Normal linear regression model via Gibbs Sampling
Gibbs Sampler Diagnostics

AAEC 6564
Instructor: KLAUS MOELTNER

Textbooks: K Ch. 4, KPT Ch. 11
Matlab scripts: `mod2s1a, mod2_convergence_plots, mod2_convergence_plots2, mod2_plots, mod2_application, mod2_blocking, mod2_ac_plots, mod2_convergence_plots3, mod2_wetlands, mod2_wetlands2, mod2_wetlands_plot`

Matlab functions: `gs_independent_normal, gs_independent_normal_keepall, klausdiagnostics, klausdiagnostics_greater0, gs_normal_blocked, gs_normal_blocked_keepall`

As mentioned previously conjugate priors may be overly restrictive in many Bayesian applications. Here we use a popular combination of independent priors for the regression model – normal for $\boldsymbol{\beta}$ and inverse-gamma for σ^2.

The enhanced flexibility in prior modeling comes at the price of abandoning analytical results for the posterior distribution. Instead, we will use posterior simulation via Gibbs Sampling to obtain draws from the joint and marginal posteriors.

The structural model is the same as for the CLRM with conjugate priors.
.

$$p(\mathbf{y}\,|\,\boldsymbol{\theta},\mathbf{X}) = (2\pi)^{-n/2}\left(\sigma^2\right)^{-n/2}\exp\left(-\tfrac{1}{2\sigma^2}\left((\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)\right) \tag{1}$$

The priors are given as follows:

$$\begin{aligned}
&p\left(\boldsymbol{\beta},\sigma^2\right) = p(\boldsymbol{\beta})\,p\left(\sigma^2\right) \quad \text{where}\\
&\boldsymbol{\beta} \sim n(\boldsymbol{\mu}_0,\mathbf{V}_0), \quad \sigma^2 \sim ig(v_0,\tau_0)\\
&p(\boldsymbol{\beta}) = (2\pi)^{-k/2}\left|\mathbf{V}_0\right|^{-1/2}\exp\left(-\tfrac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\mu}_0)'\,\mathbf{V}_0^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_0)\right)\\
&p\left(\sigma^2\right) = \frac{{\tau_0}^{v_0}}{\Gamma(v_0)}\left(\sigma^2\right)^{-(v_0+1)}\exp\left(-\frac{\tau_0}{\sigma^2}\right), \quad \text{with} \quad E\left(\sigma^2\right) = \frac{\tau_0}{v_0-1}, \quad V\left(\sigma^2\right) = \frac{\tau_0^2}{(v_0-1)^2(v_0-2)}
\end{aligned} \tag{2}$$

Note that σ^2 does not enter the prior density of $\boldsymbol{\beta}$. As mentioned before, amongst the many possible parameterizations of the inverse-gamma (*ig*) density, we choose the form given in Gelman et al. (2004), where v_0 is the shape parameter and τ_0 is the scale parameter. For the density to have a defined mean, we need $v_0 > 1$, and for a well-defined variance we need $v_0 > 2$.

Combining the priors with the likelihood, and dropping all terms that are multiplicatively unrelated to our parameters of interest yields the posterior kernel

$$p\left(\boldsymbol{\beta},\sigma^2 \mid \mathbf{y},\mathbf{X}\right)\propto \left(\sigma^2\right)^{\frac{-n-2v_0-2}{2}}\exp\left(-\tfrac{1}{2\sigma^2}\left(2\tau_0\right)\right)\exp\left(-\tfrac{1}{2}\left(\tfrac{1}{\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)'\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\left(\boldsymbol{\beta}-\boldsymbol{\mu}_\mathbf{0}\right)'\mathbf{V}_\mathbf{0}^{\mathbf{-1}}\left(\boldsymbol{\beta}-\boldsymbol{\mu}_\mathbf{0}\right)\right)\right). \tag{3}$$

We first aim to find the posterior density for $\boldsymbol{\beta}$, conditional on σ^2 (i.e. treating σ^2 as a constant). Thus, we will fist focus on the components of the posterior kernel that cannot be multiplicatively separated from $\boldsymbol{\beta}$. This leaves

$$p\left(\boldsymbol{\beta} \mid \sigma^2,\mathbf{y},\mathbf{X}\right)\propto \exp\left(-\tfrac{1}{2}\left(\tfrac{1}{\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)'\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\left(\boldsymbol{\beta}-\boldsymbol{\mu}_\mathbf{0}\right)'\mathbf{V}_\mathbf{0}^{\mathbf{-1}}\left(\boldsymbol{\beta}-\boldsymbol{\mu}_\mathbf{0}\right)\right)\right). \tag{4}$$

