la totalidad de paquetes que se encuentran albergados en la CRAN, y a su vez verificar cuales de ellos se encuentran válidos y disponibles para su uso es posible mediante los siguientes comandos:

```
installed.packages()
available.packages()
```

En el caso que se requiera conocer los paquetes que se encuentra cargados en el área de trabajo emplearemos el siguiente comando:

```
sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
##
## locale:
## [1] C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.8
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.5.5 formatR_1.0    highr_0.4      stringr_0.6.2
## [5] tools_3.1.2
```

El comando anterior imprime información acerca de la versión y plataforma del ejecutable de R que se encuentra utilizando, adicionalmente muestra información de la localidad y de los paquetes adjuntos o cargados.

2.3.2. Repositorios externos

Habiamos comentado sobre la existencia de varios repositorios adicionales al CRAN de entre los cuales destacan R-Forge[4], BioConductor[5] y Omegahat[6], dentro de estos repositorios se pueden encontrar paquetes que no se encuentran en el CRAN, o bien, versiones más actualizadas de los mismos.

[4]Sistema para el desarrollo y versionamiento de paquetes en R.
[5]Subproyecto dedicado a la investigación en bioestadística.
[6]Subproyecto dedicado al desarrollo de interfaces para la programación estadística basado en Java.

Para instalar los paquetes desde los repositorios externos antes mencionados basta la siguiente sentencia:

```
install.packages("pckgname", repos="http://r-forge.r-project.org")
install.packages("pckgname", repos="http://www.omegahat.org/R")
install.packages("pckgname", repos="http://www.bioconductor.org/
                                    packages/2.10/bioc")
```

Algunos de los paquetes se encuentran en formato binario, para su instalación basta adicionar el parámetro `type="source"`.

2.3.3. Github

GitHub es un repositorio de archivos y proyectos el cual emplea un sistema de control de versiones conocido como Git. Github fue lanzado a inicios del 2010, el código alojado en el repositorio se almacena de forma pública, aunque también se puede almacenar de forma privada siempre y cuando se tenga una cuenta de pago.

La instalación y sincronización de Github con RStudio se realiza de la siguiente manera:

1. Descargar e instalar la plataforma específica de Git[7] (no Github) con las opciones por default.

2. Configurar Git con los comandos globales a través de la versión bash, teclear lo siguiente:

   git config —global user.name ”nombre de la cuenta de Github”
   git config —global user.email ”Github-Email@something.com”

3. Abrir RStudio y configurar la ruta del ejecutable de Git. Ir a Tools >Options >Git/SVN.

En la actualidad gran cantidad de usuarios R han alojado sus proyectos en el sistema de control de versiones GitHub, Inc. por lo cual se hace necesario contar con una solución al momento de instalar algún paquete alojado en dicho sistema. Nuestra solución consiste en lo siguiente:

```
install.packages('devtools', dependencies=TRUE)
devtools::install_github("rstudio/rmarkdown")
```

Iniciamos instalando el paquete `devtools` el cual nos provee la función `install_github`, esta última es la encargada de acceder al sistema Github e instalar el paquete deseado, para esta ocasión hemos seleccionado el paquete `rmarkdown` alojado en el repositorio Github dentro del proyecto `rstudio`.

En el caso que se requiera información acerca de un paquete previamente instalado podemos recurrir a:

[7] http://www.git-scm.com/

```
library(help='pckgname')
```

También podemos enlistar todas las funciones que han sido implementadas dentro de un paquete como muestra el siguiente ejemplo:

```
library(foreign)
ls('package:foreign')

##  [1] "data.restore"  "lookup.xport"  "read.S"        "read.arff"
##  [5] "read.dbf"      "read.dta"      "read.epiinfo"  "read.mtp"
##  [9] "read.octave"   "read.spss"     "read.ssd"      "read.systat"
## [13] "read.xport"    "write.arff"    "write.dbf"     "write.dta"
## [17] "write.foreign"
```

Si deseamos conocer los parámetros de cada una de las funciones implementadas dentro de un paquete podemos recurrir a lo siguiente:

```
lsf.str('package:foreign')
```

