TRABAJO DE GRADO

PROGRAMACIÓN DE COMPUTADORES ORIENTADA AL ANÁLISIS Y LA VISUALIZACIÓN DE DATOS, COMO FORMACIÓN BÁSICA DE ESTUDIANTES DE INGENIERÍA.

Presentado Por:

Daniel F. Quintero Munevar

Universidad Tecnológica de Pereira

Ingeniería de Sistemas y Computación

Julio 15, 2021

TRABAJO DE GRADO

PROGRAMACIÓN DE COMPUTADORES ORIENTADA AL ANÁLISIS Y LA VISUALIZACIÓN DE DATOS, COMO FORMACIÓN BÁSICA DE ESTUDIANTES DE INGENIERÍA.

Presentado Por:

Daniel F. Quintero Munevar

Trabajo de Grado Para Obtener El Título De:

Ingeniero en Sistemas y Computación

Director:

Ing. Juan de Jesús Veloza Mora

Universidad Tecnológica de Pereira

Ingeniería de Sistemas y Computación

Julio 15, 2021

NOTA DE ACEPTACION

_____

_____

_____

_____

Ing. Juan de Jesús Veloza Mora

_____

Agradecimientos

Agradezco a mi madre, padre, hermana, Abuela, tío Jorge y demás familiares y amigos, por el apoyo y acompañamiento que me brindaron en este proceso de formación profesional, y por ser mi motivación para seguir adelante.

Agradezco también a mi director de proyecto Juan de Jesús Veloza, quien me acompaño, apoyo y guio a lo largo de este proceso, con toda su sabiduría, tiempo y esmero, generándome una mentalidad investigativa, logrando llegar con éxito a esta meta.

Y por último, agradezco a todas las personas, que de alguna manera, ya sea directa o indirectamente aportaron algo para lograr esta meta.

Resumen

En la actualidad, estamos en una época donde la información ha sido explorada muy poco y en donde la big-data y el análisis de datos, se tornan como proceso de alta demanda por parte de diferentes campos del saber profesional. Para esto, la lectura y manipulación de estos datos, se tornan en procesos complejos y de muchos recursos si no se tiene conocimiento de alguna herramienta que permita ejercer esta habilidad.

Es así que se pretende la creación de esta guía orientada al análisis y visualización de datos para estudiantes de ingenierías, buscando como finalidad transmitir un conocimiento fundamental al lector, sobre el lenguaje de programación R, donde con este, podamos generar cálculos estadísticos, crear gráficos y una mejor y cómoda lectura de múltiples datos, ofreciendo así un nuevo enfoque hacia el ámbito estadístico y obteniendo un mejor razonamiento para brindar soluciones a diferentes problemas de la vida cotidiana.

Palabras Claves:

Información, big-data, análisis, conocimiento, visualización, programación, razonamiento.

Abstract

At present, we are in a time where the information has been explored very little, where big data and data analysis become high demand processes by different fields of professional knowledge. For this, the reading and manipulation of this data become complex and resource intensive proccesses if there's no knowledge of any tool that allows to exercise this ability.

It's in this way we pretend the creation of this guide oriented to the analysis and visualization of data for engineering students that is intended to transmit a fundamental knowledge to the reader, about the R programming language, where with this we can generate statistical calculations, create graphs and a better and comfortable reading of multiple data, offering a new approach to the statistical field and obtaining a better reasoning to provide solutions to different problems of everyday life.

Keywords:

Information, big-data, analysis, knowledge, visualization, programming, reasoning.

Contenido

Capítulo I: Formulación Del Problema.

1.1 Introducción.

En la actualidad estamos en una época donde pretendemos replantear la formación básica para una persona que estudie alguna profesión de ingeniería, la cual esté acorde con las necesidades del entorno en que vive y se desenvuelve y la manera en que debe enfrentar los problemas que surgen de dicho entorno.

Sin duda alguna la analítica y visualización para los datos de alto volumen es considerada actualmente una competencia básica transversal en toda formación profesional en ingeniería.

1.2 Descripción del Problema.

Hay mucha información hoy en el mundo, y ha sido explorada muy poco, es esta, la necesidad que en la actualidad nos debe mover a buscar las formas de enfrentar estos nuevos retos. En esta búsqueda de fomentar nuevo conocimiento a los estudiantes de las carreras de Ingeniería, se tiene un requerimiento fundamental de implementar un nuevo enfoque hacia el ámbito estadístico, puesto que fortaleciendo esta habilidad matemática se abren caminos hacia otros campos del saber profesional.

La programación de computadores orientada a la visualización y analítica de datos es hoy en día imprescindible como conocimiento y habilidad que todo profesional debe tener en su desempeño diario. Los análisis exploratorios, predictivos, descriptivos, entre otros que se realizan sobre datos de altos volúmenes pueden ocasionar mejores soluciones en diferentes problemas de la vida cotidiana.

1.3 Justificación.

Esta guía práctica nace por la necesidad de la enseñanza de la programación de computadores, orientada a la visualización y analítica de datos como formación básica de estudiantes en las carreras profesionales de Ingeniería.

1.4 Objetivos.

1.4.1 Objetivo General.

Diseñar y elaborar una guía práctica para programación de computadores orientada al análisis y visualización de datos, como formación básica de estudiantes de ingenierías.

1.4.2 Objetivos Específicos.

- Investigar y recopilar información sobre cómo debe ser un curso de programación de computadores orientado al análisis y la visualización de datos, como formación básica en estudiantes de ingeniería.
- Diseñar y elaborar la guía práctica inicial para programación de computadores orientada al análisis y visualización de datos, con ejercicios incluidos.
- Realizar una socialización con estudiantes de básicos de ingeniería de distintos programas en distintas universidades, para recibir retroalimentación aplicable a la guía.
- Realizar y elaborar guía definitiva.

1.5 Alcance del proyecto.

Se pretende realizar un abordaje simple a través de la realización de laboratorios con ejercicios prácticos, que lleven al estudiante a comprender, desde la programación, un aprendizaje fundamental en analítica y visualización de datos en alto volumen. De este modo se plantea la

necesidad de diseñar y elaborar un material didáctico (Guía) que estimule, oriente y refuerce la habilidad de programación y las competencias básicas iniciales en los entornos Big-data.

La realización de esta guía resume pasos importantes en definiciones y listas de actividades a modo de laboratorio para que el estudiante adquiera las competencias pretendidas como formación básica en programación de computadores orientada al análisis y la visualización de datos en entornos big-data. Esta guía es destinada a todo estudiante de ingeniería y pretende generar así un mejor perfil profesional.

Capitulo II: Marco Teórico.

2.1 Introducción.

2.1.1 ¿Qué es R?

R es un entorno y leguaje de programación utilizado para crear procedimientos gráficos y estadísticos de alta complejidad, este lenguaje suministra un aprendizaje relativamente sencillo con una extensa variedad de recursos estadísticos y gráficos, además ofrece un lenguaje de programación bien desarrollado y completo.

Decir que R es un lenguaje de programación, puede desalentar a muchos que piensan que no tienen "espíritu de programadores", pero con R no se debe tener este pensamiento por dos simples razones:

- R es un lenguaje interpretado (como JAVA) y no compilado (como C, C++, etc.) lo cual significa que los comandos escritos por teclado son ejecutados directamente sin necesidad de construir ejecutables.
- R tiene una sintaxis muy simple e intuitiva, por ejemplo, una función para que R la lea así, tiene que estar acompañada siempre de paréntesis (ejemplo: ls( )), ya que si se escribe sin esto, al momento de ejecutar lo tomara esto como un contenido del código.

R es un lenguaje orientado a objetos, lo que quiere decir que las funciones, variables, datos, resultados, etc., se almacenan en la memoria activa del computador en forma de objetos con un nombre en propio. El programador puede manipular o cambiar estos objetos con operadores, ya sea de tipo lógico, aritmético o comparativo.

- La utilización de R tiene grandes ventajas, que son:
- Es software libre y podemos modificarlo a nuestro antojo.

- Es gratis.
- Existe una gran comunidad, esto quiere decir que podemos encontrar a muchos usuarios trabajando con R y podemos buscar ayuda rápida en la red.
- Es actualizado con gran frecuencia.
- Funciona en diferentes Sistemas Operativos (Windows, Mac OS, Unix, etc.).
- Es sumamente potente y versátil, gracias a los packages.
- Permite elaborar gráficos de gran calidad.
- Se esta convirtiendo en un estándar en la comunidad científica ya que muchos desarrollos en análisis de datos se convierten en packages de R.
- Utiliza líneas de comando y no un interfaz gráfico.

Tenemos que tener precaución con algunos inconvenientes que nos puede presentar R, que son:

- La sintaxis de R es relativamente exigente, por ejemplo, se debe tener cuidado con el uso de las mayúsculas y las minúsculas ya que R no va a entender lo mismo si le decimos 'HOLA' a 'hola' él lo interpretara como objetos distintos.
- R no da muchas pistas acerca de que puede estar fallando.
- Algunos de los packages que tiene R no han sido muy probados.

"*En R es muy importante mantener un orden*"… ser organizado en nuestros proyectos al momento de utilizar R es muy importante ya que este lenguaje arroja mucho output (resultados) y es fundamental ser capaces de administrar esto en nuestro dispositivo de almacenamiento, por ende, es recomendable emplear una carpeta para cada proyecto de análisis de datos. Un ejemplo de estructura propuesta es la siguiente:

- Datos: materia prima en la que vamos a trabajar.

- Scripts: códigos de R que se van a utilizar.
- Imágenes: fotografías que de lo que en un momento dado este activo en R.
- Historias: es un conjunto de todos los comandos que se han utilizado.
- Gráficos: gráficos estadísticos para observar datos cuantitativos
- Anotaciones: comentarios para saber qué es lo que estamos haciendo en nuestro proyecto.

2.1.2 ¿Cómo Instalar R?

Para llevar a cabo la instalación de R debemos visitar la pagina principal del CRAN (Red Completa de Archivos R o por sus siglas en ingles Comprehensive R Archive Network) que encontramos en el siguiente enlace https://www.cran.r-project.org/

Una vez estemos en esta página, nos aparecerá algo similar a la siguiente imagen, allí encontrará los enlaces de instalación para diferentes sistemas operativos: Linux, Mac y Windows.

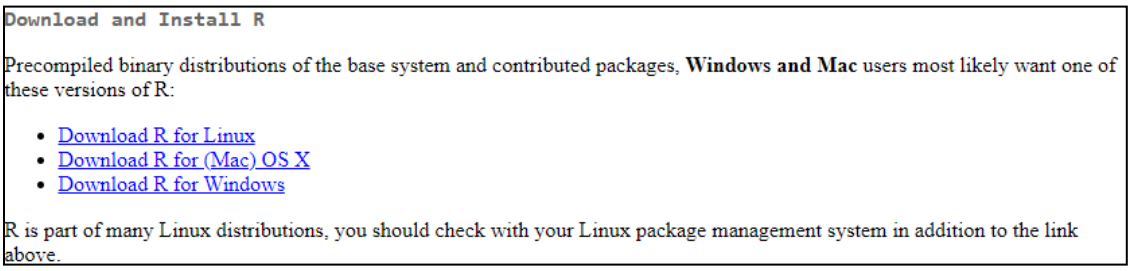


Imagen 1

Supongamos que deseamos instalar R en Windows, para esto damos clic en donde dice ***Download R for Windows***. Una vez de haber dado clic se abrirá una nueva pagina igual a la siguiente imagen, asi abriendo una pagina donde daremos clic en el enlace con el nombre de ***base*** o ***install R for the first time***.

Imagen 2

Dado clic en ese link, se abrirá otra pagina con un encabezado similar al de la siguiente imagen, al momento de la realizar esta guía se tiene que la última versión disponible de R es la 4.0.5 pero como este lenguaje es de constantes cambios, usted tendrá la versión mas actualizada. Una vez nos encontremos en esa página, daremos clic en **Download R 4.0.5 for Windows**. Luego de esto se empieza a descargar el instalador de R en el computador el cual deberá ser instalado con las opciones que vienen por defecto.

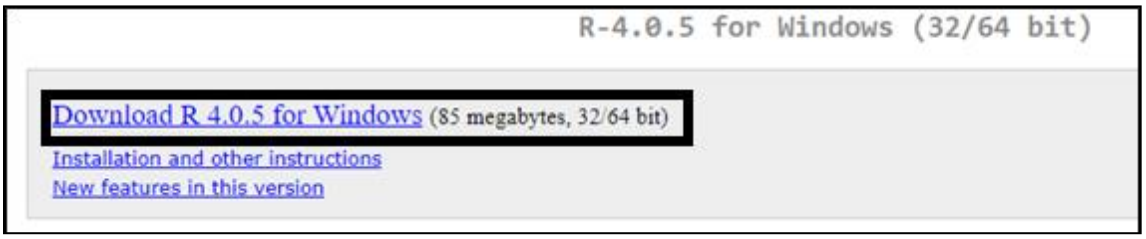


Imagen 3

2.1.3 Como Se Ve R.

Una vez realicemos la instalación de nuestro lenguaje R, podremos acceder a el por medio de la lista de programas o por el acceso directo que nos aparece en el escritorio, R tiene el siguiente logo, mismo que aparecerá como acceso directo en su escritorio:

Imagen 4

Al abrir nuestro lenguaje de programación aparecerá una interfaz algo similar a la siguiente figura; en esta ventana aparecerá unas subventanas, generalmente aparece en la parte izquierda lo

que conoceremos como consola, y es donde ingresaremos el código, instrucciones, comandos, etc. y la ventana de la derecha es donde aparecerán los gráficos, los datos almacenados o el ejecutable del código creado.

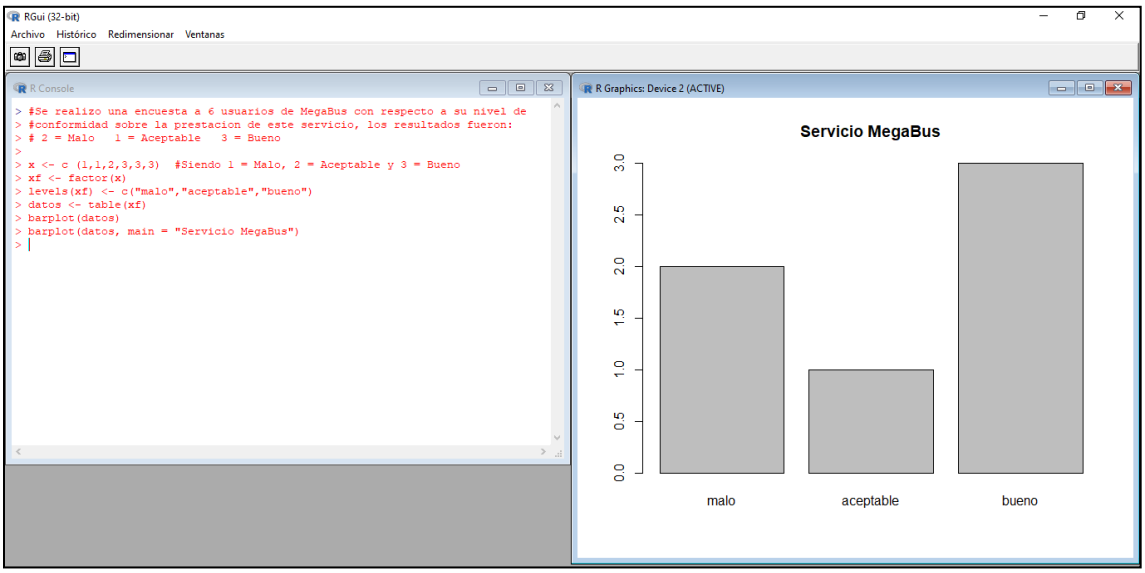


Imagen 5

2.2 Conceptos Básicos.

Para poder realizar actividades con R necesitamos adquirir un poco de conocimiento del léxico usado en este lenguaje. En este capítulo hablaremos sobre los conceptos básicos que emplearemos en este instructivo.

2.2.1 R Como Calculadora.

Como ya lo vimos en el titulo anterior, una vez ejecutado R nos aparecerá la interfaz de esta, tendremos a la vista el símbolo del sistema (>) en su color rojo, esto nos mostrará que R está preparado para admitir nuestros comandos. Por ejemplo, si digitamos "**3 + 5**" y oprimiendo "**ENTER**", R nos arrojaría el resultado de esta operación.

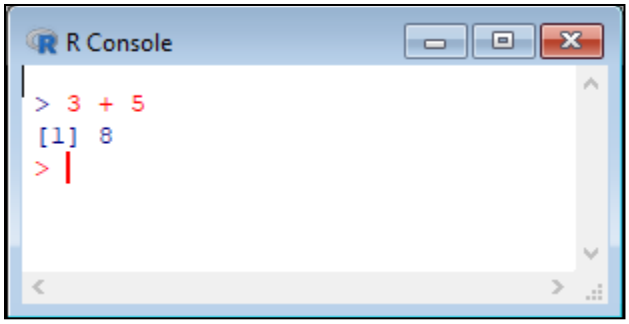


Imagen 6

El **[1]** que aparece como resultado en la consola, nos muestra el orden de los resultados, una vez nos muestra el resultado, aparecerá de nuevo nuestro símbolo del sistema (>) indicando de nuevo que R esta preparado para ser utilizado.

La Consola de R nos puede arrojar en algún momento un símbolo de " + " en color rojo, esto nos indica que el comando que ingresamos está incompleto y que esta instrucción no puede ser ejecutada sin ser finalizada correctamente esta instrucción, por ejemplo, si tecleamos " **3 + 5 -** ", R nos retornara el símbolo de " + ", por ende, tenemos que finalizar esta instrucción, por ejemplo " **4 * 2** ".

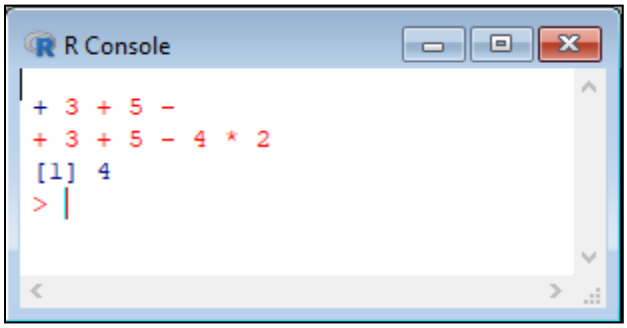


Imagen 7

Para escribir más de una función en la misma línea de comandos, necesitamos utilizar un carácter que señale que es el fin e inicio de una función y otra que sería " **;** ".

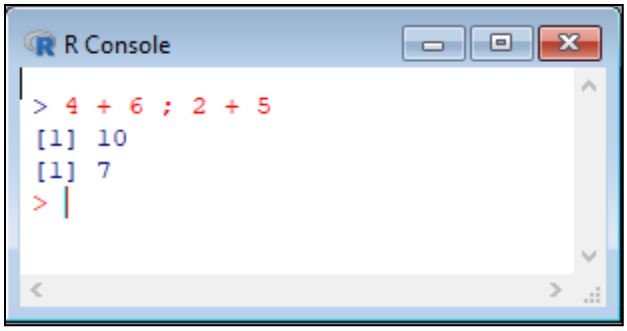


Imagen 8

La forma mas simple para comenzar a entender el leguaje de R y que es una de las funciones más potentes, es utilizarlo como una calculadora de forma interactiva, a continuación, se muestra una tabla con algunas de las funciones algebraicas y operadores, con una corta descripción de estas:

	Operadores	Descripción
ALGEBRAICAS	+	Suma.
	-	Resta.
	*	Multiplicación.
	/	División.
	%/%	División entera.
	%%	Resto de una división.
	^	Elevar a una potencia.
	round	Redondeo.
LOGICOS	<	Menor que.
	<=	Menor o igual que.
	>	Mayor que.
	>=	Mayor o igual que.
	==	Igual.
	!=	Diferente.

Tabla 1

Función	Significado
exp (x)	Exponencial de x.
log (x)	Logaritmo natural de x.
log (x, n)	Logaritmo en base n de x.
log10 (x)	Logaritmo en base 10 de x.
sqrt (x)	Raíz cuadrada de x.
factorial (x)	Factorial de x.
floor (x)	Mayor valor entero de x.
ceiling (x)	Menor valor entero de x.

signif (x, digits = #)	Devuelve el valor de x con el # de dígitos en notación científica.
cos (x)	Coseno de x (en radianes).
sin (x)	Seno de x (en radianes).
tan (x)	Tangente de x (en radianes).
acos (x), asin (x), atan (x)	Funciones trigonométricas inversas.
abs (x)	Valor absoluto de x.

Tabla 2

2.2.2 Ayuda en R.

R tiene a su disposición un sistema interno de ayuda, una serie de documentos, donde contiene una descripción de que hace la función consultada, sus argumentos, detalles sobre las operaciones que ejecuta, la soluciones que retorna y variados ejemplos.

Todo esto se puede consultar de dos formas, la primera es a través de la barra de menú en la parte superior de la interfaz, con el nombre "Ayuda" o escribiendo el comando ***> help()***.

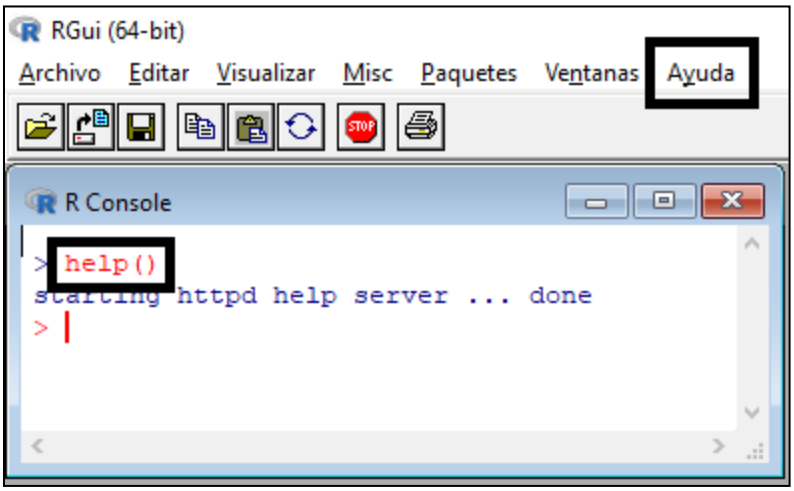


Imagen 9

las ayudas que aparecen desde la primera opción antes descrita (barra de menú):

Ayuda	Descripción
Console	Ayuda sobre el uso de las teclas y sus combinaciones en R.
R Languaje (standard)	Proporciona ayuda sobre funciones concretas.
R Languaje (HTML)	Arranca un entorno de ayuda completo en formato HTML.
Manuals	Da acceso al manual de referencia de R en formato PDF.

Apropos	Da información sobre las funciones relacionadas con una duda.
About	Versión actual de R.

Tabla 3

Ayuda en línea mediante comandos:

Ayuda	Descripción
help.start ( )	Abre ventana web para ayuda.
help (“*nombre”*) ¿*nombre*	Muestra la ayuda de la función “*nombre*”.
help.search (“*nombre*”) ¿?nombre	Muestra apariciones de la cadena “*nombre*”.
apropos (“pp”)	Lista funciones con “*pp*” en el nombre.
example(topic)	Corre el código en R con ejemplos en un tópico determinado por ejemplo “*example(plot)*”.

Tabla 4

Una vez tengamos abierta alguna de estas ayudas, nos aparecerá un formato especificando que función tiene las funciones u operadores que estábamos buscando, a continuación, mostraremos que contiene este formato:

Description	Descripción corta de la función buscada.
Usage	Para Función: suministra el nombre de la misma con todos sus argumentos y valores por defecto. Para Operador: Describe su uso típico.
Arguments	Aparecerá solamente en las funciones y describirá detalladamente cada uno de sus argumentos.
Details	Se presentará una descripción especificando su forma de uso.
Value	El tipo de objeto retornado por la función o el operador (Si se aplica).
Examples	Ejemplos de como usar el operador o la función.
References	Nos mostrara referencias bibliográficas.

Tabla 5

2.2.3 Paquetes (Packages) en R.

R es un sistema integrado al que se le pueden complementar instalándole paquetes, estos paquetes son un grupo de funciones que tienen algo relacionado entre sí, además vienen adjuntas un conjunto de archivos de ayuda y de ficheros de datos.

El paquete con que viene R al momento de su instalación, siempre estará a nuestra disposición desde que abrimos el programa, este paquete adjunta una gran cantidad de opciones de diferentes funcionalidades con las cuales podremos realizar diferentes análisis estadísticos, podremos cargar datos de fuentes externas y también generar graficas para mejores cálculos estadísticos.

2.2.3.1 Instalar Paquetes.

Tendremos ocasiones que, al momento de realizar algún calculo o algo de alta complejidad en nuestro código, necesitaremos utilizar funciones avanzadas que básicamente no vienen instaladas por defecto en R, así que necesitaremos realizar la instalación de estos paquetes.

