# Hidden Markov Chain Models

December 7, 2017

©2017 by Christopher A. Sims. This document is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
http://creativecommons.org/licenses/by-nc-sa/3.0/

# The class of models

$$y_t \mid \{y_s, s < t\} \sim p(y_t \mid \theta_t, \{y_s, s < t\})$$
$$\theta_t = \theta(S_t)$$
$$P[S_t = i \mid S_{t-1} = j] = h_{ij} \, .$$

# What's handy about it

- Evaluating the likelihood is straightforward;
- filtered estimates of $\{S_t\}$ emerge as a byproduct (as in the KF);
- the filtered estimates can be recursively "back-filtered" to generate smoothed estimates of $\{S_t\}$ (again as with the KF);
- a recursive algorithm much like the smoother will generate draws from the conditional pdf of $\{S_t\}$ given data and other parameters;
- therefore MLE and posterior simulation via MCMC are straightforward.

# Uses of and problems with the model

- easily accommodates discontinuous “regime changes”.
  - Better than deterministic changes, usually, because if they have happened once, they can probably happen again, so modeling uncertainty about them is important.
- Shifts in residual variances are easy to handle, so that the model becomes a way of modeling stochastically varying heteroskedasticity, i.e. stochastic volatility.
- Hidden Markov chain models make *all* volatility shifts discontinuous, though, which may not be appealing.

- By using many states and a suitably restricted $H$ matrix, a hidden Markov chain model can approximate a model with continuously distributed, continuously varying heteroskedasticity.

- We might like to have $H$ endogenous. This is quite inconvenient.

## Implementing the calculations for likelihood and for filtered, smoothed and simulated $S$

Recursive algorithm:

- Notation: $p$ is generic pdf. $Y_t = \{y_s \mid s \leq t\}$.
- Given from recursion: $p(S_t \mid Y_t, \theta(\cdot), H)$
- Given from the model: $p(y_{t+1} \mid Y_t, S_{t+1}, \theta(\cdot), H)$
- Object: use the observation $y_{t+1}$ to form $p(S_{t+1} \mid Y_{t+1}, \theta(\cdot), H)$. Using that, and the model, we get the likelihood element $p(y_{t+1} \mid Y_t, \theta(\cdot), H)$

$$p(y_{t+1}, S_{t+1}, S_t \mid Y_t) = p(S_t \mid Y_t) \cdot p(S_{t+1} \mid Y_t, S_t) \cdot p(y_{t+1} \mid Y_t, S_t, S_{t+1})$$
$$= p(S_t \mid Y_t) \cdot p(S_{t+1} \mid S_t) \cdot p(y_{t+1} \mid Y_t, \theta(S_{t+1}))\,.$$

We know how to evaluate this expression. But then

$$p(y_{t+1} \mid Y_t) = \sum_{i,j} p(y_{t+1}, S_{t+1} = i, S_t = j \mid Y_t) \tag{1}$$

$$p(S_{t+1} \mid Y_{t+1}) = \frac{\sum_j p(y_{t+1}, S_{t+1}, S_t = j \mid Y_t)}{p(y_{t+1} \mid Y_t)}\,. \tag{2}$$

# Initial conditions

To form a complete likelihood, we need to know the unconditional distribution of $S_1$. Usually the observed value of $y_1$ will in fact be informative about $S_1$, but calculating the unconditional joint distribution for $y_1$ and $S_1$ implied by the dynamic model is difficult, and if the model might be non-stationary, the unconditional distribution might not even exist. We can instead treat $y_1$ as non-stochastic and assume $S_1$ is drawn from the unconditional distribution of $S$ alone. This can be found from $H$ as the right-eigenvector of $H$ associated with its unit eigenvalue. This will be a vector $\bar{p}$ satisfying

$$H\bar{p} = \bar{p}\,, \tag{3}$$

which obviously makes $\bar{p}$ a steady-state for the unconditional pdf. It can happen that $H$ has multiple unit-eigenvalues. In that case the Markov

process is not ergodic, meaning that when started up from different initial conditions it can converge to different steady-state distributions. (An example: $H = I$, which imply that the process simply remains in whatever state it started in.) In such cases we take $\bar{p}$ to be the average of all the right eigenvectors corresponding to unit roots.

