Redacción de artículos académicos con R

Episodio 4: Mi primer borrador Rmarkdown

Bajaña Alex

Chanatasig Evelyn

Heredia Aracely

2022-06-25

Estudio de género

Base de datos:

ENEMDU 2019, diciembre

Crear Directorios

Crear las carpetas: Datos y Diccionarios

Antes de descargar la base de datos es necesario crear las carpetas datos y diccionarios. Estas están ubicadas dentro de la carpeta principal: `04_mi_primer_rmd`.

De esta manera estamos creando los **directorios** necesarios para nuestro caso de estudio. Cabe notar que las carpetas por crear estarán vacías puesto que allí es donde se ubicarán los archivos que descargaremos.

```
# Crear la carpeta "datos"
if(!dir.exists("04_mi_primer_rmd/datos/")){
  dir.create("04_mi_primer_rmd/datos/")}

# Crear la carpeta "diccionarios"
if(!dir.exists("04_mi_primer_rmd/diccionarios/")){
  dir.create("04_mi_primer_rmd/diccionarios/")
}

# Estas son funciones `if`, esto significa que sólo se ejecutarán
# si cumplen la condición de no tener ya creadas las carpetas.
```

Creando la base de datos

Descargar la base de datos

El siguiente comando descarga en un archivo comprimido `zip` la Encuesta Nacional de Empleo, Desempleo y Subempleo (ENEMDU) 2019 correspondiente al mes de diciembre, sólo si no se tiene descargado previamente en la carpeta.

Esta base se descarga de la página del INEC.

```
if(!file.exists("04_mi_primer_rmd/enemdu_2019_12.zip")){
  download.file(url = "https://www.ecuadorencifras.gob.ec/documentos/web-inec/EMPLEO/2019/
                destfile = "04_mi_primer_rmd/enemdu_2019_12.zip")
}
```

Revisar contenido

Una vez descargada la base de datos, podemos observar que es un archivo `zip`, por ello vamos a revisar los archivos que contiene:

```
# Guardo el archivo `zip` en mi environment:
enemdu_file <- "04_mi_primer_rmd/enemdu_2019_12.zip"

# Reviso los archivos que `enemdu_file` tiene:
unzip(zipfile = enemdu_file,list = T)
```

```
                        Name   Length                Date
1    enemdu_viv_hog_201912.csv  2531442 2020-01-11 16:06:00
2          2019_Metadatos.xlsx   135115 2020-01-13 16:09:00
3    DICCIONARIO_VARIABLES.zip   790122 2020-01-11 16:07:00
4 enemdu_consumidor_201912.csv  2565411 2020-01-11 16:05:00
5    enemdu_persona_201912.csv 25505588 2020-01-11 16:06:00
```

- **diccionario:** es un archivo zip
- **enemdu personas:** es unarchivo csv

Al revisar el archivo comprimido se puede observar que contiene 5 archivos, de los cuales necesitamos únicamente dos: el diccionario (3ro) y la base de personas (5ta).

Base de personas

Descomprimir y Extraer: base personas

De los cinco archivos que la carpeta contiene, sólo necesitamos extraer dos, que son: diccionario y base de personas, es por ello que, en el siguiente comando los especificamos, descomprimimos y extraemos en la carpeta general: `04_mi_primer_rmd`.

```
unzip(zipfile = enemdu_file,
      files = c("enemdu_persona_201912.csv",    # base de personas
                "DICCIONARIO_VARIABLES.zip"),   # diccionario
      exdir = "04_mi_primer_rmd" )               # carpeta donde se van a guardar
```

Crear directorio: datos

La `base de personas` tiene una extensión `csv`, esto significa que ya la podemos usar como nuestro insumo `base de datos`. Para guardar este insumo en la carpeta `datos` debemos hacer un "copiar y pegar" con el siguiente comando:

```
file.copy(from = "04_mi_primer_rmd/enemdu_persona_201912.csv",
          to = "04_mi_primer_rmd/datos/enemdu_persona_201912.csv")
```

> Este paso es para tener ordena nuestra carpeta, es decir, mantener los datos en una subcarpeta.

