

# SNM Guide

Michael Creel<sup>1</sup> and Dennis Kristensen<sup>2</sup>

November 13, 2009

<sup>1</sup>Universitat Autònoma de Barcelona and MOVE, [michael.creel@uab.es](mailto:michael.creel@uab.es)

<sup>2</sup>Columbia University and CREATES, [dk2313@columbia.edu](mailto:dk2313@columbia.edu)

## Abstract

This is a guide that explains how to use software that implements the simulated nonparametric moments (SNM) estimator proposed by Creel and Kristensen (2009). The guide shows how results of that paper may easily be replicated, and explains how to install and use the software for estimation of simulable econometric models.

Keywords: econometric software; dynamic latent variable models; simulation-based estimation; simulated moments; kernel regression; nonparametric estimation

JEL codes: C87; C13; C14; C15

## 1 Introduction

SNM is a simulation-based estimation method proposed by Creel and Kristensen (2009), henceforth referred to as CK2009. This guide describes how to install and use software that implements the estimator. As an example of usage, we show how the results of the paper can easily be replicated. The guide presumes that the reader is familiar with CK2009, and it makes usage of terminology and notation defined in the paper.

The SNM software is a set of scripts and compiled code that runs on GNU Octave ([www.octave.org](http://www.octave.org)). GNU Octave is a high-level matrix programming language which is mostly compatible with Matlab. GNU Octave runs on Linux, Windows, and Mac OS X, among other platforms. GNU Octave can be made to run, with relative ease, on multiple cores of a single computer, or on a distributed cluster of computers, without the need to purchase and administer licenses. For its portability, power, and low cost, GNU Octave is a good choice for econometric applications that make high computational demands.

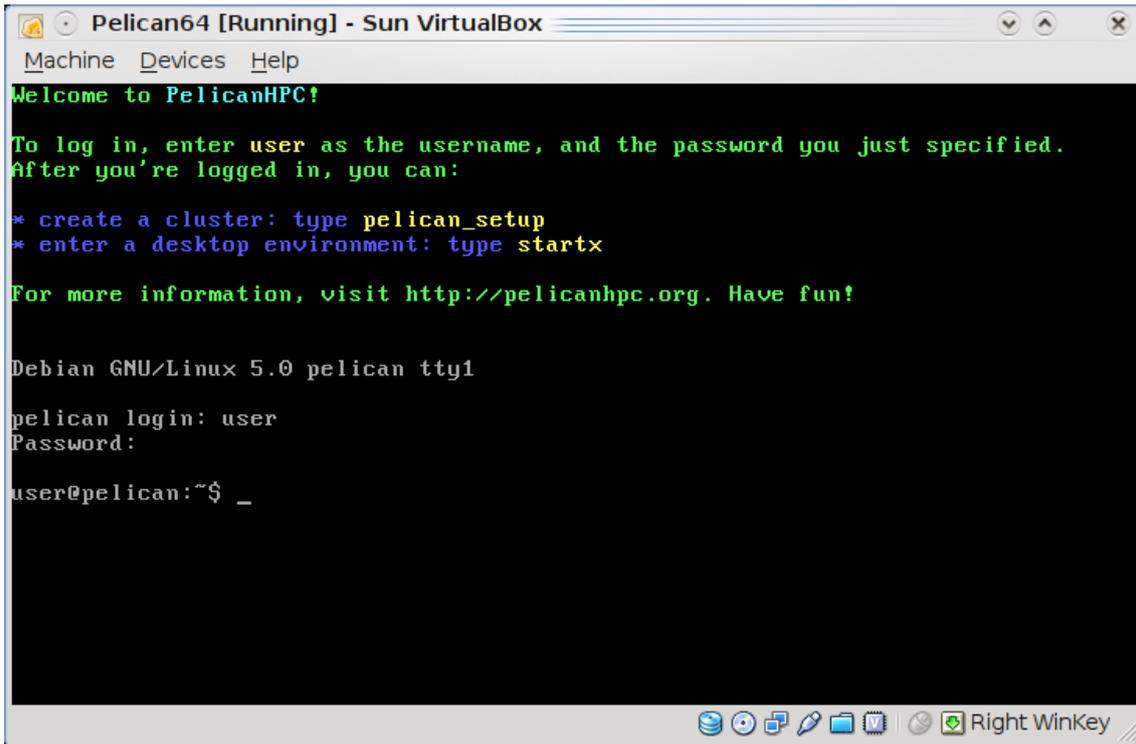
Many people who might be interested in the SNM estimator will be using different operating systems, and many will not have GNU Octave installed. The SNM estimator can be run in parallel using the “message passing interface” (MPI). Most users will not have a working MPI installation. For these reasons, the Guide first explains a method of using Octave, SNM and MPI that does not require installing any software. This method works identically independent of the user’s installed operating system, and it allows one to use multiple cores and/or a real computational cluster. Next, the SNM software is described in some detail, so that users can understand how it works. Finally, the Guide explains how to install the SNM software for use with an installed version of Octave. Installation of MPI is not dealt with here.