So to start up the recursion, we use $\bar{p}$ in place of the $p(S_1 \mid Y_1)$ that the recursion calls for.

## Initial conditions for non-stationary $H$, normalization

- Example: $H$ upper triangular. This implies that transitions always run "up" the state vector. State $j$ can transition to state $i < j$, but not to any state $i > j$. This is natural. E.g., in modeling policy regime shifts, if one thought policy was steadily improving.

- In this case clearly one wants the initial distribution to put probability one on $S_0 = n$, the highest-numbered state.

- A more general problem: Without other restrictions, renumbering the states does not change the model. Triangular $H$ resolves this.

- With just two states, it might be natural to normalize by insisting that one is for sure the first state.

- Another example: Variance parameter in state $j$ greater than in state $k$ if $k < j$. One then discards draws violating the inequality.

- This normalization doesn't matter for fitting or forecasting. It does matter if we want to look at smoothed estimates of the state and interpret them.

# Approximating smoothly evolving states

- suppose it's natural to think of a parameter $\theta$ drifting over time in a serially correlated way, e.g. $\theta_t = \rho\theta_{t-1} + \varepsilon_t$, $\varepsilon_t$ i.i.d. $N(0, 1)$, $|\rho| < 1$.
- Break up $\mathbb{R}$ into $n$ intervals that have similar probability under the unconditional distribution of $\theta_t$.
- Use the Gaussian model to calculate $h_{ij}$ values for these intervals.
- With many small intervals, this approximates the AR model for $\theta_t$ well.
- Big $H$ matrix, but only one free parameter.

- On the other hand, Dirichlet structure lost.

# Smoothed state distribution

The recursion to generate the smoothed distribution of states assumes that we start with $p(S_{t+1} \mid Y_T)$ and have available the complete filtering output from the forward recursion, i.e. $p(S_t \mid Y_t), t = 1, \ldots, T)$. We aim at finding $p(S_t \mid Y_T)$, from which we can continue the recursion. We observe

$$p(S_{t+1}, S_t \mid Y_T) = p(S_{t+1} \mid Y_T) \cdot p(S_t \mid S_{t+1}, Y_T) = p(S_{t+1} \mid Y_T) \cdot p(S_t \mid S_{t+1}, Y_t)$$
$$= p(S_{t+1} \mid Y_T) \cdot \frac{p(S_{t+1} \mid S_t) \cdot p(S_t \mid Y_t)}{\sum_j p(S_{t+1} \mid S_t = j) \cdot p(S_t = j \mid Y_t)}. \quad (4)$$

The second equality follows because $y_{t+s}$ for $s \geq 1$ depends on $S_t$ only through $S_{t+1}$, so that when $S_{t+1}$ is known, only values of $y$ dated $t$ and

earlier provide additional information about $S_t$. The last equality just applies the formulas relating conditional and joint densities.

The right-hand side of the last equality in (4) involves only factors we have assumed we already know, so we can compute this expression. But then looking at the first left-hand side, it is apparent that if we sum it over all possible values of $S_{t+1}$, we arrive at our target, $p(S_t \mid Y_T)$.