Diccionarios

Descomprimir y Extraer: diccionarios

Revisar los archivos

Como el archivo `diccionarios` tiene una extensión `zip` es necesario descomprimirlo, para ello, primero tenemos que saber su contenido. El siguiente comando mostrará los elementos.

```
unzip(zipfile = "04_mi_primer_rmd/DICCIONARIO_VARIABLES.zip",list = T)
```

```
                              Name Length                Date
1     enemdu_consumidor_2019_12.xlsx 283048 2020-01-11 16:00:00
2       enemdu_personas_2019_12.xlsx 292342 2020-01-11 16:01:00
3 enemdu_vivienda_hogar_2019_12.xlsx 283513 2020-01-11 16:03:00
```

Se observa que contiene tres documentos `xlsx` de los cuales vamos a extraer el segundo.

Crear directorio: diccionarios

Se va a extraer el documento `enemdu_personas_2012_12.xlsx` (el segundo) porque es el diccionario de la base que guardamos, es decir, nos muestra lo que las siglas de nuestra base significan.

```
unzip(
  #Documento origen a descomprimir
  zipfile = "04_mi_primer_rmd/DICCIONARIO_VARIABLES.zip",
  #Archivo específico por extraer (el segundo)
  files = "enemdu_personas_2019_12.xlsx",
  #carpeta destino (donde guardar el archivo)
  exdir = "04_mi_primer_rmd/diccionarios")
```

Limpiar la carpeta

Eliminar archivos

Una vez que hemos descargado y guardado la base de datos y su respectivo diccionario en su carpeta correspondiente, podemos borrar los archivos que no necesitamos que son: la base principal con extensión zip, la base personas de extensión `csv` y el diccionario de extensión `zip` también, los cuales se encuentran en la carpeta principal `04_mi_primer_rmd`.

De esta forma nuestra carpeta principal queda limpia.

```
file.remove(c("04_mi_primer_rmd/enemdu_2019_12.zip",
              "04_mi_primer_rmd/enemdu_persona_201912.csv",
              "04_mi_primer_rmd/DICCIONARIO_VARIABLES.zip"))
```

Metadato de la base

Metadato

Son datos que describen otros datos, se refiere a un grupo de datos que describen el contenido informativo de un objeto. En este caso, se presenta una lista de las características de la base de datos ENEMDU 2019, diciembre.

```
List of 8
 $ pais              : chr "Ecuador"
 $ titulo            : chr "Encuesta Nacional de Empleo, Desempleo y Subempleo - Diciembre 2019"
 $ subtitulo         : chr "RONDA LXVI-12-2019"
 $ tipo_estudio      : chr "Encuesta Fuerza Laboral"
 $ tipo_dato         : chr "Encuestapor muestreo (ssd)"
 $ unidad_de_analisis: chr [1:3] "Vivienda" "Hogar" "Personas miembros del hogar"
 $ alcance_tecnico   : chr [1:4] "Información de los Miembros del Hogar" "Características ocupacionales"
 "Ingresos" "Datos de la Vivienda y el Hogar"
 $ cobertura         : chr [1:3] "23 provincias del Ecuador" "5 ciudades: Quito, Guayaquil, Cuenca, Mach
ala, Ambato" "exceptúa a Galápagos"
```

```
#Ejecutar el código desde la línea 69 hasta la línea 82
```

