R para periodistas: Una introducción.

Mauricio Franco-Cisterna

http://reporterodedatos.com

Resumen Esta es una introducción a *R* , donde mostramos las características de *R* y *RStudio*, con instrucciones básicas para iniciarse en el uso de estas herramientas: Cómo instalarlas, mostrar ejemplos de funciones y gráficas básicas, e introducir el trabajo con archivo de datos.

1. Esa cosa llamada *R*

En el contexto actual, los datos que podemos recolectar para obtener la información que nos permita construir la noticia, pueden encontrarse en numerosas fuentes y alcanzar un volumen enorme. Un gran aliado en la acción de procesar dichos datos y hacer más fácil el comunicar la noticia son los computadores y las numerosas herramientas informáticas que se encuentran disponibles para realizar esta labor. En este capítulo nos centramos en el lenguaje *R* , que dentro de sus ventajas, se encuentran el ser potente, versátil, de bajo costo y que puede ser instalada en prácticamente cualquier computador. También, daremos consejos simples y efectivos para poder utilizarlo rápidamente y así, ganar confianza y destreza en su uso.

1.1. ¿Qué es *R* ?

A grandes rasgos, *R* es un **lenguaje de programación**, enfocado en el análisis estadístico, con capacidad para producir gráficos de diversos tipos. Es muy utilizado en la investigación científica y otros campos como minería de datos y **periodismo de datos**.

1.2. ¿Por qué usar *R* ?

R tiene múltiples ventajas que lo hacen muy útil para el periodismo de datos.

***R* es flexible:** Una de las principales ventajas de *R* es que, al ser un lenguaje de programación, nos otorga mucha libertad y flexibilidad al momento de definir las tareas que queremos realizar. Veámoslo de la siguiente forma: A la hora de trabajar con un computador, lo ideal sería poder decirle directamente que es lo que deseamos hacer, y que el computador automáticamente se dedique a resolver dicha tarea hasta finalizarla. Un lenguaje de programación, precisamente, nos permite "decirle" al computador, mediante órdenes llamadas *instrucciones*, cuales son las tareas que queremos que realice. Una vez escrita la lista de tareas (es

decir, el *programa*), se ejecuta, con lo cual, el computador empieza la resolución de las tareas, hasta entregarnos nuestro resultado final, ya sea un análisis de datos o un gráfico explicativo, por ejemplo.

***R* es de bajo costo:** *R* se distribuye bajo la licencia GNU GPL, lo que en términos prácticos, significa que usted puede **distribuir, copiar y usar *R* libremente y de forma gratuita**. Puede instalarlo en todos los computadores que estime conveniente, sin pagar ninguna licencia ni suscripción[1].

***R* es multiplataforma:** Es decir, usted puede instalar *R* en cualquier computador con sistema operativo Windows, MacOS o GNU/Linux[2].

***R* es versátil:** Ya de por sí, *R* permite realizar un gran número de procedimientos estadísticos. Pero además, permite cargar diferentes bibliotecas o paquetes para más funcionalidades de cálculo, presentaciones gráficas o procedimientos específicos (por ejemplo, trabajar con mapas).

***R* es potente:** *R* permite trabajar con archivos de datos de diversos tipos y pesos. Por poner un ejemplo: MS Excel permite trabajar con tablas con un máximo de poco más de un millón de filas. Aunque parece un número grande, es fáctible encontrar datos en un volumen que supere varias veces esa cantidad.

***R* está en constante actualización:** Debido a lo masivo de su uso, la variedad de campos donde se utiliza y a que hay una gran comunidad de colaboradores trabajando en su mejora, *R* es una buena opción para trabajar con una herramienta siempre actualizada.

1.3. ¿Cómo instalar *R* ?

Para **instalar *R*** , debes ir a la página oficial con la lista de servidores de descarga del proyecto (*mirrors*):

`https://cran.r-project.org/mirrors.html`

elegir uno de ellos servidores y descargar la versión correspondiente a tu sistema operativo. Una vez descargado, hacer doble click en el ejecutable para iniciar la instalación (Windows y MacOS)[3].

[1] La licencia GNU GPL también le permite acceder al código fuente de *R* (ver cómo está *fabricado*) y realizar modificaciones para adaptarlo a sus necesidades. Esto es muy útil para usuarios avanzados.

[2] Ejemplos de distribuciones Linux: Ubuntu, Linux Mint o Elementary OS.

[3] En el caso de Linux, debes compilar *R* desde el código fuente. La forma de hacerlo se encuentra en la sección de preguntas frecuentes (FAQ).