Cuando un paquete es empleado dentro de una investigación surge la necesidad de citar a los autores del paquete, lo anterior lo solventamos mediante el comando `citation`.

```
citation("foreign")

## To cite package <U+2018>foreign<U+2019> in publications use:
##
##  R Core Team (2014). foreign: Read Data Stored by Minitab, S, SAS,
##  SPSS, Stata, Systat, Weka, dBase, .... R
##  package version 0.8-61. http://CRAN.R-project.org/package=foreign
##
## A BibTeX entry for LaTeX users is
##
##  @Manual{,
##    title = {foreign: Read Data Stored by Minitab, S, SAS, SPSS,
##  Stata, Systat, Weka,dBase, ...},
##    author = {{R Core Team}},
##    year = {2014},
##    note = {R package version 0.8-61},
##    url = {http://CRAN.R-project.org/package=foreign},
##  }
```

2.4. Actualización de paquetes

En el caso que se desee actualizar todos los paquetes previamente instalados contamos con el comando `update.packages`, el mismo que revisa las actualizaciones de los paquetes. El anterior comando nos preguntará si deseamos actualizar uno por uno los paquetes, lo cual es tedioso pues si contamos con un gran número de paquetes nos demandará bastante tiempo aprobar la actualización de los mismos. Para subsanar lo anterior basta añadir el parámetro `ask = TRUE`.

```
update.packages(ask=TRUE)
```

En el caso que se desee conocer únicamente los paquetes que fueron instalados previamente y en la actualidad constan con un versionamiento emplearemos lo siguiente:

```
old.packages()
```

El comando anterior nos mostrará la versión del paquete que ha sido instalada y la última versión disponible del mismo.

Para finalizar la sección dejamos un comando útil para los desarrolladores que deseen conocer los autores de los paquetes, así como también las personas que dan mantenimiento a los mismos:

```
lapply(dir(R.home("library")), packageDescription)
```

2.5. Actualización de R

Mantener actualizado R y sus paquetes es relativamente fácil a través del tecleo de los siguientes comandos en la consola, los mismos que se detallan a continuación:

```
install.packages('installr', dependencies = TRUE)
library('installr')
updateR()
```

Iniciamos instalando el paquete **installr**, acto seguido procedemos a cargar el paquete a través del comando `library`, y finalmente, mediante el comando `updateR` verificamos la versión más reciente del programa y descargamos la misma.

En el caso que se desee conocer la versión de R que se encuentra instalada podemos teclear en la consola el comando: `R.version.string`.

2.6. Entornos de desarrollo

Un entorno de desarrollo integrado, también conocido como IDE (Integrated Development Enviroment) es un programa informático compuesto por un conjunto de herramientas de programación que contiene: un editor, un compilador, un depurador y un constructor de interfaz gráfica (GUI), el mismo que viene empaquetado como un programa `aplicación` que facilita de sobre manera la realización de operaciones al usuario mediante una serie de menús o mediante interacción con los objetos gráficos que aparecen en pantalla, a través de periféricos como: el ratón y teclado.

Los IDE's fueron desarrollados con el fin de proveer al usuario un marco de trabajo más amigable, a continuación, elistamos algunos IDE's existentes para R:

- ESS
- Eclipse
- SciViews
- JGR
- Tinn-R
- Notepad ++
- RGui

2.6.1. RStudio

RStudio[8] es un entorno IDE de código abierto lanzado en Febrero 2011 el cual nos ofrece un marco de trabajo más amigable con el software R y lo podemos encontrar para todas las plataformas (Windows, Mac, Linux) así como también puede ser ejecutado a través de un navegador web[9].

RStudio ofrece una amplia integración con ficheros de diversos formatos: R scripts (.R), Markdown (.md), LaTeX (.Rnw) entre otros. La facilidad en la generación de documentos dinámicos con `RStudio` y `knitr` han hecho que el programa se convierta en la IDE preferida por muchos usuarios de R.

El programa se encuentra organizado en cuatro ventanas de trabajo distintas:

- **Editor de código fuente:** Se encuentra en la zona superior izquierda, esta ventana nos permite abrir y editar ficheros con código R.
- **Consola:** Se ubica en la zona inferior izquierda, esta ventana es también conocida como consola y nos permite ejecutar comandos de R.
- **Navegador de objetos:** La zona superior derecha posee dos ventanas auxiliares:
  - **Workspace:** En esta ventana se enlistan todos los objetos creados en memoria.
  - **History:** En esta ventana se almacena el histórico de las líneas de código que han sido ejecutadas en R.
- **Visualización e información:** Esta última ventana ubicada en la zona inferior derecha se encuentra conformada por 4 ventanas auxiliares:
  - **Files:** Provee el acceso al árbol de directorios y ficheros del disco duro.
  - **Plots:** Ventana auxiliar en la cual aparecen los gráficos creados en la consola.
  - **Packages:** Esta ventana facilita la administración de los paquetes de R instalados en el computador.
  - **Help:** Esta última ventana nos ayuda en la búsqueda de información respecto a un comando en específico.

