ROLLING WINDOW ANALYSIS OF TIME SERIES USING SAS

Abstract. When analyzing financial time series data using a statistical model, it is critical to evaluate the underlying stationarity assumption of models because the economic environment often changes considerably, and above assumption may no long hold. Rolling analysis is a commonly used technique.

In rolling analysis, parameter estimates, either descriptive statistics for a given set of data or coefficients estimates from models, are computed over a rolling window of a certain size through the sample. These rolling estimates can be used to evaluate stability and predictive accuracy of the underlying data and hence the model built to describe the data. This paper will systematically review rolling analysis with the following scenairos:

- Descriptive statistics for rolling window of fixed length;
- Descriptive statistics for rolling window of variable length;
- Regression for rolling window of fixed length;

Several SAS procedures are highlighted in the demostration, such as PROC EXPAND, PROC MEANS, PROC FORMAT, PROC REG and the new PROC FCMP.

1. Introduction

2. Descriptive Statistics for rolling window of fixed length

To obtain descriptive statistics over a rolling window is not uncommon in SAS community. There have been a lot of discussions over SAS List Archive news group over the past several years. Vora [1] also wrote a nice introduction paper on how to obtain descriptive statistics using PROC EXPAND when the statistic is a function of one random variable or two random variable. The write up actually is pretty comprehensive and I will simply re-iterate the original ideas therein for completion purpose.

PROC EXPAND is a power procedure that can output a variety of descriptive statistics of a given random variable with a large set of parameter settings. The standard synatx to obtain a rolling window descriptive statistic for random variables X1 and X2 is as following:

```
proc expand data=   out= ;
     convert X1 = newName1 X2=newName2 /
             transformout=(&Stat  &windowlength);
run;
```

In above statement, we need to pay attention to the following four key elements.

- CONVERT statement tells SAS what variables are to be process. In above example, the variable *X1* will be processed and the new variable to be output is named as *newName1* and *X2* will be transformed and output as *newName2*.
- TRANSFORMOUT statement specifies a list of transformations to be applied to the output series. TRANSFORMOUT encompasses a large set of operations that can be applied, but for rolling window analysis, only a subset of descriptive statistical operations are applicable.

- *&Stat* indicates what descriptive statistic to be obtained. For a rolling window analysis, PROC EXPAND supports the a wide range of operations for descriptive statistics, such as MAX, MIN, AVE, CSS, GMEAN, MED, PROD, RNAK, RANGE, STD, USS, TVALUE, VAR, USS. These key words are self illustrative. Denote any of these statistic as &TYPE, then the &Stat can be specified in either of two ways:
  - CMOV&TYPE, which calculates centered moving window metrics for descriptive statistis of &TYPE;
  - MOV&TYPE, which calculates backward moving window metrics for descriptive statistic of &TYPE;
- *&windowlength* indicates the size of the rolling window. Typically, this is an integer of default value 1 specifying window length. For some &Stat, it can a list of weights in parenthese that to be applied in the calculation.

PROC EXPAND is not able to calcualte descriptive statistic that is a function of two random variables, such as Pearson Correlation. But these descriptive statistics are usually a function of the descriptive statistic of a single variable and hence a simple data step to calculate the desired descriptive statistics. Vora used the Pearson Correlation to demonstrate such case. For detailed examples, please consult with the paper of Vora [1];

3. Descriptive Statistics for rolling window of variable length

While rolling window descriptive statisitcs are relatively easy to obtain using PROC EXPAND, this technique assumes fixed length of rolling window, be it 30 observations or 400 observations. On the other hand, when analyzing time series data, the rolling window often needs to be defined based on the value of the time variable, say every 24 hours, or from date 1 to date 2. Thus the number of observations in a rolling window is uncertain or unknow *a prior*. Consider the following example. Data set *TradeDate* lists the dates a stock can be traded and the data set *Raw* lists for each company, the start date and end of holding period with purchase price.

```
data TradeDate;
input TradeDate yymmdd10.;
format TradeDate yymmdd10.;
cards;
2007-01-04
2007-01-05
2007-01-08
2007-01-09
2007-01-10
2007-01-11
2007-01-12
2007-01-15
2007-01-16
2007-01-17
2007-01-18
2007-01-19
2007-01-22
2007-01-23
```

```
2007-01-24
2007-01-25
2007-01-26
2007-01-29
2007-01-30
2007-01-31
 ;
run;
data raw;
input Stock_ID $ Date_S yymmdd10. +1 Date_E yymmdd10. Buy;
format Date_S Date_E yymmdd10.;
cards;
A001 2007-01-09 2007-01-24 24.5
A001 2007-01-12 2007-01-16 56.6
 ;
run;
```

