2022

LABORATORIO DE CIENCIAS BASICAS

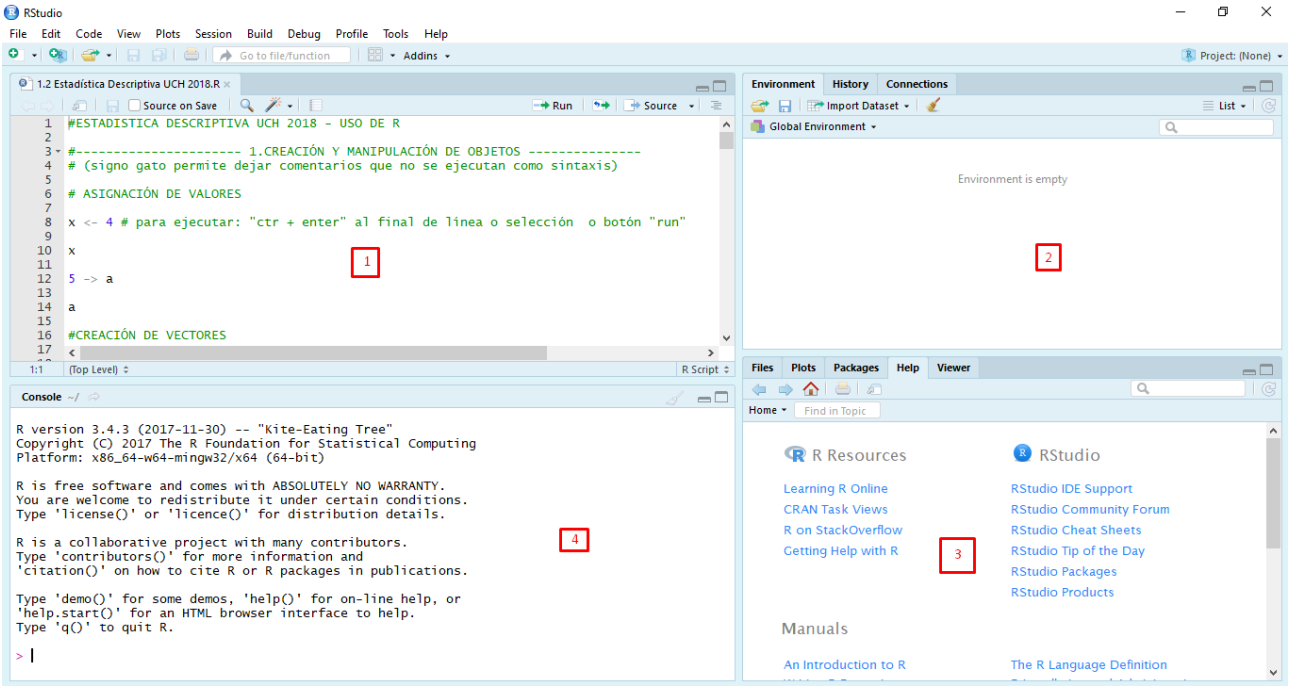
Universidad Distrital Francisco José de Caldas

Facultad Tecnológica

Tabla de Contenido

Introducción Rstudio

Para contar con más herramientas de apoyo en el uso de R emplearemos el software RStudio. Este software es una interfase - entre otras existentes como RCommander - que permite contar con una interacción más fluida con el programa R. Básicamente se trata de una máscara para visualizar el software que tiene como principales ventajas (1) el orden y (2) la visualización de los procesos que son llevados a cabo con R, todo de manera simultánea.

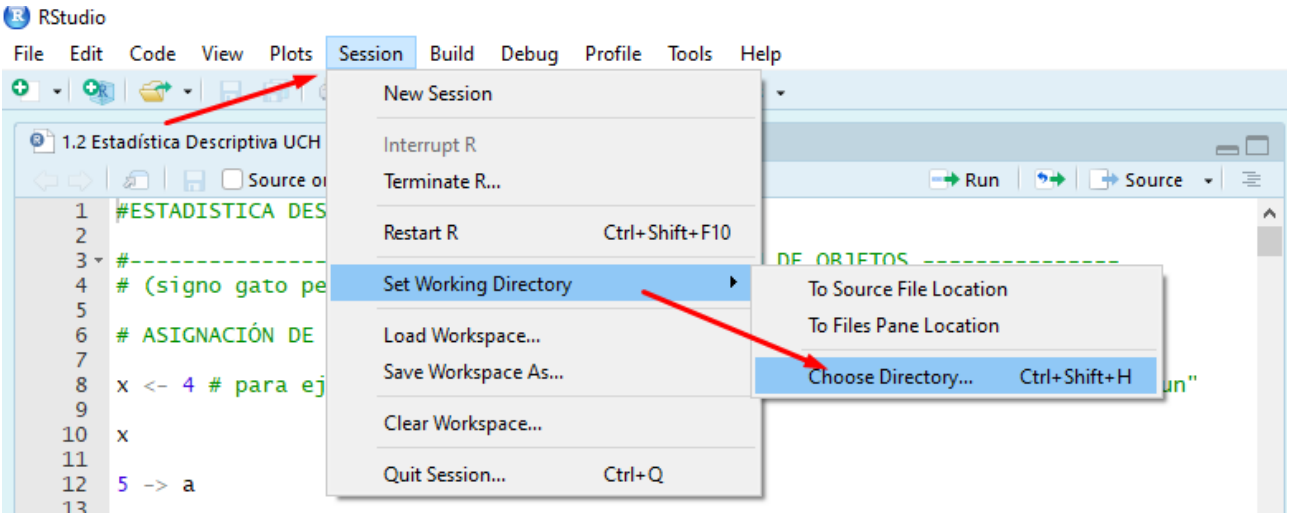


Se pueden ver 4 ventanas, además de la barra de opciones en la parte superior.

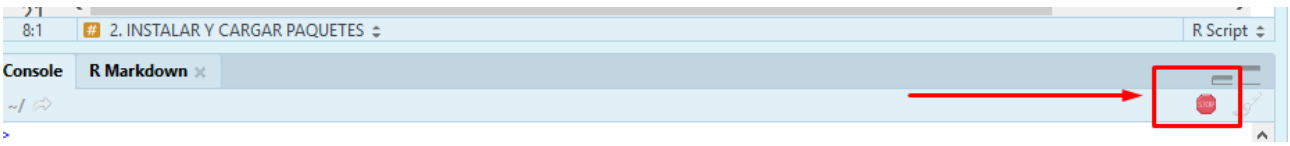
1. *Ventana (1)*: es el editor de sintaxis: se trata del lugar donde editamos la sintaxis para posteriormente ejecutarla. Al escribir allí no sucederá nada, a no ser que se apriete algún botón para ejecutar los comandos o la tecla ctrl+enter.
2. *Ventana (2)*: es el "entorno de trabajo" del programa: en este lugar se muestra el conjunto de datos y los "objetos" (resultados, variables, gráficos, etc.) que se almacenan al ejecutar diferentes análisis.
3. *Ventana (3)* tiene varias sub pestañas: (i) la pestaña **files** permite ver el historial de archivos trabajados con el programa; (ii) la pestaña **plots** permite visualizar los gráficos que se generen; (iii) la pestaña **packages** permite ver los paquetes descargados y guardados en el disco duro así como gestionar su instalación o actualización; (iv) la ventana **help** permite acceder al CRAN - Comprehensive R Archive Network (siempre que se cuente con conexión a Internet), página oficial del software que ofrece diferentes recursos para el programa: manuales para el usuario, cursos on line, información general, descarga de paquetes, información de los paquetes instalados, etc. Esta última pestaña es bastante útil: empleando el motor de búsqueda se accede de manera rápida a manuales de uso de los diferentes paquetes (y sus funciones) instalados en el computador (esto no requiere conexión a Internet).[7]; (v) la ventana **viewer** muestra los resultados al construir reportes mediante funcionalidades tipo *rmarkdown*.
4. *Ventana (4)*: es la consola. Corresponde a lo que sería el software R en su versión básica. Allí el software ejecuta las operaciones realizadas desde el editor de sintaxis.

Carpeta de trabajo y memoria temporal del programa.

El software R funciona como un **entorno temporal de trabajo**, esto quiere decir que el usuario va agregando datos y objetos (conjuntos de datos con diferentes atributos) a una “hoja en blanco”. Hay que tener en cuenta que R trabaja con la memoria activa (RAM) del computador, por lo tanto, cualquier análisis sólo mostrará la información resultante pero no permanecerá como archivo posible de utilizar de modo posterior. Es decir, si los análisis no son guardados como objetos (vectores, matrices, listas u otros tipos de objetos) se deberán repetir las instrucciones para obtener otra vez el resultado.
Dado que R opera como un espacio temporal y autónomo de trabajo, también es preciso indicarle en qué parte del disco duro del computador están los archivos a utilizar Para evitar el engorroso procedimiento de indicar todo el tiempo las rutas de acceso a los archivos (elementos del tipo: C:\escritorio\Curso R\base_datos.xlsx) es posible establecer una *carpeta de trabajo*: esto es, una carpeta predeterminada donde el programa buscará los archivos a ejecutar y guardará los archivos a conservar con cambios.
Existen dos alternativas para definir la carpeta de trabajo. La primera es emplear el siguiente comando:
setwd ("ruta de acceso a la carpeta especificada")
Otra forma de hacerlo es mediante la botonera superior; presionando el botón **Session**, luego **Set Working Directory**, **Choose Directory** y en la pestaña **examinar** se selecciona la carpeta a utilizar.

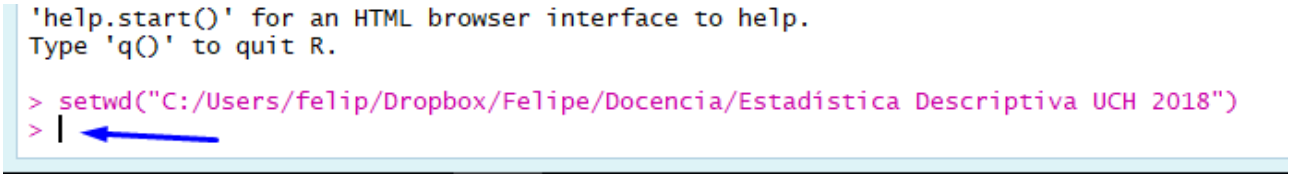


Todas las operaciones de R - sean indicadas vía sintaxis o botones - son ejecutadas según comando computacional que es visualizado en la consola. La ejecución de comandos entrega diferentes señales respecto a su funcionamiento. Por ejemplo, mientras se está ejecutando un comando, ¡el programa muestra un signo “Stop!” en la esquina superior derecha de la consola (como se ve en la imagen). Eso indica que el programa está ocupado ejecutando una acción. Si se presiona tal símbolo, se cancelará la operación en curso.



Una vez ejecutado el comando, debiera observarse el siguiente resultado en la consola.

Cuando esté todo en azul (o con otro color diferente al básico de la sintaxis dependiendo de la configuración de colores de RStudio) indica que el comando fue correctamente ejecutado. Asimismo, el signo >| con la barra parpadeando, indica que R está listo para procesar nuevas instrucciones. En el siguiente ejemplo (imagen 4.4) se muestra el establecimiento de una carpeta de trabajo cuyo desarrollo ha sido exitosamente ejecutado.



Elementos fundamentales del uso de sintaxis en RStudio

R es un lenguaje de programación En el caso de la versión básica del software, así como de la interfaz RStudio, el usuario interactuará con el programa mediante códigos. La *sintaxis* es un conjunto de códigos. Su uso en R es bastante intuitivo y sigue un patrón lógico. De modo general el lenguaje de programación de R (o sintaxis) sigue la siguiente estructura básica:

Ejercicio 4.1
comando (datos a utilizar)

El ejemplo indicado es una operación simple. Al aumentar la complejidad de los análisis la especificación de los comandos irá requiriendo una mayor cantidad de información. Por ejemplo, para construir una tabla de frecuencias de una variable ubicada en una base de datos se utiliza el siguiente comando, que le indica al software el conjunto de datos y variable específica a analizar (ejercicio 4.2):

Ejercicio 4.2
table (base de datos, variable)

En un sentido global, como se observa en la imagen 4.5, la estructura general de una sintaxis puede resumirse como sigue: a un objeto dado se le asigna el resultado de una función, que a su vez se ejecuta sobre un conjunto de datos especificado, con una serie de configuraciones particulares. En este caso, la sintaxis de la imagen es ficticia, pero nos servirá con un propósito pedagógico. Si se lee de izquierda a derecha, la línea de comando puede explicarse como sigue:

- Primero se indica un objeto a crear, con un nombre arbitrario, definido por nosotros.
- Luego se indica el asignador, que expresa que todo lo que esté a la derecha de la flecha se guardará en el objeto creado.
- Luego viene la función que en este caso permite leer archivos tipo Microsoft Excel. Luego de la función se abre un paréntesis que contiene los argumentos, es decir, instrucciones que especifican ciertos detalles de lo que queramos la función realice.
    1. El primer argumento por lo general indica la información a leer, en este caso indica un archivo de tipo Excel (extensión *xlsx*).
    2. El segundo argumento indica la hoja del archivo a leer.
    3. El tercer argumento indica qué columnas se leerán de forma específica (en este caso, las primeras diez)

4.

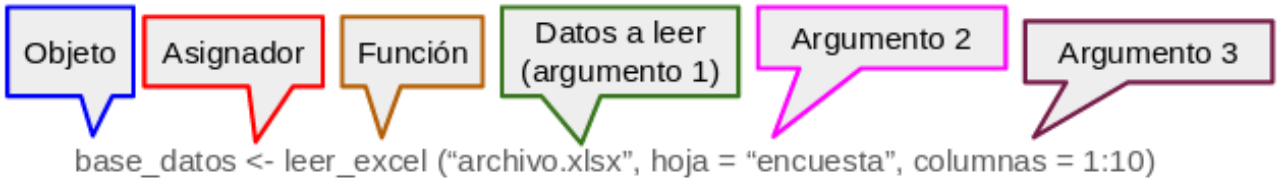
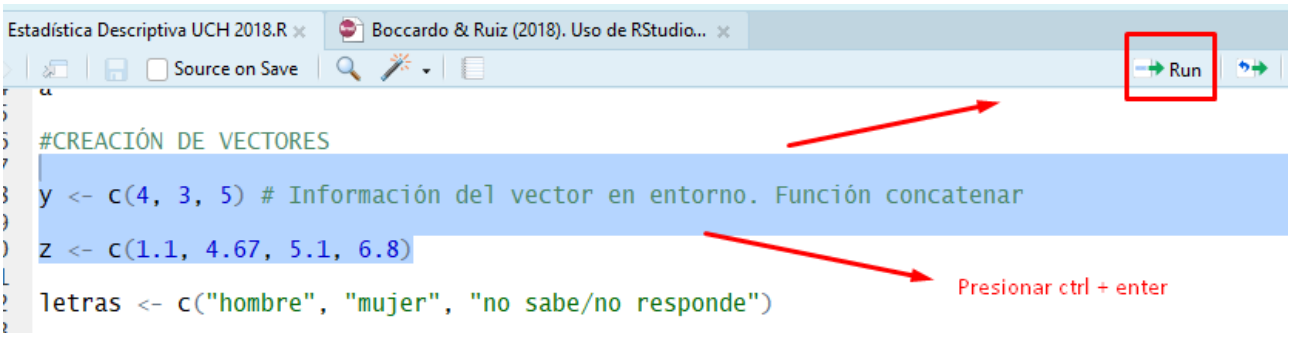


Imagen 4.5: Estructura básica de una sintaxis de R

Ahora bien, escribir tales instrucciones en el editor de sintaxis permite contar con una sintaxis editable que no se ejecutará de manera automática como operaciones computacionales (cuestión que sí sucede al escribir sintaxis directamente sobre la consola). Para ejecutar una sintaxis se debe seleccionar con el cursor - como cuando se busca *copiar* un texto - el trozo de código (*code chunk* en inglés) que interesa utilizar para luego:

1. Usando la botonera superior, apretar con el cursor del mouse el botón “Run”.
2. Apretar la combinación de teclas *ctrl + enter*.



Otra opción es ejecutar una sola línea de código. Para esto se posiciona el cursor *sobre* la línea que interesa ejecutar (en cualquier parte de ella) y se aplica algunas de las dos operaciones ya indicadas.

Manejo básico de la sintaxis de R: creación de objetos, inclusión de anotaciones y definición de secciones

Dado que el programa funciona en la memoria temporal del computador - memoria RAM - una vez que se ejecuta un proceso y se cierra el programa la operación y su resultado desaparecerán si no han sido almacenadas de alguna forma en el disco duro.
Para entender esto de manera práctica se pondrá un ejemplo con el comando más básico que se puede utilizar. Este comando consiste en crear un objeto y asignarle un dato numérico. El objetivo del ejercicio 4.1 es darle el valor 10 a X, luego pedirle a R que entregue el valor de X, para después cambiar el valor de X a 5. Para cualquiera de tales operaciones, se escribe una letra (en este caso es una X, pero podría ser cualquier otra letra) y con el asignador <- se le asigna el valor numérico deseado.

Ejercicio 4.3

```
#Asignación del valor "10" al objeto "X"
X <- 10
X
## [1] 10
#Cambiar el valor de X: cambiar el "10" por un "5"
```

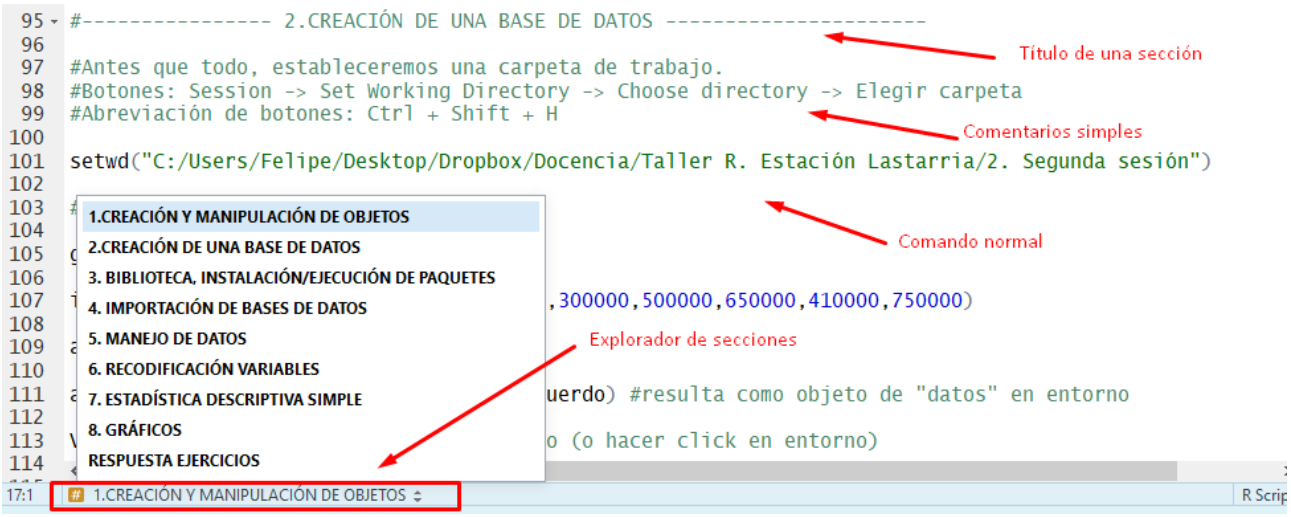
```
X <- 5
X
## [1] 5
```

Como se observa en el ejemplo, si luego de efectuar una asignación se ejecuta el objeto creado (“X”), se obtendrá como resultado el último valor almacenado. De tal modo, se entiende que un objeto es un elemento computacional que puede almacenarse en el programa para ser usado en análisis posteriores. Además, debe destacarse que al ejecutar una operación esta no es permanente: puede ser transformada si se sobreescribe la información.9 La estructura de la sintaxis computacional es bastante flexible. Sólo a modo de ejemplo, puede ejecutarse - ejercicio 4.4 - una asignación de un valor a un objeto alterando el orden de los elementos. Vale la pena destacar que el orden de los elementos no altera el resultado, mientras el asignador apunte desde los valores que nos interesa guardar hacia el nombre de un objeto a crear (o previamente existente). No obstante, ello, por una cuestión de orden sugerimos la estructura de sintaxis objeto <- valores, con el asignador apuntando hacia la izquierda de la pantalla.

Ejercicio 4.4

```
#Asignación del valor "5" al objeto "Y"
5 -> Y
Y
## [1] 5
```

Una cuestión importante es que el editor de sintaxis acepta la inclusión de comentarios en el cuerpo de los comandos. Como se ve en la imagen, al anteponer un signo gato (#) antes de un comando, este se pone de un color diferente a los códigos que representan instrucciones válidas (en la visualización por defecto de RStudio, el color será automáticamente *verde*).
¿Para qué sirve esto? Toda línea que esté con un signo gato antepuesto el programa la ignorará como comando de R. Esto permite incluir notas que indiquen lo que se está haciendo, mensajes o ayudas de memoria. Esto es relevante sobre todo en el contexto de trabajos que toman varios días - en los que será difícil recordar todo lo efectuado o probado - y en contextos de investigaciones colectivas donde se intercambian propuestas de análisis con otros investigadores.
Así, un aspecto bastante útil de la *funcionalidad de comentarios* es que permite separar mediante encabezados cualquier sintaxis. Como se ve en la siguiente imagen, si se indica el signo # y el título se encierra entre *al menos cuatro guiones* a cada lado, el programa considera tal formato como si se tratara de un título; esto permite que usando el explorador de la sintaxis quien investiga pueda **navegar** de manera más rápida por las diferentes secciones de una sintaxis.



Tipos de objetos en R (vectores)

Si los objetos creados contienen un *conjunto de datos del mismo tipo*, para el lenguaje del software (lenguaje de la informática) esto es un vector.[10] Aplicado al ámbito de las ciencias sociales se trata de una variable. Así, en R las variables (o vectores) pueden ser de 4 tipos; el tipo de variable depende del tipo de valores que se le asigna a cada objeto:

1. *numeric*: valores numéricos, incluye decimales.
2. *integer*: números enteros, no incluye decimales.
3. *character* valores alfanuméricos, es decir, letras, números y signos mezclados.
4. *logical*: valores lógicos, TRUE o FALSE.

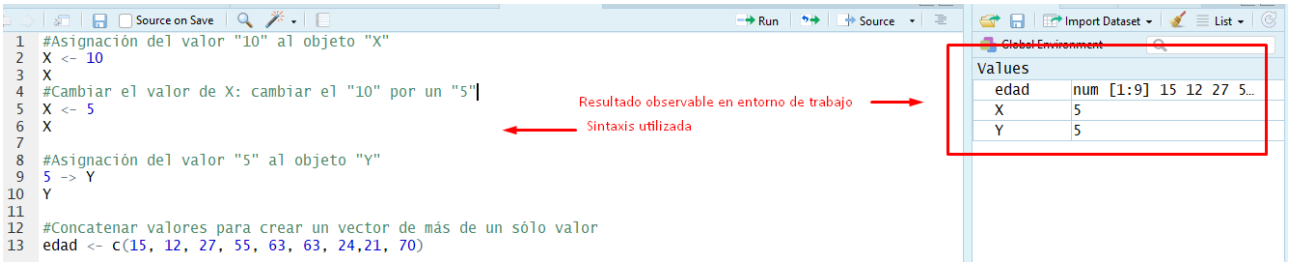
De esta forma si una variable es *numérica*, tendrá asignado números; en caso de ser una variable de tipo *cadena* (**character**) los valores se deben ingresar entre comillas ("ejemplo"), y en caso de ser *lógica* sus valores serán alguna de las opciones TRUE o FALSE.

Hasta ahora sólo se ha explicado cómo asignar un valor singular a un objeto de R. Para ingresar más de un valor en un vector se deben indicar los elementos a almacenar entre paréntesis y separados por comas, antecedidos de la función *concatenar*, que se ejecuta anteponiendo una c al conjunto de objetos a agrupar. Utilizando la función concatenar c() se puede crear un objeto que agrupe un conjunto de datos (un vector). En este caso se construirá una variable Edad con el siguiente conjunto de datos (deben separarse por comas): 15, 12, 27, 55, 63, 63, 24,21, 70. En este caso será una variable numérica:

Ejercicio 4.5

```
#Concatenar valores para crear un vector de más de un sólo valor
edad <- c(15, 12, 27, 55, 63, 63, 24,21, 70)
```

Al ejecutar el comando se observa automáticamente el nuevo objeto en el entorno de R (junto con los objetos x e y ya creados).