> Ver más: Página del INEC

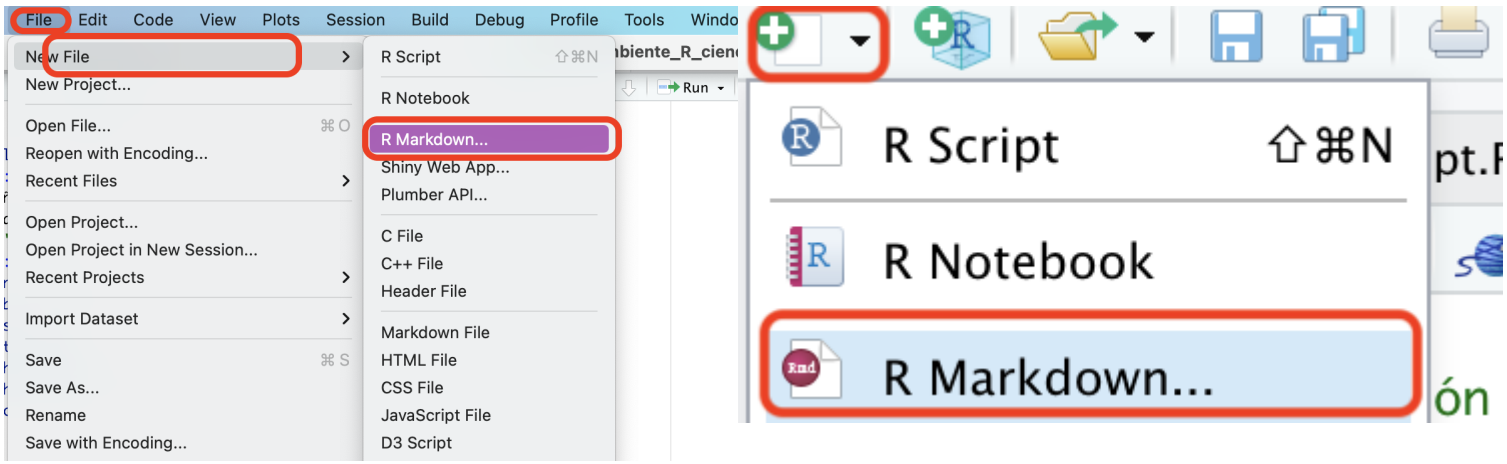
Documentos Rmarkdown

Abrir un nuevo Rmarkdown

Abrir un nuevo documento

Son ficheros de texto (se pueden escribir en cualquier editor de texto. Facilitan la tarea de generar informes o transparencias con contenido estadístico, ya que permiten mezclar en un mismo documento texto y código R.

Para abrir un `Rmarkdown` se tienen dos opciones:



Abrir un nuevo documento

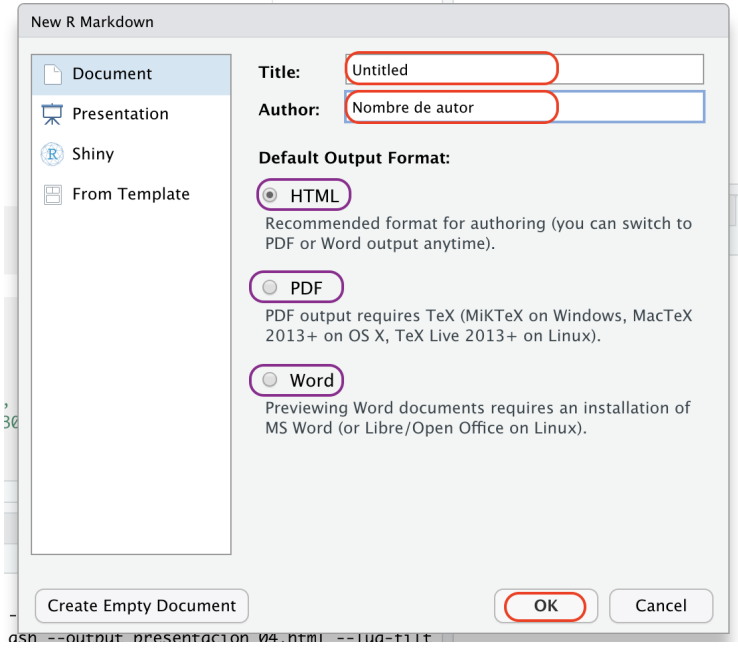
Una vez realizado los pasos anteriores aparecerá la siguiente ventana:

Cuadrados rojos:

- Title: es el nombre del documento
- Author: se escribe el nombre del autor
- OK: se presiona para crear el documento

Cuadros violeta: Aquí el tipo de documento que se quiera crear, ya sea con extensión:

- HTML
- PDF
- Word



Partes del documento

Cabecera

Aparece al principio del documento entre dos líneas `---`

En la cabecera escribes el título del documento, tu nombre y la fecha.

En output se indica el formato de salida. En el curso utilizaremos el formato `.html` y `.word.`

```
---
title: "Untitled"
author: "Nombre de autor"
date: "5/18/2021"
output: html_document
---
```

También conocido como Yaml

Es un formato para guardar objetos de datos con estructura de árbol. Sus siglas están en inglés y significan YAML: Ain't Markup Language (YAML no es otro lenguaje de marcado).

Este se utiliza al inicio de un R Markdown y sirve para declarar el preámbulo del documento.

Formatos de texto

A continuación sigue el texto del documento. El texto normal se escribe igual que en cualquier otro documento (como en un word).

