Rapid Rolling Window Regressions via Home Made Sum of Squares and Cross Products

Mark Keintz
Wharton Research Data Services
PhilaSUG
March 19, 2019

The Object: Efficient Rolling Window Regressions

- Explore Patterns and Relationships that vary over time
- Generate model for every rolling window of a given size (vs. a single model for entire span)
- The data set size becomes large quickly
- At WRDS: 29,435 firms averaging 2,810 dates

10 Billion rows for 120 day rolling windows

The Object: Efficient Rolling Window Regressions

- Take Home Points
  - Save disk space – use data set VIEWS
  - Read each record once, but write it multiple times (holding and writing)
  - Consider “type=SSCP” data sets (fixed size regardless of the size of the window)

Brute Force Approach
Make Windows Data Set, run PROC REG

- For a data set of 900 records and window size 90, output 72,990 records (=811*90)
  - Window 1: Records 1 through 90
  - Window 2: Records 2 through 91

    …
  - Window 811: Records 811 through 900

  (Question: How to order the repeated output data)

- Run PROC REG for each Window

```
PROC REG data= …;
 by window;
 …
```

Brute Force, Simple Program

```
DATA rwin / view=rwin;
  ws = 90;
  nwin = nrecs - ws +1;
 do w=1 to nwin;
  do p=w to w+ws-1;
    set myseries point=p nobs=nrecs;
    output;
  end;
 end;

 stop;
run;
proc reg data=rwin noprint outest=myests;
 by w;
 model y=x;
quit;
```

Brute Force, Holding and Writing

```
DATA rwin / view=rwin;
  array _X {90} _temporary_ ;  /*What is TEMPORARY array?*/
  array _Y {90} _temporary_ ;
  set myseries;
  i=mod(_N_,90)+1;          /*Determine which array element*/
  _X{i}=x;                  /*Park values in the array */
  _Y{i}=y;
if _N_>=90 then do i= 1 to 90;  /*Only do complete windows*/
   x=_X{i};     y=_Y{i};
   output;
  end;
run;
proc reg data=rwin noprint outest=myests;
  by date;
  model x=y;        quit;
```

Rolling Series, by ID

```
DATA rwin / view=rwin;
  array _X {90} _temporary_ ;
 array _Y {90} _temporary_ ;
 set myseries;
  by id;
 retain N 0;


 N = ifn(first.id,1,N+1);   /*Within-ID counter*/


 i=mod(N-1,90)+1;          /*Determine which array element*/
                            /*NOTE: Order doesn't matter   */

 _X{i}=x;
 _Y{i}=y;
 ….
```

Rolling Series, by ID

```
….
if N>=90 then do i= 1 to 90;    /*Complete windows only*/
    x=_X{i};
    y=_Y{i};
    output;
  end;
run;

proc reg data=rwin noprint outest=myests;
  by id date;
  model y=x;
quit;
```

Pass on Rolling SSCP
Not Rolling Data

- SSCP = “Sum of Squares and Cross Products”
  - Effectively equivalent to a correlation matrix, but is not standardized nor “de-meaned”.

- Can be generated by PROC CORR, PROC REG and other procedures

- Can be INPUT by PROC REG

- SSCP size is NOT proportional to window size
  (it is proportional to number of variables-squared

The TYPE=SSCP Data Set

- SAS also has type=CORR, COV, EST, FACTOR
- Type is reportable by PROC CONTENTS

ID	DATE	_TYPE_	_NAME_	Intercept	X	Y
1001	19860515	SSCP	Intercept	90.000	0.0916	0.5343
1001	19860515	SSCP	X	0.0916	0.0048	0.0030
1001	19860515	SSCP	Y	0.5343	0.0030	0.1754
1001	19860515	N		90	90	90

Rolling SSCP via PROC EXPAND

```
DATA vtemp / view=vtemp;
  set myseries;
  xx=x*x;         /*Prepare un-summed squares ...     */
  yy=y*y;
  xy=x*y          /* ... and crossproducts for proc expand */
  n=1;
run;
proc expand data=vtemp   method=none
     out=sscpdata (where=(_n=90));
  by id;
  id date;
  convert x y xy xx yy n /     transformout=(MOVSUM 90);
run;
```

Convert Normal SSCPDATA data set into a “type=SSCP” data set

```
data rsscp (type=SSCP keep = id date _TYPE_ _NAME_ intercept x y)
  / view=rsscp;
  retain id date _TYPE_ _NAME_ intercept x y;
  set sscpdata;
  length _TYPE_ $8  _NAME_ $32 ;
  _sumy=y;  _sumx=x;    ** Store for later use **;

  _TYPE_="SSCP";  /* For the record type, not the data set type*/
  ** First record is just N, and sums already in each original variable **;
 _NAME_='Intercept';
  Intercept=_n;
  y=_sumy;  x=_sumx;
  output;
```

Convert Normal SSCPDATA data set into a "type=SSCP" data set

```
   _name_="X";
   intercept=_sumx;  x=xx;   y=xy;
   output;
  _name_="Y";
   intercept=_sumy;  x=xy;   y=yy;
   output;
  _TYPE_='N';
  _NAME_=' ';
   Intercept = _n;   Y=_N;   X=_N;
   output;
run;
proc reg data=rsscp noprint outest=myests;
  by id date;
  model y=x;
quit;
```

SSCP Window vs Data Window

- Space Considerations
  - Data set *file* windows require a lot of disk space (proportional to window sizes)
  - Data set *view* eliminates this (it’s just-in-time data, never stored on disk)
  - But SSCP data is proportional to number of variables
- Timing
  - One test found the SSCP approach more efficient than data window at window size of 100 (for 2 variables)

Questions?

Mark Keintz
mkeintz@wharton.upenn.edu