Paper 343-2013

Writing Macro Do Loops with Dates from Then to Now

Ronald J. Fehd, SAS-L peer, Atlanta, GA, USA

Abstract

Description : Dates are handled as numbers with formats in SAS® software. The SAS macro language is a text-handling language. Macro %do statements require integers for their start and stop values.

Purpose : This article examines the issues of converting dates into integers for use in macro %do loops. Three macros are provided: a template to modify for reports, a generic calling macro function which contains a macro %do loop and a function which returns a list of dates. Example programs are provided which illustrate unit testing and calculations to produce reports for simple and complex date intervals.

Audience : macro programmers

Information : Programs in this paper are available on the web: Fehd [7, sco.Macro-Loops-With-Dates]

Keywords : do and %do statements; interval incrementing (intnx): intervals and shift-index; month, putn, %sysevalf, %sysfunc, today, day-of-the-week (weekday), year

Quote :

> The White Rabbit put on his spectacles. 'Where shall I begin, please your Majesty?' he asked. '[Start] at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.'
> — Lewis Carroll, English author and recreational mathematician (1832–1898), Alice's Adventures in Wonderland.

Contents

Introduction

Overview

This article examines the issues of obtaining integers from date literals for use in macro %do loops. Examine these examples.

- iterative do loop in data step
- error in a macro %do loop
- math for `intnx`

Data Loop

This data step shows an iterative do loop first with integers and then integers resolved from date literals, which are referencs to numbers.

```
DATA _Null_;
do date = 0 to 1;
   put date= 8. date date.;
   end;
put;
do date = '3Jan1960'd to '4Jan1960'd;
   put date= 8. date date.;
   end;
stop;
run;
```

```
date=0 01JAN60
date=1 02JAN60

date=2 03JAN60
date=3 04JAN60
```

Macro Error

This log shows that a macro %do loop cannot accept a reference but must have a value. This is an issue in all artificial languages and is addressed in Contributors [1, www-wiki.Call-by-Reference].

```
1  %DateLoop(loop_start ='1Jan1960'd
2           ,loop_stop  ='7Jan1960'd)
ERROR: A character operand was found in the %EVAL function
       or %IF condition where a numeric operand is required.
       The condition was: &loop_start
```

The problems addressed in this article are how to resolve references — date literals — so that a macro %do loop works like a data step loop.

Math or Intnx?

Can we calculate the beginning and ending dates of an interval, easily? Yes, if we remember the adjustments. Using the `intnx` function is easier.

```
%Let today           = %sysevalf('01May2013'd);
%Let day_of_week     = %sysfunc(weekday(&today));
%Let this_week_begin = %eval(&today - &day_of_week + 1);
%Let this_week_end   = %eval(&today - &day_of_week + 7);

this_week_begin: 19476       Sunday, April 28, 2013
today          : 19479       Wednesday, May 1, 2013
this_week_end  : 19482        Saturday, May 4, 2013
```

Information

Overview

Dates are numbers. The mathematical operations associated with dates are addition and subtraction. The macro language handles text. The problem is how to convert text references — a date literal which represents an integer — into an integer to be used as the argument to a macro %do loop.

Data Step

To obtain an integer use the `today` function or a date literal.

```
data _null_;
zero    = 0;
earlier = '21Dec2012'd;
today   = today();
put zero    = zero     weekdate29.;
put earlier= earlier mmddyy10.  ;
put today   = today    date9.     ;
stop;
run;

zero    =     0 Friday, January 1, 1960
earlier=19348 12/21/2012
today   =19387 29JAN2013
```

Macros

In macros we use several different functions while working with dates.

assignment : use `%sysfunc` and `today` functions or `%sysevalf` a date literal

```
%let today = %sysfunc(today());
%let today = %sysevalf('01May2013'd);
```

intnx : get the surrounding dates of the interval by using the `intnx` function

```
*syntax: intnx(interval in (day, week, month, quarter, year)
              ,start-from: an integer of a date
              ,increment: an integer: negative zero or positive
              ,alignment in (begin, middle, end, same);
%let interval = week;
%let D_Begin  =%sysfunc(intnx(&interval,&today,0,begin));
%let D_End    =%sysfunc(intnx(&interval,&today,0,end  ));
```