The business analyst wants to calculate the number of trading days the stocks are held and a desired output is as following:

```
Stock_ID      Date_S         Date_E          Buy    Hold_Days
A001        2007-01-09    2007-01-24         24.5     12
A001        2007-01-12    2007-01-30         56.6      3
```

For the first record in data set *Raw*, the number of hold days is 12 since from Jan 09, 2007 to Jan 24, 2009, there were 12 trading days; for the second record, there were 3 trading days from Jan 12, 2007 to Jan 30, 2007.

For this type of rollowing window analysis, PROC EXPAND can barely help since the length of rolling window is irregular and undefined *a prior*. In order to handle this type of problem, we turn to a usually ignored feature in SAS, the MultiLabel format which PROC MEANS supports. To obtain the desired output above, we can simply submit the following SAS code:

```
/* 1. define multi-label format, notice the value for HLO variable */
data fmt;
     set raw;
     retain fmtname 'tdate' type 'n'  hlo 'M';
     start=Date_S; end=Date_e;
     label=cats(ID, _n_);
run;
proc format cntlin=fmt  out=fmt_ref;
run;

/* 2. invoke Multi-Label in PROC MEANS using MLF PRELOADFMT option */
proc means data=tradeDate noprint  nway;
     class TradeDate/mlf exclusive preloadfmt ;
     format TradeDate tdate.;
     var TradeDate;
     output  out=_test  n()=_freq_;
run;
```

In the above code, a multi-label format is first defined by specifying $HLO = "'M'$ in the format definition data set. Then, in PROC MEANS, specifying *TradeDate* as CLASS variable with three option items: MLF EXCLUSIVE PRELODFMT.

MLF tells PROC MEANS that format *tdate.* specified in FORMAT statement is a multi-label format enables PROC MEANS to use the primary and secondary format labels for a given range or overlapping ranges to create subgroup combinations. EXCLUSIVE option tells PROC MEANS to exclude from the analysis all combinations of the class variables that are defined in the format but not found in the data. PRELOADFMT specifies all formats are preloaded for the class variable and it has to be used together with EXCLUSIVE or COMPLETETYPES option. In PROC MEANS we conduct simple frequency count to find the number of trading days within the specified date range from data set *Raw*.

This powerful technique actually can handle a variety of scenarios. Consider a new case where the mean value needs to be calculated for a list of rolling window with size of 5, 6, $\cdots$, 15. The following code solves this puzzle.

```
/* 1. define multi-label */
data fmt;
      retain fmtname 'rollwindow'  type 'n'  hlo 'M';
      do start=1 to 10;
         end=start+5;
        label=cats('time', start);
        output;
      end;
      hlo='O'; label='Out-Of-Bound';
      output;
run;
proc format cntlin=fmt; run;

/* 2. generate simulated data with time spans from 1 to 20, with fractions */
data dsn;
      do time=1 to 20;
         k=ranpoi(10, 10);
         do j=1 to k;
            time=time+j/(k+1);
            x=rannor(0); y=ranuni(0);
            output;
         end;
      end;
      drop k j;
run;
/* 3. Obtain standard deviations over rolling window of variable length */
proc means data=dsn  noprint;
      class time/preloadfmt mlf  exclusive;
      format time rollwindow.;
      var x y;
      output  out=summary_roll  mean()=  std()=/autoname;
run;
```

In this example, the simulated data has a time value spanning from 1 to 20 with fractions, and it is desired to calculate standard deviation of two variables, X and Y, over a list of time window with length 5, 6, $\cdots$, 14, 15. Note that due to fractional time value, the number of records for any given time range is variable. The powerful MultiLabel format handles this type of variable length rolling window problem effortlessly. It is able to handle problems such as shrinking length time window, growing length time window rolling calculation.

