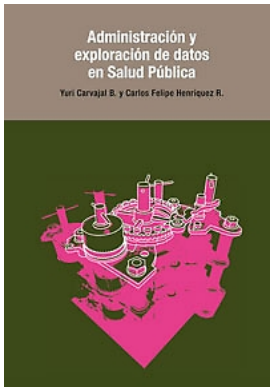
Anexo para el libro Administración y exploración de datos en Salud Pública

Yuri Carvajal B.
Escuela de Salud Pública
Facultad de Medicina
Universidad de Chile

1. Introducción

Este apéndice complementa algunos capítulos del libro y trata de reparar algunas erratas. Informes de errores y sugerencias son bienvenidas en ycarvajal@med.uchile.com



2. Erratas

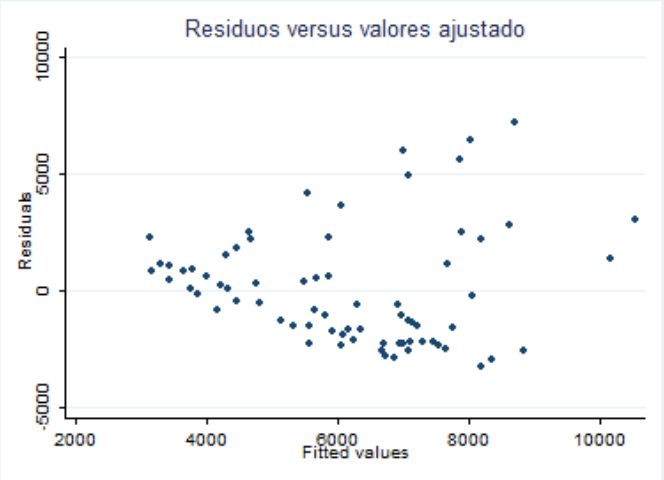
p. 22.
Dice:
Para ir de long a wide

```
reshape wide casos, i (diagnósticos) j (años)
```

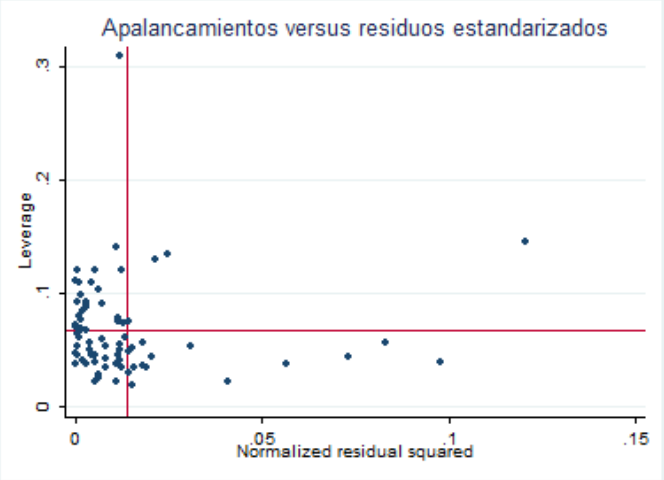
Debe decir:

```
reshape wide frecuencia, i (diagnósticos) j (años)
```

p. 68.
Figura 8.6 el título del gráfico es Residuos versus valores ajustados, así: La figura 8.6 lleva



por título Apalancamientos versus residuos estandarizados, así: La figura 8.8 es redundante



respecto de la 8.6.

3. Exploracion de datos

3.1. clonevar

Una forma alterna a `gen` o `replace` es el comando `clonevar`, que hace una copia de cualquier variable, inlcuyendo sus etiquetas.

3.2. codebook, problems

El comando `codebook,problems` | también es útil pues permite conocer variables que:

- Variables etiquetas con etiquetas no definidas
- Variables que son constantes
- Valores de fecha incompletos(Trailing, trimming, and embedded spaces in string variables)
- Espacio blancos o cortados en variables `string`

3.3. contract

p.23. Un ejemplo de contract sería:

```
contract diag1 , freq(f)
```

En este caso generamos otra base de datos que tiene las frecuencia de todos los diagnósticos. La sintaxis es diferente de collapse, ya que no permite by, exige fijar el parámetro a colapsar y el nombre de la nueva variable como opciones.
p.25

3.4. fweight

Muchos comandos permiten usar `fweight` . Se trata de un comando que pondera los valores. Es importante si usamos datos que ya vienen con frecuencia y requerimos que los promedios sean ponderados. `sum var1 [fweight=var2]`

3.5. Usando contract y collapse para encontrar el máximo dentro de un grupo

Si queremos encontrar las máximas frecuencias de diagnóstico por servicio por ejemplo, podemos usar el comando del contract anterior, agregando el servicio y guardar una base que tenga todas las frecuencias de diagnóstico y los servicios.

```
  use def2013,clear
contract diag1 serv_res, freq(f)
save fserv,replace
```

y luego, usar el comando collapse sobre esta misma base modificada, para guardar sólo las frecuencias máximas por servicio

```
collapse (max)f , by(serv_res)
```