R brinda una forma muy simple para la instalación de paquetes, ya que existen dos formas de realizarlo, que serían:

- Mediante la barra de menús, siguiendo estos pasos: nos ubicamos en la barra de menús en la parte superior y pulsamos la opción de "**Paquetes**" y en el submenú pulsamos "**Instalar paquetes(s)**"

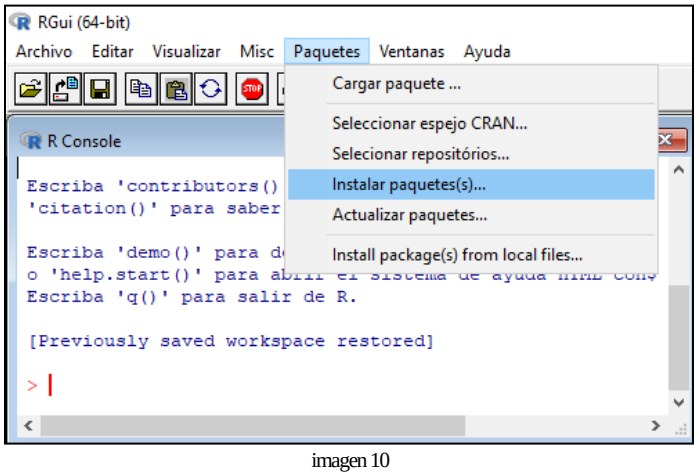


imagen 10

- Por medio de la consola, utilizando la instrucción ***install.packages("nombre")***, teniendo en cuenta que el termino nombre utilizado en esa instrucción, es el nombre

del paquete que deseamos instalar, una vez digitado este comando, se le da en la subventana "**Aceptar**" o "**Si**" según corresponda.

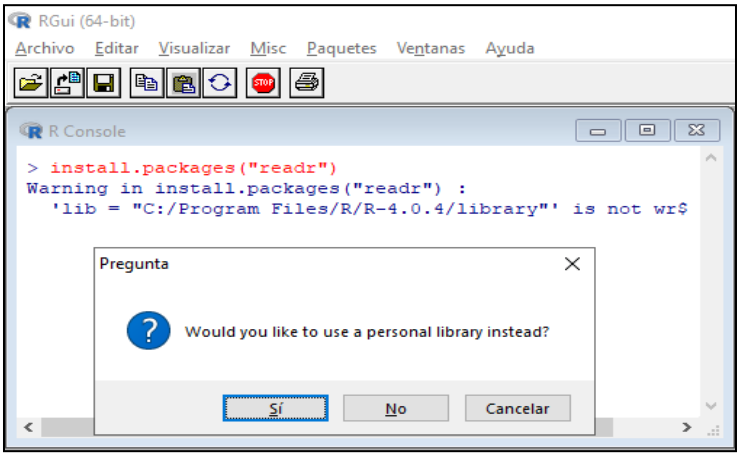


Imagen 11

Cualquiera de los dos caminos que elijamos, nos aparecerá una ventana que nos mostrara un listado de sitios llamado *"**Secure CRAN mirrors***" (Imagen 12), aquí debemos elegir la opción geográficamente mas cercana a nuestra ubicación actual, después aparecerá los paquetes disponibles en orden alfabético (Imagen 13), seleccionamos el paquete que deseamos instalar y R ya se encargara de realizar la instalación de este automáticamente.

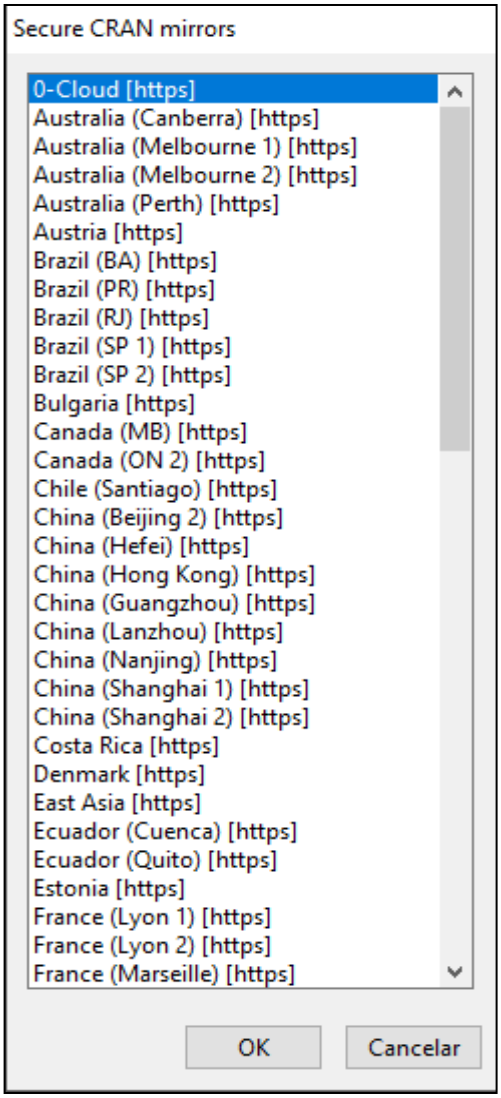


Imagen 12

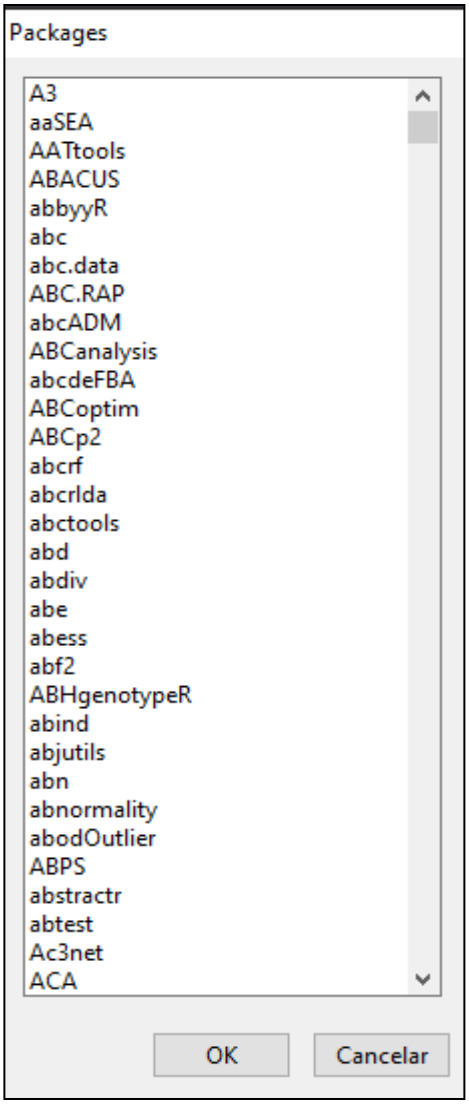


Imagen 13

2.2.3.2 Cargar Paquete.

Si deseamos utilizar las funciones que incluye un paquete, es importante tener presente dos condiciones, la primera que debemos tener ya instalada el paquete que incluye esta función y la segunda es cargarlo. Si este paquete ya lo instalamos una vez, no es necesario volverlo a realizar, pero cada vez que vamos a realizar un nuevo trabajo si es obligación cargar estos paquetes para que las funciones que vamos a utilizar se activen.

Para realizar esto tenemos dos opciones, que son:

- **Por la barra de menú**: opción Paquetes -> Cargar Paquete

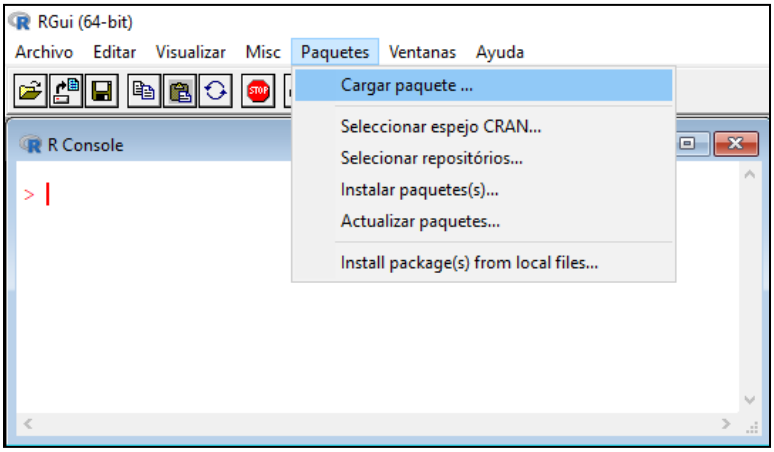


Imagen 14

- **Por consola:**
  - Comando "***library(nombre)***": Si esta el paquete lo carga con normalidad, de lo contrario, retorna un error.
  - Comando "***require(nombre)***": Si está el paquete lo carga con normalidad, de lo contrario, retorna un error.

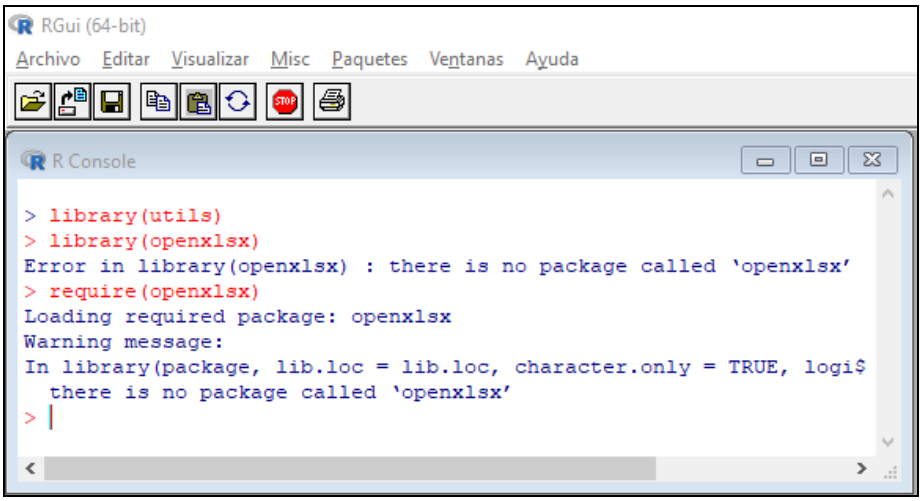


Imagen 15

2.2.3.3 Comprobar Paquetes Instalados.

Tenemos diferentes opciones para ver que paquetes tenemos instalados en nuestro Lenguaje R, para esto utilizaremos lo siguiente:

- ***installed.packages( )***: arroja como resultado una lista de todos los paquetes que se han instalado (todos menos los que vienen por defecto).
- ***(.packages( ))***: Lista los paquetes que hemos cargado
- ***(.packages(all.available = TRUE))***: arroja como resultado todos los paquetes instalados

Si con estas opciones nos aparece algún error al momento de buscar un paquete, esto quiere decir que no lo tenemos instalado y que necesitaremos instalar dicho paquete.

Capitulo III: Marco Metodológico.

3.1 Tipo de Datos y Objetos.

3.1.1 Tipo de Datos.

Los objetos en R son estructura de datos, por ello debemos hablar primero de los datos, cada uno de estos tipos de datos tiene unas características particulares que le genera una distinción de los demás, y en R tenemos 5 tipos de datos, que son:

Carácter	corresponden a caracteres o cadena de caracteres que no solo pueden contener letras si no también números, espacios, símbolos de puntuación y símbolos especiales, no olvidar que los datos de tipo carácter van dentro de comillas
Numérico	Corresponde a los valores numéricos de punto flotante, lo que significa que este tipo de datos admite valores que tengan decimales o fraccionarios, por ejemplo 5.5, 3.9 o 4/5
Enteros	Corresponde también a valores numéricos, pero de punto no flotante, esto quiere decir que no admite decimales o fraccionarios, además R los almacena de madera diferente
Complejos	Son valores numéricos con una característica, que están formados por una parte real y una imaginaria
Lógicos	Corresponde a valores que solamente van a tomar uno u otro valor (TRUE o FALSE)

Tabla 5

Los datos anterior mente explicados, son los que van a estar contenidos en los objetos.

3.1.2 Objetos.

En R, podemos decir que prácticamente todos los datos ingresados los podemos ver como objetos, además, se tiene una gran variedad de objetos lo que permitirá que el programador tenga diferentes opciones para almacenar algún tipo de información o dato para poder realizar actividades estadísticas y/o gráficos. En este capitulo hablaremos de los objetos principales que R maneja y como crearlos, estos son:

- Variables
- Vectores
- Matrices

- Arreglos
- Marco de Datos
- Listas

3.1.2.1 Variable.

La variable es el objeto más conocido en el mundo de la programación, básicamente sirve para almacenar un valor, ya sea numérico, alfanumérico o una cadena de caracteres, esto se hace con el objetivo de utilizar este dato más adelante.

Para realizar una asignación de un dato a una variable cualquiera se debe de utilizar el siguiente símbolo " <- " este se debe de colocar en el medio del nombre de la variable y el dato a asignar, así como se ve en el siguiente ejemplo:

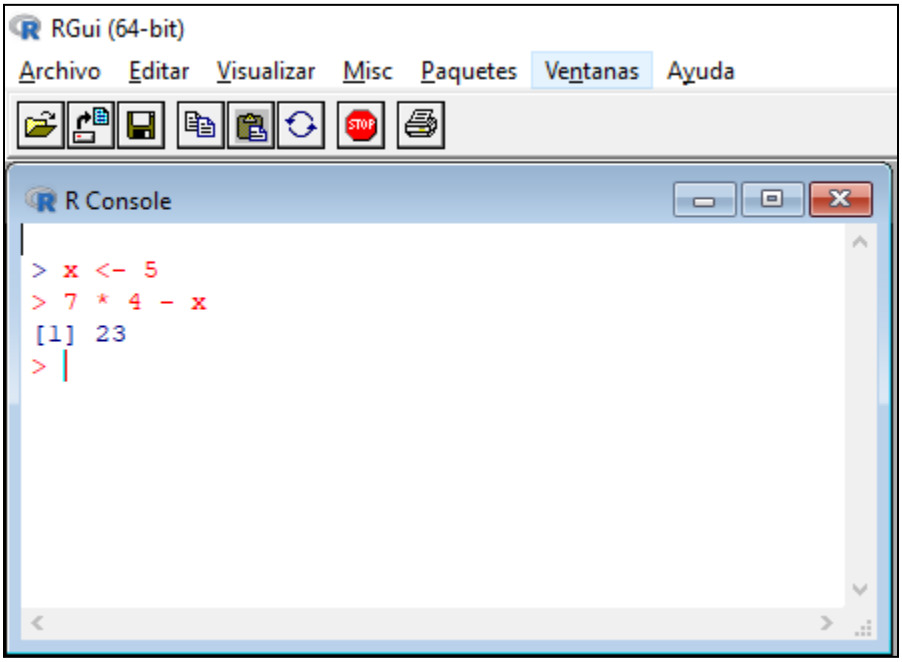


Imagen 16

Estas variables también pueden almacenar cadenas de caracteres, esto es posible colocando la cadena en medio de unos corchetes, como por ejemplo "Hola Mundo", imprimiendo así este mensaje con la función **print()** o mirando cuantos caracteres tiene esta cadena con la función **nchar()**, así como se muestra en la siguiente imagen.

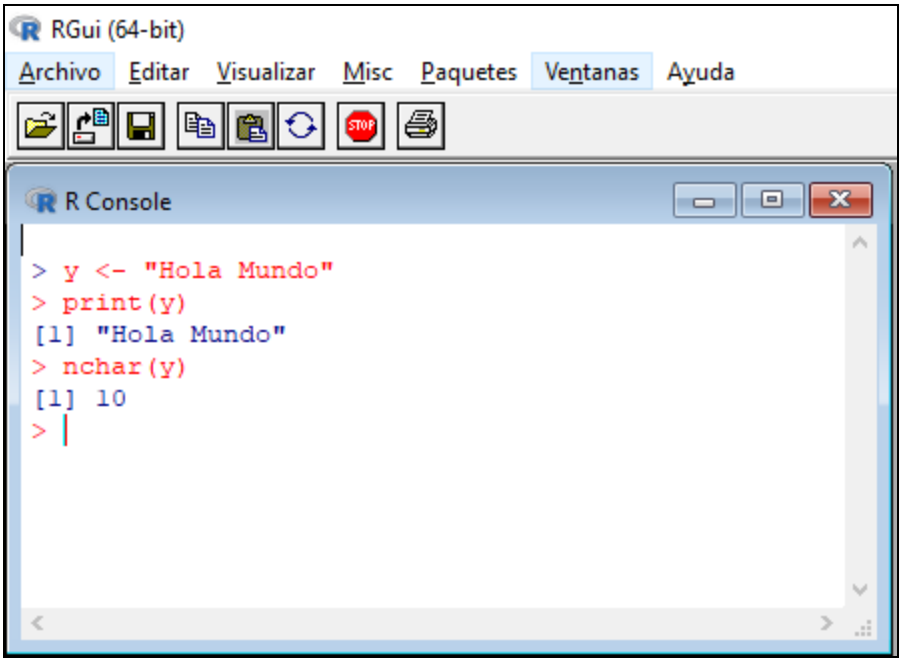


Imagen 17

3.1.2.2 Vector.

Un vector es una colección ordenada de datos o elementos de un mismo tipo lógico (TRUE o FALSE), numérico (cuantitativo), alfanumérico (cualitativo) así mismo es la estructura de datos mas sencilla en R, además es sumamente útil, pues con vectores podemos construir factores, matrices o data frames.

Para poder crear un vector, se utiliza "***c()***", lo que significa concatenar, cabe recalcar que dentro de los paréntesis debemos digitalizar los datos que deseamos almacenar, un tip para tener un mejor vector creado es que debemos nombrarlo con un nombre corto y que caracterice la información que contiene, en el medio del nombre del vector y lo concatenado, debemos utilizar los caracteres "<-" para que así se asigne esta concatenación a el vector creado.

A continuación, se mostrará un ejemplo de tres vectores que tendrán almacenada una serie de respuestas de una encuesta realizada a cuatro personas.

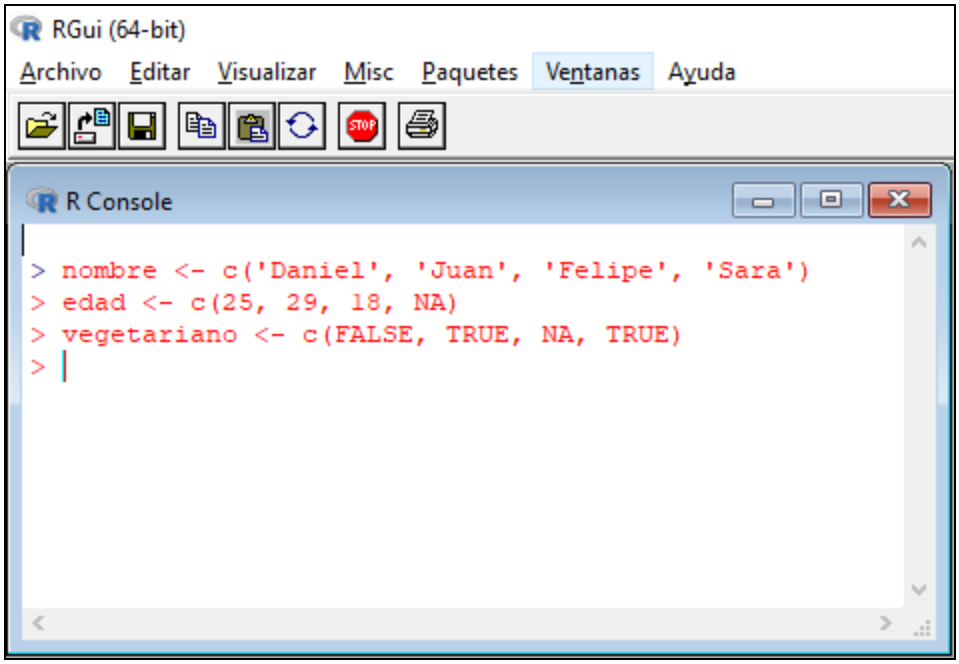


Imagen 18

El primer vector que creamos es el de ***nombre*** que contiene los nombres de las personas encuestadas, como esta variable es cualitativa (alfanumérico) es necesario la utilización de comillas simples (‘ ‘) o comillas dobles (“ “) para encerrar la información deseada. El segundo vector es llamado ***edad***, este es un vector numérico (cuantitativo) y aquí guardamos las edades de las personas encuestadas, en la posición de la cuarta persona, se colocó un ***NA*** que tiene de significado ***Not Available*** por lo que no se tomó registro de ese dato a esa persona. y por último, tenemos el vector ***vegetariano*** en este digitamos los datos de las personas encuetadas que son vegetarianos, siendo este un vector lógico, ya que la respuesta es verdadera o falsa a esta pregunta.

Ya si deseamos conocer más adelante el contenido de estos vectores, solo es necesario escribir en consola el nombre de cada uno y oprimir ***enter*** y aparecerán los datos almacenados, así como se ilustra en la siguiente imagen

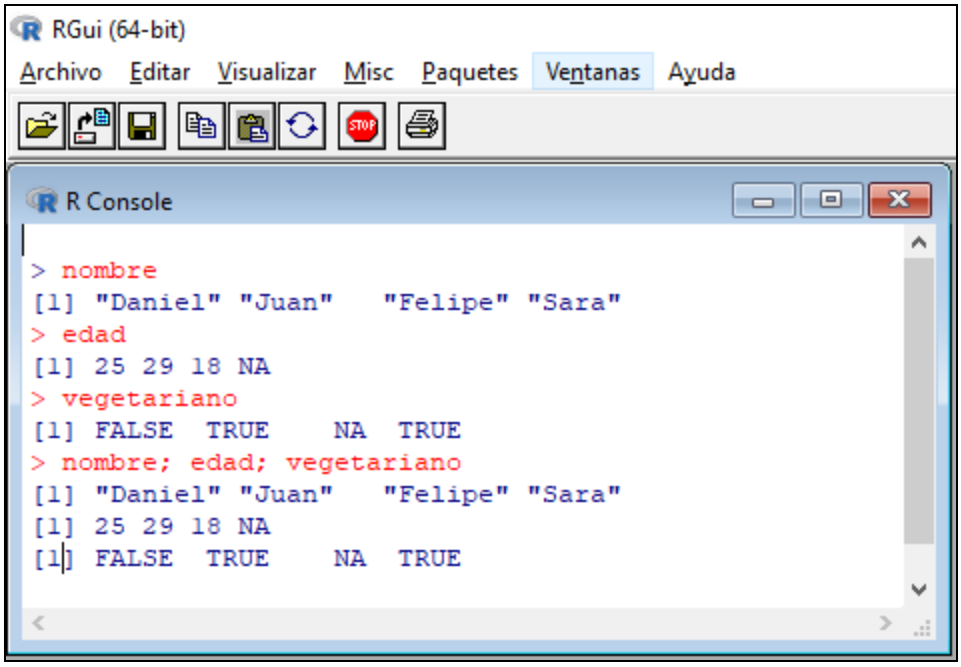


Imagen 19

3.1.2.2.1 ¿Cómo Extraer Datos de un Vector?

