RESHAPING DATA WIDE TO LONG USING A DATA STEP | SAS LEARNING MODULES

There are several ways to reshape data. You can reshape the data using proc transpose (/sas/modules/how-to-reshape-data-wide-to-long-using-proc-transpose/) or reshape the data in a data step. The following will illustrate how to reshape data from wide to long using the data step.

Example 1: A simple example

We will begin with a small data set with only one variable to be reshaped.

```
“
DATA wide;
  input famid faminc96 faminc97 faminc98 ;
CARDS;
1 40000 40500 41000
2 45000 45400 45800
3 75000 76000 77000
;
RUN;
```

The technique we will use to reshape this data set works well if you have only a few variables to be reshaped. We will create a new variable called year, which will be set equal to each year for which we have data. After setting the variable year equal to a year in our data set, we will set the value of another new variable, faminc, equal to the value of the faminc variable (faminc96, faminc97 or faminc98) for that year. Next, we will use the output statement to have SAS output the results to the data set. Note that if you do not include an output statement after creating the variables for that year, that year will not be included in the new data set. Finally, we will use the drop statement to drop faminc96, faminc97 and faminc98 from our data set once we have finished reshaping it.

```
“
DATA long1 ;
  SET wide ;

  year = 96 ;
  faminc = faminc96 ;
  OUTPUT ;

  year = 97 ;
  faminc = faminc97 ;
  OUTPUT ;

  year = 98 ;
  faminc = faminc98 ;
  OUTPUT ;

  DROP faminc96-faminc98 ;
RUN;
```

Let’s look at the data to ensure that the reshaping worked as we expected. We will run a proc print on the

long1 data file to visually inspect it, and then we will run a proc means on both the original data file, wide, and the new data file, long1, to compare the descriptive statistics.

```
“
PROC PRINT DATA=long1;
RUN;
```

```
Obs    famid    year    faminc

 1       1       96      40000
 2       1       97      40500
 3       1       98      41000
 4       2       96      45000
 5       2       97      45400
 6       2       98      45800
 7       3       96      75000
 8       3       97      76000
 9       3       98      77000
```

The above output looks like we expect: we have nine observations, the famid is the same for each of the three years for each family, and the year variable ranges from 96 to 99. Now let’s run a proc means on both the old and the new data sets.

```
“
PROC MEANS DATA=wide fw=8 ;
  VAR faminc96-faminc98 ;
RUN;
```

```
The MEANS Procedure

Variable    N        Mean     Std Dev     Minimum     Maximum
-------------------------------------------------------------
faminc96    3     53333.3     18929.7     40000.0     75000.0
faminc97    3     53966.7     19238.1     40500.0     76000.0
faminc98    3     54600.0     19546.9     41000.0     77000.0
-------------------------------------------------------------
```

```
PROC MEANS DATA=long1 fw=8 ;
  CLASS year;
  VAR faminc;
RUN;
```

```
The MEANS Procedure
                    Analysis Variable : faminc

              N
    year     Obs    N        Mean     Std Dev     Minimum     Maximum
-----------------------------------------------------------------------
      96       3    3     53333.3     18929.7     40000.0     75000.0

      97       3    3     53966.7     19238.1     40500.0     76000.0

      98       3    3     54600.0     19546.9     41000.0     77000.0
-----------------------------------------------------------------------
```

To ensure that the reshaping was successful, we need to compare the output of the proc means for both the old and the new data sets. All of the descriptive statistics for faminc96 in the first output should be the same as those for year 96 in the second output. For example, we see that there are three observations for faminc96, the mean is 53333.3, the standard deviation is 18929.7, the minimum is 40000.0 and the maximum is 75000.0. These are the exact values that we see in second output for year 96. Likewise, we compare the row in the first output for faminc97 with the corresponding row in the second output and see that they are exactly the same. This is also the case for the third variable, faminc98. While this is not absolute proof that the reshaping was successful, we can be pretty certain that it was.