Existen diferentes formas de darle formato:

- **Negrita:** Se escribe el texto entre dos asteriscos o dos guiones bajos `**Negrita**` o `__Negrita__`
- **Cursiva:** Se escribe el texto entre un par de asteriscos o un par de guiones bajos `*Cursiva*` o `_Cursiva_`
- **Cabecera de secciones**: Se usa el #, cuantos más símbolos # se escriban delante del texto, más pequeña será la letra.

```
# Título 1
## Título 2
### Título 3
```

Sentencias de R

En este tipo de documentos se puede insertar código de R. Para ello es necesario incluir este código justo después de la cabecera.

La primera línea hace que este código no aparezca en el documento final. La segunda línea es para permitir que el código R y la salida se impriman en el documento final.

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

Ejemplo de las partes

```
---
title: "Untitled"
author: "Nombre de autor"
date: "5/18/2021"
output: html_document
---
```

cabecera

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

justo después de la cabecera

```
## R Markdown

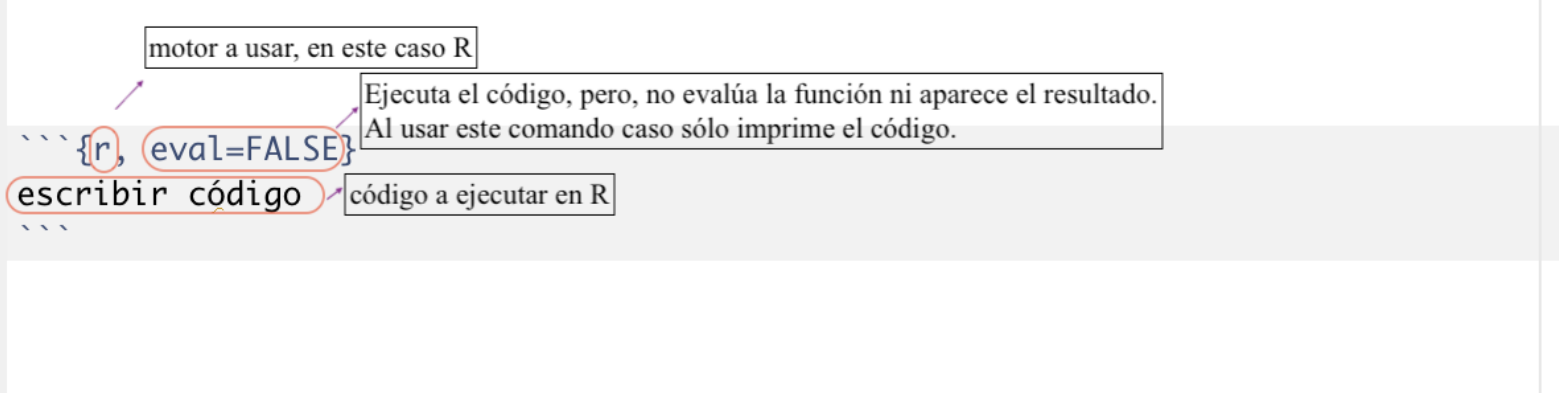
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF,
and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

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:
```

texto

Insertar código de R

Los bloques de código de R, o *chunks*, se indican dentro de un documento R Markdown de la siguiente manera:



Es útil para escribir código de R, y cuando se compila el documento, aparecen las salidas de R.

> Se puede escribir manualmente o puede insertarse de manera automática presionando `ctrl + alt + i`.

Insertar código de R

La parte entre llaves de la línea que abre el *chunk* puede contener diversos parámetros. Estos parámetros se separan de la `r` por una coma y entre ellos también con comas, y permiten determinar el comportamiento del bloque al compilar el documento. Los parámetros más útiles son:

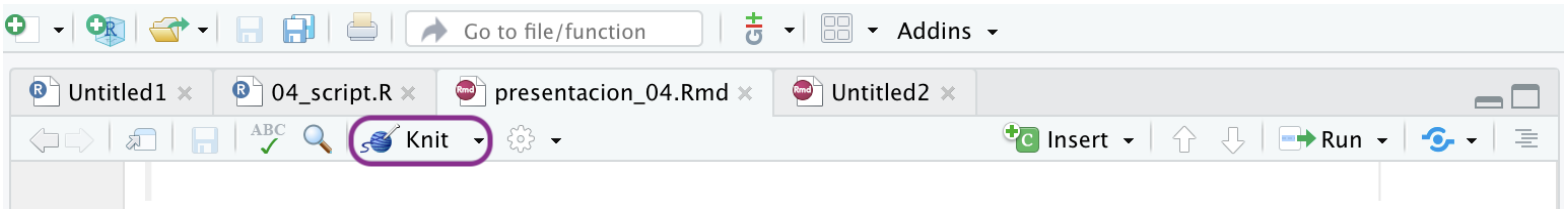
> **Nota:** Para cada parámetro, el valor por defecto no tiene por qué especificarse ya que por defecto siempre es: **TRUE**.