2. El entorno *RStudio*: Haciendo *R* amigable

RStudio es un programa diseñado con el objetivo de hacer del trabajo con *R* una tarea mucho más amigable. Presenta múltiples características pensadas con ese objetivo. Si bien no es imprescindible (uno puede trabajar con *R* sin instalar *RStudio*), es altamente recomendable su uso, tanto para usuarios que recién se inician con *R* como para usuarios más experimentados. Al igual que *R* , *RStudio* también es multiplataforma y es *software* libre[4], o sea, permite ser copiado y usado libre y gratuitamente.

Una vez tengas instalado *R* en tu computador, puedes continuar con *RStudio*. Para instalarlo, entras a su página oficial de descarga, elige la versión de tu gusto (Te recomendamos "*RStudio Desktop*"):

`https://www.rstudio.com/products/rstudio/download/`

y descarga el ejecutable correspondiente a tu sistema operativo. Finalmente, haces doble click en el ejecutable descargado para iniciar la instalación.

2.1. Ambientes de *RStudio*

Al abrir *RStudio*, notamos de inmediato que hay cuatro secciones principales (Ver figura 1):

- La sección de comandos (*prompt*), la cual es una terminal interactiva, que nos permite darle instrucciones a *R* y ejecutarlas inmediatamente. Esta se encuentra en el lado inferior izquierdo.
- Una sección de *scripts*, donde podemos crear un archivo con una lista de tareas o instrucciones (nuestro programa o *script*). De esta forma, podemos trabajar en nuestro código en varias sesiones, pudiendo diseñar tareas más complejas. Estos archivos o *scripts* los podemos ejecutar más tarde en cualquier computador con *R* [5], lo que facilita la colaboración con otros partícipes de la investigación. Esta sección la encontramos en el lado superior izquierdo.
- Un espacio de trabajo o ***workspace***, donde se muestran las variables creadas y algunas características. También permite ver el contenido de ciertas variables como tablas de datos. Esta sección se encuentra en el lado superior derecho. Nos permite ver que información es la que tenemos cargada en nuestro ambiente.
- Una sección de apoyo, donde se muestran los gráficos creados, la ayuda de las funciones cuando lo solicitemos y podemos instalar librerías.

[4] Se distribuye bajo licencia AGPLv3. También hay versiones de pago, que ofrecen servicios extras como soporte técnico y trabajar desde la nube.

[5] En el caso de que usemos librerías que no vienen por defecto, el computador donde ejecutemos el código debe tenerlas instaladas también.

Figura 1. Las distintas secciones de *RStudio*: Sección de comandos (rojo), sección de *scripts* (azul), espacio de trabajo (magenta) y sección de apoyo (verde).

3. Primeros pasos

Como mencionamos anteriormente, en *R* escribimos las instrucciones de lo que queremos hacer directamente en el prompt, el cual está señalizado por el signo `>` al inicio de la línea:

```
>
```

Por ejemplo, si queremos sumar 2 y 3, escribimos `2+3` y presionamos la tecla *Enter*:

```
> 2+3
[1] 5
```

Lo mismo si queremos restar (`-`), multiplicar (`*`) o dividir (`/`):

```
> 2*3
[1] 6
```

Muchas veces, vamos a necesitar salvar la información con alguna etiqueta que nos recuerde su uso, o puede ser una variable que cambie de valor con el tiempo. *R* nos permite asignar valores a **variables** usando el operador <—. Ejemplo:

```
> miNombre <- "Fulanito De Tal"
> miEdad <- 30
> edadJubilacion <- 65
> tiempoParaJubilar <- edadJubilacion - miEdad
```

De esta forma, si necesitase calcular el tiempo que le falta para jubilar a otra persona, basta con cambiar el valor de la variable `miEdad` y ejecutar las mismas instrucciones:

```
> miEdad <- 25
> tiempoParaJubilar <- edadJubilacion - miEdad
```

Si queremos trabajar con muchos datos, los podemos agrupar en una estructura llamada lista. Muchas veces, sobre todo al escribir *scripts*, queremos escribir texto para leerlo nosotros, como recordatorios o **comentarios**, y no que sea ejecutado. Para esto, escribimos el signo `#` delante de la línea que queremos dejar como texto. Estas líneas se conocen como comentario:

```
> # Este codigo es para calcular cuanto falta para jubilar
> miNombre <- "Fulanito De Tal" # El nombre es irrelevante
> # Esta es mi edad, por ahora.
> miEdad <- 30
> # Esta es la edad actual de jubilacion
> edadJubilacion <- 65
> # Este es el calculo final
> tiempoParaJubilar <- edadJubilacion - miEdad
```

