Using PROC FCMP to Solve Rolling Regression Rapidly
Chao Huang, Oklahoma State University
Liang Xie. SAS User

ABSTRACT

Ordinary Least Squares (OLS) regression is a widely used technique to model financial time series. A rolling regression is the regression on a fixed sample length or time horizon by moving the beginning and ending boundaries, which involves thousands of OLS equations and thus requires a lot of computing power. Starting from 9.1, the FCMP procedure in Base SAS®, provides the matrix manipulation feature by some of its native routines. PROC FCMP deals with matrices in the pre-allocated memory, which significantly increases the efficiency to solve the rolling regression. In this paper, we demonstrate how to solve rolling regressions to backtest Jensen’s alpha, portfolio beta and coefficient of determination with a simulated asset portfolio against market benchmark.

INTRODUCTION

Rolling regression backtests a statistical model on historical data and evaluates stability and predictive accuracy. Usually least-squares techniques are used to fit a linear equation and estimate the corresponding coefficients multiple times using partially overlapping subsamples from historical data. A model’s stability over time can be evaluated by rolling analysis of time series data. 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.

Since currently SAS does not provide any specialized procedure or option for rolling regression analysis, some alternative solutions have been proposed. Michael provided a macro %ROLLINGREG based on the REG procedure to solve the parameters of a rolling regression [1]. Since PROC REG and other regression procedures in SAS generate a variety of listings and graphs, the overheads may cause low efficiency in solving these matrix-related parameters, such as from rolling regression. As a matrix language, PROC IML or SAS/IML is an ideal tool but requires the purchase of extra license, which may limit its application under industrial setting.

Since SAS/BASE 9.1, the FCMP (Function Compiler) procedure provides the functionality that creates, tests, and stores user-defined functions and CALL routines in DATA step. In this paper, we explore a new method to use PROC FCMP to conduct low-level matrix operations to solve the parameters of the rolling regression. An example with a macro encapsulating the FCMP procedure that estimates Jensen’s alpha is illustrated.

AN EXAMPLE THAT ESTIMATES JENSEN’S ALPHA

Market usually produces huge volume of time series data with millions of records. Rolling regression calculates parameter estimates over a rolling window of a fixed size through the sample, and is a common technique to assess the constancy of a model’s parameters. If the parameters are indeed 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. For example, Jensen’s alpha is a widely used financial measurement to determine the abnormal return of a security or portfolio of securities over the theoretical expected return, and estimated by Ordinary Least Squares (OLS) regression [2].

\[ \alpha_j = R_j - [R_f + \beta_{IM} \times (R_M - R_f)] \]

Here \( \alpha_j \) is Jensen’s alpha, \( R_j \) is the realized return on the portfolio, \( R_M \) is the market return, \( R_f \) is the risk-free rate of return, and \( \beta_{IM} \) is the beta of the portfolio.

At the beginning, we simulate a portfolio with 20,000 daily returns for the market and a portfolio XYZ, which are randomly generated in a DATA step with an estimated portfolio beta (\( \beta_{IM} \)) that equals 0.4, an estimated Jensen’s alpha (\( \alpha_j \)) that equals 0.01, and a Mean Squared Error (MSE) that equals 0.9. The time series demonstrate the fluctuating random values of the market daily return (Figure 1a) and the realized daily return on the portfolio XYZ (Figure 1b).

/* 1. Simulate dataset of time, market return and realized portfolio return*/
data simuds;
  format  day date9.;
  _beta0 = 0.01;
\[ _\beta_1 = 0.4; \]
\[ _\text{mse} = 0.9; \]
\[ \text{do day = today() - 20000 to today();} \]
\[ \text{market\_return = rannor(1234)/100;} \]
\[ \text{portfolio\_XYZ = _beta0 + _beta1*market\_return + _mse*rannor(3421)/100;} \]
\[ \text{output;} \]
\[ \text{end;} \]
\[ \text{drop _:; run;} \]

