



# Taller R – Rifa Valentina

## Exportar a excel controlando formatos

Nicolás Ratto

2022/02/02

# Exportar archivos a excel

Hay muchas formas de exportar tablas y bases de datos desde R a Excel.

El proceso resulta fundamental cuando trabajamos en equipos y no todos manejan el lenguaje de R.

El universo `tidyverse` no cuenta con paquetes para exportar datos a excel en formato `.xlsx`.

La forma quizás más sencilla y difundida de hacerlo es con el paquete `writexl` del universo de R Open Science: <https://ropensci.org/>

La sintaxis es sencilla y corta, pero limitada.

# Exportación con writexl

En primer lugar creemos una base de datos ficticia:

```
base1<-as.data.frame(cbind(  
  a=c(1:10),  
  b=c(1:10)  
))  
  
head(base1,2)
```

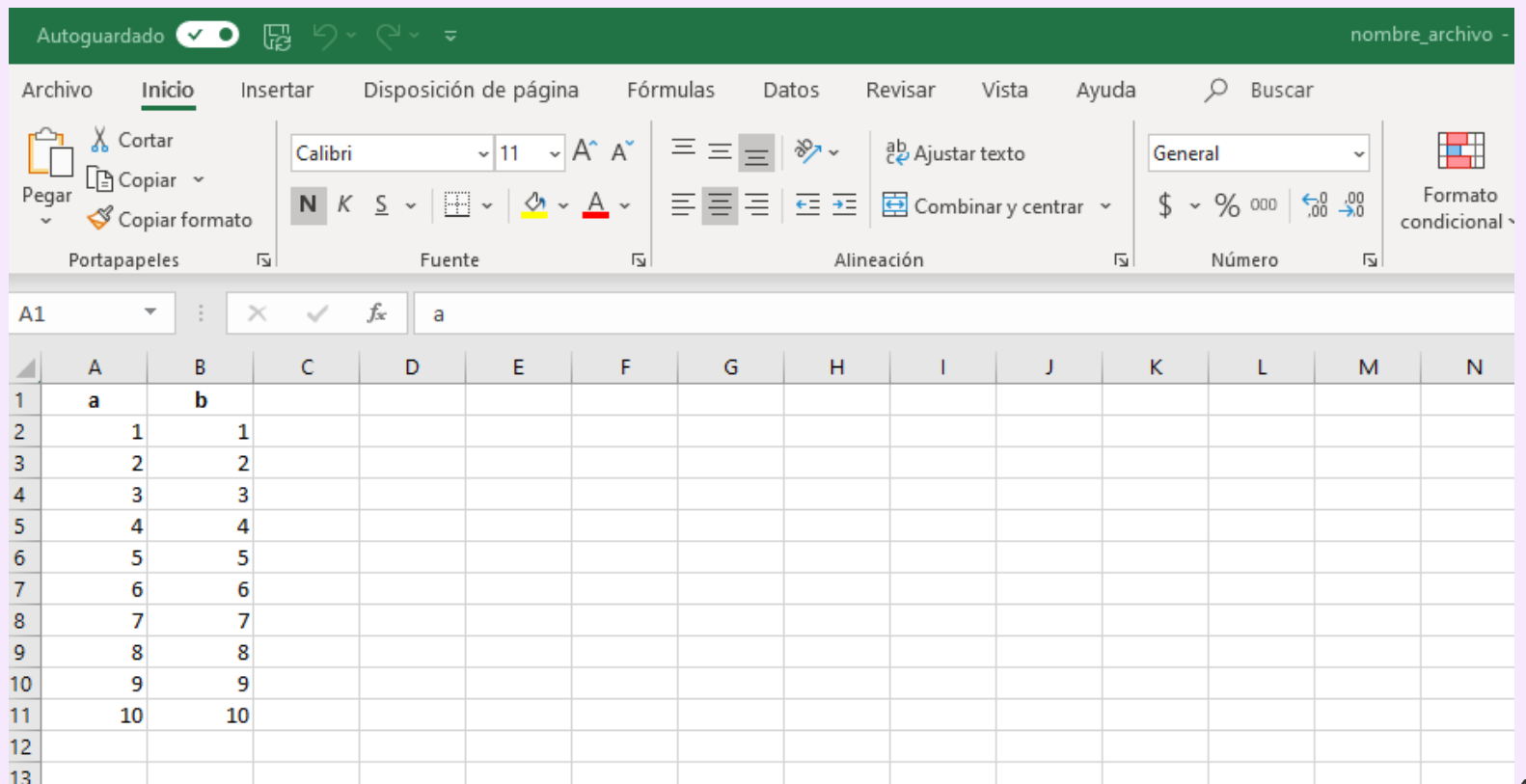
```
##    a b  
## 1 1 1  
## 2 2 2
```

Para exportar la base de datos solamente hay que cargar el paquete writexl y especificar dos argumentos en la función write\_xlsx

```
writexl::write_xlsx(base1,      # objeto a exportar (data frame)  
  "output/nombre_archivo.xlsx") # nombre de archivo a crear
```

# Exportación con writexl

Por defecto, `write_xlsx` considerará en la exportación el nombre de las columnas y las pondrá centradas y en **negrito**.



# Exportación con writexl

¿Como evitarlo?

Existen dos argumentos extra con los que se puede "jugar" en la función `write_xlsx`: `col_names` y `format_headers`.

Quitar nombres de columnas:

```
write_xlsx(base1,                # objeto a exportar (data frame)
  "output/nombre_archivo2.xlsx", # nombre de archivo a crear
  col_names = FALSE,             # Base sin nombres de columnas
)
```

Dejar nombre de columnas pero quitarles el formato

```
write_xlsx(base1,                # objeto a exportar (data frame)
  "output/nombre_archivo3.xlsx", # nombre de archivo a crear
  col_names = TRUE,               # Base con nombres de columnas
  format_headers = FALSE         # Nombres de columnas sin formato
)
```

# Exportación con writexl

Una última posibilidad a revisar con `write_xlsx` tiene que ver con el exportar más de una base de datos en un mismo archivo (un archivo por pestaña).

Para esto, las distintas bases de datos a exportar deben agruparse en una lista (`list()`).