Note: in this example the third argument of `intnx` is zero which returns the begin and end of the current time interval. Later usage in calling programs has minus one to get the begin and end of the previous interval.

display : use the `putn` function with different formats to display the integer of a date in a macro variable

```
%put D_Begin: &D_Begin %sysfunc(putn(&D_Begin,mmddyy10.  ));
%put today  : &today   %sysfunc(putn(&today   ,weekdate29.));
%put D_End  : &D_End   %sysfunc(putn(&D_End   ,date9.     ));
     D_Begin: 19476                   04/28/2013
     today  : 19479    Wednesday, May 1, 2013
     D_End  : 19482                    04May2013
```

Fehd [4, sco.Macro-Vars-of-Date-and-Time] contains other functions and formats used to display dates in titles and footnotes.

Macros to Handle Dates

Overview

The following sections show macros Date-Report-Zero, DateLoop and ccYY-MM. Macro function DateLoop calls Date-Report-Zero with the parameters Date-Begin and Date-End. Macro ccYY-MM is a subroutine of DateLoop which returns a list of dates.

The programs provided here are basic unit tests and demonstrations. For other programs see Fehd [7, sco.Macro-Loops-With-Dates].

Header

Each program has a common header.

```
 /*     name: dateloop-demo.sas
description: testing
    purpose: template
    /******/
options mprint;
%let today = %sysfunc(today());
%let today = %sysevalf('01May2013'd);*unit testing;
```

Test Data

This program is used to generate test data.

test-data.sas

```
 /*     name: test-data
description: provide data with dates
purpose    : for use by other demo programs
usage:
%daterpt0(data = Library.TestData
         ,var  = date
         ,...);
         /*******/
DATA Library.TestData;
     attrib EntityId length = 4
            Date     length = 8 format = weekdate17.
            Fact     length = 8;
do Date = 0 to '01May2013'd;
   EntityId = int(Date   /17);
   Fact     = int(Date**3/19);
   output;
   end;
stop;
run;
```

Calendar

Use this calendar when comparing dates in logs.

2013						
day 1 Sun	day 2 Mon	3 Tues	4 Wed	5 Thurs	6 Fri	day 7 Sat
Mar 31	Apr 1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	**29**	**30**	**May 1**			
	SGF			2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Macro Date-Report-Zero

Overview

This section presents a simple model — a template for a macro that can be customized — with formats and macro functions for assembling ODS filenames.

Model

A reporting macro needs the following parameters.

- data set name
- variable name
- low
- high

This example shows the data set option `where` with a between condition.

```
1  PROC Print data  = sashelp.class
2            (where = ( 11 <= age <= 13) );
NOTE: There were 10 observations read from the
      data set SASHELP.CLASS.
      WHERE (age>=11 and age<=13);
4  PROC SQL; select *
5            from sashelp.class
6            where age between 11 and 13;
7            quit;
```

Why Zero?

Dijkstra [2, utexas-trans.Why-Numbering-Starts-At-Zero] explains why zero is a good place to start: to avoid off-by-one errors.

Program Date Report Zero

This simple macro shows the basics of a reporting macro subroutine.

- ODS
- data set name
- variable
- date: begin
- date: end

daterpt0.sas

```
 /*    name: DateRpt0.sas
description: report with date-begin and -end
    purpose: template
    /******/
%MACRO daterpt0
       (data       = sashelp.class
       ,var        = age
       ,date_begin = 11
       ,date_end   = 13
       ,testing    = 0
       );
ods _all_ close;
ODS PDF
    file = "demo-ods-%sysfunc(juldate7(&Date_Begin)).pdf"
    ;
PROC Print data = &Data.
          (where = (&Date_Begin <= &Var <= &Date_End) );
title3 "data = &Data. date.var: &Var in range "
          "%sysfunc(putn(&Date_Begin,worddate18.)) "
       "-- %sysfunc(putn(&Date_End  ,mmddyy10.  ))"
       ;
run;
ods _all_ close;
ods listing;
%mend daterpt0;
```

Notes:

- the print procedure is used in this example; provide your own when modifing
- to refresh date in title1 use `options dtreset;`
- title and footnote statements may have multiple text strings as their arguments;

  ! → remember to end each string with a space
- dates may be displayed with various formats; see Fehd [4, sco.Macro-Vars-of-Date-and-Time] for other formats and functions; to justify text in title or footnote see Fehd [3, sco.Macro-TextLine]
- unit tests of this macro are on the web: Fehd [7, sco.Macro-Loops-With-Dates] See program info-in-dates.sas and Fehd [4, sco.Macro-Vars-of-Date-and-Time] for functions and formats to use in placing date-stamps in ODS filenames.

