Writing Macro Do Loops with Dates from Then to When
Ronald J. Fehd, Stakana Analytics, 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. Two macros are provided: a template to modify for reports and a generic calling macro function which contains a macro %do loop which can either call another macro or return a token with elements in the associative array 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 [11, 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
Start: Introduction 2

Begin: Macros to Handle Dates 6

Macro Date-Report-Zero ........................................ 7
Macro DateLoop ................................................. 10
DateLoop Program Listing ...................................... 11
DateLoop Testing: Simple and Complex Intervals ........... 14
Parameter MacroName .......................................... 16
End: Parameter MacroText ..................................... 17

Stop: Summary 18

Bibliography 19
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 or intnx

Data Loop

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

```
1 DATA _Null_;
2   do date = 0 to 1;
3      put date= 8. date date.;
4   end;
5   put;
6   do date = '3Jan1960'd to '4Jan1960'd;
7      put date= 8. date date.;
8   end;
9   stop;
10  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(start = ’1Jan1960’d
2   ,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: &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 with either of today or mdy functions, or %sysevalf a date literal

```
%let today = %sysfunc(today());
%let today = %sysfunc(mdy(05,01,2013));
%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 [5, sco.Macro-Vars-of-Date-and-Time] contains other functions and formats used to display dates in titles and footnotes.
Associative Array of Dates

Overview

An associative array is a natural language concept. Given, for instance, an abbreviation such as a two-letter U.S. state code, we associate other information with that code, such as the long name of the state, its capital, etc. This table shows a subset of U.S. federal information processing (fips) abbreviations to illustrate the concept.

<table>
<thead>
<tr>
<th>fips</th>
<th>abbrev</th>
<th>state</th>
<th>capital</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>AK</td>
<td>Alaska</td>
<td>Juneau</td>
<td>Fairbanks</td>
</tr>
<tr>
<td>6</td>
<td>CA</td>
<td>California</td>
<td>Sacramento</td>
<td>San Francisco</td>
</tr>
<tr>
<td>13</td>
<td>GA</td>
<td>Georgia</td>
<td>Atlanta</td>
<td>Savannah</td>
</tr>
</tbody>
</table>

The fips code is an integer in the look-up table from which a program can get the two-letter state abbreviation and the long name.

do  fips = 1 to 128;
    Abbrev = fipstate(fips);
    State = fipnamel(fips);

Date Information

A SAS date is an index to the associative array of date information.

This table shows the functions and formats of the date look-up table.

<table>
<thead>
<tr>
<th>name of day</th>
<th>day of month</th>
<th>function</th>
<th>range</th>
<th>format</th>
<th>names</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>weekday</td>
<td>1–7</td>
<td>monname</td>
<td>Sun–Sat Jan–Dec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>day</td>
<td>1–31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>juldate</td>
<td>001–366</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| week        | year of month| week    | 0–53  |        |                   |
| month       | year          | month   | 1–12  |        |                   |
| quarter     | year          | qtr     | 1–4   |        |                   |

Continued on next page.
Info in Dates

This program shows how to extract each element from the associative array of dates.

```sas
%let today = %sysfunc(today());
%let today = %sysfunc(mdy(04,1,2013));
%let today = %sysevalf('30Apr2013'd);
%let day = %sysfunc(putn(&today,downame));
%let day3 = %sysfunc(putn(&today,downame3));
%let month = %sysfunc(putn(&today,monname));
%let mon = %sysfunc(putn(&today,monname3));
%let day_month_N = %sysfunc(day(&today));
%let d = %sysfunc(weekday(&today));
%let dd = %sysfunc(putn(%sysfunc(day(&today)),z2));
%let ddd = %substr(%sysfunc(juldate7(&today)),5,3);
%let month_N = %sysfunc(month(&today));
%let mm = %sysfunc(putn(%sysfunc(month(&today)),z2));
%let quarter = %sysfunc(qtr(&today));
%let year = %sysfunc(year(&today));
%let year_ddd = %sysfunc(year(&today))_%substr(%sysfunc(juldate7(&today)),5,3);
%let week = %sysfunc(week(&today,u));
%let week_v = %sysfunc(week(&today,v));
%let week_w = %sysfunc(week(&today,w));
%put today: &today %sysfunc(putn(&today,weekdate29.));
%put _global_;
```