Nótese que los comentarios pueden ir en líneas apartes o al final de una instrucción (línea de código). Los comentarios son fundamentales para recordar o informar a otro colaborador cual es la tarea que realizan nuestros códigos, por qué están ahí y por qué fueron programados de esa forma.
R también ofrece **funciones**, las cuales permiten realizar tareas específicas, dados ciertos parámetros de entrada que le asignamos previamente. Por ejemplo, podemos sumar 2 y 3 usando la función `sum()`:

```
> sum(2,3)
```

Si queremos trabajar con varios datos a la vez, *R* nos ofrece un estructura de datos llamada *vector*, la cual es, de forma coloquial, una lista de elementos del mismo tipo (por ejemplo, números o texto). Para construir un vector, usamos la función `c()` (*concatenate*). Por ejemplo:

```
> misNumeros <- c(1,2,3,4,5,6,7,8,9,10)
```

crea un vector llamado `misNumeros` con los números del 1 al 10. Para generar secuencia de números seguidos, como nuestro vector anterior, podemos usar la función `seq` (por *sequence*):

```
> misNumeros <- seq(1,10) # Numeros del 1 al 10
> misNumeros
[1]  1  2  3  4  5  6  7  8  9 10
> misOtrosNumeros <- seq(1,10,2) # Numeros del 1 al 10,de 2 en 2
> misOtrosNumeros
[1]  1  3  5  7  9
```

Una vez creados nuestros vectores, podemos hacer sobre ellos operaciones varias, por ejemplo, sumar o multiplicar todos los elementos por un mismo número, calcular la suma o el promedio de todos ellos e incluso, crear gráficos.

```
> 2*misNumeros
[1]  2  4  6  8 10 12 14 16 18 20
> misNumeros+100
[1] 101 102 103 104 105 106 107 108 109 110
> misNumeros/10
[1] 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0
> plot(misNumeros) # Crea gráfico de puntos
> barplot(misNumeros) # Crea gráfico de barras
```

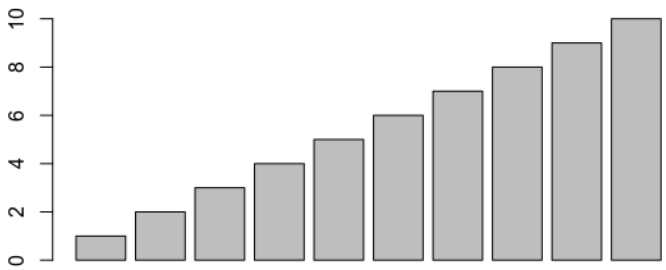

Figura 2. Nuestro primer gráfico de barras, creado usando una única instrucción, sin opciones extras. Más detalles se le pueden añadir usando las opciones de la misma función, e incluso con otras funciones.

Un error muy común al escribir funciones es no dar la cantidad correcta de argumentos. Por ejemplo, escribamos en el prompt `mean()`:

```
> mean()
Error in mean.default() :
  el argumento "x" está ausente, sin valor por omisión
```

estos mensajes de error no significan nada grave: Sólo son avisos que indican que no se pudo realizar la orden dada. Para realizar la acción pedida, basta con volver a intentarlo, pero corrigiendo la situación que fue mencionado en el mensaje. Si se necesita ayuda sobre una función en específico, existe la función `help()`, que nos mostrará información sobre la función escogida bajo la pestaña "Help" (Sección de apoyo):

```
> help(mean) # Muestra ayuda de la función "mean"
```

4. Librerías

Muchas veces, para hacer una tarea en particular con *R* necesitaremos una o más funciones especializadas. Estas funciones las podemos encontrar en *librerías*, que son, a grandes rasgos, una colección de funciones y tipos de variables diseñadas para realizar un **conjunto de tareas específicos**. Algunos ejemplos de librerías son:

- `ggplot2`: Permite crear gráficos con estética mejorada.
- `xlsx`: Permite cargar archivos de MS Excel.
- `lubridate`: Facilita trabajar con fechas y horas.
- `networkD3`: Permite crear gráficos de redes de interacción.

Para instalar una nueva librería, basta con escribir en el *prompt* la instrucción `install.packages(<Nombre de la librería>)`. Por ejemplo, si quisieramos instalar la librería `googleVis`, escribimos:

```
> install.packages('googleVis')
```

Luego, cuando requieras usar una librería, la puedes llamar con la instrucción `library(<Nombre de la librería>)`. Siguiendo con el ejemplo anterior, para cargar la librería `googleVis`, escribimos:

```
> library('googleVis')
```

Un error muy frecuente es olvidar cargar las librerías que vamos a usar o simplemente no haberla instalado. En ese caso, se muestra el siguiente error:

```
Error: could not find function "googleVis"
```

Nuevamente, este error se soluciona fácilmente: Cargando la librería correspondiente.