Note the conditionality on σ^2 on both sides of (4). Using the same algebraic manipulations and reasoning as for the previous model, we obtain:

$$\boldsymbol{\beta} \mid \sigma^2,\mathbf{y},\mathbf{X} \sim n\left(\boldsymbol{\mu}_\mathbf{1},\mathbf{V}_\mathbf{1}\right) \quad \text{with} \quad \mathbf{V}_\mathbf{1}=\left(\mathbf{V}_\mathbf{0}^{\mathbf{-1}}+\tfrac{1}{\sigma^2}\mathbf{X}'\mathbf{X}\right)^{-1} \text{ and } \quad \boldsymbol{\mu}_\mathbf{1}=\mathbf{V}_\mathbf{1}\left(\mathbf{V}_\mathbf{0}^{\mathbf{-1}}\boldsymbol{\mu}_\mathbf{0}+\tfrac{1}{\sigma^2}\mathbf{X}'\mathbf{y}\right) \tag{5}$$

To derive the conditional posterior density for σ^2, we return to our original form for the joint posterior given in (3). Ignoring terms that are not related to σ^2, we have

$$p\left(\sigma^2 \mid \boldsymbol{\beta},\mathbf{y},\mathbf{X}\right)\propto \left(\sigma^2\right)^{\frac{-n-2v_0-2}{2}}\exp\left(-\tfrac{1}{2\sigma^2}\left(2\tau_0+\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)'\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)\right)\right). \tag{6}$$

Comparing this expression to the kernel of the *ig* prior in (2), we recognize this as the kernel of another *ig* density. Specifically:

$$\begin{aligned}&\sigma^2 \mid \boldsymbol{\beta},\mathbf{y},\mathbf{X} \sim ig\left(v_1,\tau_1\right) \quad \text{with}\\ &v_1=\frac{2v_0+n}{2} \qquad \text{and} \qquad \tau_1=\tau_0+\frac{1}{2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)'\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)\end{aligned} \tag{7}$$

Thus, it will be straightforward to draw $\boldsymbol{\beta}$ conditional on σ^2 and vice versa.

Gibbs Sampler

The Gibbs Sampler (GS) has become the "workhorse" of Bayesian posterior simulation in recent years. The general idea is simple: break the joint posterior into conditional posteriors for which the analytical form of its density is known. Then sample sequentially and repeatedly from these conditionals. After a number of draws the joint sequence of conditional draws will converge to the desired joint posterior densities for all parameter. In addition, each individual sequence can be interpreted as the marginal posterior for a given parameter. The GS is an example of a "Markov Chain Monte Carlo", or MC^2 procedure.

Formally, assume our parameter vector of interest is $\boldsymbol{\theta}$, with posterior kernel $p\left(\boldsymbol{\theta} \mid \mathbf{y}\right)\propto p\left(\boldsymbol{\theta}\right)p\left(\mathbf{y} \mid \boldsymbol{\theta}\right)$. Assume further that this kernel has no known analytical form. However, we can split $\boldsymbol{\theta}$ into two components, say $\boldsymbol{\theta}_\mathbf{1}$ and $\boldsymbol{\theta}_\mathbf{2}$. For example, in our current application, $\boldsymbol{\theta}_\mathbf{1}=\boldsymbol{\beta}$ and $\boldsymbol{\theta}_\mathbf{2}=\sigma^2$. In other applications, we may want to split $\boldsymbol{\theta}$ into 3, 4, or even more components. The key notion is that we know the analytical form of the resulting *full conditional posterior distributions*, i.e.

$$p\left(\boldsymbol{\theta}_1 \mid \mathbf{y}, \boldsymbol{\theta}_2\right) \qquad \text{and} \qquad p\left(\boldsymbol{\theta}_2 \mid \mathbf{y}, \boldsymbol{\theta}_1\right). \tag{3.8}$$

