Computational Economics and Econometrics

Case Study: The Habit Model

by

A. Ronald Gallant
Penn State University

Characteristics of Models of Specific Interest

- Likelihood not available.

- Prior information $\pi_1(\theta)$ on model parameters may be available.

- Prior information $\pi_2(\theta, \psi)$ on functionals of the model may be available, i.e., $\psi = \Psi(\mathcal{M}_\theta)$.

- Model can be simulated.

Example

- Habit persistence asset pricing model.

- Has these four characteristics:
  - Likelihood not available.
  - Prior information $\pi_1(\theta)$ on model parameters is available.
  - Prior information $\pi_2(\theta, \psi)$ on functionals is available.
  - Model can be simulated.

Habit Persistence Asset Pricing Model

Driving Processes

$$\text{Consumption: } c_t - c_{t-1} = g + v_t$$

$$\text{Dividends: } d_t - d_{t-1} = g + w_t$$

$$\text{Random shocks: } \begin{pmatrix} v_t \\ w_t \end{pmatrix} \sim \text{NID}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma^2 & \rho\sigma\sigma_w \\ \rho\sigma\sigma_w & \sigma_w^2 \end{pmatrix}\right]$$

The time increment is one month. Lower case denotes logarithms of upper case quantities; i.e. $c_t = \log(C_t)$, $d_t = \log(D_t)$. From Campbell and Cochrane (1999).

Habit Persistence Asset Pricing Model

Utility function

$$\mathcal{E}_0\left(\sum_{t=0}^{\infty}\delta^t\frac{(S_tC_t)^{1-\gamma}-1}{1-\gamma}\right),$$

Habit persistence

$$\text{Surplus ratio: } s_t-\bar{s}=\phi(s_{t-1}-\bar{s})+\lambda(s_{t-1})v_t$$

$$\text{Sensitivity function: } \lambda(s)=\begin{cases}\frac{1}{\bar{S}}\sqrt{1-2(s-\bar{s})}-1 & s_t\leq s_{\max}\\ 0 & s_t>s_{\max}\end{cases}$$

$\mathcal{E}_t$ is conditional expectation with respect to S_t, S_{t-1}, ... Lower case denotes logarithms of upper case quantities: $s_t=\log(S_t)$. $\bar{S}$ and $s_{\max}$ can be computed from model parameters $\theta=(g,\sigma,\rho,\sigma_w,\phi,\delta,\gamma)$ as $\bar{S}=\sigma\sqrt{\frac{\gamma}{1-\phi}}$ and $s_{\max}=\bar{s}+\frac{1}{2}(1-\bar{S}^2)$. From Campbell and Cochrane (1999).

Utility Function

Campbell and Cochrane write habit persistence utility function as

$$\mathcal{E}_0\left(\sum_{t=0}^{\infty} \delta^t \frac{(C_t - X_t)^{1-\gamma} - 1}{1-\gamma}\right),$$

where X_t is habit.

They introduce the surplus ratio $S_t = (C_t - X_t)/C_t$ much later in the development.

The surplus ratio form is more revealing of how the habit model generates interesting returns; i.e., it changes the consumption process from C_t, which is tame, to C_tS_t, which is volatile.

Simulating Driving Processes and the State

So far so good, C_t, D_t, and S_t are easy to simulate.

We generate a long simulation of consumption, dividends, and surplus ratio in both logs and levels

$$\{c_t\}_{t=1}^{N} \qquad \{C_t\}_{t=1}^{N} \qquad N \sim 50,000$$

$$\{s_t\}_{t=1}^{N} \qquad \{S_t\}_{t=1}^{N} \qquad N \sim 50,000$$

$$\{d_t\}_{t=1}^{N} \qquad \{D_t\}_{t=1}^{N} \qquad N \sim 50,000$$

Simulation

Go over model parameters, model variables, and class habit in usrmod.h

Go over make_state in usrmod.cpp.

Returns Processes

Now comes the hard part: computing returns.

The agent desires to buy and sell assets to transfer consumption from one period to another. We must solve the agent's optimization problem to get the returns process this desire generates.

Habit Persistence Asset Pricing Model

Return on dividends

$$V(S_t) = \mathcal{E}_t \left\{ \delta \left(\frac{S_{t+1} C_{t+1}}{S_t C_t} \right)^{-\gamma} \left(\frac{D_{t+1}}{D_t} \right) \left[1 + V(S_{t+1}) \right] \right\}$$

$$r_{dt} = \log \left[\frac{1 + V(S_t)}{V(S_{t-1})} \left(\frac{D_t}{D_{t-1}} \right) \right]$$

$V(\cdot)$ is defined as the solution of the Euler condition above. It is the price dividend ratio; i.e. $P_{dt}/D_t = V(S_t)$, where P_{dt} is the price of the asset that pays the dividend stream. r_{dt} is the logarithmic real return, i.e. $r_{dt} = \log(P_{dt} + D_t) - \log(P_{d,t-1})$, where P_{dt} and D_t are measured in real (inflation adjusted) dollars. From Campbell and Cochrane (1999).

Solution Method - 1

The computational problem is this: We must find the policy function $V(\cdot)$ that solves

$$V(S_t) = \mathcal{E}_t \left\{ \delta \left(\frac{S_{t+1} C_{t+1}}{S_t C_t} \right)^{-\gamma} \left(\frac{D_{t+1}}{D_t} \right) \left[1 + V(S_{t+1}) \right] \right\}$$

and then evaluate $V(\cdot)$ over our simulated values $\{C_t, D_t, S_t\}_{t=1}^N$ to get the corresponding returns process $\{r_{dt}\}_{t=1}^N$ using the formula

$$r_{dt} = \log \left[\frac{1 + V(S_t)}{V(S_{t-1})} \left(\frac{D_t}{D_{t-1}} \right) \right]$$

Solution Method - 2

Campbell and Cochrane (1999) posit that the log policy function

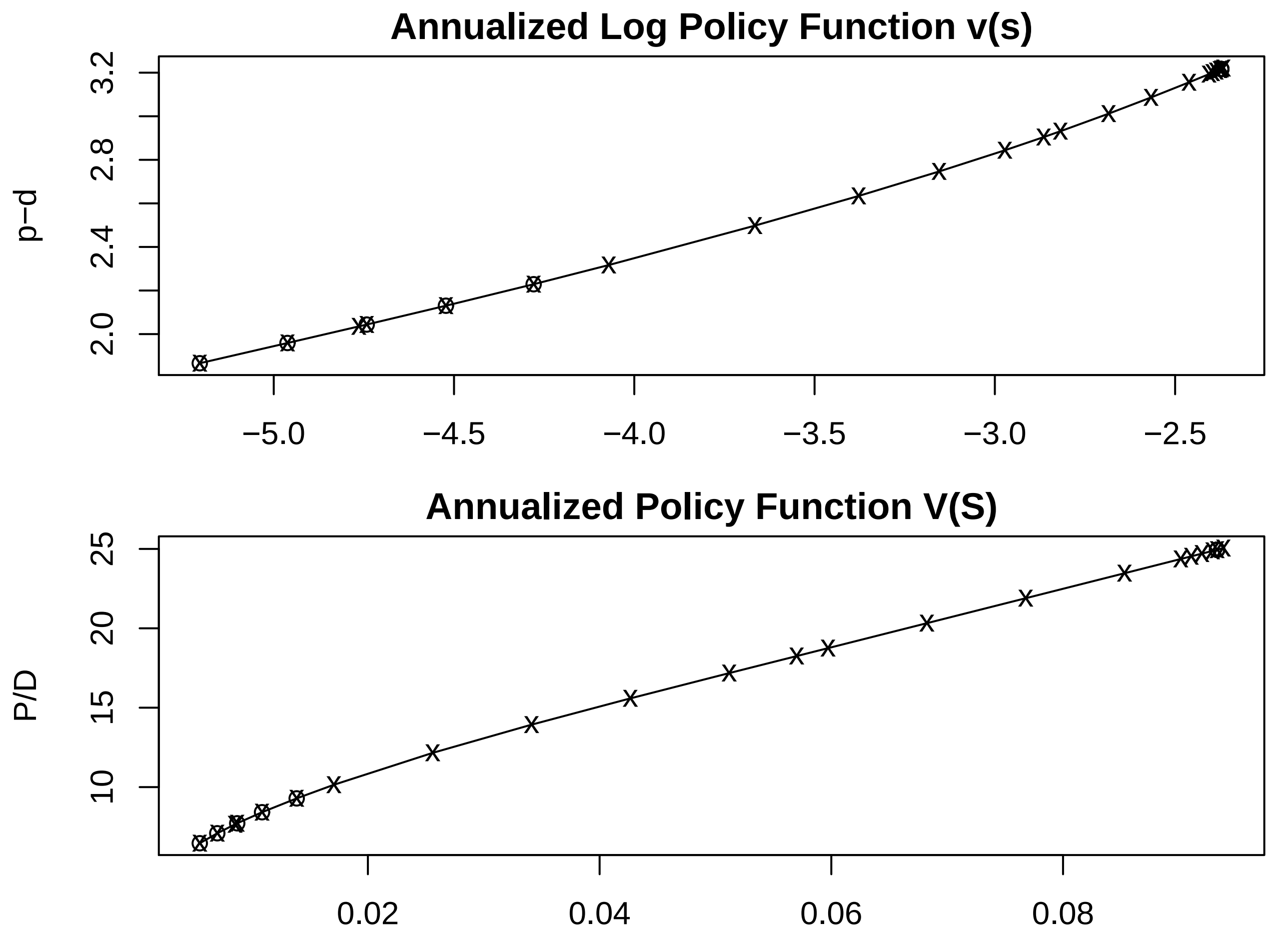
$$v(s_t) = \log V(e^{s_t})$$

can be represented as a piecewise linear function.

Their join points are $\bar{s}$, s_{max}, $s_{\text{max}} - 0.01$, $s_{\text{max}} - 0.02$, $s_{\text{max}} - 0.03$, $s_{\text{max}} - 0.04$, and $\log[iS/(m+1)]$ for $i = 1, \ldots, m = 10$. My changes: Used max of the simulated s_t if larger than s_{max}. Added the abscissae of the Gauss-Hermite quadrature formula for integrating at the maximum and minimum of the above join points. Deleted all points closer than 0.001.

Figure 1, next slide, plots the approximation at the Campbell and Cochrane parameter values.

Fig 1. Piecewise Linear Approximation



x's mark Campbell and Cochrane join points; o's mark extra join points from the quadrature rule.

Implementing a Piecewise Linear Function - 1

```
class linear_function {
private:
  REAL a;
  REAL b;
  REAL x0;
public:
  void initialize(REAL intercept, REAL slope, REAL origin)
    { a = intercept; b = slope; x0 = origin; }
  REAL operator()(REAL x) { return a + b*(x - x0); }
  REAL intercept() { return a; }
  REAL slope() { return b; }
  REAL origin() { return x0; }
};
```

Implementing a Piecewise Linear Function - 2

```
class linear_interpolater {
private:
  std::vector<linear_function> funcs;
  typedef std::vector<linear_function>::size_type lfst;
  REAL xmin;
  REAL xmax;
  lfst N;
  lfst hash(REAL x) { return lfst( REAL(N-2)*(x-xmin)/(xmax-xmin) ); }
public:
  linear_interpolater()
  {
    scl::realmat grid(2,1) ; grid[1] = 0.0; grid[2] = 1.0;
    scl::realmat vals(2,1) ; vals[1] = 0.0; vals[2] = 1.0;
    update(grid,vals);
  }
  linear_interpolater(scl::realmat& x, scl::realmat& y) { update(x,y); };
  REAL operator()(REAL x)
  {
     if (x <= funcs[0].origin()) return funcs[0](x);
     if (x >= funcs[N-1].origin()) return funcs[N-1](x);
     lfst i = hash(x);
     if (x < funcs[i].origin()) while(x < funcs[--i].origin());
     else if (x >= funcs[i+1].origin()) while(x >= funcs[++i+1].origin());
     return funcs[i](x);
   }
```

Implementing a Piecewise Linear Function - 3

```
  void update(scl::realmat& x, scl::realmat& y)
  {
    INTEGER n = x.size(); N = lfst(n);
    funcs.clear(); funcs.reserve(N);
    if (n<2)
      scl::error("Error, linear_interpolater, x.size() < 2");
    if (x.ncol() != 1 || y.ncol() != 1)
      scl::error("Error, linear_interpolater, x or y not a vector");
    if (n != y.size())
      scl::error("Error, linear_interpolater, x and y sizes differ");
    scl::intvec permutation_index = x.sort();
    y = y(permutation_index,"");
    xmin = x[1]; xmax = x[n];
    linear_function f;
    for (INTEGER i=1; i<n; ++i) {
      f.initialize(y[i], (y[i+1]-y[i])/(x[i+1]-x[i]), x[i]);
      funcs.push_back(f);
    }
    funcs.push_back(f);
  }
};
```

Implementing a Piecewise Linear Function - 4

To get the linear interpolater $v(s)$ plotted in the upper panel of Figure 1, one would fill the `realmat grid` with the abscissae of the points marked with x's and o's and fill a `realmat values` with the ordinates. Then

```
linear_interpolater v();
v.update(grid, values);
```

will be the policy function $v(s)$. The calling syntax is

```
REAL log_surplus_ratio = -4.0;
REAL log_stock_price_dividend_ratio
          = v(log_surplus_ratio);
```

Solution Method - 3

Putting everything in logs, the conditional Euler condition is

$$e^{v(s_t)} = \mathcal{E}_t\Big\{\delta e^{-\gamma(\Delta s_{t+1}+\Delta c_{t+1})} e^{\Delta d_{t+1}} (1 + e^{v(s_{t+1})})\Big\}$$

where $\Delta s_{t+1} = s_{t+1} - s_t$, etc. This is a contraction mapping so we can compute $v(s)$ by iterating the equation above.