The edited log shows the values of the macro variables of date information.

```
GLOBAL TODAY 19549
  today: 19549 Wednesday, July 10, 2013
GLOBAL DAY Wednesday
GLOBAL DAY3 Wed
GLOBAL DAY_MONTH_N 10
GLOBAL D 4
GLOBAL DD 10
GLOBAL DDD 191
GLOBAL MON Jul
GLOBAL MONTH July
GLOBAL MONTH_N 7
GLOBAL MM 07
GLOBAL QUARTER 3
GLOBAL WEEK 27
GLOBAL WEEK_V 28
GLOBAL WEEK_W 27
GLOBAL YEAR 2013
GLOBAL YEAR_DDD 2013_191
GLOBAL YEARDDD 2013191
These macro variables are calculated in macro DateLoop when using the parameter MacroText.
```
Macros to Handle Dates

Overview
The following sections show macros Date-Report-Zero and DateLoop. The macro function DateLoop can call a reporting macro like Date-Report-Zero with the parameters Begin and End. DateLoop with the parameter Macro-Text can return a token with references to the elements of the associative array of dates. The programs provided here are basic unit tests and demonstrations. For other programs see Fehd [11, sco.Macro-Loops-With-Dates].

Header
Each program has a common header.

```sas
/* name: dataloop-demo.sas 
description: testing
purpose: template /******/
options mprint;
%let today = %sysfunc(today());
*/
```

Test Data
This program is used to generate 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 today();
   EntityId = int(Date /17);
   Fact = int(Date **3/19);
   output;
   end;
   stop;
run;
```

Why Zero?

Calendar
Use this calendar when comparing dates in logs.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>day 1</td>
<td>day 2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>day 7</td>
</tr>
<tr>
<td>Sun</td>
<td>Mon</td>
<td>Tues</td>
<td>Wed</td>
<td>Thurs</td>
<td>Fri</td>
<td>Sat</td>
</tr>
<tr>
<td>Mar 31</td>
<td>Apr 1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>May 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>
This section presents a simple model — a template for a macro that can be customized — with formats and macro functions for assembling ODS filenames.

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
   (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;
```

Continued on next page.
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,
   begin = 11,
   end = 13,
   testing = 0);
ods _all_ close;
ODS PDF
  file = "demo-ods-%sysfunc(juldate7(&Begin)).pdf"
;
PROC Print data = &Data.
  (where = (&Begin <= &Var <= &End) );
title3 "data = &Data..&Var in range 
  %sysfunc(putn(&Begin,worddate18.))
  -- %sysfunc(putn(&End,mmddyy10.))"
;
run;
ods _all_ close;
ods listing;
%mend daterpt0;

Notes:
- the print procedure is used in this example; provide your own when modifying
- 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 [5] sco.Macro-Vars-of-Date-and-Time for other formats and functions; to justify text in title or footnote see Fehd [4] sco.Macro-TextLine

Continued on next page.
This program can be used to test macro \texttt{daterpt0} for any previous interval. The \texttt{intnx} increment value of minus one is used to provide dates of the previous interval.

\begin{verbatim}
*let today = %sysfunc('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
  ,begin = &begin
  ,end = &end
);
\end{verbatim}

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

\begin{verbatim}
  today: 19479 Wednesday, May 1, 2013
  begin: 19469 Sunday, April 21, 2013
  end : 19475 Saturday, April 27, 2013
\end{verbatim}

\textbf{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.

\begin{verbatim}
*report previous &interval by day;
%daterloop(start = %sysfunc(intnx(&interval.,&today,-1,begin))
  ,stop = %sysfunc(intnx(&interval.,&today,-1,end ))
  ,interval = day
);
\end{verbatim}

\textbf{Reference or Value?} Contributors [\texttt{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 tokens within a statement. Its purpose is to encapsulate a macro %do loop through a series of dates while generating calls to a macro reporting subroutine or resolving tokens within a statement.

It has parameters which support these design elements:

- %do loop
- intnx interval
- called macro, passing parameters to
- tokens: expanding references to elements of the associative array of dates
- testing

