Using a hash object to seed initial conditions

MSUG 2/2015

CHUCK ANDERSON

UFA LLC

ANDERSON@UFANET.COM

Background

- Certain characteristics of dynamic models often emerge when underlying model variables are near equilibrium
  - Mean-reversion
  - Stationarity
  - Cointegration
- We'll discuss using hash objects in the context of a predictive model which employs time series composed of observations occurring at irregular time intervals, plus other dynamic variables

Additional background: schematic of predictive model

Motivation

- Perform QA testing of model behavior
- Test simulations vs. theoretical estimates
- Compare moments, other statistics
- Perform comparison under assumption of stationarity

Goal

- Incorporate equilibrium initial conditions into time series dataset
- Why? Because arbitrary initial conditions often do not result in stationarity, mean-reversion, etc.

Data structure

- Stacked time series: time series from one model element, followed by another time series for another element
- For this example each individual time series needs 4 separate initial conditions specified

We need to provide initial conditions at the beginning of each symbol's time series

Symbol	TradeTime	TradePrice	Param1	Param2	Param3	Param4
ABC	**9:30.1**	**3.00**	**0.022**	**0.913**	**0.184**	**3.82**
ABC	9:30.3	3.04	0.023	0.928	0.191	3.96
ABC	9:35.9	3.02	0.022	0.923	0.188	3.95
ABC	9:56.2	3.01	0.021	0.919	0.193	3.74
XYZ	**9:38.0**	**20.01**	**0.056**	**0.722**	**0.412**	**2.07**
XYZ	9:45.9	20.30	0.057	0.713	0.417	2.11
XYZ	9:52.2	21.05	0.053	0.715	0.414	2.09
XYZ	9:56.5	21.50	0.050	0.721	0.416	2.10
QQQ	**9:45.0**	**15.98**	**0.087**	**0.313**	**0.210**	**1.77**
QQQ	9:45.3	16.01	0.092	0.308	0.205	1.73
QQQ	9:47.2	15.97	0.091	0.311	0.204	1.75
QQQ	9:48.1	16.02	0.090	0.309	0.207	1.76

What are our choices?

- Data merge
- SQL join
- Macro code
- Format
- Hash object
- Other?

Hash object characteristics

- Simple example of ds2 element
- Has properties of an object
  - Methods
  - Attributes
- “Lives” in volatile memory (RAM)
- Has many methods for navigating or manipulating the object within memory

General usage review

- Requires instantiation
- Commands are set up to use method calls
- Method calls generate a return code available for further programming

Specific steps required to setup hash object

1. Create and name the hash object
2. Specify key variable(s)
3. Specify data variable(s)
4. Complete definitions

Using the hash object

- Here we are using it as a look-up table
- To find an entry in the hash table we invoke its find() method
- During initialization we use the CALL MISSING routine. This prevents extraneous warning messages from being written to the SAS log
- Then at the beginning of each symbol's time series we call the find()method

Using hash object, continued

- Need to invoke hash object at beginning of DATA step execution, during the first iteration of the implicit data loop
- Invoke at _n_ = 1. SAS creates and loads the hash object
- NOTE: This is NOT related to our use for seeding initial conditions

Equilibrium values dataset structure

Our dataset named equilibParams contains the symbol and the equilibrium values for the four parameters

Symbol	Param1	Param2	Param3	Param4
ABC	0.022	0.913	0.184	3.12
XYZ	0.056	0.722	0.412	2.07
QQQ	0.087	0.313	0.210	1.77
LLL	0.038	0.441	0.387	2.89
ZZZ	0.066	0.583	0.609	4.20

SAS code

MSUG 2/2015

```
data workit2;
  attrib symbol length=$3;
  if _n_ = 1 then do;
       declare hash seedVals(dataset: "equilibParams");
       rc = seedVals.defineKey('symbol');
       rc = seedVals.defineData('param1',
                                'param2',
                                'param3',
                                'param4'
                                );
       rc = seedVals.defineDone();
       call missing(symbol,
                    param1,
                    param2,
                    param3,
                    param4
                    );
    end;
  set workit1;
```

Done invoking, now find

- In this data set we call the hash object to seed the initial value for the time series of each symbol


```
by symbol;
if first.symbol then do;
     StartTime      = TradeTime;
     LastTradePrice = TradePrice;
     rc = seedVals.find();
end;
else do;
```

Time-saver trick

- Odd and interesting usage of _n_ = 0

  We need to let the DATA step know the attributes of the key and data elements.