Specifically, start the linear_interpolater either at $v^0(s)$ of Figure 1 (better) or at $v^0(s) \equiv 0$. For $i = 0$, compute

$$e^{v^{i+1}(s_t)} = \mathcal{E}_t\Big\{\delta e^{-\gamma(\Delta s_{t+1}+\Delta c_{t+1})} e^{\Delta d_{t+1}} (1 + e^{v^i(s_{t+1})})\Big\}$$

at each of the points s_t in `realmat grid` of the previous slide. Put the corresponding $v^{i+1}(s_t) = \log e^{v^{i+1}(s_t)}$ in `realmat values`. Call

```
v.update(grid,values);
```

which overwrites $v^i(s)$ by $v^{i+1}(s)$. Continue for $i = 1, 2, \ldots$.

Solution Method - 4

What remains is to compute the integral

$$\mathcal{E}_t \Big\{ \delta e^{-\gamma(\Delta s_{t+1} + \Delta c_{t+1})} e^{\Delta d_{t+1}} (1 + e^{v(s_{t+1})}) \Big\}$$

where

$$\begin{aligned} \Delta s_{t+1} &= (1-\phi)\bar{s} + (\phi - 1)s_t + \lambda(s_t)v_{t+1} \\ \Delta c_{t+1} &= g + v_{t+1} \\ \Delta d_{t+1} &= g + w_{t+1} \end{aligned}$$

We can integrate out w_{t+1} analytically to get

$$e^{g + \frac{1}{2}(1-\rho^2)\sigma_w^2} \; \mathcal{E}_t \Big\{ \delta e^{-\gamma(\Delta s_{t+1} + \Delta c_{t+1})} e^{\rho(\sigma_w/\sigma)v_{t+1}} (1 + e^{v(s_{t+1})}) \Big\}$$

We will have to integrate out v_{t+1} numerically.

Sorry that v can mean either an error v_t or a policy function $v(s)$.

Gaussian Quadrature - 1

A Gaussian quadrature formula has the form

$$\int_a^b f(x)W(x)\,dx \approx \sum_{i=1}^{n} f(x_i)w_i.$$

The theory of the subject is devoted to how best to choose the abscissae x_i and weights w_i.

Names such as Gauss-Laguerre or Gauss-Hermite indicate what a, b, and $W(x)$ are. For instance, for Gauss-Laguerre $a = 0$, $b = \infty$, and $W(x) = e^{-x}$; for Gauss-Hermite $a = -\infty$, $b = \infty$, and $W(x) = e^{-x^2}$.

Gaussian Quadrature - 2

Construction:

1. Find coefficients for the polynomials $p_k(x) = a_{k0} + a_{k1}x + \cdots + a_{kk}x^k$, for $k = 1, \ldots, n$, such that $\int p_k(x)p_j(x)W(x)\,dx = 1$ if $k = j$ and 0 if $k \neq j$; this is not hard.

2. Find the zeros of the polynomial p_n; this is hard.

   - Golub, Gene H., John H. Welsch (1969), "Calculation of Gauss Quadrature Rules", *Mathematics of Computation* 23, 221–230

3. The zeros are the abscissae x_i for the rule.

4. Find the w_i such that $\sum_{i=1}^{n} p_0(x_i)w_i = 1$ and $\sum_{i=1}^{n} p_k(x_i)w_i = 0$; this is not hard.

This construction has the advantage that $p_m(x)$ will be integrated exactly by an n-point rule for all $m < 2n$.

The function `hquad` in `libscl` computes Gauss-Hermite rules. The function `guassq` computes just about every rule there is.

Gaussian Quadrature - 3

Using the change of variables

$$x = \frac{1}{\sqrt{2}} \left(\frac{u - \mu}{\sigma} \right) \qquad dx = \frac{1}{\sqrt{2}\sigma} du$$

we get

$$\begin{aligned} & \int_{-\infty}^{\infty} f(u) \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{u-\mu}{\sigma}\right)^2} du \\ = & \int_{-\infty}^{\infty} f(\mu + \sqrt{2}\sigma x) \frac{1}{\sqrt{\pi}} e^{-x^2} dx \\ \approx & \sum_{i=1}^{n} f(\mu + \sqrt{2}\sigma x_i) \frac{w_i}{\sqrt{\pi}} \end{aligned}$$

Thus, the abscissae and weights for $\mathcal{E} f(U)$ when $U \sim N(\mu, \sigma^2)$ are $x_i^* = \mu + \sqrt{2}\sigma x_i$ and $w_i^* = w_i / \sqrt{\pi}$, where x_i and w_i are the Gauss-Hermite abscissae and weights.

Habit Persistence Asset Pricing Model

Risk Free Rate

$$r_{ft} = -\log\left\{\mathcal{E}_t\left[\delta\left(\frac{S_{t+1}C_{t+1}}{S_tC_t}\right)^{-\gamma}\right]\right\}$$

r_{ft} is the logarithmic return on an asset that pays one real dollar one month hence with certainty. From Campbell and Cochrane (1999).

Solution method is similar to the foregoing.

Model Output

For given model parameters

$$\theta = (g, \sigma, \rho, \sigma_w, \phi, \delta, \gamma)$$

the model produces simulated consumption and returns data at an annual frequency:

$$C_t^a = \sum_{k=0}^{11} C_{12t-k}$$

$$c_t^a = \log(C_t^a)$$

$$r_{dt}^a = \sum_{k=0}^{11} r_{d,12t-k}$$

$$r_{ft}^a = \sum_{k=0}^{11} r_{f,12t-k}$$

Putting It All Together

Go over model parameters, model variables, and class habit in habit_usrmod.h

Go over gen_sim in habit_usrmod.cpp.

Data

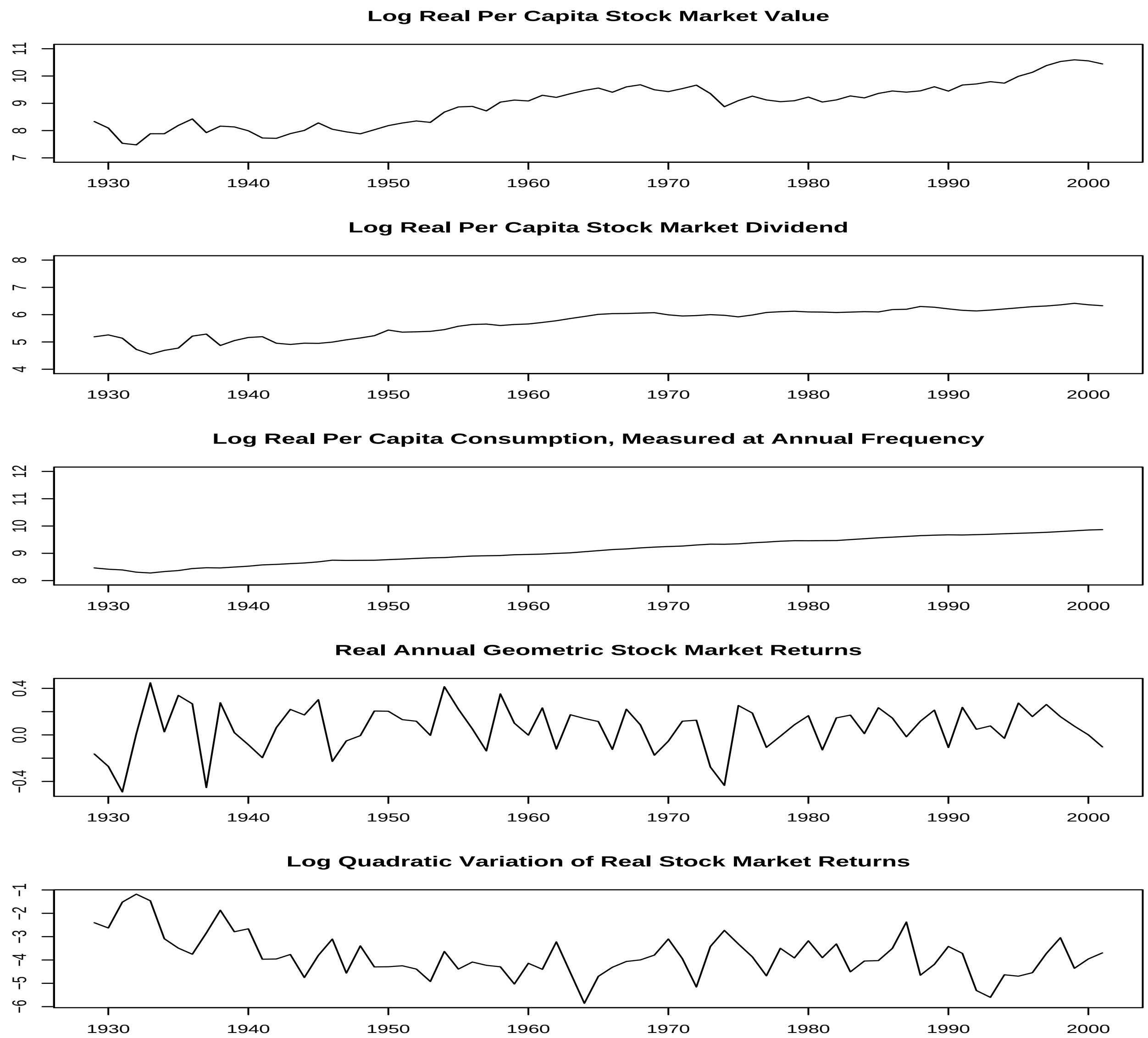
Annual observations 1929–2001, 72 years, on

P^a_{dt}	end-of-year per capita stock market value
D^a_t	annual aggregate per capita dividend
C^a_t	annual per capita consumption
r^a_{dt}	annual real geometric return
Q^a_t	annual quadratic variation

Data are real, i.e. inflation adjusted.

Source: Bansal, R., A. R. Gallant, and G. Tauchen (2007). "Rational Pessimism, Rational Exuberance, and Markets for Macro Risks," *Review of Economic Studies* 74, 1005–1033.

Fig 2. Data


Log Real Per Capita Stock Market Value
Log Real Per Capita Stock Market Dividend
Log Real Per Capita Consumption, Measured at Annual Frequency
Real Annual Geometric Stock Market Returns
Log Quadratic Variation of Real Stock Market Returns
1930
1940
1950
1960
1970
1980
1990
2000

Cointegrating Relationships

$$
\begin{aligned}
p_{dt}^{a} - d_{t}^{a} &= I(0) \quad \text{Well documented in the literature} \\
d_{t}^{a} - c_{t}^{a} &= I(0) \quad \text{Bansal, Gallant, Tauchen (2007)} \\
c_{t}^{a} - c_{t-1}^{a} &= I(0) \quad \text{Well documented in the literature}
\end{aligned}
$$

Jointly Stationary Data for Estimation

Used by Gallant and McCulloch (2009) and in case study:

$$\begin{pmatrix} c_t^a - c_{t-1}^a \\ r_{dt}^a \end{pmatrix}$$

Used by Bansal, Gallant, and Tauchen (2007):

$$\begin{pmatrix} d_t^a - c_t^a \\ c_t^a - c_{t-1}^a \\ p_{dt}^a - d_t^a \\ r_{dt}^a \end{pmatrix}$$

Unconditional Moments of Annual Data

		Mean	Std Dev
Log dividend consumption ratio	$d_t^a - c_t^a$	-3.399	0.162
Consumption growth(% Per Year)	$100 \times (c_t^a - c_{t-12}^a)$	1.95	2.24
Price dividend ratio	$\exp(v_{dt}^a)$	28.24	12.08
Return(% Per Year), dividend	$100 \times r_{dt}^a$	6.02	19.29
$100 \times \sqrt{\text{Quadratic variation}}$	$100 \times \text{std}_t^a$	16.69	09.32

Characteristics of the Monthly Data

Quantiles	c_t/c_{t-1}	r_t^e
99%	1.013425	1.121526
95%	1.009645	1.057568
90%	1.007642	1.048265
75% Q3	1.004783	1.030098
50% Med	1.002235	1.005685
25% Q1	0.999164	0.978847
10%	0.996466	0.948152
5%	0.994561	0.932595
1%	0.991563	0.890846
IQR Q3-Q1	0.005619	0.051251
Mean	1.002060	1.002851
Std. Dev.	0.004573	0.042095
Annualized		
Med	2.715	7.039
IQR	1.946	17.754
Mean	2.500	3.475
Std. Dev.	1.584	14.582

Notes: The sampling frequency is monthly. 1959–1978. Med is the median and IQR is the inter quartile range. From Hansen and Singleton (1982).

Prior Information

Support: Reasonable bounds on all parameters to include positivity restrictions on positive valued parameters and non-explosive restrictions on autoregressive parameters.

Numerical: Existence of solution to Euler condition.