## 2 Quick start using PelicanHPC

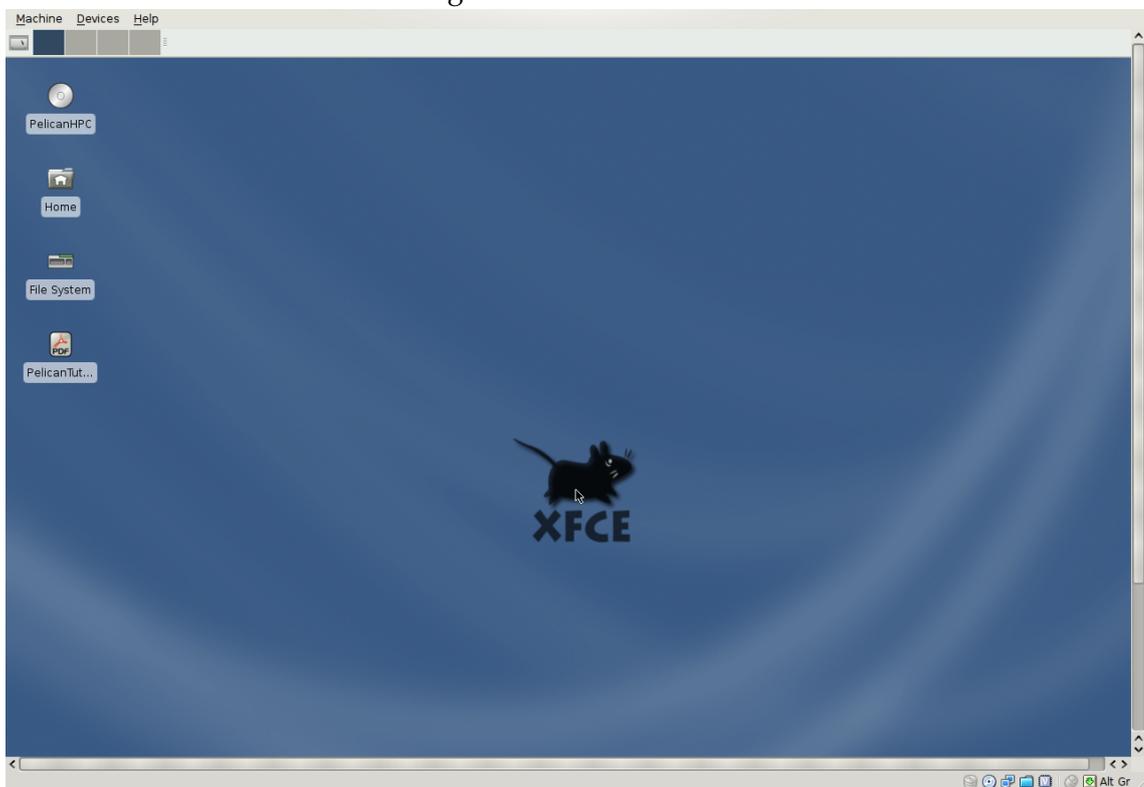
PelicanHPC (Creel, 2009) is a distribution of GNU Linux oriented to setting up a computational cluster to run parallel programs that use MPI. Recent versions of PelicanHPC contain all of the code needed to use the SNM estimator, with the examples that are presented in the paper. For purposes of exploration where compatibility is more important than performance, one should obtain version 1.9.x, for 32 bit architectures. For serious work on computers with 64 bit CPUs, the 64 bit version is recommended. A tutorial is available for PelicanHPC (Creel, 2008).

