¿Todo esto para una tabla?

Curso de introducción a 

Pablo Cabrera-Álvarez

 |  @pablocalv

Julio 2019

🤓 Quiz

¿Cómo transformar la variable `age` en tres grupos (`age_group`) y crear una variable que sea la media de edad para cada grupo de `age_group`?

```
glimpse(gss_cat)
```

```
## Observations: 21,483
## Variables: 9
## $ year    <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 20…
## $ marital <fct> Never married, Divorced, Widowed, Never married, Divorce…
## $ age     <int> 26, 48, 67, 39, 25, 25, 36, 44, 44, 47, 53, 52, 52, 51, …
## $ race    <fct> White, White, White, White, White, White, White, White, …
## $ rincome <fct> $8000 to 9999, $8000 to 9999, Not applicable, Not applic…
## $ partyid <fct> "Ind,near rep", "Not str republican", "Independent", "In…
## $ relig   <fct> Protestant, Protestant, Protestant, Orthodox-christian, …
## $ denom   <fct> Southern baptist, Baptist-dk which, No denomination, Not…
## $ tvhours <int> 12, NA, 2, 4, 1, NA, 3, NA, 0, 3, 2, NA, 1, NA, 1, 7, NA…
```

🤓 Quiz

```
gss_cat %>%
  mutate(age_group = case_when(
    between(age, min(age), 29) ~ "18-29",
    between(age, 30, 64) ~ "30-64",
    TRUE ~ "65+" )) %>%
  group_by(age_group) %>%
  mutate(mean_age = mean(age, na.rm = TRUE)) %>%
  select(age, age_group, mean_age) %>%
  head()
```

```
## # A tibble: 6 x 3
## # Groups:   age_group [3]
##     age age_group mean_age
##   <int> <chr>        <dbl>
## 1    26 18-29         24.3
## 2    48 30-64         45.9
## 3    67 65+           74.4
## 4    39 30-64         45.9
## 5    25 18-29         24.3
## 6    25 18-29         24.3
```

Frecuencias

Frecuencias en R

`table(x, y)`

- En R existe una función base para realizar **tablas de frecuencias** y tablas cruzadas (`table()`):

```
table(gss_cat$race)
```

```
##
##         Other         Black         White Not applicable
##          1959          3129         16395             0
```

- Sin embargo, ya existen paquetes con los que **producir tablas** de frecuencias como `sjmisc`.

Frecuencias en `sjmisc`

`frq(x, sort.frq, weights, show.na, out, title, file)`

- La función `frq()` permite crear tablas de frecuencias a partir de uno o varios factores.

```
frq(x = gss_cat, race)
```

```
##
## race <categorical>
## # total N=21483  valid N=21483  mean=2.67  sd=0.63
##
##              val   frq raw.prc valid.prc cum.prc
##            Other  1959    9.12      9.12    9.12
##            Black  3129   14.57     14.57   23.68
##            White 16395   76.32     76.32  100.00
##  Not applicable     0    0.00      0.00  100.00
##            <NA>     0    0.00        NA      NA
```

Frecuencias en `sjmisc`

- La función `frq()` cuenta con argumentos de tipo **estadístico**, como la inclusión de pesos (`weights`) y otros para **formatear** la tabla.

```
frq(x = gss_cat, race, sort.frq = "desc", show.na = FALSE, title = "Raza en EE.UU")
```

```
##
## Raza en EE.UU
## # total N=21483  valid N=21483  mean=2.67  sd=0.63
##
##              val   frq raw.prc valid.prc cum.prc
##            White 16395   76.32     76.32   76.32
##            Black  3129   14.57     14.57   90.88
##            Other  1959    9.12      9.12  100.00
##  Not applicable     0    0.00      0.00  100.00
```

Frecuencias en `sjmisc`

- Además, la función `frq()` permite **guardar el output** en un formato distinto a `.txt` (por defecto), utilizando los argumentos `out` y `file`. En `out` se puede establecer que el output sea visualizado en el `"viewer"` o en el `"browser"`. El output se puede guardar en `file`.

```
frq(x = gss_cat, race, sort.frq = "desc", show.na = FALSE, title = "Raza en EE.UU",
    out = "viewer", file = "tablas1.html")
```

Raza en EE.UU

val	*frq*	*raw.prc*	*valid.prc*	*cum.prc*
White	16395	76.32	76.32	76.32
Black	3129	14.57	14.57	90.88
Other	1959	9.12	9.12	100
Not applicable	0	0	0	100
total N=21483 · valid N=21483 · $\bar{x}$=2.67 · σ=0.63				

