

# Uso de RStudio para Estadística Univariada en Ciencias Sociales

Manual de apoyo docente para la asignatura Estadística Descriptiva. Carrera de Sociología,  
Universidad de Chile (primera edición)

*Giorgio Boccardo Bosoni y Felipe Ruiz Bruzzone*

*19 julio, 2018*

## Índice

<b>1. Presentación: del sentido de este material</b>	<b>3</b>
<b>2. ¿Cómo definir qué y cuantos software de análisis estadístico manejar?</b>	<b>4</b>
2.1. SPSS: uno de los programas de análisis estadístico más usado en Ciencias Sociales . . . . .	4
2.2. EXCEL: uno de los olvidados . . . . .	5
2.3. Stata . . . . .	6
2.4. Python . . . . .	7
2.5. R: el temido . . . . .	8
2.6. Ventajas del uso de R . . . . .	9
2.7. Una mirada comparada: limitantes y potencialidades . . . . .	10
<b>3. Instalación de los softwares a utilizar en este manual</b>	<b>12</b>
3.1. Instalación de R . . . . .	12
3.2. Instalación de RStudio . . . . .	13
3.3. Instalación adicional: Java 64 Bits Offline . . . . .	14
<b>4. Uso básico de RStudio</b>	<b>16</b>
4.1. ¿Qué es RStudio?: una interfaz para usar R . . . . .	16
4.2. Carpeta de trabajo y memoria temporal del programa. . . . .	17
4.3. Elementos fundamentales del uso de sintaxis en RStudio . . . . .	18
4.4. Manejo básico de la sintaxis de R: creación de objetos, inclusión de anotaciones y definición de secciones . . . . .	19
4.5. Tipos de objetos en R (vectores) . . . . .	21
4.6. Construcción de una base de datos . . . . .	23
<b>5. Manejo de la biblioteca y gestión de paquetes</b>	<b>25</b>
5.1. Descargar paquetes . . . . .	25
5.2. Cargar paquetes . . . . .	26
5.3. Actualizar la versión básica de R y los paquetes instalados . . . . .	27

<b>6. Gestión de bases de datos</b>	<b>28</b>
6.1. Descarga de una base de datos de interés . . . . .	28
6.2. ¿Cómo abrir bases de datos desde formato SPSS? . . . . .	29
6.3. ¿Cómo abrir bases de datos desde diferentes formatos de Microsoft Excel? . . . . .	31
6.4. Construir una base de datos sólo con variables de interés . . . . .	37
6.5. Exploración de base de datos y recodificación de variables . . . . .	39
<b>7. Estadística univariada con RStudio</b>	<b>47</b>
7.1. Estadística univariada muestral . . . . .	47
7.2. Estadística univariada poblacional . . . . .	55
<b>8. Construcción de gráficos usando RStudio: funcionalidades básicas y uso del paquete <i>ggplot2</i></b>	<b>58</b>
8.1. Funciones básicas para la construcción de gráficos . . . . .	58
8.2. Introducción al uso del paquete <i>ggplot2</i> . . . . .	65
<b>9. Introducción al uso de RMarkdown para la compilación de resultados de RStudio en diferentes formatos</b>	<b>73</b>
9.1. ¿Qué es RMarkdown y para qué sirve? . . . . .	73
9.2. Los diferentes elementos de una sintaxis de RMarkdown . . . . .	76
9.3. Aspectos generales (formato de texto) . . . . .	78
9.4. Aspectos generales (configuración de trozos de código) . . . . .	86
9.5. Presentación de resultados básicos en RMarkdown . . . . .	89
<b>10. Materiales de apoyo para el aprendizaje</b>	<b>95</b>
10.1. Instalación de R y RStudio . . . . .	95
10.2. Uso de RMarkdown . . . . .	95
10.3. Uso de Zotero como gestor de referencias bibliográficas . . . . .	95
10.4. LaTeX y RMarkdown . . . . .	95
<b>Referencias bibliográficas</b>	<b>96</b>

# 1. Presentación: del sentido de este material

Luego de varias ediciones de la asignatura Estadística Descriptiva de la Carrera de Sociología en la Universidad de Chile, hemos considerado necesario ofrecer a los y las estudiantes de la Carrera una sistematización del conocimiento acumulado en el uso de lenguaje de programación R y en estadística descriptiva como herramientas para la investigación sociológica.

El presente documento de apoyo docente tiene un doble propósito.

En primer lugar, convertirse en un material de apoyo para los y las estudiantes del ramo *Estadística Descriptiva*, y facilitar el aprendizaje de herramientas que en un inicio pueden resultar un poco intimidantes, y apoyar el desarrollo de sus trabajos y tareas en la asignatura.

En segundo lugar, proporcionar un documento en español a investigadores e investigadoras en ciencias sociales que se interesen en conocer las herramientas básicas de R y RStudio.

En tanto apoyo docente, el presente documento se organiza en una lógica de aprendizaje incremental. Es decir, se enseñarán herramientas de R y RStudio para el análisis de datos sociales mediante diversos ejemplos, comenzando desde los usos más simple hasta cuestiones de mayor complejidad. Con el objetivo de facilitar el aprendizaje, en la siguiente [carpeta en línea](#) se encuentra toda la documentación para poder replicar los ejemplos propuestos a lo largo de este documento (sintaxis y bases de datos), así como una plantilla básica para el uso de RMarkdown.

Este documento de apoyo docente no es completamente exhaustivo sobre el uso de R y R Studio, pues está delimitado por las especificidades de la asignatura y por el campo disciplinar en que se inscribe. Para usuarios interesado en profundizar sobre Lenguaje R se pueden consultar algunas de las referencias bibliográficas incluidas en este documento.<sup>1</sup>

Agradecemos a Cristóbal Moya por su incansable y desinteresado apoyo para la introducción de R y RStudio a la Carrera de Sociología y los valiosos comentarios realizados por Rocio Salas y Francisca Torres al borrador de este documento.

*Los autores.*<sup>2</sup>

19 de julio de 2018

---

<sup>1</sup>Sugerimos usar el siguiente estilo para citar este documento como referencia bibliográfica: Boccardo, Giorgio y Ruiz, Felipe. *Uso de RStudio para Estadística Univariada en Ciencias Sociales. Manual de apoyo docente para la asignatura Estadística Descriptiva*. Santiago: Departamento de Sociología, Facultad de Ciencias Sociales, Universidad de Chile.

<sup>2</sup>**Giorgio Boccardo** Bosoni. Sociólogo y académico del Departamento de Sociología de la Universidad de Chile. Magíster en Estudios Latinoamericanos. Email de contacto: [gboccardo@u.uchile.cl](mailto:gboccardo@u.uchile.cl) - **Felipe Ruiz Bruzzone**. Licenciado en Sociología y docente de apoyo del Departamento de Sociología de la Universidad de Chile. Estudiante del magíster en Ciencias Sociales de la misma casa de estudios. Email de contacto: [felipe.ruiz@ug.uchile.cl](mailto:felipe.ruiz@ug.uchile.cl).

## 2. ¿Cómo definir qué y cuantos software de análisis estadístico manejar?

Existen múltiples lenguajes de producción y análisis de datos. Para quienes leen este documento nombres como SPSS, Microsoft Excel, Stata o Python quizá no sean desconocidos: varias de estas herramientas computacionales son ampliamente utilizadas en el campo de las Ciencias Sociales.

Si bien presentan características disímiles en cuanto a atributos como su facilidad de uso, generalidad o especificidad de las herramientas de análisis que incorporan y el costo asociado a su utilización, es posible afirmar que su incorporación en los procesos de investigación - profesional o académica - ha contribuido de manera positiva al facilitar el procesamiento computacional de conjuntos extensos de datos y la ejecución de análisis estadísticos que en general resultan de una elevada complejidad ante volúmenes elevados de información (Elousa, 2009).

La decisión de qué software de análisis estadístico utilizar no tiene una respuesta predeterminada: la elección dependerá de las necesidades de la investigación. Esto pues los lenguajes de programación son *herramientas* y el principal criterio para decidir el uso de uno u otro debe efectuarse en función de la particularidad de los objetivos y alcances de la investigación que se busque desarrollar.

Es por eso que aunque puedan existir diferencias sustanciales entre los diferentes software mencionados cada programa puede tener una utilidad específica. En tal sentido, su aplicación debe efectuarse sin perder de vista su cualidad de herramientas con atributos y potencialidades particulares. En todo momento es la investigadora o investigador quien indica instrucciones de análisis a estos programas computacionales a partir del problema que se está investigando, el tipo de información con que se trabaja, los objetivos del estudio, etc. Así, el uso de un programa de análisis estadístico no reemplaza los procesos de decisiones metodológicas como tampoco evidencia de manera automática los errores ni las incoherencias de los análisis: por eso, su uso debe hacerse en relación a las decisiones teóricas y metodológicas realizadas previamente en el contexto de un diseño de investigación particular.

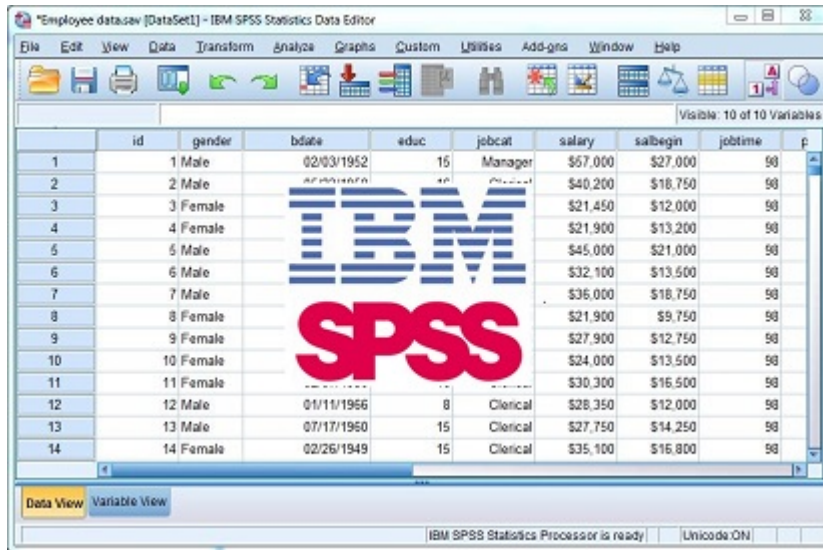
En tal medida la decisión de usar uno u otro software - o una combinación de varios - no es arbitraria. Debe basarse en una observación razonada e informada de las limitantes y potencialidades que cada herramienta ofrece, siempre pensando en los requerimientos específicos que demandan los procesos de investigación. A continuación se ofrece una breve introducción a diferentes software de análisis de datos que pueden ser considerados para el análisis estadístico de datos sociales.

### 2.1. SPSS: uno de los programas de análisis estadístico más usado en Ciencias Sociales

SPSS es una de las herramientas de análisis estadístico más extendidas en el campo de las Ciencias Sociales. De hecho, muchos de los manuales para realizar distintos tipos de análisis estadísticos traen ejemplos e instrucciones obtenidas con este programa. Este software se basa en una estructura del tipo planilla de datos pero incluye una interfaz basada en botones y ventanas que hace muy amigable su uso.

SPSS permite tanto crear bases de datos que contengan la información de un estudio cuantitativo, como realizar distintas operaciones utilizando aquellos datos: procedimientos de estadística descriptiva, inferencial o modelos predictivos, en sus variantes univariada, bivariada y multivariada, entre otras aplicaciones.

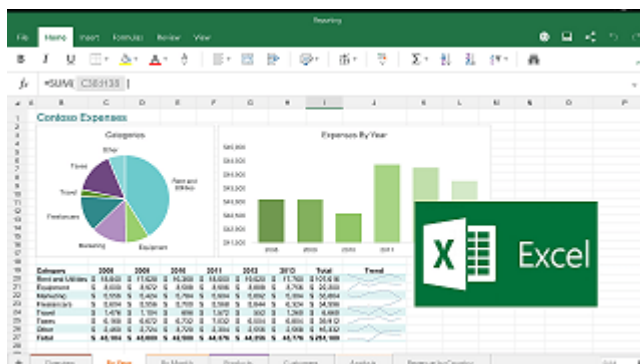
Imagen 1: Interfaz del programa Statistical Package for Social Sciences, de IBM



Este paquete estadístico requiere para usos oficiales o institucionales ser usado con una licencia pagada. De esta forma puede adquirirse el programa básico e incorporar otros paquetes de análisis de datos, según las diferentes herramientas de análisis requeridas. También existen actualizaciones periódicas del programa, las cuales incluyen tanto mejoras en la interfaz de usuario como nuevas aplicaciones para el análisis de datos.

## 2.2. EXCEL: uno de los olvidados

Imagen 2: Interfaz del software Excel, de Microsoft Office



Microsoft Excel es un software que permite crear bases de datos, analizar información y calcular diferentes estadísticos. Este tipo de software se denomina *software de hoja de cálculo* debido a que su interfaz se presenta como una planilla ordenada a partir de filas y columnas donde la unidad básica de almacenamiento de información es la *celda*. Además permite realizar operaciones de estadística descriptiva y multivariada y cuenta con un interfaz más cómoda para la digitación de datos.

En términos generales Excel permite crear tablas que calculan de forma automática los valores de ciertos análisis que el investigador o investigadora especifica, imprimir tablas con diseños cuidados y crear gráficos de manera muy simple. Este programa forma parte de “Office”, un conjunto de productos de Microsoft que combina varios tipos de software para crear documentos de texto, hojas de cálculo y presentaciones y para

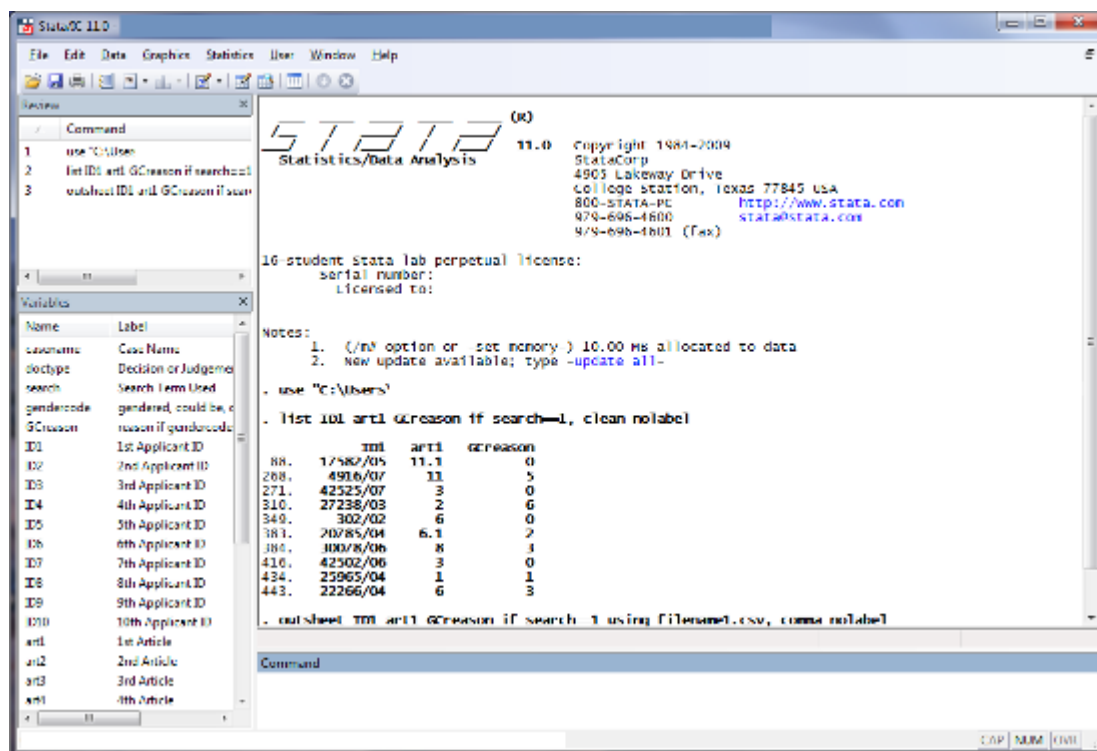
administrar el correo electrónico. También corresponde a un software pagado (requiere comprar una licencia de Microsoft Office), sin embargo es usual contar ya con una licencia básica al comprar un computador que incluye Microsoft Windows como sistema operativo o conseguir una clave de licencia para uso individual o institucional.<sup>3</sup>

Muchas de las bases de datos disponibles, ya sea que provengan de instituciones de gobierno o de otro tipo se encuentran en formato .xls (planilla Excel), lo que implica que como investigadoras e investigadores se vuelve necesario manejar herramientas de este tipo, pues en muchos casos será necesario trabajar con ellas de manera articulada con softwares de análisis estadístico más especializados (nociones claves en este sentido son **extensión de archivo** y **exportar/importar**, cuestiones que serán revisadas más adelante). La experiencia en investigación social académica y profesional indica que por lo general los datos se digitan y trabajan en bruto en hojas de cálculo para posteriormente efectuar los procesamiento estadísticos en otros programas de mayor especialización.

## 2.3. Stata

Stata es un software de uso pagado desarrollado por la compañía StataCorp. De manera similar a SPSS este programa combina un formato de entrada de datos basado en la estructura de planilla de cálculo, a la vez que presenta una amigable interfaz de botones junto con la posibilidad de manejarlo directamente a través de sintaxis.

Imagen 3: interfaz del software Stata



<sup>3</sup>Para quienes deseen acceder a una versión actualizada de este paquete de Microsoft, la Universidad de Chile ha puesto a disposición de sus alumnos una [licencia gratuita](#) para hasta 3 computadores. Por otro lado, en el mundo del *software libre* existen plataformas alternativas que ofrecen de manera gratuita el mismo conjunto de herramientas que Microsoft Office, de manera totalmente compatible con ellas: una de las más conocidas es el *paquete de oficina libre* desarrollado por The Software Foundation, conocido como [LibreOffice](#) (disponible para distintos sistemas operativos). Una opción similar es [Open Office](#), desarrollado por The Apache Software Foundation. La primera alternativa (Libre Office) presenta mayor compatibilidad con las aplicaciones de Microsoft Office, pero sus actualizaciones son menos estables que la segunda (Open Office).

Incorpora una gran cantidad de herramientas para la gestión de bases de datos y análisis estadísticos de diferente nivel de complejidad. Se evidencia un software que pretende incorporar en una sola plataforma todos los procesos de una investigación cuantitativa: construcción, validación y mantención de bases de datos, análisis estadísticos simples y multivariados, construcción de gráficos y resultados para su publicación. Es ampliamente utilizado en el campo de las ciencias económicas.

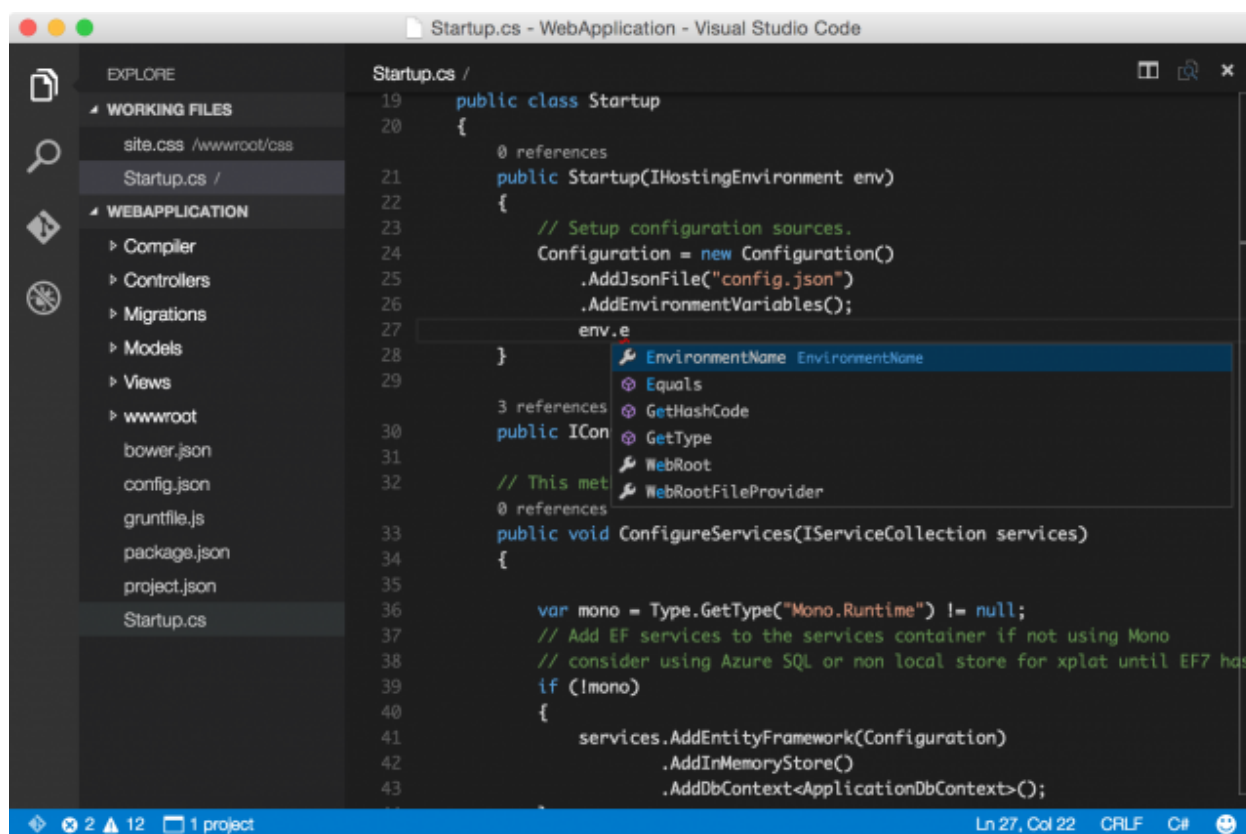
Para su uso oficial o institucional requiere la compra de una licencia pagada. Así se adquiere el programa básico que incluye todas las posibilidades de manejo y análisis de datos mencionadas, sin hacer necesaria la compra de paquetes adicionales. También existen actualizaciones periódicas del programa las cuales incluyen tanto mejoras en la interfaz de usuario como nuevas aplicaciones para el análisis de datos, que aseguran la estabilidad del lenguaje de programación entre diferentes versiones del software.

## 2.4. Python

Es un lenguaje de programación multiparadigma que soporta programación orientada a objetos, programación imperativa y programación funcional. Su principal premisa es desarrollar un lenguaje de programación que sea *simple* y *entendible* por los usuarios.

Al ser un *lenguaje de programación* sus usos son más amplios que los anteriores softwares señalados. Sus características se pueden aplicar al desarrollo de páginas web, sistemas de almacenamiento de información, programación de otros softwares, operaciones matemáticas y estadísticas. En términos generales brinda un lenguaje de sintaxis con estructura que permite una programación clara en escalas pequeñas y grandes.

Imagen 4: interfaz del software Python



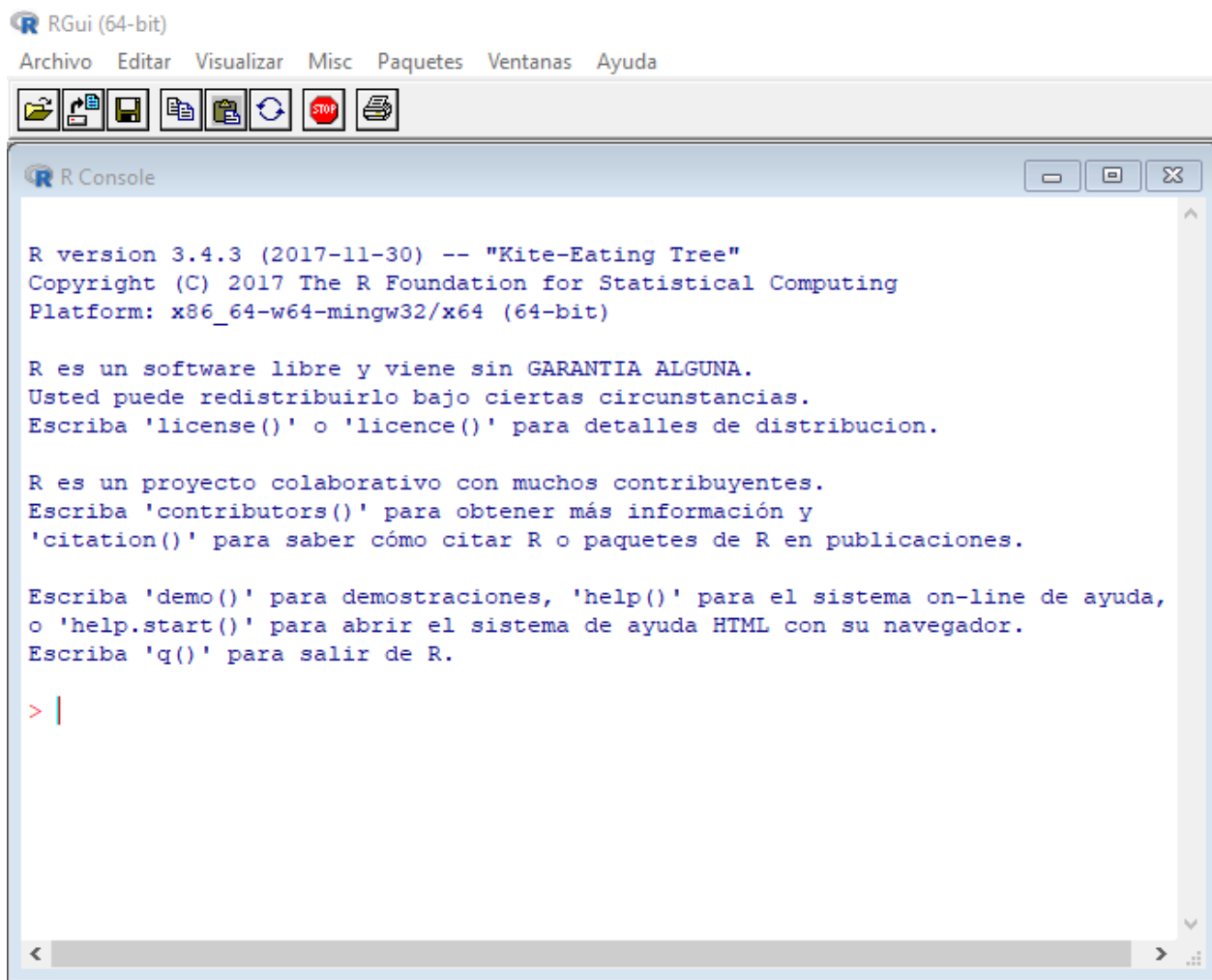
Como se verá, tiene características similares a R: es un programa gratuito de código abierto, que cuenta con una comunidad científica activa que colabora permanentemente en su desarrollo, elementos que permiten que integre múltiples aplicaciones adicionales a su versión básica, para diferentes campos de aplicación.

