Rolling Regressions in SAS: Equations with Rolling Sample Periods

Many data explorations and research projects estimate the same basic equation or relationship among variables over multiple date ranges. Rolling regressions—where a least-squares equation is estimated multiple times using partially overlapping subsamples from a larger set—is an example that belongs to this category. Typically, rolling regressions are applied to time series data in a manner that keeps the sample length fixed for each estimation step by increasing the beginning and ending dates by a particular time or date increment. Other techniques such as recursive least squares, keep the starting date fixed. In both cases, overlapping sample periods are used to estimate multiple versions of the same regression coefficients.

While SAS does not have a simple option or statement that can added to PROC REG or any of its other model estimation procedure to run rolling regressions, a macro subroutine can be created to achieve the desired effect. For example, see the sample program ROLLINGBETA1.SAS (https://wrds-web.wharton.upenn.edu/wrds/support/code_show.cfm?path=CRSP/rollingbeta1.sas) that computes CAPM betas on a per stock basis in an iterative, looping fashion from daily CRSP stock file data. This program makes use of SAS's macro facilities to define a 24-month estimation 'window' that increments forward one-month for each iteration of the loop until it reaches the end of a full sample period. The ROLLINGBETA1.SAS program also saves the results from each window in a permanent dataset of estimated coefficients and can be modified to change the estimated equation and the overall sample period. It is also possible to change the length of estimation window. In sum, this program provides a useful starting point for applying rolling regression techniques in SAS.

Below a more general date-based looping technique is shown and presented in the form of a SAS macro. This macro or subroutine makes use of date formats and functions to define the estimation window period and to allow for different types of overlapping date ranges.

Using Date Loops and PROC REG with a BY Statement

Data format assumptions.
In this example, it is assumed that the application will have both cross-sectional and time-series aspects. In particular, it is assumed that the dataset is structured in a manner that allows a 'BY' variable in PROC REG to estimate OLS coefficients by company or stock.

For example, a dataset named DSET1 might have the following structure of rows and columns: ID, DATE Y, X1, X2. In this case, the observations have an explicit cross-sectional identifier (ID), which might be PERMNO for CRSP data, GVKEY for Compustat data, or Ticker Symbol in a generic dataset. The time-series aspect is the DATE variable that could be daily, monthly, quarterly or annual in frequency. And in this example, Y is the dependent variable and X1 and X2 are the potential explanatory variables. Most important, it is assumed that the data is sorted or can be sorted by ID and DATE, such that within each ID block, the observations are in DATE order.

The ID dimension in the assumed data format makes SAS a good choice for these types of estimation problems. As long as the data is sorted in ID and DATE order, the PROC REG can be used to estimate coefficients for each ID.

```
proc reg noprint data =dset1 outest =regout1 edf ;
  where date between '01JAN2001'd and '31DEC2002'd;
  model y = x1 x2;
  by id;
run ;
```

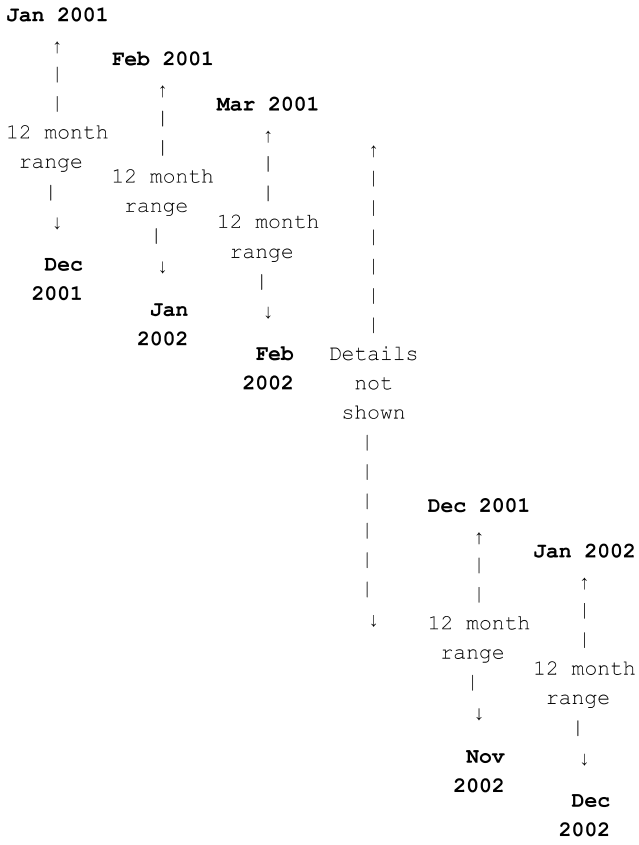
The code above places the OLS coefficients, R2, and a few other statistics and identifiers for a two-year period in an output dataset (OUTEST=REGOUT1 in this case). Suppose DSET1 has monthly data for 5 ID's, then the results might be as so

ID	_DEPVAR_	_RMSE_	Intercept	x1	x2	_P_	_EDF_	_RSQ_
10107	y	0.008	-0.008	0.895	1.327	3	249	0.417
11081	y	0.011	-0.008	0.935	1.269	3	249	0.324
12490	y	0.007	-0.007	0.809	2.264	3	249	0.409
14593	y	0.020	-0.014	1.366	0.776	3	249	0.387
14656	y	0.016	0.006	0.828	-0.313	3	249	0.128

The basic task of the rolling regression macro is to change the date range to cover different periods in an iterative or 'looping' fashion and to append the results from each loop in a single dataset that holds the OLS statistics for each identifier and date range.