Just like in section 1, if the descriptive statistic is a function of two variables, we can calculate each elements in the function separately and then post-process in a data step.

While powerful, this technique is not universal. It is subject to two disadvantages. First, MultiLabel format only supports up to 255 distinct subgroups, which limits the functionality. Second, MultiLabel format processing has larger computational overhead and requires more memory resources, especially when the data set is large. To bypass this problem in such circumstance, simply sort and divide the original data into smaller pieces with an overlap equals to the size of rolling window. When combine the summarized pieces, the overlap part should be discarded from the rest pieces.

4. Rolling window regression of fixed length

Rolling window regression using SAS is a much discussed topic within SAS community, see [2], [3], [4], [5], [6], [7], [8], for a few examples and it has important academic and practical role in analysis of financial market data. A rolling analysis of a time series model is often used to assess the model's stability over time. When analyzing financial time series data using a statistical model, a key assumption is that the parameters of the model are constant over time. However, the economic environment often changes considerably, and it may not be reasonable to assume that a model's parameters are constant. A common technique to assess the constancy of a model's parameters is to compute parameter estimates over a rolling window of a fixed size through the sample. If the parameters are truly constant over the entire sample, then the estimates over the rolling windows should not be too different. If the parameters change at some point during the sample, then the rolling estimates should capture this instability.

Financial time series data is large and we need efficient solution. There are a couple methods proposed in SAS community for such problem:

- Using SAS/Macro to loop through desired windows;
- Reshape the data and leverage the BY-processing capability of PROC REG;
- Use PROC EXPAND to calculate elements in SSCP matrix and then use PROC REG to process it;

Here I propsed a new method which beats all above ones in speed which directly implements the SWEEP operator in [12].

The solution based on SAS MACRO is very easy to code, but it is extremely inefficient and can't handle large dataset in reality, see [9]. This method is shown below as Method[4]. In our comparison, it couldn't finish the test in allowable time using the sample data below.

The other commonly accepted solution is to use the BY-processing capability of PROC REG after re-shaping the data into appropriate format, see [10], [11]. This method is demonstrated as Method[3] below. For details, refer to [10] for an excellent illustration. On the good side, this method can leverage the full analytical power of PROC REG; On the down side, while certainly much better than Method[4], it is still not the fastest and requires non-trivial memory space.

Another version under the same analytical framework will be demonstrated as Method[2], where the SSCP matrix is explicitly built using PROC EXPAND and DATA Step. This technique was also mentioned by David L. Cassell[14] and Liang Xie [15]. Method[2] recognizes that in OLS, all you need is the SSCP and this solution comes to play by realizing that instead of re-orgnize the data to obtain SSCP within PROC REG over a rolling window, we can easily build up each element of the rolling SSCP by resorting to PROC EXPAND and then reshape the data into SSCP special data set. However, this solution is mostly for demonstration because in reality, there is no efficiency gain but instead a degradation of performance because multiple pass of data is necessary and explicitly build SSCP matrix is certain slower than built-in capability of PROC REG. But this technique does serve the role to introduce our ultimate solution, a streaming type algorithm that constantly update SSCP and solve fo coefficients within one DATA Step. This method is demonstrated as Method[1] and explained below.

All we need to do to conduct OLS in a rolling window is to build the SSCP matrix over the rolling windows and obtain the coefficient estimates as well as other statistics based on the informaiton in SSCP using linear algebra techniques such as SWEEP operator [9], [12] and LU decomposition based method [16]. Notice that each element of SSCP matrix is of the following summation form [17]:

$$\mathbf{SSCP} = \begin{pmatrix} \sum \iota & \sum X & \sum Y \\ \sum X & \sum X^2 & \sum XY \\ \sum Y & \sum XY & \sum Y^2 \end{pmatrix}$$

When SWEEP operator is used, it sweeps the submatrix of:

$$\begin{pmatrix} \sum \iota & \sum X & \sum Y \\ \sum X & \sum X^2 & \sum XY \\ \sum Y & \sum XY & \sum Y^2 \end{pmatrix}$$