- **eval=F** si se iguala a **F** no lo evaluará pero sí mostrará el código.
- **include=F** ejecuta el código, pero, ni este, ni su resultado aparecen en el documento.
- **echo=F** ejecuta el código e imprime el resultado pero no el código
- **fig.{caracteristica}** opciones para modificar la presentación de los gráficos
- **warnings=F** si encuentra un *warning* no lo imprime
- **message=F** si encuentra un *message* no lo imprime
- **error=F** ejecuta el código, pero, si encuentra un *error* lo omite y pasa al siguiente (OJO con las dependencias)

Compilar el documento

Para obtener el documento final, simplemente hacer clic en el botón `Knit` y seleccionar el formato de salida desea para producir dicho documento.

Ejemplo: elegir `Knit to PDF` para producir un documento `.pdf`.



Conociendo nuevas funciones

Flextable

¿Qué es `flextable`?

- Es una librería que proporciona un marco para crear fácilmente tablas para informes y publicaciones.
- Puede crear fácilmente una tabla de informes a partir de un `data.frame`.
- Puede combinar celdas, agregar filas de encabezado, agregar filas de pie de página, cambiar cualquier formato y especificar cómo se deben mostrar los datos en las celdas. (Conocer más acerca de la función: flextable)
- **Ejemplo**: crear una tabla simple

```
library(flextable)
ejemplo <- data.frame(Sexo=c("Hombre","Hombre","Mujer"),
                      Edad=c(34,57,23),
                      Nivel_inst=c("secundaria","básica","superior"))
tabla <- flextable(ejemplo)
tabla
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Aplicar `theme` en una flextable

`R` contiene algunos diseños de tabla que permiten cambiar de forma rápida la apariencia de la misma. El objeto declarado dentro de `theme` tiene que ser de clase `flextable`. A continuación se presentan los diferentes `theme`:

Aplicar `theme` en una flextable

```
theme_alafoli(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_booktabs(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_box(tabla)
```

Sexo	**Edad**	**Nivel_inst**
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_tron(tabla)
```

Sexo	**Edad**	**Nivel_inst**
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Aplicar `theme` en una flextable

```
theme_vanilla(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_zebra(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_tron_legacy(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

```
theme_vader(tabla)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Partes de una `flextable`

- También se puede modificar cada una de las partes de la tabla de forma individual, para ello es necesario conocer las partes de la misma, una `flextable` está formada por: un encabezado, un cuerpo y un pie de página.
-

  Para especificar qué parte deben afectar las instrucciones de formato, use argumento `part.` Los posibles valores son:

  - **"header"**: la parte del encabezado de la tabla
  - **"footer"**: la parte de pie de página de la tabla
  - **"body"**: la parte del cuerpo de la mesa
  - **"all"**: el cuerpo y las partes del encabezado de la tabla

```
# Estructura general
nombre_tabla <- comando(tabla,  #nombre de la tabla tipo `flextable`
                        part="header") #Especificar parte
```

> - Clic para conocer más
> - Ayuda de viñeta: `vignette("format")`

Negrita

```
#Comando: bold()
bold(tabla, part="header")
```

Sexo	**Edad**	**Nivel_inst**
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Alineación del texto

```
#Comando: align()
align( tabla, align = "center", part = "al
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Letra cursiva

```
#Comando: italic()
italic(tabla,italic = TRUE)
```

Sexo	Edad	Nivel_inst
Hombre	*34*	*secundaria*
Hombre	*57*	*básica*
Mujer	*23*	*superior*

Tamaño celdas

```
#Comando: padding
padding( tabla, padding = 0.5, part = "all
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Tamaño de fuente

```
#Comando: fontsize()
fontsize(tabla,part = "header",size = 16)
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Color de fuente