Si queremos buscar o extraer algún dato almacenado en estos vectores, lo debemos de colocar de la siguiente forma "***nombre[]***" aquí el nombre seria el nombre del vector en cuestión y dentro de los corchetes la posición del dato que deseamos buscar, a continuación unos ejemplos para practicar:

Ejemplo:

Si queremos extraer el nombre de la segunda persona que realizo la encuesta escribimos el nombre del vector y luego [2] para indicar que nos arroje el dato de la posición 2 del vector nombrado, así como se muestra en la siguiente imagen:

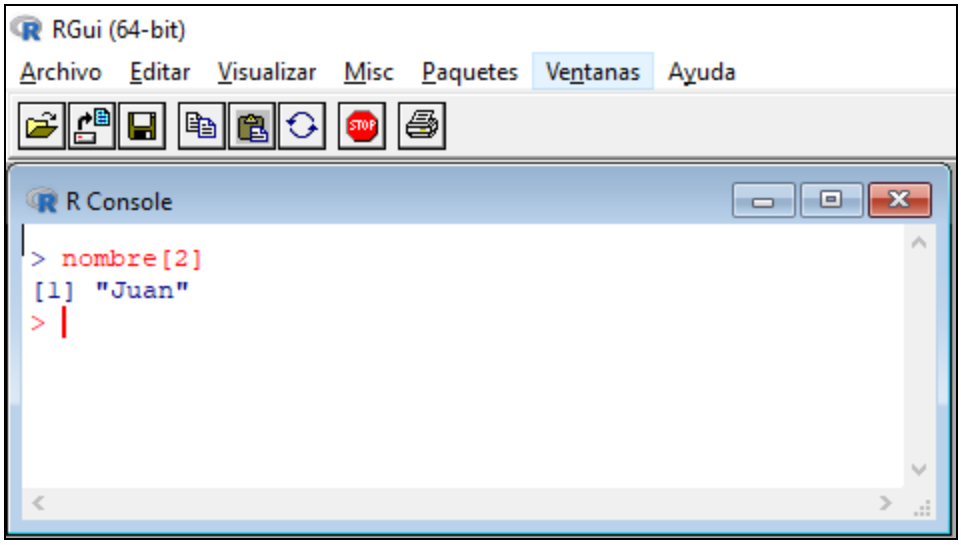


Imagen 20

Ahora si queremos que nos salga un conjunto de datos almacenados en un vector, por ejemplo, la primera y la última posición del vector **edad**, debemos escribir el nombre del vector y dentro de los corchetes escribimos otro vector con las posiciones deseadas, en este caso sería **edad[c (1, 4)],** siempre se debe crear un vector al interior si deseamos llamar un conjunto de datos de un vector en específico, ya que aparecería un error (**NO edad[1, 4]**)

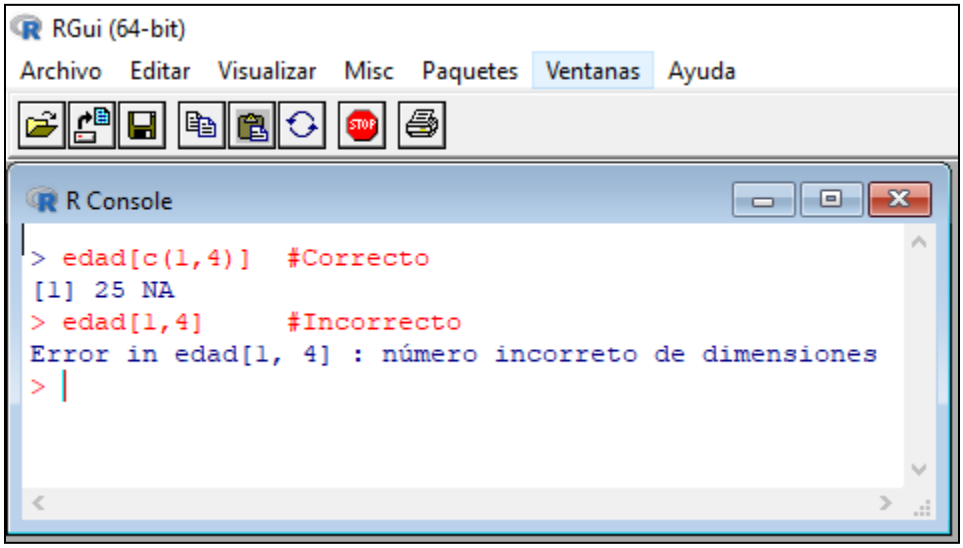


Imagen 21

Por último, si queremos que imprima todos los datos del vector, menos 1 o muchos, escribimos de la siguiente forma: **vegetariano[-2]** o **vegetariano[c (-2, -4)]**

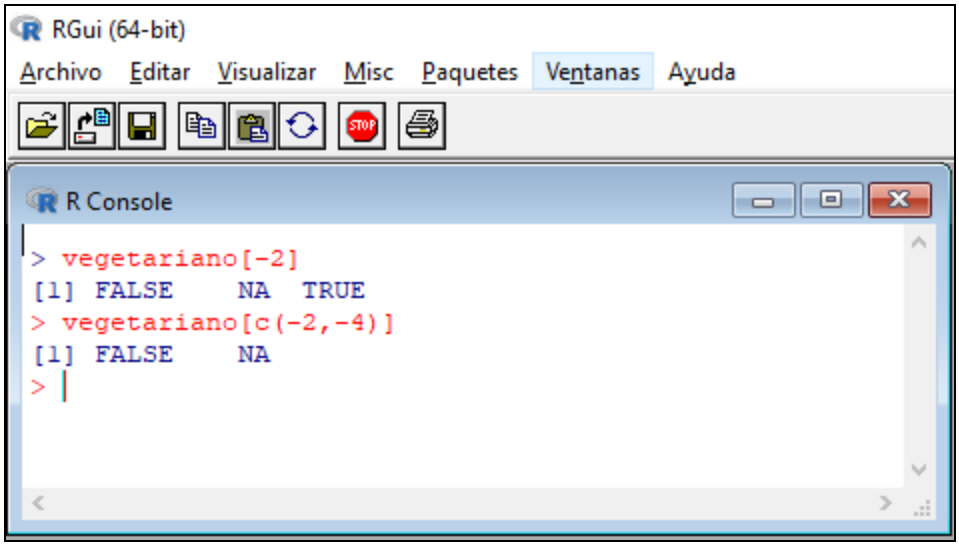


Imagen 22

3.1.2.3 Matriz.

La matriz son tablas rectangulares de filas y columnas de un mismo tipo, regularmente contiene datos numéricos, alfanuméricos o lógicos. En el ejemplo que utilizaremos para entender mejor este objeto, vamos a crear una matriz llamada mimatriz de cinco columnas y cuatro filas, con los datos del 1 al 20, asi como veremos en la siguiente imagen:

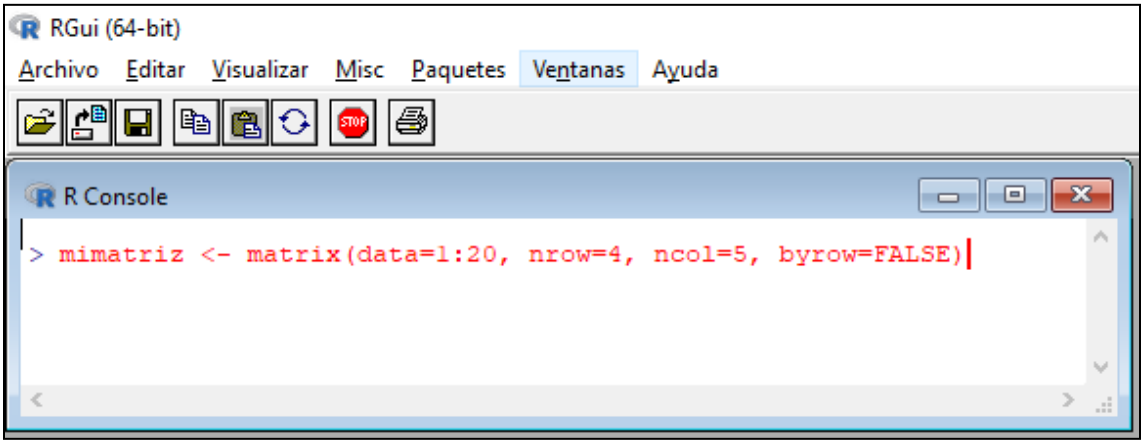


Imagen 23

En la siguiente tabla se explicara que es lo que significa los datos que se pide para crear la función matriz

data	indica los datos que se van a almacenar en la matriz
nrow y ncol	sirve para indicar la dimensión de la matriz, cuantas columnas y filas se necesitarían crear

byrow	Indica si se organiza por filas (TRUE) o por columnas (FALSE)

Tabla 6

Si queremos ver que es lo que quedo almacenado en la matriz, solo tenemos que llamarla para que nos imprima lo almacenado, como se ve en la siguiente imagen:

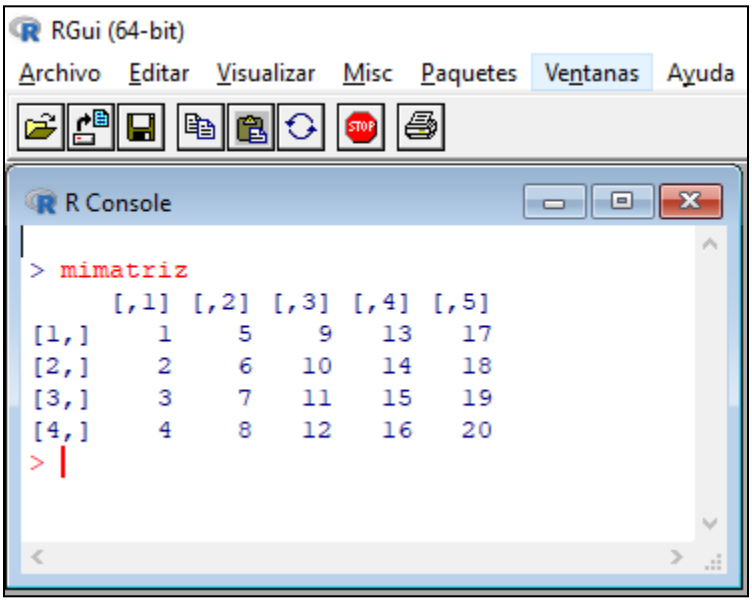


Imagen 24

3.1.2.3.1 ¿Cómo Extraer Datos de una Matriz?

Funciona casi igual como lo vimos en los vectores, para extraer un dato que guardamos en la matriz, utilizamos de nuevo los corchetes, dentro de ellos, ingresamos el numero de fila y columna que deseamos encontrar, estos datos separados con una coma "**[,]**".

A continuación, mostraremos unos ejemplos para que nos quede más claro:

Ejemplo 1: Si queremos extraer el valor almacenado en la fila 3 y columna 4

Ejemplo 2: Si queremos extraer toda la fila 2

Ejemplo 3: Si queremos extraer toda la columna 5

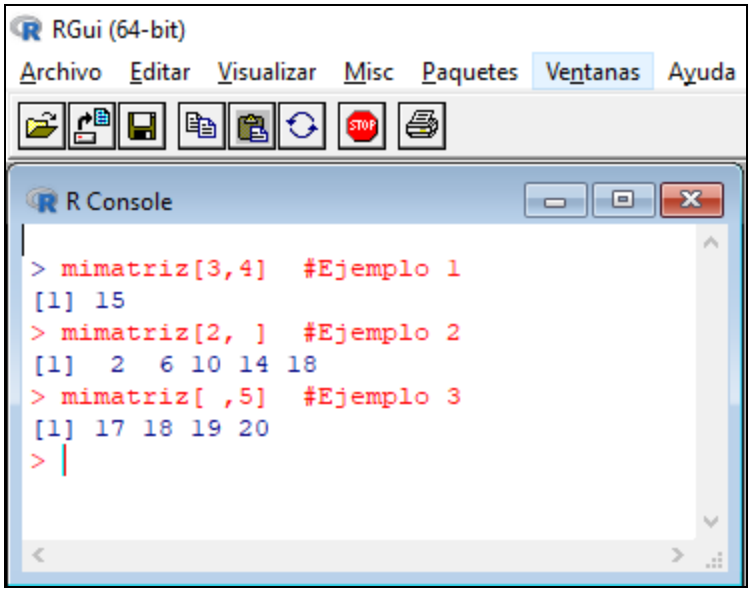


Imagen 25

Ejemplo 4: Si queremos extraer la matriz sin la fila 1 ni columna 3

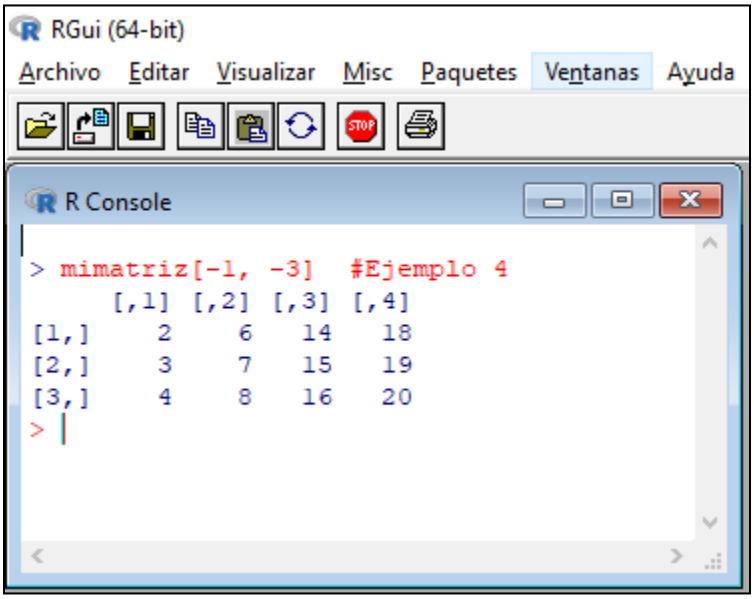


Imagen 26

Ejemplo 5: Si queremos extraer la matriz sin las columnas 2 y 4

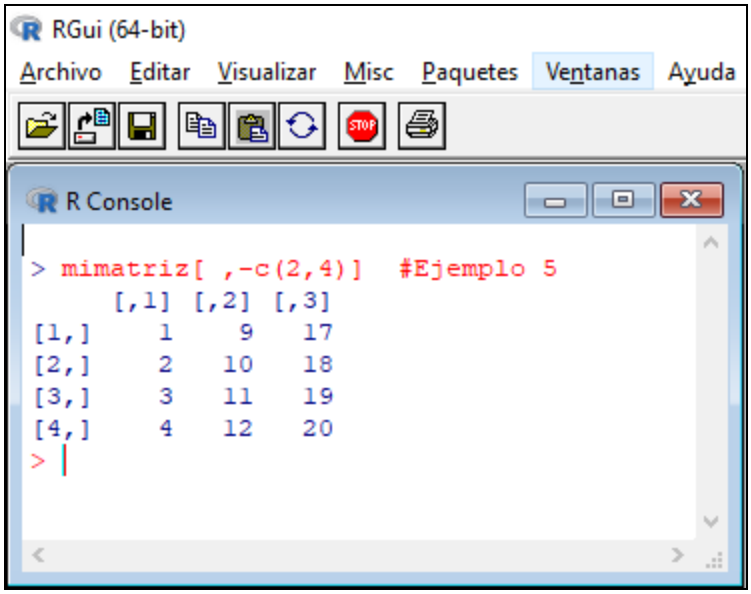


Imagen 27

3.1.2.4 Arreglo.

Un arreglo básicamente es una matriz, pero con múltiples dimensiones, también puede contener información numérica, alfanumérica o lógica, para explicar los datos que nos pide al momento de crear un arreglo:

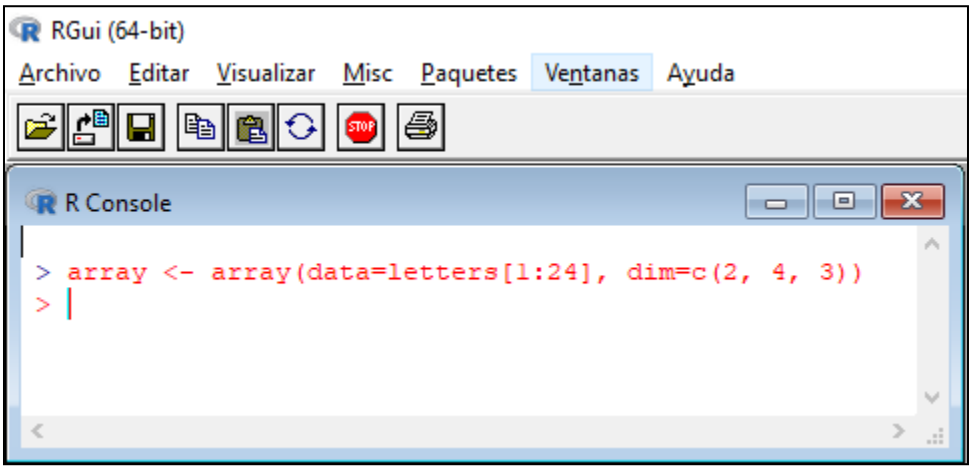


Imagen 28

data	indica los datos que se van a almacenar en la matriz
dim	Indica las dimensiones que van a tener el array

Tabla 7

Aquí en el data, le almacenamos las primeras 24 letras del abecedario en minúsculas, con las dimensiones (2,4,3) que el 2 es el numero de filas, el 4 el número de columnas y el 3 es el numero de secciones.

Si queremos ver que es lo que quedo almacenado en el array, solo tenemos que llamarla para que nos imprima lo almacenado, como se ve en la siguiente imagen:

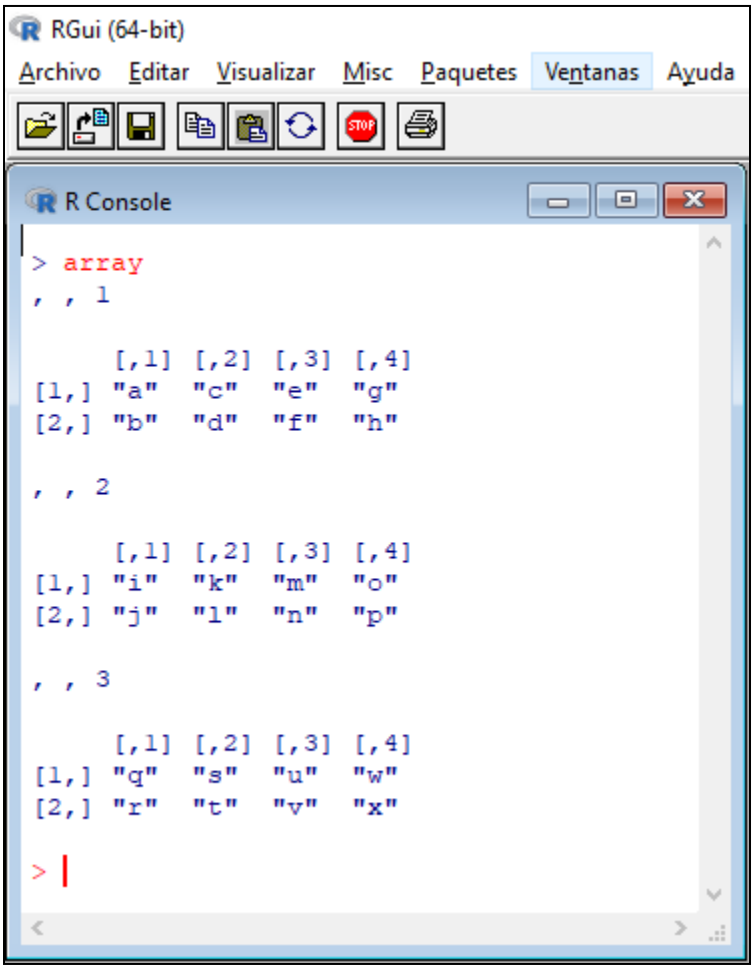


Imagen 29

3.1.2.4.1 ¿Cómo Extraer Datos de un Arreglo?

Funciona igual como lo vimos en la parte de la matriz, con la diferencia de que tiene una tercera variable que es la de la dimensión, recordemos que debemos utilizar los corchetes y dentro de ellos, ingresamos el numero de fila, columna y dimensión, separados con una coma ”**[, ,]**”.

A continuación, mostraremos unos ejemplos para que nos quede más claro:

Ejemplo 1: Si queremos extraer la letra almacenada en la fila 2 y columna 3 de la tercera capa

Ejemplo 2: Si queremos extraer la primera capa completa

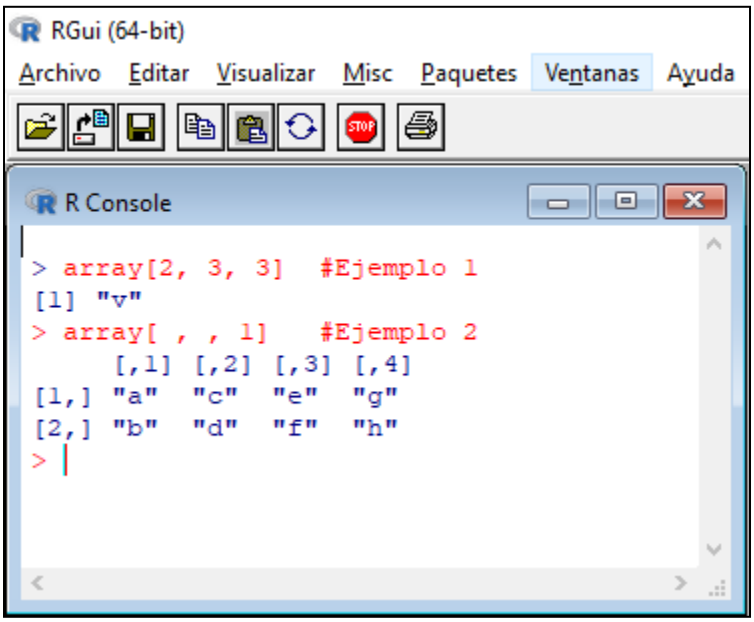


Imagen 30

Ejemplo 3: Si queremos extraer la segunda fila de todas las capas:

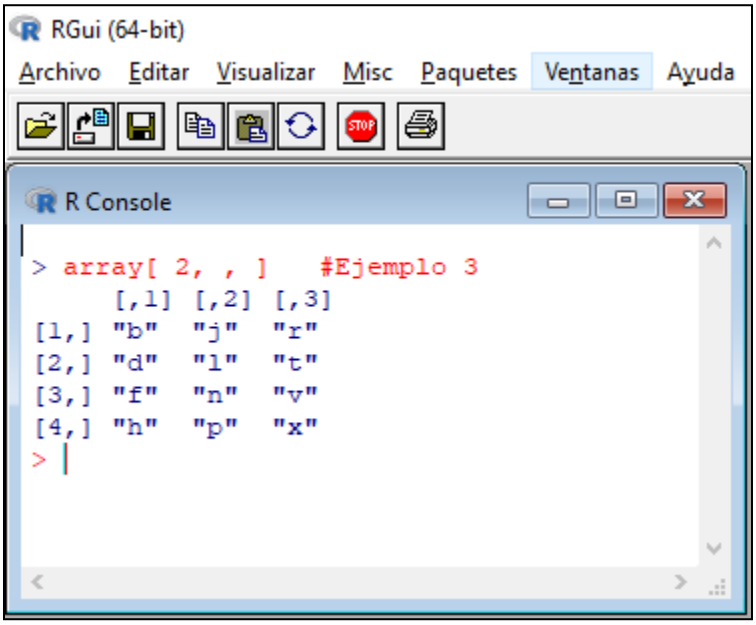


Imagen 31

3.1.2.5 Marco de Datos.

El data frame es el objeto que mas se utiliza en R, ya que permite que se agrupen vectores con datos de tipo diferente, ya sea numérica, alfanumérica o lógica. Lo único que se debe de cumplir es que estos vectores deben de tener la misma longitud. Para crear este dato de utiliza la función "**data.frame( )**". Como ejemplo crearemos este data frame llamado marco con los vectores nombre, edad y vegetariano definidos anteriormente:

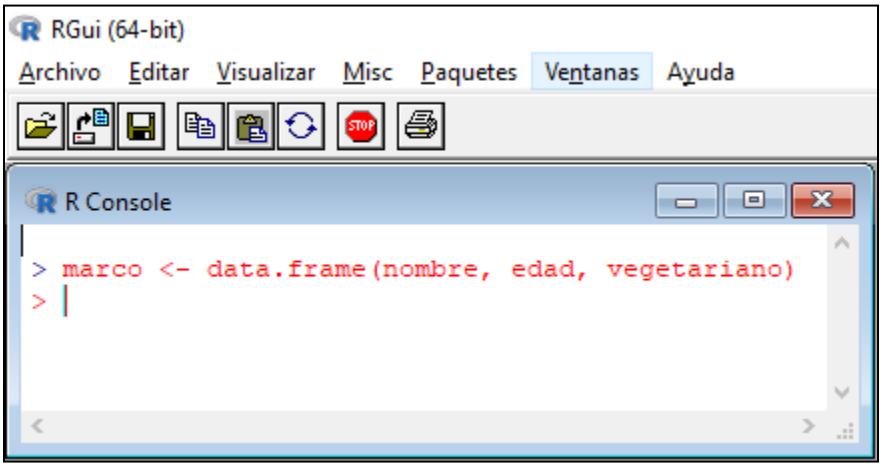


Imagen 32

Si queremos ver que es lo que quedo almacenado en el marco, solo tenemos que llamarla para que nos imprima lo almacenado, como se ve en la siguiente imagen:

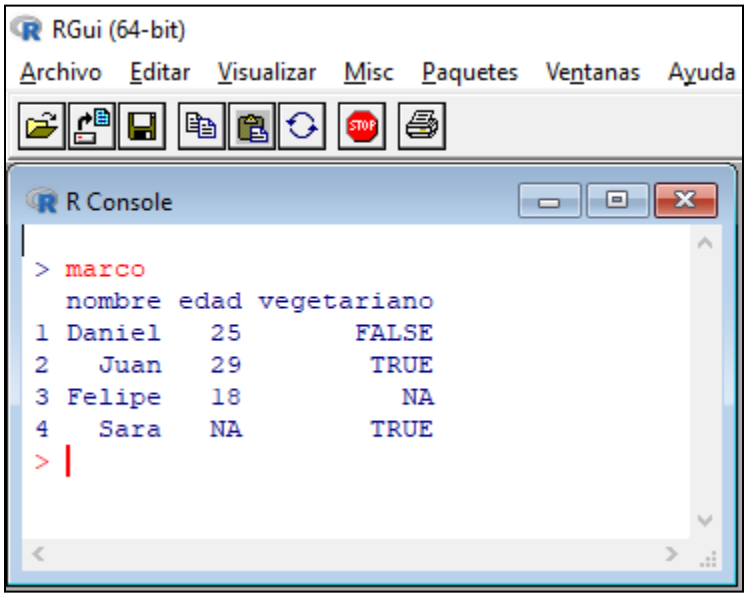


Imagen 33

Como podemos ver, el resultado nos muestra de una forma ordenada los datos almacenados, vemos 3 variables en el encabezado de las columnas, esto coincide con los nombres de los vectores que creamos, y al lado izquierdo una secuencia de números de referencia que nos permite identificar los datos de cada persona en cuestión en nuestra base de datos.

3.1.2.5.1 ¿Cómo Extraer Datos de un Marco de Datos?

Para extraer las variables que guardamos en el marco de datos, tenemos diferentes formas de realizarlo, que serian a traces de un operador $, corchetes simples [ ] o corchetes dobles [[ ]], mostraremos unos ejemplos para ver como operan:

Ejemplo 1: Buscamos extraer la variable nombre como vector.

Ejemplo 2: Buscamos extraer la variable nombre como vector, si recordamos la ubicación de esta.

Ejemplo 3: Buscamos extraer la variable nombre como vector, si recordamos el nombre de la variable.