Example 2: Reshaping one variable using an array

A second method of reshaping variables in a data step is to use an array statement. This method is useful if you have more than a few variables to reshape. We will begin with an example using only one variable, and then move on to an example with two variables to be reshaped.

As in the last example, we want to reshape the variables faminc96, faminc97 and faminc98 into two long variables, year and faminc. We will first show you the code used to accomplish this and then explain each piece of the code below.

```
“
DATA long1a;
  SET wide;

  ARRAY afaminc(96:98) faminc96 - faminc98 ;

  DO year = 96 to 98 ;
    faminc = afaminc(year);
    OUTPUT;
  END;

  DROP faminc96 - faminc98 ;
RUN;
```

Regarding the array statement (ARRAY afaminc(96:98) faminc96 – faminc98 ;), the name of the array is afaminc (many researchers will simply add an "a" (for array) to the new variable name to create the name of the array to make it easy to know what variable the array is working on). The numbers in parentheses (96:98) indicate the first and last numbers of the series to be reshaped. Finally, the actual variable names are listed. You can use a dash to indicate the inclusion of consecutive numbers.

On the first line of the do-loop (DO year = 96 to 98 ;), you put the name of the new variable that will contain the suffix for the old variables. On the second line of the do-loop, we set our new variable (faminc) equal to the value of the array for the given year (afaminc(year)), i.e., when year is 96 then afaminc(96) refers to faminc96.

We then use the output statement to force SAS to output the results before starting the loop over again. If this is omitted, only the record for the last observation in each group will be output and you will have only three records in the new data set instead of nine.

Finally, we use the drop statement to drop the variables from the wide data file that have been reshaped and are no longer needed.

Below we run proc print on the new data file and proc means on both the old and the new data sets to ensure that the reshaping went as expected.

```
PROC PRINT DATA=long1a ;
RUN ;
```

Obs	famid	year	faminc
1	1	96	40000
2	1	97	40500
3	1	98	41000
4	2	96	45000
5	2	97	45400
6	2	98	45800
7	3	96	75000
8	3	97	76000
9	3	98	77000

```
PROC MEANS DATA=wide fw = 8 ;
  VAR faminc96-faminc98 ;
RUN ;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
faminc96	3	53333.3	18929.7	40000.0	75000.0
faminc97	3	53966.7	19238.1	40500.0	76000.0
faminc98	3	54600.0	19546.9	41000.0	77000.0

```
PROC MEANS DATA=long1a fw=8 ;
  CLASS year ;
  VAR faminc ;
RUN ;
```

The MEANS Procedure

Analysis Variable : faminc

year	N Obs	N	Mean	Std Dev	Minimum	Maximum
96	3	3	53333.3	18929.7	40000.0	75000.0
97	3	3	53966.7	19238.1	40500.0	76000.0
98	3	3	54600.0	19546.9	41000.0	77000.0

The output from the proc print of the new data set looks as we expect: there are three observations per family and the variable year ranges from 96 to 99. We also compare the output of the proc means for the old and the new data sets. We compare the descriptive statistics for each variable to ensure that they did not change during the course of the reshaping. We see that they have not, which is a good indication that the reshaping was successful.

Example 3: Reshaping two variables using an array

This example is very similar to the last one except that now we will reshape two variables in the same data step. There are three places where this program has been modified from the version shown in the example above. They are denoted with a comment to the right of the statement in the program. Please note that you can reshape as many variables as you want in a single data step. To reshape additional variables, you would add an array statement, another line within the do-loop and drop the reshaped variables for each set of variables to be reshaped.