The table below helps to conceptualize the 'loops' for a case that uses data from January 2001 to December 2003, with each loop's sample period covering 12 months and the sample periods moving ahead 1 month per loop, such that 13 loops are run.

Loop 1	Loop 2	Loop 3	Loops 4 to 11	Loop 12	Loop 13



It is not necessary for a regression to be run that uses monthly data in the loop design above, as daily data could be used with the first date in each loop set at the first day in each month and the last date in each loop set at the last day in each month. Most important, it is often desirable to use more than a one-month gap between the end date of each loop. Therefore, the macro code for the rolling regression routine below allows for many more types of loops than ones that use 12 month periods and it can iterate forward almost any number of days, months or years at each step.

The macro is set up to accept input arguments that include the input and output data set names, the regression model equation specification, and the identifier variables. For different types of loops and thus, different types of rolling regressions, the macro makes use of an inter-related set of date and frequency parameters. The date settings include the identifier for the main date variable in the data set and also parameters that define the start and end dates for the entire analysis and for each loop. It is possible to set the frequency of the date periods that define each loop to be greater than the frequency of the data. In other words, loops over daily do not need to iterate forward one day at each step, and either monthly, quarterly or annual loop intervals are easy to specify.

For the macro routine's arguments, *data* , *out_ds, model_equation* , and *id* are fairly self-explanatory and only the first three are actually required. In defining this macro, a 'named' argument style is used, as opposed to positional style that has a fixed set of inputs, so that some parameters can take default values and the order of the inputs is not important. Here, he *data* and *out* designations will typically be two-level names such as *data=mylib.md* and *out_ds=mylib.mregout*. The *id* argument might be populated with PERMNO, GVKEY, TICKER, or another company, security, industry or person identifier. If this argument is excluded when invoking the macro, it will be assumed that all dates are grouped together (i.e, just a time-series dimension will be used). To specify the model, an example might be *model_equation= ret = market leverage* , where two equal signs are intentional, with one separating the argument name from the equation and the other equal sign separating the left-hand side from the right-hand side. An equation such as above will assume that an intercept is desired. To exclude the intercept add */ noint* . In fact, any addition options that are valid in model statement for PROC REG can be used.

For the dates and loop dating, *date* is the default or assumed date identifier in the input set. This can be changed by simply identifying the date variable name, e.g., *date=xdate*. If *date=year* or starts with *year* (such as *date=yeara)* , then the variables values must be a 4 digit year. In all other cases, the date variable must be a SAS date, which is a numeric system that is oriented around January 1, 1960 as day 0. These SAS dates are most useful because they allow for different frequency intervals of a day, week, month, quarter or a year to define the loops such that each regression period will be based on date range endpoints that are *S* periods apart and with sample periods that are *N* periods long.

It is important to know that *start_date* and *end_date* specify the date range for the entire analysis, such that *start_date* is the first observation used in the date range of the first loop and *end_date* is the last observation used in the last loop. If these two date parameters are not set, the macro will use the entire date range in the data set (technically, the first or minimum and last or maximum date numbers will be determined and used). Valid formats for the date parameters are 01JAN2004, 1-1-20004, 1/1//2004, JAN2004, and 2004. The macro code will convert any of these formats to the proper date number such that a 4 digit year becomes the number for January 1 of the year when used with *start_date* and becomes the number for December 31 of the year when used with *end_date* . Similarly, a month-year arguments become the date number for the first and last day of the month for *start_date* and *end_date* , respectively.

The parameters *N* and *S* and *freq* define the interval between the end of each sample period and the length of each sample period. The default looping frequency is months (i.e., monthly) such that all date counting to define the loops is based on months, even if the underlying data in the analysis is daily. A setting of *freq = daily* is permitted, as is *freq=year* and *freq=quarter*. The default values of *N=1* and *S=12* will set each loop period as 12 months (same as shown in the table above) and will iterate forward one month at a time. To iterate forward one year at a time and to use 24 months as each loop sample length, you can use *freq= month, N=12,* and *S=24* or equivalently *freq= year, N=1,* and *S=2.*