Como se observa, el programa indica que el objeto creado edad es un variable o vector de tipo numérico (numeric) de una sola dimensión y que tiene nueve casos ([1:9]).12

En el ejercicio 4.3 se construyó un objeto que consideraremos como una variable resultante de, por ejemplo, haber aplicado una breve encuesta a los estudiantes de una clase. Ahora bien, usando el comando table (edad) - ejercicio 4.5 (continuación) podemos construir una tabla de frecuencias de los valores; el número debajo del caso indica la frecuencia absoluta de ocurrencia de cada valor.

Ejercicio 4.5 (continuación)

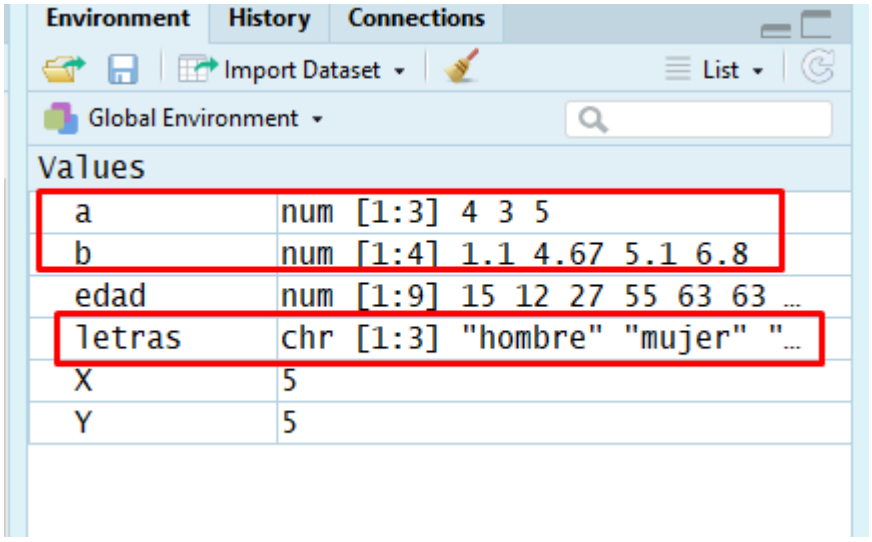
```
table(edad)
## edad
## 12 15 21 24 27 55 63 70
##  1  1  1  1  1  1  2  1
```

En el lenguaje computacional de R las variables se denominan “vectores”. A continuación se observa la creación de tres vectores, uno con números enteros, otro con números reales (incluye decimales y negativos) y un tercero con caracteres alfabéticos.

Ejercicio 4.6

```
#Vector numérico (función concatenar)
a <- c(4, 3, 5)

#Vector numérico (función concatenar)
b <- c(1.1, 4.67, 5.1, 6.8)

#Vector alfanumérico (función concatenar)
letras <- c("hombre", "mujer", "no sabe/no responde")
```

En el entorno de trabajo se observa la presencia de tres nuevas variables, con su respectiva cantidad de casos y el tipo de vector asignado por el programa.



Construcción de una base de datos

A continuación, se construirá la primera base de datos a partir de tres variables. Para esto, como se observa en los siguientes comandos, se parte por la construcción de tres variables de 9 casos cada una:

1. **Género**. Variable nominal con valores 1 y 2, que representan las categorías de respuesta “hombre” y “mujer”.
2. **Ingreso**. Variable de razón con valores arbitrarios de ingreso monetario.
3. **Acuerdo en torno al aborto libre**. Variable ordinal tipo *escala Likert* con valores 1, 2, 3, 4 o 5, que representan las categorías “nada de acuerdo”, “un poco de acuerdo”, “ni de acuerdo ni en desacuerdo”, “bastante de acuerdo”, “muy de acuerdo”.

Ejercicio 4.7

```
#Creación de las variables: todas tienen la misma cantidad de casos

genero <- c(1,1,2,1,2,2,2,1,2)

ingreso <- c(100000,300000,500000,340000,300000,500000,650000,410000,750000)

acuerdo <- c(1,1,3,2,4,1,5,3,2)
```

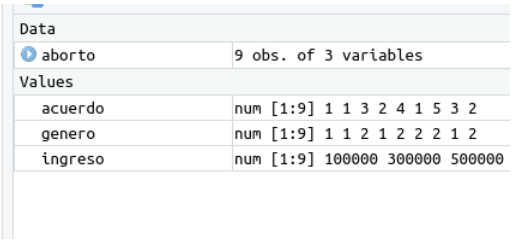
A partir de las variables ya creadas se puede construir una base de datos. Para esto se utiliza el comando data.frame asignando su resultado al objeto aborto que contendrá la base de datos construida.

Si la ejecución del comando es exitosa se verá un nuevo objeto de tipo data en el entorno de trabajo, donde además se indicará la dimensión de la base de datos (cantidad de casos y variables).

Ejercicio 4.7 (continuación)

```
#Creación de base de datos a partir de variables previamente creadas
#Se asigna el resultado al objeto "aborto"
aborto <- data.frame(genero, ingreso, acuerdo)
```

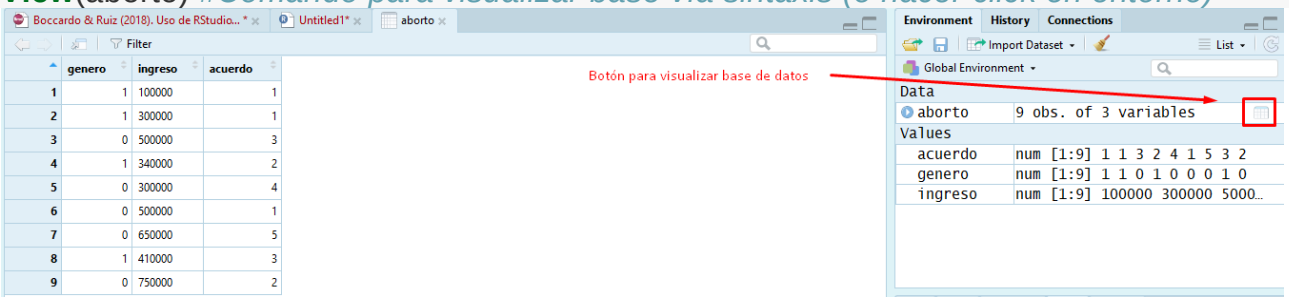
```r
#Creación de las variables: todas tienen la misma cantidad de casos

genero <- c(1,1,2,1,2,2,2,1,2)

ingreso <- c(100000,300000,500000,340000,300000,500000,650000,410000,750000)

acuerdo <- c(1,1,3,2,4,1,5,3,2)

#Creación de base de datos a partir de variables previamente creadas
#Se asigna el resultado al objeto "aborto"
aborto <- data.frame(genero, ingreso, acuerdo)
```



Posteriormente se puede visualizar la base datos de dos maneras: utilizando el comando View o presionando un botón que se encuentra al lado del objeto base de datos en el entorno de trabajo del software. En el siguiente código se explicita la primera forma y en la imagen posterior la segunda.

Ejercicio 4.8

```r
View(aborto) #Comando para visualizar base vía sintaxis (o hacer click en entorno)
```



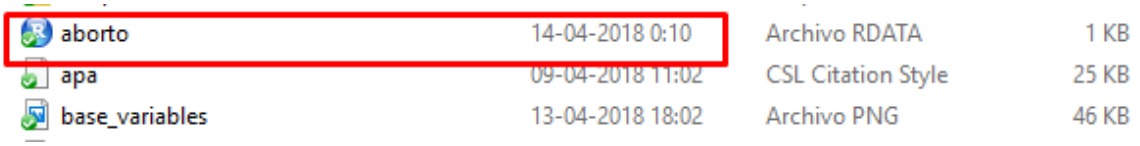
	genero	ingreso	acuerdo
1	1	100000	1
2	1	300000	1
3	0	500000	3
4	1	340000	2
5	0	300000	4
6	0	500000	1
7	0	650000	5
8	1	410000	3
9	0	750000	2

Luego de construida esta base de datos interesa guardarla como un archivo reutilizable para posteriores análisis. Para ello es útil el comando save: indicando el nombre del objeto a guardar como archivo y definiendo también el nombre del archivo. Como se ve en la siguiente línea de comando, el nombre del archivo resultante se indica con el argumento file = "nombre_archivo.extensión" (el nombre del archivo va entre comillas).

Ejercicio 4.9

```r
save(aborto, file = "aborto.RData") #Se indica primero el objeto a guardar
                                    #y luego el nombre del archivo, entre comillas.
```

Ejecutando tal comando, se creará un nuevo archivo en la carpeta que hayamos indicado como carpeta de trabajo. Como se ve en la imagen a continuación, este archivo debería ser reconocido como un archivo de formato RStudio; se observa que es un archivo recién creado gracias a la información proporcionada por la columna Fecha de modificación.



Este archivo es la base de datos recién construida pero almacenada como archivo de formato R en el disco duro, específicamente en la carpeta de trabajo fijada en la sesión de R. Este archivo puede usarse para posteriores análisis, puede ser enviado a otras personas, etc.

Finalmente, se muestran dos maneras para **“limpiar”** el entorno de trabajo. Esto resulta útil pues luego de hacer múltiples cálculos exploratorios, mientras se depura un esquema de análisis, el entorno de trabajo se irá llenando paulatinamente con objetos que no sirven para continuar trabajando y sólo pueden confundir en los pasos posteriores del análisis. Nuevamente, para limpiar el entorno de trabajo existen dos maneras:

1. La primera es vía sintaxis, con dos comandos específicos que se detallan a

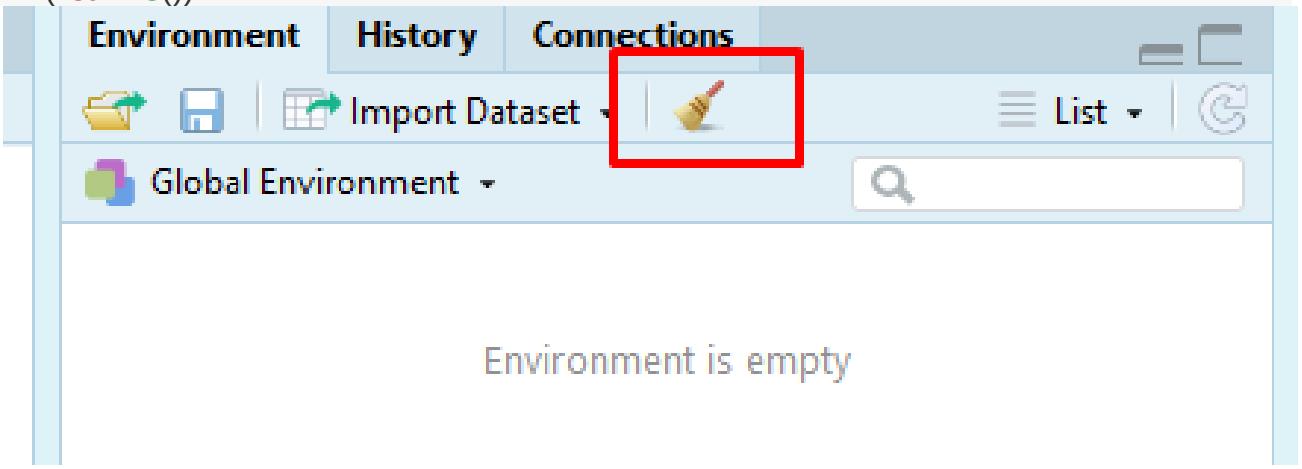
continuación:

a. El primero permite eliminar elementos específicos y,
b. El segundo permite vaciar totalmente el entorno de trabajo.

2. La segunda forma es utilizando un botón existente en la botonera superior del entorno de trabajo (con forma de escoba) que permite borrar todos los elementos del entorno de trabajo, que se detalla en la siguiente imagen.

Ejercicio 4.10

```
#Comando para eliminar elementos especificos, aquí se elimina
#la base creada.
remove(aborto)

#Comando para limpiar todos los objetos del entorno de trabajo.
rm(list = ls())
```



Como se ve en la última imagen, ya sea ejecutando el comando rm(list = ls()) o apretando el botón indicado, se puede limpiar completamente el entorno de trabajo.

Manejo de la biblioteca y gestión de paquetes

La versión básica del software R trae una cantidad limitada de herramientas para análisis estadístico. Como generalmente buscaremos usar otras funcionalidades, se vuelve necesario descargar nuevos paquetes y saber cómo cargarlos en la sesión de trabajo.

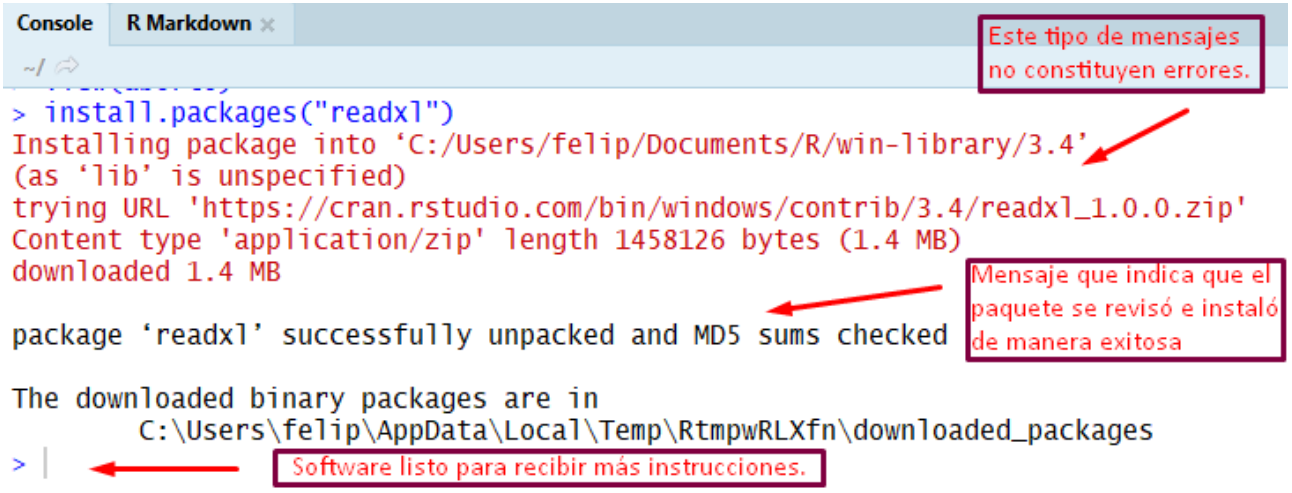
Descargar paquetes

La descarga e instalación de paquetes adicionales a la versión básica de R se realiza mediante el comando install.packages() indicando entre comillas el nombre del paquete a descargar. Este comando conecta la sesión de R directamente con el CRAN y descarga al computador - en la carpeta de instalación de R - los paquetes requeridos. Por ello, se requiere tener conexión a Internet para efectuar tal operación. En el siguiente ejemplo se descarga el paquete readxl que permite abrir bases de datos desde formato Excel en la sesión de R.

Ejercicio 5.1

```
install.packages("readxl") #Se descarga e instala el paquete readxl.
```

Una vez ejecutado tal comando saldrán distintos mensajes en la consola de R, generalmente con una apariencia como la que se muestra en la imagen a continuación. Vale destacar que muchas veces salen mensajes en rojo e incluso mensajes de alerta (warnings); por lo general se trata de mensajes que el programa muestra al usuario, pero que no implican que algo haya salido mal en la ejecución del comando. Como se aprecia en la siguiente imagen, el software indica cuándo el paquete se descargó e instaló de manera adecuada en el computador, a la vez que la consola queda lista para seguir ejecutando análisis.



Cargar paquetes

Ya se ha indicado que R funciona en la memoria temporal del programa (memoria RAM). Esto hace que cada vez que se abre el programa (vía RStudio) éste se despliega en su versión básica. Es por ello que no basta con descargar al disco duro los paquetes para poder utilizarlos. Para usar una función correspondiente a un paquete adicional a la versión básica de R tal paquete se debe cargar en la sesión de R en que se está trabajando. Esto se efectúa con el comando library() indicando entre paréntesis el nombre del paquete a cargar. A diferencia de la función install.packages acá el nombre del paquete no se indica entre comillas.

Ejercicio 5.2

```
library(readxl) #Carga el paquete descargado a la sesión de trabajo de R.
```

Como se observa en la imagen a continuación, en esta operación el software también puede arrojar mensajes en rojo que no significan que haya ocurrido un error. En este caso es un mensaje de alerta para el usuario (Warning message) que informa que el paquete "fue construido para una versión del software menor a la 3.4.4".

Actualizar la versión básica de R y los paquetes instalados

R está en permanente actualización por lo que luego de algunos meses la versión que haya sido instalada quedará desactualizada. Cada versión busca introducir mejoras, robustecer funciones ya existentes, etc.

Si se precisa actualizar la versión instalada del software una primera opción es repetir las operaciones indicadas en el capítulo 3. No obstante, es posible efectuar una actualización del software mediante la función updateR(). Esta función es parte del paquete installr y funciona solamente para el sistema operativo Windows. Como se observa en las siguientes líneas de comando, se trata de un procedimiento simple.

Ejercicio 5.3

```
install.packages("installr")

library(installr)

updateR()
```

Los diversos paquetes existentes para R son desarrollados a lo largo del mundo por una extensa red de colaboradores. Cuando estos paquetes superan su período de prueba son enviados al R Core Team (equipo a cargo de mantener y mejorar el software en sus diferentes versiones) donde son testeados y luego subidos de manera oficial al CRAN. Así como el software R, estos paquetes están sujetos a permanentes actualizaciones y mejoras. Por ello, también es preciso conocer alguna forma de actualizar de manera rápida y simultánea todos los paquetes que estén instalados en el disco duro. Para ello, se sugiere utilizar el siguiente comando.

Ejercicio 5.4

```
update.packages()
```

Gestión de bases de datos

Un aspecto importante en el uso de RStudio enfocado en el análisis de datos sociales es el manejo de base de datos. Esto puede referir tanto a bases que quien efectúa el análisis haya construido como a bases de datos de estudio sociales realizados por otros.

En este manual de apoyo docente se utilizará una base de datos secundaria. Tanto para enseñar los procedimientos de importación, validación y modificación de datos, como para los posteriores aspectos de análisis estadístico descriptivo.

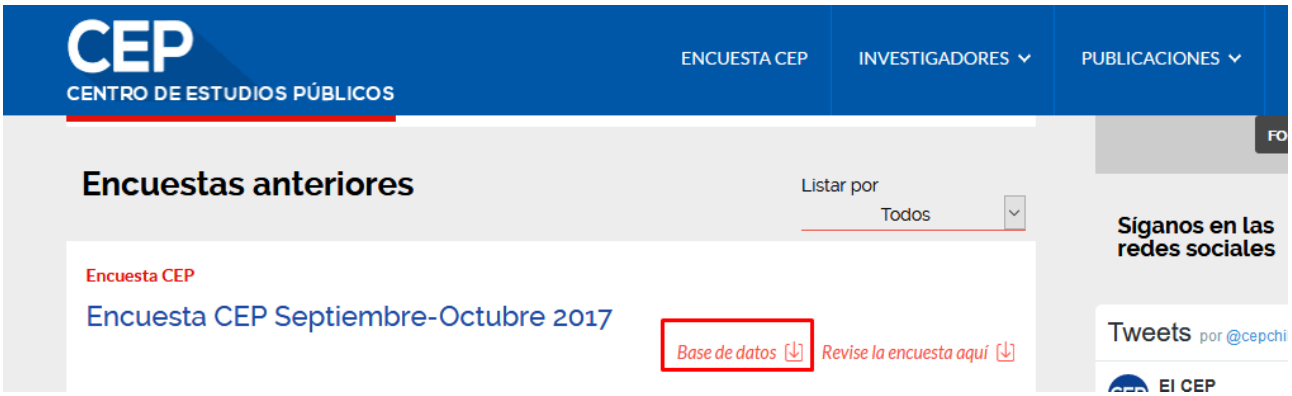
En los siguientes ejemplos se trabajará con la base de datos de la Encuesta Nacional de Opinión Pública del Centro de Estudios Públicos (Encuesta CEP, de aquí en más) correspondiente a su versión 81 cuyo trabajo en terreno se efectuó entre septiembre y octubre de 2017. Esta encuesta busca caracterizar las actitudes y opiniones, políticas, sociales y económicas de la población chilena, destacando las necesidades, principales preocupaciones y preferencias de todos los habitantes del territorio nacional. Es una de las fuentes de información más importantes para estudiar la opinión pública en Chile, respecto a temas coyunturales (CEP 2017).

Toda la información de esta encuesta se puede encontrar en el apartado Encuesta CEP de la página institucional del centro de estudios mencionado.14 Desde este sitio en línea el usuario podrá descargar las bases de datos históricas de esta encuesta, junto con sus manuales de uso metodológico.

Descarga de una base de datos de interés

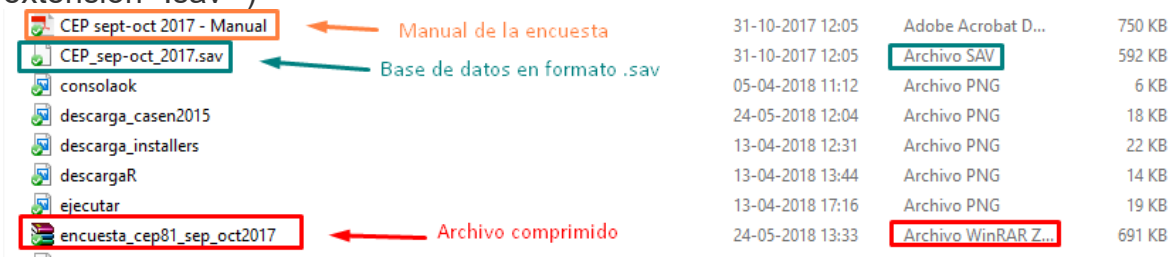
Una primera acción a realizar es la descarga de la base de datos indicada al computador. Esto puede realizarse accediendo a la página web indicada del CEP. En la botonera superior de esta página se encuentra un botón de acceso a la *Encuesta CEP* (fecha de consulta: 10 de abril 2019).

Dentro de esa página se debe buscar el apartado *Encuestas anteriores* para encontrar en el repositorio la *Encuesta CEP Septiembre-Octubre 2017*.



Allí, oprimiendo el botón Base de datos se podrán descargar los archivos de interés.15

Luego de clickear el enlace de descarga se debe guardar descargar archivo comprimido como el que se ve en la siguiente imagen (con la extensión ".rar"). Para poder descomprimir el archivo original basta con tener instalado el programa WinRar (puede descargarse desde esta página). Haciendo clic derecho sobre el archivo descargado, y seleccionando la opción Extraer aquí se obtendrá el archivo original (de extensión ".sav"")



Como se ve en la imagen 6.3 el archivo resultante es un archivo de extensión .sav, lo que indica que se trata de un archivo para abrir y trabajar en SPSS. De aquí en más se usará este archivo para desarrollar ejemplos de análisis estadístico. Además de la base de datos, se descarga el manual de uso de la encuesta en formato PDF.

¿Cómo abrir bases de datos desde formato SPSS?

¿Por qué puede ser importante manejar herramientas que permitan que desde RStudio interactuemos con diferentes formatos de bases de datos? Imagine lo siguiente: durante la formación universitaria usted ha aprendido a usar RStudio como herramienta de análisis estadístico. Pero sucede que al incorporarse a un centro de estudios sociales y observa que el resto de los analistas trabaja fundamentalmente con otro software de análisis estadístico: SPSS. Si usted quiere lograr dialogar con tales especialistas - que muy probablemente no estarán dispuestos a aprender a usar RStudio - deberá lograr al menos abrir bases de datos en formato SPSS, para así poder hacer los análisis que se le encomiende.