Used by methods proposed here (annualized, iid normal prior)

$$P\left(\left|\mathcal{E}(r_f^a) - 0.89\%\right| < 1\%\right) = 0.95$$

$$P\left(|\rho - 0.2| < 0.1\right) = 0.95$$

$$P\left(|\phi - 0.9884| < 0.01\right) = 0.95$$

Used by estimates compared with (annualized, uniform prior)

$$P\left(\left|\mathcal{E}(r_f^a) - 0.89\%\right| < 0.5\%\right) = 1.00$$

Prior Information Grouped by Cost

1. Support condition can be determined cheaply knowing model parameters θ alone

$$\pi_1(\theta) \qquad \theta = (g, \sigma, \rho, \sigma_w, \phi, \delta, \gamma)$$

2. Simulation failure is a function of θ only but is costly to determine.

$$\pi_2(\theta) \qquad \theta = (g, \sigma, \rho, \sigma_w, \phi, \delta, \gamma)$$

3. Requires a simulation to determine

$$\pi_3(\theta, \psi) \qquad \psi = \left(\mathcal{E}(r^a_{ft}), \rho, \phi)\right)$$

The difference in cost of these three sources of prior information will be taken into account in designing computational strategies.

Estimation Options Available

- Asymptotic Equivalent of MLE
  Gallant and Tauchen (2001)

- Bayesian with Synthesized Likelihood
  Gallant and McCulloch (2009)

- Simulated Method of Moments
  Duffie and Singleton (1993)

- Bayesian GMM
  Gallant (2015)

Cites are to the most closely related papers. They are not attributions.

SMM with GMM Criterion

We will illustrate the ideas using SMM with a GMM criterion.

- The GMM objective function is denoted by $s_n(\theta)$.

- Output and parameters of the habit persistence asset pricing model are

$$\widehat{y}_t = (c_t^a - c_{t-1}^a,\ r_{dt}^a) \in \Re^2$$

$$\theta = (g, \sigma, \rho, \sigma_w, \phi, \delta, \gamma) \in \Re^7$$

- Data are denoted as $\{\tilde{y}\}_{t=1}^n$, simulations as $\{\widehat{y}\}_{t=1}^N$.

GMM Criterion – Notation

$$\bar{y} = \frac{1}{n} \sum_{t=1}^{n} \begin{pmatrix} y_t \\ y_{t-1} \end{pmatrix}$$

$$S_t = \left[\begin{pmatrix} y_t \\ y_{t-1} \end{pmatrix} - \bar{y} \right] \left[\begin{pmatrix} y_t \\ y_{t-1} \end{pmatrix} - \bar{y} \right]'$$

$$m_t = \begin{pmatrix} y_t \\ y_{t-1} \\ \text{vech}(S_t) \end{pmatrix}$$

$\tilde{m}_t$ denotes evaluation at data

$\hat{m}_t$ denotes evaluation at a simulation

GMM Criterion – Moment Functions

Moment function for data:

$$\tilde{m}_n = \frac{1}{n} \sum_{t=1}^{n} \tilde{m}_t$$

Moment function for a simulation:

$$\hat{m}_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \hat{m}_t$$

GMM Cross Sectional Weight Function

$\tilde{W}_n$ is an estimate of the variance of $\sqrt{n}\,\tilde{m}_n$

$$\tilde{W}_n = \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{m}_i - \tilde{m}_n\right)\left(\tilde{m}_i - \tilde{m}_n\right)'$$

GMM Time Series Weight Function

$\tilde{W}_n$ is an estimate of the variance of $\sqrt{n}\,\tilde{m}_n$

$$\tilde{W}_n = \sum_{\tau=-[n^{1/5}]}^{[n^{1/5}]} w\left(\frac{\tau}{[n^{1/5}]}\right)\tilde{W}_{n\tau}$$

where

$$w(u) = \begin{cases} 1 - 6|u|^2 + 6|u|^3 & \text{if } 0 < u < \frac{1}{2} \\ 2(1 - |u|)^3 & \text{if } \frac{1}{2} \le u < 1 \end{cases}$$

$$\tilde{W}_{n\tau} = \begin{cases} \frac{1}{n}\sum_{t=1+\tau}^{n} (\tilde{m}_t - \tilde{m}_n)(\tilde{m}_{t-\tau} - \tilde{m}_n)' & \tau \ge 0 \\ \tilde{W}'_{n,-\tau} & \tau < 0 \end{cases}$$

GMM Criterion Function

$$s_n(\theta) = \frac{1}{2} \left[\tilde{m}_n - \widehat{m}_N(\theta)\right]' \left(\tilde{W}_n\right)^{-1} \left[\tilde{m}_n - \widehat{m}_N(\theta)\right]$$

Inference Styles

Frequentist The estimator is $\hat{\theta}_n = \text{argmin}_\theta \, s_n(\theta)$; equivalently one can put $\ell(\theta) = e^{-n\, s_n(\theta)}$ and compute $\text{argmax}_\theta \, \ell(\theta)$. In frequentist inference one would usually take support conditions into account and compute $\text{argmax}_\theta \, \ell(\theta)\pi_1(\theta)\pi_2(\theta)$. Because $\ell(\theta)$ will increase with n and $\pi_3(\theta, \psi)$ will not, the asymptotics would not change if one also multiplied by $\pi_3(\theta, \psi)$. This is easier to see by taking logs.

Bayesian $\ell(\theta) = e^{-n\, s_n(\theta)}$ is an acceptable likelihood for Bayesian inference (Gallant, 2015). $\pi_1(\theta)\pi_2(\theta)\pi_3(\theta, \psi)$ is an acceptable prior for Bayesian inference (Gallant and McCulloch, 2009). Strictly speaking, one should use the continuously updated version of $s_n(\theta)$ here. I.e. compute the weighting matrix from $\hat{m}_t$ from the simulation rather than $\tilde{m}_t$ from the data.

Asymptotics

Under weak regularity conditions that accommodate both time series and cross sectional data (Gallant, 1987) $\hat{\theta}_n$ tends to the parameter value θ^o that minimizes

$$s^o(\theta) = \lim_{n\to\infty} s_n(\theta)$$

and $\sqrt{n}(\hat{\theta}_n - \theta^o)$ is asymptotically normal with mean zero and variance $\mathcal{J}^{-1}\mathcal{I}\mathcal{J}^{-1}$, where $\mathcal{J}$ is the Hessian

$$\mathcal{J} = \frac{\partial}{\partial\theta\partial\theta'}\, s^o(\theta^o)$$

and $\mathcal{I}$ is Fisher's information

$$\mathcal{I} = \text{Var}\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right] = \mathcal{E}\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right]\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right]'$$

In some cases $\mathcal{I} = \mathcal{J}$ so that only one of the two has to be computed; e.g. correctly specified mle or GMM with correct weight matrix.

Computations

For $\quad s_n(\theta) = \frac{1}{2}\left[\tilde{m}_n - \widehat{m}_N(\theta)\right]' \left(\tilde{W}_n\right)^{-1} \left[\tilde{m}_n - \widehat{m}_N(\theta)\right]$

- must compute the estimator

$$\widehat{\theta}_n = \underset{\theta}{\text{argmin}}\; s_n(\theta)$$

- an estimate of the Hessian

$$\mathcal{J} = \frac{\partial}{\partial\theta\partial\theta'}\, s^o(\theta)$$

- an estimate of the information

$$\mathcal{I} = \text{Var}\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right] = \mathcal{E}\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right]\left[\frac{\partial}{\partial\theta'}\sqrt{n}\, s_n(\theta^o)\right]'$$

- and an estimate of the variance of $\sqrt{n}(\widehat{\theta}_n - \theta^o)$

$$V_n = \text{Var}\left[\sqrt{n}(\widehat{\theta}_n - \theta^o)\right] = \mathcal{J}^{-1}\mathcal{I}\mathcal{J}^{-1}$$

Computational Strategy – $\hat{\theta}$ & $\hat{\mathcal{J}}^{-1}$

- Chernozhukov, Victor, and Han Hong (2003), "An MCMC Approach to Classical Estimation," *Journal of Econometrics* 115, 293–346.

- Put $\ell(\theta) = e^{-n\, s_n(\theta)}$. Apply Bayesian MCMC methods with $\ell(\theta)$ as the likelihood and $\pi(\theta,\psi) = \pi_1(\theta)\pi_2(\theta)\pi_3(\theta,\psi)$ as the prior.

- From the resulting MCMC chain $\{\theta_i\}_{i=1}^{R}$, put

$$\hat{\theta}_n = \underset{\theta_i}{\text{argmax}}\, \ell(\theta_i)\pi(\theta_i,\psi^i) \text{ or } \hat{\theta}_n = \bar{\theta}_R = \frac{1}{R}\sum_{t=1}^{R}\theta_i$$

i.e. the mode or the mean, and put

$$\hat{\mathcal{J}}^{-1} = \left(\frac{n}{R}\right)\sum_{t=1}^{R}\left(\theta_i - \bar{\theta}_R\right)\left(\theta_i - \bar{\theta}_R\right)'$$

Metropolis-Hastings MCMC Chain

Proposal density: $T(\theta_{here}, \theta_{there})$

Proposal: θ_{prop} drawn from $T(\theta_{old}, \theta)$

Simulate: Get $s_n(\theta_{prop})$, ψ_{prop}, and $\pi(\theta_{prop}, \psi_{prop})$

Likelihood: Put $\ell(\theta) = e^{-n\, s_n(\theta)}$

Put θ_{new} to θ_{prop} with probability

$$\alpha = \min\left[1, \frac{\pi(\theta_{prop}, \psi_{prop})\ell(\theta_{prop})T(\theta_{prop}, \theta_{old})}{\pi(\theta_{old}, \psi_{old})\ell(\theta_{old})T(\theta_{old}, \theta_{prop})}\right]$$

Put θ_{new} to θ_{old} with probability $1 - \alpha$.

Why Does This Work?

Let x be the old and y the new and let $f(\cdot)$ be the product of the prior and the likelihood of the previous slide. The proposal density is $T(x, y)$ and the transition density determined by the chain is

$$A(x, y) = T(x, y) \min \left\{ 1, \frac{f(y)T(y, x)}{f(x)T(x, y)} \right\}$$

for $y \neq x$ and

$$A(x, x) = 1 - \int I(x, y) \, A(x, y) \, dy,$$

where

$$I(x, y) = \begin{cases} 1 & y \neq x \\ 0 & y = x \end{cases}$$

Detailed Balance

For $x \neq y$

$$f(x)A(x,y) = \min\{f(x)T(x,y), f(y)T(y,x)\}$$

which implies that $f(x)A(x,y)$ is symmetric, i.e. that

$$f(y)A(y,x) = f(x)A(x,y).$$

Symmetry holds trivially for $x = y$.

This symmetry condition is called the detailed balance condition and implies, among other things, that the chain defined by $A(x,y)$ is reversible.

Conditional Expectation

Let

$$I(x,y) = \begin{cases} 1 & y \neq x \\ 0 & y = x \end{cases}$$

Then

$$\mathcal{E}[g(Y)|x] = \int g(y) I(x,y) A(x,y)\, dy + g(x) A(x,x)$$

Unconditional Expectation

$$\begin{aligned}
&\int \mathcal{E}[g(Y)|x] f(x)\, dx \\
&\quad = \iint g(y) I(x,y) A(x,y) f(x) dx dy + \int g(x) A(x,x) f(x) dx \\
&\quad = \iint g(y) I(x,y) A(y,x) f(y) dx dy + \int g(x) A(x,x) f(x) dx \\
&\quad = \int g(y) f(y) \int I(x,y) A(y,x) dx dy + \int g(x) A(x,x) f(x) dx \\
&\quad = \int g(y) f(y) [1 - A(y,y)]\, dy + \int g(x) A(x,x) f(x)\, dx \\
&\quad = \int g(y) f(y)\, dy
\end{aligned}$$

Stationary Density of the Chain

The fact that the equation

$$\int \mathcal{E}[g(Y)|x] f(x)\, dx = \int g(y) f(y)\, dy$$

holds for all integrable $g(y)$ implies that $f(y)$ is the stationary density of the MCMC chain with transition density $A(x, y)$.

Computational Strategy – $\widehat{\mathcal{I}}$

- For θ set to $\hat{\theta}_n$, simulate the model and generate I independent data sets $\{\hat{y}_{t,i}\}_{t=1}^{n}$, $i = 1, \ldots, I$, each of exactly the same size n of the original data.

- Let $\hat{s}_{n,i}(\theta)$ denote the criterion function corresponding to data set $\{\hat{y}_{t,i}\}_{t=1}^{n}$. (Store in C++ STL vector indexed by i.)

- Compute $\frac{\partial}{\partial\theta'}\sqrt{n}\,\hat{s}_{n,i}(\hat{\theta}_n)$.