A partir de esta base, podemos pegarla usando la base fserv que hicimos en la primera operación:

```
merge 1:m f serv_res using fserv
```

la salida que nos da Stata es:

```
. merge 1:m f serv_res using fserv

    Result                           # of obs.
    -----------------------------------------
    not matched                        12,786
        from master                         0  (_merge==1)
        from using                     12,786  (_merge==2)

    matched                                30  (_merge==3)
    -----------------------------------------
```

Todos los casos que tienen `_merge==3` son entonces las máximas frecuencias.

4. Algo más de etiquetas

Para saber las etiquetas que tienen nuestras variables, podemos llamar al libro de etiquetas con labelbook.

4.1. label dir

También podemos conocer las etiquetas almacenadas mediante `label dir`

4.2. mask

Las etiquetas pueden acoplar el número junto conel nombra asociado, mediante el comando:

```
numlabel s, mask(#.)add
```

LO que estamos haciendo es usar la etiqueta ya definida `s` para anteponerle el número.

5. fechas

El trabajo con fechas en Stata es más complejo de lo que parece (y de la manera superficial en que lo abordé en el Manual). Trataré de ordenar las ideas de otra forma.

5.1. Los formatos

Usualmente trabajamos fechas en el formato día, mes, año. Sin embargo, los datos estadísticos pueden requerir algunas formas más elaboradas como semestres o cuatrimestres. También es posible que necesitemos horas, minutos, segundos o milisegundos. Sin consideramos todas estas formas, se requiere que el formato de las fechas no sea único, con lo cual se

arma un gran lío. Además los segundos y milisegundos introducen una segunda complicación, pues el movimiento de la tierra no es tan regular en este orden y hay impredicibilidad en ese nivel precisión. Por eso la lista de formatos es larga. Las fechas en Stata tienen un formato llamado SIF (Stata Internal Form) que abarca las siguientes variantes:

```
SIF type        Examples in SIF       Units
-------------------------------------------------------------
datetime/c      1,479,597,200,000     milliseconds since
                                              01jan1960 00:00:00.000,
                                              assuming 86,400 s/day

datetime/C      1,479,596,223,000     milliseconds since
                                              01jan1960 00:00:00.000,
                                              adjusted for leap seconds

date                       18,282     days since 01jan1960
                                              (01jan1960 = 0)

weekly date                 2,601     weeks since 1960w1
monthly date                  600     months since 1960m1
quarterly date                 58     quarters since 1960q1
half-yearly date              100     half-years since 1960h1
yearly date                  2010     years since 0000
-------------------------------------------------------------
```

5.1.1. Un segundo para los segundos intercalares

Casi todo bien, excepto por la expresión *leaps* que se traduce en español como intercalar, o sea segundo intercalar (sugiero que lea https://es.wikipedia.org/wiki/Segundo_intercalar) que justamente es el ajuste que se realiza introduciendo 1 segundo para acompañar la rotación mas lenta de la tierra, cuando la diferencia entre el segundo atómico y el segundo solar es mayor de 0.7 segundos. Se anuncia con seis meses de anticipación y se realiza la noche del 30 de junio para el 1 de julio o del 31 de diciembre para el 1 de enero. En esa ocasión se produce una marca de tiempo en que el 59 es sucedido por el 60, y no como usualmente sucede que continúa el 00. Desde 1972 se han introducido 26 segundos intercalares, así que no considerar esta diferencia en órdenes de segundo y milisegundos puede ser fuente de error.
Por lo tanto los dos primeros formatos que incluyen la fecha con segundos, distinguen con `c` la forma sin segundos intercalares y `C` con segundos intercalares. El resto de las formas son entendibles.

5.2. Volvamos a la fechas

Si nosotros creamos una fecha en Stata, por ejemplo escribiendo:

```
clear all
```

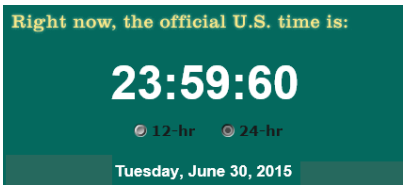


Figura 1: Inserción del segundo intercalar

```
input dia mes año
30 6 2015
end
```

Si listamos esta única observación, tenemos:

```
  list in 1

     +------------------+
     | dia   mes    año |
     |------------------|
  1. |  30     6   2015 |
     +------------------+
```

Estos datos podemos transformarlos en fecha, uniéndolas en una sola variable y usando la función mdy (en azul en stata):

```
gen fecha=mdy(mes, dia, año)
list fecha
```

Lo que nos da:

```
   list fecha

     +-------+
     | fecha |
     |-------|
  1. | 20269 |
     +-------+
```

