

Using Recursion for More Convenient Macros

Nate Derby

Stakana Analytics
Seattle, WA

Winnipeg SAS Users Group, 4/28/11

Outline

- 1 Introduction
 - Why Use Recursion?
- 2 Implementation
 - Three Easy Steps
 - Does It Work?
- 3 Conclusions

Basic Idea

Suppose we want to make forecasts for flight 1542:

```
%makeForecasts ( fnumber=1542 )
```

Now we want to make forecasts for all flights:

```
%makeForecasts ( fnumber=1542 )
```

```
%makeForecasts ( fnumber=1543 )
```

```
%makeForecasts ( fnumber=1544 )
```

Problems with above:

- We have to find the right flight numbers.
- We have to list them out individually.

Better Idea

To make forecasts for flight 1542:

```
%makeForecasts ( fnumber=1542 )
```

To make forecasts for all flights:

```
%makeForecasts
```

Flight numbers are determined, listed automatically.

- Easy for testing (test for one before performing for all)
- Easy for drill-down (perform for all, then in depth for one)

Step One

Define the macro for parameter:

```
%makeForecasts  
%MACRO makeForecasts( fnumber );  
  
    [Code for making forecasts]  
  
%MEND makeForecasts;
```

Step One

Really easy example:

```
%makeForecasts  
%MACRO makeForecasts( fnumber );  
  
    %PUT fnumber=&fnumber;  
  
%MEND makeForecasts;
```

Step Two

Step 2: Create auxiliary macro for getting flight numbers:

```
%getFlightNumbers
%MACRO getFlightNumbers;

  PROC SQL NOPRINT;
    SELECT DISTINCT flightnumber INTO :fnumbers
      SEPARATED BY ' '
    FROM datasource
    WHERE orig="&orig" AND dest="&dest"
    ORDER BY by flightnumber;
  QUIT;

%MEND getFlightNumbers;
```

Step Three: Putting It All Together

%makeForecasts

```
%MACRO makeForecasts( fnumber=all );

  %LOCAL i n fnumbers;
  %IF &fnumber = all %THEN %DO;

    %getFlightNumbers;

    %LET i = 1;
    %DO %WHILE( %LENGTH( %SCAN( &fnumbers, &i ) ) > 0 );
      %LOCAL fnumber&i;
      %LET fnumber&i = %SCAN( &fnumbers, &i );
      %LET i = %EVAL( &i + 1 );
      %END;
    %LET n = %EVAL( &i - 1 );

    %DO i=1 %TO &n;
      %makeForecasts( fnumber=&&fnumber&i );
      %END;

    %GOTO theend;
  %END;

  %PUT fnumber=&fnumber;

%theend:

%MEND makeForecasts;
```

Does It Work?

Macro call

Log output

```
%makeForecasts
```

⇒

```
fnumber=1542  
fnumber=1543  
fnumber=1544
```

```
%makeForecasts( fnumber=1542 )
```

⇒

```
fnumber=1542
```

Conclusions

- A recursive definition can make a macro more convenient.
- Recursion can be applied to one or more parameters.
(see paper!)
- It can work especially well within a larger framework.

Further Resources



Art Carpenter.

Carpenter's Complete Guide to the SAS Macro Language.
SAS Press, 2004.



Nate Derby.

Suggestions for Organizing SAS Code and Project Files.
<http://nderby.org> (Publications → Manuscripts), 2010.

Nate Derby: <http://nderby.org>

nderby@stakana.com