All we need to get the GS started is an initial value for $\boldsymbol{\theta}_2$, call it $\boldsymbol{\theta}_2^0$. This can be chosen arbitrarily, or one can use OLS results or results from previous analyses. We assume this starting value comes directly from the marginal posterior $p\left(\boldsymbol{\theta}_2 \mid y\right)$. Next, we draw $\boldsymbol{\theta}_1$ conditional on $\boldsymbol{\theta}_2^0$ from $p\left(\boldsymbol{\theta}_1 \mid \mathbf{y}, \boldsymbol{\theta}_2^0\right)$. Call this draw $\boldsymbol{\theta}_1^1$. Next, we draw another value of $\boldsymbol{\theta}_2$ conditional on $\boldsymbol{\theta}_1^1$ from $p\left(\boldsymbol{\theta}_2 \mid \mathbf{y}, \boldsymbol{\theta}_1^1\right)$. Call this draw $\boldsymbol{\theta}_2^1$. We repeat this process *R* times. In essence, we use the basic rule of conditional probabilities , i.e.

$$p\left(\boldsymbol{\theta} \mid \mathbf{y}\right) = p\left(\boldsymbol{\theta}_1 \mid \mathbf{y}, \boldsymbol{\theta}_2\right) p\left(\boldsymbol{\theta}_2 \mid \mathbf{y}\right) = p\left(\boldsymbol{\theta}_2 \mid \mathbf{y}, \boldsymbol{\theta}_1\right) p\left(\boldsymbol{\theta}_1 \mid \mathbf{y}\right) \tag{3.9}$$

over and over again. As a caveat we should note that, naturally, there is no guarantee that our starting value $\boldsymbol{\theta}_2^0$ really came from the marginal posterior $p\left(\boldsymbol{\theta}_2 \mid y\right)$. However, under relatively weak conditions (see Koop Ch. 4 for details) the starting value(s) will not matter and the GS will indeed converge to draws from $p\left(\boldsymbol{\theta} \mid \mathbf{y}\right)$. To assure that the effect of the starting value has truly "faded away", we usually discard the first r_1 draws of the sequence, and keep only the remaining $r_2 = R - r_1$ draws. The discarded draws are often referred to as "burn-ins".

Monte Carlo Integration

The derivation of moments from this simulated posterior is accomplished via Monte Carlo Integration. For example, the analytical expressions for the mean (or expectation) and variance of a given element of $\boldsymbol{\beta}$, say β_j, are given by

$$E\left(\beta_j\right) = \int \beta_j \, p\left(\beta_j \mid \mathbf{y}, \mathbf{X}\right) d\beta_j \qquad\qquad V\left(\beta_j\right) = \int \left(\beta_j - E\left(\beta_j\right)\right)^2 p\left(\beta_j \mid \mathbf{y}, \mathbf{X}\right) d\beta_j$$

A convenient alternative expression for the variance is

$$V\left(\beta_j\right) = E\left(\beta_j^2\right) - \left(E\left(\beta_j\right)\right)^2 \qquad \textit{where} \quad E\left(\beta_j^2\right) = \int \beta_j^2 \, p\left(\beta_j \mid \mathbf{y}, \mathbf{X}\right) d\beta_j$$

Also, the expectation and variance of any other function of β_j , say $g\left(\beta_j\right)$, take the form of

$$E\left(g\left(\beta_j\right)\right) = \int g\left(\beta_j\right) p\left(\beta_j \mid \mathbf{y}, \mathbf{X}\right) d\beta_j \qquad\qquad V\left(g\left(\beta_j\right)\right) = \int \left(g\left(\beta_j\right) - E\left(g\left(\beta_j\right)\right)\right)^2 p\left(\beta_j \mid \mathbf{y}, \mathbf{X}\right) d\beta_j$$

For any of these expressions, MCI approximates the integral with averaging over draws. Thus

$$E\left(\beta_j\right) \approx \frac{1}{R}\sum \beta_{j,r}$$

$$E\left(\beta_j^2\right) \approx \frac{1}{R}\sum \beta_{j,r}^2$$

$$E\left(g\left(\beta_j\right)\right) \approx \frac{1}{R}\sum g\left(\beta_{j,r}\right)$$

Implementation with Simulated Data

Matlab script `mod2s1a` is the main script for this model. It opens the log file, loads data, specifies priors and other settings for the posterior simulator, and calls the Gibbs Sampler (GS) via the function `gs_normal_independent`. The last part of the script applies diagnostic tools (discussed later) to the posterior draws and formats all output for the log file.

Convergence Plots

Convergence plots are a simple visual tool to examine if the simulator has converged to the posterior distribution for a given parameter. The plot simply shows all draws of a given parameter in chronological order, as generated by the sampler. "Convergence" usually implies that the draws are tightly clustered around a flat line (the posterior mean), and do not wander widely. This is similar to assessing stationarity for time series data.

