Applied Bayesian Computational Methods

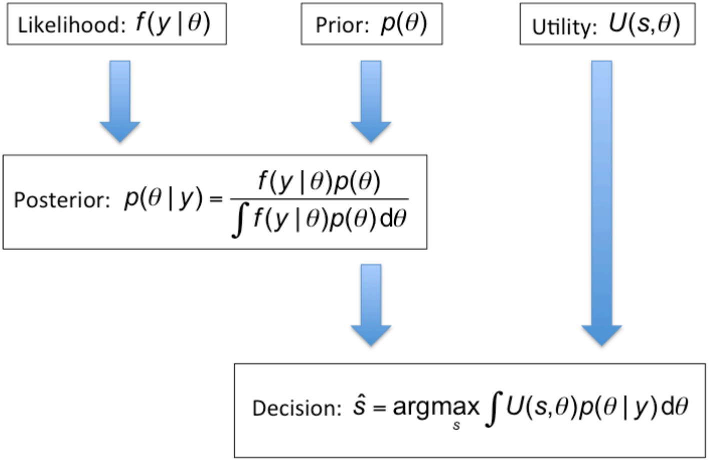
Abel Rodríguez (UC Santa Cruz)
Christopher Paciorek (UC Berkeley)

JSM Baltimore
July, 2017

Learning objectives

- Understand the landscape of current computational methods for Bayesian inference;
- Understand the statistical and probabilistic principles behind the methods;
- Understand the various MCMC alternatives and how to assess MCMC performance;
- Be able to choose a method and software tool for a real-world problem; and
- Be able to implement Bayesian inference using NIMBLE and Stan.

The Bayesian approach to statistical inference


Likelihood: $f(y \mid \theta)$
Prior: $p(\theta)$
Utility: $U(s,\theta)$
Posterior: $p(\theta \mid y) = \frac{f(y \mid \theta)p(\theta)}{\int f(y \mid \theta)p(\theta)\,d\theta}$
Decision: $\hat{s} = \underset{s}{\operatorname{argmax}} \int U(s,\theta)p(\theta \mid y)\,d\theta$

The Bayesian approach to statistical inference

- **Example (estimating the mean of a normal):** Assume $y_1, \dots, y_n$ is an independent and identically distributed sample with $y_i \mid \theta \sim \mathsf{N}(\theta, 1)$ and assign θ a Gaussian prior, i.e., $\theta \sim \mathsf{N}(\mu, \tau^2)$. Our goal is to provide (Bayesian) point and interval estimates for the unknown parameter θ.

  The first stage in any Bayesian analysis is to determine the posterior density/probability mass function of the parameter(s) of interest (in this case θ).

The Bayesian approach to statistical inference

- **Example (estimating the mean of a normal, cont):**

$$p(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{\left(\frac{1}{2\pi}\right)^{n/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(y_i-\theta)^2\right\}\left(\frac{1}{2\pi\tau^2}\right)^{1/2}\exp\left\{-\frac{(\theta-\mu)^2}{2\tau^2}\right\}}{\int_{-\infty}^{\infty}\left(\frac{1}{2\pi}\right)^{n/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(y_i-\theta)^2\right\}\left(\frac{1}{2\pi\tau^2}\right)^{1/2}\exp\left\{-\frac{(\theta-\mu)^2}{2\tau^2}\right\}\mathrm{d}\theta}$$

Doing a completion of squares we get

$$p(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{\exp\left\{-\frac{1}{2}\left(n+\frac{1}{\tau^2}\right)\left(\theta-\frac{n\bar{y}+\frac{\mu}{\tau^2}}{n+\frac{1}{\tau^2}}\right)^2+\frac{1}{2}\frac{\left(n\bar{y}+\frac{\mu}{\tau^2}\right)^2}{n+\frac{1}{\tau^2}}\right\}}{\int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2}\left(n+\frac{1}{\tau^2}\right)\left(\theta-\frac{n\bar{y}+\frac{\mu}{\tau^2}}{n+\frac{1}{\tau^2}}\right)^2+\frac{1}{2}\frac{\left(n\bar{y}+\frac{\mu}{\tau^2}\right)^2}{n+\frac{1}{\tau^2}}\right\}\mathrm{d}\theta}$$

Hence $p(\theta \mid \mathbf{y})$ correspond to a Gaussian distribution with mean $\frac{n\bar{y}+\frac{\mu}{\tau^2}}{n+\frac{1}{\tau^2}}$ and variance $\left(n+\frac{1}{\tau^2}\right)^{-1}$!

The Bayesian approach to statistical inference

- **Example (estimating the mean of a normal, cont):** Now that we have found the posterior distribution, we can use the posterior mean as a point estimator, i.e.,

$$\tilde{\theta} = \mathrm{E}\left\{\theta \mid \mathbf{y}\right\} = \frac{\int \theta p(\mathbf{y} \mid \theta) p(\theta) \mathrm{d}\theta}{\int p(\mathbf{y} \mid \theta) p(\theta) \mathrm{d}\theta} = \frac{n\bar{y} + \frac{\mu}{\tau^2}}{n + \frac{1}{\tau^2}}.$$

Besides being a "natural" measure of location of the posterior distribution, the posterior mean can be formally justified as begin the optimal estimator under squared error loss,

$$\begin{aligned} \mathrm{E}\left\{\theta \mid \mathbf{y}\right\} &= \int_{-\infty}^{\infty} \theta p(\theta \mid \mathbf{y}) \mathrm{d}\theta \\ &= \operatorname{argmax}_{\vartheta} \frac{\int (\theta - \vartheta)^2 p(\mathbf{y} \mid \theta) p(\theta) \mathrm{d}\theta}{\int p(\mathbf{y} \mid \theta) p(\theta) \mathrm{d}\theta} \end{aligned}$$

The Bayesian approach to statistical inference

- **Example (estimating the mean of a normal, cont):** Similarly, interval estimates can be obtained by computing highest posterior density intervals (shortest intervals that contain a given amount of posterior probability).