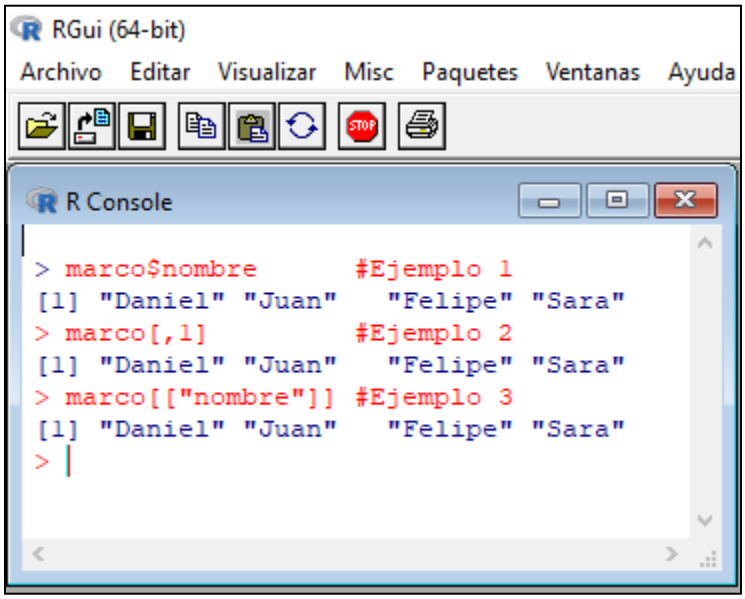


Imagen 34

Si en el ejemplo 3 colocamos solo corchetes simples, esto no nos aparecerá como vector si no como un marco de datos, así como vemos:

```
> marco["nombre"]
  nombre
1 Daniel
2   Juan
3 Felipe
4   Sara
```

Imagen 35

3.1.2.5.2 ¿Cómo extraer Subconjuntos de Datos de un Marco de Datos?

se debe de utilizar la siguiente función "**subset(x, subset, select)**" en la siguiente tabla explicaremos que se debe de tener en cuenta con estos parámetros:

x	Sirve para indicar el marco de datos original
subset	Sirve para colocar la condición
select	Sirve para quedarnos solo con algunas de las variables del marco de datos

Tabla 8

Ejemplo 1: Marco de Datos con solo personas que SEAN vegetarianas

Ejemplo 2: Marco de Datos con solo personas mayores o iguales a 25 años

Ejemplo 3: Marco de Datos con solo personas que NO son vegetarianas con su nombre

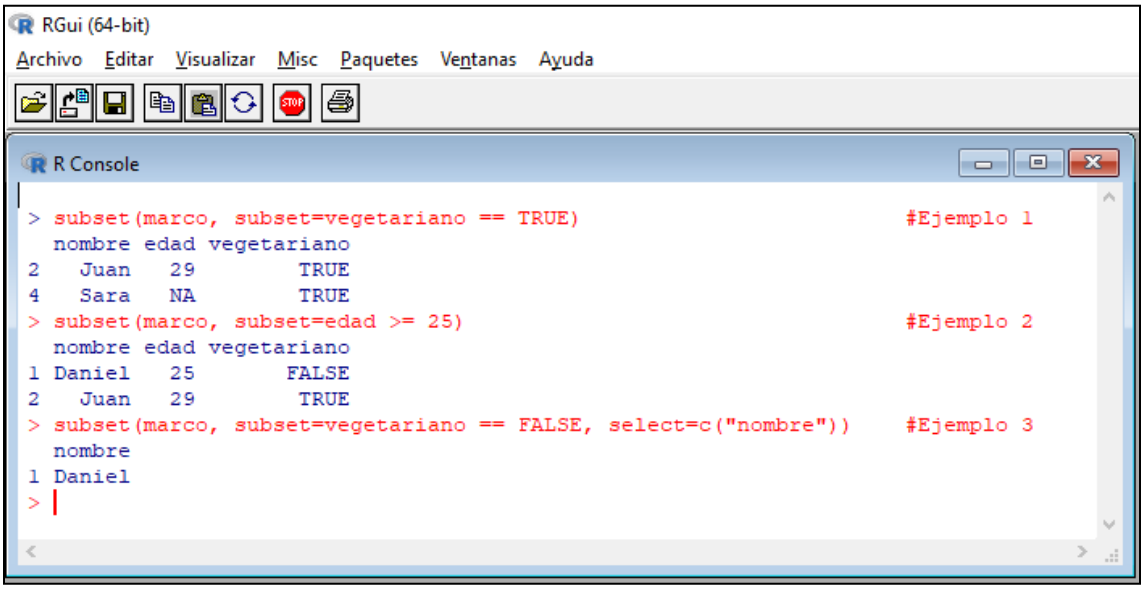


Imagen 36

3.1.2.6 Listas.

Las listas es también un objeto muy frecuente al momento de almacenar objetos de diferente tipo de dato, La función que se utiliza para crear este objeto es "**list( )**", para entender mejor este objeto, lo realizaremos con un ejemplo creando una lista llamada **lista** que va a contener tres objetos, un vector con cinco números al azar llamado **vector**, una matriz de 6 por 2 con los datos del 1-12 positivos y por último, un objeto que será el marco que creamos en el apartado anterior, a continuación mostraremos las líneas de comandos que se deben ingresar para crear la lista:

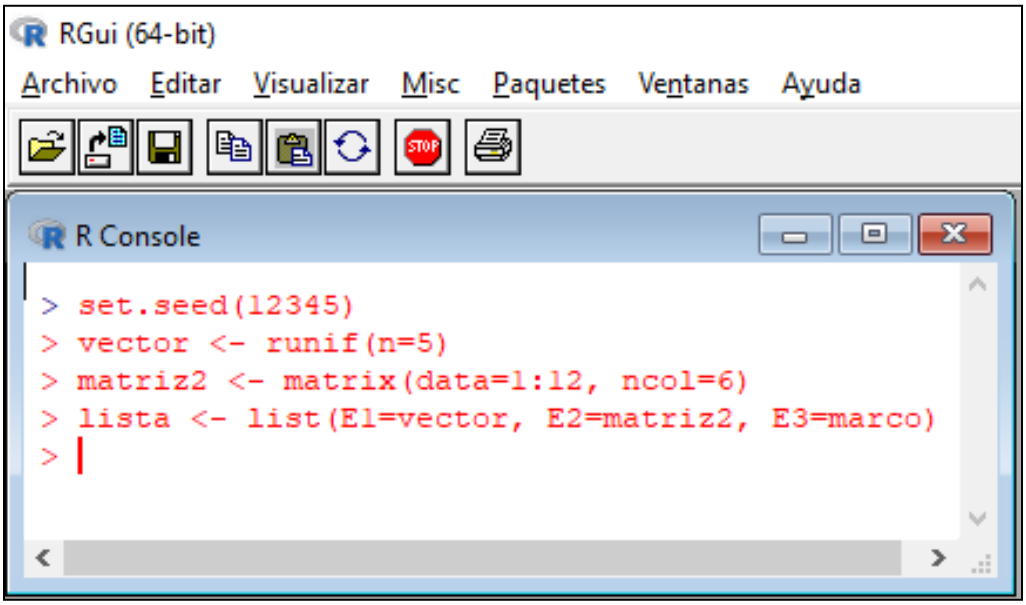


Imagen 36

la siguiente tabla explicaremos que se debe de tener en cuenta con estas instrucciones:

set.seed	Sirve para fijar que los números aleatorios generados siempre sean los mismos
runif	Genera números aleatorios, en este caso son 5
matriz	Crea una matriz con los números 1-12 con 6 columnas
list	Crea una lista con los tres objetos, vector, matriz y marco colocando un nombre especial para cada elemento de la lista

Tabla 9

Si queremos ver que es lo que quedo almacenado en la lista, solo tenemos que llamarla para que nos imprima lo almacenado, como se ve en la siguiente imagen:

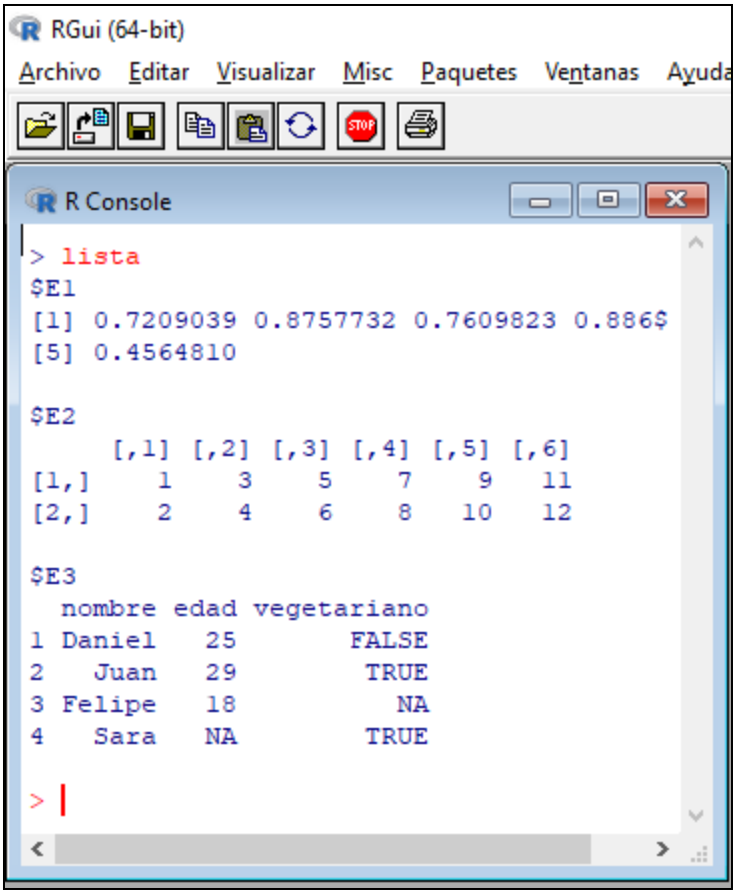


Imagen 37

3.1.2.6.1 ¿Cómo Extraer Datos de un Marco de Datos?

Para extraer las variables que guardamos en el marco de datos, tenemos diferentes formas de realizarlo, que serían a traces de un operador $, corchetes simples [ ] o corchetes dobles [[ ]], mostraremos unos ejemplos para ver cómo operan.

Ejemplos: vamos a imprimir los datos de la matriz almacenada con el nombre E2 del objeto lista con las tres opciones que tenemos

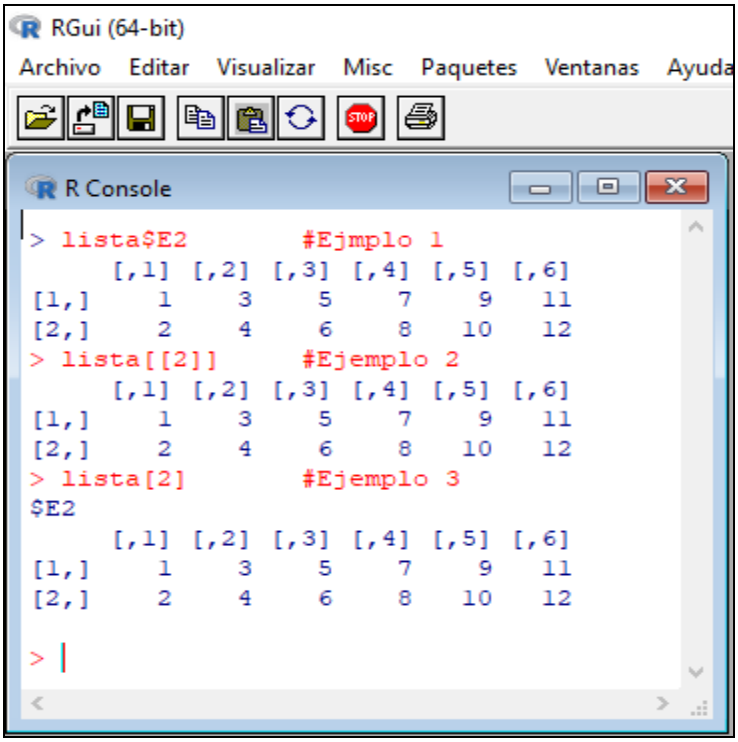


Imagen 38

Podemos observar en los resultados que son similares, pero realmente no son iguales, para poder observar la diferencia, vamos a utilizar la función "**class( )**" para poder ver la clase a la que pertenecen:

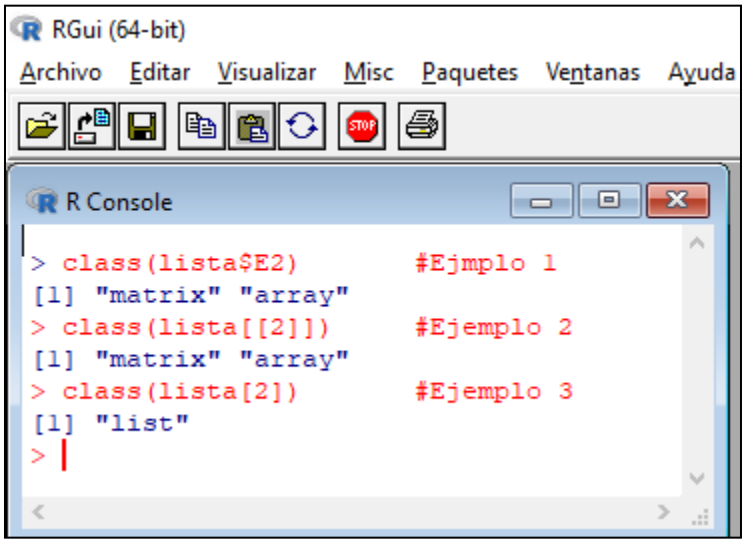


Imagen 39

Aquí ya se puede ver claramente que al momento de usar $ o [[ ]] lo que nos responde es en forma de una matriz, pero cuando se usa [ ] los corchetes simples, el resultado es una lista.

3.2 Estructura de Control.

Una ventaja de R comparada con otros lenguajes estadísticos, es la posibilidad de programar de una manera fácil y sencilla una serie de análisis que se pueden ejecutar de manera sucesiva, esto nos ayuda a optimizar tiempo en analizar muchos datos de la misma forma.

Las estructuras de control, como su nombre lo indica, permite controlar el flujo de ejecución de un programa, dándole una dirección y un orden, estas estructuras son:

if, else	Si, de otro modo
for	Para cada uno en
while	Mientras
repeat	Repetir procedimiento

Tabla 10

Estas estructuras de control permiten establecer unos condicionales en nuestro código. Un ejemplo, es que condiciones debemos cumplir para para que se ejecute una función, o que se debe de realizar para que se ejecute una operación

3.2.1 If, Else.

Se utiliza para realizar una sentencia condicional, que es una instrucción o grupo de instrucciones que se puede ejecutar o no en una función, además permite redirigir un curso de acción según la evaluación de una condición simple, sea TRUE o FASLE.

Si la condición es TRUE, se ejecuta el conjunto de instrucciones 1, pero si es al contrario, si esto es un FALSE, se ejecuta el conjunto de instrucciones 2.

A continuación, se mostrará la estructura básica de su uso y el diagrama de flujo:

```
if (condicion) {
  operacion 1
  operacion 2
  ...
  operacion final
}
else {
  operacion 1
  operacion 2
  ...
  operacion final
}
```

Imagen 40

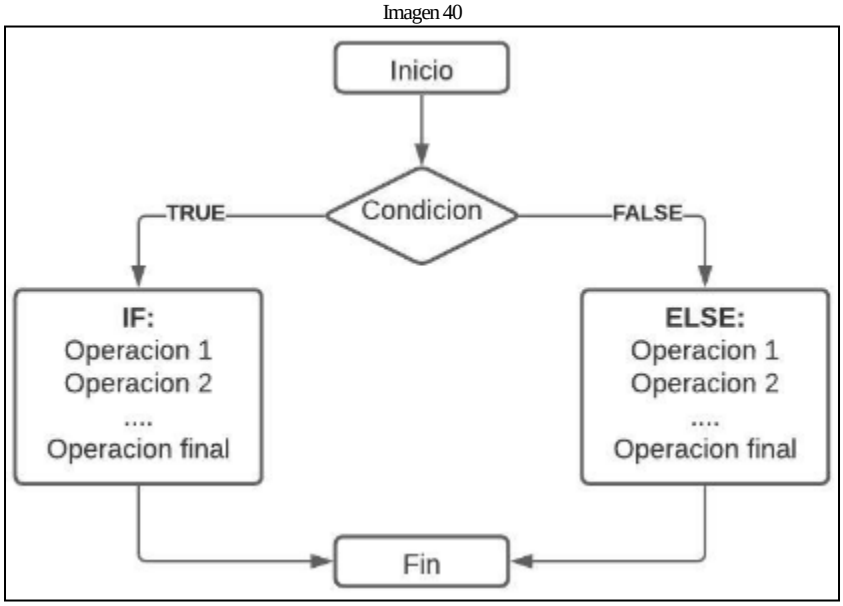


Imagen 41

Teniendo ya una idea de la estructura básica para su uso, pondremos un ejemplo para entender bien como realizar este consolidado de condiciones:

Ejemplo[1]: Un estudiante desea saber si va a aprobar la materia de Matemáticas, él tiene estas 5 notas: **2, 4, 3, 2, 3.** La nota mínima es 1 y la máxima es 5, y para pasar la materia, necesita un 3.

[1] Para este ejemplo necesitaremos la función **mean( )**, que esta es la que sacara la media de los valores que se le ingresen (el promedio), esta función ya esta disponible en R, ya que es una función básica que viene descargada por defecto.

Para darle solución a este ejemplo, debemos tener presente que **SI** el promedio de las notas es **MAYOR O IGUAL** a 3, deberá mostrar un "**APROBADO**", **DE LO CONTRARIO**, deberá mostrar un "**REPROBADO**".

Con esto presente, aplicaremos esta lógica con esta estructura **if, else** en la función llamada promedio.

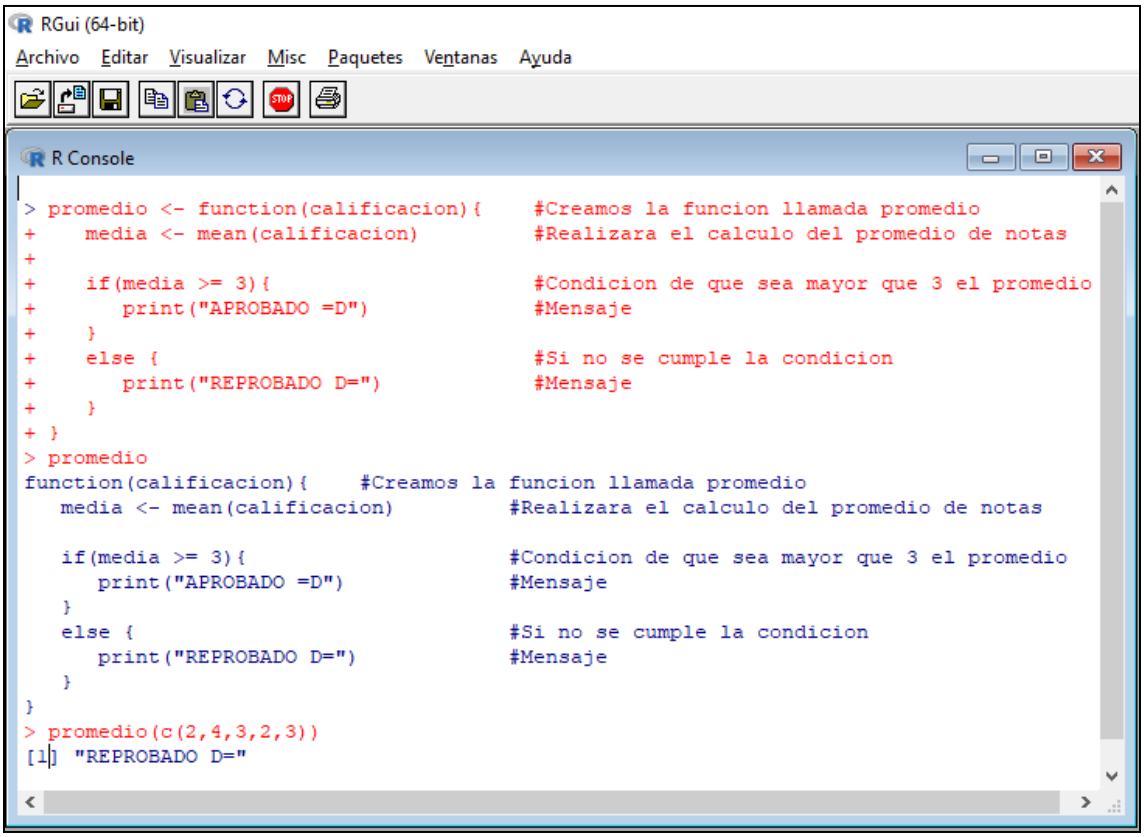


Imagen 42

Podemos ver que esta funcionando correctamente, aunque la respuesta puede ser mucho mas agradable.

Para esto utilizaremos la función "**paste0( )**" para que el resultado del promedio de las calificaciones, tenga un texto y además con el resultado de si es **APROBADO** o **REPROBADO** lo que hace esta función es que acepta cadenas de texto, y las concatena con el resultado de la función anterior, devolviendo una cadena nueva.

A continuación, mostraremos este ejercicio con esa mejora.

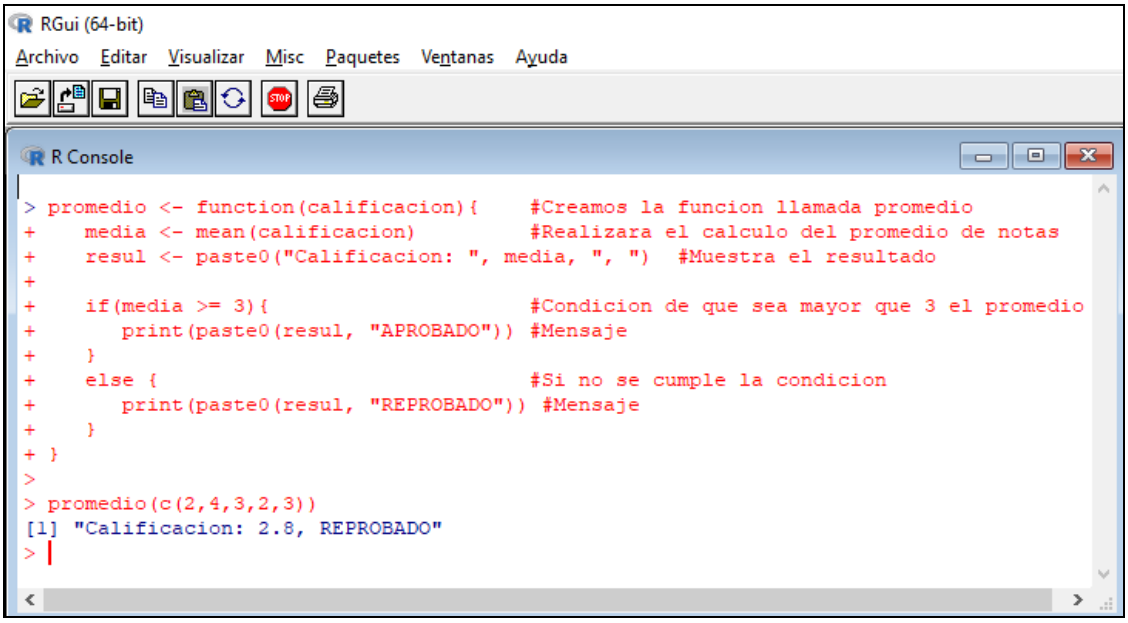


Imagen 43

Esta función se puede hacer mucho mas compleja, pero lo visto hasta ahora deja claro cómo funciona la función **if, else.**

3.2.2 Bucles.

Los bucles o ciclos que son sentencias que repite un bloque de instrucciones, hasta que la condición asignada a dicho bloque deje de cumplirse, los bloques de instrucciones que se repiten se suelen llamar cuerpo de bucle y cada repetición se llama interacción.

