Paper 1645-2014

Speed Dating: Looping Through a Table Using Dates

Scott Fawver, Arch Mortgage Insurance Company, Walnut Creek, CA

ABSTRACT

Have you ever needed to use dates as values to loop through a table? For example, how many events occurred by 1, 2 , 3 … n months ahead. Maybe you just changed the dates manually and re-ran the query n times? This is a common need in economic and behavioral sciences. This presentation demonstrates how to create a table of dates that can be used with SAS® macro variables to loop through a table. Using this dates table in combination with the SAS® do loop ensures accuracy and saves time.

INTRODUCTION

In economic and behavioral sciences the event(s) of interest often occurs over a series of time. For example, a 30 year mortgage loan takes 360 monthly payments to payoff. A loan can be in several different statues over the term, i.e. performing, delinquent, etc. A loan has both static (FICO and Loan-to-Value at origination) and dynamic (loan age at delinquency and payments missed) attributes. Similarly, in experimental design, a zebra fish can be in various states of health over its life and has both deterministic (sex and treatment) and stochastic (survival age and body temperature) properties.

Creating a historical time series of the life of a loan or zebra fish is often the first step to modeling the event of interest. The training data is created by combining both the static and dynamic components. The dependent variable is added to the data set as a 0 for non-event and 1 for event. Logistic regression is then performed on the dataset to determine the relevant characteristic to predict the event in the future.

EXAMPLE

SAS® offers a very powerful mechanism for creating time series data with the DO OUTPUT statement. The following code creates a table of dates, where the researcher has set the start month to Jan 1990 and the end month to Jul 2013. The start and end months are placed into macro variables using %LET. Next the number of months between the start and end months is calculated and placed into a macro variable using the INTCK function. Finally, the dates table is expanded one month at a time creating every first and last date of the month between Jan 1990 and Jul 2013, i.e. 1/1/1990, 1/31/1990… 7/1/2013, 7/31/2013. SAS® stores dates as numerical values, where 1/1/1960 midnight equals zero. Dates are incremented by 1 at midnight for every day after 1/1/1960. Dates prior to 1/1/1960 are negative. A FORMAT statement is included so the dates are readable as text rather than numbers.

```
%let start_date = 01jan1990;
%let end_date = 01jul2013;
%let months = intck('month',"&start_date."d,"&end_date."d);

data dates;

do m = 0 to &months.;

       first_of_month = intnx('month',"&start_date."d,m,'s');
       last_of_month = intnx('month',"&start_date."d,m,'e');
       lead_first_of_month = intnx('month',first_of_month,1,'s');
       lead_last_of_month = intnx('month',last_of_month,1,'e');
       as_of_year_month = (year(first_of_month)*100)+month(first_of_month);

       output;

end;

format first_of_month last_of_month lead_first_of_month lead_last_of_month date9.;

run;
```

The first three records of the dates table are shown below. Notice in addition to the actual dates, the code has created one additional field: as_of_year_month. Often original data snapshots are created at the end of the month and named in a common way. Typically the as_of_year_month is added to the end of the common file name. Having as_of_year_month can be extremely useful as these can be easily referenced via a macro variable in a loop.

m	first_of_month	last_of_month	lead1_first_of_month	lead1_last_of_month	as_of_year_month
0	01Jan1990	31Jan1990	01Feb1990	28Feb1990	199001
1	01Feb1990	28Feb1990	01Mar1990	31Mar1990	199002
2	01Mar1990	31Mar1990	01Apr1990	30Apr1990	199003

The dates table can now be sourced one month at a time incrementally inside a SAS® do loop. As the loop iterates, dates are placed into macro variables and used to query single or multiple table(s) containing the information of interest. The final time series is created by appending in records one run of the loop at a time.