```
..
data wide2 ;
  input famid faminc96 faminc97 faminc98 spend96 spend97 spend98 ;
cards ;
1 40000 40500 41000 38000 39000 40000
2 45000 45400 45800 42000 43000 44000
3 75000 76000 77000 70000 71000 72000
;
RUN ;

DATA long2 ;
  SET wide2 ;

  ARRAY afaminc(96:98) faminc96-faminc98 ;
  ARRAY aspend(96:98) spend96-spend98 ;          * added statement ;

  DO year = 96 to 98 ;
    faminc = afaminc(year) ;
    spend  = aspend(year) ;                      * added statement ;
    OUTPUT ;
  END ;

  DROP faminc96-faminc98 spend96-spend98 ;       * added variables ;

RUN ;
```

As before, we check to ensure that the reshaping went as expected.

```
PROC PRINT DATA=long2 ;
RUN ;
```

Obs	famid	year	faminc	spend
1	1	96	40000	38000
2	1	97	40500	39000
3	1	98	41000	40000
4	2	96	45000	42000
5	2	97	45400	43000
6	2	98	45800	44000
7	3	96	75000	70000
8	3	97	76000	71000
9	3	98	77000	72000

```
PROC MEANS DATA=wide2 fw=8 ;
  VAR faminc96-faminc98 spend96-spend98 ;
RUN ;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
faminc96	3	53333.3	18929.7	40000.0	75000.0
faminc97	3	53966.7	19238.1	40500.0	76000.0
faminc98	3	54600.0	19546.9	41000.0	77000.0
spend96	3	50000.0	17435.6	38000.0	70000.0
spend97	3	51000.0	17435.6	39000.0	71000.0
spend98	3	52000.0	17435.6	40000.0	72000.0

```
PROC MEANS DATA=long2 fw=8 ;
  CLASS year ;
  VAR faminc spend ;
RUN ;
```

The MEANS Procedure

year	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
96	3	faminc	3	53333.3	18929.7	40000.0	75000.0
		spend	3	50000.0	17435.6	38000.0	70000.0
97	3	faminc	3	53966.7	19238.1	40500.0	76000.0
		spend	3	51000.0	17435.6	39000.0	71000.0
98	3	faminc	3	54600.0	19546.9	41000.0	77000.0
		spend	3	52000.0	17435.6	40000.0	72000.0

Example 4: A more realistic example

This example is much like example 2 in that only one variable (income) is being reshaped. However, this example is somewhat more realistic in that there are more years of income and more cases. You will note that the structure of the SAS code is identical to example 2; only the variable names are changed.

```
..
data wide3;
  input id inc90 inc91 inc92 inc93 inc94 inc95 ;
cards;
1  66483 69146 74643 79783 81710 86143
2  17510 17947 19484 20979 21268 22998
3  57947 62964 68717 70957 75198 75722
4  64831 71060 71918 72514 73100 74379
5  18904 19949 21335 22237 23829 23913
6  32057 34770 35834 37387 40899 42372
7  60551 64869 67983 70498 71253 75177
8  16553 18189 18349 19815 21739 22980
9  32611 33465 35961 36416 37183 40627
10 61379 66002 67936 70513 74405 76009
11 24065 24229 25709 26121 26617 28142
12 32975 36185 37601 41336 43399 43670
13 69548 71341 72455 76552 80538 85330
14 50274 53349 55900 59375 61216 63911
15 72011 73334 76248 77724 78638 80582
16 18911 20046 21343 21630 22330 23081
17 68841 75410 80806 81327 81571 86499
18 28099 30716 32986 36097 39124 39866
19 17302 18778 18872 19884 20665 21855
20 16291 16674 16770 17182 17979 18917
21 43244 46545 47633 50744 54734 59075
22 56393 59120 60801 61404 63111 69278
23 47347 49571 50101 51345 56463 56927
24 16076 17217 17296 17900 18171 18366
25 65906 69679 76131 77676 81980 85426
26 58586 61188 66542 69267 71063 74549
27 61674 66584 69185 75193 78647 81898
28 31673 31883 32774 34485 36929 39751
29 63412 67593 69911 73092 80105 81840
30 27684 28439 30861 31406 32960 35530
31 71873 76449 80848 88691 94149 97431
32 62177 63812 64235 65703 69985 71136
33 37684 38258 39208 39489 39745 41236
34 64013 66398 71877 75610 76395 79644
35 16011 16847 17746 19123 19183 19996
36 49215 52195 52343 56365 58752 59354
37 15774 16643 17605 18781 18996 19685
38 29106 31693 31852 34505 35806 36179
39 25147 26923 28785 30987 34036 34106
40 71978 79144 80453 86580 95164 96155
41 46166 47579 49455 53849 56630 57473
42 55810 59443 65291 66065 69009 74365
43 49642 50603 53917 54858 58470 59767
44 21348 22361 23412 24038 24774 25828
45 44361 48720 51356 54927 56670 58800
46 56509 60517 61532 65077 69594 73089
47 39097 40293 43237 44809 48782 53091
48 18685 19405 20165 20316 22197 23557
49 73103 76243 76778 82734 86279 86784
50 48129 49267 53799 58768 63011 66461
;
RUN ;

DATA long3 ;
  SET wide3 ;

  ARRAY ainc(90:95) inc90 - inc95 ;

  DO year = 90 to 95 ;
    inc = ainc(year) ;
    OUTPUT ;
  END ;

  DROP inc90 - inc95 ;
RUN ;
```