Los tres bucles principales son los bucles **for, while** y **repeat.**

3.2.2.1 Bucle For.

El bucle for, repite el bloque de instrucciones un numero predeterminado de veces, en la siguiente imagen se verá su estructura y el diagrama de flujo:

```
for (i in secuencia){
  operacion 1
  operacion 2
  ...
  operacion final
}
```

Imagen 44

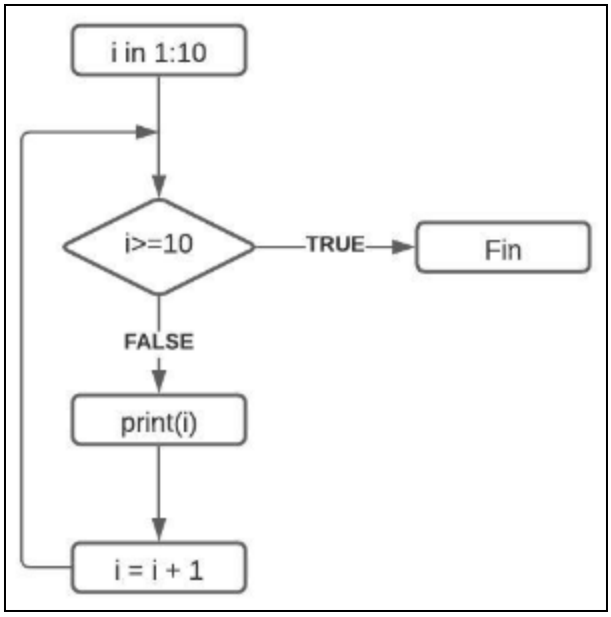


Imagen 45

Veremos un ejemplo sencillo para ver cómo actúa este bucle, generaremos un bucle que nos imprima los números del 1 al 10:

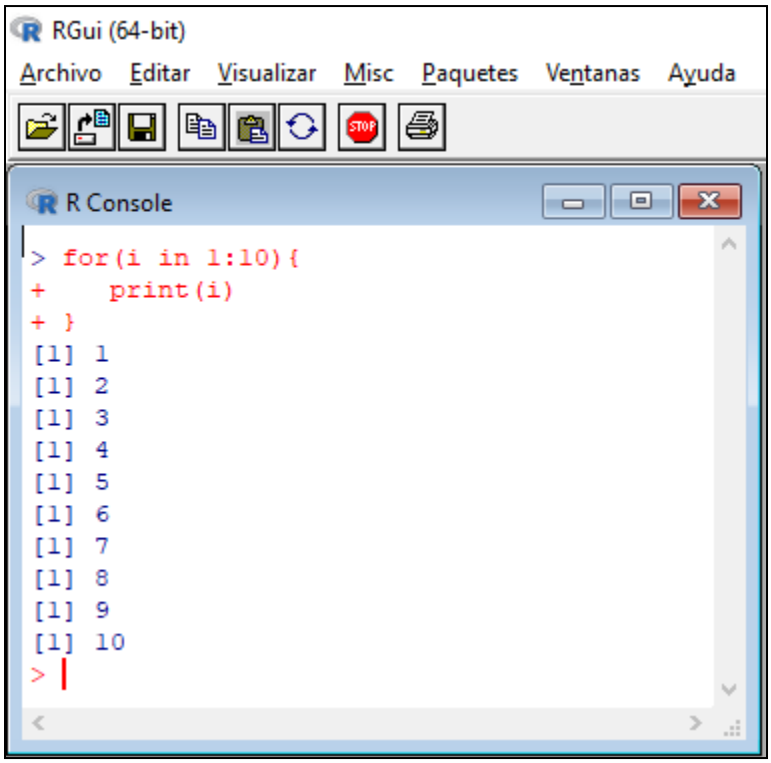


Imagen 46

Esto nos mostrara como resultado una interacción de 10 resultados, mostrándolos en orden y deteniéndose hasta cumplir la condición.

Un segundo ejemplo[2] muy interesante es el de crear un **for condicional**, por ejemplo escribiremos un vector que contendrá las repeticiones "**N**" Y "**H**" donde nos mostrara la secuencia de mujer y hombre, a continuación estará el ejemplo resuelto:

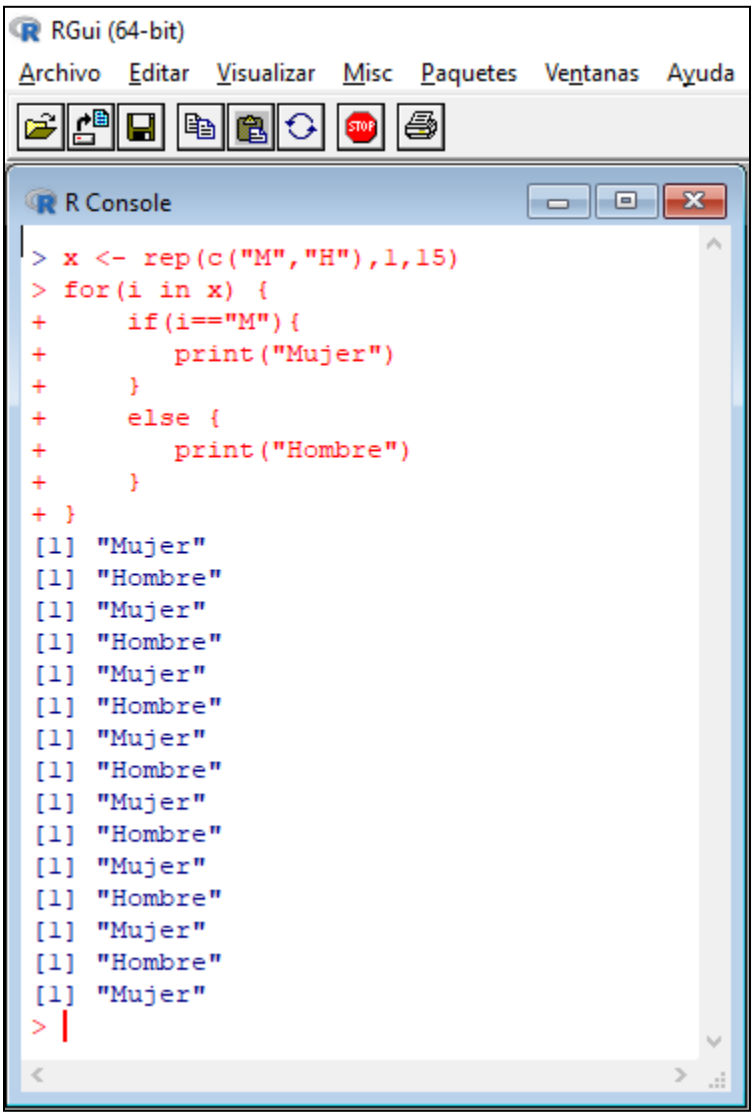


Imagen 47

[2] Para realizar este ejemplo se utilizó la función **rep()**, con esta podemos realizar repeticiones, la función completa es: **rep(x, times=1, length.out=NA, each=1)**. Los argumentos que vemos son:

x	Vector con los elementos a repetir
times	Número de veces que el vector x se debe repetir
length.out	Longitud deseada para el vector resultante
each	número de veces que cada elemento de x se debe repetir

tabla 11

3.2.2.2 Bucle While.

El bucle while permite repetir la ejecución de un conjunto de instrucciones mientras la condicional se cumpla, en la siguiente imagen se verá su estructura y el diagrama de flujo:

```
while (condicion) {
  operacion 1
  operacion 2
  ...
  operacion final
}
```

Imagen 48

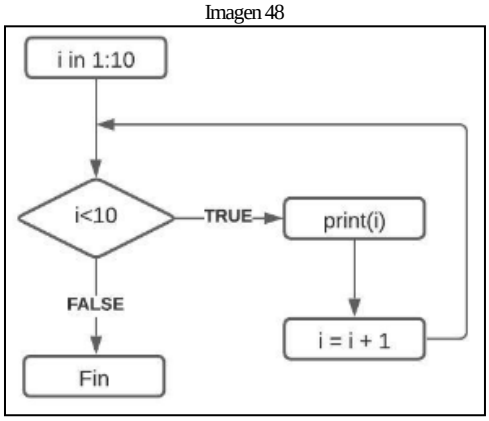


Imagen 49

El ejemplo que utilizaremos es un conteo, que el bucle que se crea es que si el conteo sea menor a 10 se va a repetir el ciclo hasta que esto ya sea falso

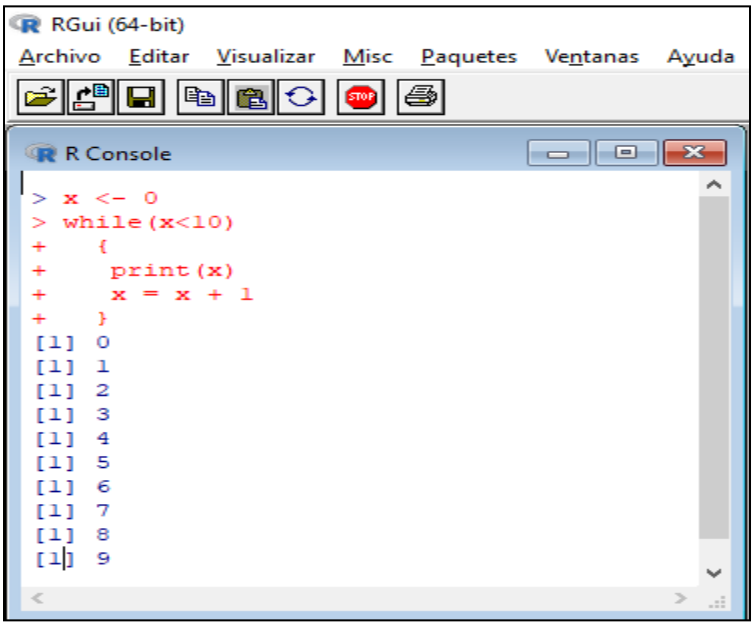


Imagen 50

Como podemos observar, nos arroja una lista ordenada de los números de x, realizando las interacciones necesarias en donde se detendrá al momento de que el ciclo ya no se cumpla[3].

3.2.2.3 Bucle Repeat.

Repeat ejecuta un bucle infinito y siempre se utiliza con otros argumentos para detenerlo[4], como, por ejemplo:

break	Suspende y detiene el bucle
next	Emite una interacción del bucle
return	Sale del bucle

Tabla 12

En la siguiente imagen se verá su estructura y el diagrama de flujo:

```
repeat {
  operación 1
  operación 2
  ...
  operación final
  if (condición) break
}
```

Imagen 51

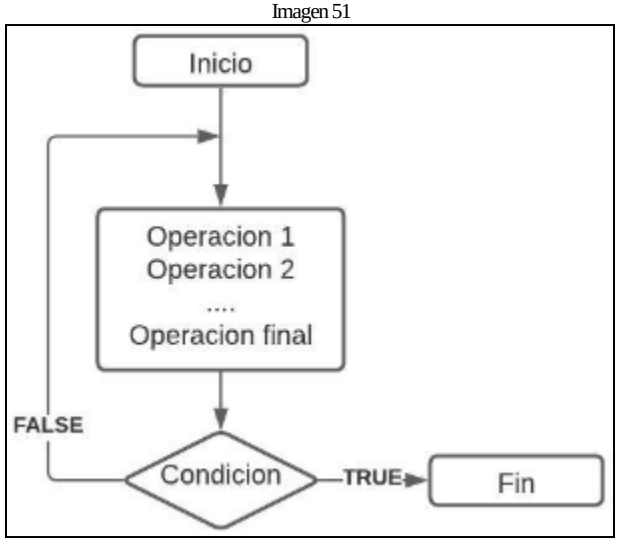


Imagen 52

[3] Como es evidente, el bucle de for y el bucle de while trabajan de la misma forma, la única diferencia es que, en el for, se necesita que sea FALSA para cumplir los ciclos y el bucle de while se necesita que sea VERDADERA para que se mantenga en el ciclo.

[4] Se debe de tener mucho cuidado al momento de utilizar repeat, ya que puede generar un bucle infinito y causar un mal funcionamiento de R y bloquear el sistema, siempre que se pueda, es mejor reemplazar el bucle repeat por el bucle for

El ejemplo que utilizaremos, vamos a crear un vector "**x**" que dentro del bucle nos mostrará en pantalla un conteo de 10 en 10 hasta 100, y se detendrá utilizando el break:

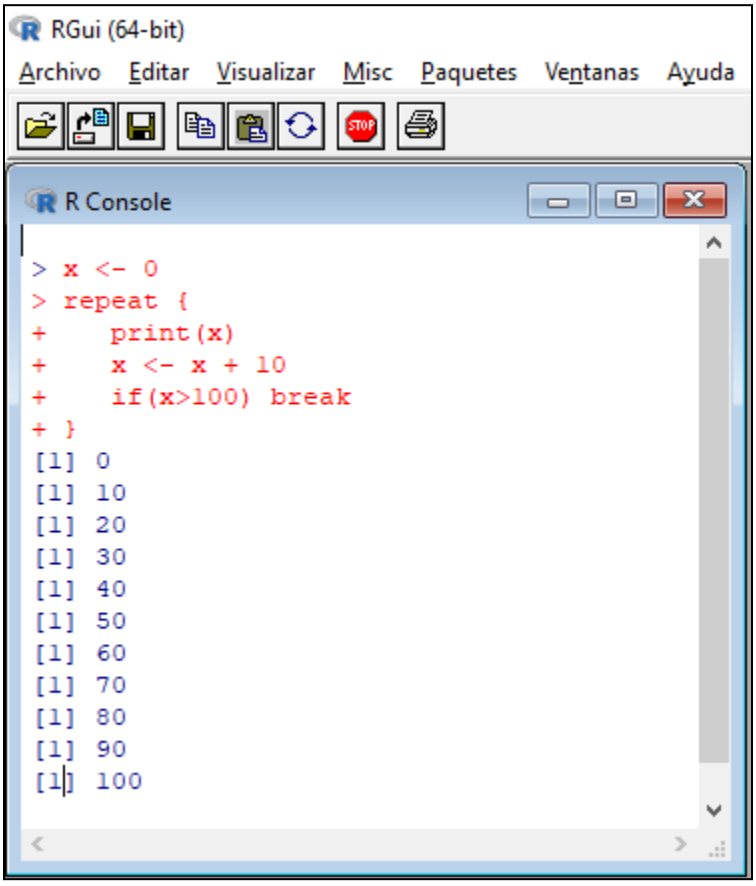


Imagen 53

3.3 Importación y Exportación de Datos.

En este capítulo veremos cómo importar datos con R, lo primero que tendremos en cuenta es que podemos importar datos de dos tipos de archivos que son:

- Extensión **.txt**
- Extensión **.csv**

3.3.1 Almacenamiento de Datos.

Antes de arrancar con la importación de datos, vamos a explicar como debemos de almacenar estos datos en los dos formatos antes descritos, para que R los pueda leer sin ningún problema. En el cuadro que se mostrará a continuación, veremos una pequeña base de datos, esta

será la misma con la que trabajamos en el capitulo de "**TIPOS DE DATOS Y OBJETOS**" como para trabajar con lo que ya habíamos creado.

Nombre	Edad	Vegetariano
Daniel	25	FALSE
Juan	29	TRUE
Felipe	18	NA
Sara	NA	TRUE

Tabla 13

Antes de continuar debemos de tener en cuenta algunas condiciones para almacenar de la mejor manera estos datos, para evitar inconvenientes a futuro:

- Las columnas y filas sean consistentes, que quiere decir, pues que debemos evitar que existan filas combinadas, o filas o columnas en blanco por verse estético.
- Debemos evitar símbolos extraños en las etiquetas de filas o columnas, ya que R no gestiona bien los símbolos como: €, Ñ, %, $, etc.
- Un fenómeno muy habitual en los conjuntos de datos, es la existencia de datos perdidos, ya que desconocemos cual es el valor que un individuo toma en una cierta característica o variable, es por esto que, si tenemos datos perdidos, su codificación debe de ser consistente, ósea utilizar el código **NA** (**Not Available**)

3.3.1.1 Guardar Datos en Excel.

Para guardar los datos que describimos en la tabla anterior en Excel, debemos abrir un nuevo archivo y copiar la información tal como se ilustra en la imagen de abajo.

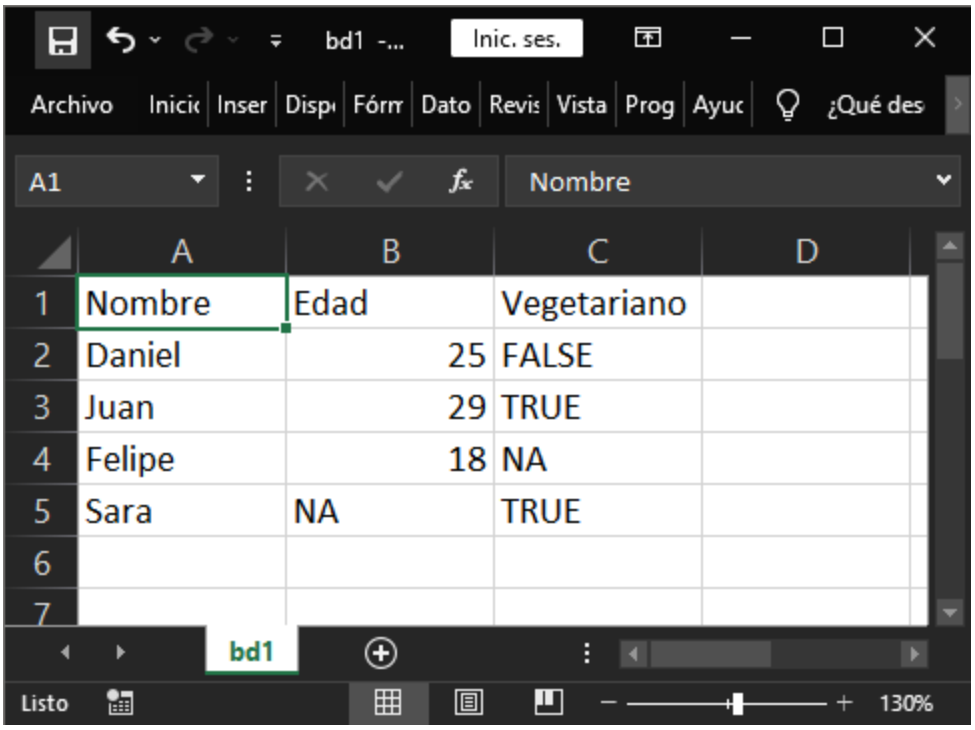


Imagen 54

Cuando se tenga listo los datos digitalizados[5], guardamos el archivo en la carpeta deseada, con un nombre sencillo para evitar malas lecturas y modificar el tipo de archivo a .**csv**.

3.3.1.2 Guardar Datos en Bloc de Notas.

Para guardar los datos que describimos en la tabla anterior en un bloc de notas, debemos abrir un nuevo archivo y copiar la información, se tiene dos formas de copia de datos, la primeria es utilizar la barra espaciadora para separar una variable o dato de otro, o con la tecla tabuladora[6], cuando finalizamos la línea debemos pulsar **enter**. en las siguientes imágenes mostraremos como se debe de almacenar esta información.

[5] Tener en cuenta que: debemos inicializar la digitalización de datos siempre en la parte superior izquierda, no se puede dejar celdas en blanco, no se puede añadir color a las celdas, no añadir bordes de ningún tipo, no adornar el contenido

[6] La única diferencia de utilizar la barra espaciadora o la tecla tabuladora es el orden que esta da, con la tabulación se ve mas ordenado la base de datos y mejora mucho más la apariencia.

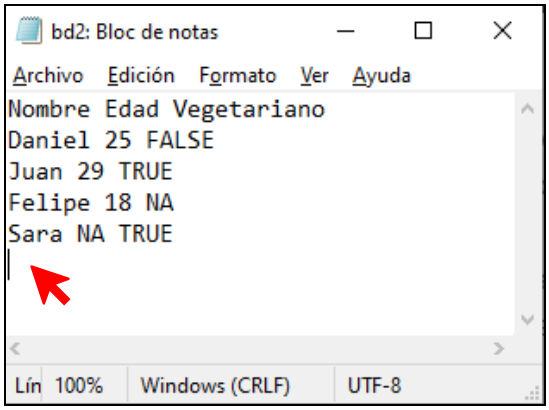


Imagen 55

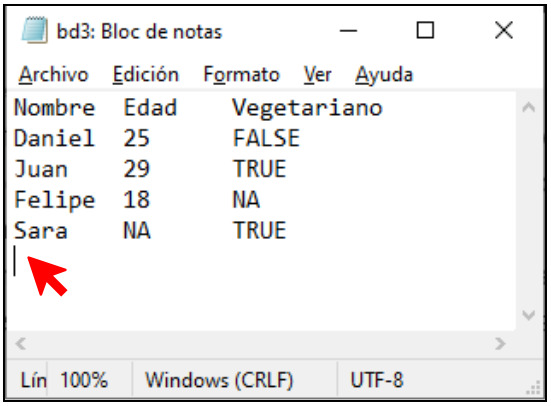


Imagen 56

Podemos ver un cursor rojo en las imágenes, esto nos está mostrando el cursor donde debe de quedar ubicado al momento de terminar de digitalizar los datos, esto sirve para que al momento de que R lea el documento, lea todos los datos hasta encontrar el cursor, si lo dejamos en la línea de arriba, esta no la leerá por completo.

3.3.2 Importación de Datos.

Para que podamos leer una base de datos directamente, el archivo externo debe de estar creado con las condiciones que explicamos en la sección "***3.3.1 Almacenamiento de Datos***".

Con esto claro, se utilizaría la función **read.table( )**, donde esta función es la básica para poder leer una base de datos hacia R, a continuación, se mostrará la estructura de la función y una tabla explicativa de los argumentos que contiene internamente

estructura: read.table(file, header, sep, dec)	
file	nombre o ruta donde este alojado el archivo. Puede ser una URL o la ruta de acceso
header	Valor lógico, donde se utilizaría TRUE si la primera línea de la base de datos tiene los nombres de las variables, en caso contrario se utilizará FALSE
sep	tipo de archivo que se desea cargar o tipo de separación interna, las instrucciones más usuales son: sep=´ , ´ documento separado con comas. sep=´ ´ documento separado con espacios. sep='\t' documento separado con tabulación.
dec	Símbolo que indica las decimales

Tabla 14

Ejemplo: Realizar la base de datos de la tabla en Excel y el bloc de notas (espaciadora y tabuladora) para ejecutar la lectura desde R.

Solución: Lo que tenemos que realizar como primer paso es crear los tres archivos solicitados, que queden igual a los mostrados en las figuras anteriores, los nombres que le colocamos a cada uno de ellos fueron: **bd1.csv**, **bd2.txt**, **bd3.txt**

3.3.2.1 Leer Base de Datos Desde Excel.

El documento de Excel fue el llamado **bd1.csv** y para leerlo utilizamos la siguiente instrucción[7].

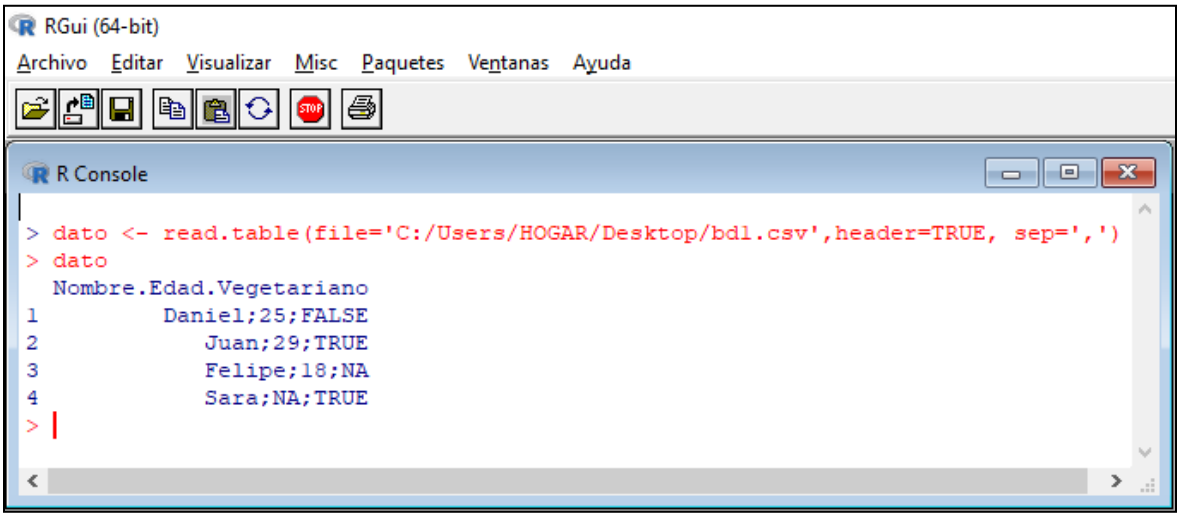


Imagen 57

La dirección que se agrego en **file="C:/Users/HOGAR/Desktop/bd1.csv"**[8] le dice a R en que parte del computador esta ubicado el documento. Si de casualidad no conocemos la ubicación del archivo que deseamos leer o es demasiado extensa, se recomienda utilizar la función **file.choose( )**[9], esta nos permite adjuntar manualmente el archivo, así como se muestra a continuación:

---

[7] El **sep** está definido por defecto con espacios

[8] Tener presente que, al momento de escribir manualmente la ubicación del archivo, este por defecto viene con el símbolo ' \ ', pero la forma correcta que debemos de digitar esta dirección es usando el símbolo ' / '.

[9] Recordar que, si conocemos la dirección del archivo, escribir el comando **file='C:/Home/Desktop/…**' o si por alguna razón no se conoce esta ubicación se debe cambiar lo anterior por **file.choose( )**.

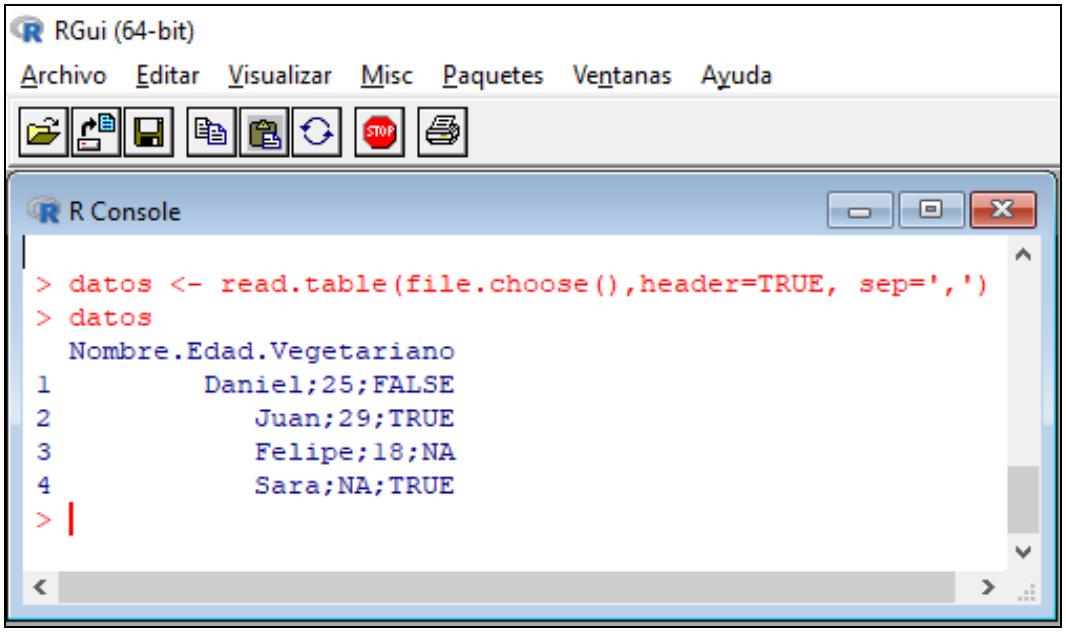


Imagen 58

3.3.2.2 Leer Base de Datos Desde Bloc de Notas con Espacios Simples.

El documento de bloc de notas con barra espaciadora fue el llamado **bd2.txt** y para leerlo utilizamos la siguiente instrucción.