Convergence plots can be used to assess the sensitivity of the algorithm to starting draws (the chain should converge to the same distribution under different starting draws), the speed of convergence (and thus the efficiency of the sampler), and the sufficiency of the chosen number of discarded draws (burn-ins).

Script `mod2_convergence plots` provides a few examples using the simulated data from `mod2s1a`. There are three parts: Full data, data truncated to 100 observations, and data truncated to 10 observations. For each case, we first use two sets of starting draws for $\boldsymbol{\beta}$ and σ^2. The first set comes from the OLS output and is thus "right on target", i.e. close to the posterior mean (and the true parameter values underlying our simulated data). The second set is deliberately located quite far from the true values.

For all cases we use a slightly modified GS via function `gs_normal_independent_keepall.` As opposed to the previous version, this algorithm preserves the burn-in draws and sends them back to the main script along with the "keepers" (i.e. retained draws from before).

You can see that with the full data set of 10,000 observations, the choice of starting draws virtually doesn't matter – the chain essentially converges after one or two iterations. Furthermore, parameter draws fluctuate very tightly around the posterior mean. A reduction in sample size implicitly assigns more weight to our diffuse priors. As a result, the chain exhibits more "noise", i.e. wider fluctuations around the mean. This will translate into a larger posterior standard deviation. However, even with just a handful of observations, convergence is virtually immediate even with off-target starting draws. This is an indication that the sampler itself is efficient, i.e. "mixes rapidly".

Posterior noise (i.e. the posterior standard deviation) is also driven by the information content of the data, regardless of sample size. Highly collinear data implies poor information content and will generate wider fluctuations around the posterior mean. This is illustrated in script `mod2_convergence plots2.` If you compare the plots from this script to the ones from before for any parameter and sample size, you will notice the increase in variability around the posterior mean for the collinear data.

Prior vs. Posterior Plots

As for the conjugate prior case, it is illustrative to plot and compare the prior and posterior distributions for our parameters. This is accomplished in script `mod2_plots`. As before, the posterior densities are much tighter than the priors in all cases.

Application: Female Earnings

Matlab script `mod2_application` implements the normal linear regression model with independent priors using Mroz's (1987) labor data and provides posterior plots for selected parameters.

Gibbs Sampler Diagnostics

This section describes in detail the output component generated by function "`klaus_diagnostics`", which is applied to parameter draws generated by the posterior simulator.

Numerical standard error (nse)

The *nse* captures simulation noise for a value of interest generated via posterior simulation. Usually this value of interest is a measure of central tendency, such as the posterior mean. Consider the mean $\overline{\theta} = \frac{1}{M}\sum_{m=1}^{M}\theta_m$ of a sequence of *m* draws of (some generic) parameter θ. Assume these draws were generated by a Gibbs Sampler (GS) or some other Markov-Chain Monte Carlo (MC^2) algorithm. If these draws were perfectly independently and identically distributed (i.i.d.) with sample variance s^2 , we could quickly derive the *nse* for $\overline{\theta}$ using the basic formula for the standard error of a sample mean, i.e.

$$nse\left(\overline{\theta}\right) = \sqrt{V\left(\overline{\theta}\right)} = \sqrt{\frac{1}{M^2}Ms^2} = \frac{s}{\sqrt{M}} \tag{1}$$

However, as with all MC^2 procedures, we have to ex ante assume that these sequential draws will have a considerable degree of *correlation*. This means we have to consider all covariance terms between all draws of θ. After some straightforward analytical simplifications (shown in detail in KPT, p. 145), we end up with a more general expression for the *nse* that allows for correlation across all draws:

$$nse\left(\overline{\theta}\right) = \sqrt{V\left(\overline{\theta}\right)} = \sqrt{\frac{s^2}{M}\left(1+2\sum_{j=1}^{M-1}\left(1-\frac{j}{M}\right)\rho_j\right)}, \tag{2}$$

where $\rho_j = \frac{s_j}{s^2}$ is the lag-correlation between some draw θ_i and a draw that was obtained j iterations prior to θ_i, i.e. θ_{i-j}, with s_j denoting the associated sample covariance. It can be easily seen from (2) that under perfect independence ($\rho_j = 0, \forall j$) we arrive again at the basic formula given in (1). In most MC^2 applications, lag-correlations will be positive but declining in magnitude. Thus, the second term under the square root in (2) will exceed 1, and the *nse* under correlation will be larger that the *nse* under independence. Note that in either case the *nse* can be made arbitrarily small by increasing *M*, the number of draws. However, this can be very costly in terms of computation time. Thus, we are always interested in devising a posterior sampler that is "efficient" in the sense of generating draws with low lag-correlation. There are many "tricks" for accomplishing this – we'll touch upon a few in this course.