We should be able to build this matrix and then update it as new data is coming in. What we need to do is simply substract the corresponding terms came in very first in the window and add in the new terms just read in. For example, let the current SSCP be $SSCP_1$, built on observation j to N with a rolling window of length $N - j + 1$:

$$\mathbf{SSCP_1} = \begin{pmatrix} \sum_j^N \iota & \sum_j^N X & \sum_j^N Y \\ \sum_j^N X & \sum_j^N X^2 & \sum_j^N XY \\ \sum_j^N Y & \sum_j^N XY & \sum_j^N Y^2 \end{pmatrix}$$

As new observation, the $(N+1)^{th}$ record is read in, we need to first delete each of the elements above that involves j^{th} record, and then add in the corresponding elements involve $(N+1)^{th}$. The order of these two operations does not matter as long as the SSCP feeded to SWEEP operator or LU method is correct. To add in the elements involving the $(N+1)^{th}$ record, we simply update the elements using the following formula:

$$\begin{aligned}
\sum_{j}^{N+1} \iota &= \sum_{j}^{N} \iota + 1 \\
\sum_{j}^{N+1} X_i &= \sum_{j}^{N} X + X_{N+1} \\
\sum_{j}^{N+1} Y &= \sum_{j}^{N} Y + Y_{N+1} \\
\sum_{j}^{N+1} X_{ij}^2 &= \sum_{j}^{N} X^2 + X_{N+1,i} \times X_{N+1,j} \\
\sum_{j}^{N+1} X_i Y &= \sum_{j}^{N} XY + X_{N+1,i} Y
\end{aligned}$$

(4.1)

Allocate a storage matrix of size (w, K) where w is the length of rolling window, $w = N - j + 1$ in this case, and K is the total number of variables including intercept and dependent variable Y, then we can deduct the corresponding terms involving the oldest record in the window using the same above formula but instead of $+$ we use $-$. The index of this oldest record in the storage matrix can be calculated as $\mod (N - w - 1, w) + 1$, where N is current number of record and w is the window length, and this storage matrix needs to be updated with the newest incoming record whose index is then $\mod (N - 1, w) + 1$.

Because the code pieces are very long, they are put into the appendix for readers' review.

In order to realize the efficiency difference of all these methods, we simulated a data with 5E5 observations and 21 independent variables including intercept. The experiment results are shown below:

Methods	Real Time	CPU Time	Memory
Method 0	1.01 (seconds)	1.01 (seconds)	611K
Method 1	0.25 (seconds)	0.24 (seconds)	432K
Method 2	1.61 (seconds)	0.94 (seconds)	50381K
Method 3	80.54 (seconds)	79.61 (seconds)	2322K
Method 4	Failed	Failed	Failed

As we can see, hard coded solvers, such as Method[0], Method[0.1] and Method[1] has materially large advantage over the publically recorded PROC REG BY-processing method. The imporved version, Method[3.1], reduce time consumption from 90sec to 66sec, a 33% improvement.

References

[1] Vora, Premal P., "'Easy Rolling Statistics with PROC EXPAND"', *Proceedings of SGF*, 2008
[2] MYSAS.NET, `http://www.mysas.net/forum/viewtopic.php?f=4&t=8070`
[3] MYSAS.NET, `http://www.mysas.net/forum/viewtopic.php?f=4&t=7898`
[4] SAS-L, `http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0604D&L=sas-l&P=R32485`
[5] SAS-L, `http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0704C&L=sas-l&P=R3305`
[6] SAS-L, `http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0802C&L=sas-l&P=R9746`

[7] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0801C&L=sas-l&P=R14671

[8] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0810A&L=sas-l&P=R19135

[9] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0802C&L=sas-l&P=R13489

[10] Michael D Boldin, "Programming Rolling Regressions in SAS", *Proceedings of NESUG*, 2007

[11] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0604D&L=sas-l&D=0&P=56926

[12] J. H. Goodnight, "The Sweep Operator: Its Importance in Statistical Computing", *SAS Tech Report R-106*, 1978

[13] Kenneth Lange, *Numerical Analysis for Statisticians*, Springer, 1998

[14] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind1201a&L=sas-l&O=A&P=825

[15] SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0704c&L=sas-l&O=A&P=63501

[16] William H. Press, Brain P. Flannery, Saul A. Teukolsky, William T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988