  For example we did that with the LENGTH statement.

  You **don't have to do this** if you are reading from a dataset. You can exploit the dataset to provide the necessary metadata:

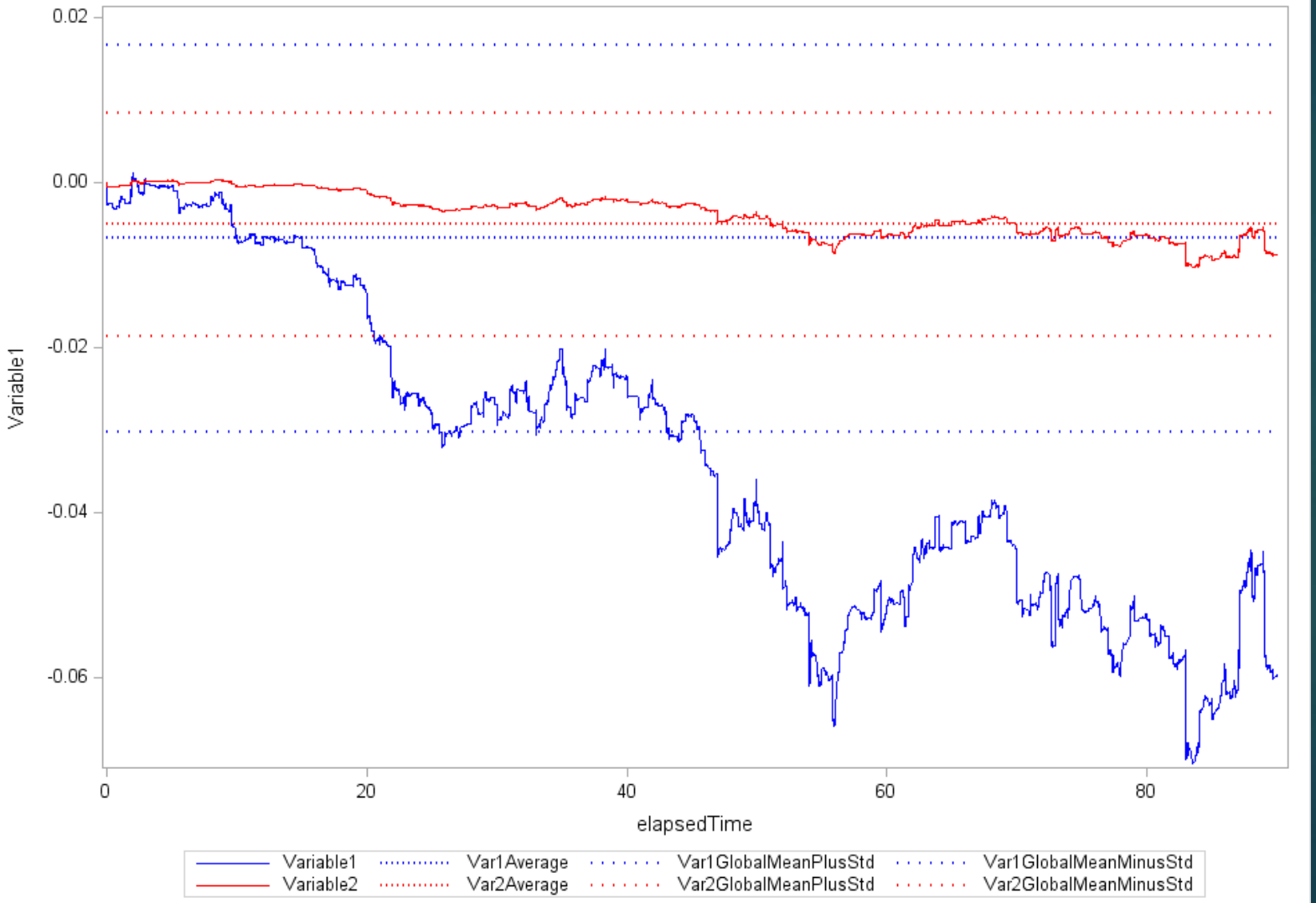
```
If _n_ = 0 then set equilibParams;
```

  Explanation:
  _n_ never equals 0, the condition is not satisfied -- nothing is executed and no dataset records are read.