5. Trabajando con datos: Archivos *CSV*

Los datos que encontrarás en el marco de tu investigación vendrán en numerosos formatos, por ejemplo:

- xls, xlsx: Hojas de cálculo de MS Excel
- csv: Archivo de texto, con datos separados por comas.
- tsv: Archivo de texto, con datos separados por tabulación.
- json: Archivo de texto, con formato para facilitar el intercambio de datos.
- ods: Hojas de cálculo de LibreOffice y OpenOffice.

En este capítulo nos centraremos en los archivos en formato *CSV*, ya que es uno de los formatos más utilizados para almacenar datos. Un archivo *CSV* es un archivo de texto, en el cual los datos se encuentran almacenados emulando una tabla. Así, cada línea representa una fila y las columnas se encuentran delimitadas por un separador, usualmente una coma. Los nombres de las columnas de datos suelen incluirse en la primera fila del archivo, la cual es conocida como encabezado (*header*).

La forma más simple de cargar datos es usando la instrucción `read.csv()`:

```
> datos <- read.csv("SueldosPorOcupacion.csv")
```

En este ejemplo, cargamos un archivo[6] llamado `SueldosPorOcupacion.csv`, que se encuentra ubicado en nuestro actual directorio de trabajo, y almacena la información en una variable llamada `datos`. Esta forma de organizar nuestra información es lo que se conoce como *dataframe*. El encabezado (los nombres de las columnas) las podemos ver con la instrucción `names()`:

```
> names(datos)
[1] "ID.ISCO"       "ISCO"          "ID.Year"       "Year"          "ID.Sex"
[6] "Sex"           "Median.Income"
```

Si queremos seleccionar una columna, escribimos el nombre de la columna junto al de nuestro *dataframe*, separados por el símbolo de dólar (`$`). Por ejemplo, supongamos que queremos seleccionar la columna `Median.Income`, escribimos:

```
> datos$Median.Income
```

Podemos tener una idea general de nuestro *dataframe*, podemos usar la instrucción `head()`

```
> head(datos)
```

lo cual nos muestra las primeras cinco filas (sin contar el encabezado):

```
  ID.ISCO                 ISCO ID.Year Year ID.Sex       Sex Median.Income
1       1 Funcionarios públicos    2010 2010      1  Femenino        600000
2       1 Funcionarios públicos    2010 2010      2 Masculino       1000978
3       1 Funcionarios públicos    2011 2011      1  Femenino        715441
4       1 Funcionarios públicos    2011 2011      2 Masculino       1000000
5       1 Funcionarios públicos    2012 2012      1  Femenino        800000
6       1 Funcionarios públicos    2012 2012      2 Masculino       1104961
```

De aquí podemos ver que nuestros datos consideran registros de varios años. Si sólo quisieramos considerar, por ejemplo, los datos del año 2013, podemos ingresar la condición que buscamos dentro de corchetes (`[ ]`). Nótese que la comparación de igualdad la hacemos con doble signo igual (`==`).

```
> datos$Median.Income[datos$Year == 2013]
```

Si ahora, quisieramos seleccionar los datos del año 2013 **y** correspondientes a mujeres, podemos unir ambas condiciones con &:

```
> # Seleccionamos datos de 2013 y correspondientes a mujeres
> mujeres2013 <- (datos$Year == 2013) & (datos$Sex == "Femenino")
> # Mostramos la mediana de los ingresos para mujeres en 2013
> datos$Median.Income[mujeres2013]
```

[6] Ejemplo creado a partir de modificar datos disponibles en `https://es.datachile.io/`.

De esta forma, si, por ejemplo, quisieramos ver gráficamente como se distribuyeron los sueldos por ocupación para mujeres el año 2013, podemos escribir las siguientes instrucciones:

```
> sueldos2013 <- datos$Median.Income[mujeres2013]
> hist(sueldos2013,xlab="Sueldos Mujeres - 2013 (CLP)")
```