Once the ISO image file has been obtained, one can either burn the image to a CD and then use the CD to boot a computer, or a virtualization platform may be used to boot a virtual computer using the ISO image file. This second method has the advantage that one may use PelicanHPC without leaving one’s operating system of choice, with no need to reboot the computer. Virtualbox ([www.virtualbox.org/wiki/Downloads](http://www.virtualbox.org/wiki/Downloads)) is a freely downloadable virtualization platform for Linux, Mac OS X and Windows.

- One should either boot a computer using a PelicanHPC CD, or boot a virtual machine using the PelicanHPC ISO image file. Once the machine has booted, the instructions in the PelicanHPC Tutorial (Creel, 2008) should be followed (important: answer YES when asked if the example software should be copied) until one has logged into PelicanHPC, and sees

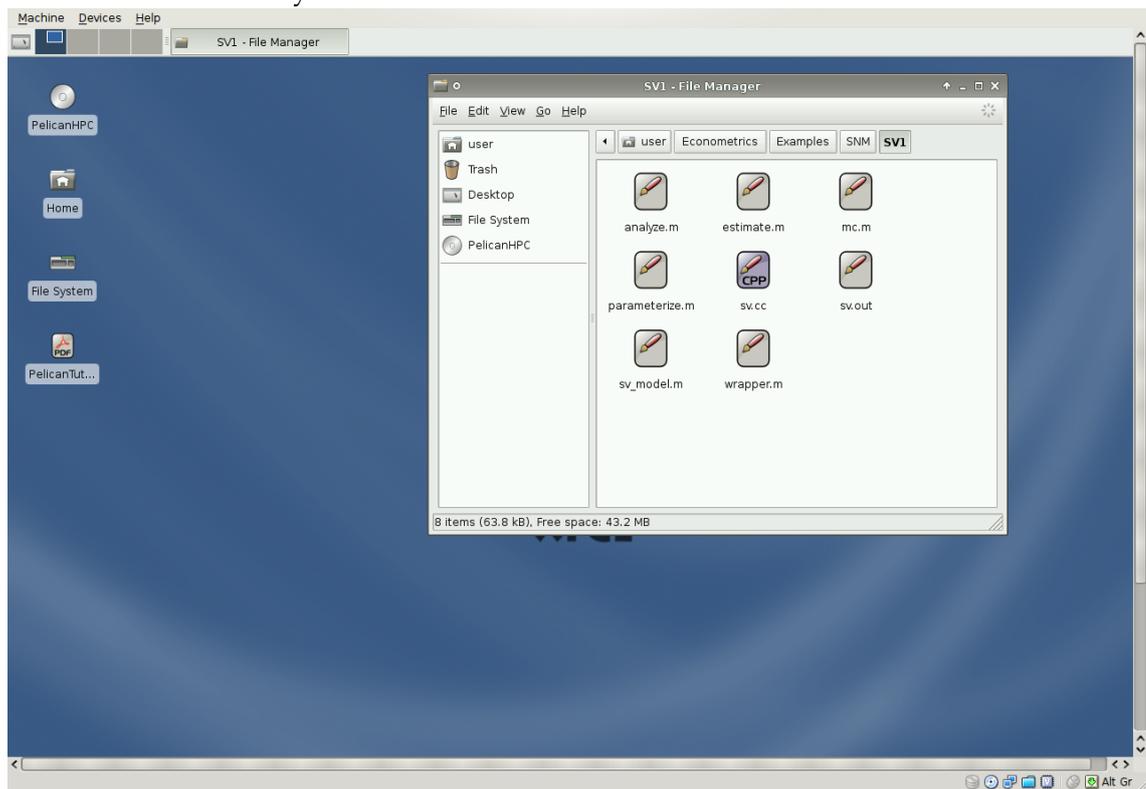


- At this point, one should type "startx" to enter into a graphical desktop environment. You should see the following:

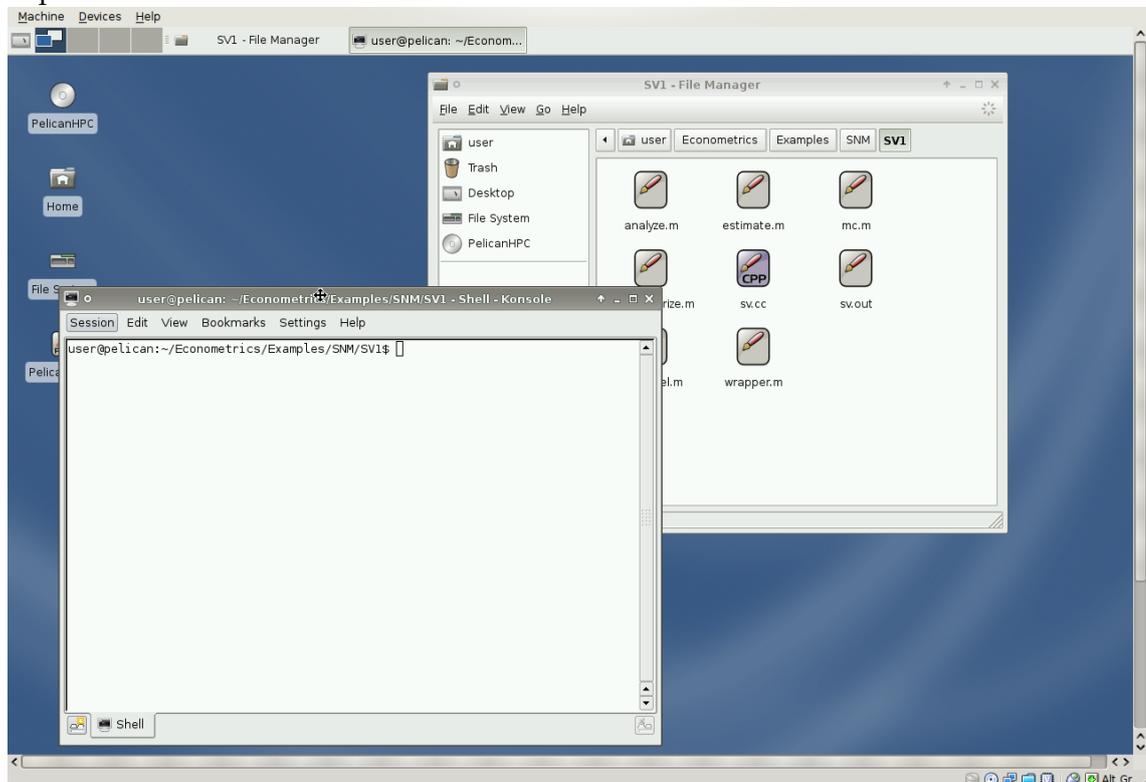


- Navigate to the folder /home/user/Econometrics/Examples/SNM by double-clicking on the Home icon, and then the folders in the mentioned path. Once there, read the REAME file, by double-clicking on it.