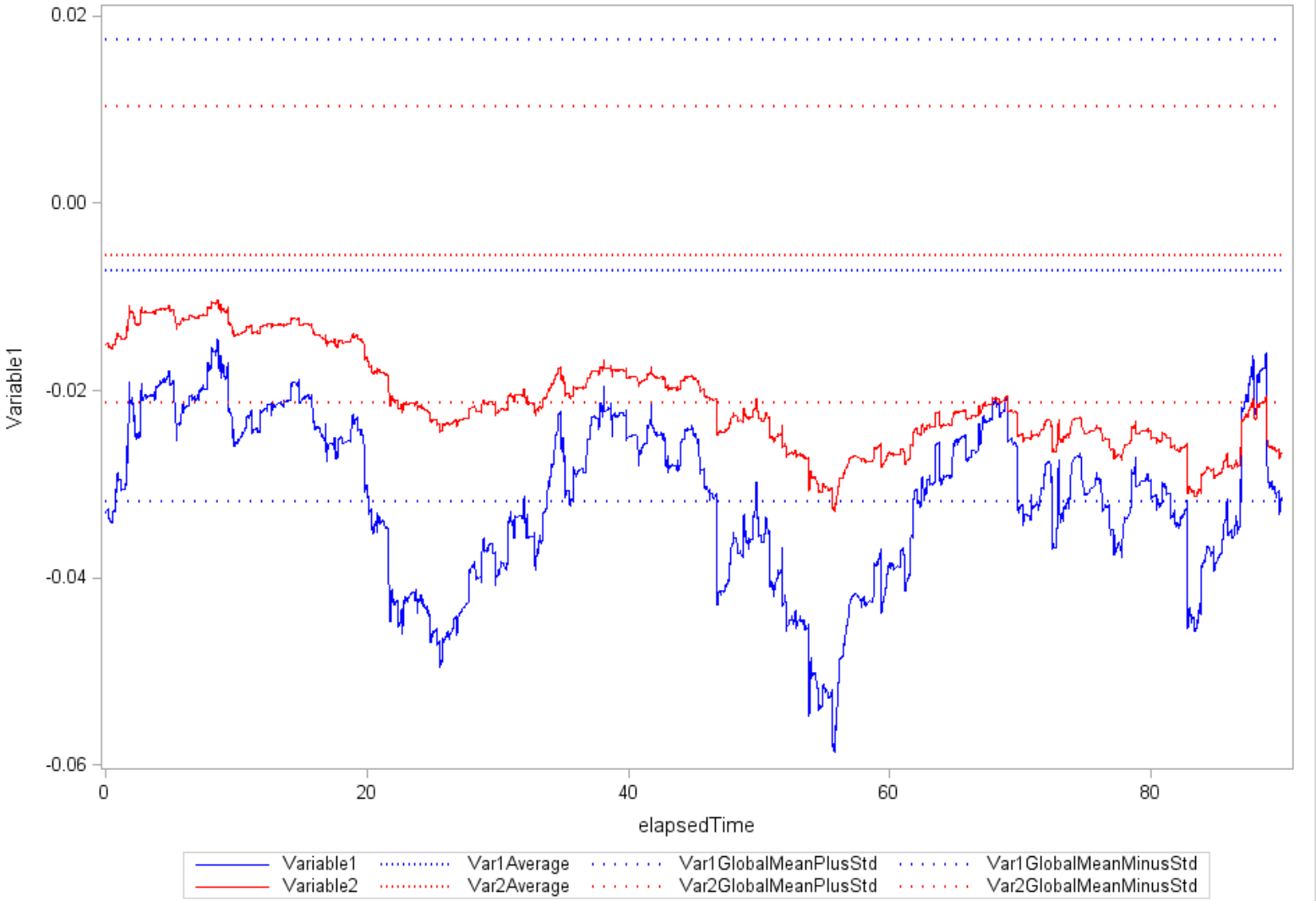
  BUT, the SAS compiler will read the metadata (in particular the column names and attributes) from the dataset and add the columns to the program data vector

Illustrative examples

Example using arbitrary seed value



Example using equilibrium seed values



Conclusions

If you use the wrong seed

If you use the right seed