Frecuencias en *pipes*

- Otra oportunidad que ofrece la función `frq()` es que puede ser utilizada dentro de *pipes*:

```
gss_cat %>%
  select(race, marital) %>%
  frq()
```

Frecuencias en *pipes*

```
##
## race <categorical>
## # total N=21483  valid N=21483  mean=2.67  sd=0.63
##
##              val   frq raw.prc valid.prc cum.prc
##            Other  1959    9.12      9.12    9.12
##            Black  3129   14.57     14.57   23.68
##            White 16395   76.32     76.32  100.00
##   Not applicable     0    0.00      0.00  100.00
##            <NA>      0    0.00        NA      NA
##
##
## marital <categorical>
## # total N=21483  valid N=21483  mean=4.48  sd=1.67
##
##              val   frq raw.prc valid.prc cum.prc
##        No answer    17    0.08      0.08    0.08
##    Never married  5416   25.21     25.21   25.29
##        Separated   743    3.46      3.46   28.75
##         Divorced  3383   15.75     15.75   44.50
##          Widowed  1807    8.41      8.41   52.91
```

Frecuencias con `group_by()`

- La función `frq()` se puede combinar con `group_by()`:

```
gss_cat %>%
  group_by(marital) %>%
  frq(race)
```

```
##
## race <categorical>
## # grouped by: No answer
## # total N=17  valid N=17  mean=2.65  sd=0.70
##
##             val frq raw.prc valid.prc cum.prc
##           Other   2   11.76     11.76   11.76
##           Black   2   11.76     11.76   23.53
##           White  13   76.47     76.47  100.00
##  Not applicable   0    0.00      0.00  100.00
##             <NA>   0    0.00        NA      NA
##
##
## race <categorical>
## # grouped by: Never married
```

Frecuencias con *labelled*

- Otra característica de `frq()` es que funciona con objetos de tipo *labelled*:

```
frq(x = cis, P1)
```

```
##
## Valoración de la situación económica general de España (P1) <numeric>
## # total N=2487  valid N=2487  mean=3.60  sd=0.93
##
##  val     label  frq raw.prc valid.prc cum.prc
##    1 Muy buena    6    0.24      0.24    0.24
##    2     Buena  185    7.44      7.44    7.68
##    3   Regular 1060   42.62     42.62   50.30
##    4      Mala  840   33.78     33.78   84.08
##    5  Muy mala  376   15.12     15.12   99.20
##    8      N.S.   19    0.76      0.76   99.96
##    9      N.C.    1    0.04      0.04  100.00
##   NA        NA    0    0.00        NA      NA
```

Descriptivos

Descriptivos en R base

`sd(x)` | `min(x)` | `max(x)`

- Existen fórmulas para calcular los descriptivos en R base:

```
desc <- c(mean(gss_cat$age, na.rm = TRUE), sd(gss_cat$age, na.rm = TRUE), min(gss_cat$age, na.rm
names(desc) <- c("Media", "Desv. Típica", "Min.", "Max.")
desc
```

```
##        Media Desv. Típica         Min.         Max.
##     47.18008     17.28750     18.00000     89.00000
```

Descriptivos en `sjmisc`

`descr(x, weights, out)`

- Con la función `descr()` se puede realizar una tabla de **estadísticos descriptivos**:

```
descr(x = gss_cat)
```

```
##
## ## Basic descriptive statistics
##
##      var        type   label     n NA.prc    mean    sd   se   md trimmed
##     year     integer    year 21483   0.00 2006.50  4.45 0.03 2006 2006.38
##  marital categorical marital 21483   0.00    4.48  1.67 0.01    5    4.61
##      age     integer     age 21407   0.35   47.18 17.29 0.12   46   46.30
##     race categorical    race 21483   0.00    2.67  0.63 0.00    3    2.83
##  rincome categorical rincome 21483   0.00    8.93  5.51 0.04    6    8.78
##  partyid categorical partyid 21483   0.00    7.14  2.10 0.01    7    7.23
##    relig categorical   relig 21483   0.00   13.59  2.45 0.02   15   14.14
##    denom categorical   denom 21483   0.00   21.21 10.74 0.07   25   22.39
##  tvhours     integer tvhours 11337  47.23    2.98  2.59 0.02    2    2.58
##           range  skew
```