- Enter the SV1 directory. You should see

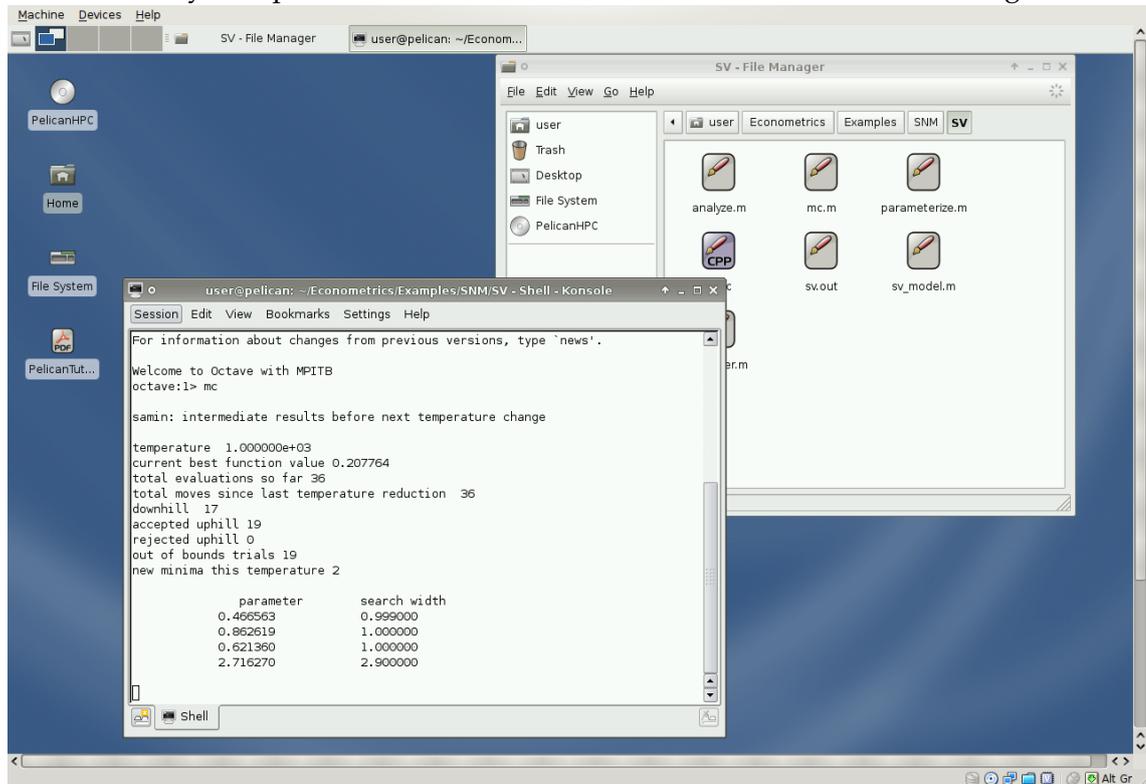


- right-click within the window that shows the icons in the SV directory, and select "open terminal here". You should see



- at the command line prompt in the terminal window, type "octave". At the octave

prompt, type “mc”. This is the method by which the Monte Carlo results in Table 2 of CK2009 may be replicated. You should see, after a minute or two, something like



Completing replication of the Monte Carlo results would take about 2 weeks on a single core, so you should interrupt the program by typing CTRL-C. To do Monte Carlo work with the SNM estimator, one should probably use a cluster. To learn how to set up a cluster, consult the PelicanHPC Tutorial. Once you have a running cluster you just need to edit the file mc.m and set compute\_nodes equal to the number of cores available in your cluster.

- to see an example of estimation, type “estimate” from the Octave prompt. This performs SNM estimation using a single sample of 500 observations, drawn from the SV1 data generating process (true parameter values are 0.900, 0.692, 0.363). Estimation on a single core will take about 10 minutes or so. When it finishes, one should see something like

```

0.7000000      0.7000000
0.305889      0.000041
2.999958      0.000001

=====
SAMIN results

==> Normal convergence <==

Convergence tolerances:
Function: 1.000000e-06
Parameters: 1.000000e-03

Objective function value at minimum: 0.005044

      parameter      search width
      0.874184      0.000020
      0.025758      0.000000
      0.305889      0.000041
      2.999958      0.000001

=====

SNM estimation results
      param      st. err      t
beta      0.874      0.157      5.583
exp(alpha/2)      0.026      0.002      14.908
sigma      0.306      0.207      1.478
octave:2>

```

- The other Monte Carlo results in the paper can be replicated using the “mc.m” files found in the other directories below the SNM directory. Some of those directories also contain scripts “estimate.m” that can be used to do SNM estimation using a sample drawn from the relevant DGP.

### 3 Usage and description

We will examine the file estimate.m from the directory /home/user/Econometrics/Examples/SNM/AR1. The contents appear in Listing 1. We now work through a description of what each line does, except for some lines that are obvious.