```
#Comando: color()
color(tabla, color = "#E4C994")
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Tipo de fuente

```
#Comando: font
font(tabla, fontname = "Brush Script MT",
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

Color de fondo

```
#Comando: bg()
bg(tabla, bg = "#E4C994", part = "body")
```

Sexo	Edad	Nivel_inst
Hombre	34	secundaria
Hombre	57	básica
Mujer	23	superior

R por defecto tiene las siguientes funciones base para leer archivos

Funciones para leer archivos

Tipo.de.archivo	Librería	Funión.para.leer
Texto plano separado por comas	Ninguna	read.csv()
Texto con otro separador	Ninguna	read.delim()
Excel	readxl, openxlsx	read_excel()
SPSS	foreign, Hmisc, haven, readr	read.spss()
Stata	haven, foreign, read.dta13	read.dta()
R	readRDS	load()

Leyendo sin librería

Ejemplo: Leer archivos de texto

Para leer este tipo de archivos se usa la función `read.csv` o `read.delim()`, aquí se tiene que especificar 3 parámetros, el nombre del archivo, la manera en la cual están separados los datos y elegir si existe cabecera o no por leer.

```
read.csv(file = nombre, #  Nombre del archivo
         sep = ";", # Separador
         header = T) # Leer con cabecera

read.delim(file = nombre, #  Nombre del archivo
         sep = ";", # Separador
         header = T) # Leer con cabecera
```

En este ejemplo, el nombre del archivo se es `nombre`, el separador es `";"` (los separadores más comunes son el espacio, la coma y el punto y coma) y tiene cabecera (`header=T`), eso significa que toma la primera fila como el nombre de cada columna.

Leyendo con librería

Ejemplo: Leer archivos excel

Cuando se necesita una librería, se debe usar la función `library()`, y dentro de esta declarar la librería necesaria, en este caso es `readxl`.

Antes de importar los datos, necesitamos conocer el contenido de nuestra hoja de cálculo. Para ello usamos la función `excel_sheets` Esta permite conocer qué pestañas contiene nuestra hoja de cálculo sin salir de R.

Conociendo su contenido, procedemos a importar el contenido de cada pestaña:

```
library(readxl)

excel_sheets("excel_a_leer.xlsx")

read_excel(path = "excel_a_leer.xlsx",  #ruta del documento
           sheet = "nombre de la hoja", #especificar la pestaña a importar
           range = "C15:G170") #Es opcional, especifica el rango a importar
```

> Se pueden agregar más parámentros. Para conocer más de acerca de este tema hacer clic en el enlace

Funciones varias

Funciones varias

- `readLines`: permite leer algunas o todas las líneas de texto. Ejemplo:

```
readLines(nombre_del_archivo,
          n=5 #leer las primeras 5 líneas)
```

- `list.files()`: Muestra los nombres de los archivos y carpetas existentes dentro una carpeta. Si se usa `full names = T` se mostrarán los nombre con la extensión de la carpeta principal

```
list.files("04_mi_primer_rmd/") # Nombre de la carpeta a mostrar
list.files("04_mi_primer_rmd/",full.names = T) #Mostrar con la carpeta principal
```

Dando formato a los resultados de un modelo

La función `tidy` del paquete `broom` es un método. Permite resumir la información de un objeto resultado de un modelo estadístico. En el caso de estudio se lo usa para resumir los componentes de un modelo lineal o `lm`.

```
> summary(modelo)

Call:
lm(formula = ingresos ~ h_trabajo + nivel_inst + a_trabajo +
    edad + h_trabajo, data = hombres)

Residuals:
    Min      1Q  Median      3Q     Max
-1545.0  -148.1   -26.8    78.9  8444.4

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -329.37769   37.69626  -8.738  < 2e-16 ***
h_trabajo                          7.84525    0.37737  20.789  < 2e-16 ***
nivel_instCentro de alfabetización -18.27052  134.15657  -0.136  0.89168
nivel_instPrimaria                 80.75105   29.89224   2.701  0.00692 **
nivel_instEducación básica        186.31843   35.58931   5.235 1.69e-07 ***
nivel_instSecundaria              227.45103   30.34192   7.496 7.22e-14 ***
nivel_instEducación media         226.88699   32.93886   6.888 6.05e-12 ***
nivel_instSuperior no universitaria 451.38945  38.95794  11.587  < 2e-16 ***
nivel_instSuperior universitaria  556.81564   31.16866  17.865  < 2e-16 ***
nivel_instPost-grado             1195.14608   40.69432  29.369  < 2e-16 ***
a_trabajo                          0.06677    0.49028   0.136  0.89168
edad                               5.79665    0.45790  12.659  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 370.6 on 8563 degrees of freedom
Multiple R-squared:  0.2913,    Adjusted R-squared:  0.2904
F-statistic:   320 on 11 and 8563 DF,  p-value: < 2.2e-16
```