Para ello se utiliza un paquete específico llamado haven que cuenta con funciones para abrir bases de datos desde diferentes formatos. Para eso primero se descarga e instala el paquete en el computador:

Ejercicio 6.1

```
install.packages("haven")
```

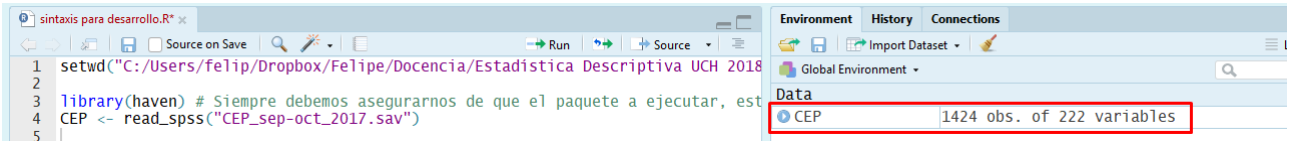
La función específica a utilizar es read_spss cuya ejecución es muy simple. Luego de la función se indica el nombre del archivo a leer entre paréntesis y entre comillas. Como se observa a continuación, esta función sirve para leer la base de datos de la Encuesta CEP descargada en el anterior apartado. Es importante recordar que para usar una fuente de datos en análisis futuros se debe guardar como un "objeto" en el entorno de trabajo. Para

eso, la ejecución de la función debe asignarse a un objeto nuevo o preexistente mediante la función de asignación. En este caso se asigna a un nuevo objeto llamado “CEP” (CEP <- lectura base de datos):

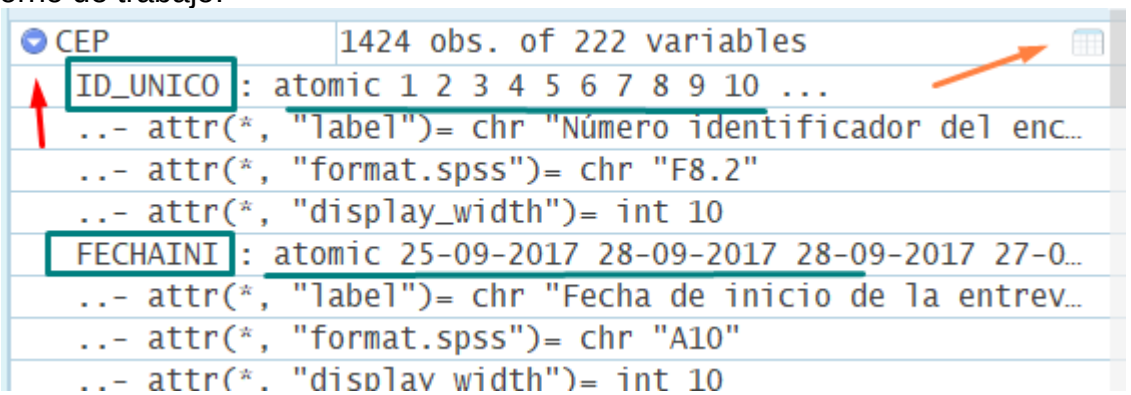
Ejercicio 6.2

```
library(haven) #Debemos asegurarnos de que el paquete a ejecutar,
#está cargado en la sesión de Rtudio.

CEP <- read_spss("CEP_sep-oct_2017.sav") # Nombre del archivo entre comillas.
```

Como se observa en la imagen a continuación, luego de tal operación ya se cuenta con un nuevo objeto (CEP) en el entorno de trabajo, que almacena la base de datos de la Encuesta CEP.



Con la base de datos cargada en el entorno de trabajo podemos explorar sus principales características. Como se observa en la imagen 6.5 existe un botón (flecha azul indicando hacia abajo) que permite desplegar la estructura interna de la base de datos existente en el entorno de trabajo.



Así se logra explorar rápidamente las principales características de la estructura interna de la base de datos cargada, es decir, los atributos de las diferentes variables en su interior. Entre estas características - a las que se puede acceder con el comando attributes - se encuentran las etiquetas, entre otros elementos de formato, que generalmente se observan en la vista de variables en SPSS. Si bien en R no se cuenta con una visualización directa de tales elementos, estos se almacenan como atributos en las variables importadas.

Como ya fue indicado en el apartado 4.5 Construcción de una base de datos aquí también es posible usar el botón con forma de planilla ubicado a la derecha del objeto en el entorno de trabajo para visualizar la base de datos como una planilla y así poder inspeccionarla visualmente. Esto permitirá explorar rápidamente su estructura y contenidos: observar los nombres de las variables, el tipo de valores que almacenan, etc.

¿Cómo abrir bases de datos desde diferentes formatos de Microsoft Excel?

Otro formato relevante para el trabajo con base de datos es el ecosistema integrado por los diferentes tipos de planillas de cálculo vinculadas al software Microsoft Excel. Muchas veces la digitación de encuestas se efectúa en softwares de estas características, por lo que un formato primario para almacenar bases de datos es, sin duda, las planillas de cálculo. Es más, muchas veces diversos fenómenos sociales son registrados por la actividad humana en este tipo de archivos.20

Si bien el formato planillo de cálculo generalmente se asocia a Microsoft Excel, debido a la masividad de sus productos, refiere a un formato más general. Hoy en día, también se asocia a aplicaciones en línea como Hojas de Cálculo de Google u otro software de funcionalidad de oficina como Calc en sus versiones de Libre Office y Open Office.

A continuación, se indican dos modalidades para importar bases de datos en formato de hoja de cálculo a R.

La primera opción es trabajar directamente con un archivo con formato para Microsoft Excel 2007 o superior. Se trata de archivos con una extensión .xlsx que tienen un formato de libro, es decir, pueden soportar en su interior a más de una hoja de trabajo. Para efectos caso de este manual se ha convertido la base de datos de la Encuesta CEP trabajada en el apartado anterior a tal formato. Este procedimiento es sencillo si se tiene instalado SPSS.21 Para efectos de este tutorial el archivo se puede descargar desde el siguiente enlace (indicado en la presentación de este documento), y que lleva por nombre CEP_sep-oct_2017.xlsx.

Al abrir el archivo desde el explorador de archivos se observa que tiene dos hojas. Una primera almacena el registro de las respuestas con un número de identificación por caso y la fecha de respuesta de la encuesta. La segunda hoja es la base de datos propiamente tal y es la que interesa cargar en el entorno de trabajo de R.

	A	B	C	D	E	F	G	H
1	ID_UNICO	FECHAINI						
2	1,00	25-09-2017						
3	2,00	28-09-2017						
4	3,00	28-09-2017						
5	4,00	27-09-2017						
6	5,00	02-10-2017						
7	6,00	29-09-2017						
8	7,00	01-10-2017						
9	8,00	01-10-2017						

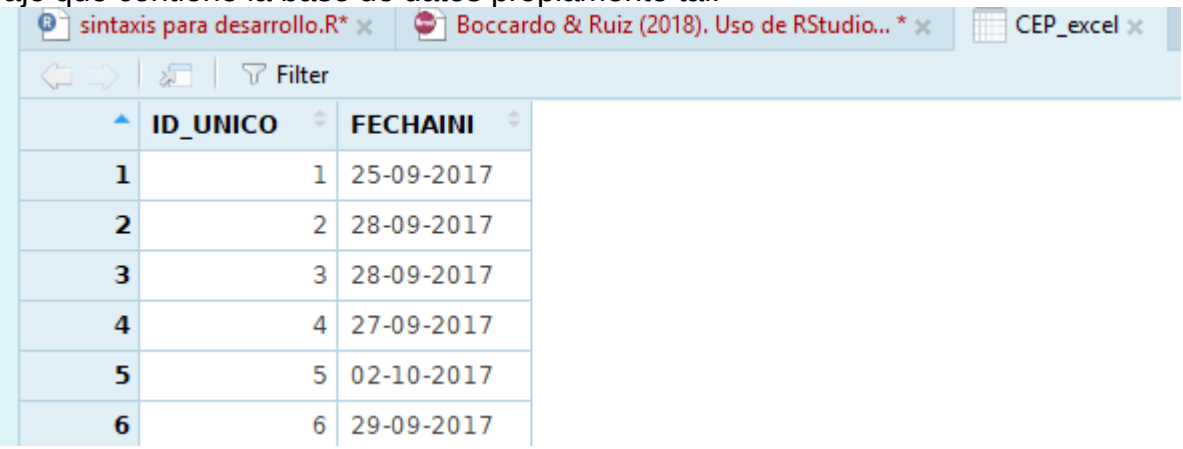
	A	B	C	D	E	F	G	H	I	J	K	L
1	VOTACION_1	VOTACION_2	VOTACION_3	VOTACION_4	SV_1	SV_2	MB_P1_1	MB_P1_2	MB_P1_3	MB_P2	MB_P3	MB_P4
2	10	4	4	4	8	3	3	5	7	1	3	4
3	2	3	3	3	10	5	10	1	2	2	2	4
4	8	5	1	5	10	10	11	10	7	4	1	4
5	7	2	2	2	8	5	7	12	6	3	1	3
6	7	2	2	2	5	5	13	1	11	2	1	2
7	7	2	2	2	9	5	11	1	5	2	1	3
8	4	1	2	2	9	5	16	3	1	4	2	3
9	8	1	1	1	6	8	2	8	1	2	2	2
10	4	1	1	1	6	7	10	1	7	3	1	3

Entonces, ¿cómo cargar una base de datos desde este formato al entorno de trabajo en R? En esta instancia se usará el paquete readxl que previamente se debe descargar e instalar en el computador. Específicamente se usará la función read_excel de este paquete en su versión más simple - sin argumentos adicionales - indicando solamente el nombre del archivo a leer. Su resultado se guardará en un nuevo objeto llamado CEP_excel.

Ejercicio 6.3

```
install.packages("readxl") #Descarga e instalación del paquete
library(readxl) #Cargar paquete en sesión de trabajo de R

CEP_excel  <- read_excel("CEP_sep-oct_2017.xlsx") #Leer libro excel
```

Al observar la planilla cargada en el entorno de trabajo se verá que no es la que interesa para desarrollar análisis estadísticos. Por defecto la función lee la primera hoja de del libro de trabajo Excel, por lo que en este caso cargó la planilla con los datos de identificación de cada respondiente, siendo que interesa la lectura de la segunda hoja del libro de trabajo que contiene la base de datos propiamente tal.



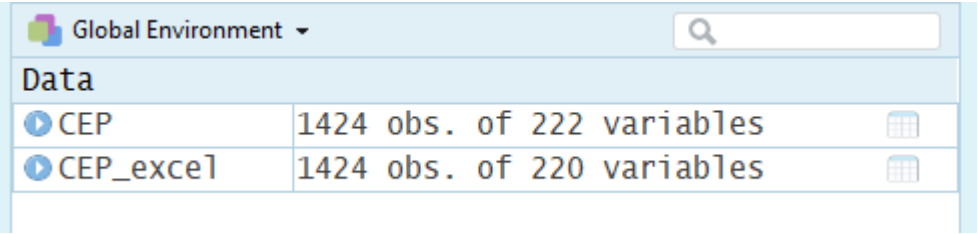
	ID_UNICO	FECHAINI
1	1	25-09-2017
2	2	28-09-2017
3	3	28-09-2017
4	4	27-09-2017
5	5	02-10-2017
6	6	29-09-2017

Para solucionar este problema se repetirá la operación pero utilizando un argumento extra en la función. Mediante el argumento sheet = se le indica al programa la posición o nombre de la hoja que interesa leer en el interior del libro de trabajo. En este caso se indicará que interesa leer la hoja ubicada la posición “2” del libro, o la hoja de nombre “DATOS” (que es lo mismo para efectos de este manual). A continuación se sobrescribe el objeto creado en el entorno de trabajo, actualizando la función con esta información.

Ejercicio 6.3 (continuación)

```
CEP_excel  <- read_excel("CEP_sep-oct_2017.xlsx", sheet = 2)
#indica posición de hoja en el libro de trabajo.

CEP_excel  <- read_excel("CEP_sep-oct_2017.xlsx", sheet = "DATOS")
#indica nombre de hoja en libro de trabajo.
```

Observando el entorno de trabajo se verá que el objeto CEP_excel se ha actualizado y ahora presenta 220 variables.



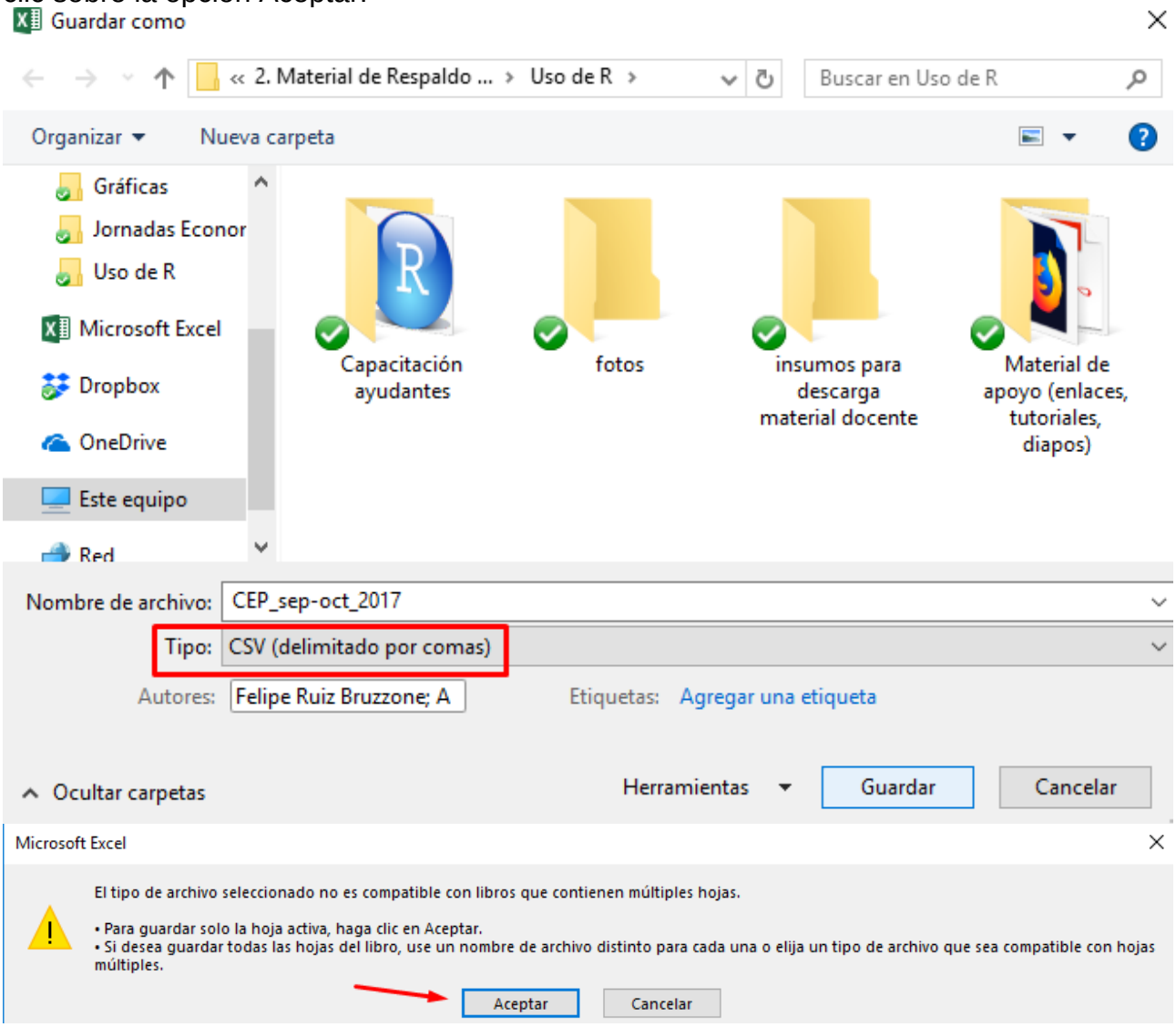
Ahora bien, la función read_excel considera a la primera línea de datos como los nombres de las variables de forma automática. Hay veces en que en la primera fila no se encuentra el nombre de las variables habiendo primero otro tipo de información. Por eso resulta importante indicarle a la función desde qué fila comenzar a leer los datos. En el caso de que la información relevante comience en la fila 2, siendo tal fila la que contiene el nombre de las variables se agrega a estos argumentos la opción skip = 1 para que el software salte u omita la primera fila y comience la lectura de los datos en la fila 2.

Ejercicio 6.3 (continuación)

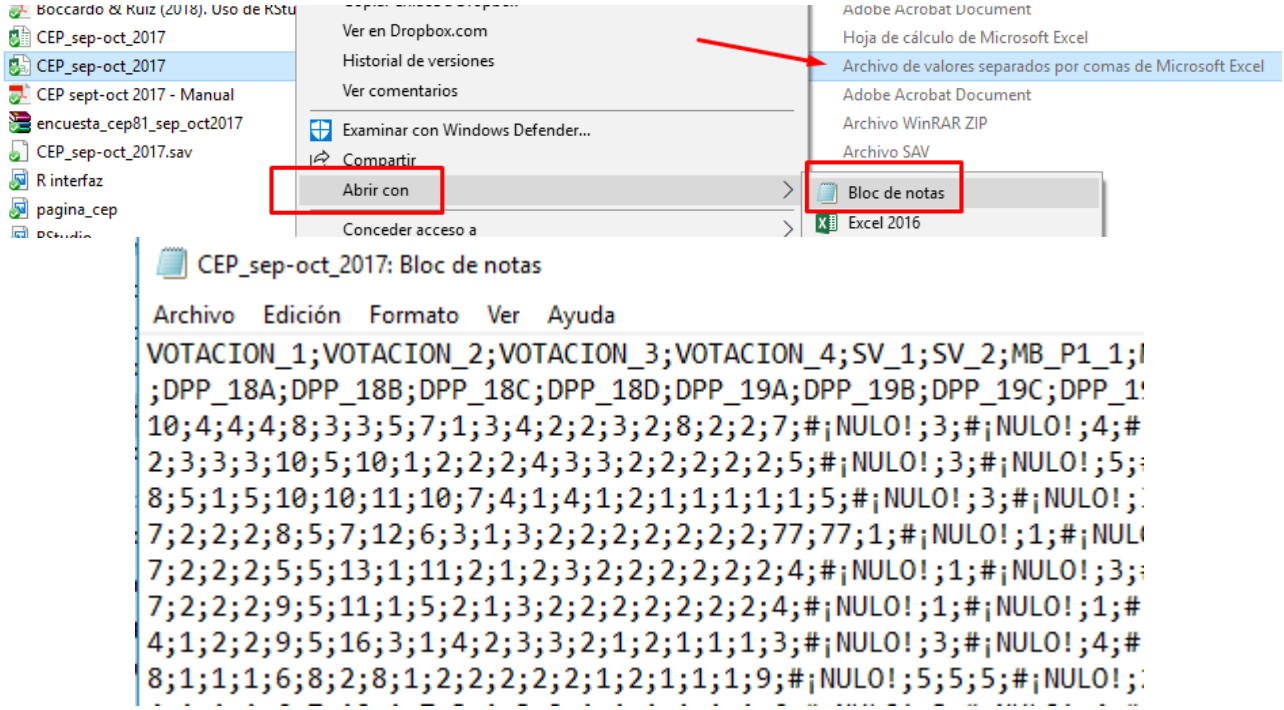
```
CEP_excel  <- read_excel("CEP_sep-oct_2017.xlsx", sheet = 2, skip = 1)
#indica posición de la hoja en el libro de trabajo.
```

Una segunda forma de importar a R archivos del tipo hoja de cálculo es trabajar con el formato CSV (comma separated values) o archivo de valores delimitados por comas. Se trata de un tipo de archivo más simple que un libro de Microsoft Excel, en la medida que puede contener sólo una - y no varias - hoja de trabajo. Si bien este tipo de archivos pueden encontrarse al descargar una base de datos secundaria, para efectos de este manual de apoyo docente se creará un archivo de este tipo a partir del archivo originalmente almacenado en formato Excel (.xlsx).

Para esto, como se observa en las imágenes 6.11 y 6.12, basta con que desde el archivo excel en cuestión, se utilice a la opción Guardar como y seleccionando el formato señalado. El programa debería advertir al usuario que el formato seleccionado no soporta múltiples hojas de trabajo (imagen 6.12), por lo que guardará sólo la hoja activa. Asegurando que la planilla que interesa guardar es la hoja que está seleccionada se hace clic sobre la opción Aceptar.



El archivo resultante puede ser leído como planilla de Microsoft Excel. Sin embargo, para entender su estructura interna primero se abrirá con la aplicación Bloc de Notas mediante la opción Abrir con.



Se observa en la imagen 6.14 que el archivo presenta una tabulación del tipo matriz de datos - en este caso, la base de datos que contiene las respuestas a una encuesta social - cuyos valores (cada variable) están separados por el signo punto y coma (;). Este signo delimita cada columna de casos.

Resulta importante considerar esta estructura pues en la notación anglosajona que subyace al lenguaje original de R (el inglés) se usa la coma para separar valores y el punto para denotar valores decimales. En el caso de la notación de habla hispana se emplea la coma para denotar decimales y el punto y coma para separar las observaciones.

Este “detalle” importa pues las funciones básicas para leer archivos CSV viene configuradas por defecto para entender a las comas como separador de los casos. Esto se muestra en el siguiente ejemplo.

Ejercicio 6.4

```
CEP_csv  <- read.csv("CEP_sep-oct_2017.csv")
```

En este caso, la ejecución del comando implica un error de ejecución.

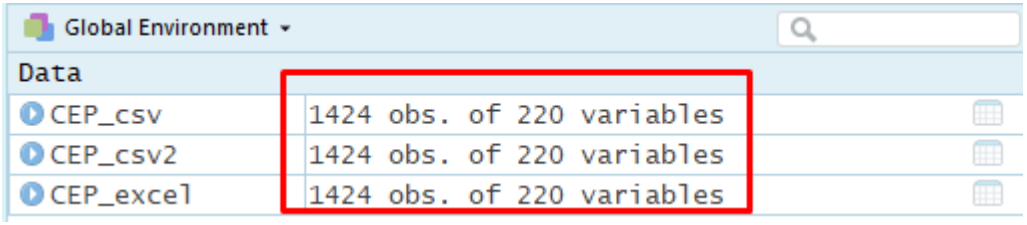
```
aldo Clases/Uso de R")
> CEP_csv  <- read.csv("CEP_sep-oct_2017.csv")
Error in read.table(file = file, header = header, sep = sep, quote = quote,  :
  more columns than column names
> CEP_csv  <- read.csv("CEP_sep-oct_2017.csv", sep = ";")
> View(CEP_csv)
```

El programa avisa que la cantidad de columnas que resultan de la lectura de la planilla es mayor a la cantidad de nombres de variables, por lo que no logra leer el archivo. Esto sucede debido a que las comas presentes en los casos, que denotan valores decimales, el software las ha entendido como separador de casos.
Para solucionar tal problema se indican dos opciones.

i. La primera es ocupar la misma función, pero agregando un argumento sep = mediante el cual se indica qué signo debe considerar para separar los valores.
ii. La segunda opción es usar una variación de la función original (read.csv2), configurada para que considere las comas como notación de decimales y el punto y coma como separador de valores.

Ejercicio 6.4 (continuación)

```
CEP_csv  <- read.csv("CEP_sep-oct_2017.csv", sep = ";")
CEP_csv2 <- read.csv2("CEP_sep-oct_2017.csv")
```



El resultado muestra que se han leído tres bases de datos coincidentes en términos de estructura, por lo que se ha logrado llegar al mismo resultado usando funciones diferentes.

Para el desarrollo de todos los ejemplos posteriores de este manual se considerará una de las bases de datos leídas. Para ello se "limpiará" por primera vez el entorno de trabajo dejando solamente la base de datos nombrada como CEP_csv.

Ejercicio 6.5

```
remove(CEP_csv2, CEP_excel)
```

Construir una base de datos sólo con variables de interés: exploración de bases de datos y recodificación de variables

Lograr cargar una base de datos a la sesión de R es un paso inicial que permite disponer de un conjunto de datos "en bruto" que, muy probablemente se deberán configurar para lograr efectuar los análisis de interés. Se sugiere trabajar solamente con aquellas variables a analizar y crear una nueva base de datos sólo con la información de interés, sin editar la fuente original de datos. Para los siguientes ejercicios se seleccionarán siete variables desde la base de datos de la Encuesta CEP ya mencionada. En la siguiente tabla se indica una descripción de la variable, su nombre en la base original y el nuevo nombre a asignar en la nueva base de datos.

