

1 Controlling print width and character formatting

Put these lines at the beginning of every SAS program if you want output to print correctly on 8-1/2 by 11 inch paper and to have lines print correctly in tables and graphs:

```
options linesize = 75
        formchar = "|----|+|----+|-/\<>*";
```

The character string for formchar is included on the course web page under “Datasets” in the file called “formchar.” You may copy it from there into your program.

The `linesize` option tells SAS how many characters to print on each line of text. For normal size text, a maximum of 80 characters can be printed per line. The `formchar` option tells SAS what characters to use to print the lines dividing cells in certain kinds of tables. If we let SAS use its default setting for formchar, these tables will not print correctly.

2 Dataset to download

Please download the dataset `gulanick.dat` from the “Datasets” section of the course web page after reading its associated `.info` files.

Gulanick (Heart and Lung, 1991) studied patients who were recovering from heart surgery. She was interested in whether different combinations of supervised exercise or teaching would affect patients’ self-efficacy (or confidence) to perform physical activity.

Patients were randomly assigned to one of three groups. Group 1 received teaching, treadmill exercise testing, and exercise training three times per week. Group 2 received only teaching and exercise testing. Group 3 received only routine care without supervised exercise or teaching. After 4 weeks, each patient was scored on self-efficacy.

Self-efficacy was measured on a continuous scale and scores were assumed to be distributed normally in each of the populations of interest.

The variables in the dataset are:

- score
- group (coded 1, 2, 3)

Data is taken from Daniel, WW (1999) *Biostatistics: A Foundation for Analysis in the Health Sciences*. Wiley.

- ## 3
- Using formats to get SAS to print something other than the values a variable actually contains
 - Using labels to get SAS to print more descriptive variable names

The values of the group variable in the dataset are the numbers 1, 2, and 3. If we want SAS to print out descriptive words instead of the numeric codes, so that tables and graphs are more understandable, we need to run a “proc format” *before* the data step. The data step must then refer to the formats defined in the format procedure.

```
proc format ;
value grpfmt 1 = 'Teaching and Training' 2 = 'Teaching' 3 = 'Neither' ;
run ;
```

Note the format statement in the data step below. It tells SAS to apply the format you have defined here to a particular variable. When you use the format statement in a data step, you must put a period at the end of the format name.

The `label` statement in a data step causes most of the subsequent procedures to display the variable labels instead of the variable names.

```
data gulan ;
*infile '/group/ftp/pub/kcowles/datasets/gulanick.dat' ;
infile 'c:\temp\gulanick.dat' ;
input score group ;
format group grpfmt. ;
label group = 'Treatment Group' score = 'Self-Efficacy Score' ;
run ;
```

Now enter and run the following code to see how the formats and labels affect the output of the “print” and “freq” procedures.

```
proc print data = gulan (obs = 20);
run ;

proc print label data = gulan (obs = 20);
run ;

proc freq data = gulan ;
tables group ;
run ;
```

4 Using `proc tabulate` to summarize the distributions of quantitative variables in different groups

```
proc tabulate data = gulan ; class group ; * class statement identifies qualitative variables ;
var score ; * var statement identifies quantitative variables ; tables group , score * (n mean
std) ; run ;
```

5 Formats for numeric variables

Formats can also be used to group numeric data. Suppose we want to identify the values of the `score` variable as either below the median score, or equal to or above the median.

First we need to find out the median score.

```
proc means data = gulan median ;
var score ;
run ;
```

Now add a line to your format procedure and change one line in the data step as follows:

```
proc format ;
value grpfmt 1 = 'Teaching and Training' 2 = 'Teaching' 3 = 'Neither' ;
value scorefmt low-<117 = 'Below median'
117high = 'At or above median' ;
run ;
```

```
data gulan ;
*infile '/group/ftp/pub/kcowles/datasets/gulanick.dat' ;
infile 'c:\temp\gulanick.dat' ;
input score group ;
format group grpfmt. score scorefmt. ;
label group = 'Treatment Group' score = 'Self-Efficacy Score' ;
run ;
```

Now we can treat the `score` variable like another categorical variable (in this case, binary). For example, we can use `proc freq` to test the null hypothesis that the population proportion of patients who score below the median in self-efficacy is the same in the three populations defined by the three types of treatment.

$$H_0 : p_1 = p_2 = p_3$$

```
proc freq data = gulan ;
tables group * score / chisq ;
run ;
```

