Analysing Social Science Data Using R

Class 7: Conditional descriptives and aggregation

Michał Bojanowski[1] mbojan@icm.edu.pl
Zbigniew Karpiński[2] zkarpinski@ifispan.waw.pl

[1]ICM, University of Warsaw

[2]IFiS PAN

April 16, 2012
SNS, Warsaw

Conditional descriptives

Using tapply function

Function tapply defined

With tapply we can summarize a variable in subgroups defined by other variables.
Basic syntax is:
`tapply(X, INDEX, FUN)`
where

- X is a numeric vector.
- INDEX is a vector or list of vectors defining groups.
- FUN is a function applied to X in all groups defined by INDEX vectors.

If FUN returns a single value, then the value returned by tapply is a vector, matrix or array (depending on the number of grouping vectors).

Small example of using tapply

Means of y in subgroups defined by x:

```
> y <- c(1,2,3,2,1,2,3,3,1,5)
> x <- c(1,2,3,1,2,3,1,2,3,1)
> tapply(y, x, mean)
   1    2    3
2.75 2.00 2.00
```

As there is only one grouping vector (x), the result of tapply is a vector of group means.

Example with two grouping vectors

Means of y in subgroups defined jointly by x and z:

```
> y <- c(1,2,3,2,1,2,3,3,1,5)
> x <- c(1,2,3,1,2,3,1,2,3,1)
> z <- c(1,1,1,1,1,2,2,2,2,2)
> tapply(y, list(z=z,x=x), mean)
   x
z     1   2   3
  1 1.5 1.5 3.0
  2 4.0 3.0 1.5
```

Result is a matrix with conditional means of y.

Example with three grouping vectors

Using a data frame d with some artificial data

```
> str(d)
'data.frame':        50 obs. of  4 variables:
 $ y : num  -0.626 0.184 -0.836 1.595 0.33 ...
 $ x1: int  2 1 1 2 2 1 1 1 2 2 ...
 $ x2: chr  "b" "b" "a" "a" ...
 $ x3: chr  "i" "i" "j" "i" ...
```

Example with three vectors contd.

```
> with(d, tapply(y, list(x1=x1, x2=x2, x3=x3), mean))
, , x3 = i

   x2
x1           a           b
  1 0.1685678 -0.08396477
  2 0.6869775  0.19462300

, , x3 = j

   x2
x1           a           b
  1 -0.4376483 0.03255051
  2  0.1902787 0.15832818
```

Using `aggregate` function

Function aggregate defined

With `aggregate` we can summarize more than one variable at a time. The syntax is:

```
aggregate(x, by, FUN)
```

where

- `x` is a data frame
- `by` is a list of grouping vectors
- `FUN` is a function to be computed on variables in `x`

The result is a data frame of summaries.

Example of aggregate using PGSS data

Data frame pgss contains data loaded from pgss1999in.tab file.

```
> res <- with(pgss,
+             aggregate( pgss[,c("in5a", "in5b", "in5c")],
+                        list(hompop=hompop, year=pgssyear),
+                        mean, na.rm=TRUE)    )
> str(res)
'data.frame':        12 obs. of  5 variables:
 $ hompop: int  1 2 3 4 5 6 7 8 9 10 ...
 $ year  : int  1999 1999 1999 1999 1999 1999 1999 1999 1999 199
 $ in5a  : num  1143 1063 1238 1054 1036 ...
 $ in5b  : num  1855 1980 2243 2011 2026 ...
 $ in5c  : num  16603 21048 18256 11692 19257 ...
```

Example of aggregate contd.

```
> res
   hompop year      in5a     in5b      in5c
1       1 1999 1142.9310 1854.915 16602.632
2       2 1999 1063.4518 1980.211 21047.644
3       3 1999 1237.6562 2242.826 18255.914
4       4 1999 1054.1667 2010.857 11691.525
5       5 1999 1035.5372 2025.776 19256.637
6       6 1999  862.5974 1658.108  9975.676
7       7 1999  992.5926 1939.286 20800.000
8       8 1999 1050.0000 1742.857 11625.000
9       9 1999  988.8889 1966.667 10375.000
10     10 1999  700.0000 1300.000  3750.000
11     11 1999 1000.0000 2500.000  4000.000
12     12 1999       NaN      NaN       NaN
```

Exporting data

Exporting to text files

Function `write.table`

Function `write.table` is a counterpart to `read.table` and can be used to write data frames to text files. Syntax:

```
write.table(x, file="", sep=" ", row.names=TRUE, col.names=TRUE)
```

where

- `x` is a data frame
- `file` is file name
- `sep` is a character used to separate columns
- `col.names/row.names` are logicals whether row and column names should be written to the file.

There are other arguments too, see help.

Example of using write.table

```
> write.table( res, file="dane.txt", sep=";", quote=TRUE,
+              row.names=FALSE, col.names=TRUE)
```

The file looks like this (first 8 rows)

```
"hompop";"year";"in5a";"in5b";"in5c"
1;1999;1142.93103448276;1854.91525423729;16602.6315789474
2;1999;1063.45177664975;1980.21052631579;21047.6439790576
3;1999;1237.65625;2242.82608695652;18255.9139784946
4;1999;1054.16666666667;2010.85714285714;11691.5254237288
5;1999;1035.53719008264;2025.77586206897;19256.6371681416
6;1999;862.597402597403;1658.10810810811;9975.67567567567
7;1999;992.592592592593;1939.28571428571;20800
8;1999;1050;1742.85714285714;11625
```

Exporting to SPSS files

Getting data to SPSS

Possible ways:

- Use plain text files as an intermediate format. Export from **R** to text, then read text file in SPSS.
- Use `write.foreign` from package `foreign`

Function `write.foreign`

Function `write.foreign` does not create a `sav` file directly. Instead, it produces two files:

1. A text file with data that SPSS can easily read.
2. An SPSS syntax file with commands that will read the data from (1), define variable types, missing values etc.

The syntax is:

```
write.foreign( df, datafile, codefile, package)
```

where

`df` is a data frame

`datafile, codefile` are file names for data and code respectively

`package` character, name of the output format. Possible values `"SPSS"`, `"Stata"` and `"SAS"`.

Example of using `write.foreign`

```
> library(foreign)
> write.foreign(res, "data.dat", "data.sps", package="SPSS")
```

Data file:

```
1,1999,1142.93103448276,1854.91525423729,16602.6315789474,
2,1999,1063.45177664975,1980.21052631579,21047.6439790576,
3,1999,1237.65625,2242.82608695652,18255.9139784946,
4,1999,1054.16666666667,2010.85714285714,11691.5254237288,
5,1999,1035.53719008264,2025.77586206897,19256.6371681416,
6,1999,862.597402597403,1658.10810810811,9975.67567567567,
7,1999,992.592592592593,1939.28571428571,20800,
8,1999,1050,1742.85714285714,11625,
9,1999,988.888888888889,1966.66666666667,10375,
```

Using write.foreign contd.

SPSS Syntax file:

```
DATA LIST FILE= "data.dat"  free (",")
/ hompop year in5a in5b in5c  .

VARIABLE LABELS
hompop "hompop"
 year "year"
 in5a "in5a"
 in5b "in5b"
 in5c "in5c"
```