Let's start our checking of the reshaping by looking at proc prints of the first five observations of both the old and the new data files. Remember that to see the data for the first five observations in the wide data set, you will need the first 30 observation in the long data set (five observations times six variables = 30). Next, we will look at the results of the proc means for both data sets.

```
PROC PRINT DATA=wide3(obs = 5) ;
RUN ;
```

Obs	id	inc90	inc91	inc92	inc93	inc94	inc95
1	1	66483	69146	74643	79783	81710	86143
2	2	17510	17947	19484	20979	21268	22998
3	3	57947	62964	68717	70957	75198	75722
4	4	64831	71060	71918	72514	73100	74379
5	5	18904	19949	21335	22237	23829	23913

```
PROC PRINT DATA=long3(obs = 30) ;
RUN ;
```

Obs	id	year	inc
1	1	90	66483
2	1	91	69146
3	1	92	74643
4	1	93	79783
5	1	94	81710
6	1	95	86143
7	2	90	17510
8	2	91	17947
9	2	92	19484
10	2	93	20979
11	2	94	21268
12	2	95	22998
13	3	90	57947
14	3	91	62964
15	3	92	68717
16	3	93	70957
17	3	94	75198
18	3	95	75722
19	4	90	64831
20	4	91	71060
21	4	92	71918
22	4	93	72514
23	4	94	73100
24	4	95	74379
25	5	90	18904
26	5	91	19949
27	5	92	21335
28	5	93	22237
29	5	94	23829
30	5	95	23913

```
PROC MEANS DATA = wide3 fw=8 ;
  VAR inc90-inc95 ;
RUN;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
inc90	50	43899.3	19523.4	15774.0	73103.0
inc91	50	46380.7	20749.4	16643.0	79144.0
inc92	50	48519.6	21720.1	16770.0	80848.0
inc93	50	50842.3	22780.1	17182.0	88691.0
inc94	50	53289.0	23824.0	17979.0	95164.0
inc95	50	55379.0	24592.8	18366.0	97431.0