```
## Se crean dos nuevas bases de datos
base2<-as.data.frame(cbind(c=c(1:10),d=c(1:10)))
base3<-as.data.frame(cbind(e=c(1:10),f=c(1:10)))
```

```
write_xlsx(list(base1,base2,base3), # Bases en una lista.
            "output/nombre_archivo4.xlsx", # nombre de archivo a crear
            )
```

# Límites de writexl

En base a los cuatro argumentos de `writeln`, ¿cómo podemos crear un archivo `excel` como el siguiente?

N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	
Po	Jer	Pond	Pond	NT(t-1)	NT(t)	ROHO(t-1)	ROHO(t)	REHE(t-1)	REHE(t)	HONT(t-1)	HONT(t)	Var_ROHC	Var_CHT(t)	Dif_NT(t)	Dif_NT(t)	Brecha	CH	Tra	Ve	Verific	COD1	COD2	COD3	NT	Observación
1	11	0,49	11	28	27	8.645	10.778			183	184	24,68	24,43	-1	-2	-0,25	NC	1							
0	17	0,35	17	5	5	7.941	9.741			186	174	22,68	22,09	0	0	-0,59	NC	1							
1	4	0,79	4	13	13	8.031	9.647			188	184	20,12	20,37	0	0	0,25	NC	1							
0	14	0,43	15	6	6	7.613	10.841	4.417		166	193	42,4	39,55	0	0	-2,85	NC	1							
0	22	0,01	22	34	35	10.148	12.413			179	170	22,32	20,48	1	3	-1,84	SI	1							
0	28	0,01	28	4	4	2.441	2.594	3.889	5.367	194	157	6,23	34,71	0	0	28,48	NC	1							
0	22	0,01	22	10	9	2.301	3.252	4.066	3.946	175	185	41,32	28,5	-1	-4	-12,82	NC	1							
0	36	0	34	1	1	2.165	3.085			194	194	42,5	47,18	0	0	4,68	NC	1							
0	16	0,01	15	2	2	1.232	1.302			97	97	5,7	26,25	0	0	20,55	NC	1							
0	11	0,02	11	5	5	60.622	39.261			194	194	-35,24	-34,44	0	0	0,8	NC	1							
0	31	0,09	32	3	3	25.561	33.803			194	194	32,24	30,58	0	0	-1,66	NC	1							
0	26	0,06	25	1	1	3.088	2.379			194	194	-22,95	-22,95	0	0	0	NC	1							
0	8	0,52	8	320	439	2.835	2.730	2.621	2.601	168	148	-3,72	-1,19	119	256	2,53	NC	1							
0	24	0,07	24	2	3	3.189	4.803		6.076	94	168	50,59	51,17	1	2	0,58	NC	1							
0	21	0,09	21	24	35	4.010	3.155	2.497	2.543	140	151	-21,32	-12,78	11	24	8,54	NC	1							
0	24	0,22	22	1	1	2.101	3.115	2.424	2.643	193	193	48,29	41,01	0	0	-7,28	NC	1							
1	1	1,25	1	732	991	3.414	3.282	4.474	4.819	139	151	-3,86	-4,27	259	426	-0,41	NC	1							
0	40	0,03	40	1	1	2.306	2.126			194	32	-7,82	-20,8	0	0	-12,98	SI	1							
0	16	0,14	16	1	1	2.216	2.662		2.917	194	167	20,11	12,54	0	0	-7,57	NC	1							
0	5	0,12	6	4	3	25.393	30.479			189	189	20,03	20,03	-1	-1	0	NC	1							