Es una fecha extraña, pero dice que el 30 de junio del 2015 es el día 20.269 a partir del 1 de enero de 1960.
Para entenderlo mejor, manejemos el formato `%td`:

```
format fecha %td
list fecha
```

lo que nos da:

```
     +-----------+
     |     fecha |
     |-----------|
  1. | 30jun2015 |
     +-----------+
```

podemos explorar unos pocos formatos de fecha:

```
format fecha %tdCCYY.NN:DD
list fecha
     +------------+
     |      fecha |
     |------------|
  1. | 2015.06:30 |
     +------------+
```

Este formato dice que usamos en primer lugar el siglo (las C en mayúsculas significan que Cristo es del siglo 01 y no del 1), un punto, el número del mes (y no el nombre; las mayúsculas significan lo mismo), dos puntos y el número del día (las mayúsculas significan lo mismo).
Otra variante:

```
format fecha %tdMonth_dd_CCYY
list fecha
     +--------------+
     |        fecha |
     |--------------|
  1. | June 30 2015 |
     +--------------+
```

El mes ahora es una palabra entera y el guión bajo señala un espacio que separa mes de día y año.
Otro formato:

```
format fecha %tdnn/dd/YY
list fecha
     +---------+
     |   fecha |
     |---------|
  1. | 6/30/15 |
     +---------+
```

Las letras del número del día son minúsculas, por tanto si se tratara de 1 de junio, sería 1 y no 01.

5.3. Importar

Datos provenientes de SAS no tienen problema porque la convención del 1 de enero de 1960 es la misma. SPSS usa como fecha 0 el 14 de octubre de 1582, por lo tanto hay que sumar es fecha en formato td. Para R que usa como fecha el 1 de enero de 1970 se debe restar esa fecha

en formato td. Excel usa el 1 de enero de 1900 como su fecha inicial.
Para pasar de una fecha en Excel a Stata, podemos hacer:

```
gen double statafecha= fecha_excel+td(30dec1899)
format statafecha %tc
```

Generamos una nueva variable double para no perder los datos de horas, minutos, segundos y mantenemos el formato tc.

5.4. A partir de string

La forma en que trajimos datos en el Manual fue a partir de una string. Podemos ahora hacer lo mismo, usando la función date, señalando la variable string que contiene la fecha y tras una coma especificando la forma en que están organizados los datos ("mask")

```
clear
input str9 fechastr
"30/6/15"
end
gen fecha=date(fechastr, "DM20Y")
list fecha
     +-------+
     | fecha |
     |-------|
  1. | 20269 |
     +-------+
format fecha %tddd/nn/YY
list fecha
     +---------+
     |   fecha |
     |---------|
  1. | 30/6/15 |
     +---------+
```

Como el año no contenía los siglos, usamos `"DM20"` para incorporarlo. En las instrucciones de Stata esa parte se denomina `"mask"`. Otras variantes, incluyendo el día secuencial del año en `JJJ` también llamado (`Juliano`).

```
format fecha %tdDD/Month/CC
list fecha
     +------------+
     |      fecha |
     |------------|
  1. | 30/June/20 |
     +------------+

format fecha %tdDD/Month/JJJ
```

```
list fecha
```

```
     +-------------+
     |       fecha |
     |-------------|
  1. | 30/June/181 |
     +-------------+
```

```
format fecha %tdDD/mon/CC
list fecha
```

```
     +-----------+
     |     fecha |
     |-----------|
  1. | 30/jun/20 |
     +-----------+
```

```
format fecha %tdDAYNAME/Month/JJJ
list fecha
```

```
     +--------------------+
     |              fecha |
     |--------------------|
  1. | Tuesday  /June/181 |
     +--------------------+
```

Un despliegue amplio de formato está en el `help dates`, pero una lista útil puede ser:

```
 Code      Meaning            Output
------------------------------------------------------------------
CC         century-1          01 - 99
cc         century-1          1 - 99
YY         2-digit year       00 - 99
yy         2-digit year       0 - 99

JJJ        day within year    001 - 366
jjj        day within year    1 - 366

Mon        month              Jan, Feb, ..., Dec
Month      month              January, February, ..., December
mon        month              jan, feb, ..., dec
month      month              january, february, ..., december
NN         month              01 - 12
nn         month              1 - 12

DD         day within month   01 - 31
dd         day within month   1 - 31

DAYNAME    day of week        Sunday, Monday, ... (aligned)
Dayname    day of week        Sunday, Monday, ... (unaligned)
Day        day of week        Sun, Mon, ...
Da         day of week        Su, Mo, ...
day        day of week        sun, mon, ...
da         day of week        su, mo, ...

h          half               1 - 2
q          quarter            1 - 4
WW         week               01 - 52
ww         week               1 - 52

HH         hour               00 - 23
Hh         hour               00 - 12
hH         hour               0 - 23
hh         hour               0 - 12
------------------------------------------------------------------
```