In summary, the first purpose of the *nse* is to provide a measure of "simulation error" or "simulation noise" surrounding a posterior construct of interest, usually the posterior mean of a given parameter. Thus, the *nse* has a similar function to the standard error (s.e.) in Classical Analysis. However, its intuition is very different – it simply captures simulation noise, i.e. the penalty for having to approximate the joint posterior via simulations (since its analytical form is unknown). The Classical s.e. conveys the notion of *sampling error* – i.e. the variability of the statistical construct of interest under (hypothetical) re-sampling. You can also think of this as the penalty from working with a small sample relative to a large population.

Inefficiency factors (IEFs)

The second purpose of the *nse* is to provide a summary measure of the "efficiency" of the posterior simulator. An efficient simulator will have low correlation across draws, and thus will be able to "tell the same story with fewer draws" – saving valuable computing time. As discussed in Chib (2001, section 3.2) the ratio of the squared *nse* under correlation over the squared *nse* under independence can be interpreted as "*inefficiency factor*" (IEF), also known as "*autocorrelation time*" for a given parameter, i.e.

$$IEF(\theta) = \frac{nse^2\left(\bar{\theta}\right)}{nse^2\left(\bar{\theta}; \rho_j = 0, \forall j\right)} = 1 + 2\sum_{j=1}^{M-1}\left(1 - \tfrac{j}{M}\right)\rho_j \tag{3}$$

A well-designed posterior simulator will generate sequences of parameter draws with low IEFs, the ideal being an IEF close to 1. Sometimes the inverse of the IEF is used to measure posterior efficiency. This quantity is called "*numerical efficiency*" (See Geweke, 1992). A third quantity to assess efficiency is the "i.i.d - *equivalent number of iterations*", labeled *M** in the following, i.e. the number of i.i.d. draws that contain the same amount of information about θ as the observed number of draws under correlation. It is easily derived via

$$\frac{s^2}{M^*} = \frac{s^2}{M}\left(1 + 2\sum_{j=1}^{M-1}\left(1 - \tfrac{j}{M}\right)\rho_j\right) \rightarrow M^* = \frac{M}{\left(1 + 2\sum_{j=1}^{M-1}\left(1 - \tfrac{j}{M}\right)\rho_j\right)} = \frac{M}{IEF} \tag{4}$$

It follows that under perfect efficiency, we have $M^* = M$, but usually we observe $M^* < M$.

It should be noted that a very high IEF (very low M*) can be indicative of identification problems in your model. How high is "very high? From personal experience, I would say that IEFs in the 1-5 range indicate good efficiency, in the 6-20 range they're still "tolerable", but anything above 20 deserves closer inspection. Certainly, IEFs of 100 and higher are almost a sure-bet indication of an identification or specification problem in the underlying structural model.

Geweke's (1992) Convergence Diagnostics(CD)

Another important sampler diagnostic is Geweke's (1992) *CD* score. It is based on the simple intuition that if the entire sequence of retained $\theta's$ (focusing on a single parameter for simplicity) can truly be interpreted as random draws from the same posterior density $p\left(\theta \mid \mathbf{y}\right)$, and we divide the sequence of R draws into three segments, the mean of the first segment of $r = 1 \cdots R_1$ draws should be "not too different" from the mean of the last segment of $r = R_2 + 1 \cdots R$ draws. A stated in Koop Ch. 4, setting $R_1 = 0.1R$ and $R_2 = 0.6R$ produces adequate results for most applications. Define the two means as $\bar{\theta}_1$ and $\bar{\theta}_2$, respectively. Then, asymptotically, the difference between these means, weighted by their respective numerical standard errors, converges to a standard normal ("z") variate, i.e.

$$CD = \frac{\bar{\theta}_1 - \bar{\theta}_2}{\sqrt{nse_1^2 + nse_2^2}} \overset{a}{\sim} n\left(0,1\right). \tag{4.5}$$

Thus, a CD value that clearly exceeds 1.96 for a specific parameter θ would raise a flag – it indicates that the sequence of posterior draws may not have converged to $p\left(\theta \mid \mathbf{y}\right)$. In practice, if a few CD values in the 2-2.5 range for a model with many parameters would hardly raise concerns. However, if your posterior simulator generates CD values of 3 or higher, an increase in the number of burn-in draws may be warranted. Similarly to IEFs, grossly inflated CD values may also indicate identification and / or mis-specification problems in the underlying model.