[17] SAS Institute Inc., *SAS/STAT User Guide*, Cary, North Carolina

5. Appendix

```
options fullstimer;
data test;
     do seq=1 to 500000;
          x1=rannor(9347957);
          *x2=rannor(876769)+0.1*x1;
          epsilon=rannor(938647)*0.5;
          y = 1.5 + 0.5*x1 +epsilon;
          output;
     end;
run;

/* Method 0.*/
sasfile test load;
data res0;
        set test;
  array _x{3,3} _temporary_ ;
  array _a{3,3} _temporary_ ;
  array _tempval{5, 20} _temporary_ ;
  m=mod(_n_-1, 20)+1;
  _tempval[1, m]=x1;
  _tempval[2, m]=y;
  _tempval[3, m]=x1**2;
  _tempval[4, m]=x1*y;
  _tempval[5, m]=y**2;

  link filler;
  if _n_>=20 then do;
        if _n_>20 then do;
                   m2=mod(_n_-20, 20)+1;
       _x[1,2]+(-_tempval[1, m2]);
```

```
        _x[1,3]+(-_tempval[2, m2]);
        _x[2,2]+(-_tempval[3, m2]);
        _x[2,3]+(-_tempval[4, m2]);
        _x[3,3]+(-_tempval[5, m2]);
      end;
         do i=1 to dim(_a, 1);
          do j=1 to dim(_a, 2);
           _a[i, j]=_x[i, j];
       end;
      end;

               do k=1 to dim(_a, 1)-1;
          link adjust;
               end;
      Intercept=_a[1,3]; beta=_a[2,3];
      keep seq   intercept  beta;
      output;
  end;

  return;
filler:
   _x[1,1]=20; _x[1,2]+x1; _x[1,3]+y;
   _x[2,2]+_tempval[3,m];  _x[2,3]+_tempval[4,m]; _x[3,3]+_tempval[5,m];
   _x[2,1]=_x[1,2]; _x[3,1]=_x[1,3]; _x[3,2]=_x[2,3];
return;

adjust:
    B=_a[k, k];
 do j=1 to dim(_a, 2);
     _a[k, j]=_a[k, j]/B;
 end;
 do i=1 to dim(_a, 1);
     if i ^=k then do;
          B=_a[i, k];
    do j=1 to dim(_a, 2);
        _a[i, j]=_a[i, j]-B*_a[k, j];
    end;
  end;
 end;
return;

run;
sasfile test close;



/* Method 1.*/
```

```
sasfile test load;
data rest0;
        set test;
  array _x{4} _temporary_;
  array _a{2,20}  _temporary_;
  m=mod(_n_-1, 20)+1;
  _a[1, m]=x1; _a[2,m]=y;
  link filler;

  m2=mod(_n_-20, 20)+1;
  if _n_>=20 then do;
    if _n_>20 then do;
              link deduct;
    end;
    beta=(_x[2]-_x[1]*_x[4]/20)/(_x[3]-_x[1]**2/20);
    intercept=_x[4]/20 - beta*_x[1]/20;
    keep  seq   intercept  beta  ;
    output;
  end;
  return;
filler:
     _x[1]+x1;
  _x[2]+x1*y;
  _x[3]+x1**2;
  _x[4]+y;
return;
deduct:
     _x[1]=_x[1]-_a[1,m2];
  _x[2]=_x[2]-_a[1,m2]*_a[2,m2];
  _x[3]=_x[3]-_a[1,m2]**2;
  _x[4]=_x[4]-_a[2,m2];
return;
run;
sasfile test close;



/* Method 2.*/


%macro wrap;
%let ncovars=10;
%let window=25;
%let diff=%eval(&window-0);
%let ntotal1=%eval(&ncovars+2);
%let ntotal=%sysevalf( &ntotal1*(&ntotal1+1)/2);
data testv/view=testv;
     set test;
```

```
 array  z{&ntotal};
 array  x{&ntotal1}  intercept y x1-x&ncovars ;
 Intercept=1;
 do i=1 to &ntotal1;
        do j=i to  &ntotal1;
   k= (i-1) + (j-1)*j/2 + 1;
   z[k]=x[i]*x[j];
end;
 end;
     keep seq z:;
run;

proc expand data=testv  method=none  out=summary(keep=seq z:);
      convert  %do i=1 %to &ntotal;
                     z&i=z&i
%end;
                /transformout=(movsum &diff);
run;


data sscp(type=SSCP);
     retain Seq;