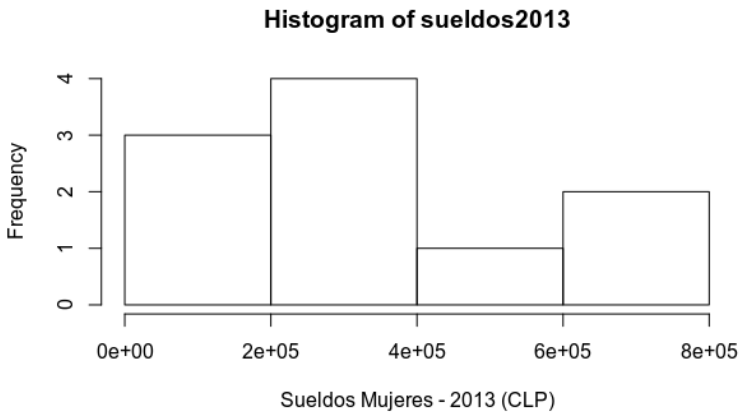

Figura 3. Nuestro primer *histograma*: Distribución de la mediana de los sueldos por ocupación. Podemos notar que la mayoría de las ocupaciones son remuneradas con sueldos cuya mediana está entre 200.000 y 400.000 pesos chilenos (CLP)

La opción *xlab* nos permite cambiar el texto de la etiqueta bajo el eje X (horizontal). Las funciones para graficar, como `hist()` nos ofrecen muchas más opciones para mejorar la presentación de los gráficos. Además, existen otras librerías que nos ofrecen mejoras estéticas con acabados profesionales. Si quisieramos guardar nuestro gráfico, una opción rápida es exportarla usando *RStudio*, haciendo click en el botón "Export", en la sección de apoyo (donde se muestran los gráficos) y luego en "Save as image".

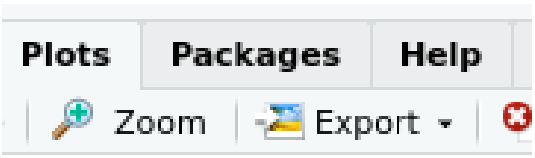

Figura 4. Botón "Export", para guardar nuestros gráficos.

5.1. Cargando archivos: Otros casos

El ejemplo que vimos anteriormente, corresponde a un archivo: a) En formato *CSV*, b) con datos separados por coma y c) con una primera fila como encabezado. Ahora veremos que hacer si alguna de estas condiciones cambia. En el caso de que no haya encabezado (*header*), o sea, nuestro archivo sólo contiene datos desde la **primera fila**,debemos indicárselo la instrucción `read.csv()` usando precisamente la opción `header`:

```
> datos <- read.csv("MisDatos.csv", header=FALSE)
```

creando un *dataframe* en el cual el nombre de las columnas se etiquetan como `V1`, `V2`, `V3` y así sucesivamente. Puedes cambiar los nombres con la siguiente instrucción, donde el nombre de las columnas se escribe dentro de una lista:

```
> names(datos) <- c("NombreColumna1","MiColumna2","Datos3")
```

Para el caso de que nuestros datos vengan delimitados por un separador que no es coma, se indica con la opción `sep`:

```
> # Ejemplo anterior
> datos <- read.csv("SueldosPorOcupacion.csv",sep=",")
> # Si el delimitador es un espacio " ".
> datos <- read.csv("DatosEspacio.csv",sep=" ")
> # Si el delimitador es una tabulacion "\t".
> datos <- read.csv("DatosTabulados.csv",sep="\t")
```

Para cargar datos en archivos *XLSX* (hojas de cálculo de MS Excel), necesitaremos la librería `xlsx`. Si se requiere instalar, podemos hacerlo con la instrucción:

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

Si ya tenemos nuestra librería instalada, cargamos los archivos en uns hoja en particular de nuestro archivo con las siguientes instrucciones:

```
> # Carga la libreria
> library("xlsx")
> # Carga la primera hoja del archivo "MisPlanillas.xlsx"
> datos <- read.xlsx("MisPlanillas.xlsx",1)
> # Carga la segunda hoja del archivo "DatosOtraHoja.xlsx"
> datos <- read.xlsx("DatosOtraHoja.xlsx",sheetIndex=2)
> # Carga hoja de nombre "Senadores" del archivo "DatosInteres.xlsx"
> datos <- read.xlsx("DatosInteres.xlsx",sheetName="Senadores")
```

6. Resumen

En este capítulo, hemos visto de forma somera las capacidades de análisis de datos y creación gráfica que *R* ofrece. Vimos como instalar *R* y *RStudio*, usar algunas funciones y obtener ayuda, crear algunos gráficos simples y trabajar

con información contenida en archivos. Lo mostrado en esta introducción es una pequeña muestra de todo el potencial que *R* ofrece. Esperamos que te atrevas a instalar, probar y sumergirte en el uso de esta potente herramienta que te ayudará en tus futuras investigaciones.