Detalle de las variables específicas a considerar para los análisis

Detalle de las variables específicas a considerar para los análisis

Descripción de la variable	Nombre en base original	Nuevo nombre	Valores	Nivel de medición
Ponderador	**POND**	*pond*	Números simples para ponderación.	No aplica.
Género informado	**SEXO**	*sexo*	1 = hombre, 2 = mujer.	Nominal.
Región de	**REGION**	*region*	Número de región, del 1	Nominal.

Detalle de las variables específicas a considerar para los análisis

Descripción de la variable	Nombre en base original	Nuevo nombre	Valores	Nivel de medición
residencia			al 15.	
Fecha de nacimiento (edad)	**DS_P2_EXACTA**	*edad*	Edad cumplida en número entero.	Razón.
Satisfacción con la propia vida	**SV_1**	*satisfaccion*	Escala del 1 al 10.	Intervalar.
Percepción de satisfacción de chilenos con su vida	**SV_2**	*satisfaccion_chilenos*	Escala de 1 al 10	Intervalar.
Evaluación de la situación económica nacional	**MB_P2**	*eval_econ*	Escala del 1	

Para construir una nueva base de datos solo con las variables especificadas será preciso efectuar varias operaciones, que se detallan en este apartado. Para este tipo de manipulación de bases de datos y variables existen diferentes herramientas: algunas forman parte de las funcionalidades básicas de R, pero otras provienen del paquete dplyr. En la siguiente tabla se resumen las principales características de las funciones a utilizar.

Funciones de utilidad para la manipulación de bases de datos

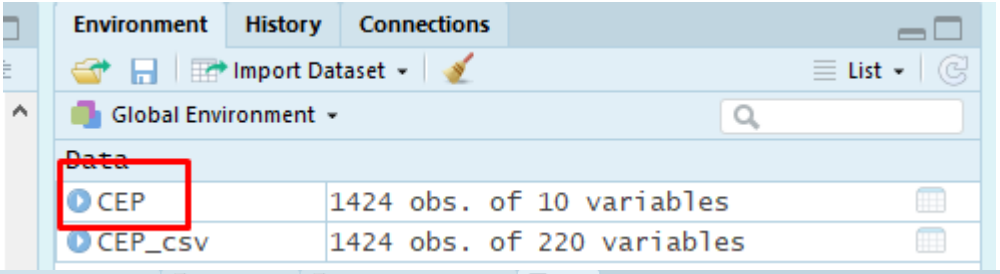
Función	Paquete	Utilidad
View	utils	Visualizar un objeto tipo matriz de datos en formato planilla.
names	base	Muestra los nombres de cada elemento incluido en un objeto determinado, por ejemplo, una base de datos.
dim	base	Entrega la dimensionalidad del objeto. En el caso de bases de datos (formato matriz) indica el número de filas (casos) y columnas (variables), en ese orden.
select	dplyr	Seleccionar variables (columnas) específicas de un objeto del tipo data.frame.
rename	dplyr	Renombrar variables dentro de una misma base de datos. Usada como la función select permite seleccionar y al mismo tiempo renombrar variables.
mutate	dplyr	Transformar variables en una nueva, sin alterar la original.
recode	dplyr	Recodificar categorías de una variable, estableciendo una a una las equivalencias entre las categorías originales y las categorías a crear.
recode	car	Recodificar categorías de una variable, permitiendo la recodificación por tramos. De especial utilidad cuando se precisa reducir las categorías de variables de nivel de medición de intervalo o razón, según tramos específicos de respuesta.
save	base	Guardar objetos desde el entorno de trabajo de R al disco duro del computador. Especialmente útil para guardar nuestras bases de datos en formato *.RData*
load	base	Cargar objetos desde el disco duro a nuestra sesión de trabajo en R. Especialmente útil para cargar bases de datos archivadas en formato *.RData*
class	base	Informar el tipo de objeto. Permite determinar cómo R ha configurado un conjunto específico de información (una base de datos, una variable en específico, etc.)

Así, de forma específica para la selección de las variables ya señaladas se utilizará la función select del paquete dplyr, como se observa a continuación:

Ejercicio 6.6

```
library(dplyr) #Cargar paquete, si no está cargado desde antes.

CEP <- select(CEP_csv, pond = POND, sexo = SEXO, region = REGION, edad =
DS_P2_EXACTA,
          satisfaccion_vida = SV_1, satisfaccion_chilenos = SV_2, eval_econ = MB_P2)
#Se indica base de datos, el nombre de variable a crear y los datos que la compondrán.
View(CEP) #Visualización de la base
```

Luego de ejecutar ese comando se habrá creado un nuevo objeto llamado CEP (del tipo base de datos) en el entorno de trabajo. Esta base de datos tiene la misma cantidad de casos (1.424) que la base de datos original (CEP_csv) pero presenta solamente las siete variables seleccionadas y renombradas mediante el comando select. Para corroborar el resultado de la construcción de esta nueva base de datos se la visualizará en formato planilla. Como se observa a continuación, esta base de datos contiene solamente las siete variables de interés.



Boccardo & Ruiz (2018). Uso de RStu... * | PRUEBA.Rmd | sintaxis para desarrollo.R* | CEP

Filter

	pond	sexo	region	edad	voto	satisfaccion_vida	satisfaccion_chilenos	identificacion_partidos	prob_voto	matri_igualitario
1	1	2	13	18	10	8	3	7	10	7
2	1	2	1	57	2	10	5	5	10	1
3	1	2	14	25	8	10	10	5	1	1
4	1	2	13	37	7	8	5	77	10	10
5	1	2	14	50	7	5	5	4	10	1
6	0	2	8	60	7	9	5	4	10	1
7	1	2	9	66	4	9	5	3	10	1
8	1	2	13	19	8	6	8	9	9	10
9	1	2	7	34	4	6	7	6	10	10
10	1	2	13	39	7	10	10	77	10	10
11	0	2	7	76	4	5	6	2	10	8
12	1	2	6	49	7	10	5	1	10	1
13	1	2	13	32	8	8	5	2	0	4
14	1	2	13	44	11	9	6	77	0	7
15	1	2	14	61	7	8	6	77	10	1
16	1	2	7	40	9	6	7	5	5	3

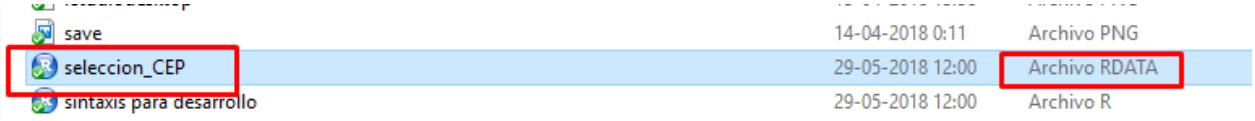
Showing 1 to 16 of 1,424 entries

Una vez hecha esta operación de selección de variables se habrá creado una nueva base de datos que contiene solo aquellas variables que resultan de interés para los análisis. Antes de proseguir conviene guardar tal objeto como una base de datos de formato R. Para esto se usa el comando save para guardar un objeto de formato (o extensión) .RData.

Ejercicio 6.7

```
save(CEP, file = "seleccion_CEP.RData")
```

Este comando creará un nuevo archivo de formato RData en la carpeta de trabajo. Este archivo puede usarse para enviar esta base de datos específica a un equipo de trabajo en el contexto de una investigación colectiva, o sencillamente contar con un archivo que ya contenga la base de datos solamente con las variables de interés.



El siguiente paso es explorar las características de tales variables y configurarlas para poder ejecutar los análisis estadísticos que precisemos para nuestro proceso de investigación. Se sugiere dejar en cero el espacio de trabajo utilizando algunas de las formas ya indicadas para hacerlo.

Ejercicio 6.8

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

Para continuar con este manual se sugiere cargar a la sesión de R (desde la carpeta de trabajo) la base construida solo con las variables de interés y guardada en formato RData (seleccion_CEP.RData).

Ejercicio 6.9

```
load("seleccion_CEP.RData")
```

Una primera opción para conocer las características de la base de datos es explorar los nombres de sus variables. Mediante el comando names aplicado sobre el objeto CEP se obtiene una lista con los nombres de cada columna de la base de datos. Esto resulta de utilidad para los procedimientos de manejo de datos pues no siempre se puede determinar con certeza el nombre de un objeto leyendo el encabezado de la columna. Al visualizarlos como se observa en el siguiente resultado se puede estar seguro del nombre exacto determinando si hay espacios en blanco antes o después de las letras que impliquen un nombre diferente al leído de manera directa.

Ejercicio 6.10

```
names(CEP)
## [1] "pond"              "sexo"              "region"
## [4] "edad"              "satisfaccion_vida" "satisfaccion_chilenos"
## [7] "eval_econ"
```

Otro comando de utilidad para conocer características generales de la base de datos es la función dim. Este comando, como se observa en el ejercicio 6.11, permite conocer la dimensión de la base de datos que se está explorando. El resultado de esta función arroja dos números: el primer número indica la cantidad de filas de la base de datos, mientras que el segundo número indica la cantidad de columnas. Las bases de datos de estudios sociales están construidas de manera que cada fila representa un caso y cada columna una variable, por lo que la dimensión de una base de datos indicará la cantidad de casos y variables (en ese orden).

Ejercicio 6.11

```
dim(CEP)
## [1] 1424    7
```

Finalmente, para conocer las características generales de las variables en la base de datos, y comenzar a evaluar que tipo de recodificaciones se deben realizar, se utiliza la función summary para obtener estadísticos descriptivos de cada una de las variables.

Ejercicio 6.12

```
summary(CEP)
```

```
> summary(CEP)
      pond             sexo           region           edad       satisfaccion_vida
 Min.   :0.0000   Min.   :1.000   Min.   : 1.00   Min.   :18.00   Min.   : 1.000
 1st Qu.:1.0000   1st Qu.:1.000   1st Qu.: 6.00   1st Qu.:36.00   1st Qu.: 6.000
 Median :1.0000   Median :2.000   Median : 9.00   Median :50.00   Median : 7.000
 Mean   :0.9993   Mean   :1.612   Mean   : 9.16   Mean   :49.87   Mean   : 7.924
 3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:13.00   3rd Qu.:64.00   3rd Qu.: 9.000
 Max.   :9.0000   Max.   :2.000   Max.   :15.00   Max.   :97.00   Max.   :99.000
 satisfaccion_chilenos   eval_econ
 Min.   : 1.000        Min.   :1.000
 1st Qu.: 4.000        1st Qu.:2.000
 Median : 5.000        Median :3.000
 Mean   : 8.879        Mean   :2.821
 3rd Qu.: 6.000        3rd Qu.:3.000
 Max.   :99.000        Max.   :9.000
>
```

¿Por qué resultan de interés estos resultados? Considerando los estadísticos descriptivos construidos se observa que algunas variables contienen códigos que refieren a casos perdidos. Es el caso de las variables satisfaccion_vida, satisfaccion_chilenos y eval_econ. Como se observa en la imagen 6.20, estas variables presentan valores 9 o 99 como máximos, siendo que se trata de variables que presentan un rango menor de valores posibles: la variable satisfaccion_vida presenta valores 99 como máximos cuando es una escala de 1 a 10; lo mismo ocurre con la variable satisfaccion_chilenos; finalmente, algo similar ocurre con la variable eval_econ, que presenta valores 9 como máximos, cuando es una escala del 1 al 5.

Tal información constituye una primera alerta sobre los ajustes a hacer sobre los datos y por tanto son un valioso insumo para el proceso de recodificación de variables.

Para avanzar en los análisis se efectuarán algunas recodificaciones. Primero que todo se observan las características de cada variable a recodificar. Para eso se utiliza las funciones table para crear una tabla de frecuencias absoluta y observar los valores de la variable, y class para conocer de qué tipo de objeto se trata (en este caso, la variable sexo).

Ejercicio 6.13

```
table(CEP$sexo)
##
##   1   2
## 553 871
class(CEP$sexo)
## [1] "integer"
```

El primer resultado muestra una variable que presenta sólo valores 1 y 2, a la vez que el

segundo resultado informa que se trata de un vector de tipo integer.

Luego de conocer estas características se cuenta con información suficiente para modificar la variable e incorporarla al análisis. Para recodificar se usará el comando mutate del paquete dplyr. Este comando permite transformar una variable guardando el resultado de tal operación en una nueva variable dentro de la misma base de datos. Resulta útil toda vez que permite editar variables sin perder la información (la variable) original.

Como se observa en las siguientes líneas de comando, el resultado de mutate se asigna a la base de datos. Luego de la función mutate, los argumentos son: i) el conjunto de datos del que provienen las variables (CEP, nuestra base de datos), ii) el nombre de la nueva variable a crear (sexo_chr) seguido de un igual y la operación de transformación de la variable original; en este caso, se emplea el comando recode del paquete dplyr cuyos argumentos son: la variable que se quiere transformar (CEP$sexo) y luego las equivalencias de cada categoría de respuesta. En este caso se indica que “1” se asigna a “hombre” y que “2” se asigna a “mujer”. Vale la pena enfatizar que cada categoría se expresa entre comillas y que cada equivalencia se separa de la siguiente mediante una coma.

Ejercicio 6.14

```
CEP <- mutate(CEP, sexo_chr = recode(CEP$sexo, "1" = "hombre", "2" = "mujer"))
table(CEP$sexo_chr)
##
## hombre  mujer
##    553    871
class(CEP$sexo_chr)
## [1] "character"
```

Al ejecutar tal recodificación y solicitar una tabla de frecuencias simple así como la información sobre el tipo de variable creada, ahora la tabla contiene las categorías “hombre” y “mujer”, haciendo más fácil su lectura, mientras que la variable recodificada es del tipo “character”, al contener ahora una codificación basada en letras. Si bien es un ejemplo muy simple, permite entender una primera utilidad de la recodificación de variables, que tiene que ver con facilitar nuestro acercamiento a los datos.

Adicionalmente, se aplicará una segunda recodificación sobre esta variable para convertirla en un vector de tipo factor, aplicando etiquetas para las dos categorías de respuesta (“Hombre” y “Mujer”). ¿Para qué resulta de utilidad esto si ya contamos con resultados entendibles?: pues bien, en el ejercicio anterior al recodificar a valores de tipo character, la información numérica almacenada en la variable CEP$sexo se perdió al ser reemplazada por letras. Para determinados análisis, precisaremos contar con ambos niveles de información: los valores numéricos asociados a las respuestas, pero también las etiquetas de tales valores, que indican lo que representan los números almacenados en la base de datos en términos analíticos.

Ejercicio 6.14 (continuación)

```
CEP <- mutate(CEP, sexo_factor = factor(CEP$sexo,
                                labels = c("Hombre", "Mujer")))
```

Como se observa en el ejercicio 6.14, el comando factor permite incorporar las etiquetas a cada código de respuesta. En este caso, al ingresar los valores Hombre y Mujer, en el argumento labels, lo que estamos haciendo es asociar las etiquetas indicadas a los

valores 1 y 2 en que está codificada la variable.

Ahora se recodificará la variable región indicando que sólo habrá dos categorías: “otras regiones” y “región Metropolitana”. Como en el caso anterior, primero conviene observar las características generales de la variable: interesa saber cuántos casos están en la categoría Región Metropolitana para así asegurar que luego de la transformación de la variable, la estructura de casos siga el mismo patrón. Primero exploraremos las características generales de la variable de interés.

Ejercicio 6.15

```
table(CEP$region) #Observar características de la variables
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
##  24  57  24  52 150  82  94 192  98  69   5  17 501  39  20
class(CEP$region)
## [1] "integer"
```

En base a estos primeros resultados se observa que se trata de una variable cuyo nivel de medición es nominal. La categoría “13”, que equivale a la región Metropolitana, presenta 501 casos. También sabemos que la variable está configurada como un vector de tipo integer. Así, usando el comando mutate se recodificará la variable en una nueva denominada region_factor; sin embargo, al interior del comando que se utilizaría por defecto, introduciremos una variante.

Si introducimos una función como argumento dentro una función determinada, R siempre aplica las funciones que corresponden al paquete de la función principal inicialmente indicada (en este caso, es el paquete dplyr). Para el caso del comando recode, el provisto por este paquete no resulta de utilidad para recodificar variables con gran cantidad de categorías, pues no permite realizar una recodificación por tramos de información. Por ello, antes de la función recode - que implicaría utilizar aquella incorporada en dplyr - incorporamos el argumento car::, lo que fuerza a que se ejecute el comando recode incluido en un paquete distinto llamado car, para así poder recodificar indicando tramos de datos.

Para el caso de la siguiente recodificación se indica el mismo valor para los códigos del 1 al 12, y del 13 al 14, mientras que se asigna un valor diferente para el valor 13 (correspondiente a la región Metropolitana). Primero se recodifica en una nueva variable indicando los tramos de recodificación ya explicados. Posteriormente se sobrescribe la variable asignándole el resultado de la operación de convertirla en una variable de tipo factor, aplicando etiquetas para las dos categorías de respuesta ahora existentes.

Ejercicio 6.15 (continuación)

```
#Transformar a una variable distinta, categorías "Otras regiones" y "Región Metropolitana".
CEP <- mutate(CEP, region_factor = car::recode(CEP$region, "1:12 = 1; 13 = 2; 14:15 = 1"))

#Sobreescribir variable con resultado de convertir a factor incorporando etiquetas
CEP$region_factor <- factor(CEP$region_factor,
                            labels = c("Otras regiones", "Región Metropolitana"))

table(CEP$region_factor) #Se sigue manteniendo la estuctura de casos.
##
##       Otras regiones Región Metropolitana
```

```
##                923                501
class(CEP$region_factor) #Cambia el formato del objeto.
## [1] "factor"
```

El resultado muestra que, por un lado, se mantuvo la estructura de casos, con 501 casos en la región Metropolitana y 923 casos en otras regiones. Eso indica que la recodificación se hizo de manera adecuada. Además, la tabla muestra etiquetas y al solicitar el formato de la variable indica que es un vector de tipo *factor*. En síntesis, la recodificación se efectuó de manera óptima.

Ahora se recodificarán las variables *satisfacción con la vida propia* y percepción de *satisfación que los chilenos en general sienten con su propia vida*. Primero se exploran ambas variables.

Ejercicio 6.16

```
#Explorar variable "satisfacción con la propia vida"
class(CEP$satisfaccion_vida)
## [1] "integer"
table(CEP$satisfaccion_vida)
##
##   1   2   3   4   5   6   7   8   9  10  88  99
##  20  10  30  63 176 169 256 244 142 304   4   6
summary(CEP$satisfaccion_vida)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   6.000   7.000   7.924   9.000  99.000
```

El análisis exploratorio de esta primera variable28 permite identificar que la variable está compuesta por valores discretos entre 1 y 10, aunque se observa también valores 88 y 99 en la distribución que - debido a la lectura del cuestionario y la ficha técnica de la encuesta - se sabe que denotan las categorías “no sabe”y “no responde.”29 Es posible afirmar entonces que el nivel de medición de esta variables de tipo intervalar, pues incluye más de 7 categorías de respuesta.

```
#Explorar variable "percepción de la satisfacción que los chilenos tienen con su vida"
class(CEP$satisfaccion_chilenos)
## [1] "integer"
table(CEP$satisfaccion_chilenos)
##
##   1   2   3   4   5   6   7   8   9  10  88  99
##  33  21  97 216 469 241 160  56  24  47  52   8
summary(CEP$satisfaccion_chilenos)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   5.000   8.879   6.000  99.000
```

Algo similar ocurre con esta segunda variable. Al igual que la anterior, presenta valores 88 y 99 como casos válidos, lo que altera la distribución de casos ceñida a la escala de 1 a 10. Tal como en la variable anterior, también es posible afirmar - dado que presenta más de 7 categorías - que el nivel de medición de esta variable es de intervalo.

Por tanto, se efectuará una operación de transformación de estas variables, asignando el valor lógico NA a aquellos valores que representan casos perdidos (sin respuesta). Este valor lógico servirá pues R lo reconoce como un valor que no puede utilizarse para operaciones matemáticas, a la vez que no representa un valor alfanumérico contable, por lo que no alterará el tipo de vector (en este caso, numérico de tipo integer). Entonces se ejecutan las siguientes funciones, para cada variable:

Ejercicio 6.16 (continuación)

```
CEP$satisfaccion_vida[CEP$satisfaccion_vida==88]<- NA #Asignar NA a casos con valor 88
CEP$satisfaccion_vida[CEP$satisfaccion_vida==99]<- NA #Asignar NA a casos con valor 99

CEP$satisfaccion_chilenos[CEP$satisfaccion_chilenos==88]<- NA
CEP$satisfaccion_chilenos[CEP$satisfaccion_chilenos==99]<- NA
```

Habiendo ejecutado tales instrucciones, R habrá cambiado los valores 88 y 99 de cada variable por el valor lógico NA. Para evaluar si la transformación de datos resultó en el sentido esperado, a continuación se solicita una tabla de frecuencias y estadísticos de resumen para cada variable.

```
table(CEP$satisfaccion_vida) #Ver resultado de codificación
##
##   1   2   3   4   5   6   7   8   9  10
##  20  10  30  63 176 169 256 244 142 304
summary(CEP$satisfaccion_vida)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   6.000   7.000   7.311   9.000  10.000      10
table(CEP$satisfaccion_chilenos)
##
##   1   2   3   4   5   6   7   8   9  10
##  33  21  97 216 469 241 160  56  24  47
summary(CEP$satisfaccion_chilenos)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   4.000   5.000   5.334   6.000  10.000      60
```

Como se observa en los resultados, si luego de la recodificación se solicitan tablas de frecuencias para ambas variables, estas ya no mostrarán los casos perdidos (88 y 99). Y si se solicitan estadísticos descriptivos, se observará que ahora los valores mínomo y máximo de la variable en la base de datos, coinciden con los valores mínimos y máximos del rango de la variable discernible desde el cuestionario y ficha técnica del estudio, indicándose además la cantidad de casos NA que han resultado excluidos de los análisis.

Ahora se explorará la variable que mide la percepción de la persona entrevistada respecto a la situación económica del país. Como se observa en el ejercicio 6.17, se utiliza el comando class, table y summary para efectuar una aproximación exploratoria a la variable.

Ejercicio 6.17

```
#Explorar variable "evaluación de la economía"
class(CEP$eval_econ)
## [1] "integer"
table(CEP$eval_econ)
##
##   1   2   3   4   5   8   9
##  74 397 730 204   5   8   6
summary(CEP$eval_econ)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.000   2.000   3.000   2.821   3.000   9.000
```

Como se observa en los resultados, la variable es una escala de acuerdo compuesta por valores del 1 al 5, aunque también se observan valores 88 y 99 en la distribución que gracias a la información disponible en el cuestionario y ficha técnica de este estudio, sabemos que representan a a las categorías “no sabe” y “no contesta” (casos perdidos). Así, es posible afirmar que se trata de una variable de nivel de medición ordinal.

Para efectos de simplificar las respuestas para su análisis, esta variable se recodificará en tres categorías de evaluación (negativa, neutra y positiva). Asimismo, en el ejercicio 6.17 también se indicará una segunda forma de asignar casos perdidos (valores NA) esta vez desde una recodificación vía comando mutate.

la variable está compuesta por valores discretos entre 1 y 10, aunque se observa también valores 88 y 99 en la distribución que - debido a la lectura del cuestionario y la ficha técnica de la encuesta - se sabe que denotan las categorías “no sabe”y “no responde”. Es posible afirmar entonces que el nivel de medición de esta variables de tipo intervalar, pues incluye más de 7 categorías de respuesta.

Ejercicio 6.17 (continuación)

```
#Recodificar variable a 3 tramos: positiva, neutra, negativa

#Valores perdidos se asignan en la misma codificación,
    #con argumento else ("todos los demás valores")
CEP <- mutate(CEP, eval_econ_factor =
            car::recode(CEP$eval_econ,
                    "1:2 = 1; 3 = 2; 4:5 = 3; else = NA"))

CEP$eval_econ_factor <- factor(CEP$eval_econ_factor,
                        labels = c("Positiva", "Neutra", "Negativa"))

table(CEP$eval_econ_factor)
##
## Positiva   Neutra Negativa
##      471      730      209
summary(CEP$eval_econ_factor)
## Positiva   Neutra Negativa     NA's
##      471      730      209       14
```

Habiendo efectuado estas transformaciones sobre las variables originales, ya se cuenta con variables de los diferentes tipos para efectuar análisis de estadística univariada: variables cuyo nivel de medición es nominal, ordinal, intervalar y de razón. En el siguiente capítulo se verá como ejecutar análisis estadísticos univariados, en sus variantes de alcance muestral y poblacional.