5.5. Conversión

Pasar de un modo a otro de fechas requiere generar nuevas variables. Algunas funciones para hacerlo se ilustran con ejemplos con lo que ya tenemos:

```
gen semana=wofd(fecha)
list semana
```

```
     +--------+
     | semana |
     |--------|
  1. |   2885 |
     +--------+
list semana
     +---------+
     |  semana |
     |---------|
  1. | 2015w26 |
     +---------+
gen mensual=mofd(fecha)
list mensual
     +---------+
     | mensual |
     |---------|
  1. |     665 |
     +---------+
gen trimestre=qofd(fecha)
list trimestre
     +----------+
     | trimes~e |
     |----------|
  1. |      221 |
     +----------+
gen semestral=hofd(fecha)
list semestral
     +----------+
     | semest~l |
     |----------|
  1. |      110 |
     +----------+
gen anual=yofd(fecha)
list anual
     +-------+
     | anual |
     |-------|
  1. |  2015 |
     +-------+
```

Estas fechas las podemos formatear a su específico modo, para apreciarlas mejor:

```
format semana %tw
list semana
     +---------+
     |  semana |
```

```
     |---------|
  1. | 2015w26 |
     +---------+
format mensual %tm
list mensual
     +---------+
     | mensual |
     |---------|
  1. |  2015m6 |
     +---------+
format trimestre %tq
list trimestre
     +----------+
     | trimes~e |
     |----------|
  1. |   2015q2 |
     +----------+
format semestral %th
list semestral

     +----------+
     | semest~l |
     |----------|
  1. |   2015h1 |
     +----------+
format anual %ty
list anual
     +-------+
     | anual |
     |-------|
  1. |  2015 |
     +-------+
```

Ahora podemos comprender mejor estos modos, suficientemente explicados como ya dijimos en el `help` bajo el nombre `dates`

```
Display format to
SIF type         present SIF in HRF
-----------------------------------
datetime/c            %tc
datetime/C            %tC

date                  %td

weekly date           %tw
monthly date          %tm
quarterly date        %tq
```

```
half-yearly date      %th
yearly date           %ty
-----------------------------------
```

6. Una nota para los gráficos

Cuando usamos fechas, podemos aprovechar para poner esa fecha como variable de tiempo, mediante el comando `tsset` que significa *time series set*. Al graficar usando este modo, tenemos que cuidar el formato en que aparezca el tiempo etiquetado en el eje x. Eso lo podemos especificar después del `tsline` var anteponiendo la correspondiente coma: `tlabel(, format(%td) angle(90))`, a lo cual agregué un ángulo de 90 grados de las etiquetas.

6.1. Días hábiles

El encargo es muestrear sólo en días hábiles. Para eso usamos un formato de fechas que tiene la siguiente forma: `%tb`*calname*. Esto significa que nosotros hacemos un pequeño programa cuya extensión es `stbcal` cuya forma puede ser:

```
version 12
range 01jan2016 30dec2016
centerdate 01jan2016
omit dayofweek (Sa Su)
omit date 25march2016
omit date 26march2016
omit date 26march2016
omit date 27june2016
omit date 15aug2016
omit date 19sept2016
omit date 10octob2016
omit date 31octob2016
omit date 01november2016
omit date 08december2016
```

Este programa se guarda en `C:\ado\personal`. Le puse como nombre `random`. Aquí yo omití sábados y domingo, pero además quité los feriados del año 2016.Teniendo este programa podemos muestrear al interiro de una base de datos con los días hábiles de ese año. Lo primero que hacemos es generar una base para guardar los días y el mes de cada muestreo. Luego hacemos una iteración para muestrear cada mes. Ponemos las 366 observaciones (año bisiesto) y le aplicamos el formato de los días hábiles: `%tbrandom`. Finalmente sembramos semillas para que el muestreo pueda ser replicable y hacemos muestras de tamaño uno para cada mes. Generamos una salida de texto para guardar los resultados:

```
  clear all
cd "C:\Users\Hp\Documents\UCALIDAD\random"
set obs 12
gen diademuestra=.
gen id1=.
```

```
gen mes=.
save sample,replace

forvalues i=1(1)12{
clear
set obs 366
set seed 123
gen diademuestra=_n
clonevar id1=diademuestra
format diademuestra %tbrandom
keep in 1/252
gen mes=irecode(id1,0,20,41,63,84,106,127,148,170,191,210,231)
keep if mes==‘i’
sample 1 ,count
append using sample
save sample,replace
}

use sample, clear
sort diademuestra

log using mensual,replace text
list diademuestra mes in 1/12,abbreviate(16)
log close
use sample,clear
sample 1 if mes<7,count
sample 1 if mes>6&diademuestra<.,count

log using semestral, replace text
sort diademuestra
list diademuestra mes if diademuestra!=.,abbreviate(16)
log close
```