```
<- # Línea 440

   # Línea 441
```

```
> tabla_mod
# A tibble: 12 x 7
   term                                estimate std.error statistic   p.value
   <chr>                                  <dbl>     <dbl>     <dbl>     <dbl>
 1 (Intercept)                         -329.        37.7      -8.74  2.83e- 18
 2 h_trabajo                              7.85       0.377    20.8   1.09e- 93
 3 nivel_instCentro de alfabetización   -18.3      134.       -0.136 8.92e-  1
 4 nivel_instPrimaria                    80.8       29.9       2.70  6.92e-  3
 5 nivel_instEducación básica           186.        35.6       5.24  1.69e-  7
 6 nivel_instSecundaria                 227.        30.3       7.50  7.22e- 14
 7 nivel_instEducación media            227.        32.9       6.89  6.05e- 12
 8 nivel_instSuperior no universitaria  451.        39.0      11.6   8.18e- 31
 9 nivel_instSuperior universitaria     557.        31.2      17.9   4.13e- 70
10 nivel_instPost-grado                1195.        40.7      29.4   1.00e-180
11 a_trabajo                              0.0668     0.490     0.136 8.92e-  1
12 edad                                   5.80       0.458    12.7   2.10e- 36
```

Combinando `broom` y `flextable`

Aplicación sobre un modelo de regresión lineal

Las librerías que hemos cargado van a trabajar de manera conjunta con las funciones estadísticas de R base. Veamos como la integración de todo lo aprendido nos lleva a una tabla que puede formar parte de un documento académico.

$$ingreso = \alpha + \beta X_i + \epsilon \qquad i \in \{caracteristicas\ individuo\}$$

Creación del modelo

```
modelo <- lm(formula = ingresos ~
               h_trabajo +
               nivel_inst +
               a_trabajo +
               edad +
               h_trabajo,
             data = hombres)

summary(modelo)
```

```
Call:
lm(formula = ingresos ~ h_trabajo + nivel_inst + a_trabajo +
    edad + h_trabajo, data = hombres)

Residuals:
    Min      1Q  Median      3Q     Max
-1545.0  -148.1   -26.8    78.9  8444.4

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                           -329.37769   37.69626  -8.738  < 2e-16 ***
h_trabajo                               7.84525    0.37737  20.789  < 2e-16 ***
nivel_instCentro de alfabetización    -18.27052  134.15657  -0.136  0.89168
nivel_instPrimaria                     80.75105   29.89224   2.701  0.00692 **
nivel_instEducación básica            186.31843   35.58931   5.235 1.69e-07 ***
nivel_instSecundaria                  227.45103   30.34192   7.496 7.22e-14 ***
nivel_instEducación media             226.88699   32.93886   6.888 6.05e-12 ***
nivel_instSuperior no universitaria   451.38945   38.95794  11.587  < 2e-16 ***
nivel_instSuperior universitaria      556.81564   31.16866  17.865  < 2e-16 ***
nivel_instPost-grado                 1195.14608   40.69432  29.369  < 2e-16 ***
a_trabajo                               0.06677    0.49028   0.136  0.89168
edad                                    5.79665    0.45790  12.659  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 370.6 on 8563 degrees of freedom
Multiple R-squared:  0.2913,    Adjusted R-squared:  0.2904
F-statistic:   320 on 11 and 8563 DF,  p-value: < 2.2e-16
```

For loop para formatear las columnas del `data.frame`

```
tabla_mod <- broom::tidy(modelo, conf.int=TRUE)

for(i in 2:5){
  tabla_mod[,i] <- round(tabla_mod[[i]],3)
}

### Print de pantalla del comando

print(tabla_mod)
```

Dando formato a la tabla

Aquí estamos creando una nueva variable con el objetivo de poner el nivel de significancia mediante el uso de asteriscos.

Así también, estamos renombrando las variables.