- An estimate of the information is

$$\widehat{\mathcal{I}} = \frac{1}{I}\sum_{i=1}^{I}\left[\frac{\partial}{\partial\theta'}\sqrt{n}\,\hat{s}_{n,i}(\hat{\theta}_n)\right]\left[\frac{\partial}{\partial\theta'}\sqrt{n}\,\hat{s}_{n,i}(\hat{\theta}_n)\right]'$$

EMM Enhancements

Nearly all of the computational cost of the MCMC chain is due to solving the asset pricing equations and computing the criterion function $s_n(\theta)$. This cost can be minimized as follows:

- Reject immediately if $\pi_1(\theta) = 0$.
- Put θ on a grid. Grid increments determined by sensitivity of $\{\hat{y}_t\}_{t=1}^{N}$ to θ elements. E.g. 0.001 for g and δ, and 0.5 for γ.
- Store $s_n(\theta)$, ψ, $\pi_2(\theta)$, $\pi_3(\theta, \psi)$ in a C++ STL associative map indexed by θ.
- Use table lookup to avoid all recomputation.
- The longer the chain, the faster it runs.

The EMM code does all of this; the case study the first only.

Comments

- $S_n(\theta) = \tau s_n(\theta)$ is a valid criterion according to the theory. This gives one a temperature parameter τ to use for tuning the chain. It can be used to adjust the relative importance of the prior and to scale proposal increments.

- It would have been better to write the per parameter rejection rates to a file rather than just the overall. The EMM code does this. However, looking at plots of the chain is the best approach.

Comments

- It would have been better to write the likelihood, the prior, and the posterior to a file rather than just the posterior. The EMM code does this.

- The justification for using a prior and Bayesian methods with the GMM criterion function is in Gallant, A. Ronald (2015), "Reflections on the Probability Space Induced by Moment Conditions with Implications for Bayesian Inference," *Journal of Financial Econometrics*, forthcoming.

Comments

- In the case study, `objfun` returns $ns_n(\theta)$ not $s_n(\theta)$.

- Similarly, in the EMM code, `objfun` returns $ns_n(\theta)$ not $s_n(\theta)$.

Computational Strategy – EMM MCMC

1. Propose: Draw θ_{prop} from $T(\theta_{old}, \theta)$.

2. Check support: Check $\pi_1(\theta)$. If $\pi_1(\theta) = 0$, then put θ_{new} to θ_{old}. Go to 1.

3. Check map: If θ_{prop} in map, α can be computed cheaply. Put θ_{new} to θ_{prop} with probability α. Put θ_{new} to θ_{old} with probability $1 - \alpha$. Go to 1.

4. Simulate: Check $\pi_2(\theta)$. If $\pi_2(\theta) = 0$, then add results to map, put θ_{new} to θ_{old}, and go to 1.

5. Evaluate: $s_n(\theta_{prop})$, ψ_{prop}, $\pi(\theta_{prop}, \psi_{prop})$ and put in map. Compute α. Put θ_{new} to θ_{prop} with probability α. Put θ_{new} to θ_{old} with probability $1 - \alpha$. Go to 1.

Simple Example, Simulated Data

Before applying the code to the habit economy, we shall first test it with a simple model using simulated data. The model is the VAR

$$\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} + \begin{pmatrix} 0.5 & 0.2 \\ 0.2 & 0.5 \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \end{pmatrix} + \begin{pmatrix} 0.001 & 0.0001 \\ 0.0 & 0.001 \end{pmatrix} \begin{pmatrix} z_{1t} \\ z_{2t} \end{pmatrix}$$

The data are n=1000 observations simulated from this VAR.

Simple Example, Simulated Data

But the model as written is very hard to tune. The following is easier to tune and will be fitted to the generated data

$$\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} + \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \begin{pmatrix} y_{1,t-1} - b_1 \\ y_{2,t-1} - b_2 \end{pmatrix}$$

$$+ \begin{pmatrix} R_{11} & R_{21} \\ 0.0 & R_{22} \end{pmatrix} \begin{pmatrix} z_{1t} \\ z_{2t} \end{pmatrix}$$

MCMC Chain

Estimation commences with tuning parameters set as shown next.

Obviously there was some preliminary fiddling, but I didn't save the earliest runs.

Notice the hill climbing early on.

The rejection rate on the following is 4%.

The code has been modified since these runs. Results will not reproduce exactly.

Tuning Parameters

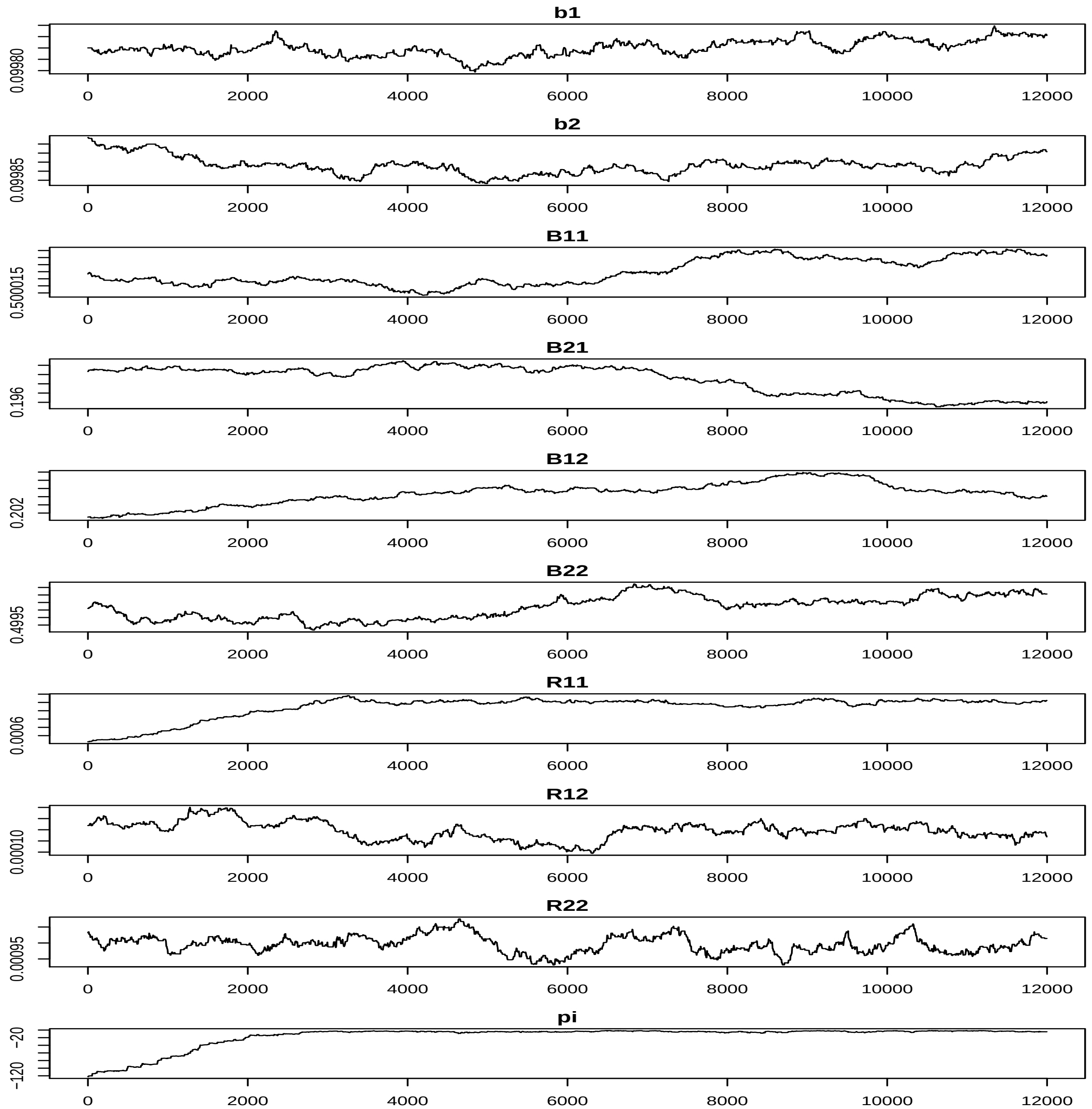
```
const INTEGER prop_def_spec = 0;  //Single move uniform

const REAL b1_range   = 0.0001;
const REAL b2_range   = 0.0001;
const REAL B11_range  = 0.001;
const REAL B21_range  = 0.001;
const REAL B12_range  = 0.001;
const REAL B22_range  = 0.001;
const REAL R11_range  = 0.0001;
const REAL R12_range  = 0.0001;
const REAL R22_range  = 0.0001;

const REAL b1_start  = 9.9900297408326275e-02;
const REAL b2_start  = 1.0008597254455662e-01;
const REAL B11_start = 5.0002840588321151e-01;
const REAL B21_start = 1.9932501881833248e-01;
const REAL B12_start = 2.0150085914184437e-01;
const REAL B22_start = 5.0060666133809983e-01;
const REAL R11_start = 5.2667359071247835e-04;
const REAL R12_start = 2.1881183967790337e-04;
const REAL R22_start = 1.0317855295544479e-03;

const REAL range_factor = (1.0/16.0);
const REAL temperature  = 1.0;
```

Fig 3. VAR MCMC Chain



MCMC Chain

This is the hill climbing or simulated annealing phase of the iterations.

The values from the end of the chain are used to restart the chain and tuning parameters are adjusted as seems appropriate.

The following is the best that I could do with a single move proposal.

Rejection rate on what follows is 8%.

Tuning Parameters

```
const INTEGER prop_def_spec = 0;  //Single move uniform

const REAL b1_range   = 0.0001;
const REAL b2_range   = 0.0001;
const REAL B11_range  = 0.015;
const REAL B21_range  = 0.015;
const REAL B12_range  = 0.015;
const REAL B22_range  = 0.015;
const REAL R11_range  = 0.0001;
const REAL R12_range  = 0.0001;
const REAL R22_range  = 0.0001;

const REAL b1_start  = 9.9954984183242251e-02;
const REAL b2_start  = 1.0001128385572358e-01;
const REAL B11_start = 5.0004171603170977e-01;
const REAL B21_start = 1.9593909758070538e-01;
const REAL B12_start = 2.0404597457467594e-01;
const REAL B22_start = 5.0156398756554943e-01;
const REAL R11_start = 1.0238303908475348e-03;
const REAL R12_start = 1.6994725445393879e-04;
const REAL R22_start = 1.0212344683387820e-03;

const REAL range_factor = (1.0/8.0);
const REAL temperature  = 1.0;
```