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

intnx interval:
- simple: year, quarter, month, week, day etc.;
- complex:
  - multiple: intervalN: week2 is 14 days
  - shift-index: interval.N, the default is week.1 which returns the week starting on Sunday; interval=week.2 returns the week starting on Monday

called macro:
- 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 — must have parameters named Begin and End

tokens:
- MacroText contains one or more tokens containing references to elements of the associative array of dates

testing:
- this variable is provided for self-reporting, see: Fehd [12] 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 [13] pwnsug2009.do-which compares data and macro do iterative, until and while.
/* name: <UNC>\SAS-site\macros\dateloop.sas
author: Ronald J. Fehd  2013
--------------------------------------------------
Summary : description : encapsulates macro do loop of dates
  1. calls macro with parameters
     of begin and end
  2. returns tokens with references
to associative array of dates
purpose : facilitates calling of macros
          with date intervals
--------------------------------------------------
Contexts : program group: list processing token generator
program type: function
SAS type: macro function
uses routines: macro named in parameter
--------------------------------------------------
Specifications: input : required: Start, Stop, Interval
     either of: MacroName
     or MacroText
     optional: MacroParms for MacroName
NOTE: called macro must have parameters Begin and End
process: call MacroName with start and stop
     or prepare macro vars for MacroText usage
output : calls macro &MacroName
          or expansion of &MacroText
---------------------------------------------------
Parameters 1: MacroName = name of macro to call,
              ,MacroName = put :: default, for testing
              ,MacroParms = additional parameters for called macro
              ! --> Constraint: must be enclosed in %nrstr:
              OR 2: ,MacroText = %nrstr(%nrstr(work.date&Year._&MM))
              ,Semicolon = 0 :: no semicolon after macro call
              ,Semicolon = 1 :: use when macroname is a statement
              note: reset when macroName=put
              ,Testing = 0 :: default, no extra messages in log
              ,Testing = 1 :: for testing, note: reset when
          options mprint source2;
---------------------------------------------------
usage: demo
%let today = %sysfunc(today());
%let interval = week;
%let Begin = %sysfunc(intnx(&interval,&today,-12,begin));
%let End = %sysfunc(intnx(&interval,&today,-1,end ));
* testing;
%dateLoop(start = &begin
    ,stop = &end
    ,interval = day
 )
* add small data: concatenate data sets for report;
%put
%dateLoop(start = &begin
    ,stop = &end
    ,interval = month
    ,MacroText= %nrstr(work.Year_Month_&year._&mm._&month)
);
* subtract from big data: make subset for periodic snapshot(s)
  of previous N intervals from transactions;

%let interval = month;
%let begin  = %sysfunc(intnx(&interval,&today,-1,begin));
%let end    = %sysfunc(intnx(&interval,&today,-1,end  ));
%dateloop(start = &begin
,stop    = &end
,interval = week
,MacroName = daterpt0
,MacroParms = %nrstr(data=library.testdata,var=date)
)

see also:
sas.help: Incrementing Dates and Times
  by Using Multipliers and by Shifting Intervals
  http://www.sascommunity.org/wiki/Macro_Loops_with_Dates
  http://www.sascommunity.org/wiki/D0_Which_Loop_Until_or_While
predecessors: http://www.sascommunity.org/wiki/Macro_CallMacro
  http://www.sascommunity.org/wiki/Do_Which_Loop_Until_or_While
**********/

%Macro dateloop
    (start =0 /* integer of date */
    ,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)*/
    ,MacroText =.
    ,semicolon =0
    ,testing =0
    ) /des = 'site: call macro w/dates or expand references'
    /* ** store /* ** source /* */;
%local D_Begin D_End Format MacroCall
    d dd ddd day day3 mm mon month
    ccyy qtr year yy week weekv weekw;
%let Format = weekdate29.;
%if "&MacroText" ne "." %then %let MacroName = ;
%if %scan(&MacroName,1) eq put %then %let Semicolon = 1;
%let Testing=%eval( &Testing or &Semicolon
or %sysfunc(getoption(mprint )) eq MPRINT
and %sysfunc(getoption(source2)) eq SOURCE2);
%put &SysMacroName start: %sysfunc(putn(&start,&format));
%put &SysMacroName stop: %sysfunc(putn(&stop ,&format));
%let D_Begin = %sysfunc(intnx(&interval,&start,0,begin));