No se puede. Hay que recurrir a otros paquetes, como `openxlsx`, para

- Definir formato de celdas (color, color condicional, %, carácter, etc.)
- Pegar más de una base de datos por pestañas
- Inmovilizar paneles y otras muchas cosas...

# El paquete openxlsx

En su forma más simple el código resulta más complejo que el de writexl:

```
library(openxlsx)

wb <- createWorkbook() ## Se crea libro de trabajo (wb)

addWorksheet(wb, ## Al objeto wb le damos una pestaña
  sheetName = "pestaña1",
  gridLines = FALSE)

writeData(wb = wb, ## En el libro wb
  sheet = "pestaña1", ## En su pestaña1
  x = base1) ## Cargamos el objeto base1

saveWorkbook(wb = wb, ## Exportamos el libro
  file = "output/nombre_archivo5.xlsx",
  overwrite = TRUE)
```

Más complejo, pero vale la pena para tener más control sobre los libros excel que crearemos a futuro.

Lo más simple es copiar y pegar este código cuando quieran exportar.



# El paquete openxlsx

Definir posición exacta de objetos en cada pestaña

Una gran ventaja de openxlsx es que en una misma pestaña de excel pueden agregarse dos o más data frames.

Esto se logra señalando la posición exacta de cada una.

La primera:

```
wb2 <- createWorkbook()  ## Se crea libro de trabajo 2 (wb2)

addWorksheet(wb2,        ## Al objeto wb2 le damos una pestaña
  sheetName = "pestaña1",
  gridLines = FALSE)

writeData(wb = wb2,      ## En el libro wb2
  sheet = "pestaña1",    ## En su pestaña1
  x = base1)             ## Cargamos el objeto base1

## Por defecto, base1 se ubica en posición (1,1).
```

# El paquete openxlsx

Y para la segunda...

Se puede determinar manualmente su posición:

```
writeData(wb = wb2,  
          sheet = "pestaña1",  
          startCol = 4,    ## Pegar nuevamente base 1 en col 4.  
          x = base1)
```

# El paquete openxlsx

Queda algo así:

	A	B	C	D	E	F	G	H
1	a	b		a	b			
2		1	1		1	1		
3		2	2		2	2		
4		3	3		3	3		
5		4	4		4	4		
6		5	5		5	5		
7		6	6		6	6		
8		7	7		7	7		
9		8	8		8	8		
10		9	9		9	9		
11		10	10		10	10		
12								

# El paquete openxlsx

Se puede volver agregar la base1 (u otra data frame) en cualquier otra posición de la pestaña1.

```
## Por ejemplo, pegar base1 nuevamente en la columna 8 y fila 20.  
writeData(wb = wb2,  
          sheet = "pestaña1",  
          startCol = 8,  
          startRow = 20,  
          x = base1)
```

```
saveWorkbook(wb = wb2,      ## Exportamos el libro  
             file = "output/nombre_archivo12.xlsx",  
             overwrite = TRUE)
```

# El paquete openxlsx

A continuación se revisan algunos comandos que se pueden agregar entre la creación del objeto wb (Work Book) y su exportación.