Contar con un lenguaje de programación intuitivo hace que una de sus principales ventajas frente a R sea la de una curva de aprendizaje más rápida. Por otro lado, presenta una mayor capacidad para trabajar con el procesamiento de datos textuales (*text mining*). Sin embargo, presenta menos desarrollo que R en relación al desarrollo de herramientas de visualización de datos y en paquetes específicos para análisis estadístico multivariado.

## 2.5. R: el temido

Se trata de un software de distribución gratuita (un *freeware*) desarrollado por *The R Foundation for Statistical Computing*. R es un lenguaje de programación utilizado para el análisis de datos cuyo énfasis de uso está en la configuración directa de los análisis de parte del usuario antes que en una interfaz amigable.

Imagen 5: Interfaz del software R en su versión básica



Se trata de un proyecto colaborativo en la medida que los mismos usuarios van desarrollando nuevas aplicaciones que son compartidas gratuitamente en la página oficial del software; así, R está en permanente ampliación: es un proyecto abierto y gratuito.

*“R, en tanto en cuanto software libre, se inscribe dentro del proyecto GNU General Public Licence (Licencia Pública General, GNU). Se trata de una licencia creada por Free Software Foundation*



*(Fundación para el software libre), organización fundada por Richard Matthew Stallman en el año 1985. El principal propósito de la licencia GNU es declarar la libertad del uso, modificación y distribución del software y protegerlo de intentos de privatización que puedan de algún modo restringir su uso (.). Parte de la vasta información disponible sobre R es accesible a través de la página CRAN (Comprehensive R Archive Network; <http://cran.r-project.org/>), sitio oficial de R. Es la página base del proyecto R, desde la cual se puede descargar la última versión del programa (un equipo formado por unas doce personas, R Development Core Team; asumió en 1997 las labores de actualización semestral del código de R), consultar manuales sobre R, obtener ayuda sobre su funcionamiento a través de un sistema de ayuda on line, y, en definitiva, estar al corriente de los movimientos en este entorno de trabajo.” (Elousa, 2009, p. 653)*

La modalidad básica de este programa de análisis estadístico está lejos de presentar una interfaz amigable para el usuario. Como se observa en la imagen anterior la interfaz no dista mucho de la sofisticación de un bloc de notas pues sólo presenta un espacio en blanco donde escribir los comandos, mientras que ninguno de los botones sirve para realizar análisis estadísticos.

Básicamente es un editor de sintaxis, lo que en principio requeriría la habilidad de manejar al dedillo todos y cada uno de los comandos necesarios para ejecutar algún tipo de análisis mediante el particular lenguaje de programación de este *freeware*. Es fundamentalmente esta característica la que inhibe al usuario inicial en su uso aunque presenta atributos que significan ventajas relativas en relación a los otros softwares mencionados en este capítulo.

Es uno de los softwares con una mayor variedad de herramientas de análisis estadístico, no sólo para las ciencias sociales sino también para otras disciplinas.<sup>4</sup> Al ser un programa de contribución libre (código abierto) no requiere la adquisición de una licencia pagada para su uso; existe además una amplia variedad de paquetes descargables de modo gratuito que son desarrollados por usuarios a lo largo de todo el mundo, lo que permite estar al día en cuanto a la exigencia de incorporar nuevas y más sofisticadas herramientas de un modo gratuito y libre. Adicionalmente se trata de un programa “liviano” que gasta poca memoria computacional para ser ejecutado. Todos estos atributos lo configuran como una alternativa bastante interesante para el análisis estadístico de datos sociales.

Si bien la versión básica del programa dista mucho de ser amigable no hay que desesperar pues existen soluciones (gratuitas) para este asunto. Es posible descargar e instalar softwares adicionales que, a partir de la instalación del software base, permitan contar con una máscara que actúe como una interfaz amigable entre el usuario y el entorno computacional que opera con códigos; todos detalles que se verán a lo largo de este documento.

## 2.6. Ventajas del uso de R

A diferencia de otros programas, para ocupar R nos adentraremos en el uso pleno de la modalidad sintaxis; esto es: no emplearemos botones para realizar los análisis, sino que nos comunicaremos con el software de manera directa, a partir de lenguaje (código) computacional, o sintaxis. El uso de la sintaxis (en cualquier software) y el conocimiento de los comandos de R nos permite las siguientes ventajas para nuestros análisis (Elousa, 2009):

1. *Replicabilidad*: Elemento fundamental en la investigación científica y cada vez más en las revistas académicas donde se exigen los archivos de sintaxis para la publicación de resultados. Permite que cualquier persona a quien enviemos nuestros análisis podrá entender cómo fueron construidos y replicarlos de manera exacta.

---

<sup>4</sup>Como lo plantea Paula Elousa, R “está constituido por más de 1.400 paquetes integrados con los que es posible ejecutar simples análisis descriptivos o aplicar los más complejos y novedosos modelos formales. Además, la incorporación a R de interfaces gráficas (...) que crean entornos de trabajo amigables muy similares al entorno del SPSS permiten saltar la barrera de la accesibilidad, y utilizarlo sin ningún tipo de reparo en la docencia. ¿Existe algo mejor? Libre, gratuito, asequible, accesible y siempre a la vanguardia.” (Elousa, 2009, p. 652)

2. *Eficiencia*: En condiciones “reales” de trabajo continuado, el uso de sintaxis representa un incremento exponencial de la eficacia; por ejemplo, para hacer un solo cálculo (como un calcular una media aritmética), en una modalidad de definición de procedimientos de análisis estadístico mediante botones debemos presionar (por ejemplo) al menos cinco botones para llegar al resultado. Esto es tiempo acumulado, y en instancias de manejo estadístico complejo de datos, consume tiempo y esfuerzo. Como contracara, el uso de sintaxis tiende a aminorar la realización de tales tareas, pues puede llegar a tratarse de una sola línea de comandos.
3. *Control*: Permite un control casi total en el trabajo de análisis, pues permite a quienes investigamos ir definiendo detalles que los programas con botones configuran por defecto; esto además permite detectar errores y potencia el trabajo colaborativo, ya que el lenguaje que diferentes investigadores(as) emplearán, es el mismo.

## 2.7. Una mirada comparada: limitantes y potencialidades

Ya se han revisado las principales herramientas computacionales que existen para el análisis estadístico de datos. ¿Es posible sintetizar en una sola evaluación las potencialidades y límites de cada una de estas alternativas? Creemos que sí, y para ellos usaremos cinco criterios: 1) generalidad, 2) costo, 3) facilidad de uso, 4) popularidad y 5) desarrollo y actualización. Las tres primeras son señaladas siguiendo lo planteado por Elousa (2009) mientras que las últimas dos se proponen como criterios propios para considerar:

1. **Generalidad**. Bajo este criterio, sin lugar a dudas R es una de las alternativas más convenientes. Su estructura como plataforma de código abierto permite incorporar de manera gratuita paquetes adicionales a su versión básica que posibilitan el desarrollo de una gran variedad de técnicas de procesamiento de datos y análisis estadístico. Si bien para los análisis que se enseñarán en este manual programas como Microsoft Excel o SPSS cuentan con las herramientas suficientes para análisis más sofisticados resultan limitados por las escasas herramientas que disponen, así como en la posibilidad de realizar análisis personalizados y no vía una configuración de fábrica que como usuario no es posible modificar. Además, su configuración como lenguaje de programación hacen que R sea una herramienta de mayor flexibilidad que las otras alternativas presentadas: permite trabajar con datos cuantitativos y cualitativos así como usar diferentes fuentes de información (por ejemplo, datos existentes en la web en un sentido amplio). Este tipo de características lo distinguen radicalmente de las alternativas de software construidas en torno a la lógica de la planilla de cálculo (como SPSS y Stata).
2. **Costo**. Herramientas como R y Python son las únicas que presentan una modalidad gratuita de distribución y uso, lo que las pone por delante de las otras alternativas presentadas. Como ya ha sido señalado Microsoft Excel, SPSS y Stata requieren la adquisición de una licencia pagada para su uso en computadores de uso personal o institucional, lo que obliga a incurrir en costos de instalación bastante elevados. Así y todo en el caso de SPSS, por ejemplo, la versión básica no viene con todos los paquetes de análisis (el paquete AMOS, para ecuaciones estructurales por ejemplo, se vende por separado) lo que entrapa aún más la utilidad de estos programas en la medida que los requerimientos de análisis sofisticados aumentan.
3. **Facilidad de uso**. Tanto SPSS como Excel y Stata cumplen con el criterio de facilidad de uso. Son softwares amigables, que ponen a disposición del usuario una amplia variedad de herramientas de uso intuitivo lo que ayuda mucho en la introducción al análisis de datos empleando softwares computacionales. Por otra parte, Python y R presentan una interfaz de mayor complejidad al basarse en un uso estrictamente a partir de instrucciones computacionales o sintaxis. Sin embargo, tal dificultad sólo implica una curva de aprendizaje más lenta que se compensa con las ventajas ya señaladas en relación a los criterios de *costo* y *generalidad*.
4. **Popularidad**. Este criterio se vincula con la “extensión del uso” de una herramienta computacional en el campo científico o profesional. Por mucho que se pueda argumentar a favor de la utilización de softwares como el que enseñaremos en este manual, también se debe considerar que hay herramientas computacionales que gozan de mayor popularidad y uso (por ejemplo SPSS, Stata o Microsoft Excel). En tal sentido, una formación integral debe propiciar un uso combinado de estas herramientas, que permita una flexibilidad y adaptación a diferentes contextos profesionales o académicos, que por lo

general no son flexibles en relación a cambiar de manera rápida el software de “cabecera” utilizado para sus procesos de investigación social.

5. **Desarrollo y actualización.** En el caso de R y Python se trata de plataformas que están en continuo desarrollo y actualización, que cuentan con una comunidad científica activa e involucrada en la producción de nuevas herramientas y soluciones para problemas y desafíos de programación. Ello implica que este tipo de herramientas tienen un potencial de desarrollo ilimitado, que no se encuentra sujeto a un uso de “moda” o según criterios netamente económicos, pues un software libre se usa siempre que la comunidad científica decida hacerlo. Si bien pueden existir otras alternativas de softwares altamente especializados en técnicas específicas de análisis de datos (sea de análisis cuantitativo o cualitativo) éstas tienden a quedar desactualizadas una vez pierden popularidad y su negocio deja de ser rentable.

Teniendo en cuenta todos los elementos que ya han sido expuestos es que en la formación estadística de la carrera de sociología de la Universidad de Chile se ha decidido adoptar progresivamente un uso extendido de R. Es por ello que resulta de central importancia poder acercar de manera amigable esta herramienta a todos los estudiantes desde el inicio de su formación estadística. Tal es el sentido del presente manual.

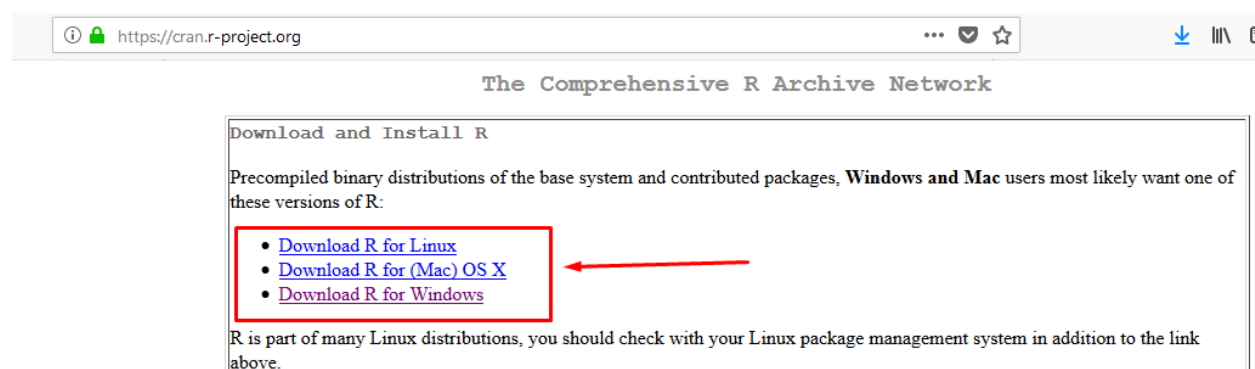
### 3. Instalación de los softwares a utilizar en este manual

Para efectos de este manual las y los lectores deberán saber como instalar dos softwares: R y RStudio. Si bien **siempre se usará el segundo**, es importante entender que éste es sólo una forma de visualizar el primero. Por tanto, siempre se instalará primero R y luego RStudio.

#### 3.1. Instalación de R

Para instalar RStudio se descarga un **installer** - software que permite instalar el programa deseado en nuestros computadores. En la [página oficial de R](#) se encuentra una sección llamada **Download and install R** que permite ingresar a las páginas de descarga de los **installers** para distintos sistemas operativos.

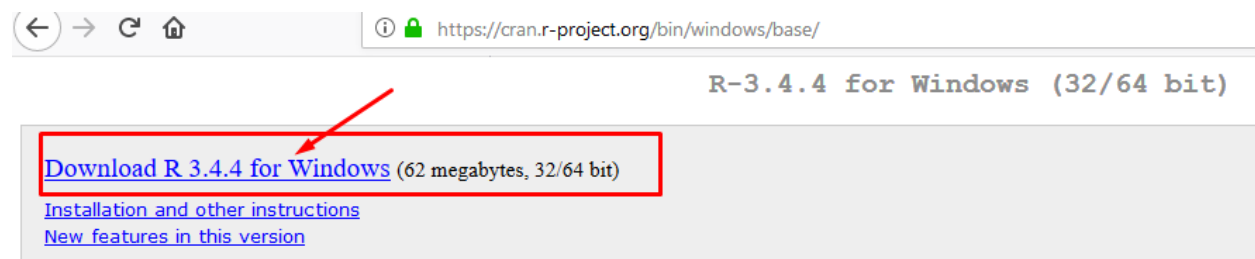
Imagen 6: Página de descarga de installers de R para distintos sistemas operativos



Para efectos prácticos, este manual mostrará como ejecutar la instalación de R para Windows. En el anexo de enlaces útiles puede encontrarse tutoriales que explican como instalar R en sistemas operativos como MacOSX o Linux.

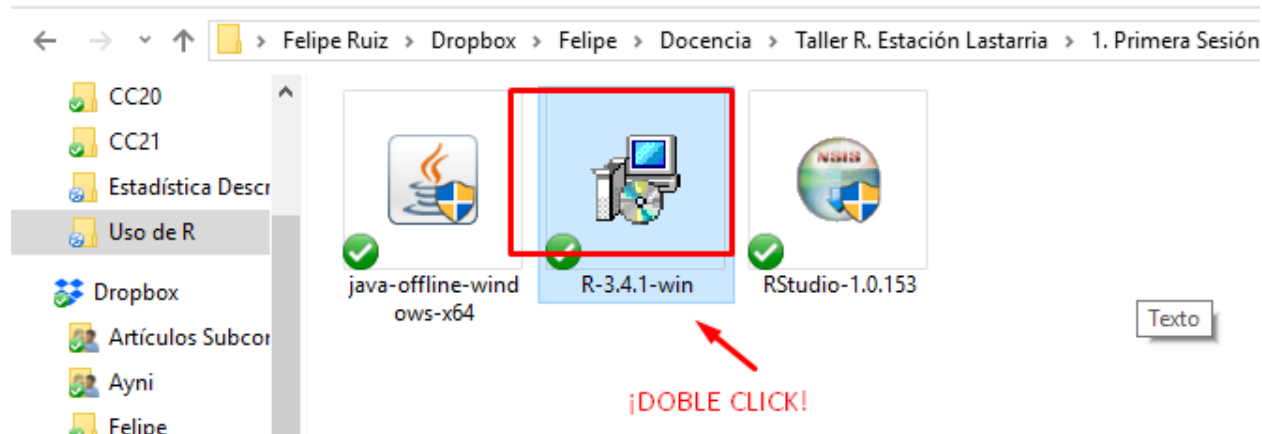
Al seleccionar la opción [Download R for Windows](#) se accederá a la siguiente página, que ofrecerá la opción de descarga de R para Windows. Eso descargará un “installer”, es decir, un archivo con extensión “.exe”.

Imagen 7: Descarga del installer de R para Windows



Luego de descargarlo es preciso tal archivo en la carpeta donde haya sido descargado, para hacer doble click sobre él y ejecutarlo.

Imagen 8: Ejecución del instalador de R para Windows

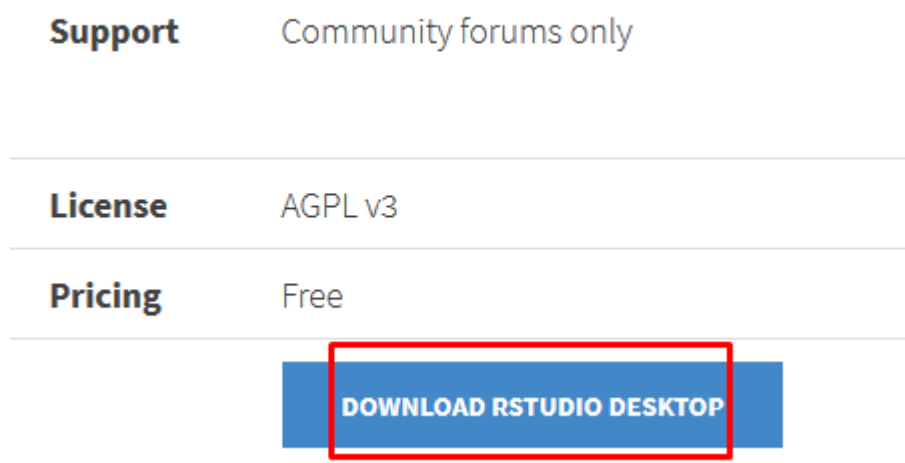


Posterior a eso, deben seguirse las instrucciones de instalación que va ofreciendo el software, hasta completar la instalación. Una vez concluido el proceso, el software habrá quedado instalado en el computador.

### 3.2. Instalación de RStudio

La lógica de instalación de RStudio es similar a lo ya presentado para R. En la [página oficial de RStudio](#) se debe acceder a la pestaña **Products**. Dentro de esa página hay que acceder al apartado de descargar el instalador presionando el botón *Download RStudio Desktop*.

Imagen 9: Página de descarga de RStudio



Dentro de la nueva página que se abrirá, se encuentra la siguiente sección donde se puede descargar el **installer** de RStudio para Windows (también se encuentran para otros sistemas operativos).

Imagen 10: Descarga del instalador de RStudio para Windows Vista, 7, 8 y 10

## RStudio Desktop 1.1.442 — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it [here](#).

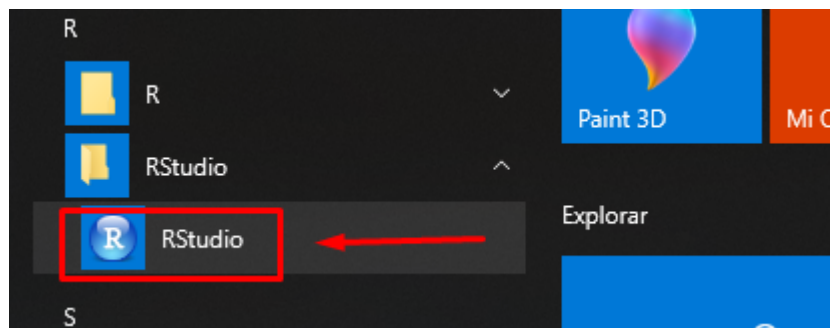
### Installers for Supported Platforms

Installers	Size
RStudio 1.1.442 - Windows Vista/7/8/10	85.8 MB
RStudio 1.1.442 - Mac OS X 10.6+ (64-bit)	74.5 MB
RStudio 1.1.442 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB
RStudio 1.1.442 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB
RStudio 1.1.442 - Ubuntu 16.04+/Debian 9+ (64-bit)	65.1 MB
RStudio 1.1.442 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB
RStudio 1.1.442 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB

Installers para otros  
sistemas operativos

Del mismo modo que para la instalación R en su versión básica es preciso buscar y ejecutar el instalador para RStudio desde la carpeta donde fue descargado. Luego de seguir todas las instrucciones del proceso de instalación el programa habrá quedado instalado correctamente. Es preciso enfatizar que en lo posterior *solamente se ejecutará RStudio, y nunca el software R*. Este último queda instalado y RStudio se encarga de ejecutarlo. En breve, siempre se usará el ícono de RStudio para abrir una nueva sesión del programa.

Imagen 11: Ícono de RStudio en el listado de programas existentes

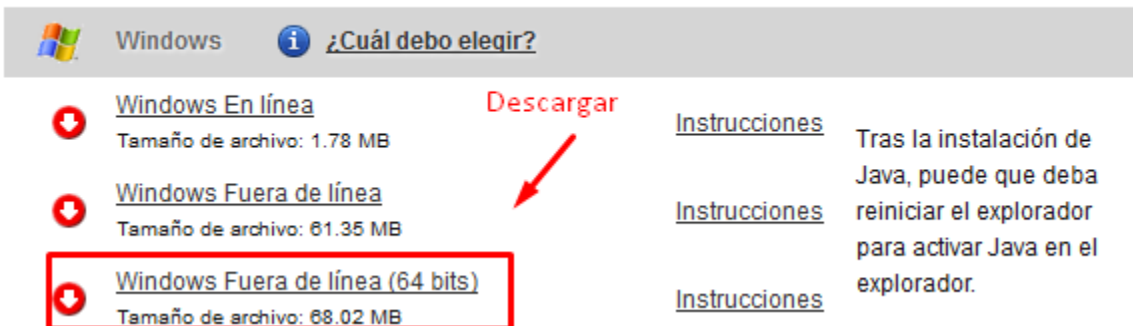


### 3.3. Instalación adicional: Java 64 Bits Offline

Al usar Windows como sistema operativo existe un problema que se produce entre la configuración del sistema operativo del computador y la configuración de los exploradores de internet. Por lo general, el sistema operativo Windows de cualquier computador (de escritorio o portátil tipo notebook) se desenvuelve en una configuración denominada de *64 bits*; por otro lado, los exploradores de internet (Microsoft Edge, Mozilla Firefox, Google Chrome, etc.) funcionan en *32 bits* de manera predeterminada. En general, para acceder a contenido vía internet se usa una aplicación externa, llamada Java, que nos permite reproducir videos y otras aplicaciones web. Ésta se asocia al explorador de internet y por tanto *por defecto se instala, funciona y actualiza en formato 32 bits*.

Distintos paquetes de R usan esta aplicación, por lo que suele producirse incompatibilidades entre un software R instalado de acuerdo al sistema operativo (y por tanto, en modo 64 bits) y una aplicación Java instalada de acuerdo a los requerimientos de los exploradores de internet (32 bits). Esto produce un error en diversos análisis cotidianos. Para prevenirlo, es preciso descargar e instalar una version de Java denominada *Offline*, en su modalidad de 64 bits. Es una versión de Java que viene en 64 bits y que no se actualizará automáticamente de manera constante, previniendo los errores de ejecución mencionados. Tal versión puede descargarse en el [siguiente enlace](#).

**Imagen 12: Descarga del installer para aplicación Java 64 Bits/Offline**



Se deben seguir las mismas indicaciones de instalación que señaladas en los subapartados anteriores de este capítulo: ubicar el instalador en la carpeta de descarga, ejecutarlo y seguir las instrucciones de instalación.

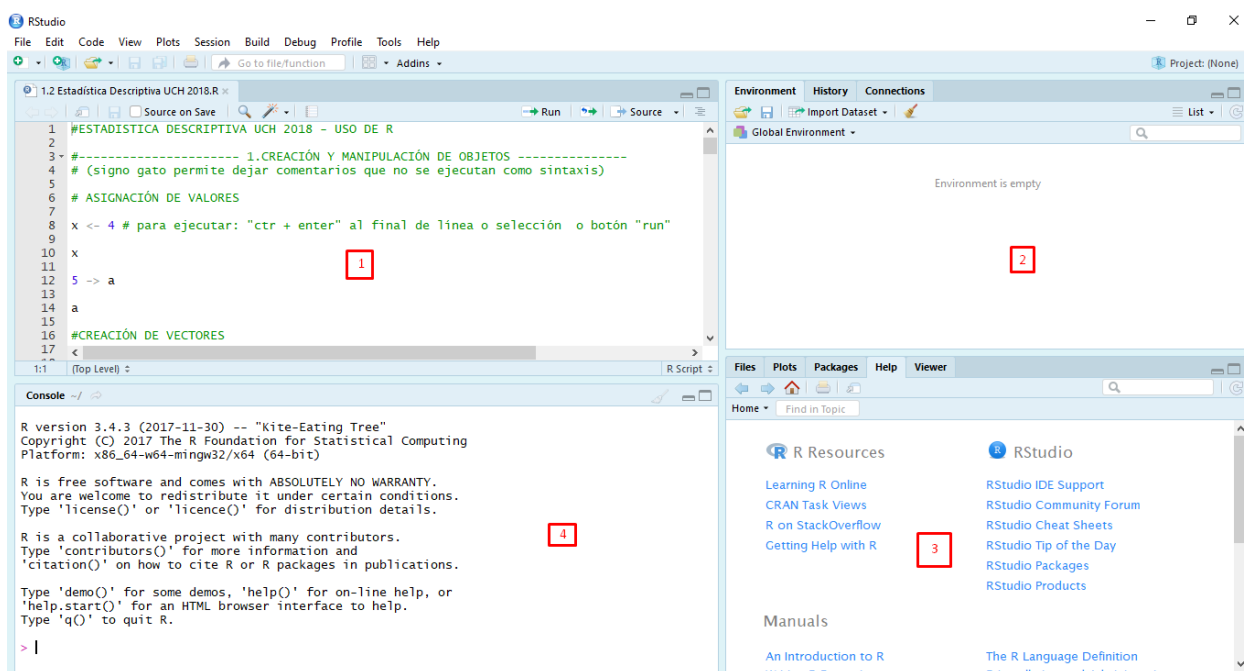
Habiendo realizado estos procedimientos ya es posible introducirse de manera específica en el funcionamiento del programa.