Más información en `datetime business calendars`

7. Gráficos

7.1. La zona de los gráficos

El área de un gráfico es una zona muy complicada. En este caso vamos a dibujar un gráfico de barras con los pesos medios de los fallecidos menores de un año por servicio (es decir hemos hecho `keep if edad_tipo>1`. El comando que ejecutamos es

```
graph bar peso, over(reg_res, label(angle(90)))
```

y nos da este complicado gráfico: Para poder expandir la zona en que aparecen los nombres de los servicios, tenemos que conocer algo de cómo se distribuyen las zonas en los gráficos de

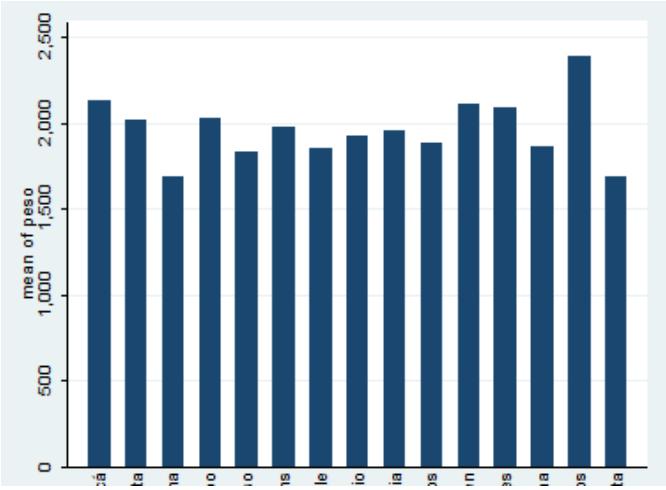


Figura 2: gráfico de barras con peso medio por servicio

Stata. La zona en cuestión es `b1title` que es donde se escriben las etiquetas de las variables graficadas.

```
The definition of ringposstyle and the default positioning of titles is

      l2  l1                          r1  r2     |
       1  2  0 0 0 0 0 0 0 0 0 0 0 0   1  2     |  ringposstyle
     ------------------------------------------+--------------------
     +---------------------------------------+    |
     |               title                   |    |    7     title
     |              subtitle                 |    |    6     subtitle
     |              t2title                  |    |    2     t2
     |              t1title                  |    |    1     t1
     |       +----------------------+        |    |
     |  l  l |                      | r  r   |    |    0
     |  2  1 |                      | 1  2   |    |    0
     |  t  t |                      | t  t   |    |    0
     |  i  i |      plot region     | i  i   |    |    0
     |  t  t |                      | t  t   |    |    0
     |  l  l |                      | l  l   |    |    0
     |  e  e |                      | e  e   |    |    0
     |       +----------------------+        |    |
     |              b1title                  |    |    1     b1
     |              b2title                  |    |    2     b2
     |              legend                   |    |    3     legend
     |       note                            |    |    4     note
     |       caption                         |    |    5     caption
     +---------------------------------------+
        where titles are located is
        controlled by the scheme
```

Para expandir esa zona, graficamos mediante:

```
graph bar peso, over(reg_res, label(angle(90)))/*
*/b1title("  ",margin(b=40))
```

Lo que hacemos es meternos en la zona del título que Stata denomina b1 y aumentar su margen con el comando `margin`. La letra b significa bottom, es decir la parte inferior. Las unidades son arbitrarias. He procedido mediante prueba y error. He puesto comillas sin agregar nada, pues no interesa el titulo de este eje. El gráfico que obtenemos es: Ahora sólo queda hacer un pequeño arreglo para ordenar de mayor a menor las barras.

```
graph bar peso, over(reg_res, sort(1) descending label(angle(90)))
b1title("  ",margin(b=40))
```

Con el uso del comando `sort` que indica el orden y además eligiendo la primera variable `(1)` para ordenar. En este caso sólo tenemos una `descending`, tenemos ya nuestro gráfico final. El sentido opuesto se logra con `reversing`:

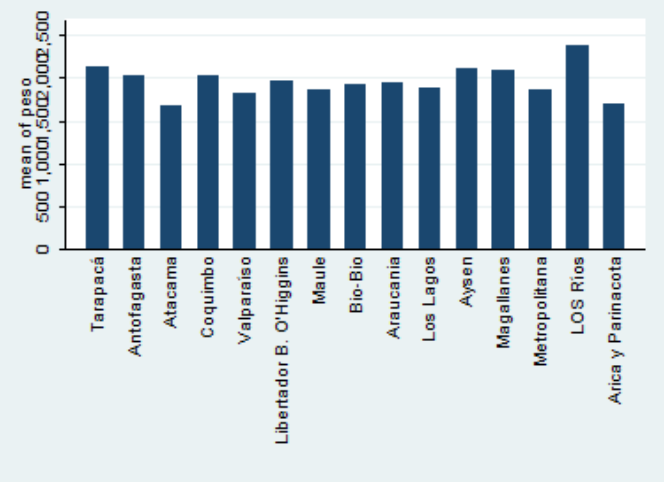


Figura 3: gráfico de barras con peso medio por servicio y legibilidad de las etiquetas