 length _TYPE_  _NAME_ $ 9;
     set summary;
 where seq>=&window;
 array x{&ntotal1} Intercept y x1-x&ncovars;
 array z{&ntotal} z1-z&ntotal;
 array _x{&ntotal1, &ntotal1} _temporary_;
 do i=1 to &ntotal1;
        do j=i to  &ntotal1;
   k= (i-1) + (j-1)*j/2 + 1;
   _x[i, j]=z[k];
   _x[j, i]=_x[i, j];
end;
 end;
 do i=1 to &ntotal1;
    do j=1 to &ntotal1;
   x[j]=_x[i, j];
end;
_TYPE_="SSCP";
_NAME_=vname(x[i]);  label _NAME_="Variable";
    output;
keep seq _TYPE_ _NAME_ Intercept y x1-x&ncovars;
 end;
 _TYPE_="N"; _NAME_=" ";
 do j=1 to &ntotal1;
x[j]= &window;
```

```
 end;
 output;
run;

proc reg data=sscp  outest=beta  noprint;
      by seq;
      model  y = x1-x&ncovars;
run;quit;


%mend;

%let t0=%sysfunc(datetime(), datetime24.);
*options nosource nonotes;
%wrap;
options source notes;
%let t1=%sysfunc(datetime(), datetime24.);
%put Start @ &t0;
%put End   @ &t1;


/* Method 3.*/
%let t0=%sysfunc(datetime(), datetime.);

data test2v/view=test2v;
       set test;
       array _x{2, 20} _temporary_ (20*0 20*0);
       k=mod(_n_-1, 20)+1;
       _x[1, k]=x1; _x[2, k]=y;
       if _n_>=20 then do;
          do j=1 to dim(_x, 2);
               x=_x[1, j]; y=_x[2, j];
               output;
               keep seq x y;
            end;
       end;
run;

ods select none;
proc  reg data=test2v  outest=res2(keep=seq x intercept);
         by seq;
         model y = x;
run;quit;
ods select all;


/* Method 3.1 */
%let t1=%sysfunc(datetime(), datetime.);
```

```
%put Start @ &t0;
%put End   @ &t1;

%let t0=%sysfunc(datetime(), datetime.);
%let ncovars=10;
%let ntotal1=%eval(&ncovars+2);
%let window=25;
data test2v/view=test2v;
       set test;
       array _x{&window, &ntotal1} _temporary_ ;
   array x{&ntotal1}  Intercept y x1-x&ncovars;
       k=mod(_n_-1, &window)+1;
   do j=1 to &ntotal1;
          _x[k, j]=x[j];
   end;
       if _n_>=25 then do;
          do k=1 to &window;
       do j=1 to &ntotal1;
                  x[j]=_x[k, j];
   end;
               output;
               keep seq  Intercept y x1-x&ncovars;
           end;
       end;
run;

ods select none;
proc  reg data=test2v  outsscp=sscp2 ;
         by seq;
         var y  x1-x&ncovars;
run;quit;
proc  reg data=sscp2   outest=res2x(keep=seq x: intercept)  noprint;
        by seq;
model y =x1-x&ncovars;
run;quit;
ods select all;

%let t1=%sysfunc(datetime(), datetime.);
%put Start @ &t0;
%put End   @ &t1;


/* Method 4. */
%macro wrap;
options nonotes;
ods select none;
%do i=20 %to 500000;
       %let fo=%eval(&i-19);
       proc reg data=test(firstobs=&fo  obs=&i)  outest=_xres(keep=x1 intercept);
           model y =x1;
       run;quit;
      %if %eval(&i=20) %then %do;
          data res3; set _xres; run;
      %end;
      %else %do;
        proc append base=res3  data=_xres; run;
      %end;
%end;

ods select all;
data res3;
       set res3;
       time=19+_n_;
run;
options notes;
%mend;

%let t0=%sysfunc(datetime(), datetime.);
%wrap;
%let t1=%sysfunc(datetime(), datetime.);
%put Start @ &t0;
%put End   @ &t1;
```