## 4. Uso básico de RStudio

### 4.1. ¿Qué es RStudio?: una interfaz para usar R

En primer lugar para contar con más herramientas de apoyo en el uso de R emplearemos el software RStudio. Este software es una interfase - entre otras como [RCommander](#) - que permiten 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.

Imagen 13: Interfaz de RStudio



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](#), 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 online se accede de manera rápida a manuales de uso de los diferentes paquetes (y sus funciones) instalados.<sup>5</sup>

<sup>5</sup>Al trabajar desde el editor de sintaxis, si se escribe el comando `help()` poniendo entre los paréntesis algún comando o palabra clave (por ejemplo `help(setwd)`) se visualizará la ayuda que brinda el software en relación a la búsqueda especificada.



4. La ventana (4) es un visualizador. Allí el software imprime los resultados de las operaciones ejecutadas desde el editor de sintaxis.

## 4.2. 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 o matrices) se deberán repetir las instrucciones para obtener otra vez el resultado.

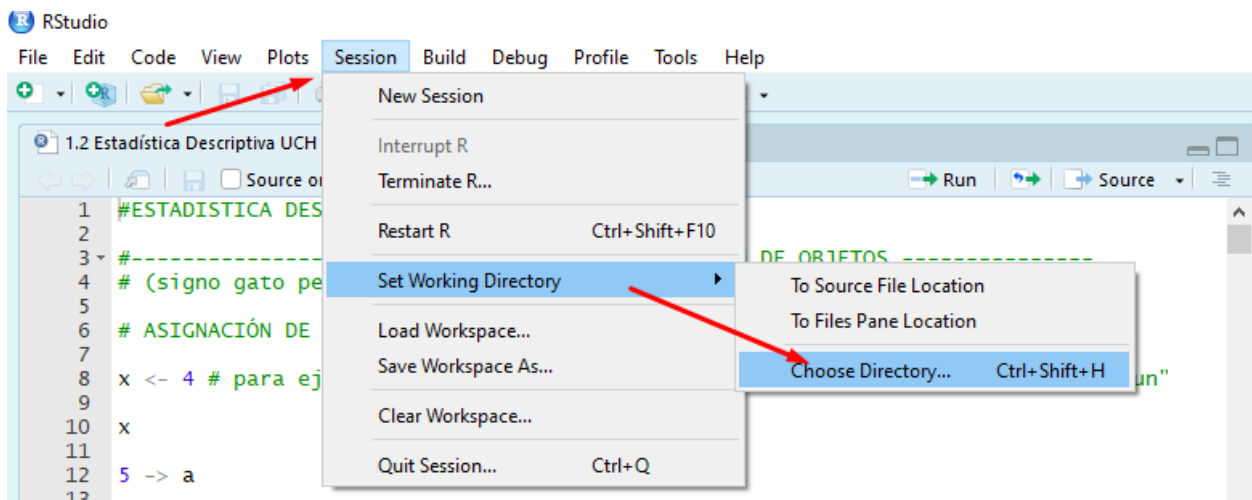
Dado que R trabaja como un espacio temporal y autónomo de trabajo, también es preciso indicarle en qué parte del disco duro del computador están los archivos a manipular. Para evitar el engorroso procedimiento de indicar todo el tiempo las rutas de acceso a los archivos (elementos del tipo: `C:\escritorio\Curso 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:<sup>6</sup>

```
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.

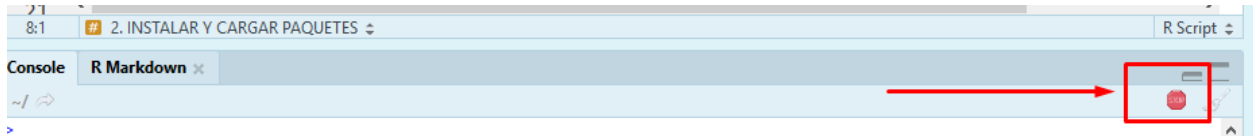
Imagen 14: Secuencia de botones para establecer carpeta de trabajo



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 “Pare” 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.

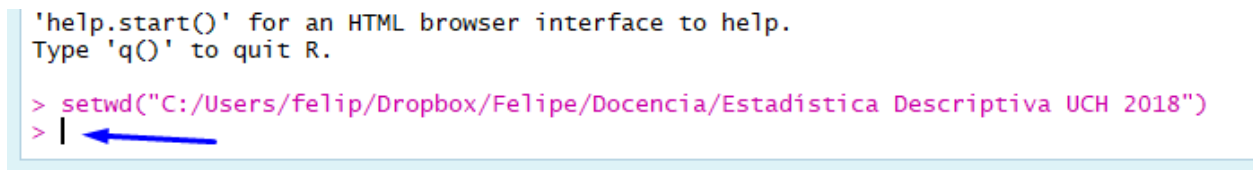
<sup>6</sup>Para efectos de este manual, quienes busquen replicar estos ejemplos deben crear una carpeta en su computador definiéndola como carpeta de trabajo para RStudio. Allí deben descargar y guardar el material a utilizar. Tales elementos se pueden encontrar en la siguiente [carpeta en línea] (<https://github.com/feliperuizbruzzone/Uso-de-R-para-Estadistica-Social>)

Imagen 15: Consola en ejecución de instrucciones



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 se muestra el establecimiento de una carpeta de trabajo cuyo desarrollo ha sido exitosamente ejecutado.

Imagen 16: Consola de R lista para ejecutar instrucciones



#### 4.3. Elementos fundamentales del uso de sintaxis en RStudio

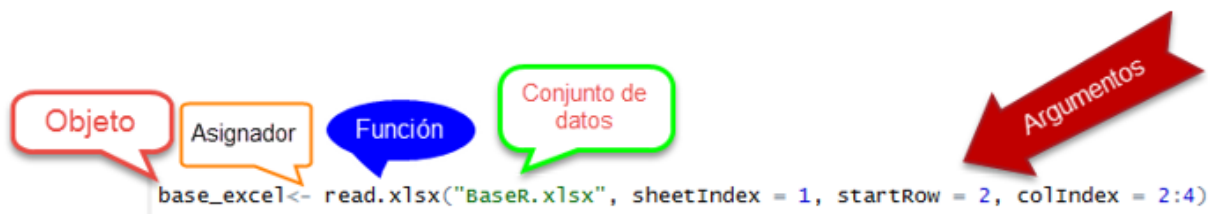
R es un lenguaje de programación. En el caso de la versión básica del software, así como de la interfaz RStudio, el usuario interactuará con el programa mediante *códigos*. La *sintaxis* es un conjunto de *códigos*. Su uso en R es bastante intuitivo y sigue un patrón lógico. De modo general se puede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

```
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:

```
table (base de datos, variable)
```

Imagen 17: Estructura básica de una sintaxis de R

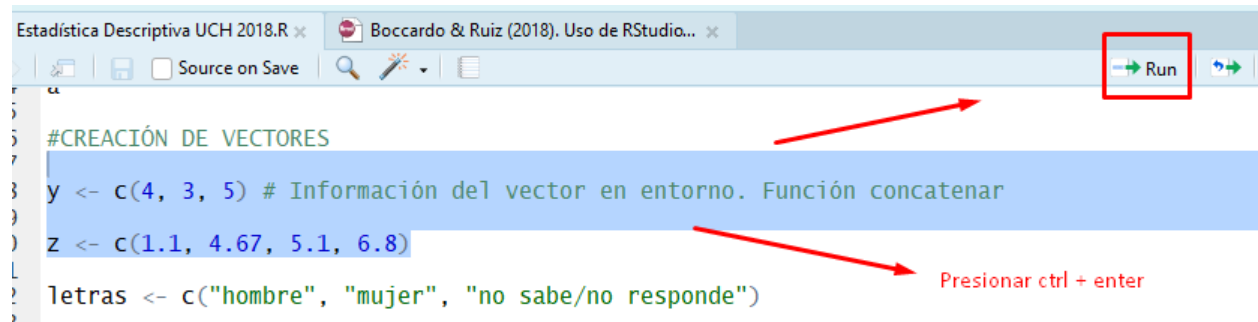


En un sentido más general, como se observa en la imagen, la estructura de una sintaxis es como sigue: a un objeto dado se asigna el resultado de una función, que a su vez se ejecuta sobre un conjunto de datos especificado, con una serie de configuraciones particulares.

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*.

**Imagen 18: Formas de ejecutar una sintaxis en RStudio**



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 aplicar algunas de las dos operaciones ya indicadas.

#### 4.4. 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 ejemplo 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 (*X*) y con el asignador *<-* se le asigna el valor numérico deseado.

##### Ejercicio 1.1

```
#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, se obtendrá como resultado el valor que almacena. 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 sobrescribe la información*.<sup>7</sup> La estructura de la sintaxis computacional es bastante flexible. Sólo a modo de ejemplo, puede ejecutarse 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).

## Ejercicio 1.2

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

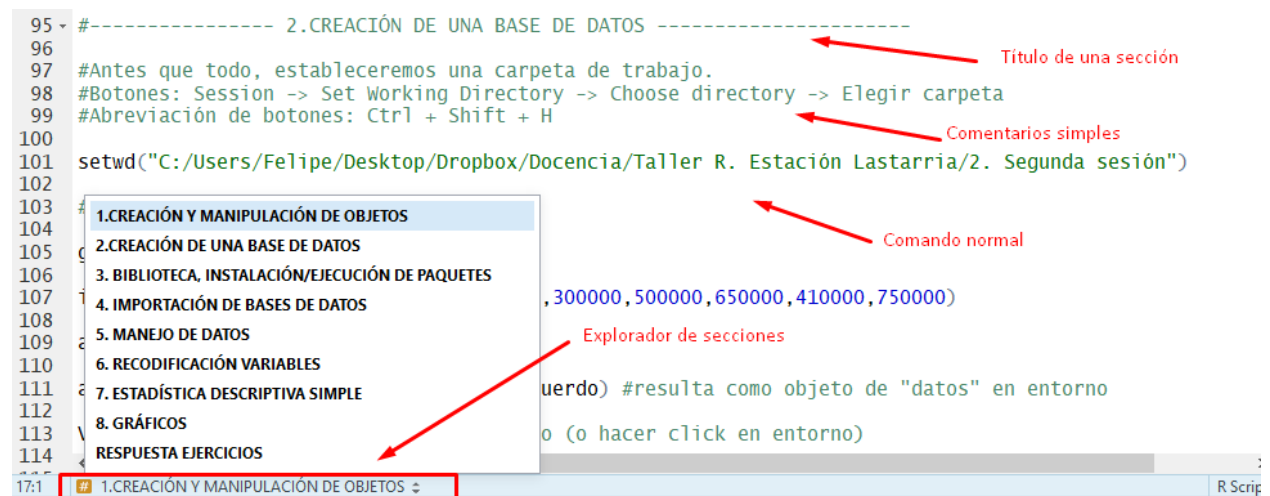
```
## [1] 5
```

Como se ve en el ejemplo, 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 mayoría de los casos, 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 # seguido de una serie de guiones 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.

Imagen 19: Explorador de secciones de una sintaxis de RStudio



<sup>7</sup>Se puede ver que R funciona mediante la creación de objetos que representan un dato o un conjunto de datos. En este caso el valor es simple (un valor numérico) pero también puede tratarse de un objeto que almacene datos de una variable que refiera a categorías sociales tales como Grupo Socioeconómico o Género, de un conjunto amplio de casos.

## 4.5. 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<sup>8</sup>. 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.

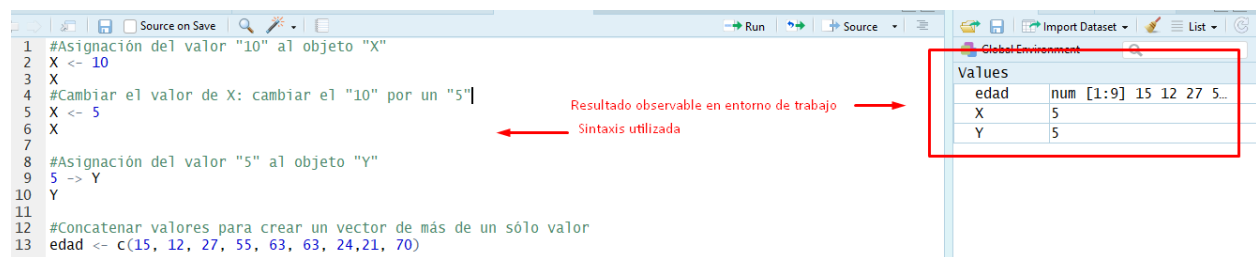
Hasta ahora sólo se ha explicado cómo asignar un valor singular a un objeto de R. Para ingresar más de un valor en un vector se deben indicar los elementos a almacenar entre paréntesis y separados por comas, anteceditos de la función *concatenar*, que se ejecuta anteponiendo una *c* al conjunto de objetos a agrupar.<sup>9</sup> 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 2.1:

```
#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).

Imagen 20: Entorno de trabajo de RStudio con objetos



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] ).<sup>10</sup>

<sup>8</sup>En el ámbito de la informática, un vector (también conocido como matriz) es una zona de almacenamiento contiguo que alberga elementos de un mismo tipo. Puede entenderse como una serie de elementos ordenados en filas o columnas. El vector ofrece una estructura que facilita el acceso a los datos para su manejo computacional. Mayores detalles de esta definición en [Definición de vector - Qué es, Significado y Concepto](#).

<sup>9</sup>Para nombrar a los objetos es importante saber que estos deben empezar con una letra, aunque puede contener dígitos y puntos (.). Además, se debe tener en cuenta que R diferencia entre mayúsculas y minúsculas.

<sup>10</sup>También se puede ejecutar el comando *attributes(edad)* para conocer tal información. En el mismo sentido, el comando *class(edad)* sirve para inspeccionar qué tipo de variable estamos considerando.

Ahora bien, se ha construido un objeto que puede ser considerado como una variable - resultante por ejemplo de haber aplicado una breve encuesta a los estudiantes de una clase. Por ello, usando el comando *table (edad)* servirá para construir una tabla de frecuencias de los valores; el número debajo del caso indica la frecuencia de ocurrencia de cada valor.

## Ejercicio 2.2

```
table(edad)
```

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

Ya ha sido señalado que 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 naturales (incluye decimales y negativos) y un tercero con caracteres alfabéticos.

## Ejercicio 3

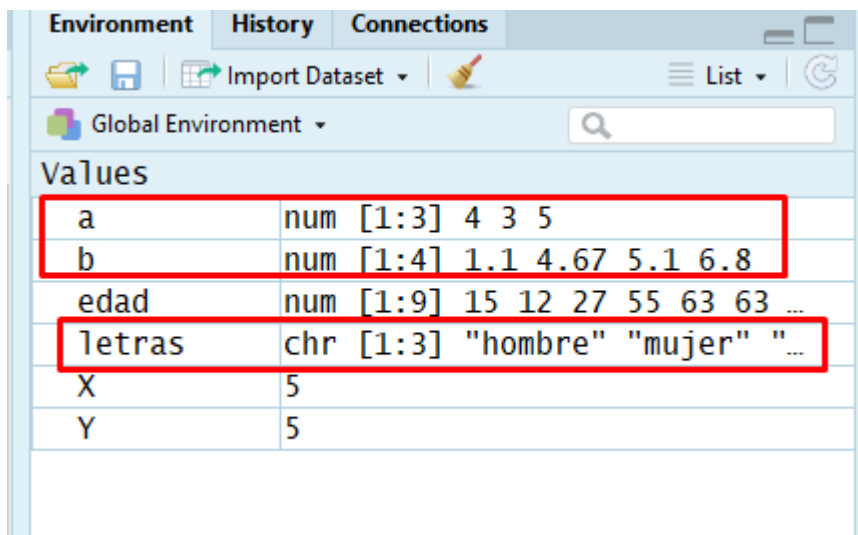
```
#Vector numérico (función concatenar)
a <- c(4, 3, 5)

#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.

Imagen 21: Información de objetos en entorno de trabajo de RStudio



Variable	Valor
a	num [1:3] 4 3 5
b	num [1:4] 1.1 4.67 5.1 6.8
edad	num [1:9] 15 12 27 55 63 63 ...
letras	chr [1:3] "hombre" "mujer" "..."
X	5
Y	5

## 4.6. Construcción de una base de datos

A continuación se construirá una primera base de datos a partir de tres variables ficticias, que se podría ser utilizada para posteriores análisis. 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 0, que representan las categorías de respuesta “hombre” y “mujer”.
2. **Ingreso.** Variable de razón con valores ficticios 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 4.1

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

```
genero <- c(1,1,0,1,0,0,0,1,0)
```

```
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).

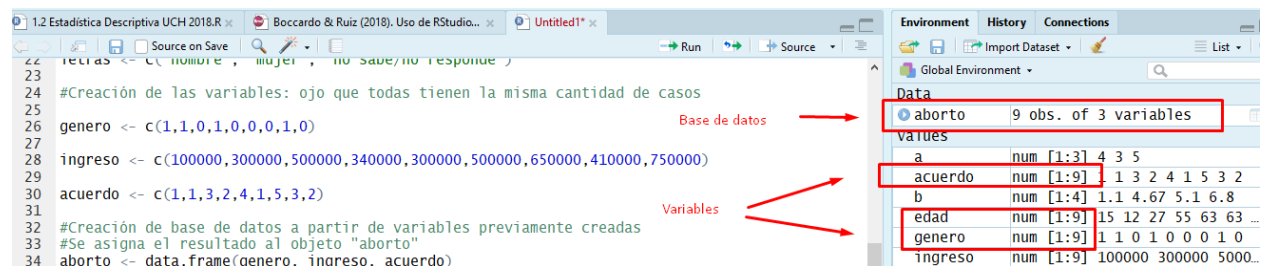
### Ejercicio 4.2

*#Creación de base de datos a partir de variables previamente creadas*

*#Se asigna el resultado al objeto "aborto"*

```
aborto <- data.frame(genero, ingreso, acuerdo)
```

Imagen 22: Entorno de trabajo con variables y base de datos creadas

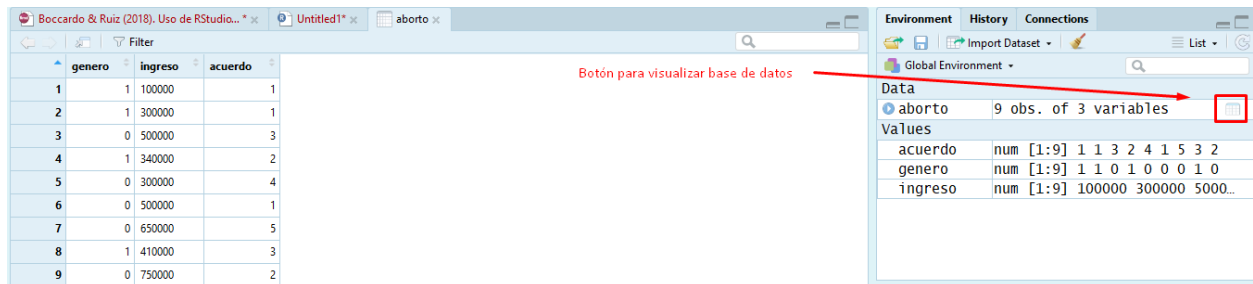


Posteriormente se puede visualizar la base de datos de dos maneras: utilizando el comando `View` o presionando un botón que se encuentra al lado del objeto base de datos en el entorno de trabajo del software. En el siguiente código se explicita la primera forma y en la imagen posterior la segunda.

### Ejercicio 4.3

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

Imagen 23: Uso de botón para visualizar base de datos



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 4.4

```
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*.

Imagen 24: Carpeta de trabajo con base de datos guardada

 aborto	14-04-2018 0:10	Archivo RDATA	1 KB
 apa	09-04-2018 11:02	CSL Citation Style	25 KB
 base_variables	13-04-2018 18:02	Archivo PNG	46 KB

Este archivo es la base de datos recién construida pero almacenada como archivo de formato R en el disco duro. Este archivo puede usarse en 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.

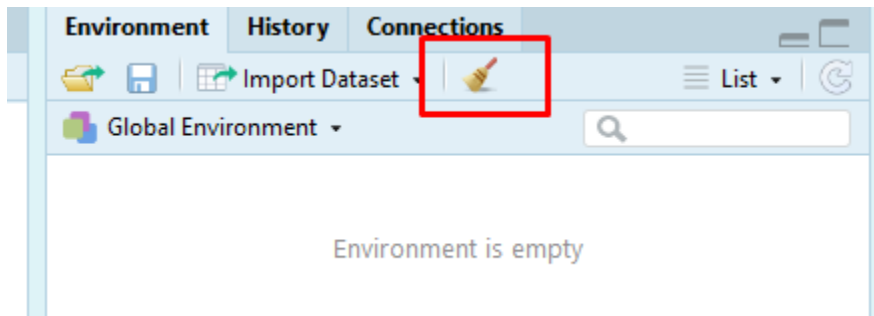
Nuevamente, para limpiar el entorno de trabajo existen dos maneras: la primera es vía sintaxis, con dos comandos específicos que se detallan a continuación; el primero permite eliminar elementos específicos y el segundo vaciar totalmente el entorno de trabajo; la segunda forma es utilizando un botón 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 4.5

```
#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())
```

Imagen 25: Uso de botón para limpiar entorno de trabajo



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.

## 5. 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.

### 5.1. 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. 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.

Imagen 26: Consola indicando instalación exitosa del paquete readxl

```
Console R Markdown x
~/
> install.packages("readxl")
Installing package into 'C:/Users/felip/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/readxl_1.0.0.zip'
Content type 'application/zip' length 1458126 bytes (1.4 MB)
downloaded 1.4 MB

package 'readxl' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\felip\AppData\Local\Temp\RtmpwRLXfn\downloaded_packages
> |
```

Este tipo de mensajes no constituyen errores.

Mensaje que indica que el paquete se revisó e instaló de manera exitosa

Software listo para recibir más instrucciones.

## 5.2. 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.<sup>11</sup> 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”.

<sup>11</sup>No es necesario ejecutar el comando `install.packages` cada vez que se requiera usar un paquete adicional a la versión básica de R. Basta con hacerlo una sola vez pues los paquetes se descargan al disco duro del computador, quedando almacenados allí. Lo que se precisa es **cargarlos** en la sesión temporal de R cada vez que se abra una sesión nueva donde no hayan sido cargados previamente.

Imagen 27: Consola indicando mensajes de alerta al cargar un paquete



```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/readxl_1.0.0.zip'
Content type 'application/zip' length 1458126 bytes (1.4 MB)
downloaded 1.4 MB

package 'readxl' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\felip\AppData\Local\Temp\RtmpwRLXfn\downloaded_packages
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.4.4
> |
```

Mensaje informativo  
No es un error

### 5.3. 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()
```

## 6. 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.

Así, 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). 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. Desde este sitio en línea el usuario podrá descargar las bases de datos históricas de esta encuesta, junto con sus manuales de uso metodológico.

En este caso interesa trabajar con la versión más reciente de tal base de datos. Al momento de la redacción de este tutorial la versión más actualizada disponible fue aquella correspondiente al período Septiembre-Octubre de 2017, denominada *Estudio Nacional de Opinión Pública N° 51*: esta versión será la que se utilizará en los análisis y ejemplos posteriores de este tutorial.

### 6.1. Descarga de una base de datos de interés

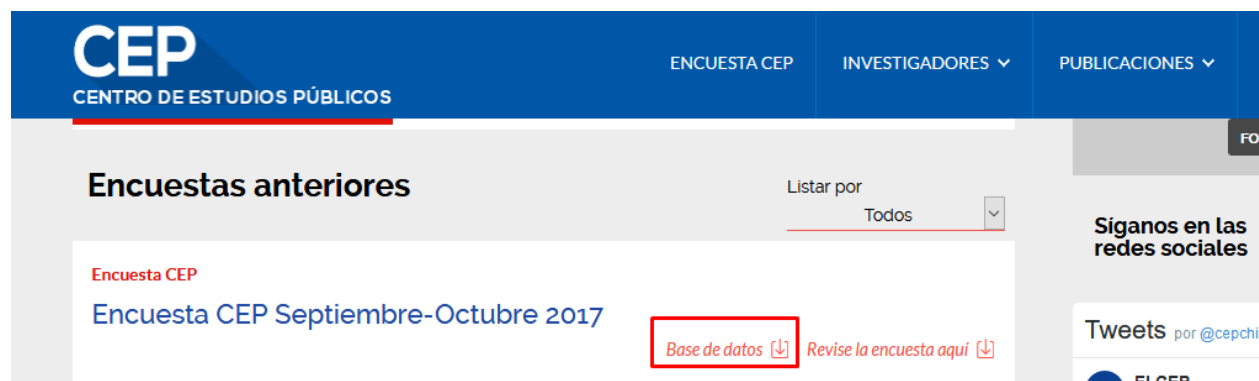
Una primera acción a realizar es la descarga de la base de datos indicada al computador. Esto puede realizarse accediendo a la página web indicada del CEP. En la botonera superior de esta página - al menos a la fecha de publicación de este documento - se encuentra un botón de acceso a la *Encuesta CEP*.

Imagen 28: Página del CEP



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

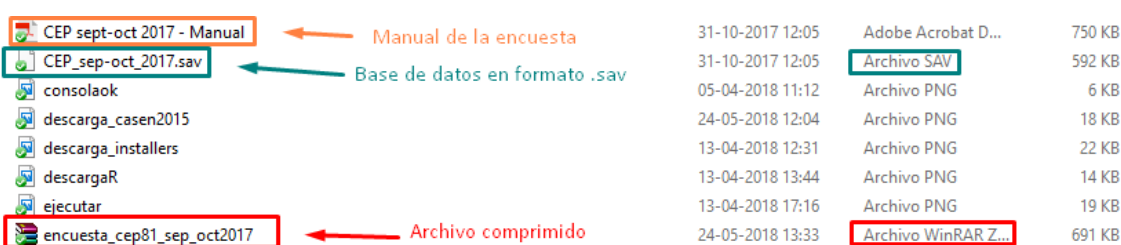
Imagen 29: Repositorio de bases de datos históricas de la encuesta CEP



Allí, oprimiendo el botón *Base de datos* se podrán descargar los archivos de interés.<sup>12</sup>

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 click derecho sobre el archivo descargado, y seleccionando la opción *Extraer aquí* se obtendrá el archivo original (de extensión “.sav”)

Imagen 30: Archivos resultantes luego de descomprimir el archivo descargado



Como se ve en la imagen el archivo resultante es un archivo de extensión .sav, lo que indica que se trata de un archivo configurado para que sea leído por SPSS. De aquí en más se usará este archivo para desarrollar ejemplos de análisis estadístico. Además de la base de datos, se descarga también un archivo PDF que es el manual de uso de la encuesta.<sup>13</sup>

## 6.2. ¿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.

<sup>12</sup>Para los siguientes apartados se trabajará con esta base de datos. El CEP sólo la dispone en formato SPSS (extensión .sav).