Fig 4. VAR MCMC Chain

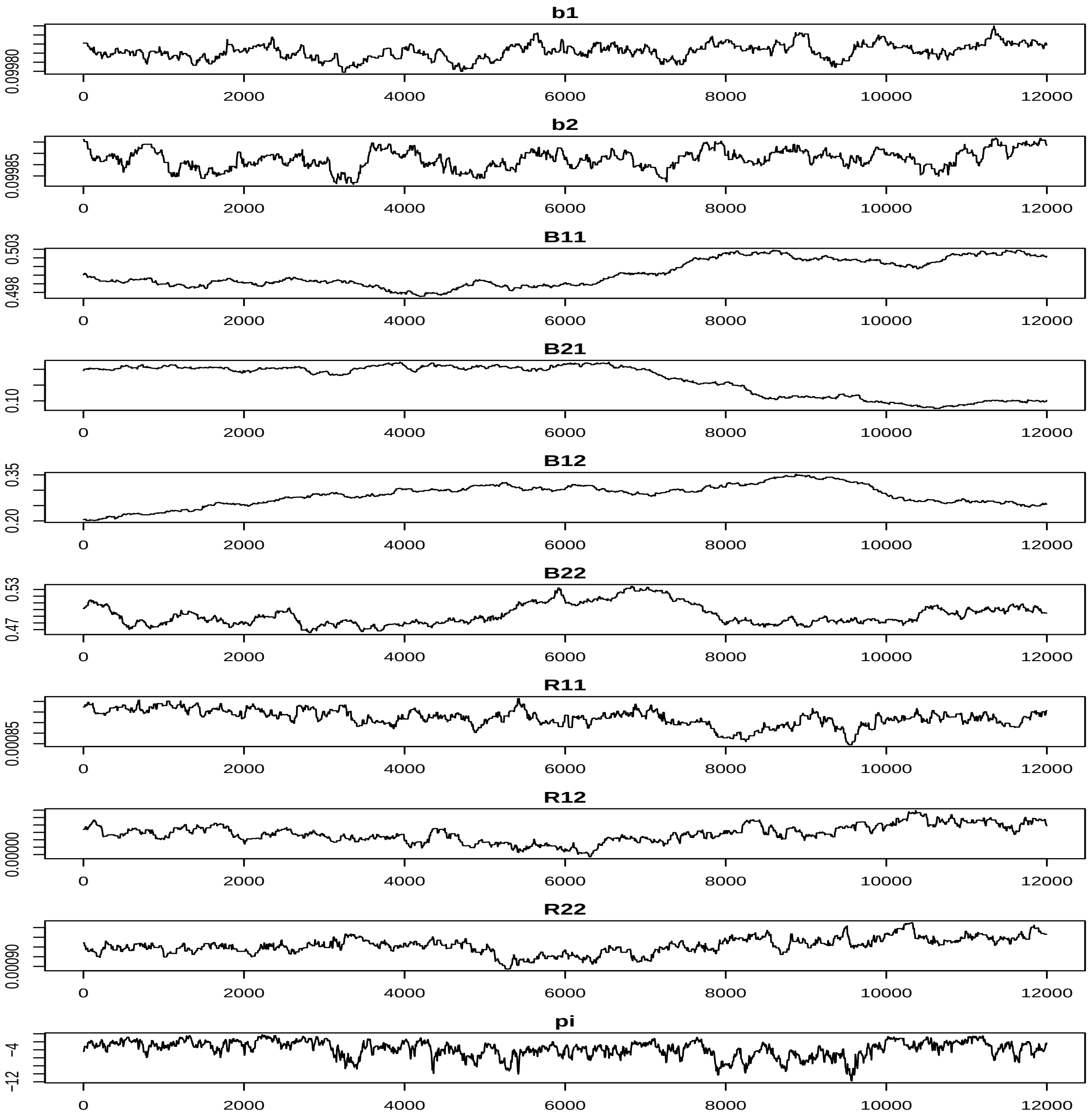


Fig 5. VAR MCMC Autocorrelations

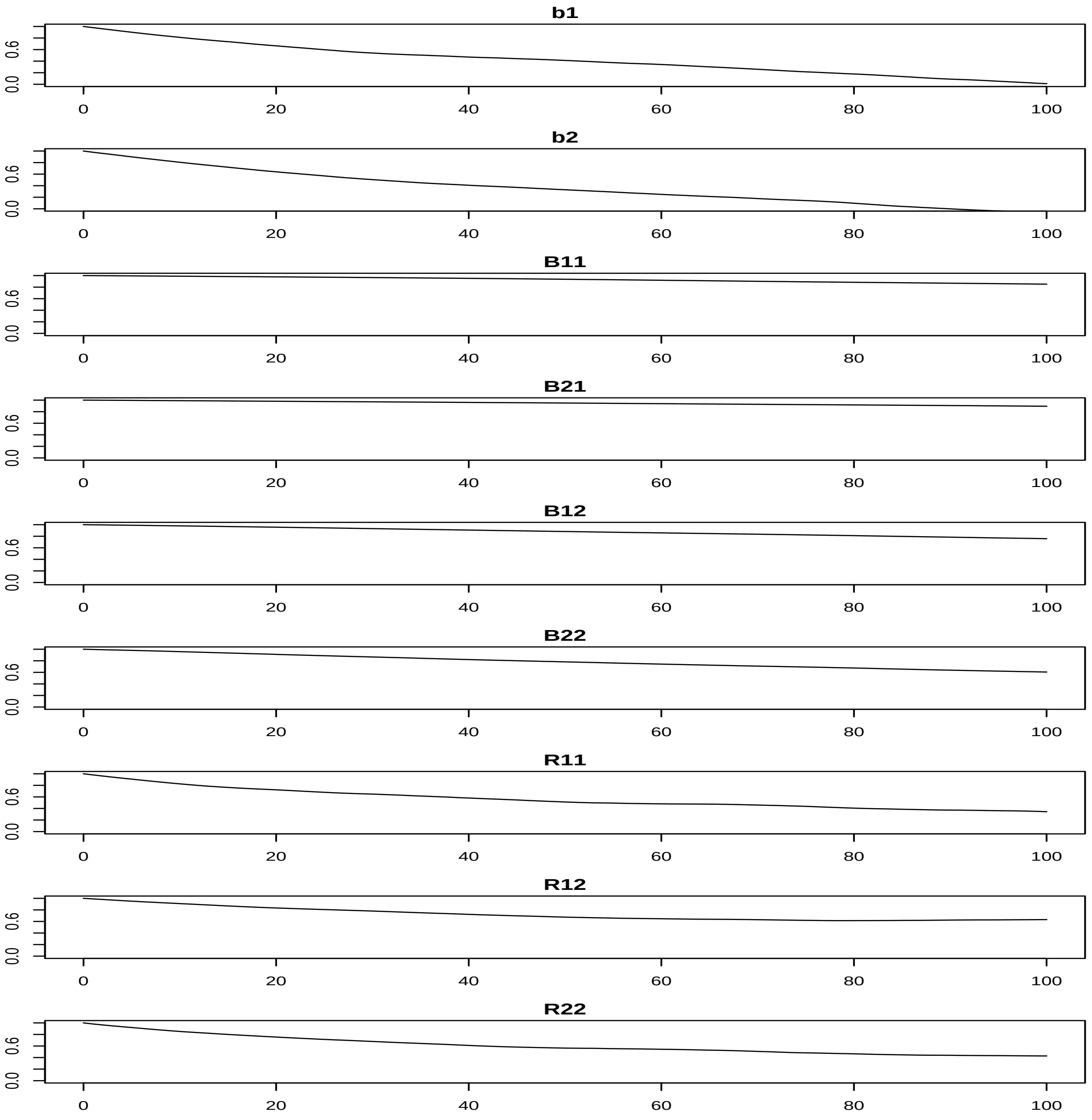


Fig 6. VAR MCMC Scatter Plots

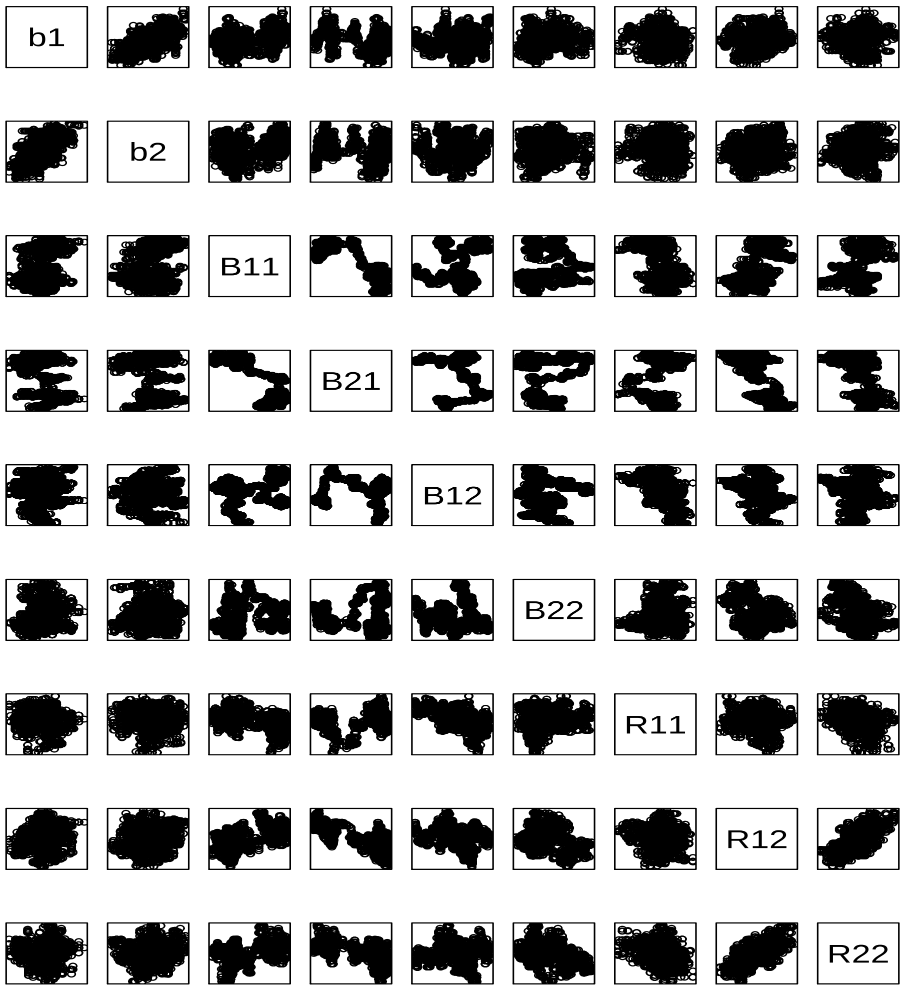
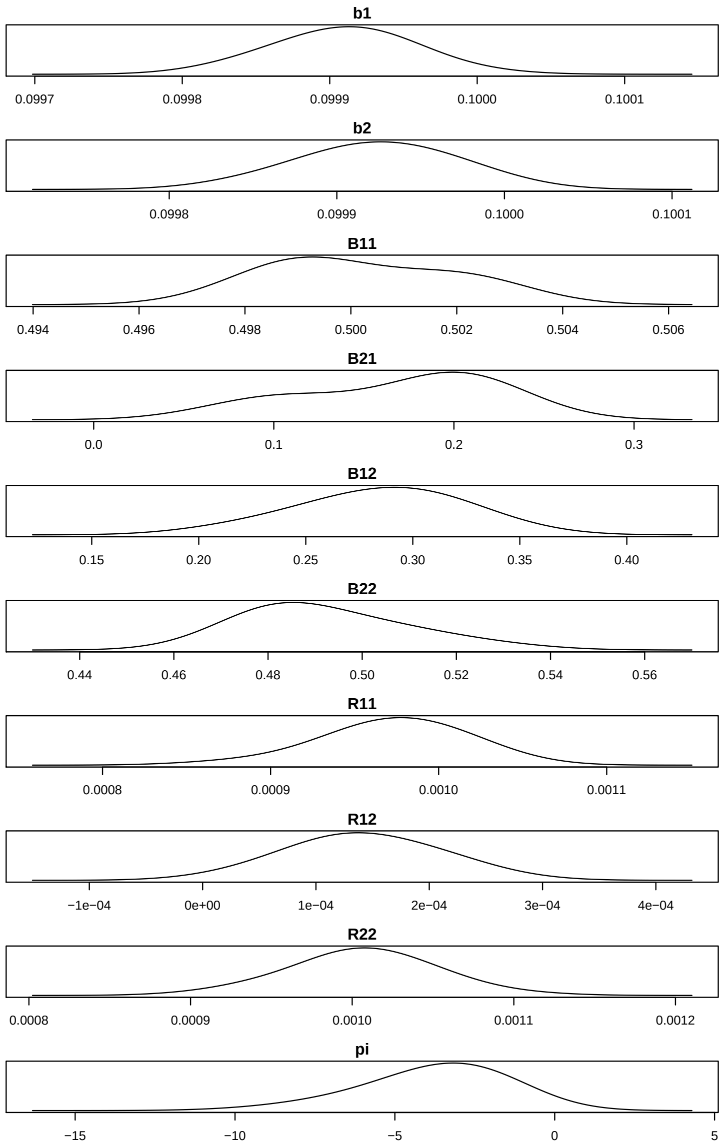


Fig 7. VAR MCMC Density Plots



MCMC Chain

Went to group move proposal. The parameters B_{11}, B_{21}, B_{12}, B_{22} were moved as a group.

The rejection rate was 20%.

Tuning Parameters

```
const INTEGER prop_def_spec = 4;  //Group move normal

const REAL b1_range   = 0.0001;
const REAL b2_range   = 0.0001;
const REAL B11_range  = 0.001;
const REAL B21_range  = 0.01;
const REAL B12_range  = 0.02;
const REAL B22_range  = 0.001;
const REAL R11_range  = 0.0001;
const REAL R12_range  = 0.0001;
const REAL R22_range  = 0.0001;

const REAL b1_start  = 9.9954984183242251e-02;
const REAL b2_start  = 1.0001128385572358e-01;
const REAL B11_start = 5.0004171603170977e-01;
const REAL B21_start = 1.9593909758070538e-01;
const REAL B12_start = 2.0404597457467594e-01;
const REAL B22_start = 5.0156398756554943e-01;
const REAL R11_start = 1.0238303908475348e-03;
const REAL R12_start = 1.6994725445393879e-04;
const REAL R22_start = 1.0212344683387820e-03;

const REAL range_factor = (1.0/8.0);
const REAL temperature  = 1.0;
```