```
# Poner asteriscos (*)
ind <- tabla_mod$p.value< 0.05
tabla_mod$p.value[ind] <- paste(tabla_mod$p.value[ind],"***")

# Renombrar las variables
tabla_mod <- tabla_mod[,-3]
names(tabla_mod) <- c("Parámetro", "Valor estimado", "Valor Z", "P-valor", "Límite inferio

# Transformar al objeto flextable
tabla_mod <- flextable(tabla_mod)

# Ajustar la tabla
autofit(tabla_mod)
```

Resultado final de la tabla

Parámetro	Valor estimado	Valor Z	P-valor	Límite inferior	Límite superior
(Intercept)	-329.378	-8.738	0 ***	-403.271446	-255.483933
h_trabajo	7.845	20.789	0 ***	7.105512	8.584989
nivel_instCentro de alfabetización	-18.271	-0.136	0.892	-281.249745	244.708697
nivel_instPrimaria	80.751	2.701	0.007 ***	22.155044	139.347055
nivel_instEducación básica	186.318	5.235	0 ***	116.554806	256.082057
nivel_instSecundaria	227.451	7.496	0 ***	167.973550	286.928509
nivel_instEducación media	226.887	6.888	0 ***	162.318880	291.455091
nivel_instSuperior no universitaria	451.389	11.587	0 ***	375.022494	527.756399
nivel_instSuperior universitaria	556.816	17.865	0 ***	495.717560	617.913718
nivel_instPost-grado	1195.146	29.369	0 ***	1115.375407	1274.916756
a_trabajo	0.067	0.136	0.892	-0.894292	1.027831
edad	5.797	12.659	0 ***	4.899058	6.694236

Funciones varias: `lapply`

La función `lapply` necesita como insumo una lista y aplica una función determinada sobre cada elemento de esa lista.

Cuando uso `lapply` en una data.frame, cada columna se convierte en el elemento de una lista.

Esta función devuelve una lista de la misma longitud que `X`.

```
#Estructura

lapply(X = nombre_lista, # Indicar la base de datos
       FUN = funcion_a_aplicar) # Especifica la función por aplicar
```

Ejemplo de aplicación: `lapply`

La lista contiene 5 edades y 6 variables de ingreso.

Necesitamos sumar los elementos que están dentro de cada lista:

```
x <- list("edad"=10:15,
          "ingresos"=seq(400,800, by=100))

b <- lapply(X = x, #nombre lista
            FUN = sum)  #función
b
```

```
## $edad
## [1] 75
##
## $ingresos
## [1] 3000
```

Necesitamos sacar el promedio los elementos de cada lista:

```
y <- list("edad"=10:15,
          "ingresos"=seq(400,800, by=100))

c <- lapply(y,    #nombre lista
            mean)      #función
c
```

```
## $edad
## [1] 12.5
##
## $ingresos
## [1] 600
```

Ejemplo de aplicación: `lapply`

Hacer un **resumen** estadístico de los componenetes de la lista `y`.

```
lapply(y, summary)
```

```
## $edad
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   11.25   12.50   12.50   13.75   15.00
##
## $ingresos
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     400     500     600     600     700     800
```

Ejemplo de aplicación: `lapply`

Ver la **clase** de cada elemento que compone el data.frame `iris`:

```
# "iris" es una base de datos que tiene R por default
lapply(iris, class)
```

```
## $Sepal.Length
## [1] "numeric"
##
## $Sepal.Width
## [1] "numeric"
##
## $Petal.Length
## [1] "numeric"
##
## $Petal.Width
## [1] "numeric"
##
## $Species
## [1] "factor"
```


Let's do this!

Brecha salarial por sexo

Introducción

Es hora de ver una aplicación con una base de datos real. El caso de estudio nos ayudará a recorrer los elementos tratados en clase.


Congratulations

Hasta ahora hemos aprendido:

1. Acerca de los elementos básicos con los que trabaja R
2. Comprender la estructura de una tabla de texto
3. Principios de funciones y librerías
4. Liberías para dar formato a los resultados
5. Análisis estadístico de los resultados


What next?

Qué veremos en las próximas semanas?

1. Imputación de base de datos
2. Contrastar un modelo
3. Programación funcional


THE
END

Recursos

- INEC: ENEMDU diciembre, 2019
- Yaml
- Usar Rmarkdown
- R for Data Science: Functions