Descriptivos en `sjmisc`

- También con esta función se pueden realizar diferentes tipos de **output**:

```
descr(x = gss_cat, out = "browser")
```

Descriptivos con `group_by()`

- La función `descr()` también se puede combinar con otras funciones tidyverse:

```
gss_cat %>%
  group_by(race) %>%
  descr(age)
```

Descriptivos con `group_by()`

```
##
## ## Basic descriptive statistics
##
##
## Grouped by: Other
##
##  var    type label    n NA.prc  mean    sd   se md trimmed      range skew
##  age integer   age 1951   0.41 39.48 14.39 0.33 37   38.12 71 (18-89) 0.84
##
##
## Grouped by: Black
##
##  var    type label    n NA.prc mean    sd   se md trimmed      range skew
##  age integer   age 3115   0.45 43.9 16.06 0.29 42   42.82 71 (18-89) 0.52
##
##
## Grouped by: White
##
##  var    type label     n NA.prc  mean   sd   se md trimmed      range skew
##  age integer   age 16341   0.33 48.72 17.5 0.14 48   48.03 71 (18-89) 0.28
```

Tablas de contingencia

Tablas en R base

- La función `table()` de `R` base sirve para realizar tablas de contingencia:

```
table(gss_cat$marital, gss_cat$race)
```

```
##
##                Other Black White Not applicable
##   No answer        2     2    13              0
##   Never married  633  1305  3478              0
##   Separated      110   196   437              0
##   Divorced       212   495  2676              0
##   Widowed         70   262  1475              0
##   Married        932   869  8316              0
```

Tabla con `sjmisc`

`flat_table(data, ..., margin)`

- En `sjmisc` existe una función para crear tablas cruzadas:

```
flat_table(data = gss_cat, marital, race)
```

```
##                race Other Black White Not applicable
## marital
## No answer             2     2    13              0
## Never married       633  1305  3478              0
## Separated           110   196   437              0
## Divorced            212   495  2676              0
## Widowed              70   262  1475              0
## Married             932   869  8316              0
```

Tabla con `sjmisc`

- La tabla puede contener celdas como porcentajes de columna o de fila en el argumento `margin`: `c("counts", "cell", "row", "col")`

```
flat_table(data = gss_cat, marital, race, margin = "col")
```

```
##                race Other Black White Not applicable
## marital
## No answer          0.10  0.06  0.08           NaN
## Never married     32.31 41.71 21.21           NaN
## Separated          5.62  6.26  2.67           NaN
## Divorced          10.82 15.82 16.32           NaN
## Widowed            3.57  8.37  9.00           NaN
## Married           47.58 27.77 50.72           NaN
```

Tabla con descr

`crosstab(dep, indep, weight, prop.r, prop.c, prop.t, missing.include, format, plot = FALSE)`

- Otra alternativa es utilizar `descr::crosstab()` que permite realizar tablas de contingencia de aspecto similar a las que se hacen en SPSS:

```
descr::crosstab(gss_cat$marital, gss_cat$race, plot = FALSE)
```

Tabla con `descr`

```
##    Cell Contents
## |-------------------------|
## |                   Count |
## |-------------------------|
##
## ================================================
##                     gss_cat$race
## gss_cat$marital    Other   Black   White   Total
## ------------------------------------------------
## No answer              2       2      13      17
## ------------------------------------------------
## Never married        633    1305    3478    5416
## ------------------------------------------------
## Separated            110     196     437     743
## ------------------------------------------------
## Divorced             212     495    2676    3383
## ------------------------------------------------
## Widowed               70     262    1475    1807
## ------------------------------------------------
## Married              932     869    8316   1.012e+04
## ------------------------------------------------
```

Tabla con `descr`

- Se pueden pedir **diferentes proporciones** usando los argumentos `prop.`:

```
descr::crosstab(gss_cat$marital, gss_cat$race,  prop.r = TRUE, plot = FALSE)
```

Tabla con descr

```
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## ==============================================
##                    gss_cat$race
## gss_cat$marital    Other   Black   White   Total
## ----------------------------------------------
## No answer              2       2      13      17
##                    11.8%   11.8%   76.5%    0.1%
## ----------------------------------------------
## Never married        633    1305    3478    5416
##                    11.7%   24.1%   64.2%   25.2%
## ----------------------------------------------
## Separated            110     196     437     743
##                    14.8%   26.4%   58.8%    3.5%
## ----------------------------------------------
## Divorced             212     495    2676    3383
##                    6.3%   14.6%   79.1%   15.7%
```