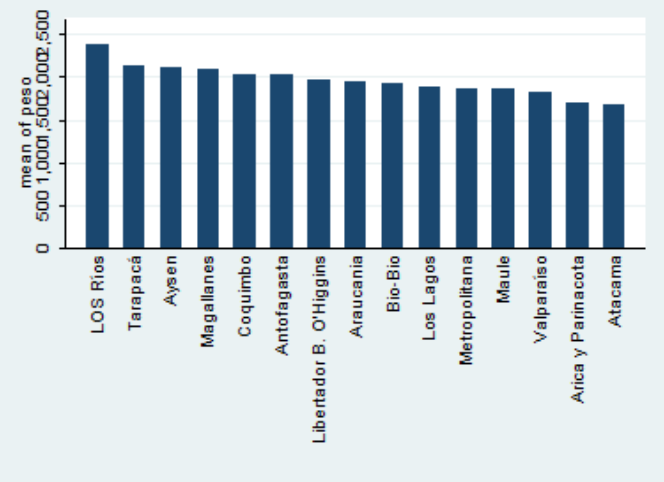


Figura 4: gráfico de barras con peso medio por servicio en orden descendente

7.2. Varias variables en un mismo histograma

Un desafío interesante puede ser graficar algunas variables en un histograma. El comando `histogram` permite la option `by`. Como en el caso de las edades cantidad por tipos de dades en

```
histogram edad_cant, by(edad_tipo)
```

que nos da el gráfico siguiente: Pero en este caso no nos referimos a eso, sino a graficar la frecuencia de defunciones por servicio y regiones. Lo que hacemos es transformar las varia-

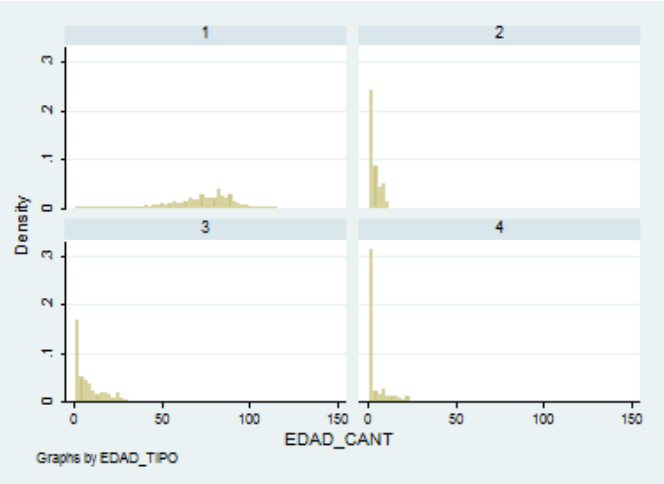


Figura 5: Histograma con opción by

bles en valores de preguntas, renombrando las dos variables y luego transformándolas a un formato `wide` mediante `reshape`.

```
rename reg_res AG1
rename serv_res AG2
gen id=_n
reshape long AG, i(id) j(pregunta)
label define v 1 Region 2  Servicio
label values pregunta v
tab AG if pregunta==1
replace AG=. if AG==99
tab AG if pregunta==2
histogram AG, by(pregunta,col(2))
```

El comando `histogram` permite graficar las frecuencias en forma porcentual (`percent`), como fracción (`frac`) o como frecuencias (`freq`). El *default* es densidad.

7.3. rcap

Para dibujar las rayitas clásicas de la desviación estándar, usamos el comando `rcap` que traza dos rayas horizontales y las une por una vetical. Entonces podemos generar los valores que queremos que sean el valor superior y, luego indicamos el valor inferior y, para finalmente decir el valor x. En este caso usamos los pesos de los fallecidos por servicio de salud (menores de un año). Superponemos un gráficos de puntos que marca el percentil 50. Hemos usado `egen` que es muy versátil, para calcular máximos por columnas. Pero hay `rowmin`, `rowmax` y `rowpctile`, entre otros útiles.

```
use def2013,clear
```