Then a macro that wraps all the matrix manipulations under PROC FCMP is created, which allows the pass-through of 6 parameters. A few CALL routines, such as TRANSPOSE, INV and MULT, are applied to inverse, transpose and multiply matrices. The arrays are declared to accommodate the transitional and outcome matrices during the overall process, before entering a looping structure. Therefore, in the rolling regression, the normal equations are solved for Least-Squares estimators. The dataset containing the source data is read by a CALL routine READ_ARRAY, and the dataset is written by another CALL routine READ_ARRAY. The final output dataset by this SAS macro is the merged product between the source dataset and the resulting dataset.

```sas
/* 2. Create a macro called rollreg based on proc fcmp */
%macro rollreg(data =, wlength =, benchmark =, portfolio =, rfree =, out =);
/* Calculate the loops needed for rolling regression */
data _1;
set &data nobs = nobs;
  _benchmark = &benchmark - &rfree;
  _portfolio = &portfolio - &rfree;
  call symput('nloop', nobs - &wlength + 1);
  call symput('nobs', nobs);
  _obsnum = _n_;
run;
/* Manipulate the matrices in the FCMP procedure */
proc fcmp;
/* Allocate memory for matrices */
array input[&nobs, 2] / nosym;
array y[&wlength] / nosym;
array ytrans[1, &wlength] / nosym;
array xtrans[2, &wlength] / nosym;
array x[&wlength, 2] / nosym;
array sscp[2, 2] / nosym;
array sscpinv[2, 2] / nosym;
array result[&nloop, 4] / nosym;
array beta_xtrans_y[1] / nosym;
array xtrans_y[2, 1] / nosym;
array ytrans_y[1] / nosym;
/* Input source dataset */
rc1 = read_array("_1", input, "benchmark", "portfolio");
/* Calculate values for OLS regression coefficients and r square */
do j = 1 to &nloop;
  ytotal = 0;
  do i = 1 to &wlength;
```

Coders' Corner

NESUG 2012
xtrans[2, i] = input[i+j-1, 1];
xtrans[1, i] = 1;
y[i] = input[i+j-1, 2];
ytotal = ytotal + y[i];
end;
call transpose(y, ytrans);
call mult(ytrans, y, ytrans_y);
call transpose(xtrans, x);
call mult(xtrans, x, sscp);
call inv(sscp, sscpinv);
call mult(xtrans, y, xtrans_y);
call mult(sscpinv, xtrans_y, beta);
ymeansquare = ytotal**2 / &wlength;
result[j, 1] = beta[1];
result[j, 2] = beta[2];
result[j, 3] = (beta_xtrans_y[1]-ynmeansquare)/(ytrans_y[1]-ynmeansquare);
result[j, 4] = j + &wlength - 1;
end;
/* Output the resulting matrix as dataset */
rc2 = write_array("_2", result, 'beta0', 'beta1', 'rsquare', '_obsnum');
if rc1 + rc2 > 0 then put 'ERROR: I/O error';
else put 'NOTE: I/O was successful';
%put 'Total observations in the source dataset = ' &nobs;
%put 'Total number of the regressions run = ' &nloop;
quit;

/* Merge the source dataset and the resulting dataset */
data &out;
merge _1 _2;
by _obsnum;
drop _;:
run;
%mend;

Eventually, the macro above is tested with the risk-free rate of return as 0.001. Under a 50-observation length window or a 50-day time horizon, 19,952 OLS regressions are carried out. As a result, the Jensen’s alpha (Figure 1c), the beta of the portfolio XYZ (Figure 1d), the coefficient of determination (Figure 1c), are estimated and visualized. In this experiment on a 2-year-old Windows laptop, the time cost, either system time or CPU time, is negligible, which proves the efficiency of this method.

/* 3. Use the macro created previously with six specified parameters */
%rollreg(data = simuds, wlength = 50, benchmark = market_return, portfolio = portfolio_XYZ, out = result, rfree = 0.001);

CONCLUSION

There are a few advantages using PROC FCMP for matrix operations such as solving rolling regression. The array structure in PROC FCMP, other than DATA step array, supports row-wise and column-wise index notation and is a proper vehicle for matrix-like data loading. In addition, PROC FCMP supplies 13 native CALL routines for fundamental matrix operations, and two other CALL routines READ_ARRAY and WRITE_ARRAY that allow the bilateral transformation between PROC FCMP array and DATA step dataset. Finally PROC FCMP belongs to SAS/BASE, which means that this procedure is available with any SAS system and thus the codes on it is portable for all platforms.

To summarize, PROC FCMP is not only a function compiler by its original purpose, but also an application development tool for complicated system such as rolling regression analysis.
Figure 1. Simulated dataset and the rolling regression results. (a) A time series plot for market daily return. (b) A time series plot for realized daily return on the portfolio XYZ. (c) A time series plot for the estimated Jensen’s alpha, after solving rolling regressions by the method described in this paper. (d) A time series plot for the estimated beta of the portfolio XYZ. (e) A time series plot for the estimated market daily return.

REFERENCES


CONTACT INFORMATION

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

Chao Huang
Email: hchao8@gmail.com
Blog: www.sasanalysis.com

Liang Xie
Email: xie1978@yahoo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.