Tabla con `descr`

- Existe la opción de que, además de la tabla, se realicen test **chi cuadrado**:

```
descr::crosstab(gss_cat$marital, gss_cat$race,  chisq = TRUE, plot = FALSE)
```

Tabla con `descr`

```
## Warning in chisq.test(tab, correct = FALSE, ...): Chi-squared approximation
## may be incorrect

##    Cell Contents
## |-------------------------|
## |                   Count |
## |-------------------------|
##
## ==========================================================
##                    gss_cat$race
## gss_cat$marital    Other   Black   White   Total
## ----------------------------------------------------------
## No answer              2       2      13      17
## ----------------------------------------------------------
## Never married        633    1305    3478    5416
## ----------------------------------------------------------
## Separated            110     196     437     743
## ----------------------------------------------------------
## Divorced             212     495    2676    3383
## ----------------------------------------------------------
## Widowed               70     262    1475    1807
## ----------
```

Un paquete integral de tablas: expss

Tablas con exspss

`apply_labels(...)`

- El paquete `exspss` ofrece una serie de funciones que permiten realizar tablas personalizadas: frecuencias, contingencia y personalizadas
- Necesidad de definir las etiquetas de los valores y las variables con `apply_labels()`

```
cis <- apply_labels(cis,
                    CCAA = "Comunidad autónoma",
                    TAMUNI = "Tamaño hábitat",
                    P1 = "Valoración economía",
                    P2 = "Valoración economía retrospectiva",
                    P29 = "Género",
                    P30 = "Edad",
                    ESTUDIOS = "Estudios")
```

Frecuencias

Tablas de frecuencias

`fre(x, weight, drop_unused_labels)`

```
fre(x = cis$P1)
```

Valoración economía	Count	Valid percent	Percent	Responses, %	Cumulative responses, %
Muy buena	6	0.2	0.2	0.2	0.2
Buena	185	7.4	7.4	7.4	7.7
Regular	1060	42.6	42.6	42.6	50.3
Mala	840	33.8	33.8	33.8	84.1
Muy mala	376	15.1	15.1	15.1	99.2
N.S.	19	0.8	0.8	0.8	100.0
N.C.	1	0.0	0.0	0.0	100.0
#Total	2487	100	100	100	
<NA>	0		0.0		

Tablas de contingencia

Tablas de contingencia

`cro(cell_vars, col_vars, row_vars, weight, total_label, total_row_position)`

- Crear una tabla de contingencia sencilla de **recuentos** se puede hacer con `cro()`:

```
cro(cell_vars = cis$P1, col_vars = cis$P29, total_label = "Total")
```

Tablas de contingencia

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	2	4
Buena	113	72
Regular	545	515
Mala	382	458
Muy mala	151	225
N.S.	10	9
N.C.	1	
#Total	1204	1283

Tablas de contingencia

`cro_cpct(cell_vars, col_vars, row_vars, weight, total_label, total_row_position)`

- Crear una tabla de contingencia con **porcentajes de columna** se puede hacer con `cro_cpct()`:

```
cro_cpct(cell_vars = cis$P1, col_vars = cis$P29, total_label = "Total")
```

Tablas de contingencia

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	0.2	0.3
Buena	9.4	5.6
Regular	45.3	40.1
Mala	31.7	35.7
Muy mala	12.5	17.5
N.S.	0.8	0.7
N.C.	0.1	
#Total	1204	1283

Tablas de contingencia

`cro_rpct(cell_vars, col_vars, row_vars, weight, total_label, total_row_position)`

- Crear una tabla de contingencia con **porcentajes de fila** se puede hacer con `cro_rpct()`:

```
cro_rpct(cell_vars = cis$P1, col_vars = cis$P29, total_row_position =  "none")
```

Tablas de contingencia

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	33.3	66.7
Buena	61.1	38.9
Regular	51.4	48.6
Mala	45.5	54.5
Muy mala	40.2	59.8
N.S.	52.6	47.4
N.C.	100.0	

Tablas personalizadas

Tablas personalizadas

`tab_cells(...)` | `tab_stat_cases(total_label, total_row_position, label)` | `tab_pivot()`

Para construir una tabla personalizada hay que encadenar funciones de tipo `tab_()`. Los **elementos imprescindibles** para construir una tabla son:

- El **data frame** necesario para construir la tabla.
- Las **variables en filas** se incluyen en la función `tab_cells()`.
- El **contenido de la tabla** (*por ej.* recuentos, porcentajes, estadísticos descriptivos) se establece en con una función del tipo `tab_stat_()`.
- La función `tab_pivot()` para crear la tabla.

```
data %>%
  tab_cells(var1) %>%
  tab_stat_cases() %>%
  tab_pivot()
```

Un ejemplo...

```
cis %>%
  tab_cells(P1, P2) %>%
  tab_stat_cases() %>%
  tab_pivot()
```

Un ejemplo...