<sup>13</sup>Es importante respetar el nombre del archivo para leerlo desde RStudio. En este caso se ha decidido nombrar al archivo de la siguiente forma: *CEP\_sep-oct\_2017.sav*.

Para ello se utiliza un paquete específico llamado *haven* que cuenta con funciones para abrir bases de datos desde diferentes formatos.<sup>14</sup> Para eso primero se descarga e instala el paquete en el computador:

### Ejercicio 6.1

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

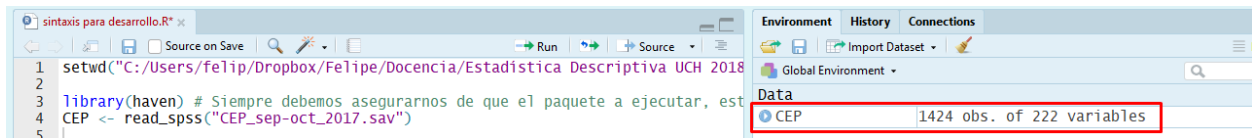
La función específica a utilizar es *read\_spss* cuya ejecución es muy simple. Luego de la función se indica el nombre del archivo a leer entre paréntesis y entre comillas. Como se observa a continuación, esta función sirve para leer la base de datos de la Encuesta CEP descargada en el anterior apartado. 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*):<sup>15</sup>

### Ejercicio 6.2

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

### Imagen 31: Base de datos CEP cargada como objeto en entorno de trabajo

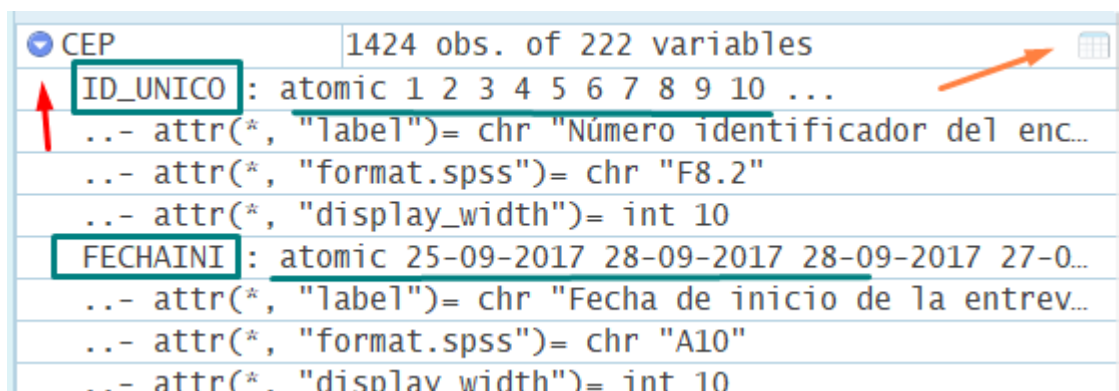


Ahora que se cuenta con una base de datos cargada en el entorno de trabajo es importante poder explorar sus principales características. Como se observa en la imagen a continuación existe un botón (flecha azul indicando hacia abajo) que permite desplegar la estructura interna de la base de datos existente en el entorno de trabajo.

<sup>14</sup>Para efectos de este documento tutorial sólo se explicará cómo importar bases de datos desde el formato SPSS, que es uno de los más utilizados en ciencias sociales, como también desde archivos generalmente usados vía Microsoft Excel. No obstante, el paquete *haven* también incorpora funciones para importar datos desde el formato para el software Stata, programa más utilizado en el campo específico de la economía. Estas funciones son similares a los códigos utilizados para trabajar con bases de datos en formato en SPSS. Para más información, leer el siguiente [enlace](#).

<sup>15</sup>Es importante recordar que para que el programa sepa donde buscar los archivos indicados, hay que especificar una “carpeta de trabajo” donde se encuentren los archivos a usar. En este caso, se debe definir una carpeta donde se guarde - por ejemplo - la base de datos de la Encuesta CEP y la sintaxis donde se irán diseñando los análisis. Para más detalles, repasar el apartado 4.2 *Carpeta de trabajo y memoria temporal del programa*.

Imagen 32: Información de los vectores que componen la base de datos



Variable	Attributes
CEP	1424 obs. of 222 variables
ID_UNICO	atomic 1 2 3 4 5 6 7 8 9 10 ...
..-	attr(*, "label")= chr "Número identificador del enc...
..-	attr(*, "format.spss")= chr "F8.2"
..-	attr(*, "display_width")= int 10
FECHAINI	atomic 25-09-2017 28-09-2017 28-09-2017 27-0...
..-	attr(*, "label")= chr "Fecha de inicio de la entrev...
..-	attr(*, "format.spss")= chr "A10"
..-	attr(*, "display_width")= int 10

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. 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.

Imagen 33: Visualización de base CEP como planilla



ID_UNICO	FECHAINI	VOTACION_1	VOTACION_2	VOTACION_3	VOTACION_4
Número identificador del encuestado	Fecha de inicio de la entrevista	Voto con urna: Primera votación	Voto con urna: Segunda votación	Voto con urna: Tercera votación	Voto con urna: Cuarta votación
1	25-09-2017	10	4	4	4
2	28-09-2017	2	3	3	3
3	28-09-2017	8	5	1	1
4	27-09-2017	7	2	2	2
5	02-10-2017	7	2	2	2
6	29-09-2017	7	2	2	2
7	01-10-2017	4	1	2	2
8	01-10-2017	8	1	1	1
9	03-10-2017	4	1	1	1
10	15-10-2017	7	2	2	2
11	22-09-2017	4	1	1	1
12	29-09-2017	7	2	2	2
13	01-10-2017	8	1	1	1
14	07-10-2017	11	5	1	1

### 6.3. ¿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.<sup>16</sup>

<sup>16</sup>Evidentemente con fines prácticos antes que de investigación. Por ejemplo, procesos contables o de salud, por ejemplo. Estos registros primarios que muchas instituciones realizan - por ejemplo una empresa o una clínica - muchas veces constituyen una interesante fuente de información que puede ser usada para el análisis social.

Si bien el formato *planilla 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 otros 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.<sup>17</sup> Para efectos de este tutorial el archivo se puede descargar desde el [siguiente enlace](#) (indicado en la presentación de este documento).

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.

**Imagen 34: Hoja con identificación de respondientes**

	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						

**Imagen 35: Hoja con respuestas de encuesta**

	A	B	C	D	E	F	G	H	I	J	K	L	M
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

<sup>17</sup>Basta con ir a *Guardar como* y seleccionar el formato compatible con Excel (*.xlsx*) para obtener una planilla de Excel con la base de datos originalmente guardada en formato *.sav*.



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`.

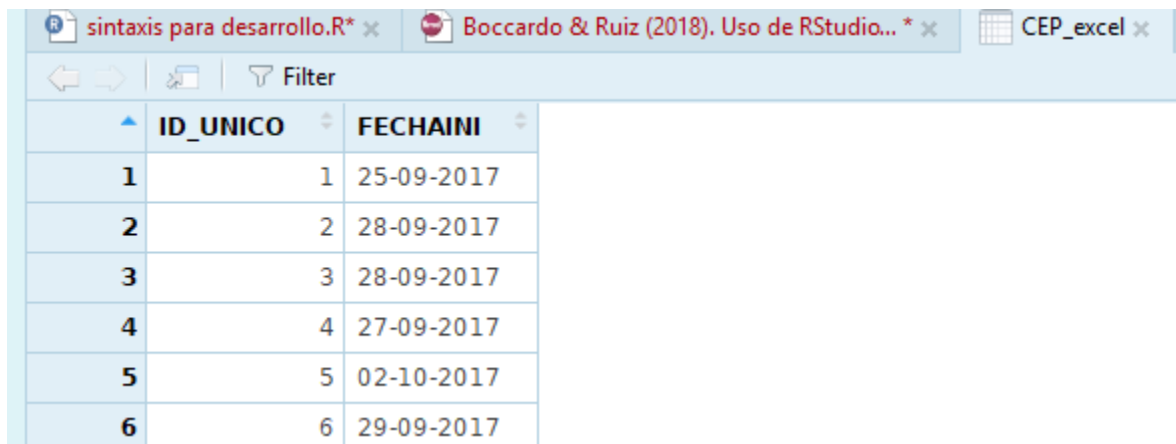
### Ejercicio 7.1

```
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.

Imagen 36: Hoja de identificación de respondientes cargada como base de datos



	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 vendría a ser lo mismo). A continuación se sobrescribe el objeto creado en el entorno de trabajo, actualizando la función con esta información.

### Ejercicio 7.2

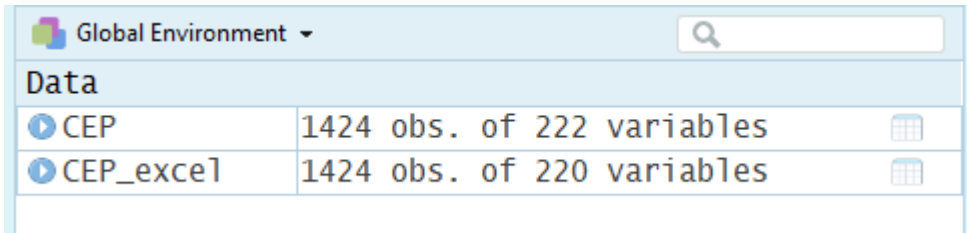
```
CEP_excel <- read_excel("CEP_sep-oct_2017.xlsx", sheet = 2)
#indica posición de hoja en el libro de trabajo.



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.<sup>18</sup>

<sup>18</sup>Dos variables menos que las 222 presentes en la base original. Tal diferencia se explica pues manualmente se pasaron a otra hoja las dos primeras variables de identificación de las variables.

Imagen 37: hoja con respuestas a encuesta cargada como base de datos



Global Environment		
Data		
▶ CEP	1424 obs. of 222 variables	
▶ CEP_excel	1424 obs. of 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 que 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.

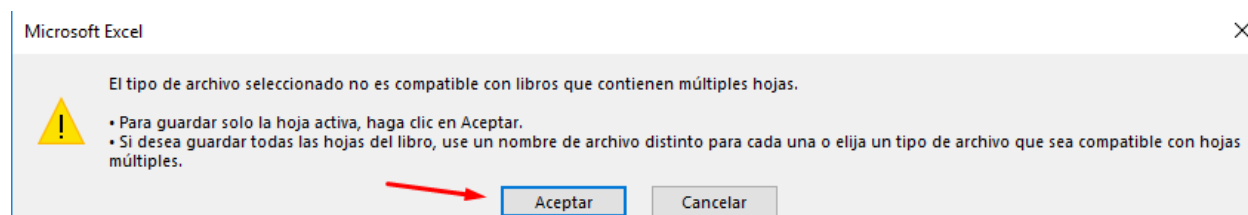
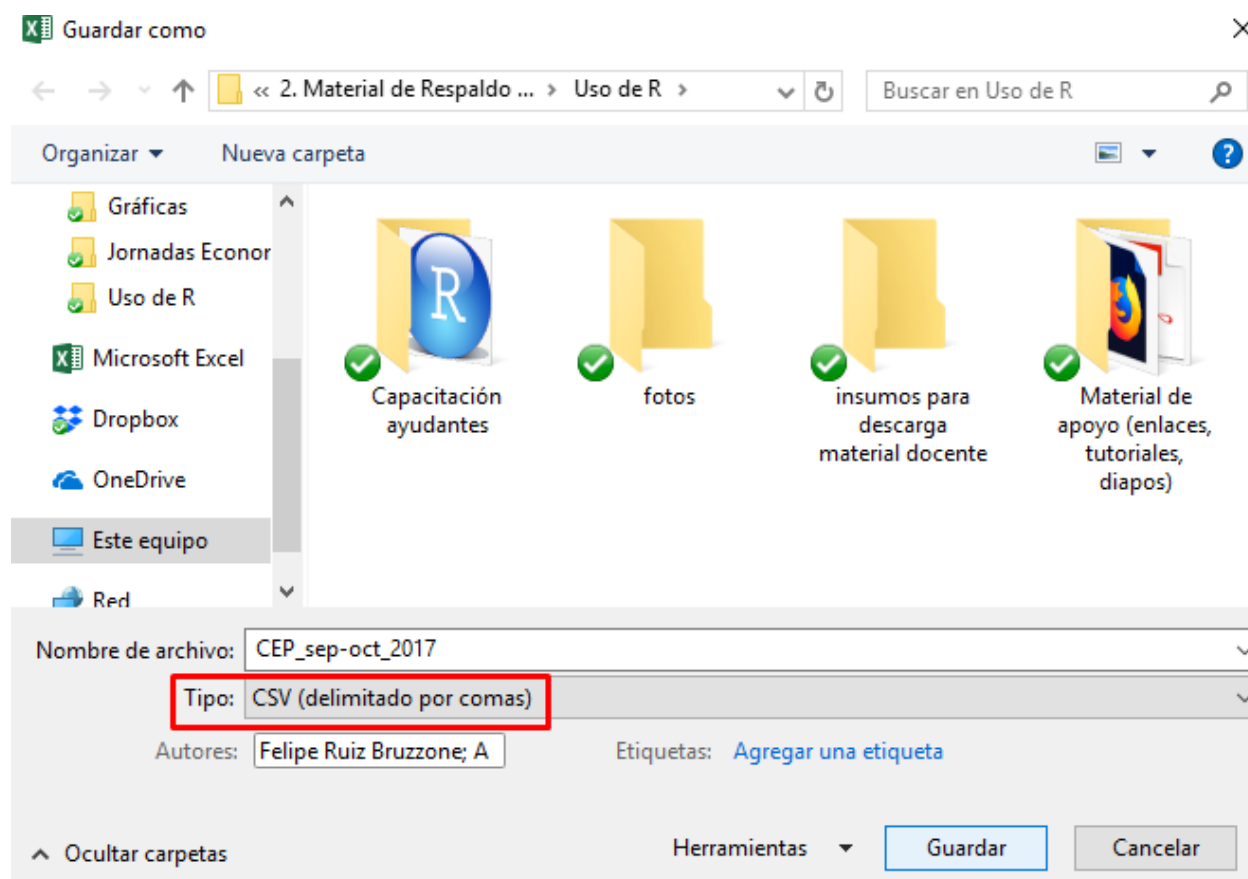
### Ejercicio 7.3

```
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*).

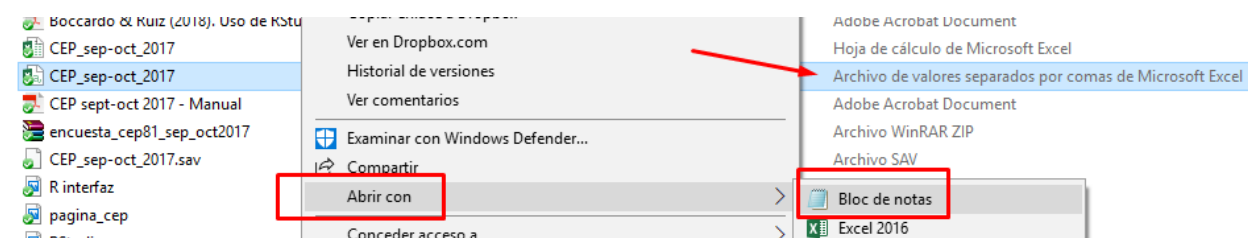
Para esto, como se observa en las siguientes imágenes, basta con que desde el archivo excel en cuestión, se utilice a la opción *Guardar como* y seleccionando el formato señalado. El programa debería advertir al usuario que el formato seleccionado no soporta múltiples hojas de trabajo, por lo que guardará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

Imagen 38: Guardar planilla de libro de Excel en formato CSV



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...*

Imagen 39: Estructura interna del archivo CSV



#### CEP\_sep-oct\_2017: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
VOTACION_1;VOTACION_2;VOTACION_3;VOTACION_4;SV_1;SV_2;MB_P1_1;
;DPP_18A;DPP_18B;DPP_18C;DPP_18D;DPP_19A;DPP_19B;DPP_19C;DPP_1
10;4;4;4;8;3;3;5;7;1;3;4;2;2;3;2;8;2;2;7;#;NULO!;3;#;NULO!;4;#
2;3;3;3;10;5;10;1;2;2;2;4;3;3;2;2;2;2;2;5;#;NULO!;3;#;NULO!;5;
8;5;1;5;10;10;11;10;7;4;1;4;1;2;1;1;1;1;5;#;NULO!;3;#;NULO!;
7;2;2;2;8;5;7;12;6;3;1;3;2;2;2;2;2;2;2;2;77;77;1;#;NULO!;1;#;NULO
7;2;2;2;5;5;13;1;11;2;1;2;3;2;2;2;2;2;2;4;#;NULO!;1;#;NULO!;3;
7;2;2;2;9;5;11;1;5;2;1;3;2;2;2;2;2;2;2;4;#;NULO!;1;#;NULO!;1;#
4;1;2;2;9;5;16;3;1;4;2;3;3;2;1;2;1;1;1;3;#;NULO!;3;#;NULO!;4;#
8;1;1;1;6;8;2;8;1;2;2;2;2;2;1;2;1;1;1;9;#;NULO!;5;5;5;#;NULO!;
```

Se observa 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 7.4

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

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

### Imagen 40: Mensaje de error al leer archivo CSV

```
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.

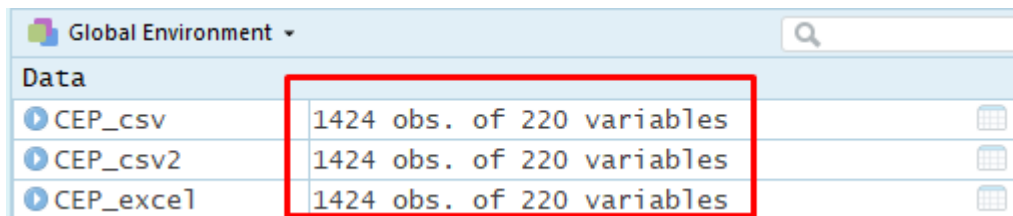
Para solucionar tal problema se indican dos opciones.

- 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 7.5

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

Imagen 41: Bases de datos Excel y CSV con la misma estructura de casos y variables



Global Environment	
Data	
CEP_csv	1424 obs. of 220 variables
CEP_csv2	1424 obs. of 220 variables
CEP_excel	1424 obs. of 220 variables

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 7.6

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

## 6.4. Construir una base de datos sólo con variables de interés

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 seleccionará 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.<sup>19</sup>

<sup>19</sup>Para evitar problemas de redacción en los códigos de sintaxis, así como para facilitar la redacción de instrucciones computacionales, se sugiere: i) usar nombres cortos pero descriptivos de las variables, ii) usar sólo mayúsculas o minúsculas, sin combinarlas, y iii) evitar usar espacios, privilegiando guiones normales o guión bajo.

Cuadro 1: Detalle de las variables específicas a considerar para los análisis

Descripción de la variable	Nombre en base original	Nuevo nombre	Valores
Ponderador	POND	<i>pond</i>	Números simples para ponderación.
Género informado	SEXO	<i>sexo</i>	1 = hombre, 2 = mujer.
Región de residencia	REGION	<i>region</i>	Número de región, del 1 al 15.
Fecha de nacimiento (edad)	DS_P2_EXACTA	<i>edad</i>	Edad cumplida en número entero.
Satisfacción la propia vida (escala de 1 a 10)	SV_1	<i>satisfaccion</i>	Escala del 1 al 10.
Percepción de satisfacción de chilenos con su vida (escala de 1 a 10)	SV_2	<i>satisfaccion_chilenos</i>	Escala de 1 al 10
Evaluación de la situación económica nacional	MB_P2	<i>eval_econ</i>	Escala del 1 al 5

Para construir una nueva base de datos sólo con las variables especificadas, la opción a usar en este tutorial es la función *select* del paquete *dplyr*, como se observa a continuación:

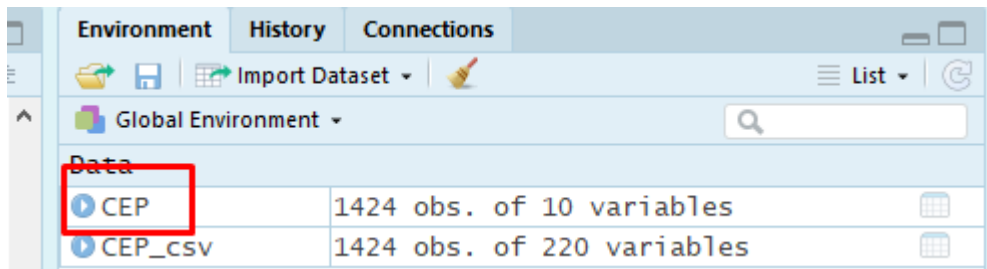
### Ejercicio 8.1

```
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 (1424) 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.

Imagen 42: Base de datos con selección de variables



	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

Una vez hecha esta operación de selección de variables se creó una nueva base de datos que contiene sólo 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` y para guardar un objeto de formato (o extensión) `.RData`.

## Ejercicio 8.2

```
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.

### Imagen 43: Nueva base de datos guardada en carpeta de trabajo

File Name	Date/Time	File Type
save	14-04-2018 0:11	Archivo PNG
<b>seleccion_CEP</b>	29-05-2018 12:00	<b>Archivo RDATA</b>
sintaxis para desarrollo	29-05-2018 12:00	Archivo R

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 8.3

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

## 6.5. Exploración de base de datos y recodificación de variables

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

### Ejercicio 9.1

```
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.<sup>20</sup>

### Ejercicio 9.2

```
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 resultado a continuación, 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. Por lo general 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 por lo general indicará la cantidad de casos y variables (en ese orden).

### Ejercicio 9.3

```
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 9.4

```
summary(CEP)
```

---

<sup>20</sup>Al momento de codificar una encuesta, debido a que se trata de un proceso manual, pueden producirse algunos errores. Por ejemplo, la persona que codifica puede haber introducido sin querer un espacio en blanco antes o después del nombre de la variable; tanto las planillas de cálculo como la visualización de bases de datos de R no muestran tales espacios en blanco pues sólo visualizan las letras. Es por eso que a simple vista resulta difícil ver si el nombre de una variable es “variable” o “\_variable” (el guión bajo representa la existencia de un espacio en blanco, no se observa así cuando R entrega la información del nombre de la variable). Por otra parte, para prevenir errores de tipeo en la redacción de las sintaxis de análisis, es preferible copiar el nombre de variables u otros objetos directamente desde el listado que resulta al aplicar la función *names*.



Imagen 44: Observación general de las variables

```
> 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 anterior, estas variables presentan valores 9 o 99 como máximos, siendo que se trata de variables que presentan un rango menor de valores posibles. Tal información constituye una primera alerta sobre las transformaciones 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 simple 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 10.1

```
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 10.2

```
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.

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*”).

### Ejercicio 10.3

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

Ahora se recodificará la variable *región* indicando que sólo habrán dos categorías: “otras regiones” y “región Metropolitana”. Como en el caso anterior, primero conviene observar las características generales de la variable: interesa saber cuantos 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 11.1

```
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 la categoría “13”, que equivale a la región Metropolitana, presenta 501 casos. A la vez 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 introduciremos una variante. Dentro de una función, R siempre aplica las funciones que corresponden al paquete de la función principal (en este caso, *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* incorporamos el argumento *car::*, lo que fuerza a que se ejecute el comando *recode* que proviene del paquete *car* y así poder recodificar indicando tramos de datos.<sup>21</sup>

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 11.2

```
#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 estructura 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 *satisfacción que los chilenos en general sienten con su propia vida*. Primero se exploran ambas variables.

<sup>21</sup>Esto resulta de utilidad en variables de nivel de medición nominal u ordinal con muchas categorías, pero especialmente en variables de nivel de medición intervalar o de razón como ingreso o edad. Dado que presentan un rango muy amplio de valores resulta mucho más sencillo recodificar indicando tramos de valores. En el caso de que los últimos tramos sean valores perdidos, la función construida puede incorporar un último tramo indicando la palabra *else* y asignando el valor lógico usado por R para denotar casos perdidos (“NA”): CEP <- mutate(CEP, region\_factor = car::recode(CEP\$region, "1:12 = 1; 13 = 2; 14:15 = 1; else = NA")) Veremos una aplicación concreta de esto en la siguiente recodificación.

## Ejercicio 12.1

```
#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
```

```
#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
```

A partir de los resultados<sup>22</sup> se ve que ambas variables están compuestas por valores discretos del 1 a 10. Sin embargo, los valores máximos son 88 y 99, que denotan las categorías “no sabe” y “no responde”. Por tanto se efectuarán dos operaciones de transformación de estas variables.

La primera será asignar 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:<sup>23</sup>

<sup>22</sup>En este caso, se indican dos maneras de explorar los valores de la variable. Una opción es el comando *table* que ya ha sido utilizado. Sin embargo, otra opción que sirve es comando *summary*, mediante el cual se construye un resumen de estadísticos descriptivos. No obstante, esta última opción impide ver el valor 88, pues solamente muestra el valor máximo de la distribución de datos de la variable (99).

<sup>23</sup>Como se observa en el siguiente código de R, el comando para codificar valores como NA en la misma variable tiene la siguiente estructura: se indica la variable (*base/variable*). luego entre corchetes se indica un valor específico dentro de la misma, eso se hace escribiendo nuevamente la variable seguida de dos signos igual y el valor de interés (*[base\$variable==88]*). Con eso se indica qué valor se busca; luego, con el asignador se indica que a tal selección se asigna el valor NA (*<- NA*).

## Ejercicio 12.2

```
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
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
```

```
CEP$satisfaccion_chilenos[CEP$satisfaccion_chilenos==88]<- NA
CEP$satisfaccion_chilenos[CEP$satisfaccion_chilenos==99]<- NA
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, ya no se mostrarán los casos *perdidos* (88 y 99). Y si se solicitan estadísticos descriptivos, se observará que ahora los valores máximos coinciden con los valores mínimos y máximos del rango de la variable, indicándose además la cantidad de casos NA que han resultado excluidos del cálculo.

Ahora se explorará la variable que mide la percepción de la persona entrevistada respecto a la situación económica del país.