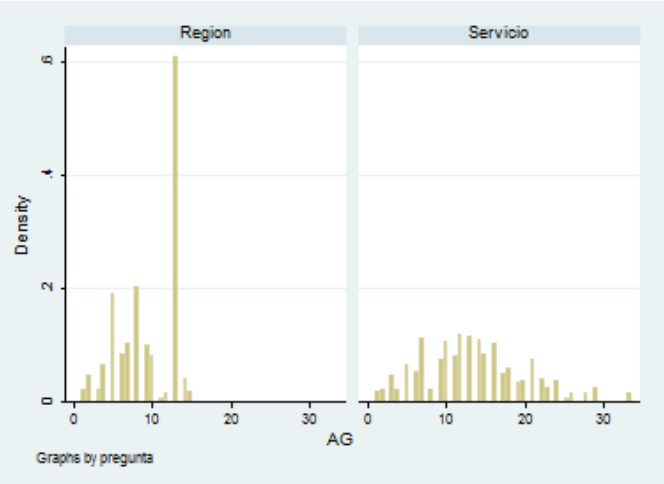


Figura 6: Histograma de dos variables con opción by

```
keep if edad_tipo>1
destring peso,replace
egen min=min(peso), by(serv_res)
drop if peso==9999
egen max=max(peso), by(serv_res)
egen mediana=pctile(peso),p(50) by(serv_res)

tw (rcap min max serv_res)(scatter mediana serv_res)
```

7.4. Graph bar

En cuanto a gráficos de barras, sólo dos cosas. El comando `asyvars` permite graficar variables especificadas en los over, especificando que sus niveles son valores del eje y. El comando `aslistofvar` permite graficar una lista de variables como barras, como si estuvieramos usando el by.

```
graph bar peso, over(reg_res, sort(1) descending label(angle(90))) over(sexo) asyvars
```

Esto resulta en el siguiente gráfico:

8. loops y matrices

8.1. scalar

Un elemento muy crucial en los loops son las locales. Una forma alternativa de almacenar algo son los escalares . Aunque sólo almacenna una sola cosa y no algo que varía como los contadores, los escalares son la forma más apropiada de guardar una cantidad. Si generamos un

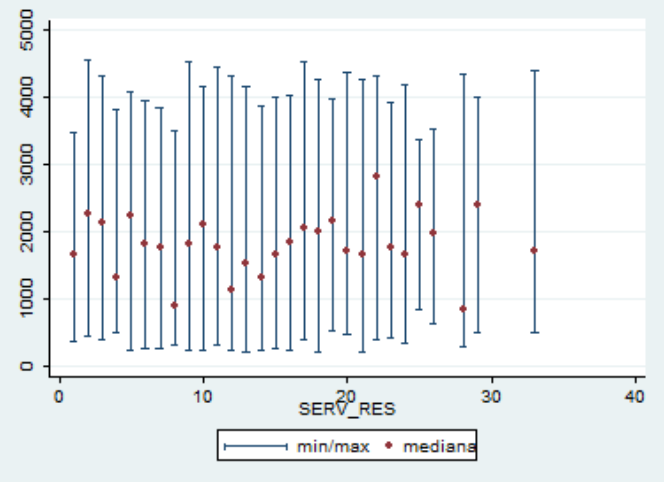


Figura 7: Gráfico de mínimos y máximos, con rcap

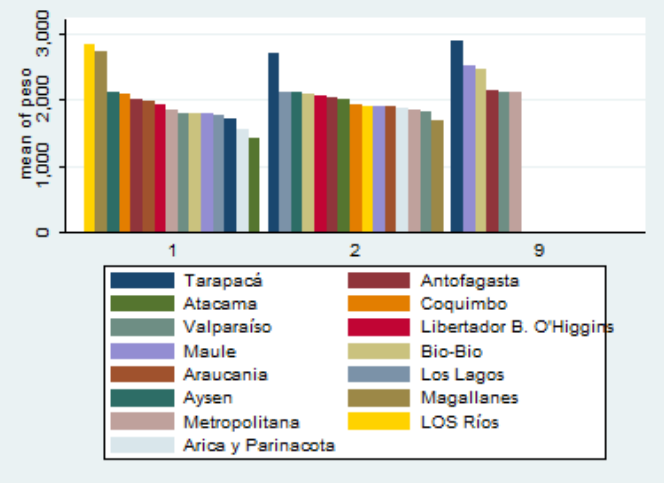


Figura 8: gráfico de barras con peso medio por servicio y legibilidad de las etiquetas

`scalar` que guarde la proporción de mujeres en nuestra base (previamente transformamos las mujeres en 1 y los hombres en cero)

```
replace sexo=sexo-1
sum sexo,meanonly

. scalar p=r(mean)

. scalar list
```

```
         p =   1.470873

. display scalar(p)
1.470873
```

podemos tener un listado de escalares con `scalar list`. Notar que cuando llamamos un escalar ponemos su nombre entre paréntesis `display scalar(p)`.