## Sample path for states, given all the data

Finally, the recursion to generate a sample path of $S_t$ from its conditional distribution given the data can also be based on (4). Note that the right-hand side of the first equality includes the term $p(S_t \mid S_{t+1}, Y_T)$, which is then constructed via the expressions developed in the remaining equalities. Note further that

$$p(S_t \mid S_{t+1}, Y_T) = p(S_t \mid \{S_{t+s}, s \geq 1\}, Y_T), \tag{5}$$

because future $S$'s (like future $y$'s) depend on current $S_t$ only via $S_{t+1}$. Thus we can implement a backward recursion. Beginning by drawing $S_T$ from the

filtered $p(S_T \mid Y_T)$, we then draw $S_{T-1}$ from

$$p(S_{T-1} \mid S_T, Y_T) = \frac{p(S_T \mid S_{T-1}) \cdot p(S_{T-1} \mid Y_{T-1})}{\sum_j p(S_T \mid S_{T-1} = j) \cdot p(S_{T-1} = j \mid Y_{T-1})}, \qquad (6)$$

and so on back to $t = 1$.

# Implementing a full Gibbs chain on all model parameters

The algorithm in the preceding section for making a draw from the distribution of $\{S_t\} \mid Y_T$ can form one component of a Gibbs sampling scheme. The other component, of course, would have to be sampling from the posterior pdf of $\theta(\cdot)$ and $H$ conditional on $\{S_t\}$. Whether this is convenient or even feasible depends on the form of $p(y_t \mid Y_{t-1}, \theta(\cdot), H)$ and on what restrictions there may be on the form of $H$.

When $H$ is unrestricted (except for the normalizing identity that requires each column to sum to one), the likelihood depends on it, for fixed $\{S_t\}$, in a straightforward way. It is not likely to be practical to leave $H$ unrestricted except in small models with few states, but the principles here will carry over to some other cases. As a function of the elements $h_{ij}$ of $H$, the likelihood

with the $S$'s fixed is proportional to

$$\bar{p}(S_1) \prod_{i,j} h_{ij}^{n(i,j)} , \tag{7}$$

where $n(i,j)$ is the number of dates $t$, $2 \leq t \leq T$, at which $S_{t-1} = i$ and $S_t = j$. The steady-state probability distribution $\bar{p}$ depends on $H$ via (3). The part of (7) following the $\bar{p}$ term has the form of the product of $m$ independent Dirichlet$(n(1,j)+1, n(2,j)+1, \ldots, n(m,j)+1)$ pdf's, for $j = 1, \ldots, m$. The Dirichlet is a standard distribution and is easy to sample from. (If $v$ is a vector of $k$ independent gamma variates, with degrees of freedom given by the $k$-vector $n$, then $v/\operatorname{sum}(v)$ is distributed Dirichlet with parameter vector $n$.) The $\bar{p}$ term gives the full expression a non-standard form, but in reasonably large samples the $\bar{p}$ term is likely to be fairly flat relative to the remainder of the expression. Therefore a Metropolis-Hastings sampling scheme, in which the values of $H$ are drawn from the Dirichlet,

then an accept/repeat decision is made based on the value of the $\bar{p}$ term under the new and old draw, is likely to be fairly efficient.

This method of sampling from the conditional posterior for $H$ under a flat prior is easily adapted to cases where there are zero restrictions on the $H$ matrix and to non-flat, but Dirichlet, priors. However it may well be attractive to parameterize $H$ more tightly than is possible with zero restrictions alone, and this may make sampling from the $H$ posterior more difficult.

Sampling from the $\theta(\cdot)$ posterior is inevitably model-dependent, so we take that up in our example below.

# Example: Poor Man's Stochastic Volatility Model

We take $p(y_t \mid \theta(\cdot), Y_t)$ from the general model of section  to be determined by the equation

$$r_t = \alpha_0 + \alpha_1 r_{t-1} + \varepsilon_t \sigma(S_t)\,, \tag{8}$$

where $r$ is thought of as an interest rate or asset yield and $\varepsilon$ is i.i.d. $N(0,1)$. The $\theta(\cdot)$ function is therefore characterized by $m+2$ parameters: The $m$ values of $\sigma_j, j = 1,\ldots,m$ corresponding to the $m$ states, plus $\alpha_0$ and $\alpha_1$, which do not vary with the state.