Estas modificaciones al libro de trabajo se van agregando progresivamente.

## Configurar la anchura de las columnas

Crear libro, pestaña y agregar data a pestaña:

```
## Se repite el proceso de crear WB, dar pestaña y cargar objeto:  
wb <- createWorkbook()
```

```
addWorksheet(wb,  
  sheetName = "pestaña1",  
  gridLines = TRUE)
```

```
writeData(wb = wb,  
  sheet = "pestaña1",  
  x = base1)
```

# El paquete openxlsx

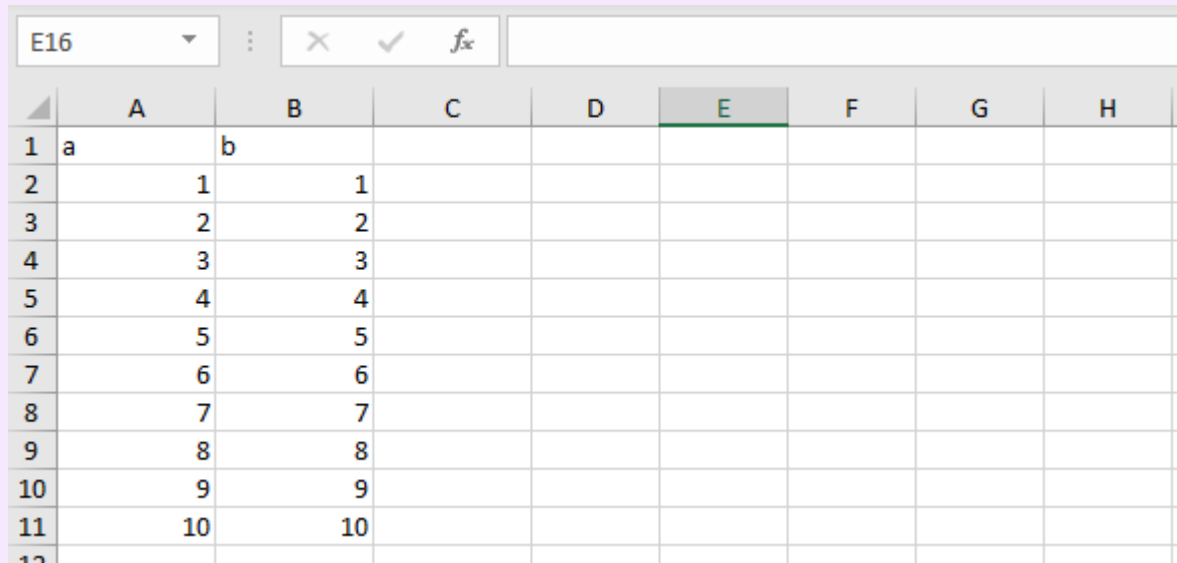
Definir la anchura de tres columnas

```
setColWidths(wb,  
             sheet = 1,  
             cols = c(1,2,3), ## Columnas a ensanchar  
             widths = 10)    ## Anchura
```

Exportar el libro

# El paquete openxlsx

Queda así:



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H
1	a	b						
2	1	1						
3	2	2						
4	3	3						
5	4	4						
6	5	5						
7	6	6						
8	7	7						
9	8	8						
10	9	9						
11	10	10						

# El paquete openxlsx

## Inmovilizar paneles

```
freezePane(wb,                                ## Libro de trabajo
            sheet= "pestaña1",                 ## Pestaña en la que se aplica
            firstRow = TRUE,                    ## Inmovilizar fila superior
            firstCol = TRUE)                   ## Inmovilizar primera columna
```

Queda así:

	A	B	C	D	E	F	G	H	I
1	a	b							
5	4	4							
6	5	5							
7	6	6							
8	7	7							
9	8	8							
10	9	9							
11	10	10							



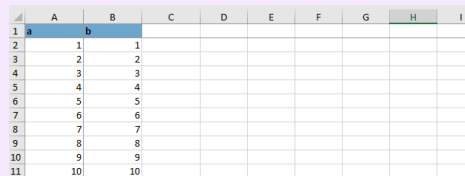
# El paquete openxlsx

## Aplicar colores y estilos

Se requieren dos pasos: crear el objeto "estilo" y aplicar el objeto "estilo" al libro de trabajo (wb)

```
## Se crea estilo de celda azul, letra negras y ennegrecidas
blue_bold <- createStyle(fontColour = "black",      ## Color letra
                        bgFill = "skyblue3",      ## Color celda
                        textDecoration = "bold")  ## Estilo de la letra
```

```
# Aplicar colores a nombres de columnas
conditionalFormatting(wb, sheet = "pestaña1",      ## libro y pestaña
                    cols = 1:2, ## columna
                    rows = 1,  ## fila
                    rule = "!=0", ## Condición
                    style = blue_bold) ## Aplicar el estilo
```



	A	B	C	D	E	F	G	H	I
1	1	2							
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									

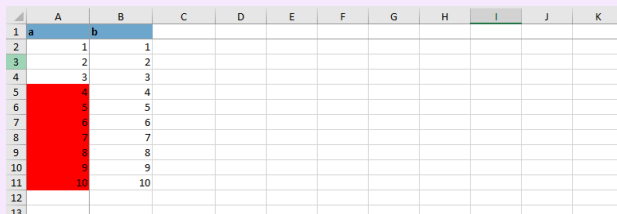
# El paquete openxlsx

## Aplicar colores y estilos

También se pueden pintar las celdas si cumplen condición

```
## Se crea estilo de celda azul, letra negras y ennegrecidas
red <- createStyle(fontColour = "black",      ## Color letra
                  bgFill = "red")           ## Color celda
```

```
# Aplicar colores a nombres de columnas
conditionalFormatting(wb, sheet = "pestaña1", ## libro y pestaña
                    cols = 1, ## solo columna 1
                    rows = 2:11, ## todas las filas, menos la primera
                    rule = ">3", ## Condición mayor a 3
                    style = red) ## Aplicar el estilo
```



	A	B	C	D	E	F	G	H	I	J	K
1	a	b									
2	1	1									
3	2	2									
4	3	3									
5	4	4									
6	5	5									
7	6	6									
8	7	7									
9	8	8									
10	9	9									
11	10	10									
12											
13											

# El paquete openxlsx

## Aplicar porcentajes

Lógica similar a la de los colores. Primero crear un estilo "porcentaje" y luego se aplica.

```
## Se crea estilo porcentaje (con un decimal)
pct = createStyle(numFmt="0.0%")

## Se aplica estilo porcentaje a las celdas deseadas
addStyle(wb,
  sheet = 1,
  style=pct,
  cols=2,
  rows=3:4,
  gridExpand=TRUE)
```

	A	B	C	D	E	F	G	H
1	a	b						
2	1	1						
3	2	200,0%						
4	3	300,0%						
5	4	4						
6	5	5						
7	6	6						
8	7	7						

# El paquete openxlsx

## Hipervínculos

```
## Creamos nueva pestaña sin agregar objeto  
addWorksheet(wb,  
             sheetName = "pestaña2",  
             gridLines = FALSE)
```

```
## Creamos hipervínculo que nos lleve a la pestaña 2  
writeFormula(wb, ## En libro WB  
             "pestaña1", ## pestaña1  
             startRow = 6, ## En la fila 6, aplica hipervínculo:  
  
             x = makeHyperlinkString(sheet = "pestaña2", ## Lleva a pestaña2  
                                     row = 1, ## Fila 1  
                                     col = 1, ## Columna 1  
                                     text = "ACA LINK")) ## Texto del link
```

# Referencias y materiales de consulta

Formating with xlsx

Manual del paquete openxlsx.

Identificación de colores en una imagen.

Colores en R y códigos de colores.