- line 1 checks that the code which is the heart of the data generating process (DGP) is compiled, and if not, compiles it. SNM is a computationally intensive estimation procedure, with simulation at its core. Using compiled code for the simulation is highly advisable if the simulation involves loops. GNU Octave does not have just-in-time compilation (yet), so loops are costly.
- line 3 sets the sample size

- line 4 set the name of the model, which is the DGP. The model must be an Octave function that follows the syntax  

```
[data, L, K, snm_draws] = modelname(theta, modelargs)
```

 where `theta` is a column vector of parameters to estimate, and `modelargs` is a cell array of other arguments. This is discussed further below.
- line 7 sets true parameter values to use for a simulation of estimation by SNM. The “real data” will be a sample of the size set in line 3, using these parameter values.
- line 10 sets the length of the simulated samples for SNM estimation, the “S” of CK2009, equation 2.5 and following.
- line 11 sets an additional number of simulation observations which will be discarded, to dampen the influence of initial conditions. The number of simulations actually used will be  $S$ , defined in line 10.
- line 12 allows one to do computations on multiple cores or on a cluster. If it is zero, a single core will be used. Higher values set the number of MPI ranks to use. Use of this feature requires a working installation of MPITB for Octave (Fernandez *et al.*, 2006). This is available if one uses PelicanHPC.
- line 13 set the number of simulations to use for estimation of the covariance matrix of the moment conditions (the  $R$  of CK2009, equation 2.19). The Jacobian of the moment conditions,  $H_0$  of CK2009 Assumption 6.1, is also estimated using a similar method.
- line 16 sets the arguments for the DGP. This is explained below.
- line 17 draws a sample from the DGP. This is the “real data” that will be used for estimation.  $L$  is the number of test variables, and  $K$  is the number of conditioning variables. The code for the DGP is explained below. The first  $L$  columns of data are the test variables, the next  $K$  columns are the conditioning variables. The remaining columns are the instruments.
- line 20 computes the number of instruments.
- lines 24-28 establish bound for the parameters, for use by the simulated annealing algorithm. First, the real parameters limits are set, then limits for the inverse of the window width are set.
- line 29 sets the start value for the minimization algorithm. It is set randomly to avoid bias toward any given value. This should not be a problem if a conservative cooling schedule is used for simulated annealing.
- lines 32-34 split up the data matrix.
- lines 37-38 get the data for OLS estimates, for comparison.

- line 41 calls the data preparation script for SNM. This script scales the test variables (endogs) and conditioning variables using a certain procedure, and returns the matrix  $P$ , so that simulated data can be scaled in the same way.
- line 44 assembles the scaled data into a single matrix. Having all the data in a single matrix makes parallelization using blocks of data relatively simple to do.
- lines 47 and 48 generate the random numbers to be used for the long simulation, of length  $S$ . These random numbers are passed as data, to keep them constant over iterations, to avoid “chatter”. The DGP should know how to generate the random draws it needs. More detail on the DGP follow below.
- line 49 now sets one of the arguments of the DGP to the matrix of random draws.
- line 50 sets the arguments for function that computes the SNM moment conditions. This is an important line. All use of SNM must define `momentargs` in the manner done here. `momentargs` must be an 8 element cell array. The first element is the name of the function that implements the DGP. This was set in line 4. The second element is a cell array that holds the arguments to the DGP, other than the parameters to be estimated. This was set in lines 47 and 49. Elements 3,4 and 5 are the numbers of test functions, conditioning variables, and instruments, respectively. These are needed so that the data matrix can be disaggregated into its parts. Element 6 is the matrix that scaled the data, which is needed so that the simulated data can be scaled in the same way. Element 7 is Boolean, and says whether or not to use conditional moments. Element 8 is Boolean, and says whether or not to use unconditional moments. CK2009 presents results using only conditional moments, but in some cases, results can be improved by including unconditional moments.
- lines 53-62 set controls for the simulated annealing minimizer.
- line 65 sets the weight matrix,  $W$ . This should either be a conformable positive definite matrix, or a scalar.
- line 66 overrides the random start defined above, and uses the OLS estimates as a start value. This is dangerous if BFGS minimization is used, because the SNM objective function has multiple local minima, and the algorithm may converge to a false minimum close to the start value. Use of SA with conservative cooling should avoid this pitfall. Because this is only an example, we will take the dangerous route here.
- lines 68-71 offer two options for minimization, BFGS or SA. As provided, the code uses BFGS.
- line 75 obtains the estimate of the covariance matrix of the moment conditions, using the CK2009 equation 2.19.