The FREQ Procedure
Table of group by score

group(Treatment Group)	score(Self-Efficacy Score)		
Frequency			
Percent			
Row Pct			
Col Pct	Below me Median a	Total	
	dian	nd above	
Teaching and Tra	5	6	11
ining	13.89	16.67	30.56
	45.45	54.55	
	29.41	31.58	
Teaching	3	9	12
	8.33	25.00	33.33
	25.00	75.00	
	17.65	47.37	
Neither	9	4	13
	25.00	11.11	36.11
	69.23	30.77	
	52.94	21.05	
Total	17	19	36
	47.22	52.78	100.00

Statistics for Table of group by score

Statistic	DF	Value	Prob
Chi-Square	2	4.9181	0.0855
Likelihood Ratio Chi-Square	2	5.0929	0.0784
Mantel-Haenszel Chi-Square	1	1.5246	0.2169
Phi Coefficient		0.3696	
Contingency Coefficient		0.3467	
Cramer's V		0.3696	

The Chi-Square test gives a test statistic (4.9181) and a p-value (0.0855) for the the null hypothesis stated above.

6 Dataset to download

Please download the datasets `175rna_fup.dat` and `175status.dat` from the course web page after reading the `175.info` files.

I purchased the data from the virology substudy of an AIDS clinical trial (ACTG 175) from the National Technical Information Service (NTIS). The data were provided as 9 separate files on a floppy disk. I was interested in relating the longitudinal trajectories of patients' RNA concentrations to their clinical disease status.

Today we will use the data file that contains only the follow-up values of RNA. (The baseline values are contained in the `baseline` data file. To evaluate HIV-1 concentration, blood was drawn from the patient at each follow-up visit. The specimen was then kept frozen. At certain times during the study, specimens were thawed and assayed. We wish to determine the average number of days between blood collection and assay performance.

7 Reading the dataset and doing arithmetic with dates

Internally, SAS stores dates as numeric variables. This makes it easy to calculate the elapsed time between two dates. The value for any date is the number of days from 01/01/1960 until that date. Date *informats* and *formats* enable SAS to correctly read dates that are stored in different forms in data files, and to print dates in ways that are meaningful to people.

The documentation that came with the disk states: "The variables in `RNA_fup.dat` are in the following locations:

```
@1 pidnum
@7 specdate date7.
@15 assaydt date7.
@23 rna;
"
```

As noted in recent lecture, we will also need to let SAS know to stop reading the `pidnum` variable at the 6th character position, even if there is no space before `specdate`; see code below. In addition, we will compute the number of days that each specimen was frozen.

```
options linesize = 75
formchar = "|----|+|---+|=|-\<>*";

data rnafup ;
*infile '/group/ftp/pub/kcowles/datasets/175rna_fup.dat' ;
infile 'c:\temp\175rna_fup.dat' ;
input
@1 pidnum 6.
@7 specdate date7.
@15 assaydt date7.
@23 rna;
timefrz = assaydt - specdate ;
format specdate assaydt mmddy8. ;
run ;
```

```
proc print data = rnafup (obs=25) ;
run ;
```

```

                                from complete dataset
                                14:02 Friday, June 13, 2003
Obs    pidnum    specdate    assaydt    rna    timefrz
  1    10341    04/07/92    05/16/95    3522    1134
  2    10341    07/02/92    05/16/95    1073    1048
  3    10341    03/17/93    05/16/95    1912     790
  4    10341    08/25/93    05/16/95    1623     629
  5    10341    02/08/94    05/16/95    1399     462
  6    10343    03/11/92    05/16/95     190    1161
  7    10343    05/26/92    05/16/95     308    1085
  8    10343    02/17/93    05/16/95    2679     818
  9    10343    08/26/93    05/16/95    1191     628
 10    10343    01/27/94    05/16/95    2334     474
 11    10361    04/08/92    05/30/95    17490    1147
 12    10361    06/17/92    05/30/95    21299    1077
 13    10364    03/16/92    06/01/95    69826    1172
 14    10364    06/09/92    06/01/95    93252    1087
 15    10378    03/10/92    05/16/95    1281    1162
 16    10378    06/02/92    05/16/95     693    1078
 17    10378    01/25/93    05/16/95    1594     841
 18    10378    07/19/93    05/16/95    2135     666
```

8 Selecting a single observation for each subject

Datasets like this one, with multiple records for each subject, are common. For example, your bank probably has a file with records on every transaction for every account number. Certain kinds of reports or analyses require extracting either the *first* record or the *most recent* record for each subject.