Shown below is where the dates are placed into macro variables. Now these dates can be referenced in the data query.

```
%macro perf_to_dq(i,ct);

%do i = &i. %to &ct.;

proc sql;
select first_of_month
into :first_of_month
from dates
where m=&i.;

select last_of_month
into :last_of_month
from dates
where m=&i.;

select lead1_first_of_month
into :lead1_first_of_month
from dates
where m=&i.;

select lead1_last_of_month
into :lead1_last_of_month
from dates
where m=&i.;

select as_of_year_month
into :as_of_year_month
from dates
where m=&i.;

quit;
```

Shown here are the macro variables for i = 0.

m	first_of_month	last_of_month	lead1_first_of_month	lead1_last_of_month	as_of_year_month
0	01Jan1990	31Jan1990	01Feb1990	28Feb1990	199001

In the code that follows, a modeling dataset of loans transitioning from performing to delinquent is created. To keep it simple, assume loans can either be performing or delinquent. A table called delinquency contains all the delinquent records. A table called loans contains a loan's book date (the date the loan was originated on).

The first query determines loans delinquent in the current period (January 1990).

```
proc sql;
create table dq as
select distinct(loanid)
from delinquency a
where period >= "&first_of_month."d and period <= "&last_of_month."d
group by loanid;
quit;
```

Next, the loans not delinquent in the current period (performing) with a book date (origination date) less than February 1, 1990 are queried.

```
proc sql;
create table perf as
select &as_of_year_month. as as_of_year_month, a.loanid, book_date
from loans a
left outer join dq b on a.loanid=b.loanid
where b.loanid is null and book_date < "&lead1_first_of_month."d;
quit;
```

Now a list of any loans performing in the current period that became delinquent one month later is created.

```
proc sql;
create table dq as
select distinct(a.loanid)
from delinquency a
inner join perf b on a.loanid=b.loanid
where period > "&last_of_month."d and period <= "&lead1_last_of_month."d
group by a.loanid;
quit;
```

The one month ahead dependent variable is generated (did the loan go delinquent?) and the loan age is calculated.

```
proc sql;
create table perf_to_dq as
select a.as_of_year_month, a.loanid,
intck('month',book_date,"&lead1_last_of_month."d) as loan_age,
case when b.loanid is not null then 1 else 0 end as dqflag
from perf a
left outer join dq b on a.loanid=b.loanid;
quit;
```

The final table is created and the macro is closed.

```
    %if &i. = 0 %then %do;

    data perf_to_dq_final; set perf_to_dq; run;

    %end;

    %if &i. > 0 %then %do;

    proc sql; insert into perf_to_dq_final select * from perf_to_dq; quit;
    %end;

    %end;

    %mend perf_to_dq;
```

A macro variable is created to determine how many months to run the loop.

```
    proc sql;
    select count(*)-1 as n
    into :n
    from dates;
    quit;
```

The macro is run.

```
    %perf_to_dq(0,&n.);
```

The modeling dataset is generated.

as_of_year_month	loanid	loan_age	dqflag
200012	1	1	0
200101	1	2	0
200102	1	3	0
200103	1	4	0
200104	1	5	0
200105	1	6	0
200106	1	7	1
200501	2	1	0
200502	2	2	0
200503	2	3	1

CONCLUSION

Using a list of dates to loop through tables is an easy, efficient and accurate way to create a time series of a given behavior of interest. Using the SAS® DO OUTPUT statement a table of dates can be created. The table of dates is referenced inside a loop where the dates are placed into macro variables using the loop counter as a filter. The dates can then be sourced to filter queries based on the event of interest.

Code demonstrating this presentation can be found at this link.

http://www.sascommunity.org/wiki/File:Paper_1645_2014.sas

ACKNOWLEDGMENTS

I would like to thank Barbara Okerson for her feedback in putting together this presentation. I would highly recommend the SAS® Global Forum Presenter Mentoring Program for anyone thinking about presenting a paper.

CONTACT INFORMATION

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

Scott Fawver, Econometrician
Arch Mortgage Insurance Company
3003 Oak Road
Walnut Creek, CA 94597
925.658.6675
sfawver@archmi.com