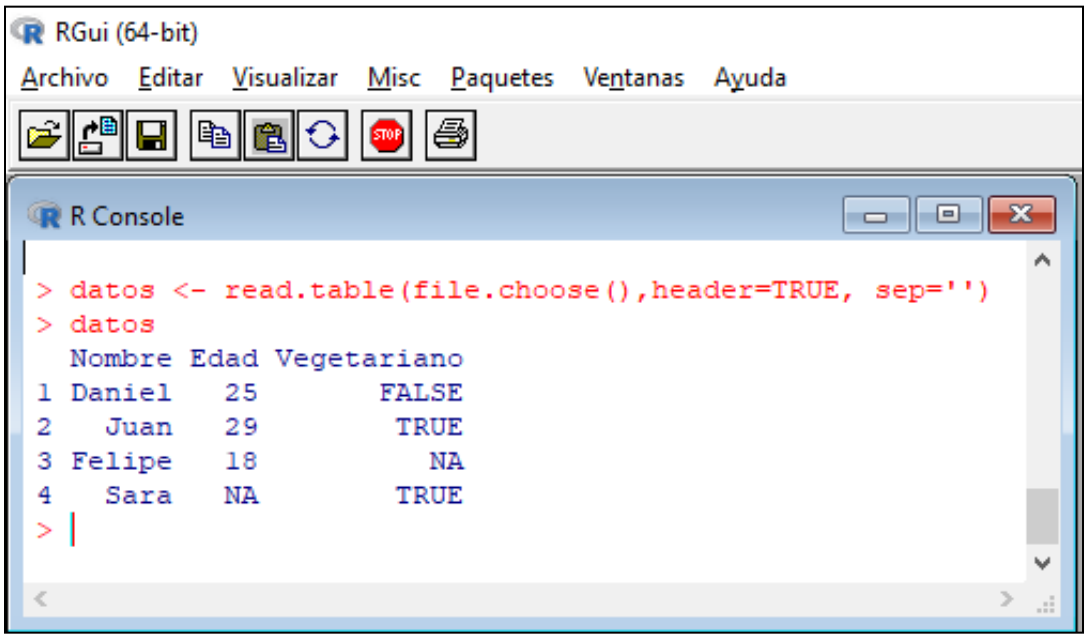


Imagen 59

3.3.2.3 Leer Base de Datos Desde Bloc de Notas con Tabulación.

El documento de bloc de notas con tecla tabuladora fue el llamado **bd3.txt** y para leerlo utilizamos la siguiente instrucción.

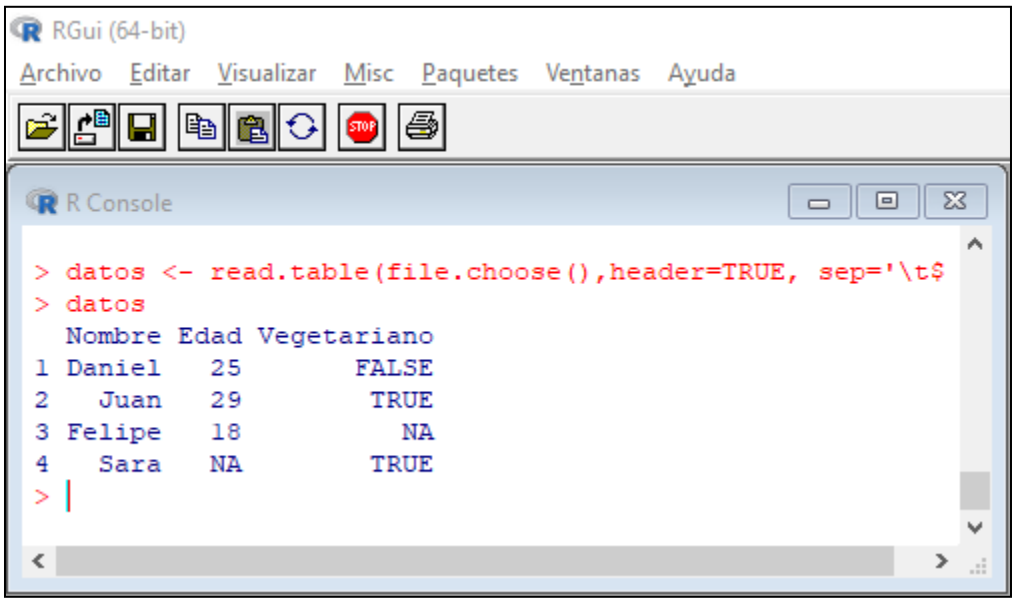


Imagen 60

3.3.3 Exportar Datos.

Para poder extraer nuestra base de datos, se deben de tener una estructura rectangular, ósea que deben de tener la misma longitud los vectores en cuestión, para exportar los datos, se utiliza la función **write.table( )**, a continuación, se explica la función y sus argumentos mas usados:

write.table(x, file, sep, row.names, col.names)	
x	Nombre del data frame a exportar.
file	Se coloca la ruta del archivo, el nombre que deseamos y la extensión. Si solo colocamos el nombre del archivo con la extensión (.csv para Excel y .txt para archivo de texto), creara el archivo en el directorio de trabajo.
sep	Carácter que se utilizara para separar las columnas. sep=´ , ´ si el documento lo quiere separado con comas. sep=´ ´ si el documento lo quiere separado con espacios. sep='\t' si el documento lo quiere separado con tabulación.
row.names	Se coloca TRUE si queremos que nuestros renglones tengan una etiqueta, como recomendación se deja FALSE, para que la base de datos sea más fácil de interpretar.

col.names	Se aconseja dejar esta condición como TRUE, para que almacene los nombres de las columnas con lo que definimos previamente.

Tabla 15

Para poder entender mejor como opera este comando para la extracción de datos, utilizaremos como ejemplo dos variables "**x**" que contendrá del 1 al 10 y "**y**" que contendrá del 11 al 20, estas las guardaremos como un **data frame** en la variable "**datos"**, dando como resultado la siguiente matriz:

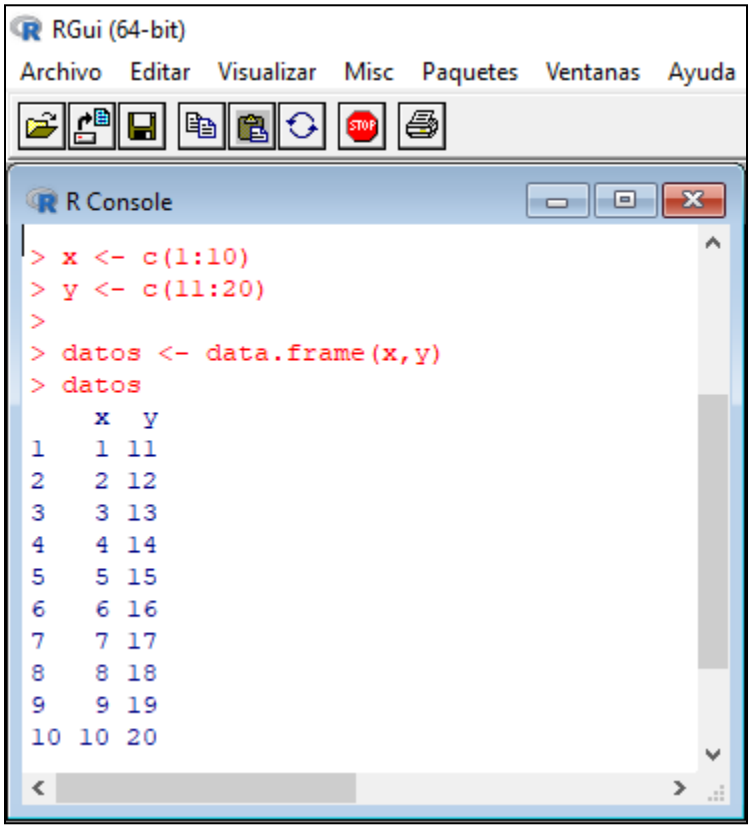


Imagen 61

Ahora exportaremos estos datos a una hoja de Excel, esta acción se hace si deseamos buscar más adelante una opción de encontrar algún dato o variable en especifico de una manera rápida o si queremos realizar alguna otra actividad con estos datos.

Para esto, utilizaremos la siguiente línea de código[10]:

[10] Para extraer los datos en un bloc de notas, se utiliza exactamente el mismo código, el único cambio en que en el nombre se le coloca una extensión "**.txt**"

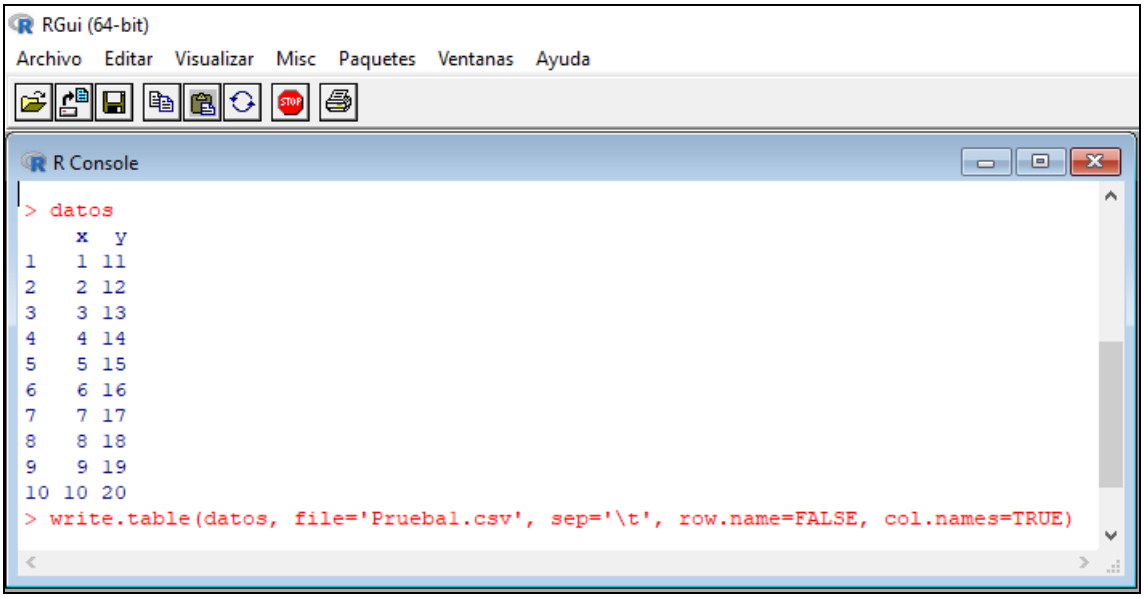


Imagen 62

podemos fijarnos que, llamamos el data frame "**datos**", que es la que contiene las variables antes creadas, le dimos un nombre al documento que crearemos[11], en este caso "**Prueba1.csv**" que será separado por "**\t**" (tabulación[12]) y definimos como verdadera la "**col.names**" para que nos aparezca el nombre de las columnas, dando como resultado lo siguiente:

[11] Los pasos que se realizaron para organizar la tabla de Excel, estaba con los datos almacenados separados con tabulación "**\t**"

[12] Se recomienda trabajar siempre con espacios tabulados (Recordar que es con la opción "**sep = '\t'**") para ver y trabajar mejor con los datos extraídos.

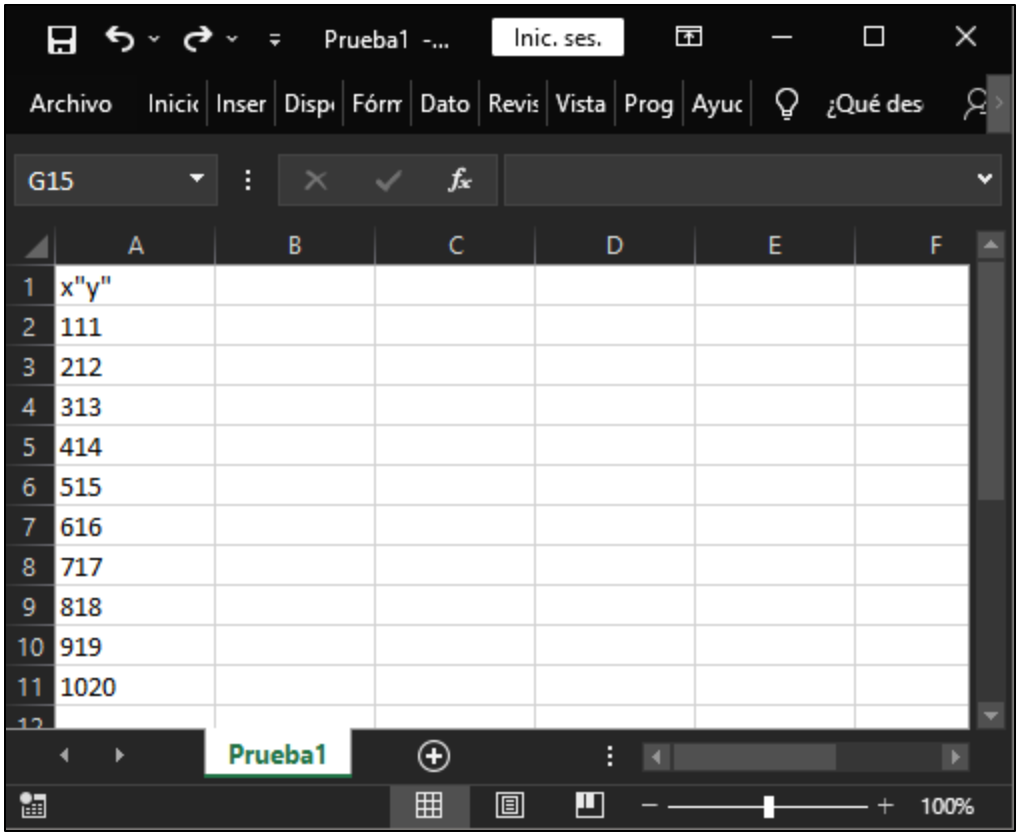


Imagen 63

Como podemos observar no queda muy agradable a la vista[13], así como esta, tendríamos que abrir celda por celda para observar que dato es el correspondiente para cada columna, dándonos cuenta que está separado cada dato por una tabulación, así como veremos a continuación:

[13] Como sugerencia, se recomienda dejar el archivo de Excel lo más básico posible, sin marcar los bordes, sin colorear, sin combinar celdas, todo esto por si queremos en un futuro importar estos datos de nuevo a R, no tengamos algún error de lectura.

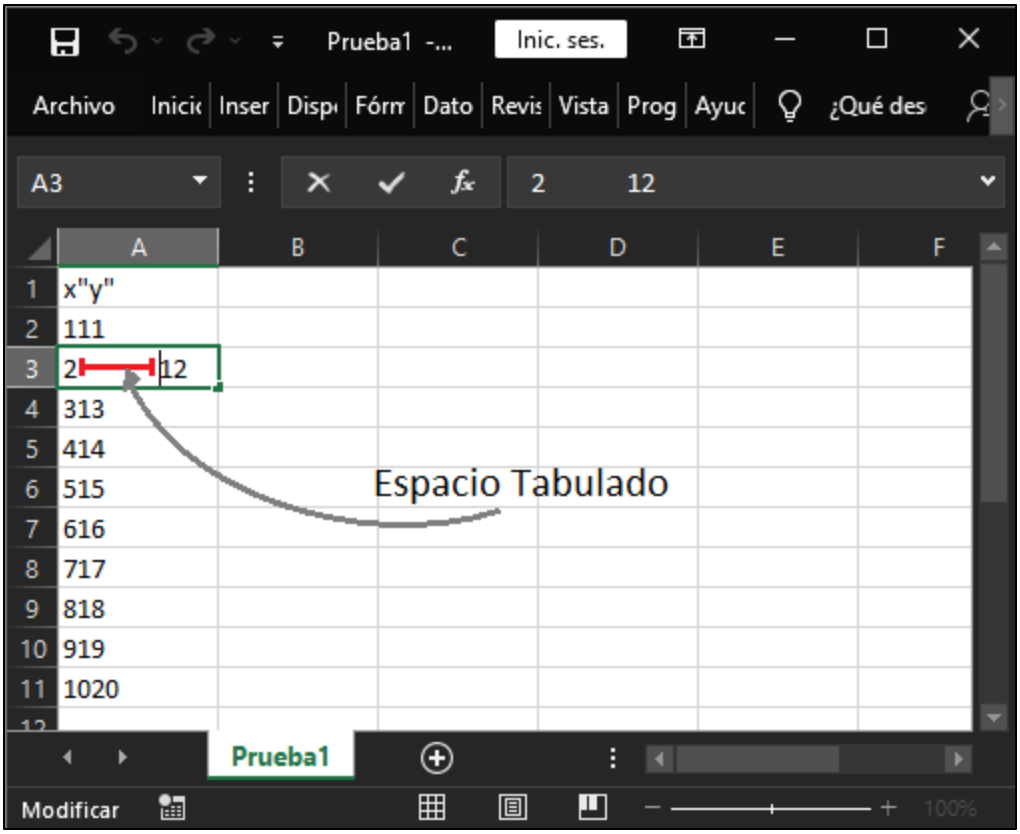


Imagen 64

Para dejar esta tabla mas organizada y agradable para el usuario, se recomienda seguir estos pasos:

- Seleccionar toda la tabla de datos.
- En la parte superior donde dice "**Datos**" dar clic
- Dar clic donde dice "**Texto en columnas**"

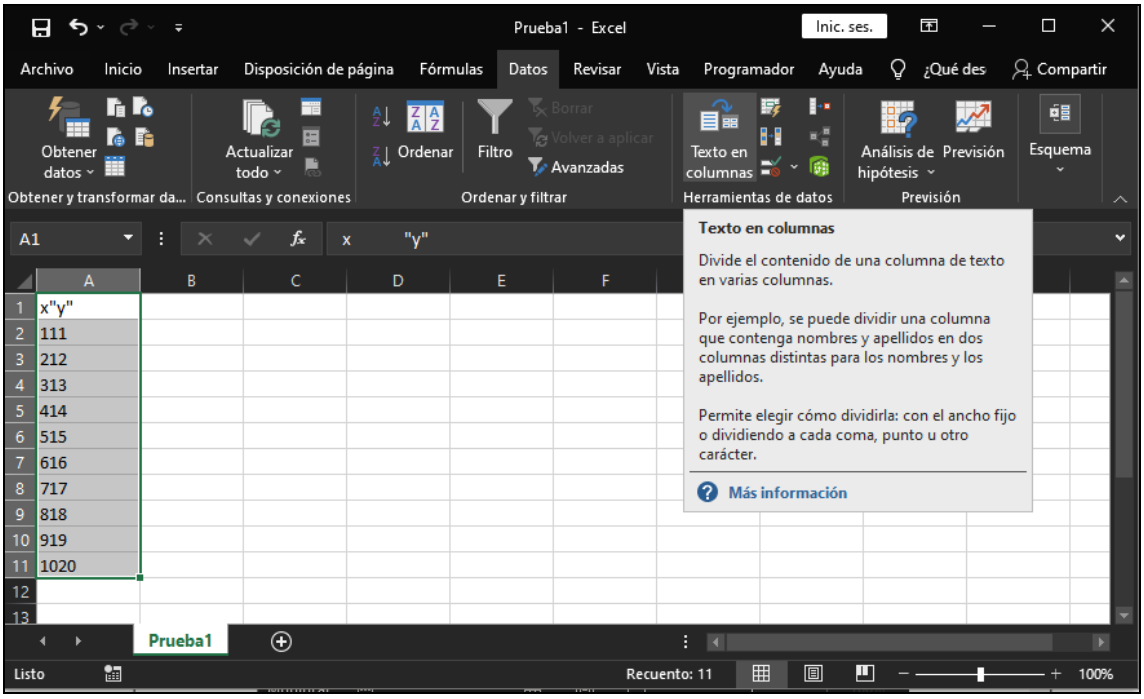


Imagen 65

- Aparecerá una ventana emergente y de ahí seleccionamos la opción de "**Delimitados**" pulsamos siguiente
- En la parte de "**Separadores**" seleccionamos que por "**Tabulación**" pulsamos siguiente
- En la parte de "**Formato de los Datos en Columnas**" seleccionamos "**General**" y pulsamos finalizar

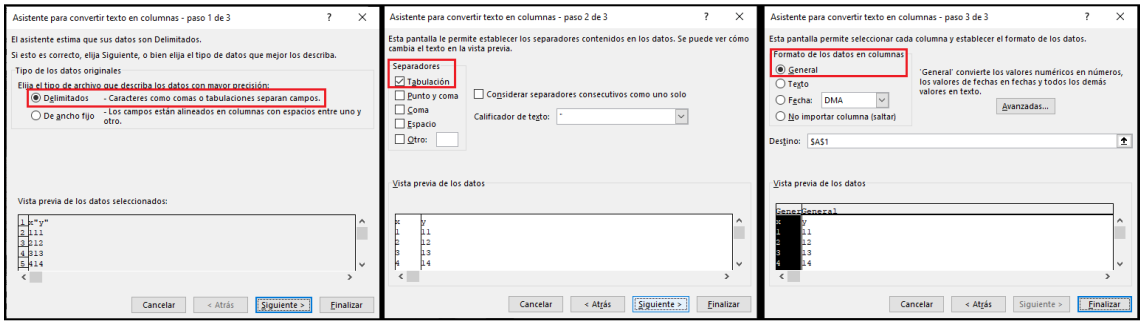
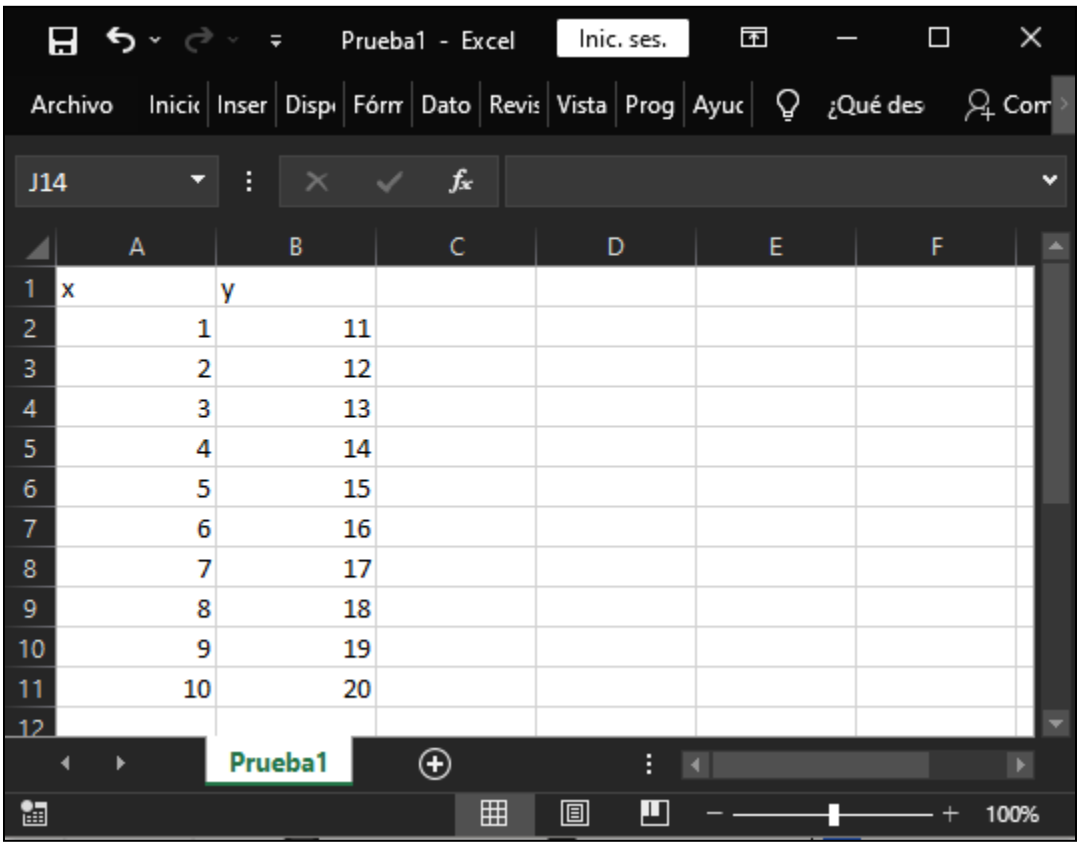


Imagen 66

Siguiendo los pasos anteriores, el documento tomara la siguiente presentación, como resultado vemos algo mucho más agradable a la vista y más entendible al momento de querer leer los datos:



	A	B
1	x	y
2	1	11
3	2	12
4	3	13
5	4	14
6	5	15
7	6	16
8	7	17
9	8	18
10	9	19
11	10	20
12		

Imagen 67

3.4 Graficas en R

3.4.1 Introducción a los gráficos en R

Para tener una mejor presentación de nuestros estudios estadísticos, y que sea de una forma que se pueda entender o explicar de una manera más eficaz dichos resultados de nuestra investigación o trabajo, siempre se es mejor realizar una gráfica para ver el comportamiento de estos cálculos estadísticos, ya que sabemos que una imagen dice más que mil palabras, es por esto que el Lenguaje R, nos ofrece una programación fácil y ágil para poder crear unas graficas eficientes y efectivas para mostrar mejor nuestros resultados.

