Paper 044-2012

Magic Spells with SAS®

Christopher J. Bost, MDRC, New York, NY

ABSTRACT

Programmers often need to calculate spells, or the number of consecutive periods of time under different conditions. For example, you might need to calculate months of employment and unemployment, or days on and off a medication. This paper describes how to use multiple arrays and index variables to identify the beginnings and endings of spells, and how to summarize the number, minimum, maximum, and average length of spells.

INTRODUCTION

Calculating spells by hand is simple. Calculating spells with SAS® might seem complicated at first given the myriad possible combinations. The steps needed to calculate spells by hand, however, can be identified and "translated" into SAS code.

SAMPLE DATA

SAS data set EMPLOYED is used in this paper:

id	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12
1	1	1	0	0	1	1	1	0	1	1	1	1
2	0	0	1	1	1	0	1	1	1	0	0	0
3	1	0	0	1	1	0	1	1	0	0	0	1

Output 1. Data set EMPLOYED

Data set EMPLOYED contains three observations and 13 variables. Each observation has a unique ID value and twelve variables, M1 through M12, representing twelve calendar months. Each month variable has the value 0 or 1: 0 when the subject was not employed during that month and 1 when the subject was employed during that month.

CALCULATING SPELLS BY HAND

Just a quick glance at the first observation in data set EMPLOYED reveals that there are three spells (i.e., one or more consecutive months of employment [1] followed by one or more consecutive months of unemployment [0] or reaching the last month variable). The first spell lasts two months (M1-M2); the second spell lasts three months (M5-M7); and the third spell lasts four months (M9-M12). The minimum spell length is 2; the maximum spell length is 4; and the average spell length is 3 ((2+3+4)/3=3).

Note that while it is easy to calculate spells by hand, we are actually performing multiple tasks simultaneously. First, we process all monthly variables in order from M1-M12. Second, we note when we initially encounter a 1. Third, we ignore additional 1s until we encounter a 0; then we know that the spell ended at the previous variable. (We could count 1s, but picking up the beginning and ending will capture all of the information about the spell.) Fourth, we continue processing monthly variables in the same way until we reach M12; if we are currently in a spell, it ends there. Fifth, we subtract the starting position of each spell from the ending position and add 1 to determine the spell length. Finally, we use basic arithmetic to calculate the number of spells and the minimum, maximum, and average length of spells.

All of these tasks can be performed with SAS.

CALCULATING SPELLS WITH SAS

Calculating spells with SAS requires the same tasks needed to calculate spells by hand. The following SAS code is broken down into three steps: 1) creating arrays for source variables and "spell variables"; 2) processing source variables; and 3) processing spell variables.

STEP 1: CREATING ARRAYS FOR SOURCE VARIABLES AND SPELL VARIABLES

Create three arrays: SOURCE is for the source variables M1 through M12; BEG is for variables that store when spells begin; and FIN is for variables that store when spells end:

```
array source{12} m1-m12;
array beg{6} beg1-beg6;
array fin{6} fin1-fin6;
```

The maximum number of spells possible in the sample data set is six; therefore, six pairs of variables are created. Array BEG references six variables, BEG1 through BEG6. Array FIN also references six variables, FIN1 through FIN6. These variables do not already exist, so the ARRAY statements create them and initialize them to missing.

Two additional variables are needed: one variable is an index to move through the BEG and FIN arrays, and the other variable is a flag to indicate whether or not a spell is in progress:

```
j=1;
inspell=0;
```

Initialize the index variable J to 1 (i.e., to point to BEG1 and FIN1, respectively).

Initialize the flag variable INSPELL to 0, or false (i.e., not in a spell).

STEP 2: PROCESSING SOURCE VARIABLES

Use a DO loop to iterate through all twelve elements in array SOURCE:

```
do i=1 to 12;
```

I is an index variable that is assigned values from 1 to 12.

Next, define the three conditions of interest in calculating spells: a) the beginning of a spell; b) the ending of a spell; and c) reaching the last source variable. The SAS code below identifies each condition and the corresponding action(s) to take.

A. BEGINNING OF A SPELL

When a source variable first indicates that a spell begins, assign the position of that variable in array SOURCE to the variable in array BEG pointed to by J:

```
if source{i}=1 and inspell=0 then do;
  beg{j}=i;
  inspell=1;
end;
```

INSPELL was initialized to 0. When a source variable equals 1, indicating the beginning of a spell, set its value to 1.

SAS continues to loop through the source variables. Note that there is no need to count each month in a spell.