- line 78 obtains the estimate of the covariance matrix of SNM estimator of the model's parameter. This estimator is valid for heteroscedastic and or autocorrelated moment contributions, and allows for the use of an inefficient weight matrix.
- the rest of the code just prints out the results and compares to the OLS estimator.

```

1  if !(exist('./arl.oct','file')) system("mkoctfile arl.cc"); endif
2
3  n = 50; # number of obs
4  model = "arl_model"; # the DGP
5
6  # true parameters
7  theta = [0; 0.9];
8
9  # settings for estimation
10 S = 10000; # number of simulations
11 burnin = 100; # initial observations to discard
12 compute_nodes = 0; # number of nodes to use, not counting master
13 vc_reps = 100; # num. simulations to use in est. of cov. and Jacobian
   of moments
14
15 # make Monte Carlo data
16 modelargs = {"", n, burnin, true};
17 [data, L, K, snm_draws] = feval(model, theta, modelargs);
18
19 # define a few things
20 M = columns(data) - L - K;
21
22 # define bounds for simulated annealing
23 # bounds for parameters of the model
24 ub = [1; 1];
25 lb = [-1; 0];
26 # append bounds for inverse window width
27 ub = [ub; 3*ones(K,1)];
28 lb = [lb; 0.1*ones(K,1)];
29 thetastart = rand(size(ub)).*(ub-lb) + lb;
30
31 # split out parts of the data
32 endogs = data(:,1:L);
33 condvars = data(:,L+1:L+K);
34 instruments = data(:,L+K+1:L+K+M);
35
36 # define these for future OLS, for comparison
37 yy = endogs;
38 xx = [ones(n,1) condvars];
39
40 # prewhiten

```

```

41 [endogs, condvars, P] = snm_dataprep(endogs, condvars);
42
43 # re-assemble data
44 data = [endogs condvars instruments];
45
46 # define momentargs for GMM, including random draws for long simulation
47 modelargs = {"", S, burnin, false};
48 [junk, junk, junk, snm_draws] = feval(model, theta, modelargs);
49 modelargs{1} = snm_draws;
50 momentargs = {model, modelargs, L, K, M, P, true, false};
51
52 # samim controls
53 nt = 3;
54 ns = 3;
55 rt = .25;
56 maxevals = 1e4;
57 neps = 3;
58 functol = 1e-6;
59 paramtol = 1e-3;
60 verbosity = 2;
61 minarg = 1;
62 sacontrol = { lb, ub, nt, ns, rt, maxevals, neps, functol, paramtol,
63             verbosity, 1};
64
65 # first round consistent estimator
66 weight = 1; # weight matrix
67 thetastart = [ols(yy,xx); 1]; # good start values for BFGS (dangerous)
68 # uncomment one of the next 2 lines. First is BFGS (risky), second is
69 # SA (safe)
70 [thetahat, obj_value, conv1] = gmm_estimate(thetastart, data, weight,
71     ...
72     "snm_moments", momentargs, {100,2}, compute_nodes);
73 # [thetahat, obj_value, conv1] = gmm_estimate(thetastart, data, weight,
74     ...
75     "snm_moments", momentargs, sacontrol, compute_nodes);
76
77 # get covariance matrix of moment conditions
78 verbose = false; # output from cov matrix routine?
79 omega = snm_moment_covariance(thetahat, data, momentargs, vc_reps,
80     verbose);
81
82 # SNM covariance estimator
83 V = snm_variance(thetahat, data, weight, "snm_moments", momentargs,
84     vc_reps, ...
85     compute_nodes, omega);
86

```

```

81 # print results for SNM
82 thetahat = thetahat(1:2,:);
83 se = sqrt(diag(V));
84 t = thetahat ./ se;
85 results = [thetahat se t];
86 rlabels = char("const", "slope");
87 clabels = char("param", "st. err", "t");
88 printf("\n\nCheck that the final objective function value is very\n");
89 printf("close to zero, otherwise the SNM results are not valid. This
      script\n");
90 printf("uses BFGS rather than SA, for speed. To chang this, edit the\n"
      );
91 printf("script to uncomment the line for SA minimization\n");
92 printf("\nSNM estimation results\n");
93 prettyprint(results, rlabels, clabels);
94
95 # OLS for comparison
96 mc_ols(yy, xx);