RStudio ofrece varios mecanismos para controlar varios aspectos de la evaluación durante una sesión. La función `options( )` es empleada para compartir los valores de parámetros entre las funciones.

[8] El programa RStudio puede obtenerse libremente de la página `http://www.rstudio.org/`
[9] Opción válida para la versión `server`.

Ancho de impresión Existen ocasiones en las cuales el usuario desea controlar el ancho de impresión de los resultados que se muestran en la pantalla, como primer paso para modificar el ancho de impresión debemos obtener el parámetro actual mediante:

```
getOption("width")

## [1] 75
```

Una vez conocido el ancho de pantalla actual procedemos a modificar el mismo cambiando el valor del parámetro `width`, de la siguiente manera:

```
options(width=40)
rnorm(10)

##  [1]  0.22201594 -1.30582687 -0.77187981
##  [4]  0.16597540  1.07228518 -1.36746863
##  [7]  1.05388690 -0.07486345  0.51808029
## [10]  0.58216820
```

```
options(width=55)
rnorm(10)

##  [1]  0.8275274  0.8909401  0.3260686  0.2957942
##  [5]  1.8104020 -1.4769572 -0.6750092 -2.3949824
##  [9] -0.8280666 -1.5499244
```

```
options(width=70)
rnorm(10)

##  [1] -0.5333324  0.1018834 -0.8854220  1.4210510  0.7199195  0.3791893
##  [7]  0.6390262  1.6086690 -0.3436589  0.4993182
```

Prompt Para los usuarios que deseen cambiar el simbolo del prompt $>$ por otro símbolo diferente como: $->$ o por su nombre, tenemos el siguiente código:

```
options(prompt="->")
options(prompt="diego >")
```

Decimales Una preocupación adicional para los usuarios es la cantidad de decimales con la cual se muestran los resultados, dicha cantidad de decimales puede ser modificada y debe encontrarse en el rango de 1 a 22.

```
getOption("digits")

## [1] 7
```

R por default muestran los resultados con 7 decimales, sin embargo los mismos pueden ser modificados como se muestra a continuación:

```
options(digits=2)
rnorm(3)

## [1] -0.3623  0.0086 -1.0726
```

```
options(digits=5)
rnorm(3)

## [1]  0.014525 -0.474048 -1.257198
```

```
options(digits=10)
rnorm(3)

## [1] 2.1662685286 0.4110596076 0.8094248309
```

Existen opciones adicionales que pueden ser modificadas de acuerdo a las necesidades que tenga el usuario, para ver el listado completo de opciones podemos teclear en la consola el comando:

```
help(options)
```

2.6.2. R Analytic Flow

2.7. Obteniendo ayuda

Existen algunas funciones en R que nos facilitan la vida al momento de indagar sobre la funcionalidad de ciertos comandos con tan sólo tener disponible una conexión de Internet. Por ejemplo, si nos encontramos interesados en obtener información acerca de la función `seq()` podemos escribir en la consola lo siguiente:

```
help(seq)
```

Una forma alternativa es

```
?seq
```

Para el caso que se encuentren trabajando con funciones especificadas por caracteres especiales, el argumento debería ir entre comillas, con el fin de transformarlo en una *cadena de caracteres*:

```
help("[[")
```

Se pueden utilizar tanto comillas simples como dobles sin que esto conlleve problemas a posteriori.