Fig 8. VAR MCMC Chain

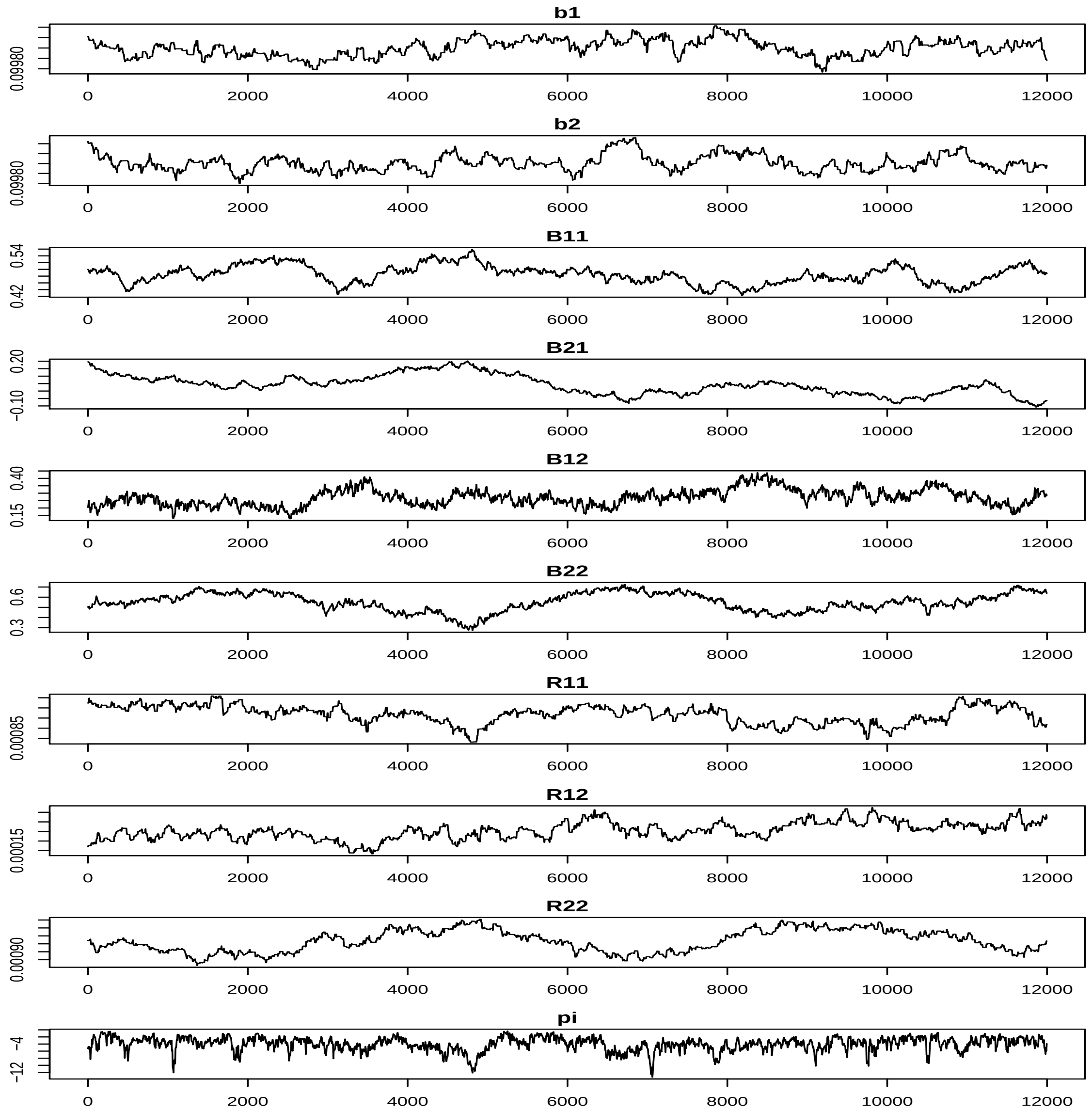


Fig 9. VAR MCMC Autocorrelations

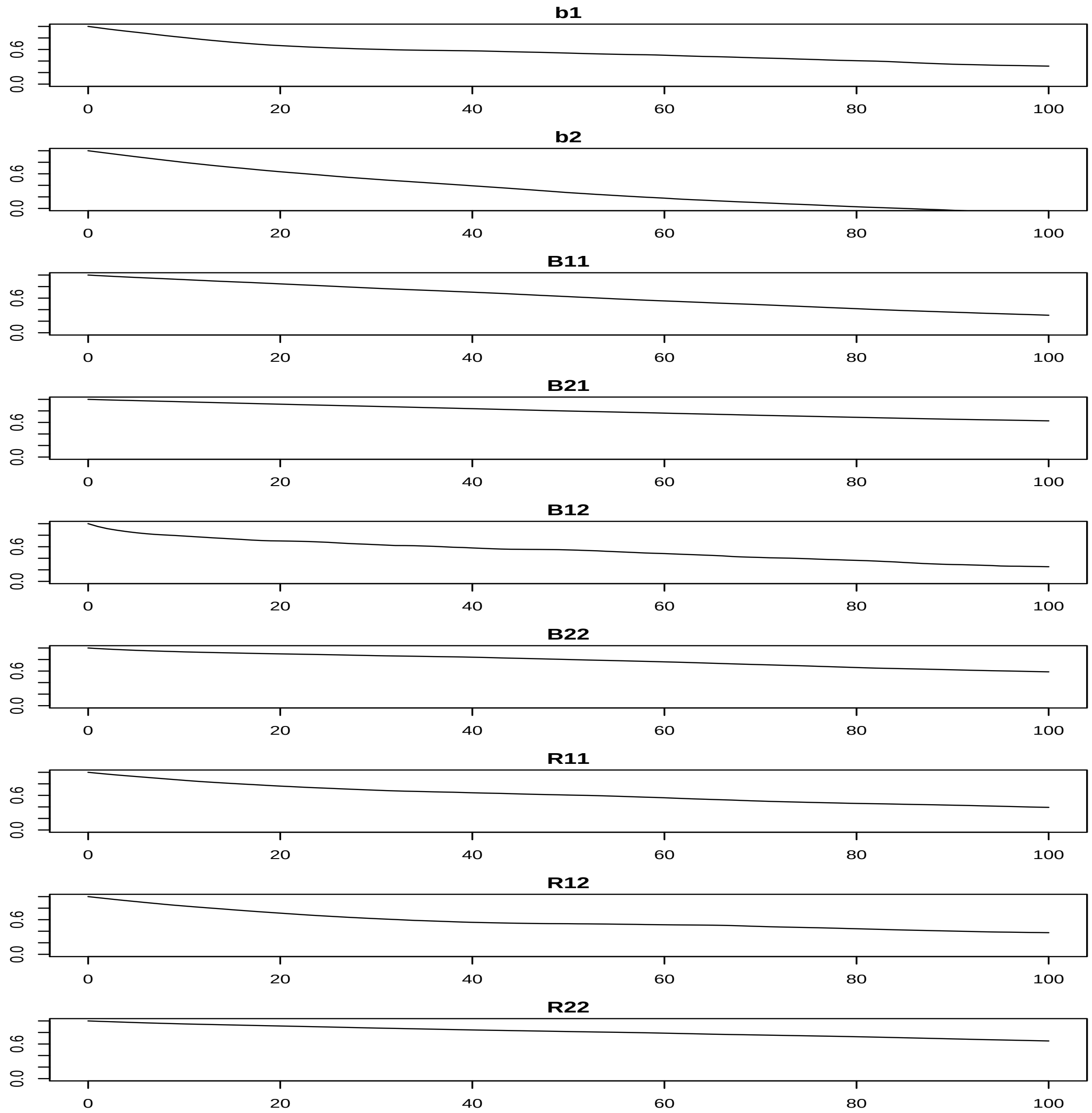


Fig 10. VAR MCMC Scatter Plots

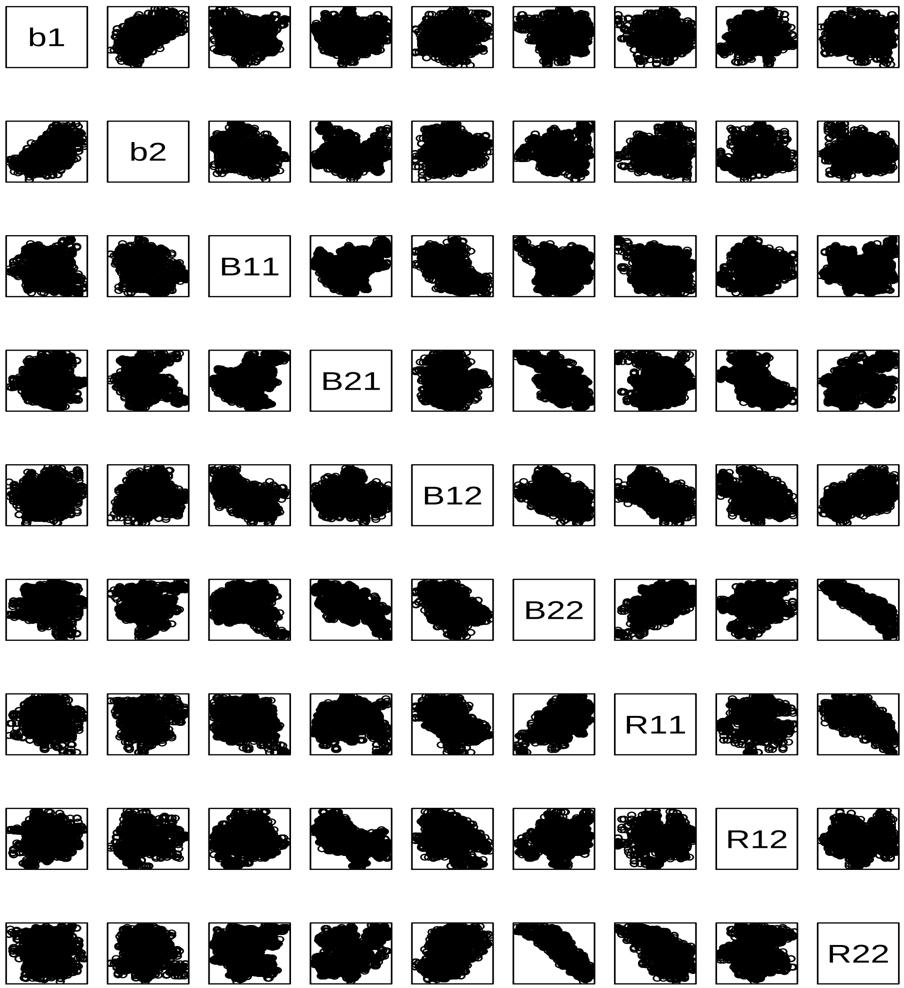
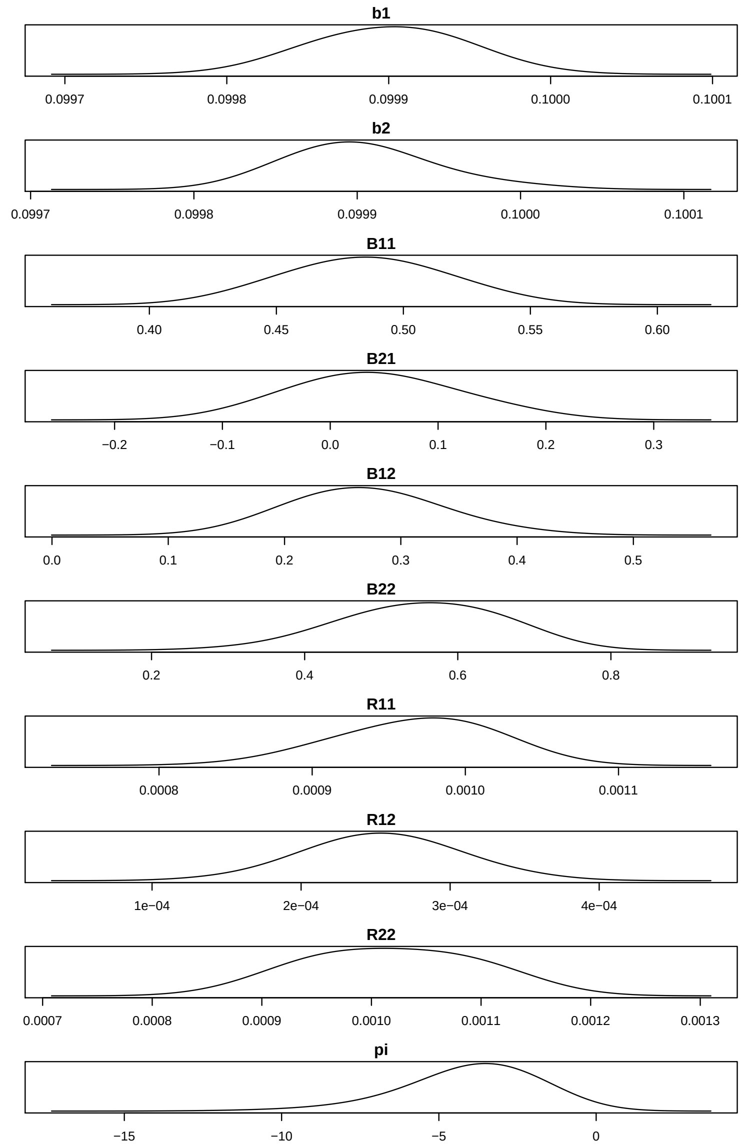


Fig 11. VAR MCMC Density Plots



Parameter Estimates

		OLS Estimates		MCMC-GMM Estimates	
Parameter		Estimate	Std. Err.	Estimate	Std. Err.
b_1	0.1	0.08427	0.00958	0.09989	0.0000328
b_2	0.1	0.10404	0.00980	0.09990	0.0000391
B_{11}	0.5	0.48892	0.02701	0.49461	0.019511
B_{21}	0.2	0.18108	0.02762	0.17448	0.040587
B_{12}	0.2	0.25827	0.02734	0.25797	0.029000
B_{22}	0.5	0.50675	0.02797	0.50470	0.032414
R_{11}	0.0010			0.00098833	0.0000267
R_{12}	0.0001			0.00015640	0.0000368
R_{22}	0.0010			0.00098587	0.0000249

MCMC estimates based on a chain of length 12,000.

Habit Model

We will now consider results for the habit model.

The process is the same as the VAR example. There is a hill climbing stage and then a tuning stage.

Finally the parallel (MPI) version of the code was used on a machine with 8 CPU's for the final run. $R = 2500 \times 6 \times 7 = 105000$, stride=25 in plots.

Results follow.

Prior Information

Support: Reasonable bounds on all parameters to include positivity restrictions on positive valued parameters and non-explosive restrictions on autoregressive parameters.

Numerical: Existence of solution to Euler condition.