	#Total
Valoración economía	
Muy buena	6
Buena	185
Regular	1060
Mala	840
Muy mala	376
N.S.	19
N.C.	1
#Total cases	2487
Valoración economía retrospectiva	
Mejor	585
Igual	1267

Añadir columnas

`tab_cols(vars)`

- Para **añadir columnas** solo es necesari incluir en el pipe la función `tab_cols()`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_cases() %>%
  tab_pivot()
```

Añadir columnas

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	2	4
Buena	113	72
Regular	545	515
Mala	382	458
Muy mala	151	225
N.S.	10	9
N.C.	1	
#Total cases	1204	1283

Añadir super filas

`tab_rows(data)`

- Existe la posibilidad de añadir **filas adicionales** con la función `tab_rows()`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_rows(P29) %>%
  tab_stat_cases() %>%
  tab_pivot()
```

Añadir super filas

			Género	
			Hombre	Mujer
Género				
Hombre	Valoración economía	Muy buena	2	
		Buena	113	
		Regular	545	
		Mala	382	
		Muy mala	151	
		N.S.	10	
		N.C.	1	
		#Total cases	1204	
Mujer	Valoración economía	Muy buena		4
		Buena		72

Añadir porcentajes de fila o columna

`tab_stat_rpct(label)` | `tab_stat_cpct(label)`

- Dos funciones usuales son las que sirven para calcular los **porcentajes de filas y columnas**:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_rpct() %>%
  tab_pivot()
```

Añadir porcentajes de fila o columna

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	33.3	66.7
Buena	61.1	38.9
Regular	51.4	48.6
Mala	45.5	54.5
Muy mala	40.2	59.8
N.S.	52.6	47.4
N.C.	100.0	
#Total cases	1204	1283

Añadir estadísticos descriptivos

`tab_stat_mean(label)` | `tab_stat_median(label)` | `tab_stat_se(label)` | `tab_stat_sum(label)`

- También es posible añadir **estadísticos descriptivos** para variables de escala:

```
cis %>%
  tab_cells(P30) %>%
  tab_cols(P29) %>%
  tab_stat_mean() %>%
  tab_pivot()
```

	Género	
	Hombre	Mujer
Edad		
Mean	49.1	51.5

Añadir estadísticos personalizados

`tab_stat_fun(..., method)`

- La función `tab_stat_fun()` permite crear **combinaciones personalizadas** de contenido para la tabla:

```
cis %>%
  tab_cells(P30) %>%
  tab_cols(P29) %>%
  tab_stat_fun(Mean = w_mean, "Std. dev." = w_sd, "Valid N" = w_n, method = list) %>%
  tab_pivot()
```

Añadir estadísticos personalizados

	Género					
	Hombre			Mujer		
	Mean	Std. dev.	Valid N	Mean	Std. dev.	Valid N
Edad	49.1	17.9	1204	51.5	18	1283

Varios estadísticos

- Existe la posibilidad de combinar varios `tab_stat_()`. Para ello se puede manipular la función `tab_pivot()` para decidir el posicionamiento en la tabla. Existen tres opciones: `stat_position = c("outside_rows", "inside_rows", "outside_columns", "inside_columns")`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_cases() %>%
  tab_stat_cpct() %>%
  tab_pivot()
```

Varios estadísticos

Con la opción `"inside_rows"`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_cases() %>%
  tab_stat_cpct() %>%
  tab_pivot(stat_position = "inside_rows")
```

Varios estadísticos

	Género	
	Hombre	Mujer
Valoración economía		
Muy buena	2.0	4.0
	0.2	0.3
Buena	113.0	72.0
	9.4	5.6
Regular	545.0	515.0
	45.3	40.1
Mala	382.0	458.0
	31.7	35.7
Muy mala	151.0	225.0
	12.5	17.5

Varios estadísticos

Con la opción `"outside_columns"`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_cases() %>%
  tab_stat_cpct() %>%
  tab_pivot(stat_position = "outside_columns")
```