Date-Report-Zero Demo

This program can be used to test macro daterpt0 for any previous interval. The `intnx` increment value of minus one is used to provide dates of the previous interval.

daterpt0-demo-prev-any.sas
```
%let today = %sysevalf('01May2013'd);

*choice: previous which?;
%let interval = year;
%let interval = quarter;
%let interval = month;
%let interval = week;

%let begin =%sysfunc(intnx(&interval.,&today,-1,begin));
%let end   =%sysfunc(intnx(&interval.,&today,-1,end  ));

%put today: &today %sysfunc(putn(&today,weekdate29.));
%put begin: &begin %sysfunc(putn(&begin,weekdate29.));
%put end  : &end   %sysfunc(putn(&end  ,weekdate29.));

%daterpt0(data       =library.testdata
         ,var        =date
         ,date_begin =&begin
         ,date_end   =&end
         );
```

The log shows the beginning and ending dates of the previous week.

```
today: 19479  Wednesday,   May 1, 2013
begin: 19469  Sunday,    April 21, 2013
end  : 19475  Saturday, April 27, 2013
```

Note: The above program shows the calculation of the begin and end dates before calling the macro in order to write the values and dates to the log.

In all later examples these calculation are in the macro call.

```
*report previous &interval by day;
%dateloop(loop_start = %sysfunc(intnx(&interval.,&today,-1,begin))
         ,loop_stop  = %sysfunc(intnx(&interval.,&today,-1,end  ))
         ,interval   = day
         );
```

Reference or Value?

Contributors [1, www-wiki.Call-by-Reference] explains the difference between calling by reference and calling by value.

Macro DateLoop

Overview

Macro DateLoop is a function: its output is either a macro call or notes written in the log while testing. Its purpose is to encapsulate a macro %do loop through a series of dates while generating calls to a macro reporting subroutine.

It has parameters which support these design elements:

- `%do` loop
- `intnx` interval
- called macro, passing parameters to
- testing

`%do` loop :
- loop-start: date-begin
- loop-stop: date-end

`intnx` interval : simple: year, quarter, month, week, day etc.;

complex:

- multiple: intervalN: week2 is 14 days
- shift-index: inverval.N, the default is week.1 which returns the week starting on Sunday; `interval=week.2` returns the week starting on Monday

called macro : and passing parameters to

- MacroName: the default is `put note:` for testing
- MacroParms: this parameter must be enclosed in the `%nrstr` function to hide the special characters equal sign and comma
- called macro — e.g.: Date-Report-Zero — parameters must be Date-Begin and Date-End

testing : this variable is provided for self-reporting, see: Fehd [8, nesug2007.cc12] and is related to:

- MacroName, whose default is `put note:`
- semicolon: if the MacroName is `put note:` then each macro call is a `%put` statement which must end with a semicolon

see also : Fehd [9, pnwsug2009.do-which] compares data and macro do iterative, until and while.