Used by methods proposed here (annualized, iid normal prior)

$$P\left(\left|\mathcal{E}(r_f^a) - 0.89\%\right| < 1\%\right) = 0.95$$

$$P\left(|\rho - 0.2| < 0.1\right) = 0.95$$

$$P\left(|\phi - 0.9884| < 0.01\right) = 0.95$$

Used by estimates compared with (annualized, uniform prior)

$$P\left(\left|\mathcal{E}(r_f^a) - 0.89\%\right| < 0.5\%\right) = 1.00$$

Results Will Be Compared to EMM Estimates

EMM Heuristics: For any QMLE estimator

$$\tilde{\eta}_n = \underset{\eta}{\text{argmax}} \frac{1}{n} \sum_{t=1}^{n} \log f(\tilde{y}_t | \tilde{x}_{t-1}, \eta),$$

a sample average satisfies

$$0 = \frac{1}{n} \sum_{t=1}^{n} \frac{\partial}{\partial \eta} \log f(\tilde{y}_t | \tilde{x}_{t-1}, \tilde{\eta}_n)$$

because these are the first order conditions of the optimization problem.

Therefore a large simulation from a putative DGP $p(y_t | x_{t-1}, \theta)$ will satisfy

$$0 = m(\theta, \tilde{\eta}_n) = \frac{1}{N} \sum_{t=1}^{N} \frac{\partial}{\partial \eta} \log f(\hat{y}_t | \hat{x}_{t-1}, \tilde{\eta}_n),$$

except for sampling variation in $\tilde{\eta}_n$. The equality holds exactly in the limit as n and N tend to infinity.

The EMM estimator attempts to find θ that solves these estimating equations as nearly as possible:

$$\hat{\theta}_n = \underset{\theta}{\text{argmin}}\, m'(\theta, \tilde{\eta}_n)(\tilde{\mathcal{I}}_n)^{-1} m(\theta, \tilde{\eta}_n)$$

Tuning Parameters

```
const INTEGER prop_def_spec = 0;  //Single move normal

REAL g_range     = 0.001;
REAL R11_range   = 0.01;
REAL R12_range   = 0.05;
REAL R22_range   = 0.05;
REAL phi_range   = 0.01;
REAL delta_range = 0.006;
REAL gamma_range = 1.10;

REAL g_start     = 1.976079088512222668e-03;
REAL R11_start   = 5.289677322278638245e-04;
REAL R12_start   = 1.078692952608877541e-04;
REAL R22_start   = 8.759089089812089474e-03;
REAL phi_start   = 9.886147326276307767e-01;
REAL delta_start = 9.940743738336396129e-01;
REAL gamma_start = 1.404090722074126996e+00;

const REAL range_factor = (1.0/16.0);
const REAL temperature  = 5.0;
```