Varios estadísticos

	Género			
	Hombre	Mujer	Hombre	Mujer
Valoración economía				
Muy buena	2	4	0.2	0.3
Buena	113	72	9.4	5.6
Regular	545	515	45.3	40.1
Mala	382	458	31.7	35.7
Muy mala	151	225	12.5	17.5
N.S.	10	9	0.8	0.7
N.C.	1		0.1	
#Total cases	1204	1283	1204	1283

Añadir etiquetas a los estadísticos

- Utilizar el argumento `label` de las funciones de tipo `tab_stat_()`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(P29) %>%
  tab_stat_cases(label = "Casos") %>%
  tab_stat_cpct(label = "% Col.") %>%
  tab_pivot(stat_position = "inside_rows")
```

Añadir etiquetas a los estadísticos

		Género	
		Hombre	Mujer
Valoración economía			
Muy buena	Casos	2.0	4.0
	% Col.	0.2	0.3
Buena	Casos	113.0	72.0
	% Col.	9.4	5.6
Regular	Casos	545.0	515.0
	% Col.	45.3	40.1
Mala	Casos	382.0	458.0
	% Col.	31.7	35.7
Muy mala	Casos	151.0	225.0
	% Col.	12.5	17.5

Añadir una columna de totales

- Se puede añadir una columna de totales con `total()`:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(total(), P29) %>%
  tab_stat_cases(label = "Casos") %>%
  tab_stat_cpct(label = "% Col.") %>%
  tab_pivot(stat_position = "inside_rows")
```

Añadir una columna de totales

		#Total	Género	
			Hombre	Mujer
Valoración economía				
Muy buena	Casos	6.0	2.0	4.0
	% Col.	0.2	0.2	0.3
Buena	Casos	185.0	113.0	72.0
	% Col.	7.4	9.4	5.6
Regular	Casos	1060.0	545.0	515.0
	% Col.	42.6	45.3	40.1
Mala	Casos	840.0	382.0	458.0
	% Col.	33.8	31.7	35.7
Muy mala	Casos	376.0	151.0	225.0
	% Col.	15.1	12.5	17.5

Añadir un título a la tabla

`set_caption(caption)`

- Con la función `set_caption()` se pueden incluir títulos:

```
cis %>%
  tab_cells(P1) %>%
  tab_cols(total(), P29) %>%
  tab_stat_cases(label = "Casos") %>%
  tab_pivot(stat_position = "inside_rows") %>%
  set_caption("Tabla 1. Valoración económica según género")
```

Añadir un título a la tabla

Tabla 1. Valoración económica según género

		#Total	Género	
			Hombre	Mujer
Valoración economía				
Muy buena	Casos	6	2	4
Buena	Casos	185	113	72
Regular	Casos	1060	545	515
Mala	Casos	840	382	458
Muy mala	Casos	376	151	225
N.S.	Casos	19	10	9
N.C.	Casos	1	1	
#Total cases	Casos	2487	1204	1283

Exportar a MS Excel

Exportar a Excel

- Para **exportar** las tablas a **MS Excel** se puede utilizar el paquete `openxlsx`.
- Antes de hacer la exportación es necesario **crear un libro de Excel**.
- **Escribir la tabla** en el libro de MS Excel y guardarlo

Crear una tabla y asignar el objeto

```
tabla_pers <- cis %>%
  tab_cells(P1, P2) %>%
  tab_cols(total(), P29, TAMUNI) %>%
  tab_stat_cases(label = "Casos") %>%
  tab_stat_cpct(label = "% Col.") %>%
  tab_pivot(stat_position = "inside_rows")
```

Crear un libro de trabajo y grabar la tabla

`CreateWorkbook()` | `addWorksheet(wb, sheetname)` | `xl_write(obj, wb, sheet)` | `saveWorkbook(wb, file, overwrite)`

- **Crear un libro** y una hoja de cálculo:

```
wb <- createWorkbook()
sh <- addWorksheet(wb = wb, sheetName = "Tables")
```

- **Grabar** el objeto tabla:

```
xl_write(obj = tabla_pers, wb = wb, sheet = sh)
```

- Guardar el **libro de cáculo**

```
saveWorkbook(wb = wb, file = "table1.xlsx", overwrite = TRUE)
```

```
## Note: zip::zip() is deprecated, please use zip::zipr() instead
```

Output en MS Excel

Curso de introducción a R