Es por esto que en este capítulo hablaremos sobre las gráficas principales que tiene a la disposición R y como podremos trabajar con ellas. A continuación, mostraremos un adelanto de las gráficas que podemos trabajar con el comando ***demo(graphics)***

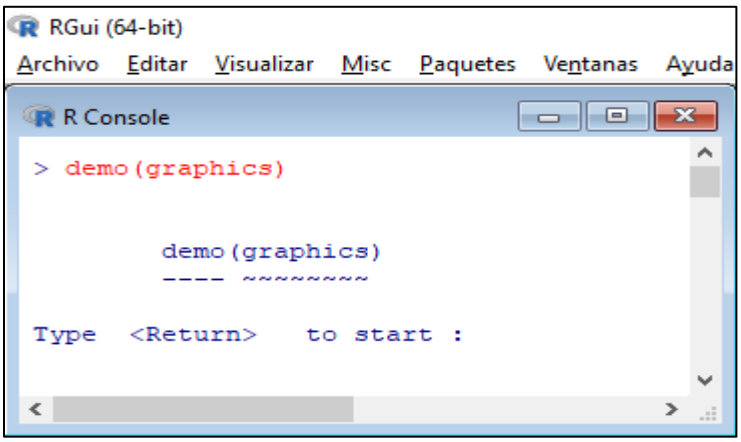


Imagen 68

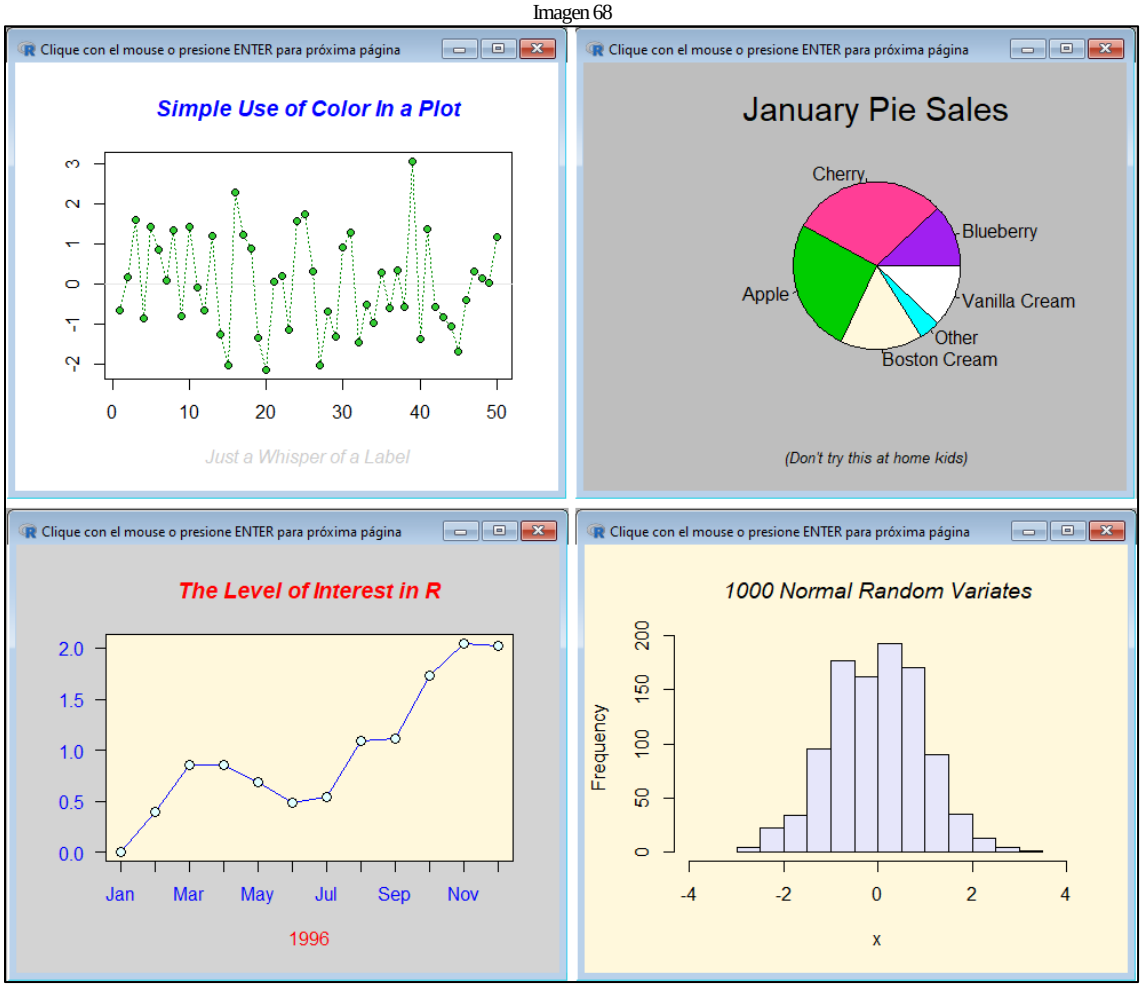


Imagen 69

Las gráficas que el lenguaje de programación de R nos ofrece, las utilizamos principalmente, para comprender y representar de mejor manera los datos que estamos utilizando, tenemos tres formas de crear estas gráficas, estas son:

- Gráficos básicos de R: La función "*plot*" es la función más básica de R para crear gráficos, con esta función podemos crear diferentes tipos de graficas como por ejemplo las gráficas de barras, de líneas o podemos crear hasta matrices de dispersión o diagramas de caja. esta función la encontramos en el paquete "*graphics*"
- Librería ggplot2: Técnicamente ggplot2 es un paquete de visualización de datos que tiene una estructura especial, donde primero debemos de definirle con que datos hay que trabajar, se le añade la estética de cómo vamos a presentar esos datos, se le añade unas capas de datos que deseamos presentar, y si queremos presentar varios gráficos, uno al lado de otros, añadimos unas facetas.
- Paquete Lattice: Este paquete ofrece una flexibilización especial, para representar interacciones y datos multivariados. Las funciones que tiene, son ligeramente distintas de las funciones de graficas básicas, además de presentar esto mismo para la gramática.

3.4.2 Tipos de Gráficos

3.4.2.1 Histograma

Un histograma se representa gráficamente, por la distribución de frecuencias de una variable en específico, estas variables son representadas por unas barras, y cada barra tiene una altura delimitada por la cantidad de veces que se repite el mismo valor de la variable, además de que esta variable tiene un rango determinado.

En la siguiente tabla se explicara la línea de código y que debemos de tener presente para poder crear el histograma:

hist(x = " ", main = " ", xlab = " ", ylab = " ", col = " ")	
x	Vector numérico
main	Titulo de la grafica
xlab	Nombre del eje X
ylab	Nombre del eje Y
col	Color de la columna (ya sea por nombre en ingles o forma hexadecimal"

Antes de arrancar, debemos tener presente que mostraremos los ejemplos de como funcionan esta grafica haciendo el uso de una base de datos llamada "valores.csv", que la puede descargar en el siguiente enlace[14], donde podemos ver un conjunto de personas, con su respectivo equipo, nombre, barrio donde viven, compañía y un saldo:

https://github.com/dfq725/AnaliticaConR/blob/main/valores.csv

En la siguiente imagen podemos ver como llamamos el archivo para guardarlo en un vector, de ahí, con el comando hisp() anteriormente explicado[15], realizamos el llamado[16] de la columna equipo, para poder crear nuestro histograma, le añadimos como título del histograma "Histograma de Saldos" y un nombre en el eje X, llamado "saldo", y un coloro rojo para verlo más presentado.

[14] Se recomienda guardar el documento en el escritorio, para tener un mejor acceso a el

[15] En la sección 3.3 tratamos ya el tema de como guardar un archivo .csv en un vector para después ser leída

[16] Para poder que el vector extraiga del documento "valores" los datos de la columna equipo, se usa el signo de dólar "$".

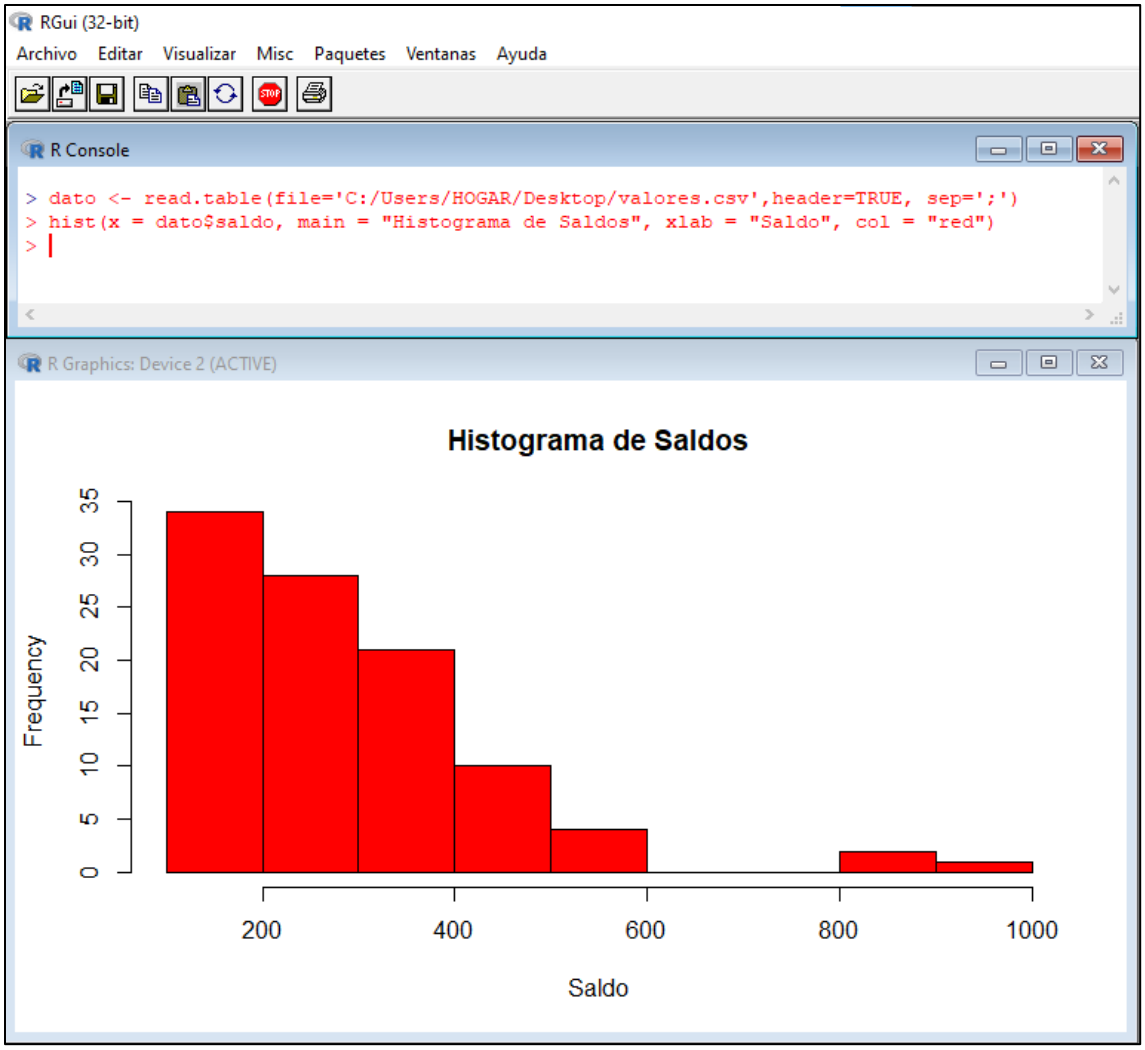


Imagen 70

3.4.2.2 Grafica de Barras

Esta grafica es la más conocida y usada de todas. Esta consiste básicamente en que nos muestra la cantidad de veces con la que se ha leído una variable en particular, colocando cada lectura en una barra para cada una de la categoría en cuestión.

Se usa más que todo, cuando queremos saber que variable es la que más se repite, dando a entender que podemos enfocarnos en esa porque tiene más datos y realizar un estudio, o porque nos da una información del comportamiento de una población.

La función barplot() nos permite crear esta grafica de barras, pidiendo como parámetro unas variables que explicaremos a continuación:

barplot(x = “ “, main = “ “, xlab = “ “, ylab = “ “, col = “ “)	
x	Vector numerico
main	Titulo de la grafica
xlab	Nombre del eje X
ylab	Nombre del eje Y
col	Color de la columna (ya sea por nombre en ingles o forma hexadecimal”

Ejemplo: Se realizo una encuesta a diferentes usuarios de una empresa de transporte masivo, donde se les preguntaba qué tan buena era la calidad del servicio que esta empresa les prestaba, los encuestados tenían tres opciones de respuesta 1 era bueno, 2 era aceptable o 3 era malo, los usuarios podían responder cualquiera de estas tres opciones, los datos recolectados, fueron los siguientes:

	Nombre	Calificacion
1	Daniel	1
2	Juan	1
3	Felipe	2
4	Sara	1
5	Dayana	3
6	Davier	1
7	Pedro	2
8	Maria	2
9	Angela	1
10	Patricia	2

Deseamos realizar un diagrama de barras para leer mucho mejor los resultados de estas respuestas, para esto, realizamos el siguiente código:

Imagen 71

Para que quede mejor presentado, le añadiremos los nombres correspondientes y le aplicaremos un color para diferenciarlos, quedando asi:

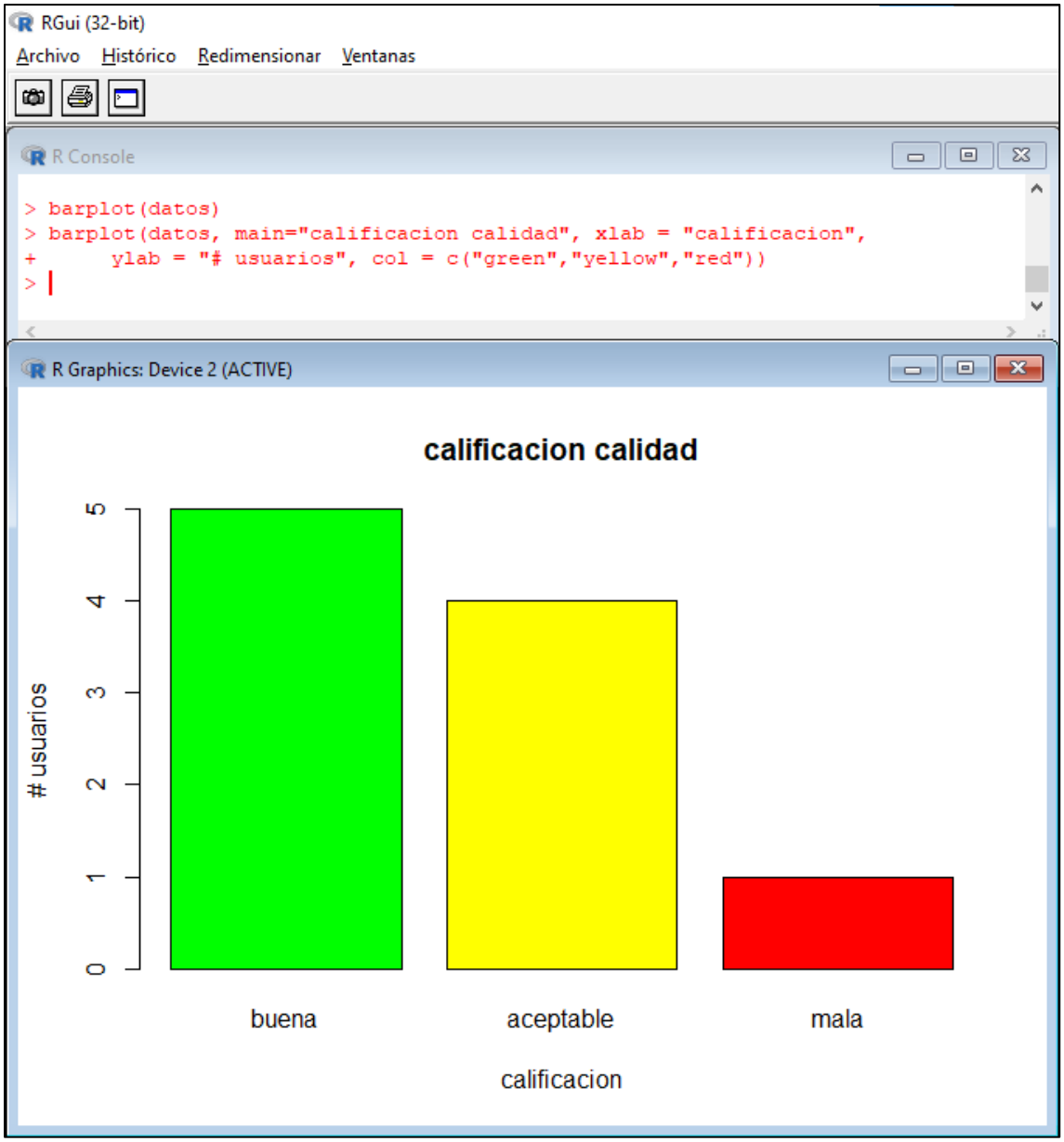


Imagen 72

3.4.2.3 Diagrama de Dispersión

Este Diagrama se utiliza comúnmente cuando deseamos ver la relación que tienen dos variables numéricas, esta crea una serie de puntos, dando una imagen de donde se ve la mayor población en esa relación, se usa para determinar en qué campo hay más cantidad de datos, y poder trabajar con este.

La función plot() nos permite crear este Diagrama de Dispersión, pidiendo como parámetro unas variables que explicaremos a continuación:

plot(x = “ “, main = “ “, xlab = “ “, ylab = “ “, col = “ “)	
x	Vector numerico
y	Vector numerico
main	Titulo de la grafica
xlab	Nombre del eje X
ylab	Nombre del eje Y
col	Color de la columna (ya sea por nombre en ingles o forma hexadecimal”

Ejemplo, con el documento descargado en la sección del Histograma, realizaremos la relación que hay en la columna de la edad y del saldo, ya que buscamos determinar en que rango de edad la población tiene un saldo X, para determinar como darle solución a que se pongan al dia con estos.

Para poder darle solución a esto, recordemos que con el símbolo dólar ($) podemos extraer los datos de la columna que deseamos, con esto presente y lo explicado anteriormente, se tiene:

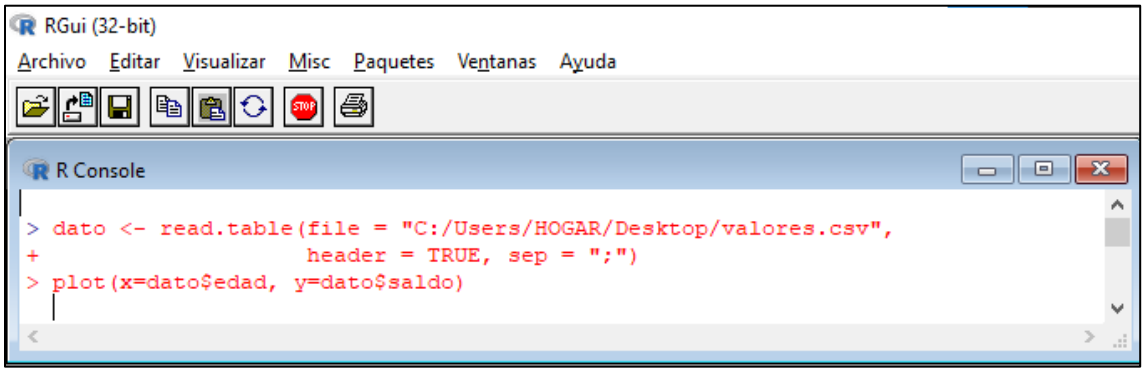


Imagen 73

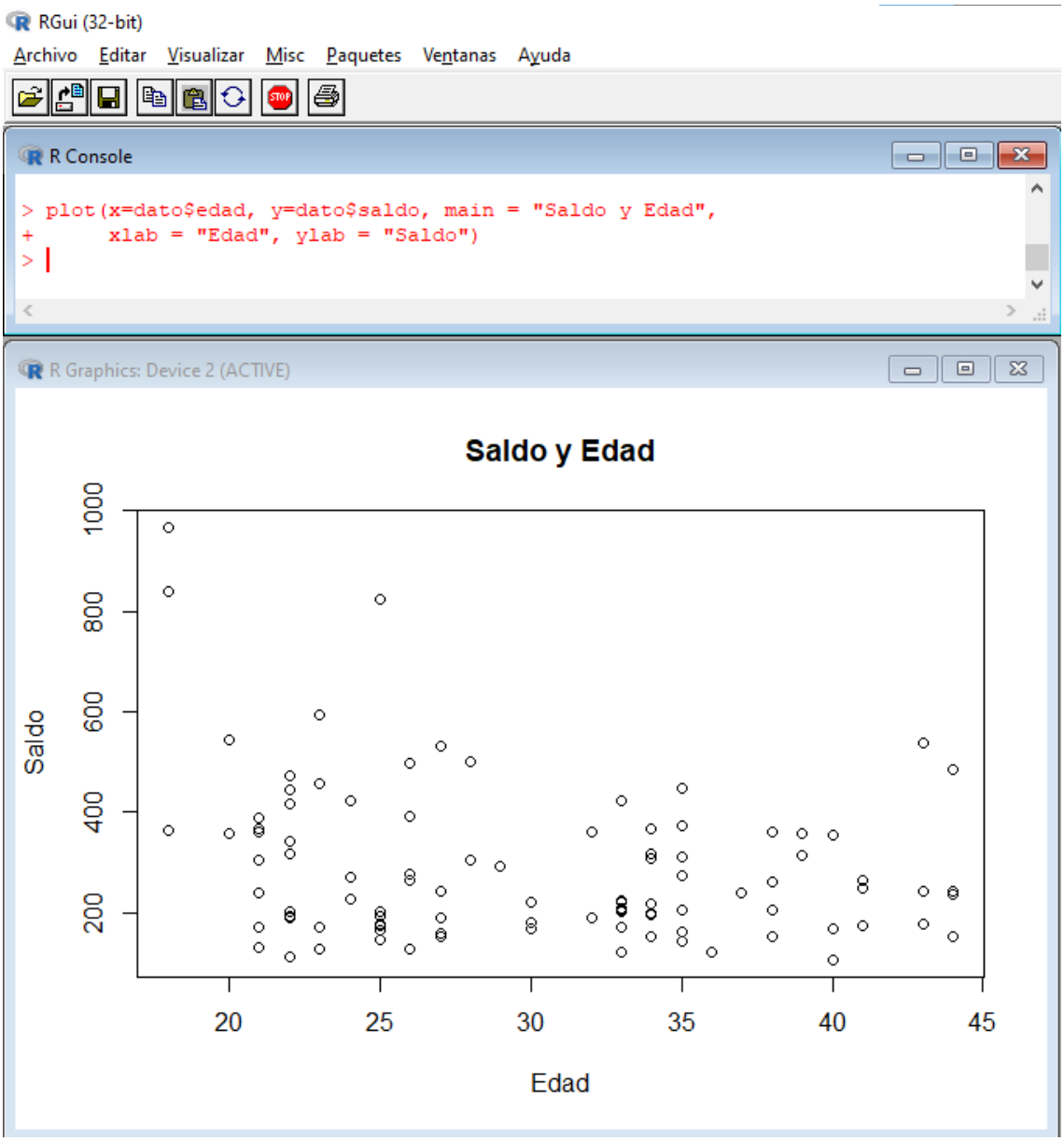


Imagen 74

3.4.2.4 Diagrama Circular

El grafico circular[17] es también conocido como grafico de sectores o grafico de torta (en inglés pie) este es un gráfico que nos representa porcentajes o proporciones en sectores, donde el área de cada sector y longitud de este corresponde a la cantidad de datos representados.

[17] Este diagrama es uno de los más complejos de leer al momento de querer dar una explicación de nuestros resultados, ya que en ocasiones los cambios no son muy visibles a la vista.