In SAS we can do this in two steps. First we must sort the dataset in such a way that all the records for each subject are grouped together, and that within subject, the records are in date order.

```
proc sort data = rnafup ;
by pidnum specdate ;
run ;
```

Next we need to create a new dataset that contains only the most recent (last in date order) record for each patient.

```

data recent ;      * name the new dataset ;
set rnafup ;      * find the data for the new dataset in the existing dataset called patients ;
by pidnum ;      * recognize that the incoming data is sorted by pidnum ;
if last.pidnum ; * keep only the last record from each pidnum group ;
run ;

```

```

proc print data = recent (obs = 25) ;
run ;

```

```

10341      5      1905.80
10343      5      1340.40
10361      2      19394.50
10364      2      81539.00
10378      5      1453.60
10386      4      20716.00
10389      3      89694.33

```

```

from recent dataset      2
14:02 Friday, June 13, 2003

```

Obs	pidnum	specdate	assaydt	rna	timefrz
1	10341	02/08/94	05/16/95	1399	462
2	10343	01/27/94	05/16/95	2334	474
3	10361	06/17/92	05/30/95	21299	1077
4	10364	06/09/92	06/01/95	93252	1087
5	10378	01/19/94	05/16/95	1565	482
6	10386	08/05/93	06/05/95	74726	669
7	10389	01/11/93	06/05/95	113653	875
8	10432	12/20/93	05/19/95	24354	515
9	10456	12/28/93	05/19/95	735	507

Now we will get SAS to create the corresponding dataset, that has pidnum as the unit of observation instead of visit.

```

proc means data = rnafup noprint nway ; * noprint: don't print results of proc ;
* nway: give results only for individual
pidnums; don't give grand mean ;

class pidnum ;
var rna ;
output out = rnasum      * out = < name of new dataset > ;
mean = m_rna ;          * which summary stats to include,
                        and what to name variables containing
                        them ;

run ;

```

9 Getting summary statistics on each patient

Suppose we want to create a dataset that has a single record for each patient, but we want that record to contain the number of RNA measurements and the average RNA value from all his/her records. We can use proc means to create such a dataset.

First, let's just have proc means display the appropriate output.

```

proc means data = rnafup mean maxdec = 2 ;
class pidnum ;
var rna ;
run ;

```

```

The MEANS Procedure

Analysis Variable : rna

      N
      pidnum  Obs      Mean
-----

```

```

proc print data = rnasum ;
title 'rnasum dataset' ;
run ;

```

```

rnasum dataset      18
13:55 Sunday, June 15, 2003

Obs  pidnum  _TYPE_  _FREQ_  m_rna
1    10341    1        5      1905.80
2    10343    1        5      1340.40
3    10361    1        2      19394.50
4    10364    1        2      81539.00
5    10378    1        5      1453.60
6    10386    1        4      20716.00
7    10389    1        3      89694.33
8    10432    1        5      27132.40

```

9	10456	1	5	214.20
10	10462	1	5	0.00
11	10476	1	2	370633.00
12	10478	1	5	3123.60
13	10889	1	2	2549.50
14	10894	1	5	2950.80
15	10896	1	2	1752.50

The new `_FREQ_` variable gives the number of observations in the original dataset for each value of the class variable.

We can have `proc means` create output datasets with more than one summary statistic and/or more than one variable summarized. For example, suppose we wanted the sample means and sample standard deviations of both `rna` and `timefrz`.

```
proc means data = rnaful noprint nway ;
class pidnum ;
var rna timefrz ;
output out = rnasum
      mean = m_rna m_frz stdev = sd_rna sd_frz ;
run ;

proc print data = rnasum (obs = 15);
run ;
```

rnasum dataset 19
14:04 Sunday, June 15, 2003

Obs	pidnum	_TYPE_	_FREQ_	m_rna	m_frz	sd_rna	sd_frz
1	10341	1	5	1905.80	812.6	954.27	281.003
2	10343	1	5	1340.40	833.2	1139.15	292.497
3	10361	1	2	19394.50	1112.0	2693.37	49.497
4	10364	1	2	81539.00	1129.5	16564.68	60.104
5	10378	1	5	1453.60	845.8	525.53	282.224
6	10386	1	4	20716.00	942.0	36018.97	231.318
7	10389	1	3	89694.33	1071.0	21191.53	174.860
8	10432	1	5	27132.40	857.2	16464.42	276.460
9	10456	1	5	214.20	871.8	325.47	285.361
10	10462	1	5	0.00	855.2	0.00	300.345
11	10476	1	2	370633.00	1157.0	141097.50	59.397
12	10478	1	5	3123.60	811.6	1521.21	280.420
13	10889	1	2	2549.50	931.0	2284.66	435.578
14	10894	1	5	2950.80	898.0	2452.18	288.628
15	10896	1	2	1752.50	1042.0	2478.41	237.588