## Ejercicio 13.1

```
#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* con valores del 1 al 5. En este caso los valores 8 y 9 corresponden a las categorías “no sabe” y “no contesta” (*casos perdidos*). Esta variable se recodificará en tres categorías de evaluación (negativa, neutra y positiva). También se indicará una segunda forma de asignar casos perdidos (valores *NA*), esta vez desde una recodificación vía comando *mutate*.<sup>24</sup>

### Ejercicio 13.2

```
#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.

<sup>24</sup>Los valores originales son “1 = *Muy mala*”, “2 = *Mala*” “3 = *Ni buena ni mala*”, “4 = *buena*” y “5 = *Muy buena*”.

## 7. Estadística univariada con RStudio

Habiendo construido y configurado una base de datos *ad hoc* para nuestros análisis, en el presente capítulo se presenta la forma de calcular diferentes estadísticos univariados en sus vertientes de estimación muestral o poblacional.<sup>25</sup>

Cabe recordar que cuando se habla de estadística univariada **muestral** refiere a la construcción de estadísticos que provienen directamente desde la información presente en una muestra de casos; para esta modalidad, basta calcular los diferentes estadísticos de interés para describir los datos muestrales. Por otra parte, cuando se habla de estadística univariada **poblacional** refiere al cálculo de parámetros, es decir corresponde a una *evaluación estadística* de los datos muestrales para conocer en qué medida logran representar al conjunto de la población de interés; esto generalmente implica calcular los estadísticos (*estadística muestral*) para luego determinar un intervalo de variación donde se encuentran *probabilísticamente* los parámetros que representan los datos de la población. Este ámbito de la estadística se conoce como **estadística poblacional o inferencial**: nociones como intervalo de confianza, error de estimación y nivel de confianza forman parte de esta segunda forma de aproximación a los datos.

### 7.1. Estadística univariada muestral

#### 7.1.1. 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.default`, 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.<sup>26</sup> Así, se obtienen los resultados de ambos estadísticos muestrales.

##### Ejercicio 14.1

```
mean.default(CEP$satisfaccion_vida, na.rm = TRUE)
```

```
## [1] 7.311174
```

```
mean.default(CEP$satisfaccion_vida, na.rm = TRUE, trim = 0.025)
```

```
## [1] 7.390625
```

Para el caso de la mediana, se trata de un comando similar. Se utiliza el comando `median.default`, 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 el resultado de este estadístico muestral.

<sup>25</sup>En este apartado no se profundizará en métodos avanzados para optimizar la presentación de resultados: es importante distinguir entre la construcción de resultados estadísticos para su uso en el proceso de investigación, respecto a la construcción de informes y resultados con un formato adecuado para su divulgación. Éste último aspecto lo revisaremos en los dos últimos capítulos de este documento, enfocados en el uso de los paquetes *ggplot* y las funcionalidades para la construcción de informes vía *RMarkdown*.

<sup>26</sup>Es por eso que si se busca contar con una media recortada al 5 %, se indica una proporción de 0.025, que se recortará a ambos extremos de la distribución, alcanzando un recorte total de una proporción de casos del 0.05 o 5 %.

## Ejercicio 14.2

```
median.default(CEP$satisfaccion_vida, na.rm = TRUE)
```

```
## [1] 7
```

Para el cálculo de la moda, debe haberse instalado previamente el paquete *modeest* y haberlo cargado en la sesión de trabajo. Posteriormente, se utiliza el comando *mfv*<sup>27</sup> para calcular la moda. El resultado de este comando indica el valor, o los valores, con más frecuencia dentro de la distribución.

## Ejercicio 14.3

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

```
## [1] 50
```

Así, el valor que más se repite para la variable *satisfaccion\_vida* en la muestra es el número 50.

### 7.1.2. 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. Para estos ejemplos, asignaremos el resultado a un nuevo objeto que quedará almacenado en el entorno de trabajo.

## Ejercicio 15.1

```
tabla <- table(CEP$eval_econ_factor)  
tabla
```

```
##  
## Positiva   Neutra Negativa  
##      471      730      209
```

Para el cálculo de frecuencias relativas, se usa la tabla de frecuencias simples ya construida. 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 con el total de casos.

## Ejercicio 15.2

```
#Frecuencias relativas (proporciones)  
tabla_prop <- prop.table(tabla)  
tabla_prop
```

```
##  
## Positiva   Neutra Negativa  
## 0.3340426 0.5177305 0.1482270
```

---

<sup>27</sup>Sigla en inglés correspondiente a la expresión *most frequent values* o *valores más frecuentes* en castellano.



Como se observa en las líneas de comandos, para construir una tabla de porcentajes se multiplica la tabla de proporciones por 100. Este resultado también se almacena en un objeto diferente en el entorno de R.

### Ejercicio 15.3

```
#Frecuencias relativas (porcentajes)
tabla_porcentaje <- ((tabla_prop)*100)
tabla_porcentaje
```

```
##
## Positiva  Neutra Negativa
## 33.40426  51.77305 14.82270
```


¿De qué sirve haber ido guardando las tablas anteriores como objetos? Esto permite exportar los resultados a un formato de planilla que facilitará su edición para presentarlos en un reporte formal de resultados. Con tal objetivo se emplea la función `write.csv2` para escribir cada una de esas tablas en una planilla de formato CSV ejecutable con cualquier programa para edición de planillas cálculo como Microsoft Excel o Calc de Libre Office y Open Office. 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.


### Ejercicio 16


```
write.csv2(tabla, file = "Tabla 1.csv")
write.csv2(tabla_prop, file= "Tabla 2.csv")
write.csv2(tabla_porcentaje, file= "Tabla 3.csv")
```

El resultado de la anterior operación serán tres archivos de tipo CSV creados en la ubicación definida como carpeta de trabajo para la sesión de R. Así, se cuenta con tres archivos editables desde excel que permitirían unificar tales tablas y editarlas en un formato adecuado para su presentación en reportes formales.

### Imagen 45: Exportación de resultados a archivos CSV

 **Tabla 1**

 **Tabla 2**

 **Tabla 3**

03-06-2018 20:29

03-06-2018 20:29

03-06-2018 20:29

Archivo de valores separados por comas de Microsoft Excel

Archivo de valores separados por comas de Microsoft Excel

Archivo de valores separados por comas de Microsoft Excel

**Tabla 1 - Excel**

	A	B	C
1		Var1	Freq
2	1	Positiva	471
3	2	Neutra	730
4	3	Negativa	209
5			
6		FRECUENCIAS ABSOLUTAS	
7			
8			

**Tabla 2 - Excel**

	A	B	C
1		Var1	Freq
2	1	Positiva	0,33404255
3	2	Neutra	0,5177305
4	3	Negativa	0,14822695
5			
6		FRECUENCIAS RELATIVAS (PROPORCIONES)	
7			
8			
9			

**Tabla 3 - Excel**

	A	B	C
1		Var1	Freq
2	1	Positiva	33,4042553
3	2	Neutra	51,7730496
4	3	Negativa	14,822695
5			
6		FRECUENCIAS RELATIVAS (PORCENTAJES)	
7			
8			
9			
10			

Para el cálculo de cuantiles, con la función *quantile* es posible calcular los casos equivalentes a diferentes proporciones de la distribución. El primer argumento de la función es la variable a considerar; 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 17

```
quantile(CEP$satisfaccion_chilenos, prob = c(0.25, 0.5, 0.75), na.rm = TRUE)
```

```
## 25% 50% 75%  
##    4    5    6
```

Con estos resultados se puede concluir que solamente el 25 % superior de los casos presenta una evaluación de su vida mayor o igual a 6.

### 7.1.3. 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 a continuación, 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*.

### Ejercicio 18.1

```
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
```

El cálculo de la varianza y desviación estándar, sigue la misma lógica en lo que refiere a la estructura de la función. Respectivamente las funciones utilizadas son *var* para varianza y *sd*, para desviación estándar. También debe incluirse el argumento *na.rm = TRUE* en el caso de que hubieran sido codificados como *NA* los valores perdidos.

## Ejercicio 18.2

```
var(CEP$satisfaccion_chilenos, na.rm = TRUE)
```

```
## [1] 3.017771
```

```
sd(CEP$satisfaccion_chilenos, na.rm = TRUE)
```

```
## [1] 1.737173
```

En el caso del coeficiente de variación, como se observa en el siguiente código, puede calcularse de manera sencilla, ejecutando una división entre la desviación estándar y la media. Sin embargo, 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 para así evitar errores de especificación de tales valores, ya sea por tipeo del valor o por su aproximación decimal.

## Ejercicio 18.3

```
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
```

### 7.1.4. Cálculo agregado de estadísticos muestrales para su exportación a planillas

A continuación, se explicará cómo calcular de manera agregada diversos estadísticos para la construcción de una tabla exportable a formato planilla, con todos los estadísticos muestrales de interés.

En relación al primer aspecto es de utilidad la función *summary* que se ejecuta con la variable de interés como principal argumento. 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 se obtiene una tabla resumen de estadísticos muestrales: valores mínimo y máximo, primer cuartil, mediana, media, tercer cuartil y cantidad de valores codificados como perdidos (valores *NA*).

## Ejercicio 19.1

```
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 los siguientes trozos de código de R. Primero, los resultados del comando *summary* se guardan como un objeto específico (*descriptivos*). Luego, si se le aplican las funciones *names* y *as.numeric* a tal objeto se obtiene un vector que contiene los encabezados de los resultados y otro vector que contiene su valores numérico. 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*.

## Ejercicio 19.2

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

#Ver nombres y valores del objeto
names(descriptivos)

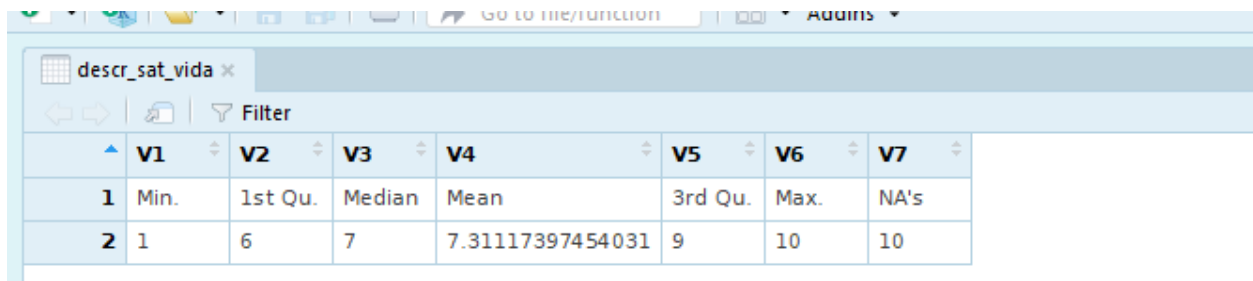
## [1] "Min."      "1st Qu."  "Median"   "Mean"     "3rd Qu."  "Max."     "NA's"

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)
```

Imagen 46: Comando summary configurado como 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 19.3

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

Imagen 47: Resultados de summary exportados a archivo CSV

	V1	V2	V3	V4	V5	V6	V7
1 Min.	1,0	6,0	7,0	7,3	9,0	10,0	10
2 1st Qu.							
3 Median							
4 Mean							
5 3rd Qu.							
6 Max.							
7 NA's							

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 y unirlos en una matriz de datos para así poder guardarlos en formato planilla de cálculo. Como se observa a continuación, estos valores posteriormente se concatenan como un sólo vector (*descriptivos\_satvida*), que será una fila de la planilla a construir. Además, se construye un vector llamado *nombres* con los encabezados de cada estadístico, que se utilizará como fila de títulos de la planilla de cálculo a exportar.

### Ejercicio 20.1

```
#Cálculo simple de estadísticos 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)

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

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

Con esta información ya es posible configurar una planilla de estadísticos descriptivos completamente personalizada. Para ello se configura una matriz de datos (*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 20.2

```
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 a continuación, editable para la construcción de reportes formales.

**Imagen 48: Tabla de estadísticos muestrales personalizada y exportada a CSV**

	A	B	C	D	E	F	G	H	I	J	K
1		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
2	nombres	Mínimo	Q1	Media	Media recort	Mediana	Moda	Varianza	Desviación Estándar	Q3	Máximo
3	descriptivos	1,0	6,0	7,3	7,4	7,0	10,0	4,5	2,1	9,0	10,0

## 7.2. Estadística univariada poblacional

En el apartado anterior se revisaron procedimientos para calcular estadísticos univariados de alcance muestral. Ahora se explicará como calcular estadísticos univariados pero de alcance poblacional. Concretamente se indicarán los procedimientos para calcular intervalos de confianza para medias y proporciones.

### 7.2.1. Cálculo de intervalos de confianza para proporciones

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, 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 21

```
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 48,63 % y 53,89 % de los casos.

### 7.2.2. 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).

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 22

```
library(Publish)

ci.mean(CEP$satisfaccion_vida) #Nivel de confianza por defecto.

## mean CI-95%
## 7.31 [7.20;7.42]

ci.mean(CEP$satisfaccion_vida, alpha = 0.2) #Definición manual del nivel de confianza.

## mean CI-80%
## 7.31 [7.24;7.38]
```

Como se observa en los resultados anteriores, 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.

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 el segundo ejemplo se define el valor 0,2 o 20 %. Esto implica que observando los resultados se puede afirmar que con un nivel de confianza del 80 % 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, 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 virguitilla (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 23

```
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 permitirían 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.

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

### 8.1. 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 el diseño de gráficos que permite una mejor visualización de resultados, sobre todo enfocados en el momento de *divulgación* de resultados de investigación (A. Field, Miles, & Field, 2012, pp. 116-117).

#### 8.1.1. Gráficos de barras y gráficos de tortas

De modo general, los gráficos de barras y de tortas se recomiendan para 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, pp. 26-37; Ritchey, 2008, pp. 42-48).

Como un paso preliminar para la demostración de la construcción de gráficos mediante R se deberá configurar 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.

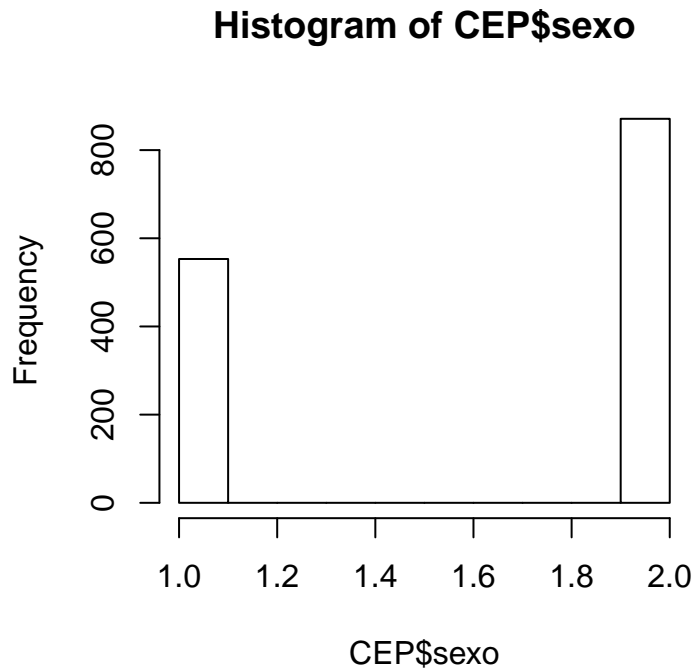
#### Ejercicio 24

```
# 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 25.1

```
#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.

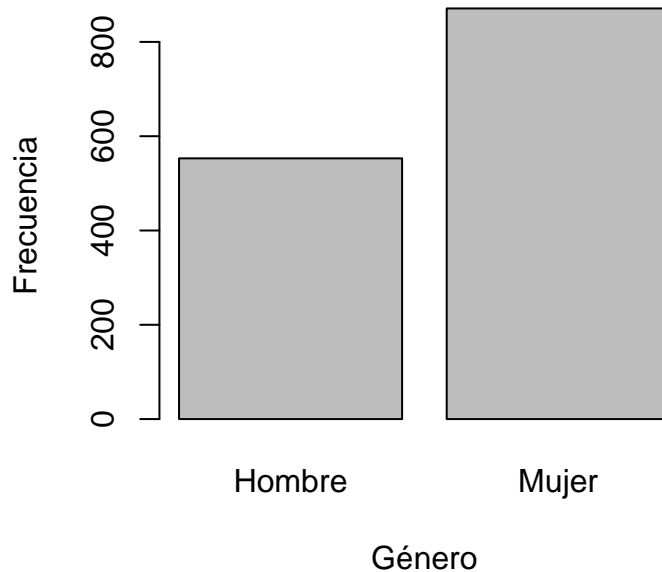
Por ello se sugiere el uso del comando *plot* para construir gráficos de barras. De manera general, este tipo de gráficos se usan para variables cualitativas (nominales u ordinales). Las categorías de una variable se sitúan en el eje X, mientras que la frecuencia absoluta o relativa (generalmente porcentajes) se sitúa en el eje Y (Ritchey, 2008, pp. 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 25.2

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

**Gráfico de barras 1**



Otra forma de presentar variables nominales son los gráficos circulares (o de “torta”); se trata de gráficos circulares 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, pp. 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, 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 26.1

```
#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
```

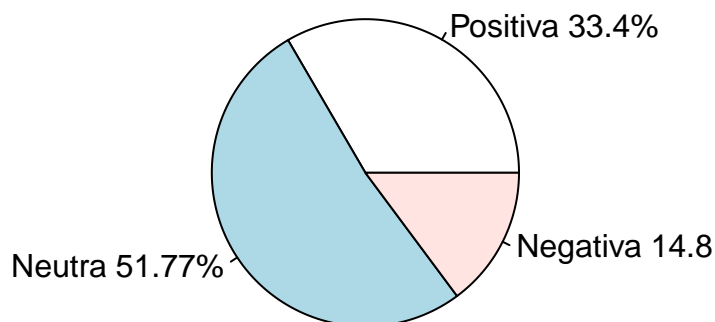
```
## [1] "Positiva 33.4%" "Neutra 51.77%" "Negativa 14.82%"
```

Considerando los dos elementos (*porcentajes* y *etiquetas*) se construye el gráfico de torta. 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 comandos *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 26.2

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

**Gráfico de torta 1**



Evaluación de la situación económica

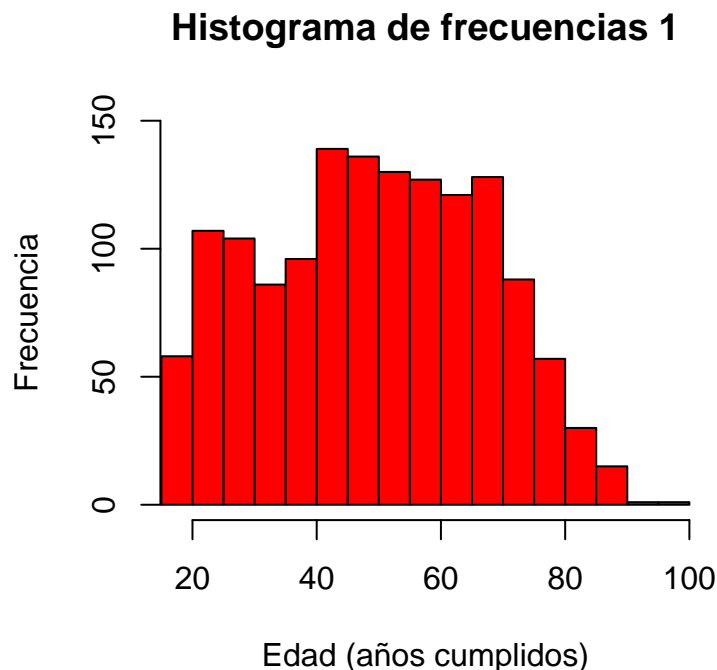
### 8.1.2. Histogramas

El histograma se utiliza para la representación gráfica 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, pp. 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).<sup>28</sup>

#### Ejercicio 27.1

```
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))
```

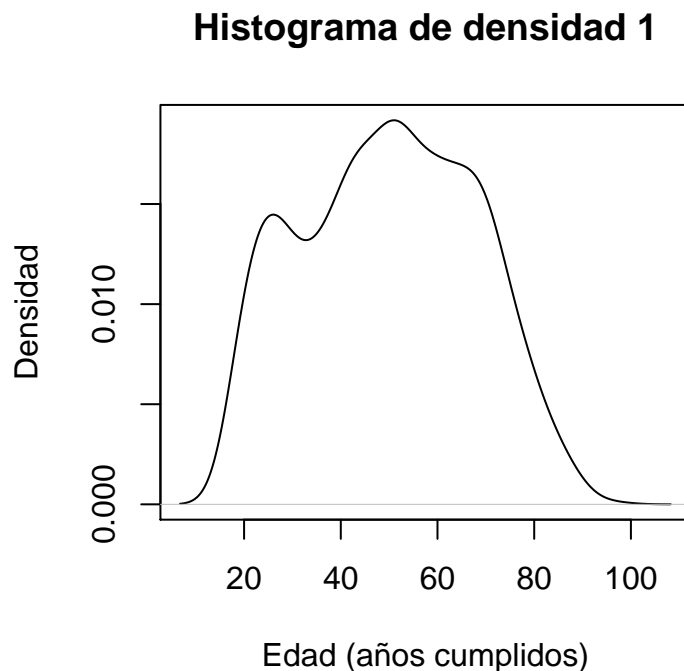


<sup>28</sup>R incluye una paleta bastante grande de colores. Para una explicación general de cómo funcionan los colores en R se sugiere visitar este [enlace](#) o descargar la [siguiente guía en inglés](#). Para descargar la plantilla general de colores, y asociar visualmente un color con su nombre estándar, presionar el siguiente [enlace que efectúa la descarga de tal plantilla en formato PDF](#). Si se ejecuta el comando `colors()` se obtiene el listado general de posibilidades de colores con nombre, que incluye R.

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 grafica esta información indicando título al gráfico y a los ejes de la forma que ya ha sido explicada.

### Ejercicio 27.2

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

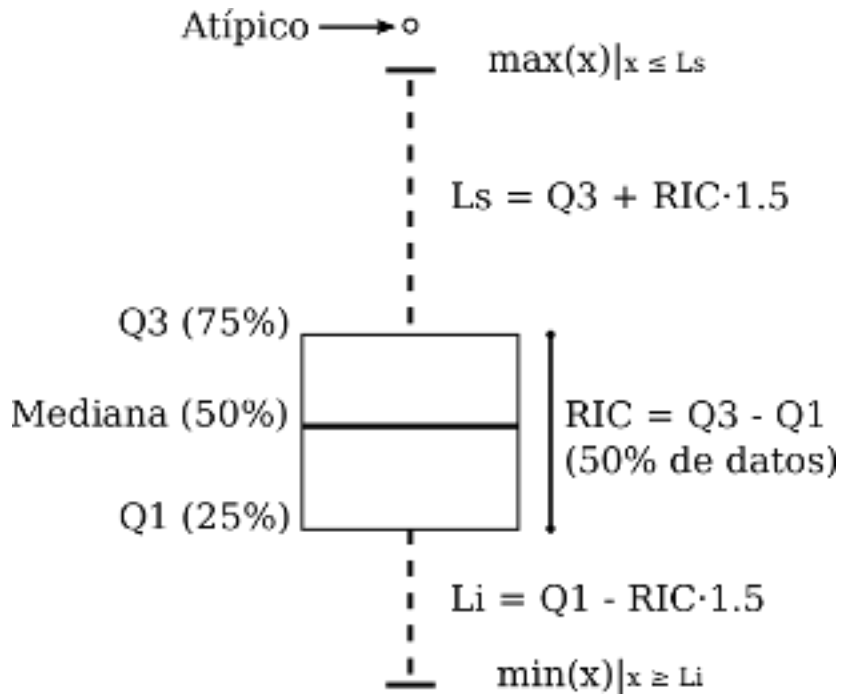


#### 8.1.3. Diagrama de cajas

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 (A. Field et al., 2012, pp. 151-152).

**Imagen 49: Gráfico de caja**



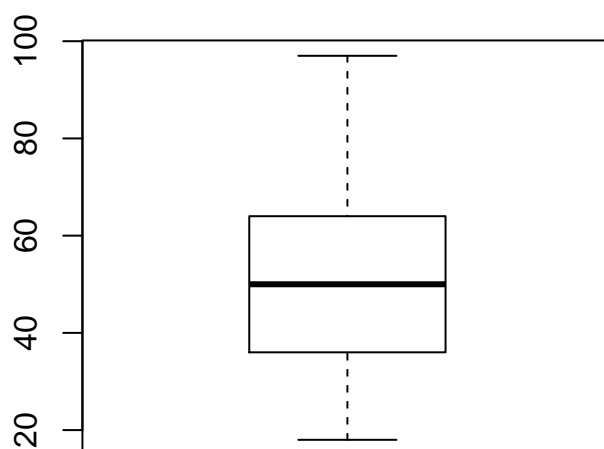
Para construir un gráfico de cajas el código de R es bastante simple. Basta con ejecutar la función `boxplot` indicando como primer argumento la variable de interés. A esto se le puede agregar un título (argumento `main`) y el argumento `outline` especificado como `TRUE`, para que si existen casos atípicos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

### Ejercicio 28

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



## Gráfico de cajas 1



## 8.2. Introducción al uso del paquete *ggplot2*

### 8.2.1. 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 es preciso 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.<sup>29</sup>

### 8.2.2. Gramática del paquete *ggplot2*

El paquete *ggplot2* presenta distintas características que lo distinguen:

1. 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,
2. 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.
3. 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, pp. 121-136) 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, al usar la función *ggplot2* 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, & 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.

---

<sup>29</sup>Una buena fuente de información, que incluye ejemplos interesantes de visualización de datos junto con sus respectivas sintaxis de configuración puede encontrarse en el apartado *ggplot2* de la [Galería de Gráficos de R](#). Recomendamos explorar este tipo de recursos cuando se desee implementar una visualización cuyo código no se maneje.

Cuadro 2: Elementos que componen un gráfico construido mediante la función *ggplot*.