```
PROC MEANS DATA = long3 fw=8  ;
  CLASS year ;
  VAR inc ;
RUN;
```

```
The MEANS Procedure
                        Analysis Variable : inc

              N
    year     Obs       N        Mean     Std Dev     Minimum     Maximum
-------------------------------------------------------------------------
      90      50      50     43899.3     19523.4     15774.0     73103.0
      91      50      50     46380.7     20749.4     16643.0     79144.0
      92      50      50     48519.6     21720.1     16770.0     80848.0
      93      50      50     50842.3     22780.1     17182.0     88691.0
      94      50      50     53289.0     23824.0     17979.0     95164.0
      95      50      50     55379.0     24592.8     18366.0     97431.0
-------------------------------------------------------------------------
```

Example 5: Reshaping with a string variable

This example is very similar to example 3, except we will add a string (i.e., character) variable that also needs to be reshaped. In this example we will reshape three variables, faminc, spend and debt. Note that in this data set, debt is a string variable. Fortunately, reshaping string variables is as easy reshaping numeric variables. Note that the reshaped variables that are based on the string variable will be string variables in the new data set, so you cannot include them in the proc means to check if the variables were reshaped correctly. However, we can do a proc freq to check the reshaping of the string variables. Also, we have included a length statement after the set statement to set the length of our new string variable debt. If we did not include this statement, SAS would assign the length of the variable to be the same as the first value encountered. In this example, the first value is "yes", which happens to be the longest string in this variable. However, if "no" was the first value SAS encountered, then the length of debt would be set to 2, and instead of seeing "yes", we would see "ye".

```
..
DATA wide4;
  INPUT famid faminc96 faminc97 faminc98 spend96 spend97
  spend98 debt96 $ debt97 $ debt98 $ ;
cards;
1 40000 40500 41000 38000 39000 40000 yes yes no
2 45000 45400 45800 42000 43000 44000 yes no  no
3 75000 76000 77000 70000 71000 72000 no  no  no
;
RUN ;

DATA long4 ;
  SET wide4 ;
  LENGTH debt $ 3;

  ARRAY afaminc(96:98) faminc96-faminc98 ;
  ARRAY aspend(96:98) spend96-spend98 ;
  ARRAY adebt(96:98) debt96-debt98 ;

  DO year = 96 to 98 ;
    faminc = afaminc(year) ;
    spend = aspend(year) ;
    debt = adebt(year) ;
    OUTPUT ;
  END;

  DROP faminc96-faminc98 spend96-spend98 debt96-debt98 ;

RUN;

PROC PRINT DATA=long4 ;
RUN ;
```

Obs	famid	year	faminc	spend	debt
1	1	96	40000	38000	yes
2	1	97	40500	39000	yes
3	1	98	41000	40000	no
4	2	96	45000	42000	yes
5	2	97	45400	43000	no
6	2	98	45800	44000	no
7	3	96	75000	70000	no
8	3	97	76000	71000	no
9	3	98	77000	72000	no

```
PROC MEANS DATA=wide4;
  VAR faminc96-faminc98 spend96-spend98;
RUN;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
faminc96	3	53333.33	18929.69	40000.00	75000.00
faminc97	3	53966.67	19238.07	40500.00	76000.00
faminc98	3	54600.00	19546.87	41000.00	77000.00
spend96	3	50000.00	17435.60	38000.00	70000.00
spend97	3	51000.00	17435.60	39000.00	71000.00
spend98	3	52000.00	17435.60	40000.00	72000.00

```
PROC MEANS DATA=long4;
  CLASS year;
  VAR faminc spend debt;
RUN;
```

```
The MEANS Procedure

                N
        year   Obs   Variable   N           Mean        Std Dev        Minimum
-------------------------------------------------------------------------------
          96     3   faminc     3       53333.33       18929.69       40000.00
                     spend      3       50000.00       17435.60       38000.00

          97     3   faminc     3       53966.67       19238.07       40500.00
                     spend      3       51000.00       17435.60       39000.00

          98     3   faminc     3       54600.00       19546.87       41000.00
                     spend      3       52000.00       17435.60       40000.00
-------------------------------------------------------------------------------

                N
        year   Obs   Variable        Maximum
-----------------------------------------
          96     3   faminc         75000.00
                     spend          70000.00

          97     3   faminc         76000.00
                     spend          71000.00

          98     3   faminc         77000.00
                     spend          72000.00
-----------------------------------------
```

```
PROC FREQ DATA=wide4;
  TABLE debt96 debt97 debt98;
RUN ;
```

```
The FREQ Procedure

                                          Cumulative    Cumulative
debt96    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
no                1       33.33             1        33.33
yes               2       66.67             3       100.00


                                          Cumulative    Cumulative
debt97    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
no                2       66.67             2        66.67
yes               1       33.33             3       100.00


                                          Cumulative    Cumulative
debt98    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
no                3      100.00             3       100.00
```

```
PROC FREQ DATA=long4;
  TABLE year*debt / norow nocol nopercent ;
RUN ;
```

```
The FREQ Procedure

Table of year by debt

year      debt

Frequency|no      |yes     |  Total
---------+--------+--------+
      96 |      1 |      2 |      3
---------+--------+--------+
      97 |      2 |      1 |      3
---------+--------+--------+
      98 |      3 |      0 |      3
---------+--------+--------+
Total           6        3        9
```