8.2. include

El comando `include` permite llamar un archivo `do` guardado. Por ejemplo, podríamos escribir todas nuestras órdenes de encabezado, considerando el cambio de directorio en un archivo do, que se llame encabezado. Basta entonces escribir al inicio de cualquier do, `|includeencabezado—`, para que se ejecuten todas las órdenes.

8.3. matrices

Si queremos una tabla con datos organizados por regiones, podemos iterar (*loops*) por el número de cada región. Pero guardar esos datos no siempre es fácil. Una solución es usar matrices .
Lo primero es definir una matriz que tiene 15 filas y seis columnas. La llenamos de 99 para verificar que esos valores son reemplazados durante los cálculos. Ponemos títulos a las columnas y a las filas. Luego generamos una local que usamos como contador (el nombre de la local es arbitrario) usando el signo `++` , para ir contando desde la fila primera hasta la 15 y llenando con los elementos de la matriz, columna a columna, con los valores acumulados en `r( )` tras el `sum`.

```
matrix defunciones=J(15,6,99)
matrix colnames defunciones=  N p1 p5 p25 p50 p75
matrix rownames defunciones= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
matrix list defunciones

local filas=0
forvalues i=1(1)15{
local ++filas
quietly sum peso if reg_res==‘i’,d
matrix defunciones[‘filas’,1]=r(N)
matrix defunciones[‘filas’,2]=r(p1)
matrix defunciones[‘filas’,3]=r(p5)
matrix defunciones[‘filas’,4]=r(p25)
matrix defunciones[‘filas’,5]=r(p50)
matrix defunciones[‘filas’,6]=r(p75)
}
matrix list defunciones
```

El uso de estos comandos debe darnos:

```
  matrix defunciones=J(15,6,99)

. matrix colnames defunciones=  N p1 p5 p25 p50 p75

. matrix rownames defunciones= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

. matrix list defunciones

defunciones[15,6]
      N   p1   p5  p25  p50  p75
 1   99   99   99   99   99   99
 2   99   99   99   99   99   99
 3   99   99   99   99   99   99
 4   99   99   99   99   99   99
 5   99   99   99   99   99   99
 6   99   99   99   99   99   99
 7   99   99   99   99   99   99
 8   99   99   99   99   99   99
 9   99   99   99   99   99   99
10   99   99   99   99   99   99
11   99   99   99   99   99   99
12   99   99   99   99   99   99
13   99   99   99   99   99   99
14   99   99   99   99   99   99
15   99   99   99   99   99   99

.
. local filas=0

. forvalues i=1(1)15{
  2. local ++filas
  3. quietly sum peso if reg_res==‘i’,d
  4. matrix defunciones[‘filas’,1]=r(N)
  5. matrix defunciones[‘filas’,2]=r(p1)
  6. matrix defunciones[‘filas’,3]=r(p5)
  7. matrix defunciones[‘filas’,4]=r(p25)
  8. matrix defunciones[‘filas’,5]=r(p50)
  9. matrix defunciones[‘filas’,6]=r(p75)
 10. }

. matrix list defunciones

defunciones[15,6]
         N      p1      p5     p25     p50     p75
 1      34     450     459     863    2250    2909
 2      68     390     590     900  2132.5    2920
```

```
 3      39      498      518      800     1300     2400
 4      91      240      400      820     2220     2980
 5     156      260      380      780   1677.5     2860
 6      73      260      450      800     1645     2900
 7      83      223      390      700     1830     2990
 8     206      270      430      660   1997.5     2965
 9      87      200      340      754     1720     2985
10      90      330      500      850   1706.5     2790
11       8      824      824   981.5     2395     2966
12      13      624      624     1230     1960     3115
13     659      235      380      630     1590     2760
14      43      400      450     1300     2800     3150
15      32      350      400   874.5   1642.5   2512.5

.
end of do-file
```

Estos datos no quedan guardados. Si los queremos transformar en variables, usamos `svmat defunciones` que transforma las columnas de la matriz en variables. De modo inverso para transformar variables en matriz, usamos `mkmat`.

Índice de comandos y conceptos

aslistofvar, 19
asyvars, 19

clonevar, 2
codebook, problems, 3
contador, 21
contract, 3
conversión de fechas, 10

días hábiles, 13
datetime business calendars, 14
descending, 16

etiquetas de fecha en gráficos tsset, 13

fdías Julianos, 8
fechas, 5
fechas a partir de string, 8
formato mdy, 6
formato de fechas, 4
Formato SIF, 5
frecuencias máximas, 3

histograma, 18
histogramas con varias variables, 17

Importar SPSS; SAS; R, Excel, 8
include, 21
iterar, 21

label dir, 4
labelbook, 4
leap seconds, 5
loops, 21

margin, 16
mask en fechas, 8
matrices, 21
mkmat, 23

pweight, 3

rcap, 18
reversing, 16

scalar, 19
segundo intercalar, 5
sort, 16
stbcal, 13
svmat, 23