La función pie() nos permite crear este Diagrama de Dispersión, pidiendo como parámetro unas variables que explicaremos a continuación:

pie(x = “ “, main = “ “, xlab = “ “, ylab = “ “, col = “ “)	
x	Vector numerico
y	Vector numerico
main	Titulo de la grafica
xlab	Nombre del eje X
ylab	Nombre del eje Y
col	Color de la columna (ya sea por nombre en ingles o forma hexadecimal”

Ejemplo: Se realizo una encuesta a diferentes usuarios de una empresa de transporte masivo, donde se les preguntaba que tan buena era la calidad del servicio que esta empresa les prestaba, los encuestados tenían tres opciones de respuesta 1 era bueno, 2 era aceptable o 3 era malo, los usuarios podían responder cualquiera de estas tres opciones, los datos recolectados, fueron los siguientes:

	Nombre	Calificacion
1	Daniel	1
2	Juan	1
3	Felipe	2
4	Sara	1
5	Dayana	3
6	Davier	1
7	Pedro	2
8	Maria	2
9	Angela	1
10	Patricia	2

Realizar el diagrama circular para leer estos datos:

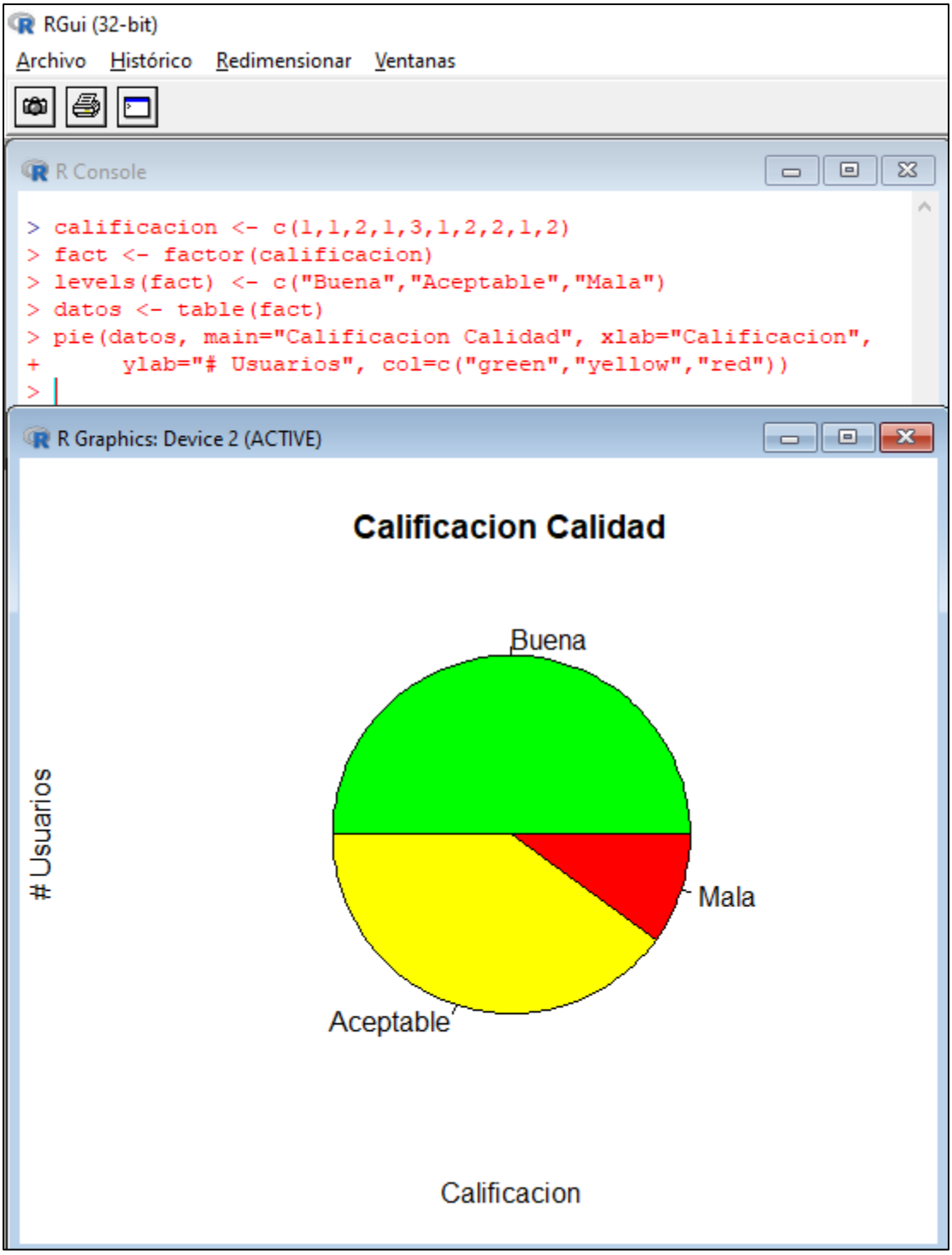


Imagen 75

Si por alguna razón, queremos que nos aparezca la cantidad de personas que seleccionaron una de esas opciones, adicionamos el argumento **labels**, quedando de la siguiente manera:

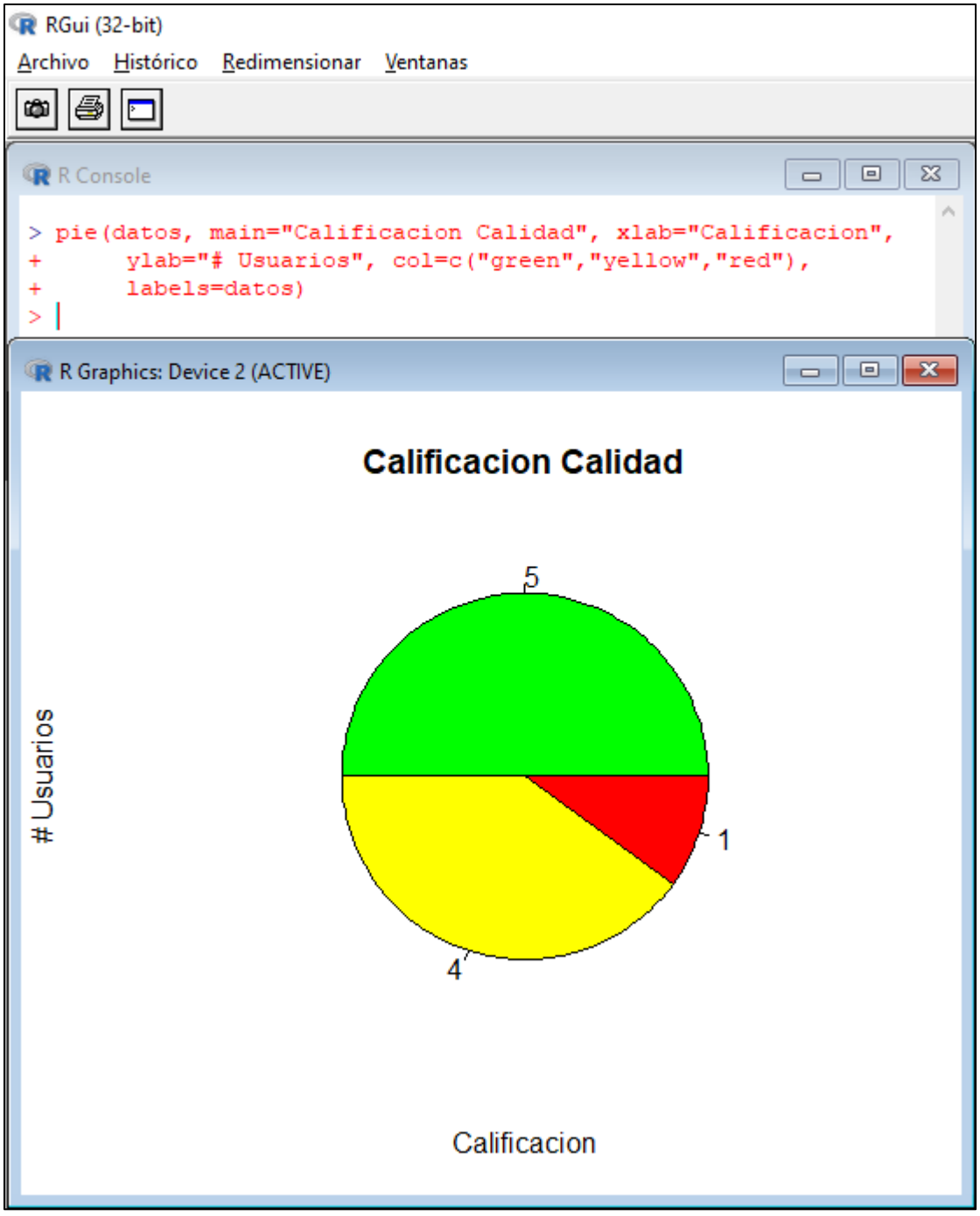


Imagen 76

3.4.3 Exportar Gráficos

Una buena idea al momento de trabajar gráficos en R, es tener principalmente el conocimiento de cómo exportar estos gráficos a nuestro disco duro, esto con el objetivo de que

queden almacenados en nuestra memoria y podamos acceder más adelante a ellos de una manera más rápida y eficiente.

Este proceso puede parecer algo confuso, pero poniéndolo en práctica, se dará de cuenta que es algo simple y rápido.

3.4.3.1 ¿Por qué exportar los gráficos?

Cuando queremos llamar unas de estas funciones (graficas), le estamos informando a R que "mande" nuestro diagrama o grafico a un dispositivo grafico (Graphic Device) en nuestro computador, donde nos mostrara la imagen, que por defecto es una ventana secundaria.

Una falencia de esta forma de operar de R, es que, si creas esta función y lo manda a el dispositivo gráfico, el grafico que creo remplazara el anterior, por ejemplo, si realizamos el llamado a la función **pie()** para crear el grafico circular, y anteriormente teníamos una función **plot()** de un diagrama de barras, el contenido de este último se reemplazara, perdiendo así estos datos graficados. Esto ocurrirá siempre que queremos crear un gráfico con este mecanismo.

Ejemplo: En las siguientes imágenes vemos que se actualizo la ventana **Graphic Device**, con la gráfica **pie()**, estando anteriormente la función **plot()**.

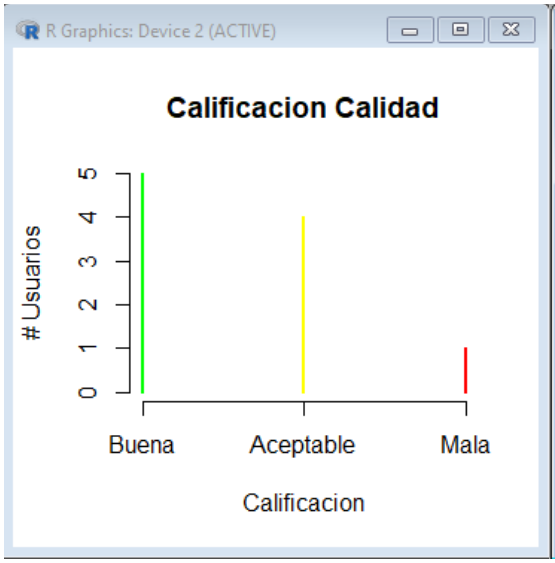


Imagen 77

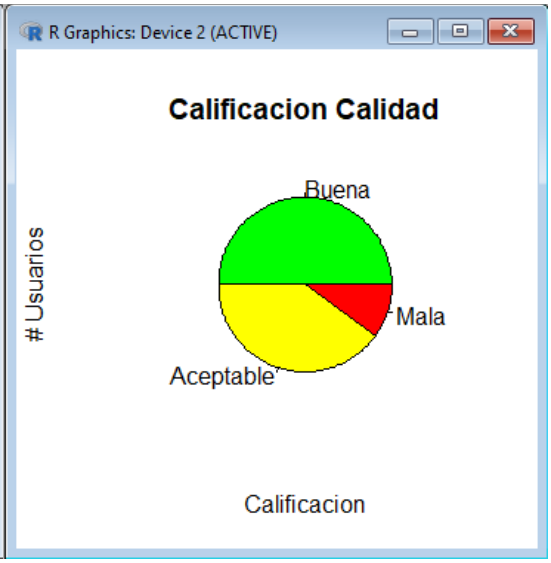


Imagen 78

3.4.3.2 ¿Cómo exportar mi grafica?

Para exportar nuestra gráfica, utilizaremos cualquiera de las siguientes funciones, cada una de estas, corresponde a un archivo diferente, estas no son las únicas, pero si son las más comunes:

- bpm()
- jpeg()
- pdf()
- png()
- tiff()

Cada una de las funciones que nombramos, deben de llevar los siguientes argumentos obligatoriamente:

filename	Nombre y ruta[18] del archivo a crear.
width	El ancho de la imagen a crear, en pixeles
height	El alto de la imagen a crear, en pixeles

Debemos de llamar esta función antes de utilizar la función que genere la gráfica, al realizar esto, le indicamos a R que, en vez de mandar el diagrama a la ventana emergente por defecto, lo mande a donde le indicamos y lo almacene en nuestro dispositivo.

Al finalizar esto, ósea, el llamado a la función de exportación y al llamado a la función de la gráfica, debemos de poner el siguiente código: **dev.off()**; esto para cerrar el dispositivo grafico y proceder a que R cree el archivo y poder así crear mas diagramas o gráficos sin temor a que modifique el que creamos anteriormente.

Ejemplo: Exportar[19] el grafico circular de la sección 4.2.3.

[18] Si no se coloca la ruta, el archivo será creado en el directorio de trabajo
[19] Para exportar el grafico con un archivo, tenemos que dar la extensión de archivo de la función que estamos llamando.

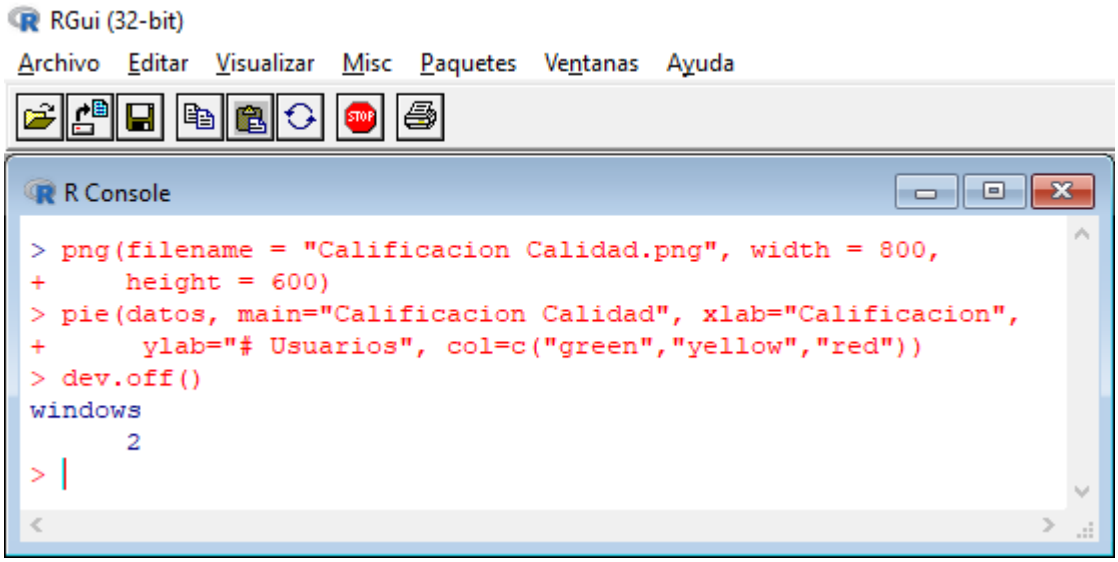


Imagen 79

Resultado[20]:

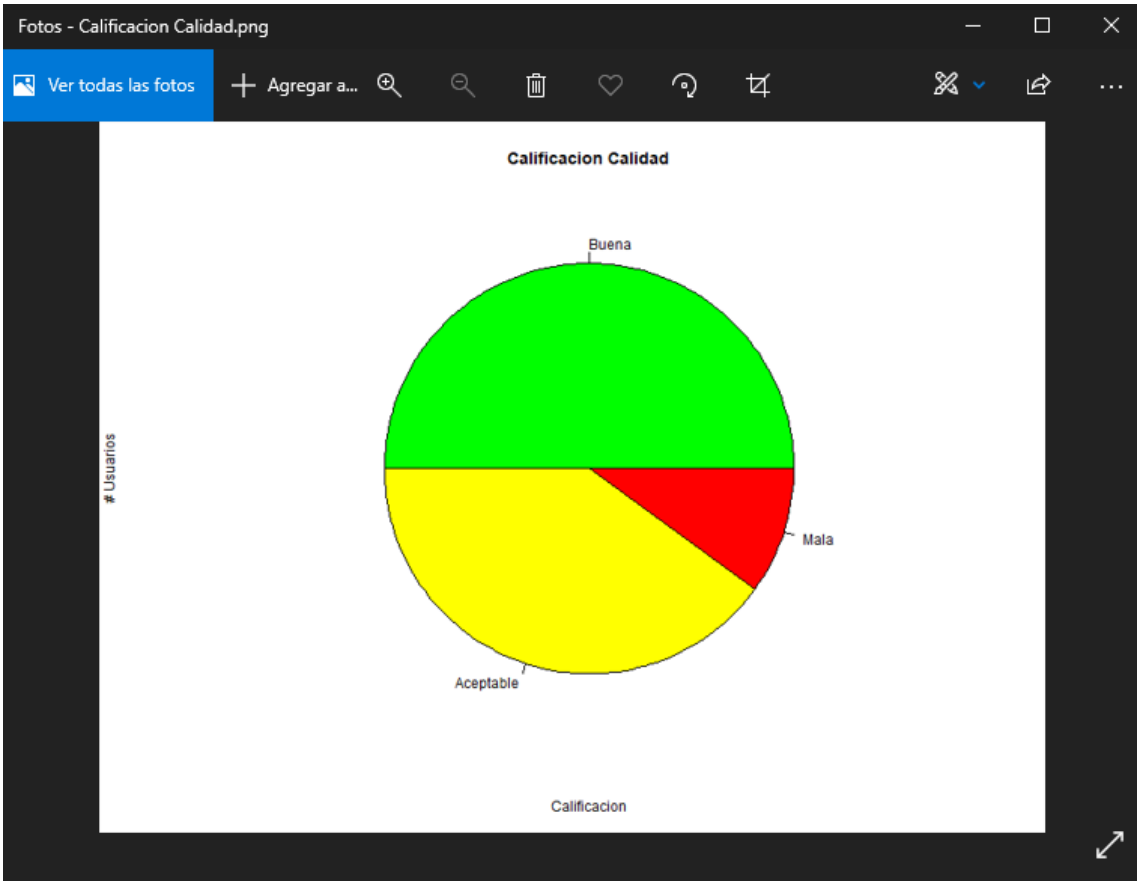


Imagen 80

[20] Para ver el resultado en R, utilizamos file.show("NombreArchivo.extencion").

Capitulo IV: Test de Conocimiento

En este capitulo se pretende a través del planteamiento de ejercicios prácticos, introducir al estudiante, en la analítica y visualización de datos haciendo uso de las herramientas mostradas en los capítulos anteriores, estos ejercicios abarcan todo lo que se explico en el transcurso de cada capitulo y no se saldrá del contexto del mismo.

4.1 Ejercicio 1

Realizar las siguientes operaciones:

a) 1049 + 256 + 2234 + 21223

b) 589 – 2122 + 9029

c) 5.49 + 789.45

d) $50 * \frac{223}{212}$

e) $\frac{30}{40} * 25$

f) 5^{23}

g) 25^3

h) $\sqrt{245}$

i) $log_{10}25$

4.2 Ejercicio 2

- Se realizo una encuesta a un conjunto de personas donde se les pidió los siguientes datos, un género en donde la “h” es de hombre y la “m” es de mujer y la edad, obteniendo como resultado la siguiente tabla:

Genero	Edad
h	25
h	28
m	35
h	32

m	34
h	32
h	33
m	33
h	31
m	29

Guardar estos datos en una variable “genero” y “edad” correspondientemente, crear el data.frame llamado “encuesta” e imprimir estos datos en orden de menor edad a mayor edad.

- Complementar el pinto anterior con dos nuevas variables (en el mismo orden de secuencia), e incorporarlas al data.frame, las nuevas variables serían las siguientes:

Peso	Sobrepeso
77	no
90	si
55	si
85	si
49	no
105	no
88	no
50	no
93	si
47	no

Realizar:

a) Imprimir cuales son las personas que tienen sobrepeso

b) Imprimir las personas que están por encima de 90 kilos de peso

c) Imprimir la edad máxima registrada

d) Imprimir la edad mínima

e) Imprimir la muestra media de las edades

f) Imprimir la desviación estándar del peso, con dos decimales

g) Realizar un diagrama de barras utilizando la variable Genero y Sobrepeso

h) Realizar un Histograma con la variable Peso

i) Exportar cada una de las gráficas anteriormente realizadas

4.3 Ejercicio 3

Importación y exportación de datos

Para realizar este ejercicio, debemos ingresar al enlace que aparece a continuación:

https://github.com/dfq725/AnaliticaConR

Ahí debemos descargar el documento "pacientes.txt" y guardarlo[21] en nuestro disco.

a) Importar los datos del documento "pacientes.txt" en R para trabajar con estos y realizar los siguientes procedimientos
b) Imprimir solo las variables de la tabla
c) Mostrar en pantalla la tercera variable de la tabla
d) Imprimir las ultimas 6 filas
e) Imprimir la edad de la quinta persona, la séptima y última (comandos aparte)
f) Imprimir el género y el colesterol de la persona octava, a la décima.
g) Imprimir todas las personas, pero solo las variables Diabetes y Colesterol
h) Exportar estos datos en un documento .csv y corroborar[22] que hayan quedado guardados correctamente.

4.4 Ejercicio 4

Para este ejercicio, necesitaremos el archivo "valores.csv" con el que trabajamos en la sección 3.4.2 Tipos de Gráficos, de aquí necesitaremos los datos almacenados y poderle dar solución a este ejercicio:

[21] Se recomienda guardar el documento en el directorio donde estamos trabajando, o en el escritorio para tener un mejor acceso a el

[22] Se corrobora haciendo el llamando el documento e imprimiendo los datos, puede utilizar cualquiera de los puntos anteriormente realizados en este ejercicio.

En la base de datos que extraemos del documento “valores.csv” la empresa que realizo esta recolección de datos, desea poder realizar diferentes graficas con diferentes variables de la tabla, para poder realizar una presentación a las directivas y que estas puedan entender mejor estos gráficos, la empresa nos solicita las siguientes graficas almacenadas en la memoria del computador.

Los diagramas[23] con sus variables, son:

a) Histograma de saldos
b) Diagrama de barras de cuantas personas hay en los diferentes barrios del equipo 2
c) Diagrama Circular de cuantas personas están vinculadas a diferentes operadores en el equipo 5
d) Diagrama de Dispersión para ver que saldos tienen los usuarios con respecto a la edad
e) La directiva desea conocer cuál es el operador con mayor demanda en el barrio Barajas y Sauces III (en gráficos diferentes)

[23] Los diagramas deben de tener color, titulo principal y títulos de las coordenadas

Conclusiones

Con la realización de este proyecto logre aplicar gran parte del conocimiento adquirido en mi carrera, al estructurar de manera pedagógica la guía. La componente pedagógica de la guía esta desde la perspectiva que, como estudiante, recibí en mis clases de lenguajes de programación. El paso a paso por el cual tuve que transitar en el aprendizaje, me sirvió ahora para plasmarlo en la estructura de esta guía.

El planteamiento desde la perspectiva de ingeniería en los problemas de análisis y visualización de datos, se logro gracias a los conocimientos adquiridos durante la formación que como ingeniero de sistemas recibí en la universidad.

La aplicación de este proyecto en desarrollo sostenible, se puede lograr, si el estudiante enfoca su conocimiento adquirido a través de la guía, en analítica y visualización de datos para la toma de decisiones en el ámbito ecológico.

Esta guía puede ser mejorada en un futuro mediante el planteamiento de distintos ejercicios de aplicación especifica a cada rama de la ingeniería.

Muchos éxitos y recuerda que “El Aprendizaje Nunca Agota la Mente”

Bibliografía

- Paradis, E. (3 de marzo de 2003). *R para principiantes.* Obtenido de Project: https://cran.r-project.org/doc/contrib/rdebuts_es.pdf
- Gonzales, A. (16 de mayo de 2000). *Introducción a R.* Obtenido de Project: https://cran.r-project.org/doc/contrib/R-intro-1.1.0-espanol.1.pdf
- Correa, J. y Gonzales, N, (2002). *Gráficos Estadísticos con R.* Obtenido de Project: https://cran.r-project.org/doc/contrib/grafi3.pdf
- Diaz, R. (2003). *Introducción al uso y programación del sistema estadístico R*. Obtenido de Cran: http://cran.uni-muenster.de/doc/contrib/curso-R.Diaz-Uriarte.pdf
- Gil, C. (22 de abril de 2018). *R para profesionales de los datos: Introducción.* Obtenido de Data Analytics: https://datanalytics.com/libro_r/_main.pdf
- Santana , J. y Farfán, E. (27 de noviembre de 2014). *El arte de programar en R.* Obtenido de Project: https://cran.r-project.org/doc/contrib/Santana_El_arte_de_programar_en_R.pdf
- Contento, M. (2019). *Estadifica con aplicaciones en R.* Obtenido de Editorial Utadeo: https://www.utadeo.edu.co/sites/tadeo/files/node/publication/field_attached_file/libro_estadistica_con_aplicaciones_en_r_def_ago_11.pdf
- Charte, F. (20 de agosto de 2014). *Análisis exploratorio y visualización de datos con R.* Obtenido de Fcharte: http://www.fcharte.com/libros/ExploraVisualizaConR-Fcharte.pdf