The Matlab function "`klausdiagnostics`" automatically generates the following posterior statistics for all model parameters: posterior mean, posterior standard deviation, nse, IEF, M*, and CD. This is illuatrated in our application of the normal regression model with independent priors. The Matlab function "`klausdiagnostics_greater0`" produces, in addition, the posterior probability for a parameter to exceed zero. This conveys a quick picture as to where the bulk of the posterior is located vis-a-vis zero. The classically trained reader can relate to this quite well as it provides "similarly flavored" information to the standard t-statistic or *p*-value in a typical regression output.

Autocorrelation (AC) plots

There are two additional noteworthy diagnostics tools: autocorrelation plots ("AC plots") and re-running the posterior sampler with different starting values, as implemented in Session 6 of this course. An AC plot provides a simple visual inspection of the lag-correlation terms (ρ_j in (2)) for a given parameter.

An example for AC plots is provided in the Matlab script `mod2_ac_plots`. A "well-behaved" AC plot will exhibit small correlation effects that randomly fluctuate around zero. A plot with high (usually positive) correlations that taper off only very slowly with increasing lag would be indicative of inefficiencies in the posterior simulator.

Blocking

As discussed in previous Sessions of this course, a Gibbs Sampler operates by splitting the full set of parameters into different groups or "*blocks*", which are then drawn sequentially and repeatedly, conditional on *all other* blocks.

The main consideration in designing these blocks for a standard Gibbs Sampler is that the conditional posterior density for each block is known, else we wouldn't be able to take any draws from it. However, this requirement can be relaxed by employing other posterior simulation techniques, such as the Metropolis Hastings (MH) algorithm, which does not require full knowledge of the conditional posterior density for a given parameter or block of parameters. Thus, the "optimal blocking" of the full set of parameters becomes a more general question.

As discussed inter alia in Chib (2001, section 7.1) it is generally recommended that parameters be drawn in as few blocks as possible, and that parameters that tend to be highly correlated be collected in the same block.

Identifying what constitutes a full-fledged block in a given posterior algorithm can be tricky. This is because blocks can be combined by the method of composition, i.e. by exploiting the fact that *partially* conditional posterior distributions may be known (or can be approximated at low computational cost) for some parameters. For example, consider an initial blocking of the full parameter vector into three groups, $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, and $\boldsymbol{\theta}_3$. A "naïve" posterior sampler will then operate as follows:

1. Draw $\boldsymbol{\theta}_1$ from $p\left(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \mathbf{y}\right)$, where $\mathbf{y}$ represents the available data.
2. Draw $\boldsymbol{\theta}_2$ from $p\left(\boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_1, \boldsymbol{\theta}_3, \mathbf{y}\right)$
3. Draw $\boldsymbol{\theta}_3$ from $p\left(\boldsymbol{\theta}_3 \mid \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{y}\right)$
4. Repeat.

Now suppose the partially conditional density $p\left(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_3, \mathbf{y}\right)$ is known or can be approximated at low computational cost. A (likely) more efficient posterior sampler would then collect $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ in a single block and operate as follows:

1. Draw $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ from $p\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_3, \mathbf{y}\right) = p\left(\boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_1, \boldsymbol{\theta}_3, \mathbf{y}\right) * p\left(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_3, \mathbf{y}\right)$ as follows:
    a. Draw $\boldsymbol{\theta}_1$ from $p\left(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_3, \mathbf{y}\right)$
    b. Draw $\boldsymbol{\theta}_2$ from $p\left(\boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_1, \boldsymbol{\theta}_3, \mathbf{y}\right)$
2. Draw $\boldsymbol{\theta}_3$ from $p\left(\boldsymbol{\theta}_3 \mid \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathbf{y}\right)$

Thus, even though step 1 involves 2 sub-steps, it is considered a single block. The key notion in identifying the number of blocks is that for a set of parameters to constitute a self-standing block, it needs to be drawn conditional on *all other* blocks, and *vice versa*, i.e. all other blocks also need to be conditioned on the first block.