Estadística descriptiva con RStudio

Habiendo construido y configurado una base de datos ad hoc para nuestros análisis, en el presente capítulo se presenta un modo de calcular frecuencias, medidas de tendencia central, de posición y de dispersión a nivel de la estimación puntual y del parámetro poblacional.32

Para efectos de este manual distinguiremos aquellos estadísticos que se calculan desde una muestra (probabilística o no probabilística) de aquellos que se obtienen vía registros administrativos o de censos. Específicamente, en el caso de estadísticos que se estiman a partir de una muestra probabilística avanzaremos desde la estimación puntual al cálculo del parámetro poblacional (con su respectivo intervalo de confianza, nivel de confianza y error de estimación). Esto último nos permitirá estimar los valores que la estimación puntual alcanza en la población (que representa la muestra) y si existen diferencias estadísticamente significativas a nivel del parámetro entre dos grupos (prueba de hipótesis).

El presente capitulo se ordena como sigue: en el primer apartado se indicará cómo realizar la estimación puntual de diferentes estadísticos descriptivos usando R (medidas de tendencia central, distribuciones de frecuencia y medidas de posición, medidas de dispersión y forma de distribuciones) y su exportación a planillas de cálculo para su presentación en reportes de investigación académica o profesional; en el segundo apartado se presentará la forma de realizar inferencia estadística, construyendo intervalos de confianza de parámetros, a partir de estimaciones puntuales, y calculando indicadores insesgados a partir de factores de expansión.

Estimación puntual de estadísticos descriptivos usando R

Medidas de tendencia central: cálculo de la media, mediana y moda

El cálculo de la media es sencillo en cuanto a los códigos y aritmética necesarios. En las dos siguientes líneas de comandos se observa el cálculo de una media simple y luego de una media recortada. En ambos casos se utiliza el comando mean, indicando la variable sobre la cual se desea hacer el cálculo, acompañado del argumento na.rm = TRUE para excluir del cálculo a los casos dados como perdidos (valores NA que ya fueron codificados al preparar las variables). Para el caso de la media recortada, se agrega el argumento trim, que permite indicar la proporción de casos que se eliminan en cada extremo de la distribución.33 Así, se obtiene la estimación puntual de ambos estadísticos.

Ejercicio 7.1

```
mean(CEP$satisfaccion_vida, na.rm = TRUE)
## [1] 7.311174
mean(CEP$satisfaccion_vida, na.rm = TRUE, trim = 0.025)
## [1] 7.390625
```

Como se observa en el ejercicio 7.1 la media aritmética arroja un valor de 7,31 en una una escala de 1 a 10, mientras la media recortada al 5% presenta un valor de 7,39 (ambos valores se redondean al segundo decimal). Como fue anticipado, la interpretación de estos resultados (y sus diferencias) se explorarán en el siguiente apartado.

Para el caso de la mediana, se trata de un comando similar. Se utiliza el comando median, indicando la variable sobre la cual se desea hacer el cálculo, acompañado del

argumento na.rm = TRUE para excluir del cálculo a los casos dados como perdidos (valores NA que ya fueron codificados al preparar las variables). De esa forma se obtiene la estimación puntual de este estadístico.

Ejercicio 7.2

```
median(CEP$satisfaccion_vida, na.rm = TRUE)
## [1] 7
```

Como se ve en el ejercicio 7.2, el valor de la mediana es 7, lo que indica que la mitad de los casos indica tener una satsfacción con la propia vida menor a 7, considerando una escala de 1 a 10.

Para el cálculo de la moda, debe haberse instalado previamente el paquete modeest y haberlo cargado en la sesión de trabajo; esto permitirá contar con una función que calculará de forma automática el o los valores más frecuentes de la distribución, herramienta que no existe en los paquetes básicos de R. Así, como se ve en el ejercicio 7.3, se utiliza el comando mfv34 para calcular la moda de la variable edad. El resultado de este comando indica el valor, o los valores, con más frecuencia dentro de la distribución de casos de la variable señalada.

Ejercicio 7.3

```
#Ejecutar previamente "install.packages("modeest")"
library(modeest)
mfv(CEP$edad) #Indica el o los valores con más frecuencia
## [1] 50
```

En este caso, la moda de la variable edad es 50 años, lo que indica que el valor que más se repite en la distribución de casos de tal variable corresponde a tal respuesta.

Medidas de posición: cálculo de frecuencias absolutas y relativas, cuantiles

El cálculo de tablas de frecuencias absolutas para una variable se efectúa mediante el comando table, indicando como argumento de la función la variable sobre la cual se ejecuta el cálculo. En este sub apartado, trabajaremos con la variable ordinal de nuestra base de datos que refiere a una evaluación de la economía (CEP$eval_econ_factor) por parte de la persona encuestada.

Ejercicio 7.4

```
tabla <- table(CEP$eval_econ_factor)
tabla
##
## Positiva   Neutra Negativa
##      471      730      209
```

De forma muy sucinta, los resultados del ejercicio 7.4 muestran un importante predominio de opiniones “neutras” con 730 casos, seguida por una concentración de 471 casos en respuestas “positivas” y 209 casos en respuestas “negativas”.

Para facilitar la comparación con otros datos, estos resultados pueden expresarse como frecuencias relativas. Para el cálculo de frecuencias relativas, se usa el objeto tabla de frecuencias simples ya construido (tabla): como se ve en el ejercicio 7.5, sobre tal objeto, se ejecuta la función prop.table que construye una nueva tabla en la que cada celda es la división simple entre la cantidad de casos de la categoría y el total de casos.

Ejercicio 7.5

```
#Frecuencias relativas (proporciones)
prop.table(tabla)
##
##  Positiva    Neutra  Negativa
## 0.3340426 0.5177305 0.1482270
```

De tal forma, los resultados del ejercicio 7.5 confirman lo ya observado con frecuencias absolutas. En este caso, los valores se presentan como proporciones de una unidad (1). Así, se observa que la categoría “neutra” concentra la mayor cantidad de casos con aproximadamente la mitad de los mismos (0,52), mientras que la categoría “positiva” concentra un tercio de las respuestas (0,33).

Ahora bien, para facilitar aún más la lectura y divulgación de estos resultados resultará de utilidad convertir estas frecuencias relativas en porcentajes. Como se observa en el ejercicio 7.6, para construir una tabla de porcentajes se multiplica el resultado de calcular la tabla de proporciones por 100; si a ello además se le agrega la función round, es posible configurar un resultado redondeado a dos decimales.

Ejercicio 7.6

```
#Frecuencias relativas (porcentajes)
prop.table(tabla)*100
##
## Positiva   Neutra Negativa
## 33.40426 51.77305 14.82270
round((prop.table(tabla)*100),2)
##
## Positiva   Neutra Negativa
##    33.40    51.77    14.82
```

De tal modo, los resultados del ejercicio 7.6 permiten afirmar que la mayor cantidad de respuestas se concentran en una evaluación “neutra” con el 51,8% del total, seguido por evaluaciones positivas que representan un 33,4% del total de respuestas, y finalmente las respuestas que reflejan evaluaciones negativas, que alcanzan un 14,8% de los casos.

Una última configuración que resulta de utilidad para el examen de distribuciones de frecuencias es la modalidad de cantidades acumuladas. Este tipo de tablas se construye a partir de tablas de frecuencias absolutas y relativas (sea en su modalidad proporcional o porcentual). En el ejercicio 7.7 se aplica la función cumsum a las diferentes modalidades de tablas de frecuencias construidas a partir del objeto tabla, que almacena una tabla de frecuencias absolutas. Esta función (cumsum) permite construir tablas de frecuencias acumuladas, a partir del cálculo de cada distribución de frecuencias ya construidas en los ejercicios anteriores.

Ejercicio 7.7

```
#Frecuencias absolutas acumuladas
cumsum(tabla)
## Positiva   Neutra Negativa
##      471     1201     1410
#Frecuencias relativas acumuladas
cumsum(prop.table(tabla))
```

```
##  Positiva    Neutra  Negativa
## 0.3340426 0.8517730 1.0000000
#Porcentaje acumulado redondado en dos decimales
round(cumsum(prop.table(tabla)*100),2)
## Positiva   Neutra Negativa
##    33.40    85.18   100.00
```

Así los resultados del ejercicio 7.7, permiten observar que una amplia mayoría de los casos presenta una evaluación neutra o positiva de la economía nacional con el 85,2% de los casos (en términos relativos), lo que equivale a 1.201 casos en términos absolutos. Sólo un 14,8% de las respuestas se concentran en una evaluación negativa de la economía.

Una última forma de describir la concentración de casos de una distribución es mediante los cuantiles. Tales estadísticos permiten describir la concentración de casos según cualquier posición relativa de los datos en la distribución, que resulte de interés. Para el cálculo de cuantiles la función quantile permite calcular los casos equivalentes a diferentes proporciones de la distribución (ejercicio 7.8). El primer argumento de la función es la variable a considerar, en este caso CEP$atisfaccion_chilenos, que refiere a la evaluación que la persona cree los chilenos tiene de su vida, en una escala de 1 a 10; luego, mediante el argumento prob se indica en formato vector los cuantiles a calcular (expresados como proporción). Si existen casos perdidos (codificados como NA) se debe indicar el argumento na.rm = TRUE.

Ejercicio 7.8

```
quantile(CEP$satisfaccion_chilenos, prob = c(0.25, 0.5, 0.75), na.rm = TRUE)
## 25% 50% 75%
##   4   5   6
```

De tal forma, en el ejercicio 7.8 se solicita al software el cálculo del cuartil 1 (percentil 25 o caso que corta el 25% de la distribución), el cuartil 2 (percentil 50, mediana o caso que corta el 50% de la distribución) y el cuartil 3 (percentil 75, o caso que corta el 75% de la distribución). Los resultados obtenidos en el ejercicio 7.6 permiten concluir que solamente el 25% superior de los casos cree que los chilenos tienen una evaluación de su vida mayor o igual a 6, mientras que la mitad de los casos cree que los chilenos tienen una evaluación de su vida igual o menor a 5, en una escala del 1 al 10.

Medidas de dispersión: rango, varianza, desviación estándar y coeficiente de variación

En relación a las medidas de dispersión se partirá por cálculo del rango. Los valores mínimo y máximo de la distribución pueden calcularse de manera simultánea con la función range, indicando como argumentos la variable de interés y adicionando también el argumento na.rm = TRUE en el caso de que hubieran sido codificados como NA los valores perdidos. Como se observa en el ejercicio 7.9, los valores mínimo y máximo pueden calcularse de forma independiente, con las funciones min y max respectivamente, que siguen la misma lógica que la función range. En este caso se trabajará sobre la variable CEP$edad.

Ejercicio 7.9

```
range(CEP$edad, na.rm = TRUE)
## [1] 18 97
min(CEP$edad, na.rm = TRUE)
## [1] 18
max(CEP$edad, na.rm = TRUE)
```

```
## [1] 97
max(CEP$edad, na.rm = TRUE) - min(CEP$edad, na.rm = TRUE)
## [1] 79
```

Luego, para conocer el rango de los valores se debe restar el valor máximo al mínimo. Los resultados del ejercicio 7.9 indican que, para el caso de la variable edad el rango es de 79 años, siendo su valor mínimo 18 años y su valor máximo 97 años.

Otras dos medidas de uso generalizado para describir la dispersión de una variable son la varianza y la desviación estándar. El cálculo de ambas medidas, sigue la misma lógica en lo que refiere a la estructura de la función utilizada. Respectivamente, las funciones utilizadas son var para varianza y sd, para desviación estándar. Como se observa en el ejercicio 7.10 también debe incluirse el argumento na.rm = TRUE en el caso de que hubieran sido codificados como NA los valores perdidos. En este ejercicio se trabaja sobre la variable CEP$atisfaccion_chilenos, que refiere a la evaluación que la persona cree los chilenos tiene de su vida, en una escala de 1 a 10.

Ejercicio 7.10

```
var(CEP$satisfaccion_chilenos, na.rm = TRUE)
## [1] 3.017771
sd(CEP$satisfaccion_chilenos, na.rm = TRUE)
## [1] 1.737173
```

Atendiendo a los resultados del ejercicio 7.10 puede observarse que la dispersión de la variable que representa la evaluación que cada entrevistado/a cree que los chilenos tienen sobre su propia vida es de 3,02 unidades atendiendo a la varianza y de 1,73 unidades atendiendo a la desviación estándar.

Otra medida que debemos aprender a calcular para nuestros análisis es el coeficiente de variación. Como se observa en el siguiente ejercicio (7.11), el coeficiente de variación puede calcularse de manera sencilla, ejecutando una división simple entre la desviación estándar y la media de la variable de interés. Sin embargo, también existe la función coefficient.variation (del paquete FinCal) que permite calcular este estadístico indicando como argumento la media y desviación estándar de la variable de interés. En este caso tales valores se incluyen indicando la función para calcularlos: de tal modo se evitan errores humanos al momento de especificar los valores, ya sea por tipeo del valor o por su aproximación decimal. En este ejercicio trabajamos con la variable edad

Ejercicio 7.11

```
sd(CEP$edad)/mean(CEP$edad)
## [1] 0.3566407
#Asegurarse de ejecutar previamente el comando "install.packages("FinCal")"
library(FinCal)
coefficient.variation(sd=sd(CEP$edad), avg = mean(CEP$edad))
## [1] 0.3566407
```

De este modo, los resultados del ejercicio 7.11 muestran que la variable edad presenta una dispersión del 35,67% según el valor obtenido en el coeficiente de variación.

Forma de una distribución: simetría, curtosis y normalidad

Un último grupo de estadísticos que resulta de importancia para el análisis estadístico de variables refiere a aquellos que dan cuenta de la forma general que asume la distribución de una variable. En específico, mostraremos en este apartado el cálculo e interpretación de estadísticos que dan cuenta de la simetría, curtosis y normalidad de una variable.

En el siguiente ejercicio (7.12) se muestra cómo calcular los coeficientes de simetría y curtosis mediante las funciones skew y kurtosi presentes en el paquete psych.

Ejercicio 7.12

```
library(psych)
skew(CEP$satisfaccion_vida)
## [1] -0.5335721
kurtosi(CEP$satisfaccion_vida)
## [1] -0.1461258
```

Como se observa en los resultados del ejercicio 7.12, los coeficientes de simetría y curtosis para la variable satisfaccion_vida son de -0,53 y -0.15 unidades de forma respectiva. Dado que tales valores están expresados en la unidad de medida de la variable en cuestión, no son útiles para comparar variables de diferentes unidades de medición. Es por eso que en el ejercicio 7.13 se calculan los coeficientes de simetría y curtosis estandarizados: para ello, se divide la simetría calculada, por la raíz cuadrada del valor 6 (este número es una constante en la fórmula) dividido en la cantidad de casos de la variable (en este caso, 1.401).

Ejercicio 7.13

```
skew(CEP$satisfaccion_vida)/sqrt(6/1401)
## [1] -8.153359
kurtosi(CEP$satisfaccion_vida)/sqrt(6/1401)
## [1] -2.232906
```

Un criterio general para determinar si los coeficientes de simetría y curtosis reflejan una variable semejante a una distribución normal es que ambos valores se encuentren entre -2 y 2. Como puede observarse en los resultados, ambos coeficientes escapan a tal rango por lo que se observa una distribución poco similar a una normal.

Finalmente, la forma más certera para evaluar si la distribución de datos de una variable se comporta según los parámetros que asume una distribución normal es aplicando un test estadístico específicamente orientado a tal evaluación. Para ello, se diferencia el test de Shapiro Wilk y el de Kolmogorov Smirnov. El primero se adecua a muestras pequeñas (menores a 50 casos), mientras que el segundo sirve para muestras de entre 50 y 1.000 casos.

El ejercicio 7.14 muestra la aplicación de tales pruebas sobre la variable edad. En específico, se utilizan las funciones shapiro.test y ks.test del paquete básico de R (stats). Su ejecución es bastante sencilla pues sólo requieren como argumento la variable sobre la cual se aplicará la prueba.

Ejercicio 7.14

```
#Prueba de Shapiro Wilk (muestras pequeñas)
shapiro.test(CEP$edad)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  CEP$edad
## W = 0.97588, p-value = 1.037e-14
#Prueba Kolmogorov Smirnov (muestras grandes)
ks.test(CEP$edad, "pnorm", mean(CEP$edad, na.rm=T), sd(CEP$edad,na.rm=T))
## Warning in ks.test(CEP$edad, "pnorm", mean(CEP$edad, na.rm = T),
## sd(CEP$edad, : ties should not be present for the Kolmogorov-Smirnov test
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  CEP$edad
## D = 0.056969, p-value = 0.0001935
## alternative hypothesis: two-sided
```

Para interpretar uno u otro estadístico, debemos hacerlo atendiendo a la cantidad de casos de la variable en cuestión. En este caso, la variable edad presenta 1.424 casos. Esto provoca que ninguno de los resultados sea confiable pues las pruebas tienden a fallar cuando no se cumple el supuesto del margen de casos tolerado.

No obstante, vale la pena recordar que para determinar normalidad univariada se busca un valor p mayor a 0,05 para no rechazar la hipótesis nula de igualdad entre la distribución normal teórica y la distribución empírica de datos que estamos evaluado. Si los resultados fueran válidos en términos estadísticos, en ambas situaciones deberíamos rechazar la normalidad pues se aprueba la hipótesis alternativa dado que ambas pruebas presentan valores p menores a 0,05.

Tablas para informes e interpretación de resultados

En el presente apartado se presenta una forma para poder exportar resultados construidos en R a un formato compatible con softwares de tipo planilla de datos como Excel de Microsoft Office o Calc de Libre u Open Office. Es importante recalcar que aquí se enseña una forma simple para exportar resultados, que no se apoya en paquetes especializados en tales procedimientos. Una aproximación general a formas más avanzadas de construcción de reportes de resultados con formato, se presenta en el capítulo 9 de este manual.

En términos generales, el método de tratamiento de datos que presentaremos a continuación supone tres operaciones: en primer lugar, los resultados calculados deben existir en forma de matriz de datos en el entorno de trabajo; en segundo término, tales objetos deben imprimirse a un archivo de tipo CSV (para lo cual emplearemos la función write.csv2), el cual podrá ser abierto con alguno de los softwares tipo planilla de cálculo ya mencionados; así, en tercer lugar, tales datos podrán manipularse y editarse, ajustando su formato según lo que se desee, siendo fácilmente exportables hacia procesadores de texto.

Distribuciones de frecuencias

En primera instancia se guardarán como objetos las tablas de frecuencias ya existentes en nuestro entorno de trabajo de R (creadas en apartados anteriores de este capítulo). En el ejercicio 7.15 se observa la creación de tres objetos que contendrán, de forma respectiva, una tabla de frecuencias absolutas, una tabla de frecuencias relativas (porcentajes) y una tabla de frecuencia relativa acumulada (porcentajes) para la variable

evaluación de la economía chilena (eval_econ_factor).

Ejercicio 7.15

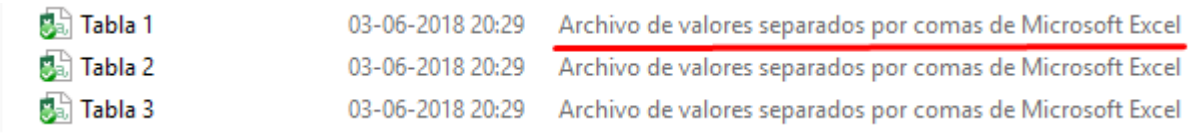
```
f <- table(CEP$eval_econ_factor)
f_porc <- round((prop.table(tabla)*100),2)
f_porc_acum <- round(cumsum(prop.table(tabla)*100),2)
```

Luego, con la función write.csv2 tales tablas pueden imprimirse, de forma individual, a archivos de tipo CSV. Como se observa en la continuación del ejercicio 7.15, al ejecutar esta función se indica el objeto a imprimir como planilla y luego con el argumento file se indica (entre comillas) el nombre del archivo a crear con su respectiva extensión (.csv en este caso).

Ejercicio 7.15 (continuación)

```
write.csv2(f, file = "Tabla 1.csv")
write.csv2(f_porc, file= "Tabla 2.csv")
write.csv2(f_porc_acum, file= "Tabla 3.csv")
```

El resultado de la anterior operación será la creación de tres archivos de tipo CSV en la ubicación definida como carpeta de trabajo para la sesión de R. La aparición de tales archivos en la carpeta de trabajo que hemos definido para nuestra sesión de R se observan en la imagen a continuación.



A partir de estos archivos - como se observa en la siguiente imagen - es posible unificar y editar las tablas en un formato adecuado para su inserción en reportes de investigación académica o profesional. No se profundiza aquí en el manejo de datos en softwares tipo planilla de cálculo, pero las simples operaciones copiar y pegar permiten consolidar los diferentes resultados en una sola tabla. Las tres primeras tablas de la imagen 7.2 son los datos impresos por R en archivos CSV independientes; la cuarta tabla es una tabla configurada manualmente mediante Microsoft Excel, lista para ser copiada y pegada en cualquier procesador de texto.

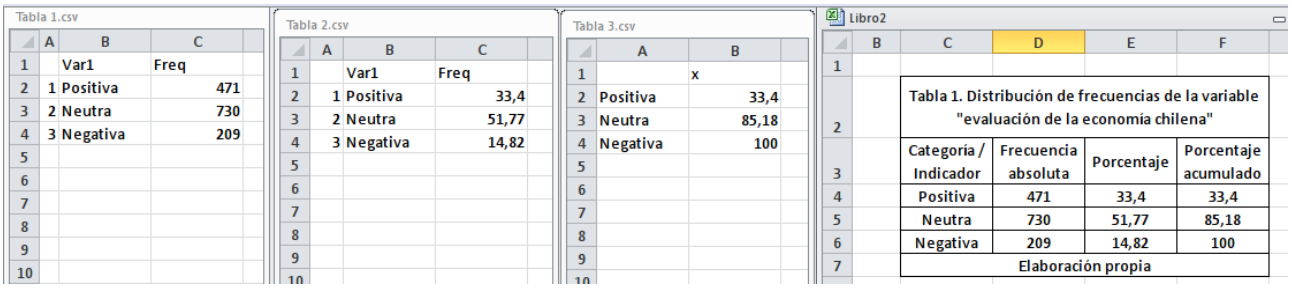


Tabla 1.csv

	Var1	Freq
1	Positiva	471
2	Neutra	730
3	Negativa	209

Tabla 2.csv

	Var1	Freq
1	Positiva	33,4
2	Neutra	51,77
3	Negativa	14,82

Tabla 3.csv

	x
Positiva	33,4
Neutra	85,18
Negativa	100

Libro2

Tabla 1. Distribución de frecuencias de la variable "evaluación de la economía chilena"

Categoría / Indicador	Frecuencia absoluta	Porcentaje	Porcentaje acumulado
Positiva	471	33,4	33,4
Neutra	730	51,77	85,18
Negativa	209	14,82	100
Elaboración propia			

Ahora bien, es importante recalcar cómo se debe realizar la interpretación de una tabla de resultados como la presentada (a continuación se incorpora en un formato adecuado para este documento interactivo).

Distribución de frecuencias de la variable Evaluación de la Economía Chilena

Categoría/Indicador	Frecuencia absoluta	Porcentaje	Porcentaje acumulado
Positiva	471	33,4	33,4
Neutra	730	51,77	85,18
Negativa	209	14,82	100

Reiterando lo señalado para los resultados del ejercicio 7.7, se observa que una amplia mayoría de los casos presenta una evaluación neutra o positiva de la economía nacional con el 85,2% de los casos (en términos relativos), lo que equivale a 1.201 casos en términos absolutos. Dentro de ello destaca que un tercio (33,4% o 471 casos) de las personas declaran una opinión neutra. Finalmente, es posible señalar que sólo un 14,8% de las respuestas (209 casos) se concentran en una evaluación negativa de la economía.

Estadísticos descriptivos

A continuación, se explicarán dos formas para calcular de manera agregada diversos estadísticos descriptivos de interés, para la posterior construcción de una tabla exportable a formato planilla que contenga tales valores.

Una primera forma es utilizando la función summary, que se ejecuta con la variable de interés como principal argumento (en este caso, la variable CEP$satisfaccion_vida). A diferencia del cálculo de estadísticos de forma individual, en este caso no es necesario indicar mediante argumento el tratamiento de los casos codificados como NA pues esta función reconoce de manera automática tales valores lógicos y los excluye del análisis.