When comparing the output from the proc freq for the old data set with the one for the new data set, we can see that the distribution of debt is the same in each of the years for the old data file as in the new data file.

Example 6: Character suffixes

All of the previous examples have shown how to reshape variables that have had numeric suffixes. However, you can reshape variables that have string (i.e., character) suffixes as well. The only modification to the "template" is in the array statement. In our example, we have simply listed the variables. For example, ARRAY aname(2) named namem ; contains the elements named and namem. However, this could be cumbersome if you have many elements in the array. If the elements are positionally consecutive in the data set, you can separate the first and last element with a double dash (–). In SAS, one dash (-) indicates elements that are numerically consecutive, while two dashes (–) indicate elements that are positionally consecutive.

```
DATA wide5;
  INPUT famid named $ incd namem $ incm ;
CARDS;
1.00 Bill 30000.00 Bess 15000.00
2.00 Art 22000.00 Amy 18000.00
3.00 Paul 25000.00 Pat 50000.00
;
RUN;

DATA long5 ;
  SET wide5 ;
  LENGTH name $ 4;

  ARRAY aname(2) named namem ;
  ARRAY ainc(2) incd incm ;

  DO parent = 1 to 2 ;
    name = aname(parent) ;
    inc  = ainc(parent) ;
    OUTPUT ;
  END ;

  DROP named namem incd incm ;

RUN ;

PROC PRINT DATA=long5;
RUN;
```

Obs	famid	name	parent	inc
1	1	Bill	1	30000
2	1	Bess	2	15000
3	2	Art	1	22000
4	2	Amy	2	18000
5	3	Paul	1	25000
6	3	Pat	2	50000

```
PROC MEANS DATA=wide5;
  VAR incd incm;
RUN;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
incd	3	25666.67	4041.45	22000.00	30000.00
incm	3	27666.67	19399.31	15000.00	50000.00

```
PROC MEANS DATA=long5;
  VAR inc;
RUN;
```

The MEANS Procedure

Analysis Variable : inc

N	Mean	Std Dev	Minimum	Maximum
6	26666.67	12580.41	15000.00	50000.00

```
PROC FREQ DATA=wide5;
  TABLE named namem;
RUN;
```

```
The FREQ Procedure
                                          Cumulative    Cumulative
named    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
Art              1       33.33             1        33.33
Bill             1       33.33             2        66.67
Paul             1       33.33             3       100.00


                                          Cumulative    Cumulative
namem    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
Amy              1       33.33             1        33.33
Bess             1       33.33             2        66.67
Pat              1       33.33             3       100.00

PROC FREQ DATA=long5;
  TABLE parent name;
RUN;

The FREQ Procedure
                                          Cumulative    Cumulative
parent    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
     1           3       50.00             3        50.00
     2           3       50.00             6       100.00


                                          Cumulative    Cumulative
name     Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
Amy              1       16.67             1        16.67
Art              1       16.67             2        33.33
Bess             1       16.67             3        50.00
Bill             1       16.67             4        66.67
Pat              1       16.67             5        83.33
Paul             1       16.67             6       100.00
```

Click here to report an error on this page or leave a comment

How to cite this page (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-cite-web-pages-and-programs-from-the-ucla-statistical-consulting-group/)