Blocking in the normal linear regression model

For the normal regression model we have considered so far the parameter blocking was quite obvious and, as it turns out, efficient: We grouped the constant term and all slope parameters into a single block (

$\boldsymbol{\beta}$), which left the regression variance σ^2 as the only other (single-parameter) block. Our GS proceeded as follows:

1. Draw $\boldsymbol{\beta}$ from $p\left(\boldsymbol{\beta} \mid \sigma^2, \mathbf{y}, \mathbf{X}\right)$
2. Draw σ^2 from $p\left(\sigma^2 \mid \boldsymbol{\beta}, \mathbf{y}, \mathbf{X}\right)$

Matlab script `mod2_blocking` proposes a different approach based on 4 blocks. Specifically, we split $\boldsymbol{\beta}$ into three parts of equal length, labeled $\boldsymbol{\beta_1}$, $\boldsymbol{\beta_2}$, and $\boldsymbol{\beta_3}$. We then proceed as follows, using function `gs_normal_blocked`:

1. Draw $\boldsymbol{\beta_1}$ from $p\left(\boldsymbol{\beta_1} \mid \boldsymbol{\beta_2}, \boldsymbol{\beta_3}, \sigma^2, \mathbf{y}, \mathbf{X}\right)$
2. Draw $\boldsymbol{\beta_2}$ from $p\left(\boldsymbol{\beta_2} \mid \boldsymbol{\beta_1}, \boldsymbol{\beta_3}, \sigma^2, \mathbf{y}, \mathbf{X}\right)$
3. Draw $\boldsymbol{\beta_3}$ from $p\left(\boldsymbol{\beta_3} \mid \boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \sigma^2, \mathbf{y}, \mathbf{X}\right)$
4. Draw σ^2 from $p\left(\sigma^2 \mid \boldsymbol{\beta}, \mathbf{y}, \mathbf{X}\right)$

In practice, draws of $\boldsymbol{\beta_j}, j = 1 \cdots 3$ can be obtained as follows:
We know that for the basic linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{6}$$

we obtain conditional draws of $\boldsymbol{\beta}$ via

$$\boldsymbol{\beta} \mid \sigma^2, \mathbf{y}, \mathbf{X} \sim n\left(\boldsymbol{\mu_1}, \mathbf{V_1}\right) \quad \text{with} \quad \mathbf{V_1} = \left(\mathbf{V_0^{-1}} + \tfrac{1}{\sigma^2}\mathbf{X}'\mathbf{X}\right)^{-1} \text{ and } \quad \boldsymbol{\mu_1} = \mathbf{V_1}\left(\mathbf{V_0^{-1}}\boldsymbol{\mu_0} + \tfrac{1}{\sigma^2}\mathbf{X}'\mathbf{y}\right) \tag{7}$$

Now partition X and $\boldsymbol{\beta}$ into three parts corresponding to our new blocking, i.e.

$$\mathbf{y} = \mathbf{X_1}\boldsymbol{\beta_1} + \mathbf{X_2}\boldsymbol{\beta_2} + \mathbf{X_3}\boldsymbol{\beta_3} + \boldsymbol{\varepsilon} \tag{8}$$

Consider the "transformed" dependent variable $\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{X_2}\boldsymbol{\beta_2} - \mathbf{X_3}\boldsymbol{\beta_3}$ and the resulting modified regression model

$$\tilde{\mathbf{y}} = \mathbf{X_1}\boldsymbol{\beta_1} + \boldsymbol{\varepsilon} \tag{9}$$

We can thus derive the conditional posterior for $\boldsymbol{\beta_1}$ as

$$\begin{gathered}\boldsymbol{\beta_1} \mid \boldsymbol{\beta_2}, \boldsymbol{\beta_3}, \sigma^2, \mathbf{y}, \mathbf{X} \sim n\left(\boldsymbol{\mu_1}, \mathbf{V_1}\right) \quad \text{with} \quad \mathbf{V_1} = \left(\mathbf{V_{01}^{-1}} + \tfrac{1}{\sigma^2}\mathbf{X_1'}\mathbf{X_1}\right)^{-1} \text{ and} \\ \boldsymbol{\mu_1} = \mathbf{V_1}\left(\mathbf{V_{01}^{-1}}\boldsymbol{\mu_{01}} + \tfrac{1}{\sigma^2}\mathbf{X_1'}\tilde{\mathbf{y}}\right) = \mathbf{V_1}\left(\mathbf{V_{01}^{-1}}\boldsymbol{\mu_{01}} + \tfrac{1}{\sigma^2}\mathbf{X}'\left(\mathbf{y} - \mathbf{X_2}\boldsymbol{\beta_2} - \mathbf{X_3}\boldsymbol{\beta_3}\right)\right)\end{gathered} \tag{10}$$