We impose the requirement that $\sigma_j < \sigma_{j+1}$, all $j$. Since the states differ only in their $\sigma$ values, we can apply the same permutation to the

$\sigma$ sequence and to the subscripts of the $H$ matrix without changing the model's implications for the behavior of the observable $y$'s. The restriction that the $\sigma$ sequence be a monotone function of the state is therefore necessary to avoid redundancy in the parameterization.

Sampling from the conditional posterior of these $m+2$ parameters conditional on $H$ and an $\{S_t\}$ sequence is best accomplished in two steps. First, conditioning on the $\{\sigma_j\}$ values, the likelihood is that of a normal linear regression with known, but time varying variances, i.e. a "weighted least squares" model. The posterior is therefore normal, centered on the least-squares estimates of $\alpha_0$ and $\alpha_1$ based on the appropriately weighted data. (Data for a period $t$ in which $S_t = j$ are weighted by $1/\sigma_j$). The covariance matrix of this normal distribution is just the "$(X'X)^{-1}$" matrix for the weighted data. So sampling from this conditional distribution is simple.

With the $\alpha$'s held fixed, the likelihood as a function of the $\sigma$'s is

proportional to

$$\prod_{j=1}^{m} \sigma_j^{-n(j)} e^{-\frac{s_j^2}{2\sigma_j^2}}, \tag{9}$$

where $n(j)$ is the number of occurrences of state $j$ in the $\{S_t\}$ sequence and $s_j^2$ is the sum of squared residuals, calculated using the fixed values of $\alpha$, over time periods in which $S_t = j$. The expression (9) is proportional to the product of $m$ inverse-chi-squared pdf's, for with the $j$'th pdf having $s_j^2/2$ as its inverse-scale parameter and $n(j) - 1$ as its shape parameter. It is therefore easy to draw from this distribution. To impose our monotonicity requirement on the $\sigma$ sequence, we can just discard any draw that violates the requirement. So long as the data contain substantial evidence for time-varying variances, this should result in few discards, though with $m$ large and slight evidence for differences among the $\sigma_j$'s, the result could be inefficiently many discards.

But we will not want to sample directly from this conditional likelihood in any case. when $H$ implies that a certain state $j$ should be rare, it can easily happen that a draw of the $\{S_t\}$ sequence contains no occurrences of that state. In that case $n(j) = s_j^2 = 0$, and the conditional posterior pdf for $\sigma_j$ is flat. It therefore cannot be normalized to be a proper pdf and we cannot draw from it. A proper prior pdf on the $\sigma$'s is therefore necessary so that the conditional posterior remains well-defined for states that fail to occur in a sampled $S$ sequence. A convenient way to introduce such a prior is to add a dummy observation for each state with a squared residual $u_0^2$, where $u_0$ is some reasonable guess as to the likely usual size of residuals for the model. This is equivalent to increasing each $n(j)$ count by 1 and each $s_j^2$ by $u_0^2$. If the data turn out to imply that some state or states are rare, however, this prior could influence results, so checks (e.g. by varying the choice of $u_0$) for how sensitive results are to the prior would be important.

# Software

A set of matlab functions is available on the course website that implement all the calculations described in the preceding sections. Here is a list of the functions, with descriptions of what they do. Note that for the exercise you are required to do for this course, only the last two of these programs (plus a function minimizer) is needed. The others are described here so that you will understand how the calculations are being done and so that you could modify them yourself to estimate some model other than the model of section (referred to henceforth as the PMSV model).

**[lh,sp=mclh(y,x,sig,b,H)** Generates likelihood and filtered state probabilities for given values of the parameters.

y: $T \times 1$ vector of left-hand-side variable data.

`x`: $T \times k$ matrix of right-hand-side variable data (lagged $r$ and a column of ones for the PMSV model).

`sig`: The $1 \times m$ vector of residual standard deviations, as a function of the state

`b`: The $k \times m$ matrix of coefficient vectors for the $m$ states (all columns identical for the PMSV model).