Mediante este simple comando (ejercicio 7.16) se obtiene una tabla de estadísticos de resumen: valores mínimo y máximo, primer cuartil, mediana, media, tercer cuartil y cantidad de valores codificados como perdidos (valores NA).

Ejercicio 7.16

```
summary(CEP$satisfaccion_vida)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   6.000   7.000   7.311   9.000  10.000      10
```

Ahora bien, para exportar estos resultados a una planilla de cálculo se deberán ejecutar diversas operaciones que se muestran en el siguiente ejemplo (7.17). Primero, los resultados del comando summary se guardan como un objeto específico (al cual se le asigna el nombre de descriptivos).

Ejercicio 7.17

```
#Guardar summary como objeto
descriptivos <- summary(CEP$satisfaccion_vida)
```

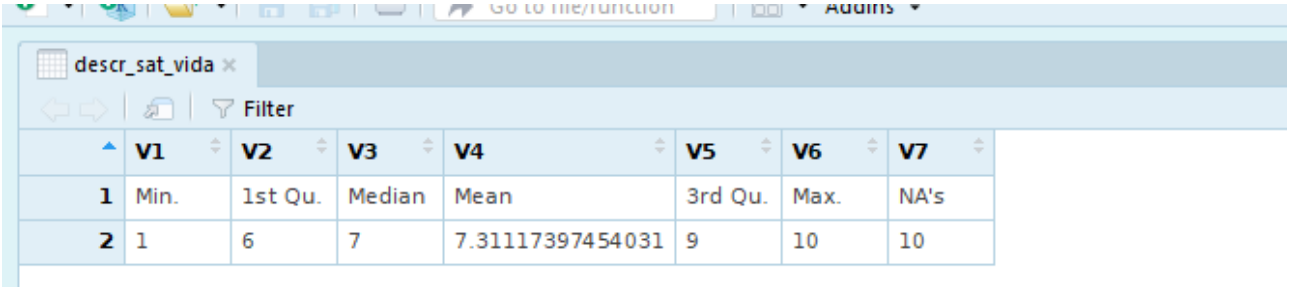
Luego, si se le aplican las funciones names y as.numeric a tal objeto (descriptivos) se obtiene un vector que contiene los encabezados de los resultados y otro vector que contiene su valores numéricos: en la continuación del ejercicio 7.17 se observa el resultado de aplicar ambas funciones sobre el objeto mencionado, que muestran en primer lugar los encabezados de la tabla y luego sus valores.

Finalmente, usando la función as.data.frame se configura como matriz de datos el resultado de unir como filas - mediante la función rbind (unir como filas, o row bind en inglés) el encabezado de los estadísticos descriptivos y sus valores numéricos. Tal objeto se nombra como descr_sat_vida y se configura como un objeto de tipo data.frame.

Ejercicio 7.17 (continuación)

```
#Ver nombres y valores del objeto
names(descriptivos)
## [1] "Min."    "1st Qu." "Median"  "Mean"    "3rd Qu." "Max."    "NA's"
as.numeric(descriptivos)
## [1]  1.000000  6.000000  7.000000  7.311174  9.000000 10.000000 10.000000
#Configurar como matriz de datos
descr_sat_vida <- as.data.frame(rbind(names(descriptivos),
as.numeric(descriptivos)))
View(descr_sat_vida)
```

El resultado del ejercicio (como se observa en la imagen 7.3) muestra cómo ya hemos configurado el resultado en formato matriz de datos.



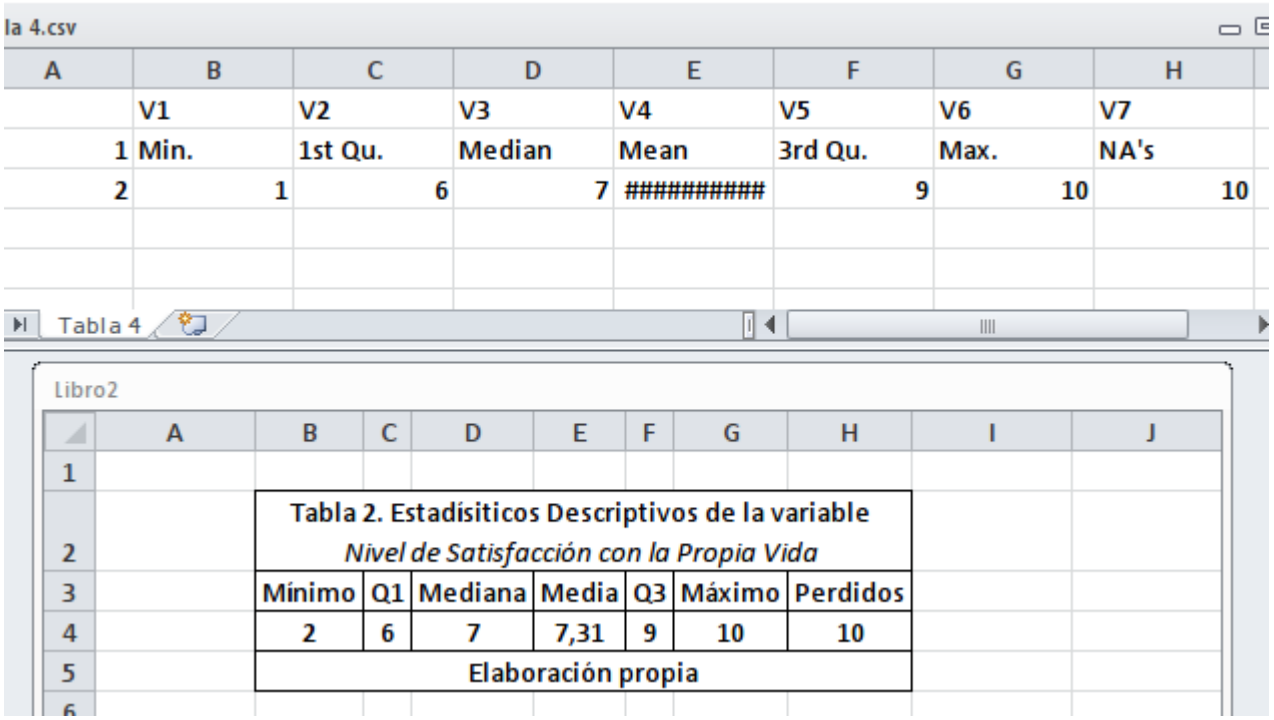
	V1	V2	V3	V4	V5	V6	V7
1	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
2	1	6	7	7.31117397454031	9	10	10

Así, a partir de esta matriz se puede crear un nuevo archivo tipo planilla de cálculo con estos resultados. Que quedará guardado en la carpeta de trabajo con el nombre de “Tabla 4.csv”.

Ejercicio 7.17 (continuación)

```
#Exportar matriz a archivo CSV
write.csv2(descr_sat_vida, file = "Tabla 4.csv")
```

Como se observa en la imagen 7.4, la planilla superior es la exportada desde R, mientras la inferior es la que resulta luego de una edición simple enfocada en editar los encabezados de la tabla y editar el formato de la misma destacando todos los bordes de la tabla y centrando sus valores (entre otras ediciones).



la 4.csv

	A	B	C	D	E	F	G	H
		V1	V2	V3	V4	V5	V6	V7
1		Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
2		1	6	7	##########	9	10	10

Tabla 4

Libro2

	A	B	C	D	E	F	G	H	I	J
1										
2		Tabla 2. Estadísiticos Descriptivos de la variable *Nivel de Satisfacción con la Propia Vida*								
3		Mínimo	Q1	Mediana	Media	Q3	Máximo	Perdidos		
4		2	6	7	7,31	9	10	10		
5		Elaboración propia								
6										

Ahora bien, la función summary no calcula todos los estadísticos que pueden ser de interés. Para construir una tabla completamente personalizada es necesario calcular cada valor que sea de interés y disponerlo en una matriz de datos para así poder guardar una tabla en formato planilla de cálculo.

Como se observa en el ejercicio 7.18, cada valor construido se concatena como un sólo vector (descriptivos_satvida) de valores numéricos, que será una fila de la planilla a construir (la que contendrá los valores de cada estadístico). Adicionalmente, se construye un vector llamado nombres con los encabezados de cada estadístico a escribir en la tabla, que se utilizará como fila de títulos de la planilla de cálculo a exportar.

Ejercicio 7.18

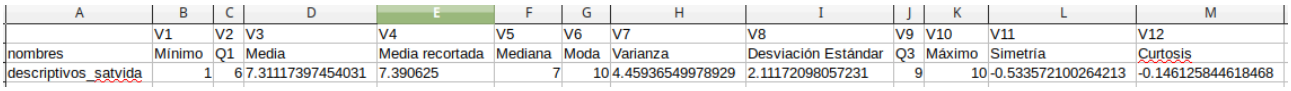
```
#Cálculo simple de estadíticos descriptivos
min <- min(CEP$satisfaccion_vida, na.rm = TRUE)
q1 <- quantile(CEP$satisfaccion_vida, probs = 0.25, na.rm = TRUE)
media <- mean.default(CEP$satisfaccion_vida, na.rm = TRUE)
media_rec <- mean.default(CEP$satisfaccion_vida, trim = 0.025, na.rm = TRUE)
mediana <- median.default(CEP$satisfaccion_vida, na.rm = TRUE)
moda <- mfv(CEP$satisfaccion_vida)
var <- var(CEP$satisfaccion_vida, na.rm = TRUE)
desvest <- sd(CEP$satisfaccion_vida, na.rm = TRUE)
q3 <- quantile(CEP$satisfaccion_vida, probs = 0.75, na.rm = TRUE)
max <- max(CEP$satisfaccion_vida, na.rm = TRUE)
s <- skew(CEP$satisfaccion_vida)
c <- kurtosi(CEP$satisfaccion_vida)

#Valores de estadísticos como vector
descriptivos_satvida <- as.numeric(c(min, q1, media, media_rec, mediana, moda,
                        var, desvest, q3, max, s, c))

#Encabezados de cada estadístico como un vector
nombres <- c("Mínimo", "Q1", "Media", "Media recortada", "Mediana", "Moda",
          "Varianza", "Desviación Estándar", "Q3", "Máximo", "Simetría", "Curtosis")
```

Con esta información ya es posible configurar una planilla de estadísticos descriptivos completamente personalizada. Para ello, como se muestra en la continuación del ejercicio 7.18, se configura una matriz de datos (función as.data.frame) a partir de los vectores de encabezados de los estadísticos (nombres) y los valores de tales coeficientes (descriptivos_satvida), que se almacena como un objeto de nombre descr2. Finalmente, esta matriz de datos se exporta a una planilla de cálculo mediante la función write.csv2.

Ejercicio 7.18 (continuación)

```
descr2 <- as.data.frame(rbind(nombres,descriptivos_satvida))

write.csv2(descr2, file = "Tabla 5.csv")
```

Todo esto resulta en una planilla como la que se observa en la imagen 7.5, editable para la construcción de reportes formales.



	A	B	C	D	E	F	G	H	I	J	K	L	M
		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
	nombres	Mínimo	Q1	Media	Media recortada	Mediana	Moda	Varianza	Desviación Estándar	Q3	Máximo	Simetría	Curtosis
	descriptivos_satvida	1	6	7.31117397454031	7.390625	7	10	4.45936549978929	2.11172098057231	9	10	-0.533572100264213	-0.146125844618468

En base a una edición muy simple en software tipo planilla de cálculo, se puede llegar a darle un formato como el que se observa en la imagen 7.6, útil para su presentación en reportes formales de resultados.

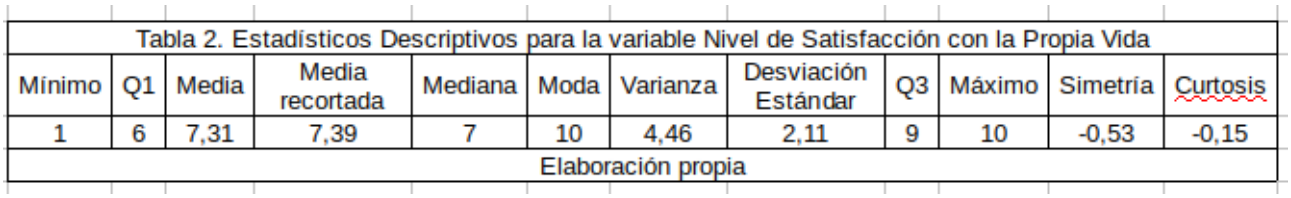


Tabla 2. Estadísticos Descriptivos para la variable Nivel de Satisfacción con la Propia Vida											
Mínimo	Q1	Media	Media recortada	Mediana	Moda	Varianza	Desviación Estándar	Q3	Máximo	Simetría	Curtosis
1	6	7,31	7,39	7	10	4,46	2,11	9	10	-0,53	-0,15
Elaboración propia											

Ahora bien, es importante recalcar cómo se debe realizar la interpretación de una tabla de resultados como la presentada (a continuación se incorpora en un formato adecuado para este documento interactivo).

Estadísticos Descriptivos de la variable Nivel de Satisfacción con la Propia Vida

Estadístico	Valor
Mínimo	1,0
Q1	6,0
Media	7,31
Media recortada	7,39
Mediana	7,0
Moda	10,0

Estadísticos Descriptivos de la variable Nivel de Satisfacción con la Propia Vida

Estadístico	Valor
Varianza	4,46
Desviación Estándar	2,11
Q3	9,0
Máximo	10,0
Simetría	-0,53
Curtosis	-0,15

En la tabla de estadísticos descriptivos presentada destacan diferentes elementos. En primera instancia, los valores mínimo (1) y máximo (10) indican que el rango de las respuestas observadas abarca todas las respuestas posibles de la pregunta del cuestionario.

En segunda instancia, el valor de la media (7,31) es bastante similar al de la mediana, lo que indica que la mitad de las personas de la distribución le asigna la nota 7 a la satisfacción con su propia vida, en la escala de 1 a 10 ya señalada.

En tercera instancia, se puede señalar - respecto a la dispersión de los datos - que la desviación estándar es de 2,11 unidades.

Finalmente, sobre la forma de la distribución se destacan dos elementos: dado que el coeficiente de simetría es negativo (-0,53), se está ante una variable de tipo asimétrica hacia la izquierda de la distribución de casos, en la cual los casos tienden a concentrase hacia la derecha de la media, es decir, en las puntuaciones más altas de la distribución de respuestas; dado que el coeficiente de curtosis es negativo (-0,15), estamos ante una distribución platicúrtica, en la que existe una menor concentración de casos en torno a la media y presenta una forma más “achatada”.

Inferencia estadística univariada: de la estimación puntual al parámetro

En el apartado anterior se revisaron procedimientos para estimar de forma puntual estadísticos univariados provenientes de una muestra probabilística. En el presente apartado se explicará como calcular intervalos de confianza para parámetros a partir de muestras probabilísticas. Concretamente se indicarán los procedimientos para calcular intervalos de confianza para medias y proporciones. En primera instancia se presenta los procedimientos más simples que presenta el lenguaje R, adecuados para diseños de muestras del tipo aleatorio simple; en segundo término, se expondrá como calcular estimadores insesgados a partir del uso de factores de expansión, y cómo estimar intervalos de confianza apropiados ante la utilización de diseños muestrales complejos.

Cálculo de intervalos de confianza para proporcionesEstadísticos descriptivos

Para calcular el intervalo de confianza de una proporción es de utilidad la función exactci del paquete PropCIs (recordar instalarlo vía install.packages("PropCIs") de manera previa).

La información que necesita esta función es la cantidad de casos que coinciden con la condición de interés (la categoría de la variable cuya frecuencia relativa interesa) y la cantidad que representa a la totalidad de casos de la muestra (n muestral). En este caso, a partir de la encuesta CEP se evaluará la hipótesis de si la mayoría de las y los chilenos declara tener una opinión negativa de la situación económica del país.

Como se observa a continuación en el ejercicio 7.19, solicitando una tabla de frecuencias es posible identificar la cantidad de casos probables que tienen una percepción negativa de la situación económica del país (730). Por otra parte, solicitando la cantidad de filas de la base de datos (usando la función nrow) se conoce el n total de casos (1.424). Con ambos valores es posible aplicar la función exactci; un tercer argumento a indicar es el nivel de confianza (conf.level), que se expresa en términos numéricos y proporcionales (para este caso, un 95% de confianza se expresa como 0.95).

Ejercicio 7.19

```
library(PropCIs)

table(CEP$eval_econ_factor)
##
## Positiva   Neutra Negativa
##      471      730      209
nrow(CEP)
## [1] 1424
exactci(x = 730, n = 1424, conf.level = 0.95)
##
##
##
## data:
##
## 95 percent confidence interval:
##  0.4863248 0.5389039
```

Así, el resultado de esta función es el límite superior e inferior del intervalo de confianza, construido a partir de la probabilidad ya indicada (proporción de personas que declaran tener una percepción positiva de la situación económica del país).

Con este resultado es posible concluir que, con un 95% de confianza, no es posible afirmar que la proporción de personas que declaran tener una percepción positiva de la situación económica del país sea más que la mitad de población nacional, debido a que el parámetro poblacional se sitúa entre el 48,6% y 53,9% de los casos.

Cálculo de intervalos de confianza para medias

Una segunda variante para el cálculo de parámetros poblacionales es la estimación del intervalo de confianza de una media. Para calcular el intervalo de confianza de una media es útil la función ci.mean del paquete Publish (recordar instalarlo vía install.packages("Publish") de manera previa). Su uso se detalla en el ejercicio 7.20.

La información que necesita esta función solamente es la variable a utilizar. En el caso

que se observa a continuación, se calcula el intervalo de confianza para la media de la variable nivel de satisfacción con la propia vida.

El resultado es bastante sencillo e indica el valor del estadístico muestral (el valor de la media bajo la etiqueta mean) a la vez que indica el límite superior e inferior del intervalo de confianza para la media poblacional, o parámetro, bajo la etiqueta CI-95%; esto último indica el nivel de confianza utilizado para la construcción del intervalo.

Ejercicio 7.20

```
library(Publish)

ci.mean(CEP$satisfaccion_vida) #Nivel de confianza por defecto.
##  mean CI-95%
##  7.31 [7.20;7.42]
```

Como se observa en los resultados del ejercicio 7.20, al ejecutar el comando con su configuración por defecto, este utiliza un 95% de confianza. Con este valor se puede afirmar con un 95% de confianza que la media poblacional, es decir el promedio de satisfacción que los chilenos declaran tener respecto a su propia vida, se encuentra entre los valores 7,2 y 7,42, en una escala de 1 a 10.
Ahora bien, si al comando básico se le agrega el argumento alpha es posible definir el valor complementario al nivel de confianza, es decir la proporción de error aceptable para la estimación. En la continuación del ejercicio 7.20 se define el valor 0,2 o 20%.

Ejercicio 7.20 (continuación)

```
ci.mean(CEP$satisfaccion_vida, alpha = 0.2) #Definición manual del nivel de
confianza.
##  mean CI-80%
##  7.31 [7.24;7.38]
```

Lo realizado en la continuación del ejercicio 7.20 implica que, observando los resultados, se puede afirmar con un nivel de confianza del 80% que el promedio de satisfacción que los chilenos declaran tener respecto a su propia vida, se encuentra entre los valores 7,24 y 7,38, en una escala de 1 a 10.

Un uso muy frecuente para este tipo de cálculos es evaluar si la diferencia entre una misma media para dos grupos es diferente, de manera estadísticamente significativa. Supondremos que se busca calcular el mismo indicador anterior pero efectuando el cálculo de intervalo de confianza para medias, diferenciando según hombres y mujeres.

Como se observa en el código a continuación (ejercicio 7.21), para indicar a la función cual es la variable de clasificación para construir los grupos, se debe indicar la variable de interés seguida a continuación de una virgulilla (mismo signo usado en la ñ, ~) que la separa de la variable de clasificación. Luego se debe indicar el conjunto de datos de donde provienen tales variables con el argumento data. Como antes, si no se le indica el valor del alpha, la función asume por defecto un cálculo con un 95% de confianza.

Ejercicio 7.21

```
ci.mean(satisfaccion_vida~sexo_factor, data=CEP)
##  sexo_factor mean CI-95%
##  Hombre      7.46 [7.29;7.62]
##  Mujer       7.22 [7.07;7.37]
```

Como se observa en los resultados, las medias muestrales permitirán afirmar que hombres y mujeres presentan una evaluación levemente diferente en relación a la

satisfacción que declaran tener con su propia vida. Específicamente, la media de esta variables es de 7,46 para los hombres y de 7,22 para las mujeres. Sin embargo, al observar el comportamiento de tales mediciones a nivel poblacional, es posible afirmar que las variables no difieren de manera estadísticamente significativa. Para un nivel de confianza del 95%, la media de esta variable para los hombres se sitúa entre los valores 7,29 y 7,62 para los hombres y entre los valores de 7,07 y 7,37 para las mujeres. Los valores indican que los intervalos se intersectan, es decir, existe una elevada probabilidad de que estos parámetros sean muy similares e incluso iguales. Por lo tanto, no es posible afirmar que tales valores sean diferentes, de manera estadísticamente significativa, pues existe una alta posibilidad de que sean iguales.

Coeficientes de expansión para estimación de parámetros poblacionales

Las investigaciones sociales basadas en un diseño muestral probabilístico (aleatorio simple, estratificado, por conglomerado o diseños complejos) deben utilizar un ponderador en la estimación de variables de interés para alcanzar validez sobre la población objetivo. Lo anterior se relaciona con las probabilidades de selección de las distintas unidades de muestreo y da cuenta del número de personas de la población que representa cada individuo de la muestra. Este ponderador es conocido como factor de expansión.

Los factores de expansión buscan incorporar en las estimaciones puntuales surgidas desde la muestra, las características y el peso que tienen esos atributos en la población. De esa forma, se producen estimadores insesgados, es decir, que están centrados en el parámetro población. Así, denominaremos “sesgo de un estimador” a la diferencia entre la esperanza (o valor esperado) del estimador y el valor observado del parámetro a estimar. Decimos que un estimador es “insesgado” o “centrado” cuando su sesgo es nulo: o sea, cuando el valor esperado es igual al valor observado.

Configuración de los datos a utilizar

Para hacer esto más claro, y a la vez indicar cómo se trabaja con coeficientes de expansión en R, lo explicamos con un ejemplo. Específicamente utilizaremos la Encuesta Nacional de Empleo, trimestre móvil Enero-Febrero-Marzo de 2019 publicada por el Instituto Nacional de Estadísticas de Chile. (2019a).

Según lo que se indica en el libro de códigos asociado a la base de datos, utilizaremos las siguientes variables para ejemplificar el uso de coeficientes de expansión.

Variables a utilizar desde la ENE

Nombre de variable	Definición	Valores	Estado en base
fact	Factor de expansión	Peso de cada caso en la población	Creada por INE
edad	Edad de la persona entrevistada	Cantidad de años	Creada por INE

Variables a utilizar desde la ENE

Nombre de variable	Definición	Valores	Estado en base
PET	Población en edad de trabajar	1 representa personas de 15 y más años - 2 representa a personas menores a 15 años	A crear

Como se observa en el siguiente ejemplo (7.22), primero se carga la base de datos con la función read.csv2, para luego recodificar la variable edad en la variable PET36. A esta última variable le agregamos etiquetas mediante la función factor para poder identificar los casos en las salidas de resultados.

Ejercicio 7.22

```
#Cargar ENE y guardar como objeto
ENE <- read.csv2("ENE 2019 02 EFM.csv", sep = ";", dec = ",")

#Recodificacón edad en PET
library(dplyr)
ENE <- mutate(ENE, PET = car::recode(ENE$edad,
                              "0:14=2; else = 1"))
# Etiquetado de categorías
ENE$PET <- factor(ENE$PET, labels = c(">=15", "<15"))
table(ENE$PET)
##
##  >=15   <15
## 86197 20254
```

Como se observa en los resultados del ejercicio 7.22, en la muestra se observa que existen 86.197 personas en edad de trabajar (con una edad igual o mayor a 15 años), mientras 20.254 personas no están en edad de trabajar (menores de 15 años).