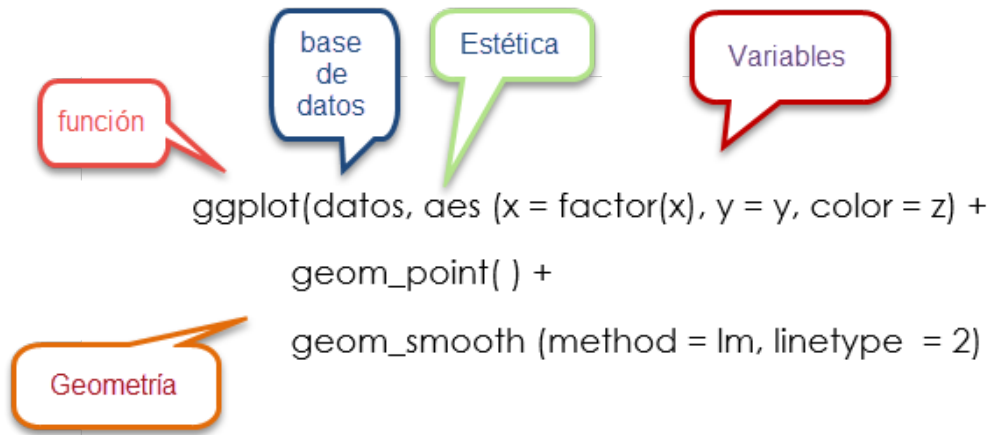
Capa	Descripción
<b>Datos</b>	Conjunto de información que se representará de manera gráfica. En nuestro caso se trata de una o más variables, o una base de datos.
<b>Estética</b>	Escala en la cual se mapeará la información de interés. Refiere al posicionamiento de la información a representar sobre los diferentes ejes y dimensiones del gráfico resultante. Hablamos del posicionamiento de variables en los ejes X e Y, indicar variables que pueden ser posicionadas como color de relleno dentro de los diferentes ejes, etc.
<b>Geometría</b>	Formas, elementos visuales, que se emplearán para representar visualmente la información ya consignada en los <i>datos</i> y ubicada en las diferentes posiciones del gráfico mencionadas en la <i>estética</i> . <sup>30</sup> 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 +.**

<sup>30</sup>Internamente, esta capa también puede contener propiedades *estéticas*, como la la transparencia, el color o tamaño de los objetos geométricos utilizados. Tal información *estética* interna a la geometría, no debe confundirse con la información *estética* general del gráfico.

Imagen 50: Estructura básica de una sintaxis de la función `ggplot`



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*, *geoms*) 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.<sup>31</sup>

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

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*.<sup>32</sup> 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 subtítulo general para el gráfico (*labs*).

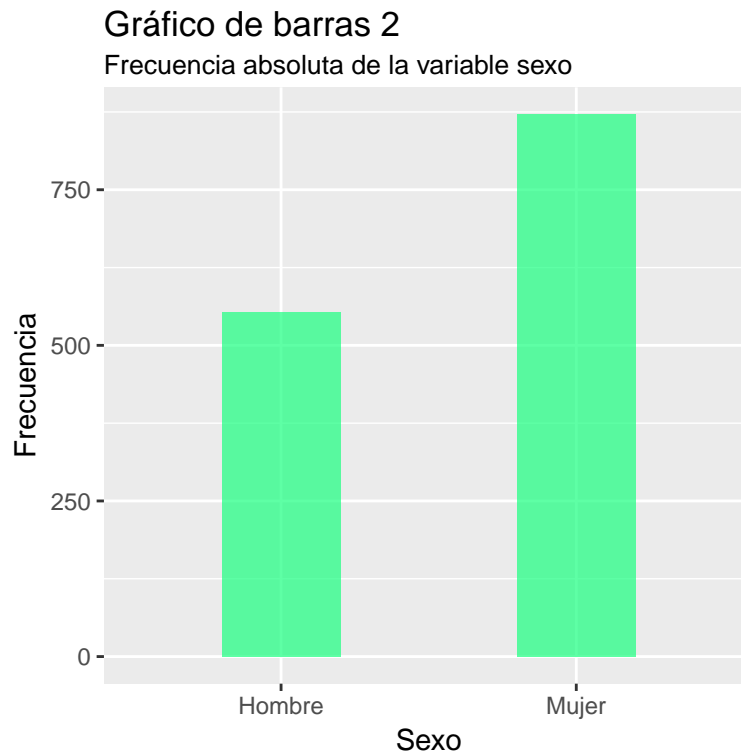
#### Ejercicio 29.1

```
library(ggplot2)  
#Gráfico de barras 2: sexo en frecuencias absolutas  
ggplot(CEP, aes(x = sexo_factor)) +
```

<sup>31</sup>Para contar con un listado explicativo completo de las configuraciones posibles para la función *ggplot*, sugerimos revisar la [Guía de Referencia para ggplot2](#) elaborada y puesta en línea por su equipo de desarrollo. Desde esta página se puede acceder a la [guía completa en línea de la familia de paquetes tidyverse](#), que incluye documentación de los siguientes paquetes: dplyr, tidyr, tibble, purr, readr y, por supuesto, ggplot2.

<sup>32</sup>Ver nota 28 para materiales explicativos de las diferentes configuraciones de colores que funcionan en R. En este caso, se trata de la función *rgb*, que define concentraciones de los colores básicos rojo, verde y azul (*red*, *green* and *blue*).

```
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")
```

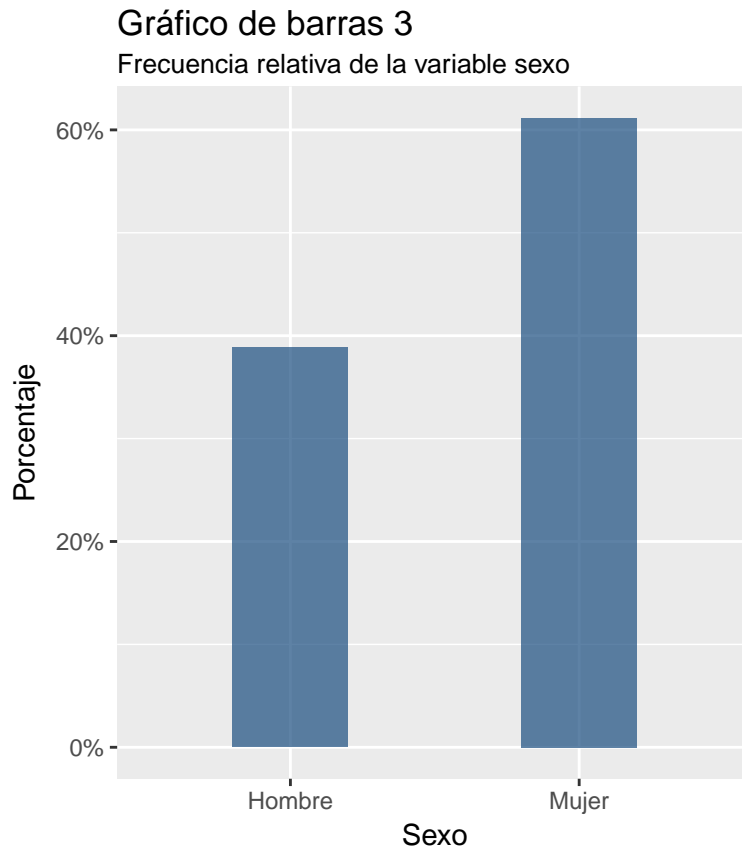


Si bien se trata de un código algo más complejo que la función básica que incluye el software, es posible afirmar que hay una mejora estética notable en comparación con los gráficos más básicos.

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 29.2

```
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:

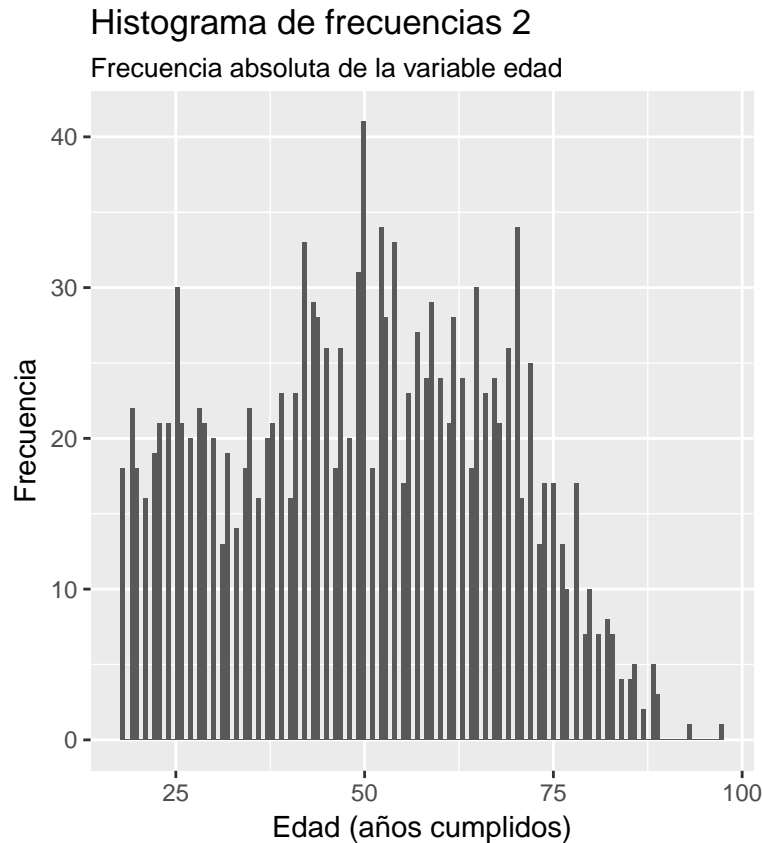
- Se modificaron algunos valores del argumento *fill* de la función *geom\_bar* para alterar el color resultante de las barras.
- En la función *geom\_bar* se agregó un argumento *aes* para editar el tipo de conteo ejecutado en el eje Y. Específicamente se le indica que el conteo hecho en Y es igual al conteo simple, dividido en la suma total de casos.
- Luego, se agrega un *scale\_y\_continuous* para 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 29.3

```
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")
```



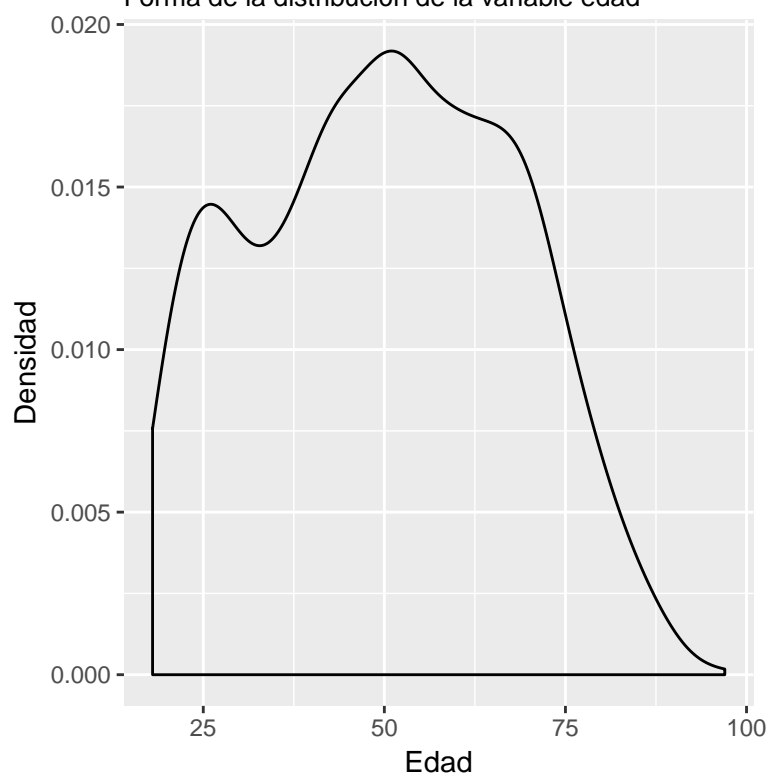
Como fue señalado en el apartado anterior la manera que resulta más óptima para observar la forma de la distribución de una variable es la construcción de un histograma de densidad. A continuación se muestra la sintaxis que permite construir un histograma de densidad de la variable *Edad*. La estructura 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 29.4

```
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





## 9. Introducción al uso de RMarkdown para la compilación de resultados de RStudio en diferentes formatos

### 9.1. ¿Qué es RMarkdown y para qué sirve?

Por su amplio y fácil uso, las y los estudiantes, y posteriormente las y los académicos e investigadores en Ciencias Sociales, generalmente utilizan Microsoft Word para construir sus reportes de investigación.<sup>33</sup> Si bien es un software fácil de utilizar presenta algunas limitaciones en el manejo de sus atributos estéticos y de contenido. Por una parte es difícil controlar las ediciones de su formato de presentación pues es bastante engorroso para dar formato a documentos extensos: obliga a invertir mucho tiempo haciendo clicks, indicando formatos que se pueden descalibrar al pasar a otro formato de texto o versión de software. Por otra parte, cualquier contenido que no sea texto resulta difícil de incorporar pues se generan incompatibilidades según las versiones del programa, a la vez que no se articula de manera sencilla con otras fuentes de información: por ejemplo los resultados construidos a partir de un software de análisis estadístico.

A estas limitaciones, que han sido señaladas por otros autores (Miller, 2018) aquí se agrega otra. La interacción entre un software para análisis estadístico como R y Microsoft Word, es quizá una de las que presenta más complicaciones. Como ya fue visto en el capítulo 7 de este documento, para incorporar resultados de formato R a Microsoft Word se debe construir un tipo de datos *ad hoc* (matrices de datos) para poder *grabarlos* en un documento tipo planilla de cálculo. Recién desde allí se podrá editar el formato de tales resultados y copiarlos a Microsoft Word (por ejemplo, a un trabajo universitario, un reporte de investigación para consultoría, un borrador de un artículo académico o documento de trabajo, un libro, etc.)

¿Cuáles son, entonces, las características de RMarkdown que lo diferencian de esta forma de trabajo? En términos sencillos RMarkdown es un *procesador de texto* que ofrece además la posibilidad de incluir *trozos de código* desde R (u otros formatos). El principal beneficio de esta herramienta es que permite trabajar en un sólo documento tanto la *redacción* del contenido narrativo de reportes de investigación, como también la construcción y *presentación formal de resultados de análisis estadísticos*.<sup>34</sup>

Se distinguen entonces dos paradigmas de trabajo en los procesos de construcción de los diferentes formatos de presentación de resultados de investigación. Por un lado un *paradigma no integrado* en la construcción de informes de investigación: en este formato, el autor generalmente construye por separado los elementos de un reporte de resultados (texto, tablas de resultados estadísticos, gráficos, formato general del documento, referencias y listado de bibliografía utilizada). Por ejemplo, si construye una tabla de datos en cualquier software de análisis estadístico, luego la debe copiar y pegar en su reporte definitivo, editando allí su formato final. Si los datos o el sentido del análisis cambia, el autor debe repetir todo el proceso para actualizar la información su informe final. Por otro lado, en un *paradigma integrado* para la construcción de informes, como el que subyace a RMarkdown, existe un sólo lugar (archivo *RMarkdown*) donde se edita tanto el texto del reporte, como los códigos para construir los resultados a incluir, así como el formato general del documento y la gestión bibliográfica. Si el documento debe actualizarse, se efectúan en un sólo lugar todos los cambios y se re-compila el informe en el formato final deseado (sea PDF, Editor de Texto tipo Microsoft Word, diapositivas de presentación o HTML) (Grolemund, 2014).

Ahora bien, ¿cómo funciona esta herramienta? Crear reportes desde RMarkdown combina diferentes aplicaciones computacionales. En tal sentido, es de las operaciones de uso de R que más precisa de la correcta instalación de una serie de elementos adicionales. En una breve síntesis, la construcción de informes de resultados desde R demanda la ejecución simultánea de cuatro elementos (Grolemund, 2014):

1. **RMarkdown.** Basado en el lenguaje *markdown* - funcionalidad que busca convertir rápida y fácilmente texto plano tipo bloc de notas a formato HTML - RMarkdown es un tipo de documento de RStudio que permite integrar texto con código de R.

<sup>33</sup>Debido a que es ampliamente utilizado se hace referencia al procesador de texto de Microsoft Office. Estas anotaciones también resultan aplicables a otros paquetes de software de oficina ya nombrados en este documento como Open Office y Libre Office.

<sup>34</sup>Para información de carácter general sobre RMarkdown como plataforma de trabajo (fichas técnicas, manuales, plantillas y tips de trabajo) sugerimos visitar la [página oficial del proyecto](#).

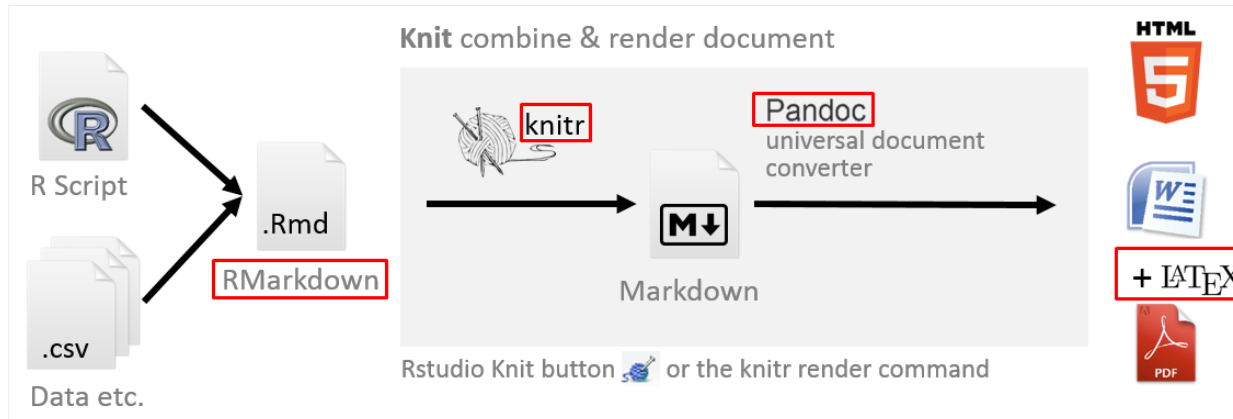
2. **Knitr**. Este paquete integra en un sólo formato *markdown* el texto ingresado en formato markdown y los resultados de la ejecución de los códigos construidos mediante R.
3. **Conversor a formato final**:
  - **Pandoc**. Se trata de un paquete de R que convierte el formato *markdown* a alguno de los diferentes formatos de reporte ya señalados (HTML y editor de texto tipo Word).
  - **LaTeX**. Es una aplicación computacional en sí misma, enfocada en la *preparación de documentos* para su publicación con una alta calidad profesional del formato final. Está pensado para ser utilizado en procesos editoriales de alta complejidad y exigencia de calidad. Permite convertir los documentos *markdown* a PDF (Navarro, 2014).<sup>35</sup>

Todos estos elementos se ponen en juego en un flujo de trabajo (*workflow*) que, luego de una corta espera, generará un documento final con una elevada calidad final en su presentación de resultados. En la imagen a continuación (Workshop, 2016), se grafica tal flujo.

---

<sup>35</sup>Es importante señalar que *RMarkdown* simplifica el lenguaje LaTeX, haciendo más intuitivo su uso para un usuario que no está enfocado en la utilización profesional de sus prestaciones. Desde este lenguaje viene la impronta asimilada por RMarkdown de separar el **contenido** del **formato** de un documento .

Imagen 51: Elementos necesarios para construir reportes con RMarkdown



Para ejecutar correctamente un documento de RMarkdown y lograr construir un reporte final en el formato deseado, es preciso asegurar que todos estos elementos estén instalados en el computador. A continuación se presenta una breve tabla que resume los elementos a instalar y cómo hacerlo.

Cuadro 3: Elementos necesarios para ejecución de RMarkdown: procedimientos de instalación

Elemento	Procedimiento de instalación	Código para instalación
<b>RMarkdown</b>	No es necesario ningún procedimiento adicional a la instalación de R y RStudio pues viene instalado con este último.	<i>No aplica</i>
<b>Knitr</b>	Debe instalarse como cualquier otro paquete de R, asegurando su disponibilidad para ser utilizado por RMarkdown al compilar documentos.	<code>install.packages("knitr")</code>
<b>Pandoc</b>	No es necesario ningún procedimiento adicional a la instalación de R y RStudio pues viene instalado con este último.	<i>No aplica</i>
<b>LaTeX</b>	Debe descargarse como un paquete de R. Para asegurar su disponibilidad para ser utilizado por RMarkdown al compilar documentos, debe instalarse con un comando adicional. <sup>36</sup>	<code>install.packages("tinytex")</code> <code>tinytex::install_tinytex()</code>

En síntesis: antes de proceder a la compilación de un documento RMarkdown, debemos asegurar la instalación de estos componentes en nuestros computadores.

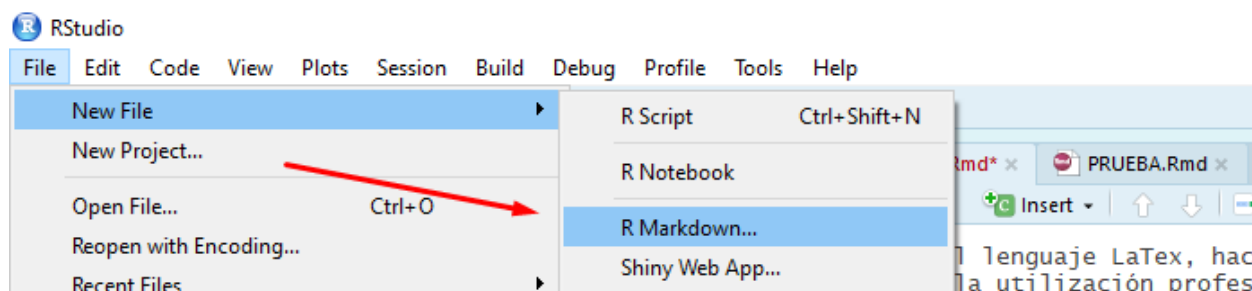
<sup>36</sup>Existen diferentes versiones de LaTeX, de descarga gratuita. Sin embargo, estas distribuciones de LaTeX (por ejemplo [MikTeX](#) es una de las más conocidas para sistema operativo Windows) presentan diversas limitaciones y complicaciones de configuración que hacen difícil su uso integrado con RMarkdown. Por eso se sugiere usar TinyTex, paquete diseñado para disminuir tales complicaciones y facilitar la integración de RMarkdown con las funcionalidades de LaTeX. Para mayores detalles se recomienda revisar la [página web](#) de su desarrollador.

## 9.2. Los diferentes elementos de una sintaxis de RMarkdown

En este apartado se indican las nociones generales a tener en cuenta para trabajar con una sintaxis de RMarkdown.

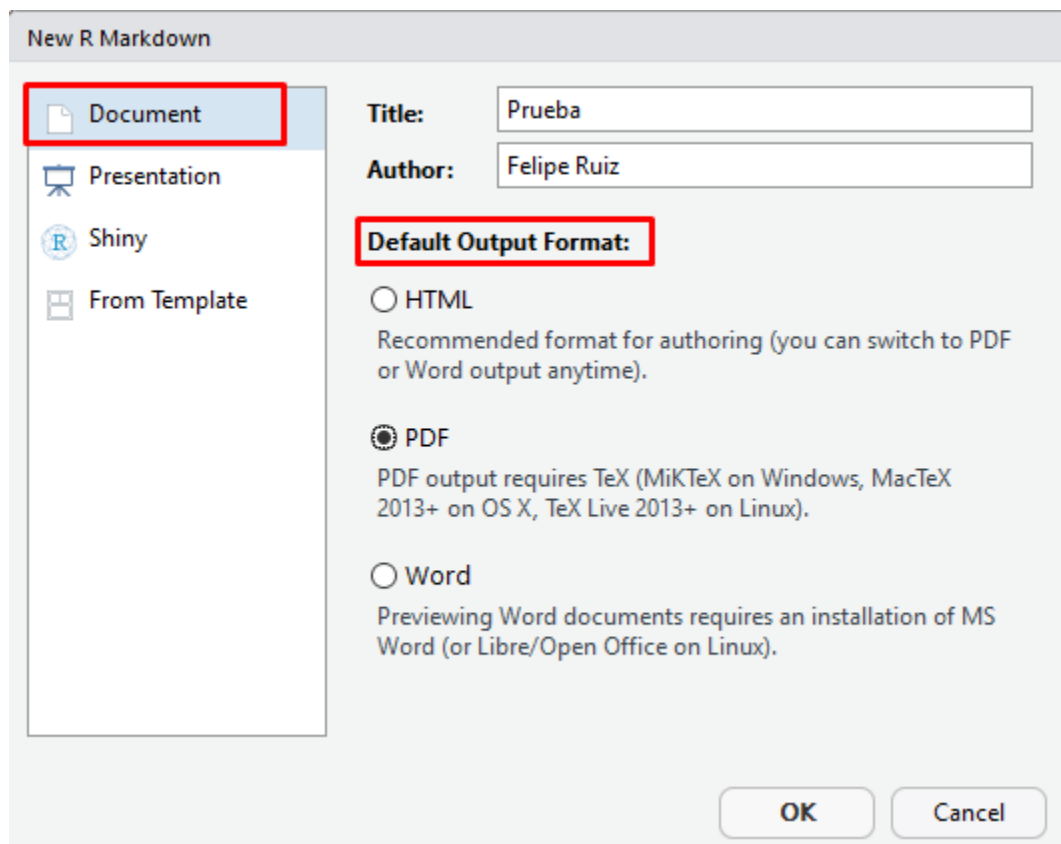
El procedimiento básico para abrir una sintaxis de RMarkdown es similar al usado para una sintaxis de R. Es posible crear un nuevo documento de RMarkdown usando la botonera superior de RStudio. Como se observa a continuación, yendo a *Archivo*, *Nuevo Archivo* y seleccionando la opción *RMarkdown*.