Fig 12. Habit Model MCMC Chain

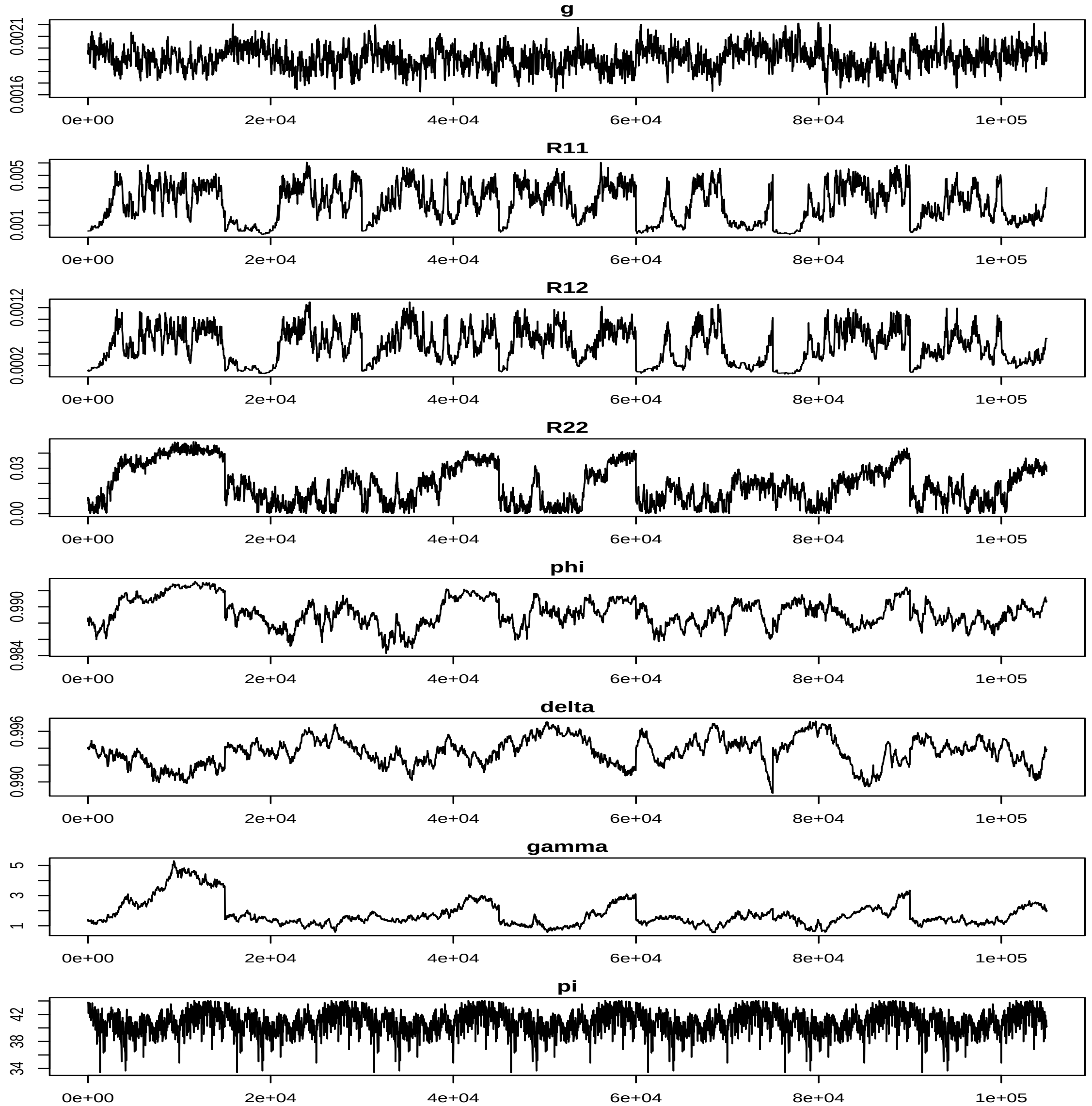


Fig 13. Habit Model Autocorrelations

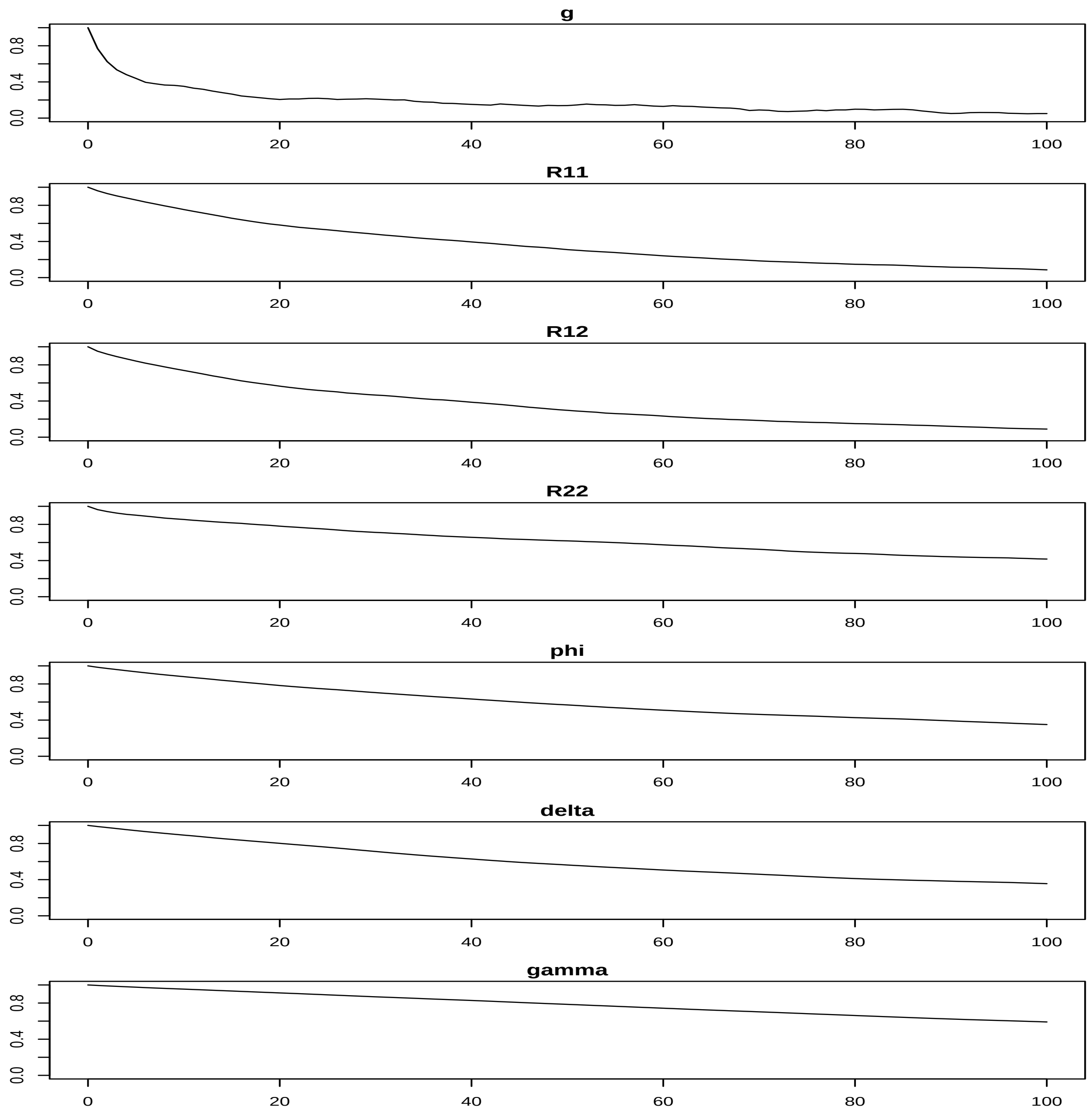


Fig 14. Habit Model Scatter Plots

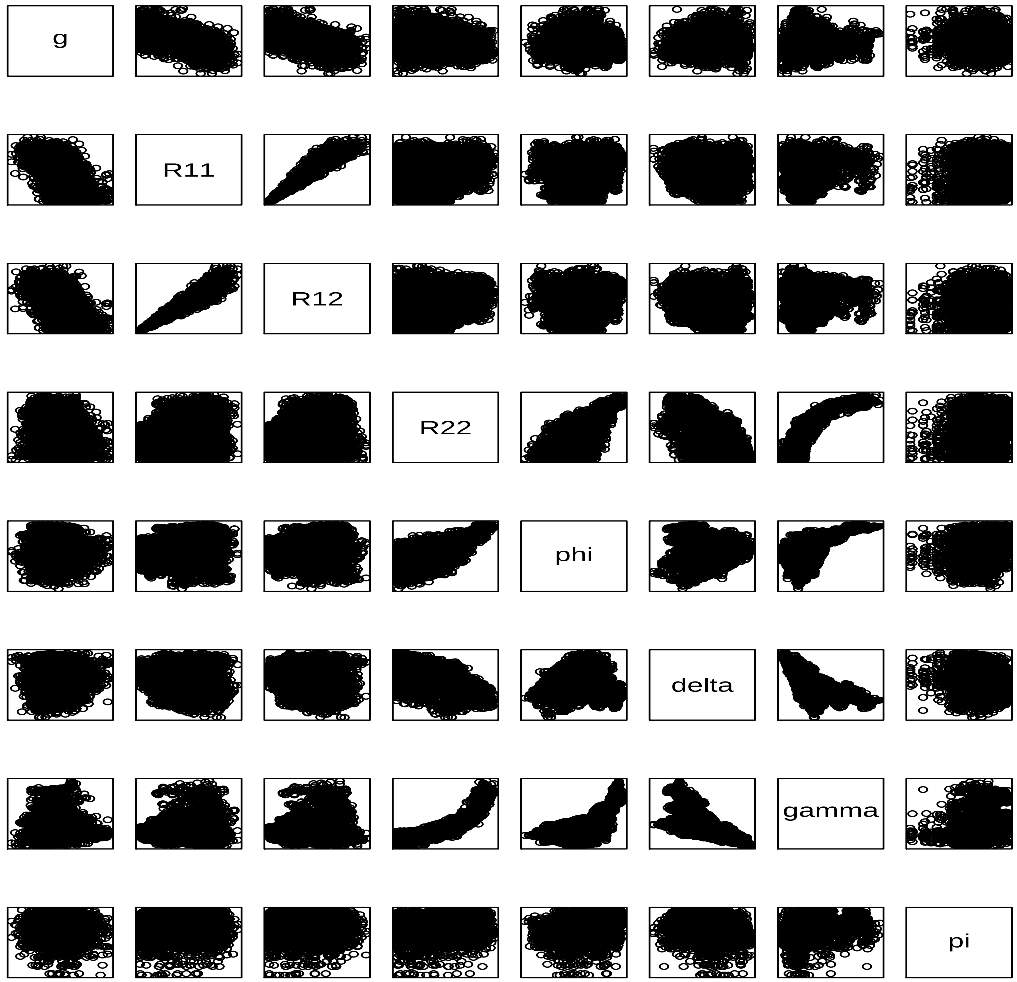


Fig 15. Habit Model Density Plots

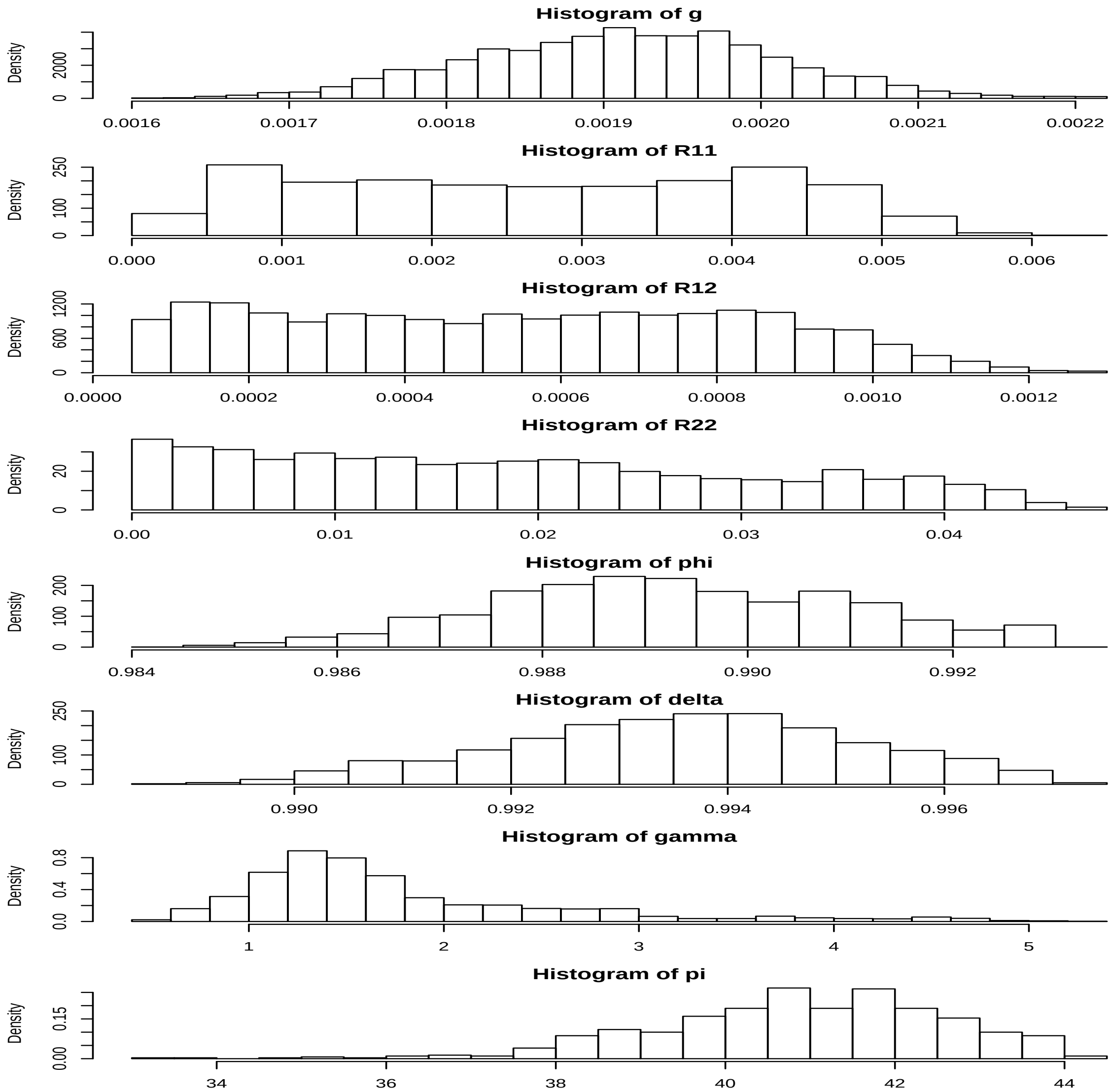


Fig 16. Habit Model Functionals Density Plots

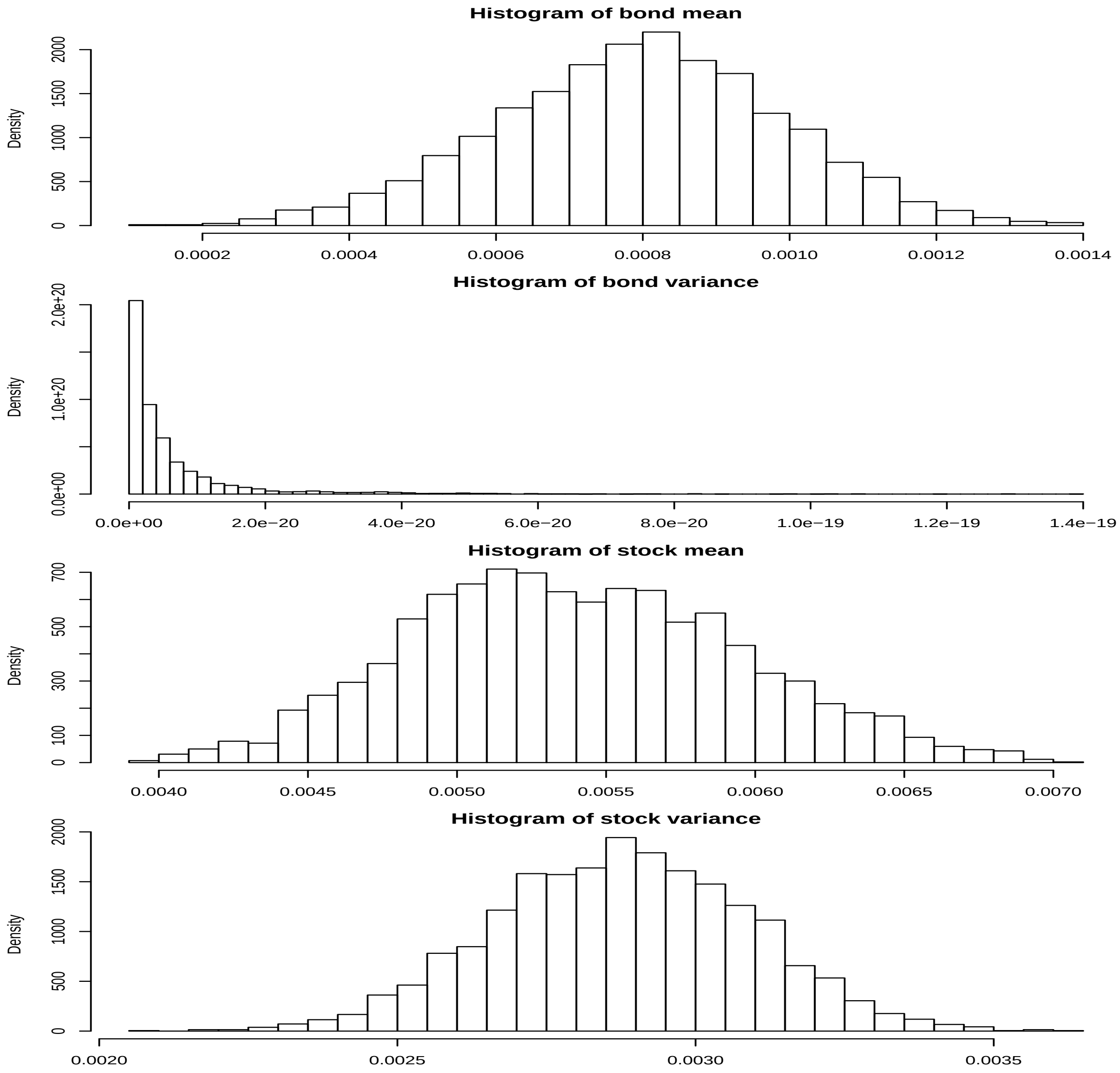


Table 2. Parameter Estimates (Monthly Frequency)

	EMM Estimates		MCMC-GMM	
Parameter	Estimate	Std. Err.	Estimate	Std. Err.
g	0.002116	0.000250	0.0019963	0.000085
ψ_{11}	0.006151	0.000896		
ψ_{12}	0	0		
ψ_{22}	0.036503	0.007716		
ρ_s	0.971900	0.015449		
R_{11}			0.0010254	0.001125
R_{12}			0.0001982	0.000224
R_{22}			0.0300096	0.009592
ϕ	0.9853	0.0026	0.9898	0.0010
δ	0.9939	0.0005	0.9916	0.0012
γ	0.8386	0.2462	2.4850	0.4522
μ_{dc}	-3.3587	0.0380		
	$\chi^2(4) = 7.109$ (0.7894)		$R = 105,000$	

Note: c and d are cointegrated for EMM estimates; Ψ and R are upper triangular matrices related as follows

$$\text{Var}\begin{pmatrix} c_t - c_{t-1} \\ d_t - d_{t-1} \end{pmatrix} = RR' = \left[\begin{pmatrix} 1 & 0 \\ 1 & (\rho_s^2 - 2\rho_s)^{-1} \end{pmatrix}\Psi\right]\left[\begin{pmatrix} 1 & 0 \\ 1 & (\rho_s^2 - 2\rho_s)^{-1} \end{pmatrix}\Psi\right]'$$

Table 3. Parameter Estimates (Annual Frequency)

	Data		EMM Estimates		MCMC-GMM	
Parameter	Estimate	Std Dev	Estimate	Std Dev	Estimate	Std Dev
g			2.539	0.0087	2.239	0.1021
σ			1.7626		0.3618	0.3971
ρ			0.2062		0.1898	0.0226
σ_w			9.5965		10.4254	3.2228
ϕ			0.8372	0.0090	0.8844	0.0101
δ			0.9292	0.0017	0.9039	0.0187
γ			0.8386	0.2462	2.4854	0.4522
μ_{dc}			-3.3587	0.0380		
$d_t^a - c_t^a$	-3.40	0.16	-3.37	0.15		
$c_t^a - c_{t-1}^a$	1.95	2.24	2.52	1.76	2.33	0.3618
P_{dt}^a/D_t^a	28.24	12.08	27.75	7.04		
r_{dt}^a	6.02	19.29	6.54	16.9	6.93	18.53
$\sqrt{Q_t^a}$	16.69	09.32	14.41	9.69		
r_{ft}^a			1.07	3.23	0.87	0.00
$r_{dt}^a - r_{ft}^a$			5.46	17.1	6.06	

Results for Bayesian Estimation

Results for Bayesian estimation are next. The prior used was

$$P\left(\left|\mathcal{E}(r_f^a) - 0.89\%\right| < 1\%\right) = 0.95$$

$$P\left(|\rho - 0.2| < 0.1\right) = 0.95$$

$$P\left(|\phi - 0.9884| < 0.02\right) = 0.95$$

which is the same as the foregoing.

Table 4. Parameter Estimates (Monthly Frequency)

	EMM Estimates		Bayesian	
Parameter	Estimate	Std. Err.	Estimate	Std. Err.
g	0.002116	0.000250	0.001803	0.000684
ψ_{11}	0.006151	0.000896		
ψ_{12}	0	0		
ψ_{22}	0.036503	0.007716		
ρ_s	0.971900	0.015449		
R_{11}			0.007254	0.001903
R_{12}			0.001350	0.001068
R_{22}			0.003125	0.034435
ϕ	0.9853	0.0026	0.9804	0.0095
δ	0.9939	0.0005	0.9898	0.0070
γ	0.8386	0.2462	1.0744	1.7638
μ_{dc}	-3.3587	0.0380		
	$\chi^2(4) = 7.109$ (0.7894)		$R = 800,000$	

Note: c and d are cointegrated for EMM estimates; Ψ and R are upper triangular matrices related as follows

$$\text{Var}\begin{pmatrix} c_t - c_{t-1} \\ d_t - d_{t-1} \end{pmatrix} = RR' = \left[\begin{pmatrix} 1 & 0 \\ 1 & (\rho_s^2 - 2\rho_s)^{-1} \end{pmatrix}\Psi\right]\left[\begin{pmatrix} 1 & 0 \\ 1 & (\rho_s^2 - 2\rho_s)^{-1} \end{pmatrix}\Psi\right]'$$

Table 5. Parameter Estimates (Annual Frequency)

	Data		EMM Estimates		Bayesian	
Parameter	Estimate	Std Dev	Estimate	Std Dev	Estimate	Std Dev
g			2.539	0.0087	2.164	0.2300
σ			1.7626		2.5589	
ρ			0.2062		0.1830	
σ_w			9.5965		1.0825	
ϕ			0.8372	0.0090	0.7890	0.0328
δ			0.9292	0.0017	0.8845	0.0244
γ			0.8386	0.2462	1.0744	1.7638
μ_{dc}			-3.3587	0.0380		
$d_t^a - c_t^a$	-3.40	0.16	-3.37	0.15		
$c_t^a - c_{t-1}^a$	1.95	2.24	2.52	1.76	2.164	2.56
P_{dt}^a/D_t^a	28.24	12.08	27.75	7.04		
r_{dt}^a	6.02	19.29	6.54	16.9	11.14	24.22
$\sqrt{Q_t^a}$	16.69	09.32	14.41	9.69		
r_{ft}^a			1.07	3.23	1.21	0.42
$r_{dt}^a - r_{ft}^a$			5.46	17.1	9.94	