Ahora bien, si se observan los resultados oficiales publicados por el INE (2019b), se observa que la cantidad de personas de 15 años o más, no es 86.197, sino 15.286.507 personas. Para lograr llegar a tal cifra se deben aplicar el factor - o coeficiente - de expansión de la base de datos, para luego de ello calcular la cifra de interés.

Utilización del paquete survey para la utilización de coeficientes de expansión

Para aplicar coeficientes de expansión sobre nuestros resultados utilizaremos el paquete survey (Lumley 2019). Como se observa en el ejercicio a continuación, se debe crear un nuevo objeto que almacene la base de datos ponderada. En este caso llamaremos a tal objeto con el nombre ENE_ponderada, que almacenará el resultado de aplicar la función svydesign, con el argumento data indicando la base de datos a ponderar (ENE), con el argumento id indicando la unidad de muestreo que en este caso es el valor 1 pues cada caso se muestrea de forma equiprobable, y finalmente se indica el argumento weights indicando la variable que indica el peso que cada observación de la muestra representa en la población. Como puede verse, cada variable se indica luego de una virgulilla (~).

Ejercicio 7.23

```
#Creación de base de datos ponderada
library(survey)
ENE_ponderada <- svydesign(data = ENE, id=~1, weights = ~fact)

#Características base ponderada
class(ENE_ponderada)
## [1] "survey.design2" "survey.design"
ENE_ponderada
## Independent Sampling design (with replacement)
## svydesign(data = ENE, id = ~1, weights = ~fact)
```

El resultado de aplicar la función class sobre el objeto que almacena la base de datos ponderada (ENE_ponderada) y ejecutar directamente tal objeto, es la información de que se trata de un objeto del tipo survey.design y que se trata de un diseño de muestreo con reemplazo, es decir, cada caso tuvo siempre la misma probabilidad de ser elegido (diseño equiprobable).

Ahora bien, sobre este objeto, podemos seguir utilizando funciones del mismo paquete survey para calcular algunos resultados. Retomando el ejemplo sobre la Población en Edad de Trabajar, con la función svytotal es posible calcular nuevamente las frecuencias de la variable PET. Como se observa en el ejercicio 7.24, esto se efectúa mediante la función svytotal, a la cual se le indica la variable de interés antecedida por una virgulilla (~PET) y el nombre del objeto que corresponde a uno del tipo survey.design.

Ejercicio 7.24

```
svytotal(~PET, ENE_ponderada)
##             total    SE
## PET>=15 15286507 68710
## PET<15   3649759 42750
```

Como se observa en los resultados del ejercicio 7.24, ahora sí logramos calcular las frecuencias reales de ocurrencia de la variable Población en Edad de Trabajar, pues la categoría igual o mayor a 15 años presenta 15.286.507 casos, equivalente a las cifras publicadas de manera oficial por el INE (2019b).

No se profundiza de manera adicional, pero el cálculo de otro tipo de resultados con una base ponderada es bastante similar a lo ya revisado en relación a frecuencias simples. Para mayores detalles revisar la página web mantenida por el equipo desarrollador del paquete survey (Lumley 2019).

Cálculo de intervalos de confianza en diseños muestrales complejos

Para efectuar una correcta utilización de diseños muestrales complejos en análisis de datos sociales orientados hacia la inferencia estadística, es preciso realizar el cálculo de intervalos de confianza utilizando estimadores insesgados. Para ello, debemos ejecutar tal tipo de análisis habiendo ponderado de forma correcta la base de datos mediante la utilización de factores de expansión. Dado que tal procedimiento fue revisado en el apartado anterior, a continuación presentamos de forma específica el modo de calcular intervalos de confianza para medias y proporciones incluido en el paquete survey.

En primer término, en el ejercicio 7.25 se presenta el cálculo del intervalo de confianza

para la proporción que representa la población en edad de trabajar (PET). La función svyciprop precisa definir explícitamente la categoría sobre la cual se desea calcular el intervalo de confianza, luego la base ponderada (ENE_ponderada) en la cual se encuentran los datos ponderados para el cálculo y finalmente el método (method) para realizar la estimación.

Ejercicio 7.25

```
svyciprop(~I(PET==">=15"), ENE_ponderada, method = "li")
##                       2.5% 97.5%
## I(PET == ">=15") 0.807 0.803  0.81
```

Como se observa en el resultado del ejercicio 7.25, el cálculo arroja tres números expresados como proporciones. El primero es 0,807 (u 80,7%), que corresponde a la estimación puntual - basada en las frecuencias ponderadas - de la Población en Edad de Trabajar. Luego se observan los valores 0,803 (80,3%) y 0,81 (81,0%) que marcan, según sus encabezados lo indican los límites del intervalo de confianza calculado con un error tipo 1 (o alfa) del 5%.39.

Este resultado permite concluir que, en base a una estimación realizada con un nivel de confianza del 95%, siendo la estimación puntual un 80,7%, el valor del parámetro que expresa la proporción de las personas en edad de trabajar en la población chilena se encuentra entre el 80,3% y 81,0%.

Esto mismo puede replicarse para frecuencias absolutas. En el ejercicio 7.26 se observa una ejemplificación con los mismos datos que hasta ahora han sido usados: las frecuencias de la variable Población en Edad de Trabajar. En primer término se muestra el cálculo de frecuencias expandidas sobre la variable PET: el cálculo arroja el resultado para las dos categorías de la variable. En segundo lugar se aplica la función confint sobre el resultado de frecuencias absolutas expandidas, indicando de forma explítica el nivel de confianza a utilizar con el argumento level.

Ejercicio 7.26

```
#Frecuencias expandidas para variable edad
svytotal(~PET, ENE_ponderada)
##             total    SE
## PET>=15 15286507 68710
## PET<15   3649759 42750
#IC para frecuencias absolutas
confint(svytotal(~PET, ENE_ponderada), level = 0.95)
##             2.5 %   97.5 %
## PET>=15 15151838 15421176
## PET<15   3565970  3733548
```

Como se observa en el resultado del ejercicio 7.26, la funció svytotal calcula las frecuencias (expandidas) de la variable PET, indicando que la población en edad de trabajar son 15.286.507 personas, mientras que las personas menores a la edad de trabajar son 3.649.759. Luego, la función confint arroja los intervalos de confianza construidos con un 5% de error alfa o tipo 1: en el caso de las cantidad de personas en edad de trabajar, se observa que el parámetro de tal atributo se encuentra entre 15.151.838 personas y 15.421.176 personas; en el caso de la población en una edad menor a la de trabajar (menos que 15 años) el parámetro de tal atributo se encuentra entre los 3.565.970 personas y los 3.733.548 personas.

Finalmente, en el ejericio 7.27 se presenta, en primera instancia, el cálculo de la media

expandida para la variable edad mediante la función svymean. Luego tal cálculo se utiliza para calcular un intervalo de confianza a partir de la estimación puntual, introduciendola como argumento de la función confint e indicando el nivel de confianza a utilizar para la estimación con el argumento level (en este caso, 95% de confianza expresado con la proporción 0.95).

Ejercicio 7.27

```
# Cálculo de media expandida para la variable edad
svymean(~edad, ENE_ponderada)
##       mean     SE
## edad 38.313 0.1169
# Cálculo de intervalo de confianza para la media expandida de variable edad
confint(svymean(~edad, ENE_ponderada), level = 0.95)
##        2.5 %   97.5 %
## edad 38.08348 38.54182
```

Como se observa en los resultados del ejericio 7.27 podemos afirmar que la estimación puntual de la variable edad es 38,31 años. Mientras que el parámetro de tal atributo se encuentra entre los 38,08 y 38,54 años, con un nivel de confianza para tal estimación del 95%.

Construcción de gráficos usando RStudio: funcionalidades básicas y uso del paquete *ggplot2*

Funciones básicas para la construcción de gráficos

De modo general R, en su versión básica, incluye funciones para crear gráficos. Sin embargo, estas herramientas son bastante limitadas en cuanto a las posibilidades de edición que incluyen. Con todo, resultan válidas para un uso de análisis exploratorio. Esto es, un uso enfocado en la visualización de información que permita - dentro del contexto de un proceso de investigación - tomar decisiones para posteriores análisis estadísticos. Luego de explorar estas alternativas se profundizará en el uso de ggplot2, paquete especializado en en el diseño de gráficos que permite una mejor visualización de resultados, sobre todo enfocados en el momento de divulgación de resultados de investigación (Field, Miles, and Field 2012, 116–17).

Gráficos de barras y gráficos circulares

De modo general, los gráficos de barras y circulares40 se recomiendan para visualizar los valores de frecuencias absolutas o relativas de variables nominales u ordinales. Es decir, aquellas variables cuyos valores en la base de datos representan una simple clasificación de datos, sin que sea posible aplicar todas las propiedades aritméticas para tales números (Blalock 1966, 26–37; Ritchey 2008, 42–48).

Como un paso preliminar para la demostración de la construcción de gráficos mediante R configuraremos una nueva variable en la base de datos. Para esto, la variable Sexo (CEP$sexo) será modificada en una nueva variable (sexo_factor) agregando las etiquetas “Hombre” y “Mujer” a los valores numéricos, al configurarla como un vector de tipo factor. Realizamos esta operación para contar con una variable con valores numéricos y con sus respectivas etiquetas de respuesta, con el fin de observar si tal configuración de la variable es soportada por diferentes herramientas para construir gráficos.

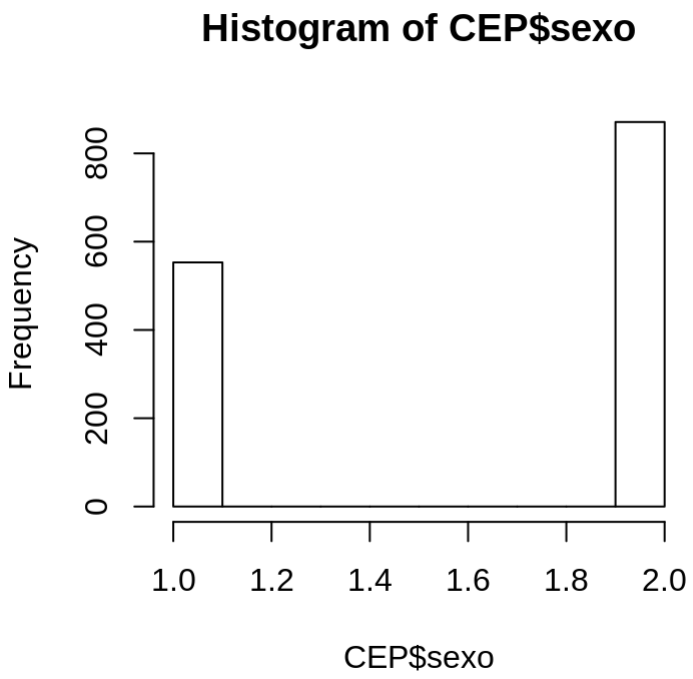
Ejercicio 8.1

```
# Transformar variable sexo a factor (contar con valores y etiquetas)
CEP <- mutate(CEP, sexo_factor = factor(CEP$sexo,
                                        labels = c("Hombre", "Mujer")))
```

Un primer comando básico para la construcción de gráficos en R es hist. Este comando está pensado para construir histogramas. Como se observa a continuación, no resulta apropiado para variables nominales (aunque están codificadas como numéricas); tampoco soporta los formatos character ni factor como formato de entrada de los datos a graficar.

Ejercicio 8.2

```
#Limitaciones de la función hist
hist(CEP$sexo)
```




```
hist(CEP$sexo_chr)
## Error in hist.default(CEP$sexo_chr): 'x' must be numeric
hist(CEP$sexo_factor)
## Error in hist.default(CEP$sexo_factor): 'x' must be numeric
```

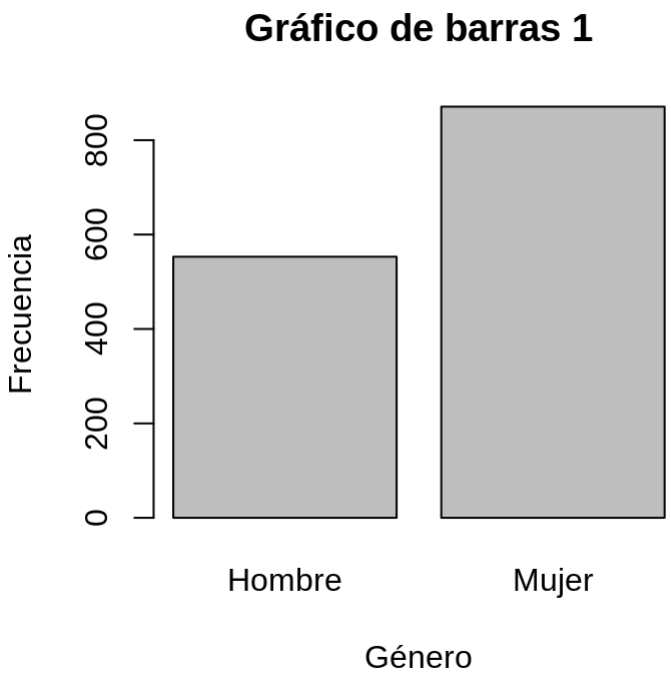
Los resultados de las líneas de código anterior muestran cómo solamente el primer gráfico es construido de manera adecuada, pero asume una continuidad de valores entre las categorías “hombre” y “mujer” (valores 1 y 2, respectivamente), lo que arroja un gráfico poco útil. Por otra parte, al indicar como variable de entrada un vector de tipo character o factor, el comando arroja error.

Por ello se sugiere el uso del comando plot para construir gráficos de barras. De manera general, este tipo de gráficos se usan para variables cualitativas (nominales u ordinales). Las categorías de una variable se sitúan en el eje X, mientras que la frecuencia absoluta o relativa (generalmente porcentajes) se sitúa en el eje Y (Ritchey 2008, 83–85)

A continuación se muestra el uso del comando plot para construir un gráfico de barras. El primer argumento es la variable a graficar. Posteriormente se usan argumentos para agregar títulos: main sirve para agregar un título general; xlab sirve para agregar un título al eje X; ylab sirve para agregar un título al eje Y. Como se observa, este resulta bastante más adecuado para la visualización exploratoria de datos.

Ejercicio 8.3

```
plot(CEP$sexo_factor, main = "Gráfico de barras 1",
    xlab = "Género", ylab = "Frecuencia")
```



Otra forma de presentar variables nominales son los gráficos circulares (o de “torta”)“; se trata de gráficos cuya división proporcional busca representar la distribución de categorías dentro de una variable nominal u ordinal: el área del círculo representa el 100% de los casos de una variable, mientras que el área de cada división del círculo, representa el porcentaje de casos en una categoría específica de esa variable (Ritchey 2008, 80–83). Estos gráficos, a pesar de ser muy utilizados, no resultan tan recomendados pues tienden a distorsionar la percepción de la información presentada: cuando se trabajan variables con muchas opciones de respuesta la mirada tiende a distorsionar el tamaño relativo de las divisiones del círculo. No obstante pueden utilizarse cuando construimos gráficos con variables nominales de pocas categorías (entre dos y cinco), sobre todo para mostrar diferencias entre pocas categorías.
A continuación se muestra una configuración de elementos que servirán para construir el gráfico.

1. Primero se usa la tabla de frecuencias relativas construida en el apartado 7.1.2. Sobre ella se aplica la función round, que permite seleccionar la cantidad de decimales deseados para hacer la aproximación. Mediante la función as.numeric se guardan en un nuevo vector (*porcentajes*) los números que configuran los resultados de tal tabla.
2. Luego se construye un vector de caracteres (*etiquetas*) con las etiquetas de cada resultado.
3. A continuación se realizan dos operaciones sobre tal vector: usando el comando paste cada etiqueta se una con los valores porcentuales que serán cada proporción del gráfico; además, al final de cada valor se incorpora un signo % que quedará expresado en las etiquetas del gráfico. En el siguiente ejemplo se muestra el vector resultante de cada parte de la operación.

Ejercicio 8.4

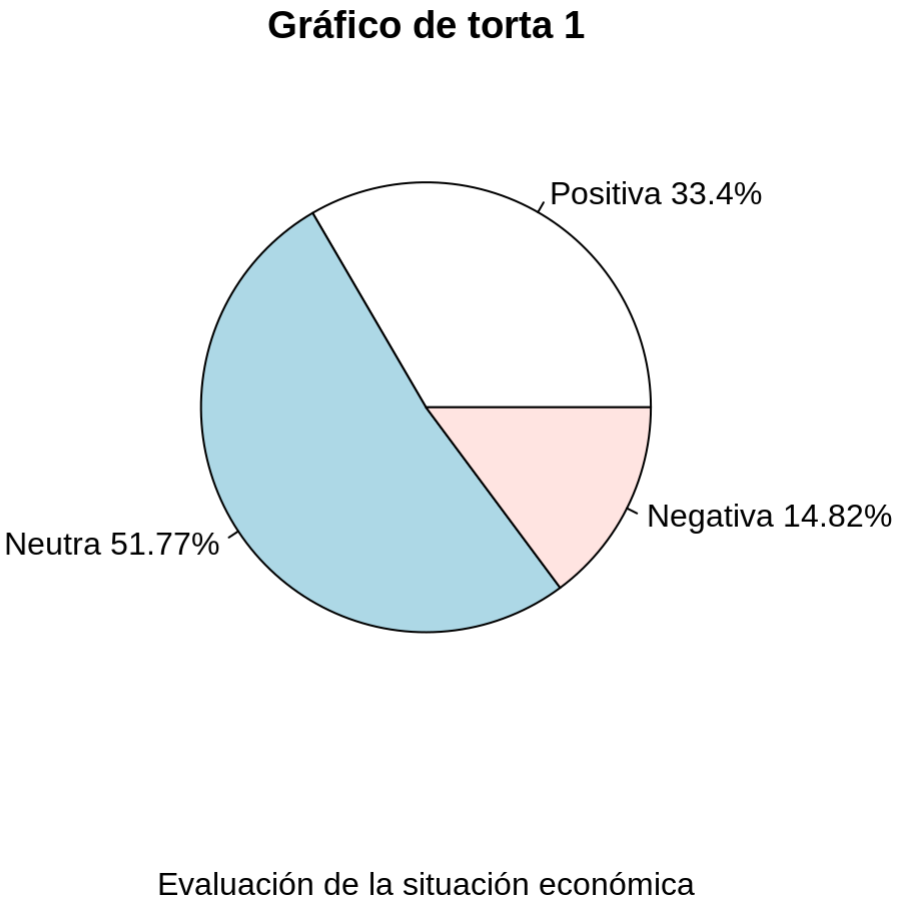
```
#Valores que dividirán el gráfico
porcentajes <- as.numeric(round(((prop.table(table(CEP$eval_econ_factor)))*100),2))
porcentajes
## [1] 33.40 51.77 14.82
#Etiquetas para el gráfico
etiquetas <- c("Positiva", "Neutra", "Negativa")
```

```
etiquetas
## [1] "Positiva" "Neutra"   "Negativa"
etiquetas <- paste(etiquetas, porcentajes)
etiquetas
## [1] "Positiva 33.4"  "Neutra 51.77"   "Negativa 14.82"
etiquetas <- paste(etiquetas, "%", sep = "")
etiquetas
## [1] "Positiva 33.4%"  "Neutra 51.77%"   "Negativa 14.82%"
```

Considerando los dos elementos (porcentajes y etiquetas) se construye el gráfico circular. Como se observa a continuación, este se construye mediante la función pie. El primer argumento son los valores que demarcarán las divisiones del círculo que representa al 100% del área del círculo. Luego se indican los valores que determinarán la construcción de etiquetas. Posteriormente se indica mediante los argumentos main y sub, un título general para el gráfico y una descripción en su borde inferior, respectivamente. Ejecutando tal comando se obtiene el gráfico deseado.

Ejercicio 8.4 (continuación)

```
pie(porcentajes, etiquetas,
    main = "Gráfico de torta 1",
    sub = "Evaluación de la situación económica")
```



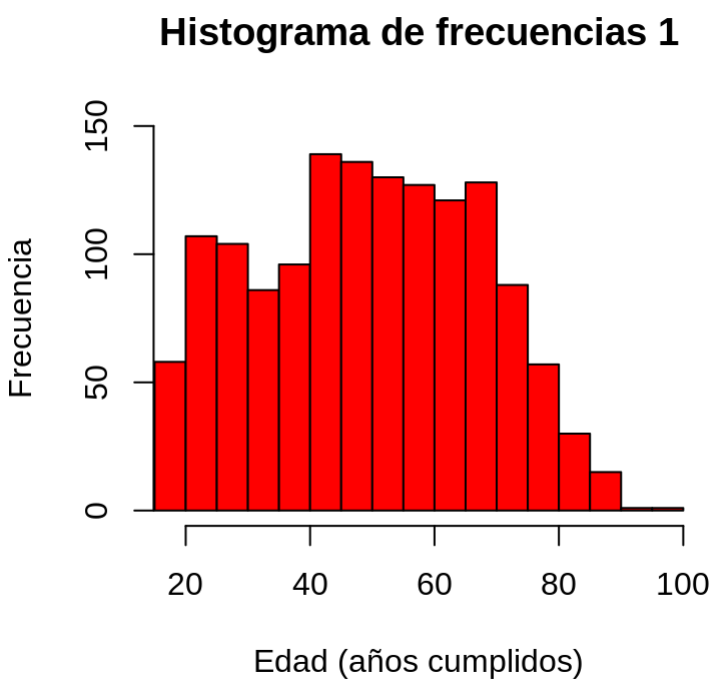
Histogramas

El histograma es una representación gráfica que se utiliza para observar la forma de la distribución de variables cuantitativas (intervalo o razón). De manera similar al gráfico de barras, sobre el eje X se posicionan las puntuaciones de la variable, mientras que la frecuencia (absoluta o relativa) de cada valor se posiciona en el eje Y. Así, se construye un gráfico de columnas verticales; la principal diferencia entre un gráfico de barras y un histograma es que en este último las columnas se tocan entre sí, pues representa a un continuo de valores numéricos (Ritchey 2008, 86–89).

Con el siguiente comando se ilustra la construcción de un histograma usando R. El comando utilizado es hist; sus argumentos son main, xlab e ylab para incorporar un título general y para cada eje. Además podemos indicar el color de relleno de las barras mediante el argumento color, mientras que el color de los bordes de la barra se establece con el argumento border. Adicionalmente, los argumentos xlim e ylim permiten definir manualmente la extensión de los ejes (se indica un vector numérico con los valores mínimo y máximo).

Ejercicio 8.5

```
hist(CEP$edad, main = "Histograma de frecuencias 1",
     xlab = "Edad (años cumplidos)",
     ylab = "Frecuencia",
     col = "red",
     border = "black",
     xlim = c(18, 97),
     ylim = c(0, 150))
```



Existe otro tipo de histograma denominado de “densidad”. Este gráfico no representa las barras de frecuencias de cada alternativa de respuesta de la variable, sino que aplica una *suavización* sobre los valores observados con el objetivo de conseguir una representación de la forma de la distribución que no se vea alterada por la cantidad de barras que incluya el histograma. La principal utilidad de este tipo de gráficos es poder observar la forma de la distribución de una variable, de la manera más precisa posible.

En el ejemplo a continuación se muestra cómo construir un histograma de este tipo: en primer lugar se calcula la información de **densidad** de una variable con la función density asignando su resultado al objeto densidad_edad; luego, mediante la función plot se gráfica esta información indicando título al gráfico y a los ejes de la forma que ya ha sido explicada.