Imagen 52: Apertura de nuevo documento Rmarkdown vía botonera



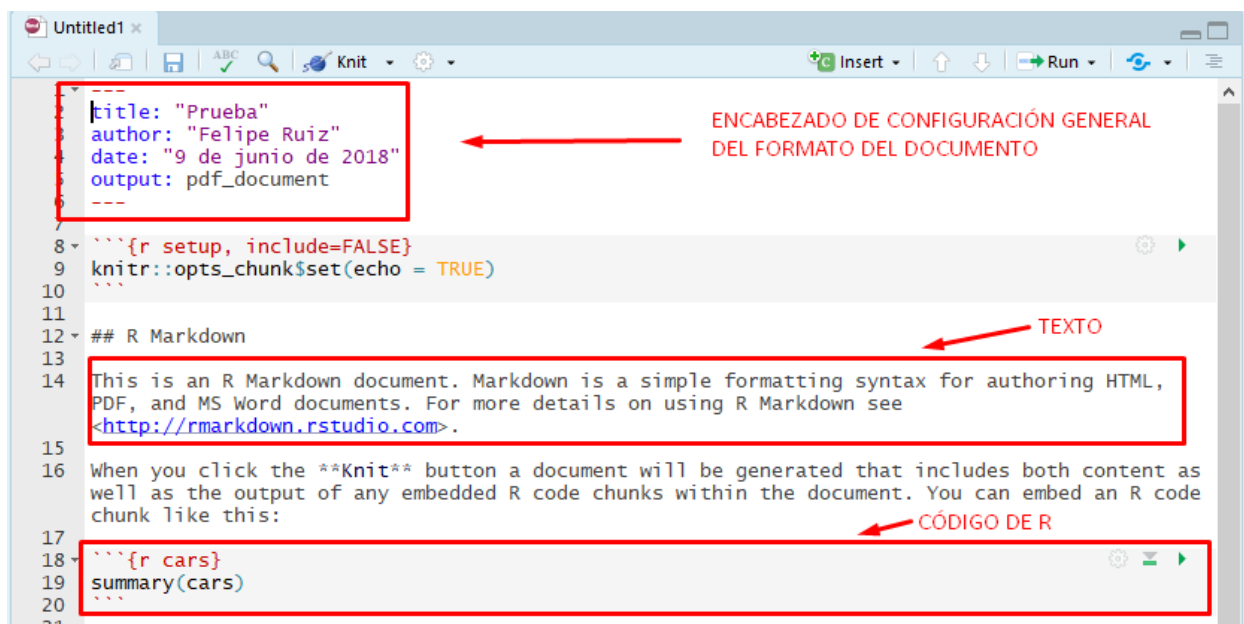
Una vez hecho tal procedimiento aparecerá una nueva pantalla de configuración general. Aquí se establece la configuración básica del reporte de resultados, indicando un título para el documento, el nombre del autor, así como el formato de informe a construir (en este ejemplo se selecciona el formato *PDF*).

Imagen 53: Ventana de preconfiguración del documento de RMarkdown



Si se especifican tales condiciones, luego de apretar *aceptar* se desplegará una nueva pestaña en R (con extensión *.Rmd*) que será la sintaxis de RMarkdown en la cual se editará el reporte de investigación. Como se observa en la imagen a continuación RStudio ofrece una sintaxis de RMarkdown preconfigurada.

Imagen 54: Sintaxis Rmarkdown con preconfiguración



De esta sintaxis conviene poner atención sobre tres tipos de elementos que resultan distintivos del trabajo con RMarkdown.

1. **Encabezado general de formato.** Al inicio del documento, y encerrado por tres guiones continuados, por arriba y por abajo de manera respectiva, se encuentra el encabezado que permite establecer los parámetros generales de formato que estructurarán el producto final. En la imagen destacada se observan los campos (en inglés): *título del documento*, *autor del documento*, *fecha* y *formato*.
2. **Texto.** A diferencia de una sintaxis de R, RMarkdown considera como formato primario de redacción el texto plano. Todo lo que se escriba - a no ser que se indique lo contrario - será considerado como texto en el documento final.
3. **Trozos de código de R.** Para indicar que se escribirá un código de R, las líneas de comando deben encerrarse entre tres cremillas seguidas de corchetes curvos con la letra *r* en su interior; luego de eso se escriben los comandos de R; al finalizar los comandos, se agrega una última línea con tres cremillas para indicar el cierre del trozo de código.

Si el documento de RMarkdown construido por defecto al abrir uno nuevo se compila en formato PDF, el inicio de este documento se verá como sigue. Se observa que los campos puestos en el encabezado han pasado a ser parte del título general del documento. A la vez, se distingue el texto plano del código de R que, en este caso, integra en el documento el resultado de una función *summary*:

# Prueba

*Felipe Ruiz*

*9 de junio de 2018*

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0   Min.   : 2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

En los siguientes apartados se explicará la configuración específica de diferentes elementos de formato básico para la edición de cualquier informe de resultados vía RMarkdown. Se ilustrará la visualización básica de los diferentes elementos indicados en RMarkdown. Todos los elementos indicados pueden ser encontrados de manera unificada en el [siguiente enlace](#), donde nuestros lectores encontrarán una plantilla básica de RMarkdown disponible para descarga, así como el documento PDF que resulta de su ejecución.

## 9.3. Aspectos generales (formato de texto)

### 9.3.1. Formato general: encabezado YAML

Este encabezado determina los parámetros generales de formato para el reporte a compilar. En lenguaje computacional se entiende como los *metadatos* del documento; esto es, información que define el formato del archivo resultante, más no su contenido. En el ejemplo que aquí presentamos se observan las siguientes definiciones:

- **Título** (*title*). Texto entre comillas que servirá de título general al documento.
- **Subtítulo** (*subtitle*). Texto entre comillas que servirá de subtítulo para el título general del documento.
- **Autor** (*author*). Texto entre comillas para indicar el nombre del o los autores.
- **Fecha** (*date*). Campo para indicar la fecha. En el texto, con la expresión *today*, se solicita que imprima la fecha actual según el calendario del sistema operativo.
- **Bibliografía** (*bibliography*). Se indica el nombre del archivo que contiene los datos para construir el listado de referencia bibliográficas. Como será explicado más adelante, este archivo es de formato BibTeX y se construye usando un gestor de referencias como *Zotero*.
- **Formato de bibliografía** (*csl*). Nombre de un archivo de extensión *.csl* para indicar el formato de referencias en el cuerpo del texto y el listado de bibliografía al final del documento. La sigla refiere a *Estilo del Lenguaje de Referencias* (*Citation Language Style*) y permite definir si se usarán citas al estilo APA, ASA, Chicago, etc. Se adecúa a los diferentes requerimientos de referencias bibliográficas.

- **Color de los enlaces.** Mediante las opciones *linkcolor* y *urlcolor* se define el color asignado al indicador de notas al pie y a los enlaces a páginas web, respectivamente. En este caso se sugiere el color azul (*blue*).
- Los argumentos dentro del apartado *resultado* (*output*) son los siguientes:
  - **pdf\_document:** indica el formato preestablecido para compilar el documento. En este caso, se trata de un PDF. Puede ser *html\_document* o *word\_document*. El usuario puede escoger la modalidad que desee al compilar usando las opciones del botón *knit*; si se compila sin escoger ninguna opción, se compilará según el formato indicado en este encabezado.
  - **fig\_caption:** indica si las figuras deben incorporar leyendas.
  - **latex\_engine:** permite definir el motor de LaTeX utilizado para compilar los documentos.
  - **number\_sections:** si está definido como *yes* define que se numerarán los títulos y subtítulos a lo largo del documento, de manera automática y correlativa.
  - **toc:** es la abreviación de *table of contents*; si está definido como *yes* incorporará al inicio del documento una tabla de contenidos construida a partir de los tres primeros niveles de los títulos y subtítulos de sección utilizados.

### Ejercicio 30.1

```

---
title: "Ejemplo Uso de RMarkdown"
subtitle: "Material de apoyo docente, asignatura Estadística Descriptiva - Semestre de otoño 2018"
author: "Giorgio Boccardo Bosoni y Felipe Ruiz Bruzzone"
date: "\today"
bibliography: bibliografia.bib
csl: apa.csl
linkcolor: blue
urlcolor: blue
output:
  pdf_document:
    fig_caption: yes
    latex_engine: xelatex
    number_sections: yes
    toc: yes
---

```

En definitiva, todos estos elementos configuran una importante información para indicar el formato general del documento que se busca compilar. No se trata específicamente del contenido a utilizar pero es, por así decirlo, toda la información de “contexto” que el software necesita manejar para construir un reporte final con el formato deseado.

### 9.3.2. Títulos, subtítulos y saltos de página

En el caso de RMarkdown los signos “gato” (*#*) indican que se está señalando un título o subtítulo de sección. Mientras más signos gatos se pongan se estará indicando que se trata de un título de menor jerarquía. Por otra parte, con el comando *pagebreak* se pueden indicar saltos de página. Esto último sirve, por ejemplo, para comenzar cada sección del documento en una página nueva.

### Ejercicio 30.2

```

# Título 1 (nivel de mayor jerarquía)
## Título 2 (nivel de segunda jerarquía)
### Título 3 (nivel de tercera jerarquía)

```

```
\pagebreak #Salto de página.
```

Si se utilizan tales instrucciones el documento final irá adquiriendo la siguiente forma:

## Imagen 56: Compilación de Rmarkdown con índice, títulos y subtítulos

### Contents

1	Ejemplo de Título 1	1
1.1	Ejemplo de Título 2 . . . . .	1

## 1 Ejemplo de Título 1

El texto antecedido de signo gato genera un título. El texto simple, indica cuerpo de texto para el documento. Al compilar el documento, el programa utiliza estas expresiones para construir la tabla de contenidos al inicio del documento,

### 1.1 Ejemplo de Título 2

Adicionando signos gatos, se construyen títulos de menor jerarquía. Por tanto, funcionan como subtítulos para la estructura interna de cada capítulo.

#### 1.1.1 Ejemplo de Título 3

### 9.3.3. Énfasis de texto

Otro elemento importante son los distintas marcas que permiten introducir diferentes énfasis en el texto. A lo largo del texto se pueden introducir las clásicas configuraciones de formato que, en los procesadores de texto más utilizados, se configuran mediante la selección del texto con el cursor y la utilización de uno o más botones. A continuación se muestra cómo se indican tales configuraciones en el texto plano (sin compilar):

### Ejercicio 30.3

```
**negrita**
```

```
*cursiva*
```

```
_subrayado_
```

```
> Texto con un tabulado mayor al párrafo normal.
```

```
Texto que está hablando de un tema y quiere poner una nota al pie [1].
```

```
[1]: texto de la nota al pie
```

```
Para ingresar un enlace asociado a una palabra, se debe encerrar la palabra o palabras  
que [queremos sea un enlace](www.url.com)
```



Considerando tales elementos, a continuación se presenta un texto con sus especificaciones de énfasis en formato RMarkdown y a continuación su resultado luego de la compilación.

### Ejercicio 30.4

Así, si se aplica lo recién anotado, podemos configurar palabras en **negrita** y en *cursiva*.

> Podemos incluir un párrafo con un tabulado mayor al del párrafo regular.

Asimismo, en cualquier parte del documento podemos incluir una nota al pie [<sup>1</sup>].

[<sup>1</sup>]: texto de la nota al pie. En cualquier parte del documento, aquí lo haremos en esta nota al pie, p

### Imagen 57: Compilación de documento RMarkdown con énfasis de texto

A lo largo del texto se pueden introducir las clásicas configuraciones de formato que, en los procesadores de texto más utilizados, se configuran mediante la selección del texto con el cursor y la utilización de uno o más botones. A continuación se muestra cómo se ven tales configuraciones en el texto plano (sin compilar):

Así, si aplicamos lo recién anotado, podemos configurar palabras en **negrita** y en *cursiva*.

Podemos incluir un párrafo con un tabulado mayor al del párrafo regular.

Asimismo, en cualquier parte del documento podemos incluir una nota al pie <sup>1</sup>.

---

<sup>1</sup>texto de la nota al pie. En cualquier parte del documento, aquí lo haremos en esta nota al pie, podemos definir enlaces que redireccione a páginas web.

### 9.3.4. Listas

El proceso de construcción de listas de elementos en RMarkdown es muy sencillo. Pueden ocuparse números, para listas numeradas o signos asterisco (\*) o signos más (+), para definir listas no numeradas.

### Ejercicio 30.5

*#Construcción de una lista numerada*

1. Elemento 1
2. Elemento 2

```
#Construcción de una lista sin números

* Elemento 1
* Elemento 2

#Lista sin números, con sublista numerada (tabulado es de 4 espacios)

+ Elemento 1
+ Elemento 2
  1. Sub elemento 1.
  2. Sub elemento2.
```

Al compilar los elementos recién expuestos, se obtiene el siguiente resultado:

#### Imagen 58: Compilación de documento RMarkdown con listas

##### Construcción de una lista numerada

1. Elemento 1
2. Elemento 2

##### Construcción de una lista sin números

- Elemento 1
- Elemento 2

##### Lista sin números, con sublista numerada

- Elemento 1
- Elemento 2
  1. Sub elemento 1.
  2. Sub elemento2.

Para una sublista el tabulado se consigue con 4 espacios



### 9.3.5. Tablas

La construcción de tablas en RMarkdown se realiza separando los elementos de cada columna con un tabulador vertical (|). Cada fila de texto corresponderá a una fila de la tabla. Se puede definir la alineación del texto según la distribución de guiones debajo de los títulos. Si los guiones exceden hacia ambos lados del texto, este quedará centrado. Si excede hacia uno u otro lado quedará alineado hacia la izquierda o la derecha. Las reglas para destacar texto, incluir enlaces y notas al pie, también aplican las tablas. No es necesario separar cada fila con guiones, basta comenzar una nueva fila de texto.

#### Ejercicio 31.1

Table: Tabla simple

```
**Título Columna 1** | **Título columna 2** |
-----|-----|
*Texto 1* | Texto 2
*Texto 3* | Texto 4
```

A continuación se ve el resultado de la ejecución del comando anterior resultando una tabla simple en formato RMarkdown.

Cuadro 4: Tabla simple

Título	Columna 1	Título columna 2
	Texto 1	Texto 2
	Texto 3	Texto 4

### 9.3.6. Bibliografía

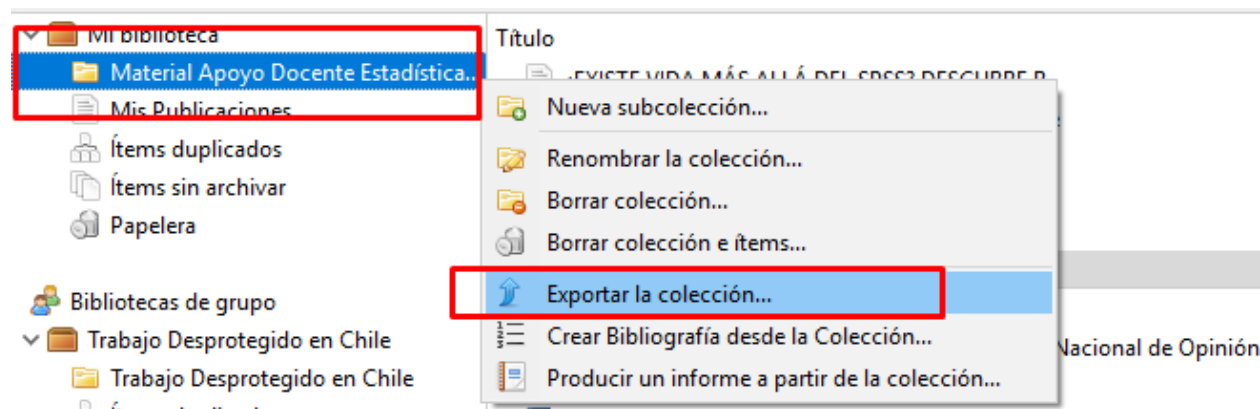
Este apartado asume que quien lee este documento tiene experiencia con gestores electrónicos de referencias bibliográficas como Zotero. Por lo tanto no se profundizará en explicaciones sobre cómo utilizarlo: se sugiere revisar [esta documentación](#) preparada por la unidad de *Información y Bibliotecas* de la Univeridad de Chile, en torno al uso del software *Zotero* como gestor de referencias.

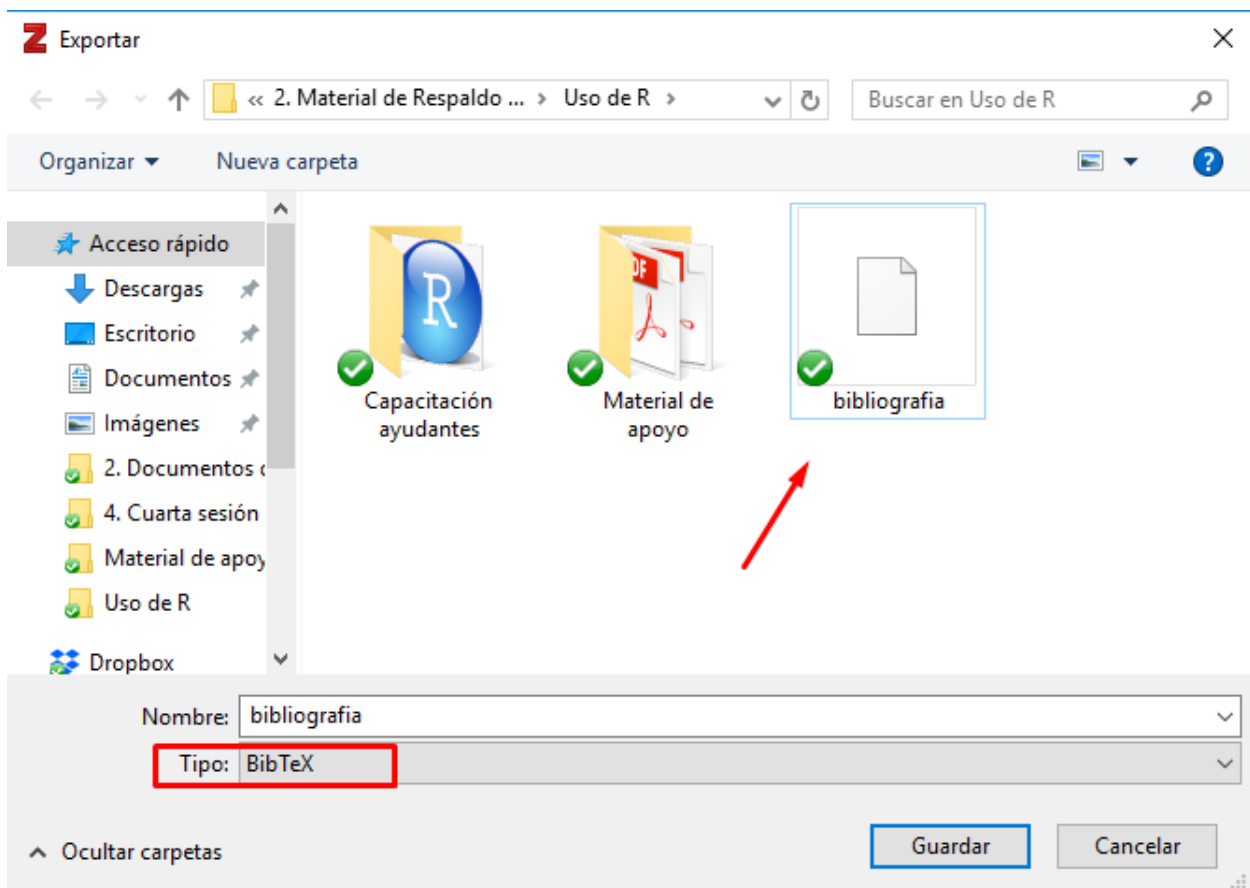
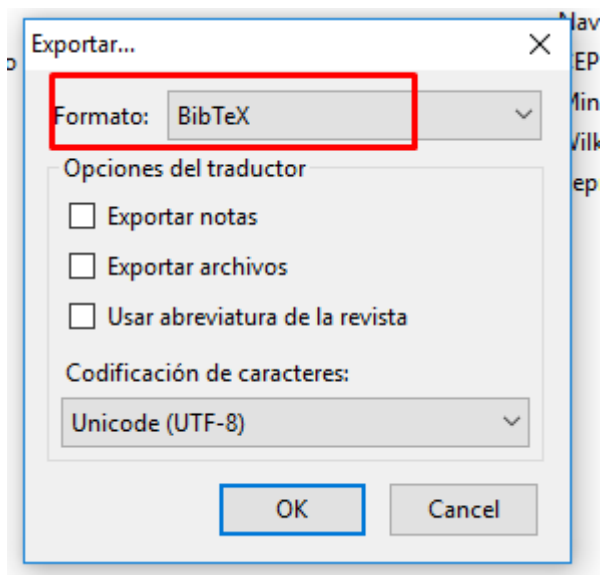
Para incorporar una bibliografía formal a un reporte, tanto en lo que respecta a referencias en el cuerpo del texto como para construir un listado de referencias bibliográficas utilizadas al final del documento, se deben manejar dos elementos:

1. *Zotero* y *Zotero connector*. Mediante este software se crea un listado bibliográfico compatible con LaTeX (archivo *.bib*) que - cargado a RMarkdown - permitirá insertar referencias y que luego se construya un listado bibliográfico. Ambos pueden descargarse desde [este enlace](#).
2. Contar con un archivo *.csl* para indicarle el formato de cita al documento (*Citation Language Style*). Se puede descargar el adecuado para formato APA 6a Edición desde [esta página](#). También se pueden descargar otros formatos como *Vancouver*, *Chicago* o *ASA* (*American Sociological Asociation*).

El primer elemento se construye exportando un listado de bibliografía desde Zotero al formato BibTex. Como se observa en la imagen a continuación es una exportación simple. Se maneja haciendo click derecho sobre el conjunto de referencias bibliográficas que interesa, seleccionando la opción *exportar*. En la siguiente pestaña se selecciona el formato *BibTex* para luego introducir el nombre del archivo de extensión (.bib) a guardar. Importa guardarlo en la misma carpeta que funcione como carpeta de trabajo para el documento RMarkdown.

Imagen 59: Construcción de archivo de referencia bibliográficas usando Zotero





El segundo elemento (archivo *CSL*) se puede descargar desde la página web de Zotero. Como se observa en las imágenes a continuación, mediante el buscador puede indicarse el estilo deseado (en este caso, APA). Luego se descarga la versión de formato de referencias que se adecúe al requerimiento específico. Para este material se utiliza la 6ta versión del formato APA de referencias bibliográficas, que además está recientemente actualizada. Tal archivo también debe quedar guardado en la carpeta de trabajo del documento RMarkdown.

Imagen 60: Descarga de archivo CSL para formato de referencias

## Zotero Style Repository

Here you can find [Citation Style Language](#) 1.0.1 citation styles for use with [Zotero](#) and other CSL 1.0.1-compatible software. [Zotero wiki](#).

**Style Search**

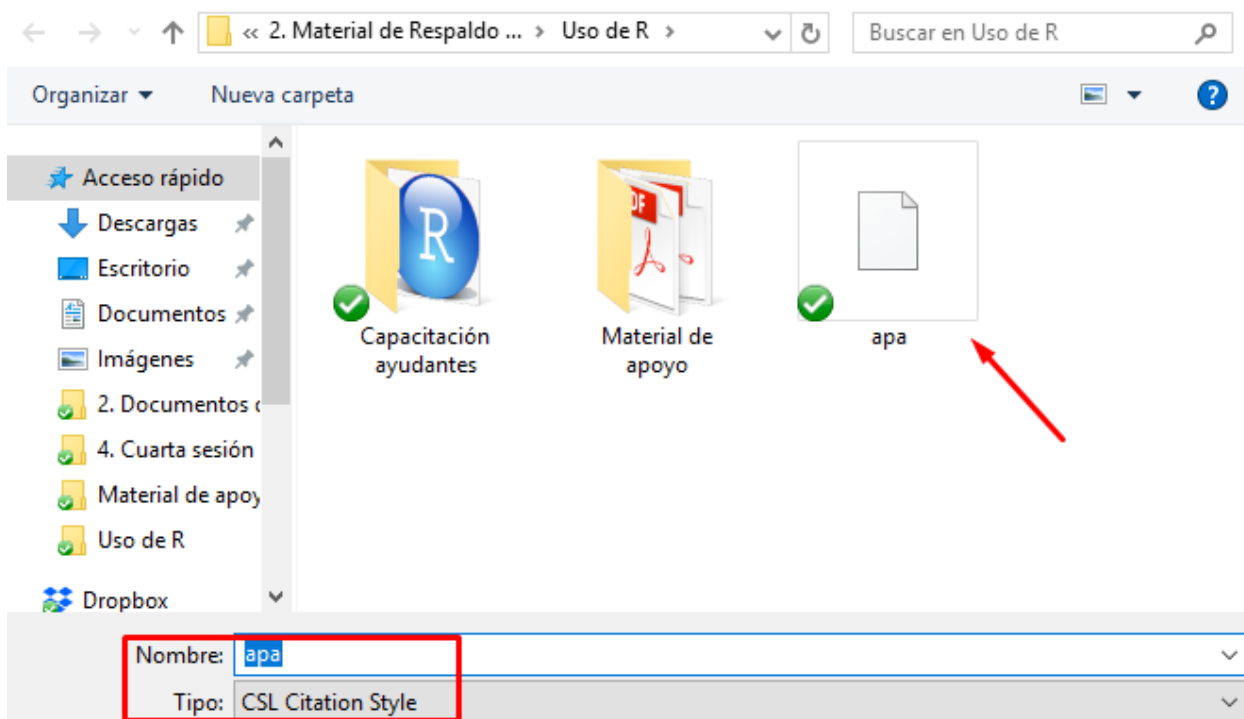
Format:

Fields:

☐ Show only unique styles

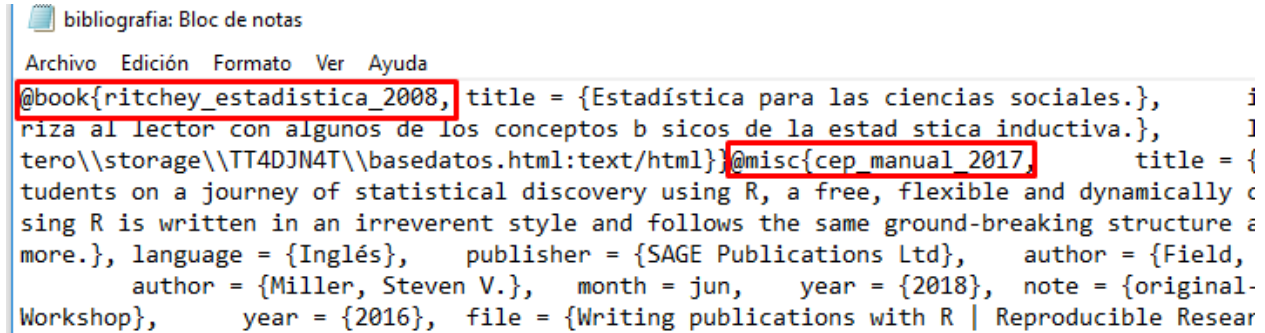
33 styles found:

- [American Psychological Association 5th edition](#) (2014-09-06 22:02:33)
- [American Psychological Association 6th edition](#) (2018-05-22 00:59:23)



En el encabezado del archivo RMarkdown se deben indicar los nombres de estos dos archivos para poder construir de forma adecuada las referencias bibliográficas. Ahora bien, ¿cómo agregar las referencias en el cuerpo del texto? Para determinar cómo se recomienda abrir con el *bloc de notas* el archivo *.bib* donde está el listado bibliográfico.