where $\boldsymbol{\mu_{01}}$ and $\mathbf{V_{01}}$ are the prior mean and variance for $\boldsymbol{\beta_1}$. Draws of $\boldsymbol{\beta_2}$ and $\boldsymbol{\beta_3}$ can be obtained in analogous fashion.

The posterior output clearly shows efficiency losses compared to the original version, as judged by IEF and M* scores. This is depicted graphically in script `mod2_ac_plots`, which compares autocorrelation plots for the original and the "excessively-blocked" version for selected parameters.

Also note the loss in speed – the inefficient sampler takes about twice as long to take the same number of draws.

We can also compare the relative performance of the two samplers based on convergence plots, as implemented in script `mod2_convergence_plots3.` The plotted chain of draws for "age" is especially illustrative: the chain wanders widely, with a clear auto-correlation pattern. The main drawback of such a highly correlated chain is that it takes much longer to "visit" the entire posterior distribution with appropriate frequencies . This may result in misleading posterior inference, based on overly tight or otherwise "incomplete" distributions. (In fact, the posterior standard deviations flowing from the inefficient sampler are actually slightly smaller than those generated by the efficient sampler) .

Effect of Priors under Small Sample Sizes

As you may expect, the choice of priors becomes especially important under small sample sizes, where the data will be much less dominant in shaping the posterior distribution.

As mentioned previously, this potentially pronounced effect of prior distributions and parameters on "final results" is both a curse and a blessing. It requires great care in prior selection, but it also allows for the introduction of pertinent information that is exogenous to the (small) data at hand.

With our diagnostic tools in hand, we are now well-equipped to take a closer look at the role of priors in small-sample application.

Matlab script `mod2_wetlands` applies the normal linear regression model with independent priors to a small data set of 12 observations as used in Moeltner and Woodward (2009).

This is an example of a *meta-dataset*, compiled from results and summary statistics reported in existing studies. This secondary data set is then processed via meta-regression to yield insights into potential outcomes associated with a new policy site or setting. This strategy is commonly referred to as "Benefit Transfer", and constitutes a low-cost alternative to primary data collection.

For our purposes, think of this application simply as a regression model with a very small sample. The outcome variable of interest is the average willingness to pay (per year) across a group of residents ("sub-population") to preserve a specific wetland area. Explanatory variables are the percentage of active wetland users in the sub-population, average annual household income (in log form), and wetland size (in 1000 acres).

Script `mod2_wetlands` estimates the model using our efficient 2-block Gibbs Sampler and vague priors for all parameters. The posterior output suggests convergence (based on CD scores) and good efficiency (based on IEF scores). As expected, the posterior standard deviations are relatively large (say, compared to the posterior mean), a direct effect of the vague priors.

Script `mod2_wetlands2` implements the same model with informed priors based on estimated parameters reported in the literature, as described in Moeltner and Woodward (2009). Again, the diagnostics point at convergence and good efficiency. The posterior standard deviations are smaller than in the original model, which is unambiguously desirable.

However, the posterior means have changed as well, which leads to changed inference on expected marginal effects. For example, the original model produces a posterior mean for income elasticity of 0.288. This increases to 0.388 in the refined model.

Script `mod2_wetlands_plots` compares the posterior and prior densities across the two models for a selected set of parameters. Note that there are efficiency spillovers even for parameters that themselves did not receive informed priors (such as the marginal effect of "users").

References

Chib, Siddartha. 2001. "Markov Chain Monte Carlo Methods: Computation and Inference," in J. J. Heckman and E. Leamer (eds), *Handbook of Econometrics*: Elsevier.

Geweke, J. 1992. "Evaluating the Accuracy of Sampling-based Approaches to the Calculation of Posterior Moments," in J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith (eds), *Bayesian Statistics 4* Oxford, UK: Oxford University Press.

Moeltner, K. and R. Woodward. 2009. "Meta-Functional Benefit Transfer for Wetland Valuation: Making the Most of Small Samples." *Environmental and Resource Economics* 42, 89-109.