Ejercicio 8.6

```
densidad_edad <- density(CEP$edad)
plot(densidad_edad,
     main = "Histograma de densidad 1",
     xlab = "Edad (años cumplidos)",
     ylab = "Densidad")
```

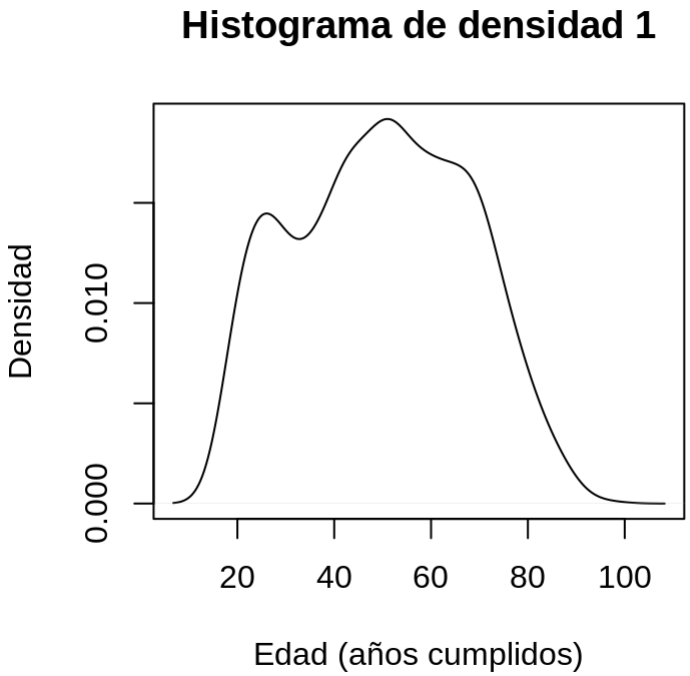
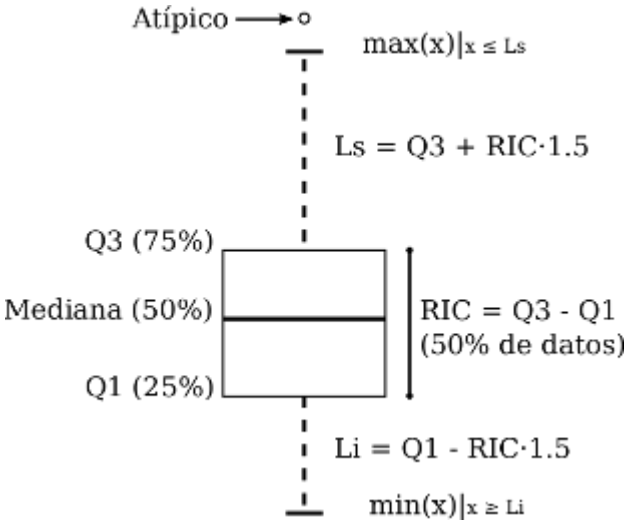


Diagrama de cajas

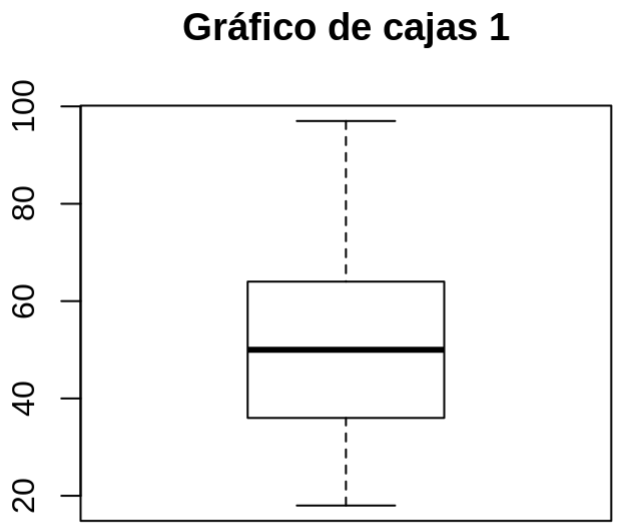
El último gráfico a construir con las funcionalidades básicas de R es el diagrama de cajas (boxplot en inglés). La línea central de este gráfico representa a la mediana, es decir al valor que señala el 50% de la distribución de la variable en estudio. La parte superior e inferior de la caja demarcan al tercer y segundo cuartil respectivamente (esto es, el 75% y 25% de los casos): así, la caja demarca al 50% central de los casos. Por otra parte, los extremos superior e inferior de la línea vertical, demarcan los valores máximo y mínimo de la distribución. Así, la distancia entre el extremo superior o inferior de tal línea y el borde superior o inferior de la caja, demarca la distribución de casos en el cuartil superior (25% de casos con puntuaciones más elevadas) y en el cuartil inferior (25% de casos con menores puntuaciones) de la distribución, respectivamente (Field, Miles, and Field 2012, 151–52).



Para construir un gráfico de cajas el código de R es bastante simple. Basta con ejecutar la función boxplot indicando como primer argumento la variable de interés. A esto se le puede agregar un título (argumento main) y el argumento outline especificado como TRUE, para que si existen casos atípicos estos se grafiquen. A continuación se muestra un comando de este tipo y su resultado.

Ejercicio 8.7

```
boxplot(CEP$edad, main = "Gráfico de cajas 1",
        outline = TRUE)
```



Introducción al uso del paquete *ggplot2*

Nociones generales

El paquete ggplot2 es un paquete de R especializado en la construcción y diseño para la visualización de datos. En este sentido, se trata de un paquete cuyas funcionalidades van más allá de un uso puramente "científico" o *exploratorio* y se orienta a las diferentes dinámicas de divulgación de resultados de procesos de investigación, esto incluye:

- **Divulgación de resultados de procesos de investigación científica para público especializado**. Esto puede referir a contextos académicos (publicaciones en revistas especializadas, libros, etc.) o profesionales (informes de investigación para actividades de consultoría en el ámbito público o privado, por ejemplo).
- **Divulgación de resultados de procesos de investigación científica para público no especializado**. Esto refiere por ejemplo a la difusión de información en contextos como medios de comunicación masivos (televisión, diarios en papel o digital) o redes sociales (twitter, facebook, instagram, etc.).

Se trata, en suma, de un conjunto de funciones altamente especializadas en la construcción de resultados visualmente atractivos para quien leerá la información. Debido a esta especialización, el nivel de configuración y trabajo de codificación para llegar a un resultado deseable, puede ser elevado.
En tal medida, para quien desempeña tareas de investigación social, se abre la tensión entre enfocar su trabajo en la construcción de productos para análisis exploratorio o enfocarse en diseñar visualizaciones de datos estéticamente atractivas para públicos amplios. Vale la pena señalar que el objetivo de este manual es enfocarse en un punto intermedio: manejar los elementos básicos de un paquete como ggplot2 para no depender de que un tercer actor configure nuestros resultados básicos, pero sin profundizar su estudio hasta el punto de que nos aleje de nuestro principal objetivo: construir análisis sociológicamente relevantes y estadísticamente rigurosos.
Es por ello que en este manual centraremos los contenidos en una introducción general al uso de este paquete. Enfatizamos en que no es necesario convertirse en un experto, sino más bien manejar a cabalidad los fundamentos de su funcionamiento. Resultará central en esta dinámica manejar elementos de apoyo para realizar cuestiones más avanzadas.

Gramática del paquete *ggplot2*

El paquete ggplot2 presenta distintas características que lo distinguen:

En primer lugar, los objetos resultantes de la construcción de un gráfico no son una imagen sino un objeto de tipo graphic específico. Esto permite configurar un gráfico como cualquier otro elemento de R, directamente desde la sintaxis,
Debido a lo anterior, la editabilidad de los gráficos construidos es mayor. Definiendo el conjunto de información a visualizar, se pueden configurar diferentes tipos de gráficos.
En tercer lugar, puede señalarse que la estructura de este paquete presenta una gramática específica en relación a sus sintaxis. Como veremos a continuación, su sintaxis guarda directa relación con tres elementos que compondrán la estructura de cualquier visualización de datos: la información (data) a utilizar, la estética (aesthetics) o la definición de los ejes donde se posicionarán los datos a visualizar, y la geometría (geometry) o los elementos visuales que se posicionarán en la gráfica para representar los datos que interesa visualizar.
Resulta importante comprender que en el paquete ggplot2 los gráficos se construyen en base a una serie de capas de información que superpuestas, configuran el resultado final. Andy Field et.al. (2012, 121–36) propone que los gráficos construidos mediante la función ggplot pueden entenderse como una serie de transparencias donde se imprime cierta información; esta información pueden ser números, títulos, barras, líneas, puntos, etc. Luego de definir tales capas de datos, lo que realiza la función ggplot es presentar

el resultado de la superposición de tales capas de información como un elemento unitario.

Para estandarizar una gramática general para la construcción de gráficos mediante capas de información, el paquete se basa en un lenguaje estándar para la construcción de gráficos. Esta nomenclatura se basa en los planteamientos de un famoso libro titulado “The Grammar of Graphics (Statistics and Computing)” (Wilkinson, Wills, and Rope 2005), en el que sus autores definen una base común para la producción de gráficas basadas en información cuantitativa, aplicable a casi cualquier campo de producción de conocimiento.

Elementos que componen un gráfico construido mediante la función *ggplot*.

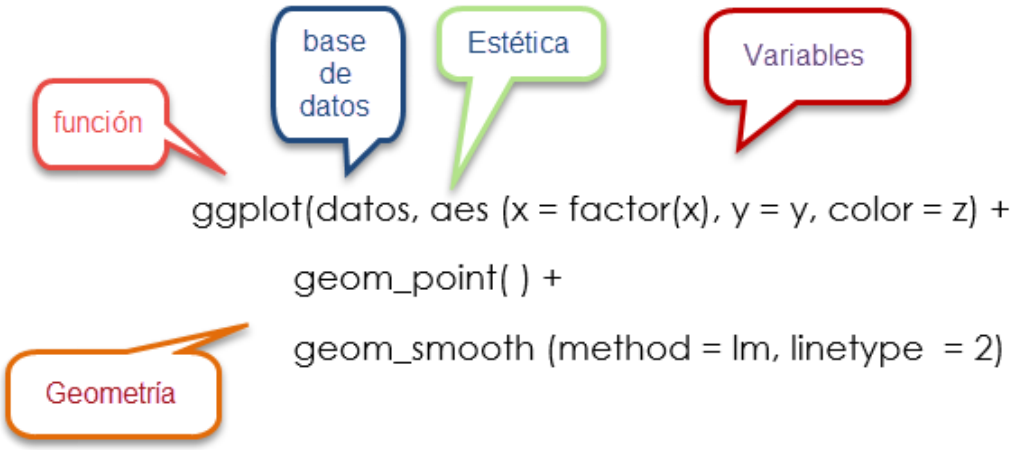
Capa	Descripción
Datos	Conjunto de información que se representará de manera gráfica. En nuestro caso se trata de una o más variables, o una base de datos.
Estética	Escala en la cual se posicionará la información de interés. Refiere al posicionamiento de la información a representar sobre los diferentes ejes y dimensiones del gráfico resultante. Hablamos del posicionamiento de variables en los ejes X e Y como también de la posibilidad de indicar variables que pueden ser posicionadas como color de relleno dentro de los diferentes ejes, como una función de transparencia, etc.
Geometría	Formas, elementos visuales, que se emplearán para representar visualmente la información ya consignada en los *datos* y ubicada en las diferentes posiciones del gráfico mencionadas en la *estética*.[43] Cada especificación de geometría permite visualizar diferentes características de la(s) variable(s) y su distribución.

Es importante entender esta clasificación de elementos que componen un gráfico. Tal diferenciación de elementos comanda la estructuración de la sintaxis específica para crear gráficos mediante la función ggplot.

Como se observa a continuación, en la primera línea de argumentos de esta función (primer paréntesis) se definen los datos a visualizar y luego la estética. Luego, agregando un signo más al final de cada línea, se agregan líneas de comando adicionales que permiten agregar diferentes capas de información al gráfico final, cuyos datos y estética ya han sido definidos.

En la figura a continuación se observa una sintaxis que: i) posiciona una variable X en el eje X, configurándola como factor; que ii) posiciona una variable Y en el eje Y sin transformarla; y que iii) utiliza una tercera variable para el relleno de color de las figuras geométricas finales. En las dos líneas adicionales se establece: en primer lugar, el argumento geom_point() indica que uno de los elementos geométricos a posicionar son puntos, sin configuraciones adicionales; en segundo lugar, se posiciona - mediante el comando geom_smooth(method=lm, linetype=2) una línea de tendencia suavizada (mediante un procedimiento de regresión lineal) que mostrará la tendencia lineal de la nube de puntos.

No olvidar que luego de la primera línea, las funciones de geometría deben agregarse indicando un signo +.



En términos generales ésta es la estructura general para construir gráficos mediante ggplot. Puede que se introduzcan más o menos configuraciones dentro de cada elemento (data, aesthetics, geom) y que se agreguen más o menos capas de objetos geométricos sobre el gráfico a desplegar. No obstante, la lógica básica seguirá siendo siempre la misma.

Construcción de gráficos usando la función *ggplot*: diagramas de barras, histogramas de frecuencia absoluta y relativa.

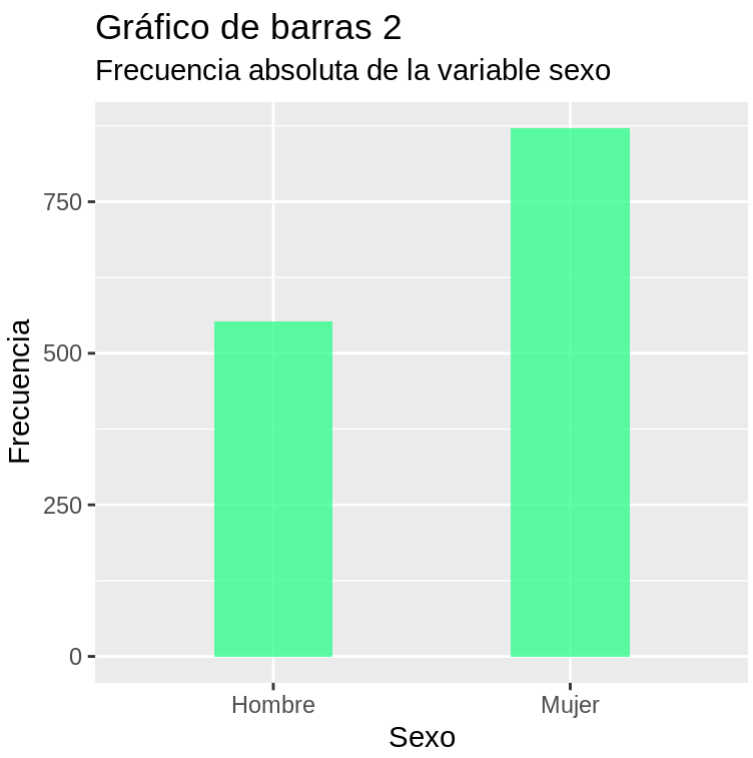
En este apartado se replicarán los gráficos construidos con las funcionalidades básicas de R, pero ahora aplicando la función ggplot. Se comienza por un diagrama de barras.

Como se observa en el código a continuación, este comando sigue la lógica y estructura de sintaxis ya explicada para el paquete ggplot2. En la primera línea de la función se definen los datos (CEP), y con el argumento aes (estética) se define qué variable irá en el eje X (sexo_factor). En este caso, al no definir una variable para el eje Y, el software efectúa un conteo simple de los casos que caen en cada categoría de la variable definida para el eje X. Luego, separado por un signo +, se define la figura geométrica o geometría a utilizar: en este caso se trata de un gráfico de barras (geom_bar); se agrega como argumento de esta subfunción un argumento width que permitirá definir el ancho de las barras, así como un argumento fill para definir el color de relleno de las barras, con la función rgb.45 Finalmente los argumentos siguientes (también especificados luego de un signo +) definen diferentes elementos adicionales: título del eje X (scale_x_discrete), título del eje Y (scale_y_continuous) y título y subtitulo general para el gráfico (labs.)

Ejercicio 8.8

```
library(ggplot2)
#Gráfico de barras 2: sexo en frecuencias absolutas
ggplot(CEP, aes(x = sexo_factor)) +
  geom_bar(width = 0.4,  fill=rgb(0.1,1,0.5,0.7)) +
  scale_x_discrete("Sexo") +     # configuración eje X (etiqueta del eje)
  scale_y_continuous("Frecuencia") +
  labs(title = "Gráfico de barras 2",
```

```
    subtitle = "Frecuencia absoluta de la variable sexo")
```

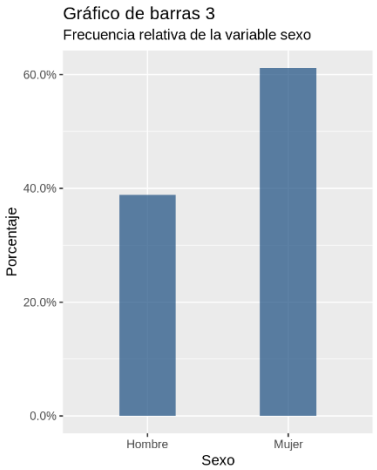


Si bien se trata de un código algo más complejo que la función básica que incluye el software, es posible afirmar que hay una mejora estética notable en comparación con los gráficos más básicos.

Una variante del gráfico anterior es construirlo para frecuencias relativas expresadas en porcentajes, uso muy común para la presentación de datos en contextos de divulgación de resultados.

Ejercicio 8.9

```
ggplot(CEP, aes(x = sexo_factor)) +
  geom_bar(width = 0.4, fill=rgb(0.1,0.3,0.5,0.7), aes(y = (..count..)/sum(..count..))) +
  scale_x_discrete("Sexo") +     # configuración eje X (etiqueta del eje)
  scale_y_continuous("Porcentaje",labels=scales::percent) + #Configuración eje y
  labs(title = "Gráfico de barras 3",
       subtitle = "Frecuencia relativa de la variable sexo")
```

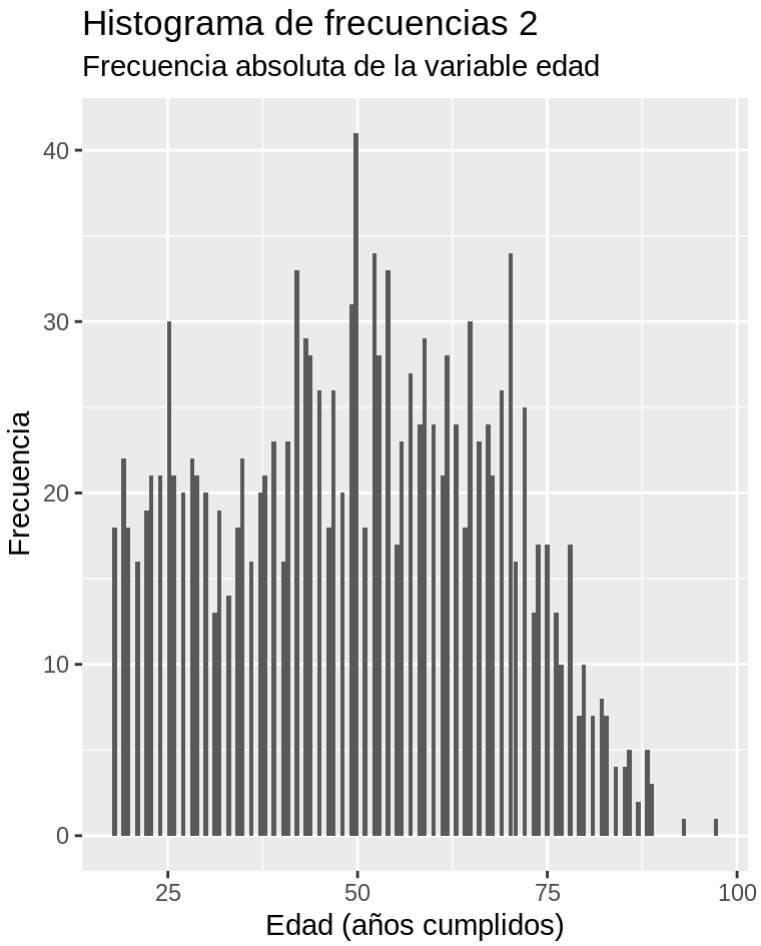


A continuación se indican las modificaciones realizadas al código anterior:

Se modificaron algunos valores del argumento fill de la función geom_bar para alterar el color resultante de las barras.
En la función geom_bar se agregó un argumento aes para editar el tipo de conteo ejecutado en el eje Y. Específicamente se le indica que el conteo hecho en Y es igual al conteo simple, dividido en la suma total de casos.
Luego, se agrega un scale_y_continuous editar los parámetreos del eje vertical. Ello permite agregar el título de porcentaje al eje Y, a la vez que definir que las etiquetas del eje (labels) respondan a una escala de tipo porcentual (label=scales::percent).
El resto de los comandos se mantuvo igual.
El siguiente comando muestra como construir un histograma de frecuencias absolutas. En este caso, luego de los datos a utilizar (CEP), se indica que en el eje X se posicione a la variable edad pero considerándola como variable numérica continua (as.numeric) pues originalmente es una variable de tipo integer, lo que impide posicionarla en el gráfico como variable continua. El resto de los comandos es similar a un gráfico de barras, sólo que ahora se le indica la función geom_histogram para construir el histograma; dentro de esta función se indica un binwidth para ajustar el ancho de la barra. Nótese que la configuración de ambos ejes indica que se trata de variables continuas (scale_x_continuous, scale_y_continuous), de lo contrario no podría configurarse el gráfico.

Ejercicio 8.10

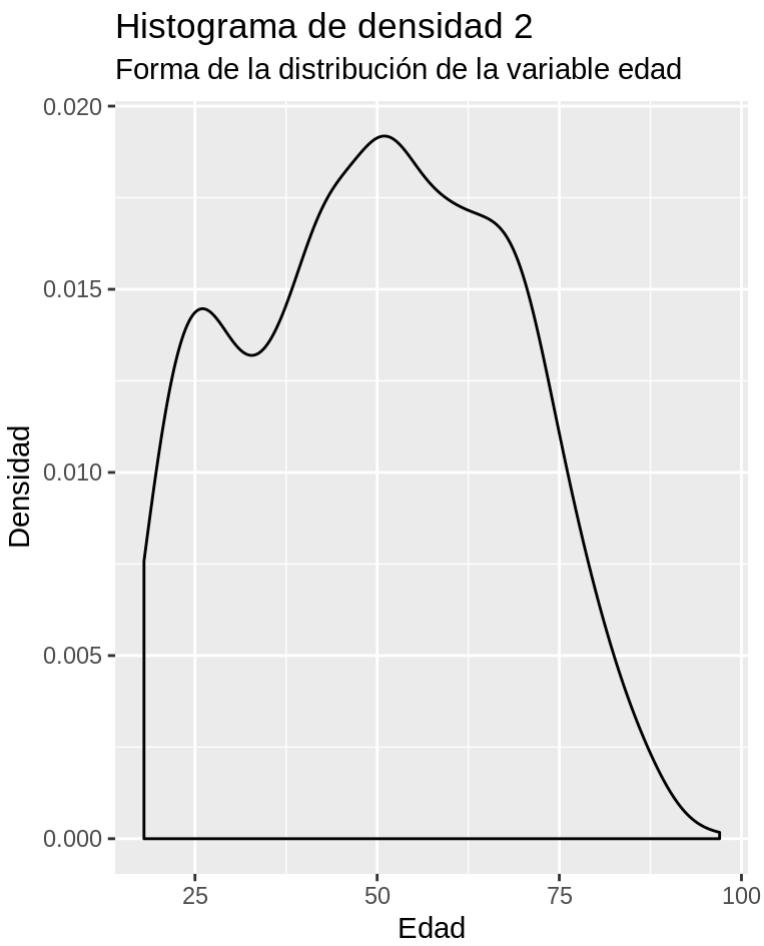
```
ggplot(CEP, aes(x = as.numeric(edad))) +
  geom_histogram(binwidth = 0.6) +
  scale_x_continuous("Edad (años cumplidos)") +
  scale_y_continuous("Frecuencia") +
  labs(title = "Histograma de frecuencias 2",
       subtitle = "Frecuencia absoluta de la variable edad")
```



Como fue señalado en el apartado anterior la manera que resulta más óptima para observar la forma de la distribución de una variable es la construcción de un histograma de densidad. A continuación se muestra la sintaxis que permite construir un histograma de densidad de la variable Edad. La estructura de la sintaxis sigue la misma lógica que los otros gráficos construidos con el paquete ggplot2; en este caso, la especificación de la geometría es el argumento geom_density. El resultado permite observar de manera precisa la forma de la distribución de la variable Edad.

Ejercicio 8.11

```
ggplot(CEP, aes(x = edad)) +
  geom_density() +
  scale_y_continuous("Densidad") +
  scale_x_continuous("Edad") +
  labs(title = "Histograma de densidad 2",
       subtitle = "Forma de la distribución de la variable edad")
```


Histograma de densidad 2
Forma de la distribución de la variable edad
0.020
0.015
0.010
0.005
0.000
Densidad
25
50
75
100
Edad

Bibliografía

Referencias tomadas de: https://bookdown.org/gboccardo/manual-ED-UCH/


UBI
VERITAS
IBI
LIBERTAS
UNIVERSITAS DISTRITALIS
FRANCISCIA JOSEPH CALDAS