10th August 2011

Rolling Window Regression of Time Series

We demonstrate a comparison among various implementations of Rolling Regression in SAS and show that the fastest implementation is over **3200X** faster than traditional BY-processing approach.

More often than not, we encounter a problem where an OLS over a rolling time window is required, see [1], [2], [3], [4], [5], [6], [7], for a few examples.

One solution is to resort to SAS MACRO, but it is extremely inefficient and can't handle large dataset in reality, [8]. This method is shown below as Method[4]. It couldn't finish the test in allowable time using the sample data below.

The other common solution is to use the BY-processing capability of PROC REG after re-shaping the data into appropriate format, see [9], [10]. This method is demonstrated as Method[3] below. While certainly much better than above one, it is still not the fastest and requires more memory.

The third solution comes to play by recognizing that in OLS, what you need is the SSCP and you can easily build up the SSCP over rolling time window by resorting to PROC EXPAND. This is demonstrated as Method[2] below. This approach will further improve the speed but still requires large amount of memory if the data is big and many rolling windows are generated.

Since what we need to do is to build the SSCP matrix and obtain the coefficient estimates based on the informaiton in SSCP, we can certainly code this in a DATA Step using ADJUST operator, which provides a solution that is both fast and low memory occupancy. See [11] for an introduction to ADJUST operator. To make this even faster, a modification of ADJUST operator, the SWEEP operator, can be used. For an introduction to SWEEP operator, see [11], [12]. In the code below, Method[0] implements the ADJUST operator, while Method[1] implements the SWEEP operator.

The experiment literally runs **499980 regressions** each with **20 observations and 2 predictors**, and the results are shown below:

	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

Reference:

[1]MYSAS.NET, http://www.mysas.net/forum/viewtopic.php?f=4&t=8070 [http://www.mysas.net/forum/viewtopic.php?f=4&t=8070]

[2]MYSAS.NET, http://www.mysas.net/forum/viewtopic.php?f=4&t=7898 [http://www.mysas.net/forum/viewtopic.php?f=4&t=7898]

[3]SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0604D&L=sas-l&P=R32485 [http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0604D&L=sas-l&P=R32485]

[4]SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0704C&L=sas-l&P=R3305 [http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0704C&L=sas-l&P=R3305]

[5]SAS-L, http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0802C&L=sas-l&P=R9746 [http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0802C&L=sas-l&P=R9746]

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

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

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

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

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

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

[12]Kenneth Lange, "Numerical Analysis for Statisticians", Springer, 1998

```
proc datasets library=work kill; run;


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 window=20;
%let diff=%eval(&window-0);
data testv/view=testv;
     set test;
       xy=x1*y;
run;

proc expand data=testv  method=none  out=summary(keep=seq sumxy  sumx1  sumy  ussx1
       convert  x1=sumx1/transformout=(movsum &diff);
       convert  xy=sumxy/transformout=(movsum &diff);
       convert  x1=ussx1/transformout=(movuss &diff);
       convert  y =sumy /transformout=(movsum &diff);
       convert  y =may / transformout=(movave &diff);
       convert  x1 =max / transformout=(movave &diff);
run;

data result1;
     set summary(firstobs=&window);
       beta = (sumxy - sumx1*sumy/&window)/(ussx1 - sumx1/&window.*sumx1);
       alpha= may - beta*max;
       keep seq  beta  alpha;
run;
%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;

%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;
```

Posted 10th August 2011 by Liang Xie

Labels: Data Manipulation, PROC EXPAND, PROC REG

1 View comments

CHARLIE HUANG 1:49 PM, August 12, 2011

Excellent show of using DATA STEP array in complicated computation. Thanks.

Reply