```

Listing 1: estimate.m

Next we examine the code for the DPG, `ar1_model.m`, which was set in line 4 of `estimate.m`. This code should be quite self-explanatory. The syntax for a model must be of the form `[data, L, K, snm_draws] = modelname(theta, modelargs)`. The model must be able to return

- a data matrix
- $L$ , the number of test functions
- $K$ , the number of conditioning variables
- `snm_draws`, a matrix of random numbers that can be used to generate data.

Looking at the code line by line:

- In lines 7-9, we see that the model will generate the needed random draws if they are not received as an element of `modelargs`.
- line 16 calls the compiled code which generates data from the AR1 model. If the model can be written without loops, this might be omitted. For models that use loops, it is strongly advised to use compiled code. The provided `ar1.cc` and other `.cc` files in other directories serve as examples.
- lines 28-31 compute instruments, if asked to do so. This is useful if the model is to be used for Monte Carlo. For use with real data, this part would not be needed.

```

1 function [data, L, K, snm_draws] = ar1_model(theta, modelargs)
2   snm_draws = modelargs{1};
3   simlength = modelargs{2};
4   burnin = modelargs{3};
5   make_instruments = modelargs{4};
6   if ischar(snm_draws)
7     snm_draws = randn(simlength + burnin + 10, 2);
8     modelargs{1} = snm_draws;
9   endif
10
11  n = modelargs{2};
12  maxlag = 1;
13  n = n + maxlag;
14  modelargs{2} = n;
15
16  y = ar1(theta, modelargs);
17  data = [y lags(y, maxlag)];
18  data = data(maxlag + 1:n, :);
19  n = rows(data);
20
21  y = data(:, 1);
22  endogs = y;
23  condvars = data(:, 2);
24  data = [endogs condvars];
25  L = columns(endogs);
26  K = columns(condvars);
27
28  if make_instruments
29    instruments = [ones(n, 1) st_norm(condvars)];
30    data = [data instruments];
31  endif
32
33 endfunction

```

Listing 2: ar1\_model.m

## 4 Installation of SNM and support code

To install SNM and support code (minimization, MPITB, utilities, etc.), one should obtain the file `Econometrics.zip` that accompanies Creel (2003). This archive contains the SNM code and examples.

- It extracts to a folder `./Econometrics`.
- The SNM example code is in `./Econometrics/Examples/SNM`.

- The SNM estimation code is in `./Econometrics/MyOctaveFiles/Econometrics/SNM`.

To be able to use MPITB, one needs to install LAM/MPI ([www.lam-mpi.org](http://www.lam-mpi.org)), as well as GNU Octave ([www.octave.org](http://www.octave.org)). To set the Octave path, compile MPITB, and so forth, one can execute the script `./Econometrics/setup_econometrics`. Those who already use Octave should examine the script before running it, as it will overwrite the file `~/.octaverc` (leaving a backup).

## References

- [1] Creel, M. (2003). Econometrics, working paper and computer code, <http://econpapers.repec.org/paper/aubautbar/575.03.htm>.
- [2] Creel, M. (2008) PelicanHPC Tutorial, working paper, <http://econpapers.repec.org/paper/aubautbar/749.08.htm>.
- [3] Creel, M. (2009) PelicanHPC, GRECS computer code, <http://econpapers.repec.org/software/aubgrecss/005.09.htm>.
- [4] Creel, M. and D. Kristensen (2009) Estimation of Dynamic Latent Variable Models using Simulated Nonparametric Moments, working paper, <http://econpapers.repec.org/paper/aubautbar/792.09.htm>
- [5] Fernández, J., M. Anguita, E. Ros, J.L. Bernier (2006) SCE Toolboxes for the development of high-level parallel applications, 6th International Conference Computational Science - ICCS 2006, Reading, United Kingdom, May 28-31, 2006. Proceedings of the..., Part II, Lecture Notes in Computer Science, vol.3992, pp.518-52