Continued on next page.
%do %until(&D_Begin gt &Stop);
  %let D_End = %sysfunc(intnx(&Interval,&D_Begin,0,end));
%if &Testing %then %put &SysMacroName: begin %sysfunc(putn(&D_Begin,&format));
  %if "$MacroText" ne "." %then %do;
    %let d = %sysfunc(weekday(&D_begin));
    %let dd = %sysfunc(putn(%sysfunc(day(&D_begin)),z2));
    %let ddd = %substr(%sysfunc(juldate7(&D_begin)),5,3);
    %let day = %sysfunc(putn(&D_begin,downname));
    %let day3 = %substr(&Day,1,3);
    %let mm = %sysfunc(putn(%sysfunc(month(&D_begin)),22));
    %let mon = %sysfunc(putn(&D_begin,monname3));
    %let month = %sysfunc(putn(&D_begin,monname));
    %let qtr = %sysfunc(qtr (&D_begin));
    %let ccyy = %sysfunc(year(&D_begin));
    %let year = &ccyy;
    %let yy = %substr(&ccyy,3,2);
    %let week = %sysfunc(week(&D_begin,u));
    %let weekv = %sysfunc(week(&D_begin,v));
    %let weekw = %sysfunc(week(&D_begin,w));
    %unquote(&MacroText)
%end;
%else %do;
  %if &Testing %then %put &SysMacroName: end %sysfunc(putn(&D_End ,&format));
  %let MacroCall = &MacroName(;
  %if %length(&MacroParms) %then %let MacroCall = &MacroCall.%unquote(&MacroParms,);
  %let MacroCall = &MacroCall.begin=&D_Begin,end=&D_End);
  %put &SysMacroName calling &MacroCall;
  %&MacroCall
%end;
%if &Semicolon %then %do;
; %end;
%let D_Begin = %eval(&D_End +1);
%end;
%mend dateloop;

Notes:

- unit tests of this macro are in Fehd [11] sco.Macro-Loops-With-Dates
- parameters MacroName, MacroParms and Semicolon, and assemblage of MacroCall are from Fehd [7] sco.Macro-CallMacro
- parameter MacroText is from Fehd [8] sco.Macro-CallText
- macro variable Testing is explained in Fehd [12] nesug2007.cc12
- Fehd [13] pwsug2009.do-which compares data and macro do iterative, until and while
DateLoop Testing: Simple and Complex Intervals

Overview

The default value of the DateLoop parameter MacroName is put note:. This feature facilitates testing because the macro writes two set of notes to the log using the format weekdate29.

1. start and stop
2. date-begin and -end inside the loop

start, stop: These notes are always written to the log; they resolve the dates passed to the macro.

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

begin, end: These notes from within the loop are the values passed to the macro named in the parameter MacroName. Alternately, if the parameter MacroText is used then it is expanded from the begin date.

DATELOOP: begin Sunday, March 31, 2013
DATELOOP: end Saturday, April 6, 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 date-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.

13 %let today = %sysevalf('01May2013'd);
14 %let interval = month;
15 %let date_start = %sysfunc(intnx(&interval.,&today,-1,begin));
16 %let date_stop = %sysfunc(intnx(&interval.,&today,-1,end));
17 *report first (short) week report includes first of month;
18 %dateloop(start = &date_start,
19 stop = &date_stop,
20 interval = week)
21 DATELOOP start: Monday, April 1, 2013
22 DATELOOP stop: Tuesday, April 30, 2013
23 DATELOOP: begin Sunday, March 31, 2013
24 DATELOOP: end Saturday, April 6, 2013
25 note:(begin=19448,end=19454)
...
26 DATELOOP: begin Sunday, April 28, 2013
27 DATELOOP: end Saturday, May 4, 2013
28 note:(begin=19476,end=19482)

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(start = &date_start,
31 stop = &date_stop,
32 interval = week.%sysfunc(weekday(&date_start)));
33 DATELOOP: begin Monday, April 1, 2013
34 DATELOOP: end Sunday, April 7, 2013

Continued on next page.
This program can be used to test macro DateLoop for any previous interval, by any lesser interval. The intnx increment value of minus one is used to provide dates of the previous interval.

```
%let today = %sysfunc(today());
** choices: previous which?, by?;  
%let intervals = year-quarter;
*let intervals = year-month;
*let intervals = quarter-month;
*let intervals = quarter-week;
*let intervals = month-week;
*let intervals = month-day;
%let intervals = week-day;

%let int_main = %scan(&intervals,1,-);
%let int_by = %scan(&intervals,2,-);
** report previous &int_main by &int_by;
%dataloop(start = %sysfunc(intnx(&int_main.,&today,-1,begin))
 ,stop = %sysfunc(intnx(&int_main.,&today,-1,end ))
 ,interval = &int_by
 );
```

The log shows the date of the previous week.

```
DATELOOP start: Sunday, June 30, 2013
DATELOOP stop: Saturday, July 6, 2013
DATELOOP: begin Sunday, June 30, 2013
DATELOOP: end Sunday, June 30, 2013
  note:(begin=19539,end=19539)
DATELOOP: begin Monday, July 1, 2013
DATELOOP: end Monday, July 1, 2013
  note:(begin=19540,end=19540)
...  
DATELOOP: begin Saturday, July 6, 2013
DATELOOP: end Saturday, July 6, 2013
  note:(begin=19545,end=19545)
```
Parameter MacroName

Overview

Macro DateLoop has a parameter MacroName and optional parameter MacroParms which are used to call a reporting macro.