Imagen 61: Estructura interna del archivo CSL



Si se observa la estructura interna del archivo CSL. Se notará que a lo largo del texto se van definiendo los diferentes campos de información que tiene cada referencia bibliográfica (autor, año, título, edición, tipo de referencia, etc.). Así, hay que encontrar el *identificador de la referencia*, luego del signo arroba (@). Abriendo un paréntesis de corchetes en el cuerpo del documento e incluyendo tal texto luego de una arroba, se incorporará una (Elousa, 2009) o más referencias bibliográficas (Elousa, 2009, p. 22; Grolemond, 2014).

A continuación se observa el texto plano que define las referencias recién escritas:

## Ejercicio 32

Abriendo un paréntesis de corchetes en el cuerpo del documento e incluyendo tal texto luego de una arroba, se incorporará una [Elousa\_existe\_2009] o más referencias bibliográficas [Elousa\_existe\_2009, 22; Grolemond\_introduction\_2014].

La bibliografía siempre se compila al final del documento. Eso significa que estará al final del último elemento escrito. El listado bibliográfico final se contruirá siempre con base a las citas realizadas en el cuerpo del texto y según todas las referencias presentes en el archivo de referencias exportado desde Zotero. Podemos asegurarnos de que el último título del documento sea algo como *Referencias bibliográficas*, para que tal contenido tenga un encabezado apropiado.

## 9.4. Aspectos generales (configuración de trozos de código)

El tercer elemento de importancia (además del *encabezado de configuración* y del *texto* propiamente tal) son los *trozos de código de R* o *code chunks* (en inglés). Se trata de bloques de código delimitados por la siguiente estructura: el inicio de un código está delimitado por tres apóstrofes seguidos por un `r` entre corchetes curvos `{r}`, y su cierre por otros tres apóstrofes. Eso delimita lo que se ejecutará como código de computación, diferenciándolo respecto al texto simple. A continuación se observa un ejemplo de esto:

Imagen 62: trozo de código de R en documento RMarkdown



El título del código sirve para incorporarlo a la estructura de contenidos del archivo RMarkdown de manera similar a los títulos y subtítulos de secciones de texto. Por otra parte los argumentos sirven para configurar

el comportamiento del código al momento de compilar el documento. Como no siempre se buscará que en el reporte final se despliegue la sintaxis original, o los mensajes y/o advertencias que reporta R luego de ejecutar un comando, es posible configurar la ejecución de cada trozo de código agregando diferentes opciones.

Por ejemplo, al abrir el siguiente código de nuestra sintaxis de Markdown con las siguientes opciones  $\{r, echo = FALSE, results = 'asis', message = FALSE\}$  les estamos indicando lo siguiente:

- a. **echo = FALSE** significa que no se desplegará la sintaxis en el reporte, pero sí se ejecutará la operación y mostrarán los resultados.
- b. **results = 'asis'** indica que el resultado se exportará directamente al nuevo archivo, sin que sea configurado por *RMarkdown*. Esto es útil con funciones que formatean de manera inmediata los resultados al formato deseado.
- c. **message = FALSE** indica que no se mostrarán los mensajes de información en el informe final.

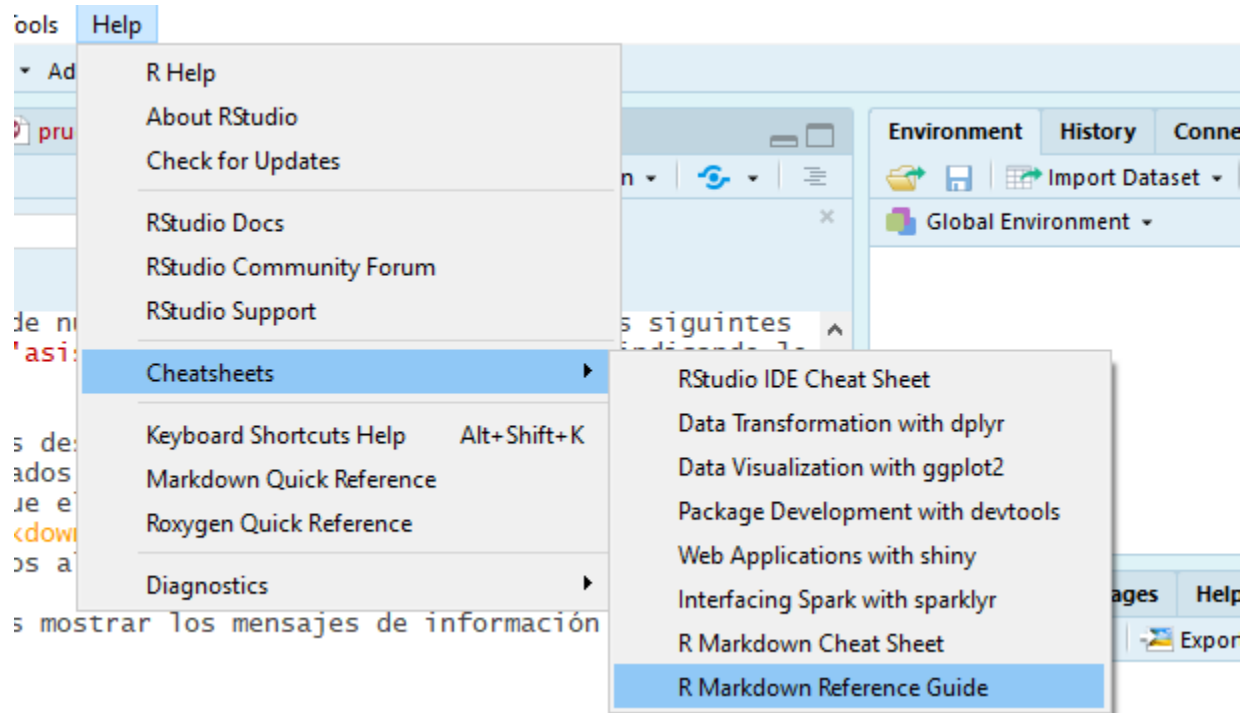
En la siguiente tabla se indican algunos de los argumentos de mayor utilidad para configurar trozos de código en RMarkdown.

Cuadro 5: Algunos argumentos para configurar los trozos de código en RMarkdown

Argumento	Valor por defecto	Detalle
<b>eval</b>	TRUE	Si se configura como FALSE, R sólo mostrará, pero no correrá el código
<b>include</b>	TRUE	Si se configura como FALSE, R no mostrará el código, pero correrá el comando y mostrará sus resultados.
<b>error</b>	TRUE	Si se configura como FALSE, R no mostrará los mensajes de errores que resulten de la ejecución del código.
<b>results</b>	—	Si se configura como <i>hide</i> , R no mostrará los resultados del código aunque lo ejecutará tras bambalinas. Si se configura como <i>delay</i> , R mostrará sólo el último resultado del trozo de código. Si se configura como <i>asis</i> R no configurará con <i>markdown</i> la estructura de los resultados, imprimiéndolos de manera directa en el reporte final. Esto resulta útil cuando usamos funciones específicas para presentar resultados en algún formato de reporte específico (Word, PDF, etc.)
<b>warning</b>	TRUE	Si se configura como FALSE R no mostrará los mensajes de advertencia que resulten de la ejecución del código.
<b>message</b>	TRUE	Si se configura como FALSE R no mostrará ningún tipo de mensaje que resulten de la ejecución del código.

Se indican sólo algunos de muchos argumentos posibles de utilizar. Para un listado completo de estas configuraciones posibles para los trozos de código, así como para la configuración y uso general de RMarkdown, se sugiere ver las diapositivas *RMarkdown Cheat Sheet* y *RMarkdown Reference Guide*. Estos materiales preparados por el equipo desarrollador de RStudio y RMarkdown resumen de manera muy sintética la mayor cantidad de argumentos y estructuras de código que pueden utilizarse en RMarkdown. Puede accederse a ello mediante la botonera de RStudio, como se muestra a continuación:

Imagen 63: Uso de material de apoyo para RMarkdown





## 9.5. Presentación de resultados básicos en RMarkdown

Habiendo indicado una parte importante de los elementos básicos de redacción de RMarkdown, a continuación se explica como incorporar en un reporte final tablas de frecuencias y de estadísticos univariados a nivel muestral.

Para incorporar esos elementos se crearán funciones incorporadas en un paquete llamado *summarytools*. Este paquete es de alta utilidad pues simplifica el proceso de construcción de tablas de resultados para análisis estadístico univariado.

La principal característica de este paquete es que su lógica pone énfasis tanto en la selección de resultados específicos a calcular como en su formato de presentación. Por eso permite controlar *qué* resultados se construyen y *cómo* son presentados (Comtois, 2018). Esto hace que las funciones de este paquete sirvan para construir resultados tanto para un **uso exploratorio** como para el diseño de su presentación en reportes de investigación, es decir un **uso orientado a la divulgación**.<sup>37</sup>

Debido a esta última característica las funciones de este paquete permiten obtener resultados en texto plano para ser visualizados en la consola de R (lo que sería el comportamiento predefinido para casi cualquier función de R), como también especificar formatos de presentación compatibles con RMarkdown para hacer posible su compilación en reportes.

### 9.5.1. Tablas de frecuencias

El primero comando a utilizar de este paquete será *freq*. Permite obtener una tabla de frecuencias absolutas y relativas para una variable de tipo nominal u ordinal. En el primer comando se utiliza el formato por defecto para la construcción de resultados en la consola de R (argumento *style = simple*). Con el argumento *justify* se indica la alineación del texto de la tabla, en este caso se configura un formato de texto **centrado** (“center”). Luego, con el argumento *omit.headings = TRUE* se indica que no se imprima un encabezado con la especificación de la tabla solicitada.

#### Ejercicio 33.1

```
library(summarytools)
freq(CEP$sexo_factor, style = "simple", justify = "center", omit.headings = TRUE)
```

```
##
##           Freq  % Valid  % Valid Cum.  % Total  % Total Cum.
## -----
##  Hombre     553    38.83    38.83      38.83    38.83
##  Mujer      871    61.17   100.00      61.17   100.00
##   <NA>         0         0.00      0.00   100.00
##  Total    1424   100.00   100.00     100.00   100.00
```

En la tabla resultante se presentan las frecuencias absolutas y relativas (expresadas como porcentaje) para la variable *sexo*, indicando también los porcentajes acumulados válidos (es decir, sin contar los NA's) y los porcentajes acumulados totales. Esta tabla es construida por defecto por la función, pero no aquella con un formato específico para RMarkdown.

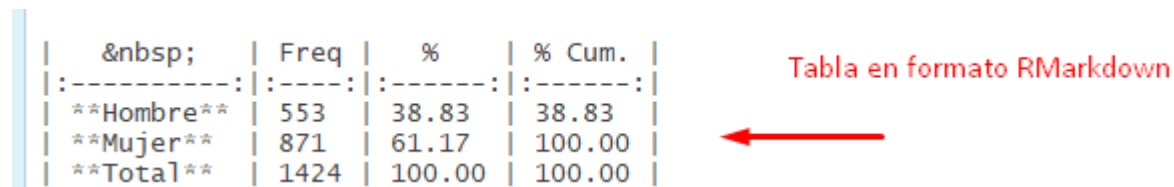
En el siguiente trozo de código de R se configura este comando para su ejecución con RMarkdown. En primer lugar, resulta la configuración específica del trozo de código: mediante el argumento *results = 'asis'* se configura que el resultado sea impreso directamente al formato RMarkdown para que la tabla resultante esté construida en la lógica de construcción de tablas que ya ha sido presentada en este documento; en

<sup>37</sup>Para una introducción (en inglés) detallada al uso general de este paquete, se recomienda la lectura del siguiente [documento](#) elaborado Dominic Comtois para el sitio oficial de R (2018)

segundo lugar, con el argumento `warning = FALSE` se especifica que en el reporte no sean impresos los mensajes de alerta que pueda arrojar la ejecución del comando. También se puede incorporar el argumento `echo = FALSE` para evitar que el trozo de código se visualice en el reporte de resultados, dejando como resultado de esta operación sólo la compilación de la tabla de frecuencias; aquí se utiliza pues interesa mostrar el trozo de código con fines pedagógicos.

En segundo lugar, destacan las configuraciones adicionales establecidas para la ejecución del código con formato adecuado a RMarkdown. En primer lugar, el *estilo* definido para la tabla se ha indicado como “rmarkdown”. Esto último permite que la tabla resultante sea como se observa en la siguiente imagen: integra los resultados a un tabulado de texto plano que permitirá contar con una tabla adecuadamente configurada al compilar el documento con RMarkdown.

**Imagen 64: Tabla de frecuencias en formato RMarkdown**



&nbsp;	Freq	%	% Cum.
Hombre	553	38.83	38.83
Mujer	871	61.17	100.00
Total	1424	100.00	100.00

Finalmente, el argumento `report.nas = FALSE` le indica al software que no incluya en la tabla el conteo de casos codificados como *NA* (perdidos), puesto que ya se sabe que esta variable no presenta casos de este tipo.

### Ejercicio 33.2

```
library(summarytools)
freq(CEP$sexo_factor, style = "rmarkdown", justify = "center", omit.headings = TRUE,
     report.nas = FALSE)
```

	Freq	%	% Cum.
Hombre	553	38.83	38.83
Mujer	871	61.17	100.00
Total	1424	100.00	100.00

Así, luego de incorporar el comando como trozo de código en un documento RMarkdown se obtiene una tabla con un diseño estético adecuado a RMarkdown que resulta apropiada como formato final para la presentación de resultados en un reporte de investigación.

#### 9.5.2. Tabla de estadísticos univariados de alcance muestral

Una segunda herramienta de utilidad del paquete *summarytools* es la función *descr* que permite obtener tablas de estadísticos univariados de nivel muestral, para variables continuas (de intervalo o razón). En el primer trozo de código se ejecuta la función en su configuración por defecto, indicando solamente argumentos adicionales para la *alineación del texto* y la presentación del *encabezado* (ya explicados para la función *freq*).

### Ejercicio 33.3

```
descr(CEP$edad, style = "simple", justify = "center", omit.headings = T)
```

	edad
Mean	49.87
Std.Dev	17.79
Min	18.00
Q1	36.00
Median	50.00
Q3	64.00
Max	97.00
MAD	20.76
IQR	28.00
CV	2.80
Skewness	0.01
SE.Skewness	0.06
Kurtosis	-0.93
N.Valid	1424.00
Pct.Valid	100.00

El resultado obtenido es una tabla vertical, con trece estadísticos univariados de tipo descriptivo. Este resultado podría configurarse para su presentación mediante RMarkdown simplemente indicando que utilizaremos el *estilo* de tabla *RMarkdown*. Sin embargo, como se observa en el siguiente trozo de código, se recomienda incorporar algunas configuraciones adicionales.

1. Indicar mediante el argumento *transpose = TRUE* que el formato final de la tabla sea construido horizontalmente; es decir, que tanto el título de cada estadístico como su valor sean ingresados como filas y no como columnas. 2. Indicar mediante un vector de valores alfabéticos el detalle específico de los estadísticos univariados descriptivos a incluir (y en qué orden) en la tabla de resultados. 3. Especificar una tabla construida en *estilo RMarkdown* con texto centrado y que omita el conteo de casos perdidos.

### Ejercicio 33.4

```
descr(CEP$edad, transpose = TRUE,
      stats = c("N.Valid", "min", "q1", "med", "mean", "sd", "q3", "max", "iqr"),
      style = "rmarkdown", justify = "center", omit.headings = T)
```

	N.Valid	Min	Q1	Median	Mean	Std.Dev	Q3	Max	IQR
edad	1424.00	18.00	36.00	50.00	49.87	17.79	64.00	97.00	28.00

De tal modo se construye una tabla de resultados adecuada para un reporte de investigación, con la información específica que interesa desplegar para análisis.

#### 9.5.3. Tablas de estadísticos univariados de alcance poblacional

En este apartado se indica como construir tablas de presentación para los resultados de estimación de parámetros poblacionales a partir de resultados que ya han sido calculados en ejemplos anteriores.

### 9.5.3.1. Intervalos de confianza para proporciones

Para el caso de construir una tabla para el intervalo de confianza de una proporción se considera lo ya ejecutado en el ejercicio 21. Luego de contar con los resultados de tal ejercicio se procede a definir como vector simple a cada resultado. En el caso de los límites del intervalo de confianza simplemente se copia cada resultado (producto de las función *exactci*) y se almacena como un vector simple (*linf* y *lsup* respectivamente); lo mismo se realiza con el nivel de confianza (vector *nc*). Luego, mediante la función *cbind*, se configura una matriz (*data.frame*) con tales valores.

#### Ejercicio 33.5

```
#Ejercicio 21
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

#Definición de cada valor como vector simple
linf <- (0.4863248*100)
lsup <- (0.5389039*100)
nc <- 0.95*100

#Configuración de un data.frame a partir de los vectores creados
ICP <- cbind(linf, lsup, nc)
```

Con tales elementos se puede utilizar la función *kable* del paquete *knitr* para imprimir tal matriz de datos en formato RMarkdown. Así, se le indica el objeto a imprimir (*ICP*), con el argumento *caption* se le agrega un título a la tabla, mediante el argumento *align* se le indica la alineación del texto (en este caso, 'c' significa texto centrado), el argumento *digits* permite indicar la cantidad de decimales aceptados en el redondeo y finalmente el argumento *col.names* permite indicar un vector de caracteres con los títulos de cada columna. Especificar la opción *asis* en la configuración del trozo de código permite que los resultados se impriman sin ser editados en el formato RMarkdown.

### Ejercicio 33.5 (continuación)

```
#Paquete necesario para imprimir tablas
library(knitr)
#Construcción de tabla de resultados con formato
kable(ICP, caption = "Tabla 1. Estimación de un intervalo de confianza para proporciones",
      align = 'c', digits = round(2),
      col.names = c("Límite inferior", "Límite superior",
                    "Nivel de confianza"))
```

Cuadro 9: Tabla 1. Estimación de un intervalo de confianza para proporciones

Límite inferior	Límite superior	Nivel de confianza
48.63	53.89	95

#### 9.5.3.2. Intervalos de confianza para medias

Algo similar a lo ya indicado se efectúa para construir una tabla de resultados de la estimación del intervalo de confianza de una media. En este caso se replica lo calculado en el ejercicio 22 y los resultados de la función *ci.mean* se almacenan configurados como matriz de datos (usando la función *as.data.frame*) en un nuevo objeto denominado *ic*.

### Ejercicio 33.6

```
# Intervalos de confianza
library(Publish)
#Nivel de confianza por defecto.
ci.mean(CEP$satisfaccion_vida)
```

```
## mean CI-95%
## 7.31 [7.20;7.42]
```

```
ic <- as.data.frame(ci.mean(CEP$satisfaccion_vida))
```

Sobre este objeto se aplica la función *kable* del paquete *knitr*. De forma similar al ejemplo anterior, se aplica la función señalada sobre el objeto *ic*; en este caso, se seleccionan las columnas 1, de la 3 a la 5, y la columna 2 (en ese orden). Con el argumento *caption* se le agrega un título a la tabla, mediante el argumento *align* se le indica la alineación del texto (en este caso, 'c' significa texto centrado), el argumento *digits* permite indicar la cantidad de decimales aceptados en el redondeo y finalmente el argumento *col.names* permite indicar un vector de caracteres con los títulos de cada columna. Especificar la opción *asis* en la configuración del trozo de código permite que los resultados se impriman sin ser editados en el formato RMarkdown.

### Ejercicio 33.6 (continuación)

```
kable(ic[c(1,3:5,2)], caption = "Tabla 2. Estimación de un intervalo de confianza para media",
      align = 'c', digits = round(2),
      col.names = c("Media", "Límite superior", "Límite inferior",
                    "Nivel de confianza", "Error estándar"))
```

Cuadro 10: Tabla 2. Estimación de un intervalo de confianza para media

Media	Límite superior	Límite inferior	Nivel de confianza	Error estándar
7.31	7.2	7.42	0.05	0.06

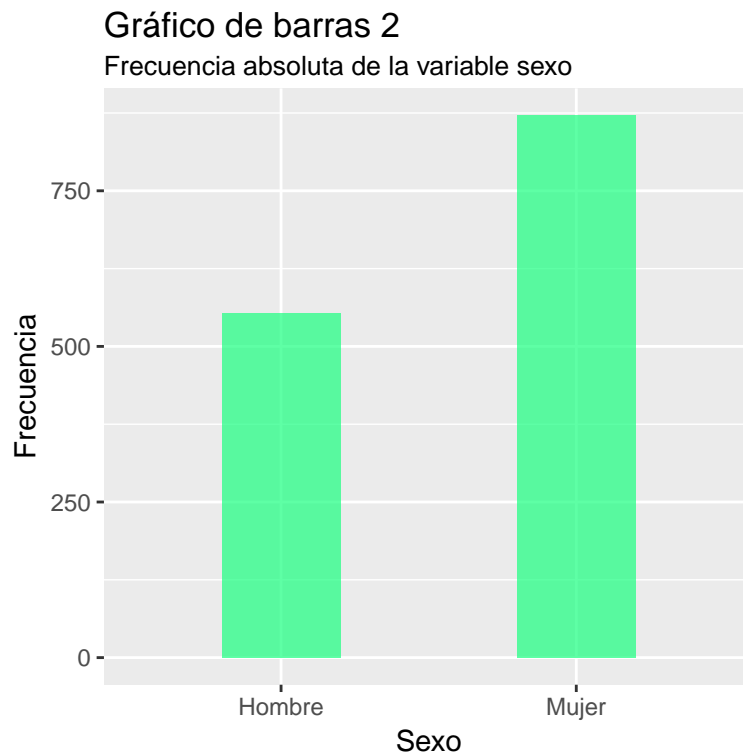
Con tales configuraciones ya es posible contar con tablas de resultados configuradas para ser impresas vía RMarkdown, con un formato adecuado para su presentación en un reporte de resultados.

#### 9.5.4. Gráficos

La inserción de gráficos en un reporte a compilar vía RMarkdown es simple. Sólo se precisa incorporar alguna de las sintaxis para procesar gráficos ya construidas anteriormente en este material, en un trozo de código de R. En este caso se replica el gráfico construido en el ejercicio 29.1. Vale la pena recordar que si se busca no compilar el trozo de código de R en el reporte, hay que agregar el argumento `echo = TRUE` a las especificaciones del trozo de código. Con los argumentos `fig.height`, `fig.width` y `fig.align` es posible configurar el ancho y alto máximos de la figura así como su alineación respecto a la página.

#### Ejercicio 33.6

```
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")
```



## 10. Materiales de apoyo para el aprendizaje

### 10.1. Instalación de R y RStudio

- Página explicativa de la [instalación de R en diferentes sistemas operativos]((<http://www.meccanismocomplesso.org/en/guide-installing-r-for-windows-linux-mac-os-x/>)) Windows, Linux y MacOSX

### 10.2. Uso de RMarkdown

- Video (en inglés): cómo [insertar bibliografía en RMarkdown](#), usando un archivo .bib existente:
- Página explicativa de diferentes [herramientas de formato \(metadatos\) para RMarkdown](#), pensando especialmente en la construcción de reportes académicos.
- Enlaces de referencia para el uso de RMarkdown:
  - <https://www.r-bloggers.com/how-to-create-reports-with-r-markdown-in-rstudio/amp/>
  - [http://www.geo.uzh.ch/microsite/reproducible\\_research/post/rr-r-publication/](http://www.geo.uzh.ch/microsite/reproducible_research/post/rr-r-publication/)
  - [https://rmarkdown.rstudio.com/authoring\\_pandoc\\_markdown.html#images](https://rmarkdown.rstudio.com/authoring_pandoc_markdown.html#images)
- Plantillas de [formato para pdfs](#) contruidos desde RMarkdown

### 10.3. Uso de Zotero como gestor de referencias bibliográficas

- Página para [descargar Zotero y Zotero connector](#)
- Página para [descargar diferentes formatos para referencias bibliográficas](#).

### 10.4. LaTeX y RMarkdown

- Explicación sobre programa [TinyTex](#) como plataforma LaTeX adecuada a RMarkdown

## Referencias bibliográficas

- Blalock, H. M. (1966). *Estadística social*. Fondo de Cultura Económica.
- CEP. (2017). Manual del Usuario Encuesta CEP N° 81. Estudio Nacional de Opinión Pública N° 51 – Tercera Serie, Septiembre-Octubre 2017. Centro de Estudios Públicos. Recuperado a partir de <https://www.cepchile.cl/estudio-nacional-de-opinion-publica-septiembre-octubre-2017/cep/2017-10-25/105022.html>
- Comtois, D. (2018). Introduction to summarytools. Recuperado a partir de <https://cran.r-project.org/web/packages/summarytools/vignettes/Introduction.html>
- Elousa, P. (2009). ¿EXISTE VIDA MÁS ALLÁ DEL SPSS? DESCUBRE R. *Revista Psicothema*, 21(4), 652-655. Recuperado a partir de <http://www.psicothema.com/psicothema.asp?id=3686>
- Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R* (Edición: 1). London ; Thousand Oaks, Calif: SAGE Publications Ltd.
- Grolemund, G. (2014). Introduction to R Markdown. Recuperado a partir de [https://rmarkdown.rstudio.com/articles\\_intro.html](https://rmarkdown.rstudio.com/articles_intro.html)
- Miller, S. V. (2018, junio). A Pandoc Markdown Article Starter and Template. Recuperado a partir de <https://github.com/svmiller/svm-r-markdown-templates>
- Navarro, J. (2014). LaTeX Fácil: Guía rápida de LaTeX. Recuperado a partir de <http://nokyotsu.com/latex/guia.html>
- Ritchey, F. J. (2008). *Estadística para las ciencias sociales*. McGraw-Hill Interamericana de España S.L.
- Wilkinson, L., Wills, D., & Rope, D. (2005). *The Grammar of Graphics* (Edición: 2nd edition. 2005). New York: Springer.
- Workshop, R. R. (2016). Writing publications with R. *Writing publications with R*. Recuperado a partir de [http://www.geo.uzh.ch/microsite/reproducible\\_research/post/rr-r-publication/](http://www.geo.uzh.ch/microsite/reproducible_research/post/rr-r-publication/)