`H`: The $m \times m$ matrix of state transition probabilities.

`lh`: The $T \times 1$ vector of multiplicative likelihood increments. The product of these is the likelihood, the sum of their logs the log likelihood.

`sp:` The $T \times m$ matrix of filtered state probabilities. Each row is the conditional probability distribution for $S_t$ given $Y_t$.

**`smsp=mcsmooth(H,sp)`** Generates smoothed state probabilities from the filtered probabilities and the transition probability matrix.

`smsp:` The $T \times m$ matrix of smoothed state probabilities. Each row is the

conditional probability distribution for $S_t$ given $Y_T$.

**`spdraw=mcdraw(H,sp)`** Generates a draw from the posterior on $\{S_t\}$ from the transition matrix and the filtered state probabilities.

`spdraw:` A $T \times 1$ vector representing a draw from the posterior pdf of the $\{S_t\}$ sequence.

**`H=mcHdraw(spdraw,oldH)`** Generates a Metropolis-Hastings draw from the conditional posterior on the transition matrix $H$ from a given draw on $\{S_t\}$ and the previous value of $H$.

`H:` A Metropolis-Hastings draw from the conditional posterior on $H$.

`OldH:` The previous draw's value for $H$, which the Metropolis-Hastings algorithm may decide to repeat.

**b=mcbdraw(y,x,sig,spdraw)** Generates a draw from the conditional posterior of the regression parameter vector ($\alpha$ in the PMSV model).

b: A $k \times 1$ vector representing a draw from the posterior probability over the coefficient vector. Assumes that only $\sigma$, not the regression coefficients, varies with the state.

**sig=mcsdraw(y,x,b,sdraw)** Generates a draw from the conditional posterior on $\sigma$. Note that this routine incorporates a prior with a particular value of $u_0$ and may need adjustment and/or sensitivity analysis.

**[bdraw,sigdraw,pdraw,spsum,spssq] =mcexGibbs(y,x,b0,sig0,p0,nit)** Generates a set of Monte-Carlo draws from the joint posterior pdf on all the parameters of the PMSV model.

b0: A starting value for the $k \times 1$ coefficient vector.

`sig0:` A starting value for the $1 \times m$ vector of state-specific residual standard deviations.

`p0:` A starting value for the diagonal of the $H$ matrix. This assumes that $m = 2$, so that these two values determine the entire $H$ matrix.

`nit:` The number of Monte Carlo draws. On a 500Mhz Pentium III with 128k, 1000 draws for the PMSV model complete in a few minutes. 10000 take over an hour.

`bdraw:` `nit`$\times k$ matrix of draws on the regression coefficients.

`sigdraw:` `nit`$\times m$ matrix of draws on the $\{\sigma_j\}$ vector.

`pdraw:` `nit`$\times m$ matrix of draws of the diagonal of $H$. Here again, $m = 2$ is assumed.

`spsum:` $T \times 1$ vector of the sums of the draws of the smoothed estimate of the probability of state 1.

`spssq:` $T \times 1$ vector of the sums of squares of the draws of the probability of state 1. From `spsum` and `spssq` the posterior means and standard

errors of the state probabilities at each date can be constructed.

**`lh=mcexlh(z,y,x)`** Returns minus the log likelihood for the PMSV model, with all the parameters packed in to a single vector. This is the way the likelihood evaluation must be set up if the function is to be used as input to a function minimization routine, like `csminsel.m` or the similar routines packaged with matlab. Note that the program does not weight by the proper prior on $\sigma$ that is used in the Gibbs sampling program, so if it is used for likelihood maximization it does not find exactly the posterior mode of the distribution from which the Gibbs sampling program generates draws.

`z:` This is a $6 \times 1$ vector with elements $\alpha_0, \alpha_1, \sigma_1, \sigma_2, h_{11}, h_{22}$.