! → The reporting macro must have parameters begin and end which are passed in the macro call.

This feature is used to subtract a subset from a larger data set. In database terminology the report produced is called a periodic snapshot; see also Fehd [3, sco.DatabaseVocabulary].

Demo: Weekly Reports

Here is an example program.

```sas
*weekly reports begin on first: ThisDay in (1 8 15 22 29);
%dateloop(start=&begin,
  stop=&end,
  interval=week.%sysfunc(weekday(&begin))
  ,MacroName=daterpt0
  ,MacroParms=%nrstr(data=library.testdata,var=date)
  ,testing=1);
```

and the log:

```
DATELOOP start: Monday, April 1, 2013
DATELOOP begin Monday, April 1, 2013
DATELOOP: begin Monday, April 1, 2013
DATELOOP calling daterpt0(data=library.testdata
  ,var=date,begin=19449,end=19455)
```

```
... ...
DATELOOP calling daterpt0(data=library.testdata
  ,var=date,begin=19456,end=19462)
... ...
DATELOOP calling daterpt0(data=library.testdata
  ,var=date,begin=19477,end=19483)
```
Macro DateLoop is a function: it returns no statements, only tokens, which may be either macro calls, or text with references to elements of the associative array of dates, which are provided in the parameter MacroText.

This feature is used to add or concatenate a number of date-stamped data sets into a super-set.

### Demo: Previous 7

This macro call returns a series of tokens containing a date-stamped data set name: libref.date-ccyy-mm.

This example shows how to check the resolution by using

```
MacroText= %nrstr(%put ...).
```

! → Remember to add `semicolon=1` with this feature.

```
%dateloop(start = %sysfunc(intnx(&interval,&today,-7,sameday)),
stop = %sysfunc(intnx(&interval,&today, 0,end )),
interval = day,
MacroText = %nrstr(&Library..date&ccYY._&MM._&DD),
semicolon=1)
```

The log shows the set of tokens returned by DateLoop.

| DATELOOP start: | Wednesday, July 3, 2013 |
| DATELOOP stop:  | Wednesday, July 10, 2013 |
| work.date2013_07_03 |
| work.date2013_07_04 |
| ... |
| work.date2013_07_09 |
| work.date2013_07_10 |

### Demo: Previous 12

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

```
%let Library = work;
DATA Previous_12;
do until(EndoFile);
  set %dateloop
    (start = %sysfunc(intnx(month,&today,-12,sameday)),
     stop = %sysfunc(intnx(month,&today,-1,end ) ),
     interval = month,
     MacroText = %nrstr(&Library..date&ccYY._&MM),
     end = EndoFile;
  output;
end;
stop;
run;
```

The log shows the set of tokens returned by DateLoop.

| DATELOOP start: | Tuesday, July 10, 2012 |
| DATELOOP stop:  | Sunday, June 30, 2013  |
| ... |
| NOTE: There were 31 observations read from WORK.DATE2012_07. |
| NOTE: There were 31 observations read from WORK.DATE2012_08. |
| ... |
| NOTE: There were 31 observations read from WORK.DATE2013_05. |
| NOTE: There were 31 observations read from WORK.DATE2013_06. |
| NOTE: WORK.PREVIOUS_12 has 366 observations and 5 variables. |
Summary

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.

Conclusion

Further Reading

Programs: and updates to this paper are in Fehd [11] sco.Macro-Loops-WithOptions.

Inspiration: Rhodes [17] sugi31.015 provides a report of the previous week and all weeks preceding using formats.


Fehd [6] sco.SmryEachVar is a list processing suite using parameterized %includes.

Fehd [10] sco.Routine-CxMacro routine CxMacro is the predecessor of Macro CallMacro.

Macro %Do: These papers contain tips for modifying macro loops. Fehd [9] 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 [13] pnwsug2009.do-which compares data and macro do iterative, until and while. Woodruff and Dunn [19] sesug2010.ff01 compare do and %do loops.

Bibliography


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

Trademarks
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries ® indicates USA registration.

Close Quote
A SAS date is an index to the associative array of date information.

— R. J. Fehd