Adicionalmente, si nos encontramos interesados en buscar información relacionada al término "normal", o algún otro término en específico podemos teclear lo siguiente:

```
help.search("normal")
help.search("termino_especifico")
```

En el caso que se requiera un manual de ayuda completo en formato HMTL, empleamos el comando:

```
help.start()
```

Por último, existen circunstancias en las cuales se hace indispensable conocer todas las funciones relacionadas a un nombre en específico, en estos casos es de gran utilidad el comando:

```
apropos("mean")

##  [1] ".colMeans"     ".rowMeans"     "colMeans"      "kmeans"
##  [5] "mean"          "mean.Date"     "mean.POSIXct"  "mean.POSIXlt"
##  [9] "mean.default"  "mean.difftime" "rowMeans"      "weighted.mean"
```

En el ejemplo anterior se muestran todas las funciones relacionadas con el término **mean**.

3. Estructura de datos

En las secciones previas hemos mencionado que R es un lenguaje de programación orientada a objetos, por lo que cualquier cosa que exista en R (variables, datos, funciones, etc.) es un objeto. Para conocer los nombres de los objetos presentes en el área de trabajo utilizamos el siguiente comando:

```
ls()

## character(0)
```

Ahora si creamos los vectores xvec y yvec, evidenciamos que dichos objetos ya aparecen en nuestra área de trabajo.

```
xvec <- c(1,2,3)
yvec <- c(4,5,6)
ls()

## [1] "xvec" "yvec"
```

Los nombres que se les asigna a los objetos en R pueden ser de cualquier longitud y éstos pueden combinar letras, números y caracteres especiales (coma, punto, guión bajo, etc.),

la única exigencia al momento de asignar un nombre a un objeto es que el mismo inicie con una letra.
La construcción explícita de un objeto nos proporcionará un mejor entendimiento de su estructura, y nos permitirá ahondar en algunas nociones mencionadas previamente.

Las estructuras de datos en R pueden ser organizados por su dimensionalidad (1 dimensión, 2 dimensiones o n-dimensiones), así como también por su tipo (homogéneo, heterogéneo) lo anterior da lugar a 5 tipos de estructuras que se resumen a continuación:

Dimensión	**1-d**	**2-d**	**n-d**
Homogéneo	Vector	Matriz	Array
Heterogéneo	Lista	Data Frame	

Cuadro 1: Estructura de datos

3.1. Vectores

Es la estructura más simple de R que sirven para almacenar un conjunto de valores del mismo tipo llamados *elementos* y consta de dos parámetros o argumentos: `mode`, `length`. El primero de los argumentos especifica el tipo de elementos que serán almacenados de entre los cuales destacan los tipos numéricos, booleanos y caracteres. El segundo argumento especifica la longitud del vector.

3.2. Factores

Son variables en R que tienen un número limitado de valores diferentes, dichas variables se conocen como variables categóricas a menudo; por ejemplo: el conjunto de observaciones acerca del color de ojos de un grupo de personas.

```
ojos <- c("negro", "azul", "negro", "verde", "verde", "caf<U+00E9>", "negro")
table(ojos)

## ojos
##        azul caf<U+00E9>       negro       verde
##           1           1           3           2
```

3.2.1. Uso de factores

El uso más importante de los factores se encuentra en el modelamiento estadístico; dado que ciertas variables categóricas ingresan a los modelos de manera diferente a las variables continuas.
Los factores en R se almacenan como un vector de valores enteros con un conjunto correspondiente de valores de caracteres para usar cuando aparezca el factor.

3.3. Matrices

Colección de datos a los que se accede por varios índices enteros (dimensiones).

3.4. Listas

Colección ordenada de objetos, en la que los elementos pueden ser de distinto tipo.

3.5. Arrays

Un arreglo (array) de datos es un objeto que puede ser concebido como una matriz multidimensional (hasta 8 dimensiones). Una ventaja de este tipo de objeto es que sigue las reglas que hemos descrito para las matrices. La sintaxis para definir un arreglo es

```
array(data, dim)
```

Las componentes data y dim deben presentarse como una sola expresión, por ejemplo

```
c(2,4,6,8,10)
c(2,3,2)
x <- array (1:24, c(3,4,2))
```

produce un arreglo tridimensional: la primera dimensión tiene tres niveles, la segunda tiene cuatro y la tercera tiene dos. Al imprimir el arreglo R comienza con la dimensión mayor y va bajando hacia la dimensión menor, imprimiendo matrices bidimensionales en cada etapa.

3.6. Data Frames

Tipo particular de listas de gran utilidad para el trabajo estadístico.

4. Funciones

Objetos que pueden ser creados por el usuario y reutilizados para realizar operaciones específicas.