Program DateLoop

dateloop.sas

```
 /*    name: ...\SAS-site\macros\dateloop.sas
     author: Ronald J. Fehd  2013
description: macro function do loop from loop-start to loop-stop
purpose: standardize calling of report macros for many intervals
NOTES: called macro must have parameters Date_Begin and Date_End
sas.help: Incrementing Dates and Times
             by Using Multipliers and by Shifting Intervals
sas.wiki: http://www.sascommunity.org/wiki/Macro_Loops_with_Dates
    http://www.sascommunity.org/wiki/Do_which_loop_until_or_while
predecessor: http://www.sascommunity.org/wiki/Macro_CallMacr
***********/
%Macro dateloop
         (loop_start =0 /* integer of date */
         ,loop_stop  =2 /* integer of date */
         ,interval   =day /* simple in (week month quarter year)
 /* intnx(interval: complex: week2==14 days week.2==Monday */
         ,MacroName  =put note:
         ,MacroParms = /*%nrstr(data=sashelp.class,var=sex)*/
         ,semicolon  =0
         ,testing    =0
         )/des = 'site: calls reporting macro with dates';
%local Format MacroCall D_Begin D_End;
%let Format = weekdate29.;
%if %scan(&MacroName,1) eq put %then %let Semicolon = 1;
%let Testing = %eval(   &Testing      or &Semicolon
                     or %sysfunc(getoption(mprint)) eq MPRINT);
%put &SysMacroName start: %sysfunc(putn(&loop_start,&format));
%put &SysMacroName  stop: %sysfunc(putn(&loop_stop ,&format));
%let D_Begin = %sysfunc(intnx(&Interval,&loop_start,0,begin));

%do %until(&D_Begin gt &loop_stop);
    %let D_End = %sysfunc(intnx(&Interval,&D_Begin,0,end  ));
    %let MacroCall = &MacroName(;
    %if %length(&MacroParms) %then
        %let MacroCall = &MacroCall.%unquote(&MacroParms,);
    %let MacroCall=&MacroCall.date_begin=&D_Begin,date_end=&D_End);
    %if &Testing %then %do;
        %put &SysMacroName: begin %sysfunc(putn(&D_Begin,&format));
        %put &SysMacroName:   end %sysfunc(putn(&D_End  ,&format));
        %end;
    %&MacroCall.
    %if &Semicolon %then %do;
        ;
        %end;
    %let D_Begin = %eval(&D_End +1);
    %end;
%mend dateloop;
```

Notes:

- unit tests of this macro are in Fehd [7, sco.Macro-Loops-With-Dates]
- parameters MacroName, MacroParms and Semicolon, and assemblage of MacroCall are from Fehd [5, sco.Macro-CallMacro]
- macro variable testing is explained in Fehd [8, nesug2007.cc12]
- Fehd [9, pnwsug2009.do-which] compares data and macro do iterative, until and while

DateLoop Demos

This program can be used to test macro dateloop for any previous interval, by day. The `intnx` increment value of minus one is used to provide dates of the previous interval.

dateloop-demo-prev-any.sas

```
%let today = %sysevalf('01May2013'd);

*choice: previous which?;
%let interval = year;
%let interval = quarter;
%let interval = month;
%let interval = week;

*report previous &interval by day;
%dateloop(loop_start = %sysfunc(intnx(&interval.,&today,-1,begin))
         ,loop_stop  = %sysfunc(intnx(&interval.,&today,-1,end  ))
         ,interval   = day
         );
```

The log shows the date of the previous week.

```
DATELOOP start:        Sunday, April 21, 2013
DATELOOP  stop:      Saturday, April 27, 2013
```

Previous Month by Week

These logs show the use of the week.shift-index which can be used to change the date-start of each weekly report.

interval=week

The log of this macro call shows the loop-start is 2013-Apr-01. The default shift-index is week.1: reports begin on Sunday, the first day of the week. The date range for the first week report contains Monday, April 1, 2013.

```
11 %let today = %sysevalf('01May2013'd);
12
13 %let interval = month;
14 %let date_start =%sysfunc(intnx(&interval.,&today,-1,begin));
15 %let date_stop  =%sysfunc(intnx(&interval.,&today,-1,end  ));
16
17 *report first (short) week report includes first of month;
18 %dateloop(loop_start = &date_start
19          ,loop_stop  = &date_stop
20          ,interval   = week
21          );
DATELOOP start:     Monday, April 1, 2013
DATELOOP  stop:   Tuesday, April 30, 2013
DATELOOP: begin    Sunday, March 31, 2013
DATELOOP:   end   Saturday, April 6, 2013
```

interval=week.weekday

Here the shift-index is soft-coded as the weekday of the first. The date range for the first week's report is always the 1st through the 7th.

```
29   *weekly reports begin on first: ThisDay in (1 8 15 22 29);
30   %dateloop(loop_start = &date_start
31            ,loop_stop  = &date_stop
32            ,interval   = week.%sysfunc(weekday(&date_start))
33            );
DATELOOP start:     Monday, April 1, 2013
DATELOOP  stop:   Tuesday, April 30, 2013
DATELOOP: begin     Monday, April 1, 2013
DATELOOP:   end     Sunday, April 7, 2013
```

Macro ccyy-mm

Overview

Macro DateLoop is a function: it returns no statements, only macro calls. Macro ccyy-mm is also a function: given a date it returns that date in a special format. This is a template. See program info-in-dates.sas in Fehd [7, sco.Macro-Loops-With-Dates] for other date parts that may be used to generate a list of customized date-stamped tokens.