The RRLOOP macro code can be copy and pasted into a SAS program and invoked as below.

```
%macro RRLOOP (year1= 2001, year2= 2005,  nyear= 2, in_ds=temp1, out_ds=work.out_ds);

%local date1 date2 date1f date2f yy mm;

/*Extra step to be sure to start with clean, null datasets for appending*/
proc datasets nolist lib=work;
  delete all_ds oreg_ds1;
run;

/*Loop for years and months*/
 %do yy = &year1 %to &year2;
   %do mm = 1 %to 12;

 /*Set date2 for mm-yy end point and date1 as 24 months prior*/
 %let xmonths= %eval(12 * &nyear); *Sample period length in months;
 %let date2=%sysfunc(mdy(&mm,1,&yy));
 %let date2= %sysfunc (intnx(month, &date2, 0,end)); *Make the DATE2 last day of the month;
 %let date1 = %sysfunc (intnx(month, &date2, -&xmonths+1, begin)); *set DATE1 as first (begin) day;
 /*FYI --- INTNX quirk in SYSFUNC:  do not use quotes with 'month' 'end' and 'begin'*/

/*An extra step to be sure the loop starts with a clean (empty) dataset for combining results*/
proc datasets nolist lib=work;
  delete oreg_ds1;
run;

/*Regression model estimation -- creates output set with coefficient estimates*/
proc reg noprint data=&in_ds outest=oreg_ds1 edf;
  where date between &date1 and &date2;  *Restricted to DATE1- DATE2 data range in the loop;
  model retrf = vwretdrf;
  by permno;
run;

/*Store DATE1 and DATE2 as dataset variables
and rename regression coefficients as ALPHA and BETA;*/
data oreg_ds1;
  set oreg_ds1;
  date1=&date1;
  date2=&date2;
  rename intercept=alpha  vwretdrf=beta;
  nobs= _p_ + _edf_;
  format date1 date2 yymmdd10.;
run;

/*Append loop results to dataset with all date1-date2 observations*/
proc datasets lib=work;
  append base=all_ds data=oreg_ds1;
run;

 %end;  % /*MM month loop*/

 %end; % /*YY year loop*/

/*Save results in final dataset*/
data &out_ds;
  set all_ds;
run;

%mend RRLOOP;
```

The primary output of this program is a data set with these items:

CRSP Permanent Number=10107

date1	date2	_RMSE_	Intercept	VWRETD	regobs
01JAN2002	31DEC2004	0.057470	-.005836843	0.97255	36
01FEB2002	31JAN2005	0.057446	-.004874504	0.95958	36
01MAR2002	28FEB2005	0.057358	-.004550887	0.92014	36
01APR2002	31MAR2005	0.057448	-.005048308	0.92846	36
01MAY2002	30APR2005	0.056784	-.000042066	0.81758	36
01JUN2002	31MAY2005	0.056721	0.000278585	0.80891	36
01JUL2002	30JUN2005	0.051923	-.006355348	0.99118	36
01AUG2002	31JUL2005	0.051481	-.004585000	0.91877	36
01SEP2002	31AUG2005	0.053227	-.002505193	0.89010	36
01OCT2002	30SEP2005	0.054340	-.003183394	0.84427	36
01NOV2002	31OCT2005	0.045522	-.003110266	0.52825	36
01DEC2002	30NOV2005	0.045962	-.002508023	0.50723	36
01JAN2003	31DEC2005	0.044918	0.000198639	0.35030	36

CRSP Permanent Number=11081

date1	date2	_RMSE_	Intercept	VWRETD	regobs
01JAN2002	31DEC2004	0.055252	0.008974707	0.94293	36
01FEB2002	31JAN2005	0.055182	0.008669580	0.94526	36
01MAR2002	28FEB2005	0.054267	0.009601962	0.88876	36
01APR2002	31MAR2005	0.054588	0.008332045	0.89608	36
01MAY2002	30APR2005	0.055504	0.004430406	0.97562	36

In this output, 'date1' and 'date2' show the date range for the sample that corresponds to the estimates in each row. A count of regression observations (regobs) is also included. An additional check to determine that the loops are specified correctly can be made by examining the output of '%put; statements that show a count of each loop as so

Loop: 1 -- 01JAN2002 31DEC2004

Finally, a default setting in the macro supresses the printout of each regression equation in the '.lst' output file, and this output be shown by using regprint=yes as input argument.

ID	_DEPVAR_	_RMSE_	Intercept	x1	x2	_P_	_EDF_	_RSQ_
10107	y	0.008	-0.008	0.895	1.327	3	249	0.417
11081	y	0.011	-0.008	0.935	1.269	3	249	0.324
12490	y	0.007	-0.007	0.809	2.264	3	249	0.409
14593	y	0.020	-0.014	1.366	0.776	3	249	0.387
14656	y	0.016	0.006	0.828	-0.313	3	249	0.128

The basic task of the rolling regression macro is to change the date range to cover different periods in an iterative or ‘looping' fashion and to append the results from each loop in a single dataset that holds the OLS statistics for each identifier and date range.

The table below helps to conceptualize the ‘loops' for a case that uses data from January 2001 to December 2003, with each loop's sample period covering 12 months and the sample periods moving ahead 1 month per loop, such that 13 loops are run.