B. ENDING OF A SPELL

After a spell has started, the next condition of interest is when the spell ends:

```
if inspell=1 and source{i}=0 then do;
  fin{j}=i-1;
  j=j+1;
  inspell=0;
end;
```

When in a spell (INSPELL=1) and the next source variable equals 0, the spell has ended. Assign the position of the current variable in array SOURCE minus 1 (i.e., to go back one period of time to when the spell ended) to the variable in array FIN pointed to by J.

Values for the beginning and ending of a spell have now been assigned. Increment the value of J by 1 to point to the next pair of variables (i.e., to allow for additional spells).

Assign the value 0 to INSPELL to indicate that a spell is no longer in progress.

SAS continues to loop through the remaining source variables. Additional spells are calculated the same way.

C. REACHING THE LAST SOURCE VARIABLE

If the last source variable has been reached and a spell is still in progress, assign the position of the last variable in array SOURCE (i.e., 12) to the variable in array FIN pointed to by J. Close the DO I=1 TO 12; loop with END:

```
if i=12 and inspell=1 then fin{j}=12;
end;
```

The beginning and ending of each spell have now been assigned to the pairs of variables in arrays BEG and FIN.

Steps 1 and 2 produce the following BEG and FIN variables:

id	beg1	beg2	beg3	beg4	beg5	beg6	fin1	fin2	fin3	fin4	fin5	fin6
1	1	5	9	.	.	.	2	7	12	.	.	.
2	3	7	.	.	.	.	5	9	.	.	.	.
3	1	4	7	12	.	.	1	5	8	12	.	.

Output 2. BEG and FIN variables

STEP 3: PROCESSING SPELL VARIABLES

Processing spell variables requires: a) calculating the length of each spell, and b) calculating summary statistics on all spells within each observation.

A. CALCULATING THE LENGTH OF EACH SPELL

Create array LEN for variables that store the length of each spell:

```
array len{6} len1-len6;
```

Array LEN references six variables, LEN1 through LEN6. These variables do not already exist, so the ARRAY statement creates them and initializes them to missing.

Use a DO loop to iterate through all six elements. I and J were previously used as index variables, so K is used:

```
do k=1 to 6;
  if beg{k} ne . then len{k}=fin{k}-beg{k}+1;
end;
```

The IF statement checks if the current variable in array BEG pointed to by K is not missing. If so, it means there is a pair of values for a spell. Subtract the beginning of the spell from the ending of the spell, add one for the ending period, and assign the value to the variable in array LEN pointed to by K.

B. CALCULATING SUMMARY STATISTICS

Calculate the number, minimum, maximum, and average length of spells using the respective function:

```
Nspells   =    n(of len{*});
MinSpell  =  min(of len{*});
MaxSpell  =  max(of len{*});
MeanSpell = mean(of len{*});
```

Note that OF LEN{*} is “shorthand” for all variables in array LEN. The OF operator is required.

Step 3 produces the following LEN variables and summary statistic variables:

id	len1	len2	len3	len4	len5	len6	Nspells	Min Spell	Max Spell	Mean Spell
1	2	3	4	.	.	.	3	2	4	3.0
2	3	3	.	.	.	.	2	3	3	3.0
3	1	2	2	1	.	.	4	1	2	1.5

Output 3. LEN variables and summary statistic variables

FINAL PROGRAM

The above code is incorporated into a single program below:

```
data spells;
set employed;

**Step 1: Creating arrays for source variables and spell variables;

array source{12} m1-m12;
array beg{6} beg1-beg6;
array fin{6} fin1-fin6;

j=1;       *initialize index variable for BEG and FIN arrays;
inspell=0; *initialize flag variable to 0 (not in a spell);

**Step 2: Processing source variables;

do i=1 to 12;

*A. Beginning of a spell;
if source{i}=1 and inspell=0 then do;
  beg{j}=i;
  inspell=1;
end;

*B. Ending of a spell;
if inspell=1 and source{i}=0 then do;
  fin{j}=i-1;
  j=j+1;
  inspell=0;
end;

*C. Reaching the last source variable;
if i=12 and inspell=1 then fin{j}=12;

end;

**Step 3: Processing spell variables;

*A. Calculating the length of each spell;
array len{6} len1-len6;
do k=1 to 6;
  if beg{k} ne . then len{k}=fin{k}-beg{k}+1;
end;

*B. Calculating summary statistics;
Nspells   =    n(of len{*});
MinSpell  =  min(of len{*});
MaxSpell  =  max(of len{*});
MeanSpell = mean(of len{*});

run;
```

CONCLUSION

Multiple arrays and index variables can be used to identify the beginnings and endings of spells. SAS functions can then be used to calculate the number, minimum, maximum, and average length of spells for each observation.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Christopher J. Bost
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
(212) 340-8613 telephone
(212) 684-0832 fax
christopher.bost@mdrc.org
www.mdrc.org