Program ccyy-mm

This macro returns a single token containing a date-stamped data set name: libref.date-ccyy-mm.

```
%Macro ccyy_mm(date_begin =
              ,date_end   =
              ,library    = work);
%local ccyy mm;
%let ccyy = %sysfunc(year(&date_begin));
%let mm   = %sysfunc(putn(%sysfunc(month(&date_begin)),z2));
&library..date&ccyy._&mm
%mend;
```

Demo: Previous-12

This program shows a data step which reads the previous twelve months of date-stamped data set names.

```
DATA Previous_12;
do until(EndoFile);
   set %dateloop
       (loop_start = %sysfunc(intnx(month,&today,-12,sameday))
       ,loop_stop  = %sysfunc(intnx(month,&today,-1 ,end  ))
       ,interval   = month
       ,MacroName  = ccyy_mm
       ) end = EndoFile;
   output;
   end;
stop;
run;
```

log

```
DATELOOP start:      Tuesday, May 1, 2012
DATELOOP  stop:   Tuesday, April 30, 2013
MPRINT(CCYY_MM):  work.date2012_05
MPRINT(CCYY_MM):  work.date2012_06
MPRINT(CCYY_MM):  work.date2012_07
...
MPRINT(CCYY_MM):  work.date2013_02
MPRINT(CCYY_MM):  work.date2013_03
MPRINT(CCYY_MM):  work.date2013_04
42            output;
43            end;
44         stop;
45         run;
NOTE: There were 31 observations read from WORK.DATE2012_05.
NOTE: There were 30 observations read from WORK.DATE2012_06.
NOTE: There were 31 observations read from WORK.DATE2012_07.
...
NOTE: There were 31 observations read from WORK.DATE2013_02.
NOTE: There were 31 observations read from WORK.DATE2013_03.
NOTE: There were 30 observations read from WORK.DATE2013_04.
NOTE: WORK.PREVIOUS_12 has 366 observations and 5 variables.
```

Summary

Conclusion

Splitting a reporting task with dates into a reporting subroutine and a calling routine function which standardizes the macro %do loop and calculation of date intervals yields tools that are easy to test and reuse.

Further Reading

Programs :	and updates to this paper are in Fehd [7, sco.Macro-Loops-With-Dates]
Inspiration :	Rhodes [13, sugi31.015] provides a report of the previous week and all weeks preceeding using formats.
Dates and Times :	Morgan [12, SAS-Press-2006.Morgan-Guide-Dates-Times]. Finley [11, sugi25.084] shows how to use year.N for fiscal year.
Functions :	Fehd [5, sco.Macro-CallMacro] is the predecessor and model for the macro function DateLoop. Fehd [3, sco.Macro-TextLine] provides methods to justify text in titles and footnotes.
List Processing :	Fehd and Carpenter [10, sco.List-Processing-Basics] recommend separating the %do loop from the report.
Macro %Do :	These papers contain tips for modifying macro loops. Fehd [6, sco.Macro-Do-Loop] is a macro function that returns tokens in a statement; macro function DateLoop can call a macro function that does this. Fehd [9, pnwsug2009.do-which] compares data and macro do iterative, until and while. Woodruff and Dunn [14, sesug2010.ff01] compare do and %do loops.
Testing :	Fehd [8, nesug2007.cc12] explains the use of the macro variable testing.

References

[1] Wikipedia Contributors. Evaluation strategy. In *Wikipedia, The Free Encyclopedia*, 2013. URL `http://en.wikipedia.org/w/index.php?title=Evaluation_strategy&oldid=540886400`. call by reference or call by value; online; accessed 2013-Mar-01.

[2] Edsger W. Dijkstra. Why numbering should start at zero. In *Transcripts*. Univ/Texas, 1982. URL `http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html`. online; accessed 2013-Mar-01.

[3] Editor R.J. Fehd. Macro TextLine. In *sasCommunity.org*, 2008. URL `http://www.sascommunity.org/wiki/Macro_TextLine`. justify text within a title or footnote.

[4] Editor R.J. Fehd. Macro variables of date and time. In *sasCommunity.org*, 2008. URL `http://www.sascommunity.org/wiki/Macro_Variables_of_Date_and_Time`. formats and functions for date-stamping.

[5] Editor R.J. Fehd. Macro Call-Macro. In *sasCommunity.org*, 2012. URL `http://www.sascommunity.org/wiki/Macro_CallMacr`. using SCL functions to read a data set and call macros.

[6] Editor R.J. Fehd. Macro do-loop. In *sasCommunity.org*, 2012. URL `http://www.sascommunity.org/wiki/Macro_Do-Loop`. macro function to return tokens inside a statement.

[7] Editor R.J. Fehd. Macro loops with dates. In *sasCommunity.org*, 2013. URL `http://www.sascommunity.org/wiki/Macro_Loops_with_Dates`. example macros, programs and updates.

[8] Ronald J. Fehd. Writing testing-aware programs that self-report when testing options are true. In *NorthEast SAS Users Group Annual Conference Proceedings*, 2007. URL `www.nesug.org/Proceedings/nesug07/cc/cc12.pdf`. topics: functions: sysfunc, getoption; macro variable Testing.

[9] Ronald J. Fehd. Do which? loop, until or while? a review of data step and macro algorithms. In *Western Users of SAS Software Annual Conference Proceedings*, 2009. URL `http://www.lexjansen.com/pnwsug/2009/Fehd,%20Ron%20-%20Do%20Which%20Loop,%20Until%20or%20While.pdf`.

[10] Ronald J. Fehd and Art Carpenter. List processing basics: Creating and using lists of macro variables. In *sasCommunity.org*, 2009. URL `http://www.sascommunity.org/wiki/List_Processing_Basics_Creating_and_Using_Lists_of_Macro_Variables`. Hands On Workshop, 20 pp.; separate report macro from list processing macro; comparison of methods: making and iterating macro arrays, scanning macro variable, writing calls to macro variable, write to file then include, call execute and nrstr; 11 examples, bibliography.

[11] Wayne Finley. What fiscal year is this and when does it start and end? In *Proceedings of the 25th Annual SAS® Users Group International Conference*, 2000. URL `http://www2.sas.com/proceedings/sugi25/25/cc/25p084.pdf`. using interval=year.N for beginning of fiscal year.

[12] Derek Morgan. *The Essential Guide to SAS Dates and Times*. SAS Institute, 2006. ISBN 978-1-59047-884-4. URL `https://support.sas.com/pubscat/bookdetails.jsp?pc=59411`. 5 chapters: intro, displaying, converting, functions, macro variables, shifting and graphing; 172 pp., index.

[13] Dianne Louise Rhodes. Pretty dates all in a row. In *Proceedings of the 31st Annual SAS® Users Group International Conference*, 2006. URL `http://www2.sas.com/proceedings/sugi31/015-31.pdf`. topics: dates, functions: intck, intnx; intervals.

[14] Sarah Woodruff and Toby Dunn. Take control: Understanding and controlling your do-loops. In *SouthEast SAS Users Group Annual Conference Proceedings*, 2010. URL `http://analytics.ncsu.edu/sesug/2010/FF01.Woodruff.pdf`. Foundations and Fundamentals, 21 pp.; comparison of data step and macro do loops.

Closure

Acknowledgements

My thanks to SAS-L contributors Yevgeniy Bolotin and Howard Schreier for commentary and proofreading.

Contact Information:

Ronald J. Fehd `mailto:Ron.Fehd.macro.maven@gmail.com`
`http://www.sascommunity.org/wiki/Ronald_J._Fehd`

About the author:

education:	B.S. Computer Science, U/Hawaii,	1986
	SAS User Group conference attendee since	1989
	SAS-L reader	since 1994
experience:	programmer: 25+ years	
	data manager using SAS:	17+ years
	statistical software help desk:	7+ years
	author: 30+ SUG papers	
	sasCommunity.org: 300+ pages	
SAS-L:	author: 6,000+ messages to SAS-L since	1997
	Most Valuable SAS-L contributor:	2001, 2003

Trademarks