  In our running example, and for an interval with $1 - \alpha$ coverage, this reduces to

$$\left(\frac{n\bar{y} + \frac{\mu}{\tau^2}}{n + \frac{1}{\tau^2}} - z_{\alpha/2} \left\{ n + \frac{1}{\tau^2} \right\}^{-\frac{1}{2}}, \frac{n\bar{y} + \frac{\mu}{\tau^2}}{n + \frac{1}{\tau^2}} + z_{\alpha/2} \left\{ n + \frac{1}{\tau^2} \right\}^{-\frac{1}{2}} \right).$$

  HPD intervals can be justified using utility functions too!

The Bayesian approach to statistical inference

- Generally speaking, once the model for the data and the priors/hyperpriors have been selected, performing Bayesian inference involves
  - Computing expectations of functions of the parameters with respect to the posterior distribution.
  - Maximizing the corresponding expected utility functions.
- We will mostly be focusing on cases where the second step is available in closed form (i.e., we assume simple, “standard” utility functions).
  - Historically, the main challenge in using Bayesian methods for real-world problems has been how to do computation beyond simple conjugate models.
  - Numerical integration methods such as Gaussian quadrature scale very poorly (typically not useful if $\dim(\boldsymbol{\theta}) > 4$).

A summary of Bayesian computational methods

Monte-Carlo integration

- Generate samples from $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(B)} \sim p(\boldsymbol{\theta} \mid \mathbf{y})$.
- Approximate

$$\int h(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta} \approx \sum_{b=1}^{B} h\left(\boldsymbol{\theta}^{(b)}\right)$$

  (WLLN / Ergodic theorem).
- Examples: Adaptive Rejection Sampling, Importance Sampling, Markov chain Monte Carlo, Sequential Monte Carlo, Approximate Bayesian Computation.

Approximation methods

- Find a distribution $q(\boldsymbol{\theta})$ that is analytically tractable and is "close" to $p(\boldsymbol{\theta} \mid \mathbf{y})$.
- Approximate

$$\int h(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta} \approx \int h(\boldsymbol{\theta}) q(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

- Examples: Laplace approximations, mean-field variational approximations, integrated nested Laplace approximations.

A summary of Bayesian computational methods

- The previous methods are *general*, in the sense that they allow for point and interval estimation, prediction and, in some cases (e.g., reversible jump MCMC, spike-and-slab priorss) variable selection/model comparison.
- In addition to them, there are methods that are *specific* to either point estimation (typically finding posterior mode) or model selection (typically for computing marginal likelihoods and Bayes factors).
    - Point estimation: expectation-maximization (yes, it can be used in a Bayesian setting), (stochastic) conjugate gradient, feature selection.
    - Bayes factors: Harmonic Mean Estimation, Bridge Sampling, the marginal likelihood identity method of Chib, etc.

What this course is about!

- As you can see, the literature is extensive, so we will select a small number of topics to discuss:
    - MCMC techniques, including reversible jump, adaptive Monte Carlo, and Hamiltonian methods.
    - Importance sampling and sequential Monte Carlo algorithms for state-space models, but not population Monte Carlo.
    - Variational methods.
    - Laplace approximations and INLA
- Heavy focus on using `NIMBLE` and other software options to simplify implementation.

What is NIMBLE?

- A flexible extension of the BUGS and JAGS systems
- A system for using algorithms on hierarchical statistical models
- A system for programming algorithms for hierarchical models
- A partial compiler for math, flow control, and related R code

At its most basic level, you can use NIMBLE in place of BUGS or JAGS. But even if you don't use algorithms other than MCMC or take advantage of the NIMBLE programming system, NIMBLE provides additional flexibility on top of BUGS or JAGS:

- User-defined distributions and functions for use in BUGS code
- Alternative parameterizations of distributions
- Flexibility in choosing the samplers to use in an MCMC
- Ability to arbitrarily block parameters in MCMC sampling

The BUGS language

BUGS is a declarative language for graphical (or hierarchical) models.

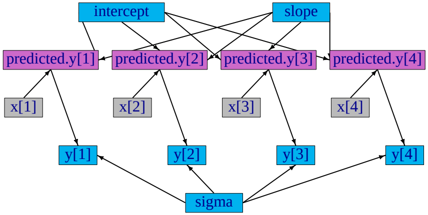
- Most programming languages are imperative, which means a series of commands will be executed in the order they are written.
- A declarative language like BUGS is more like building a machine before using it.
- Each line declares that a component should be plugged into the machine.

The machine in this case is a graphical model (in particular, a *directed acyclic graph*). A *node* (sometimes called a *vertex*) holds one value, which may be a scalar or a vector. *Edges* define the relationships between nodes. A huge variety of statistical models can be thought of as graphs. See also:

- Chapter 5 of the NIMBLE manual – http://r-nimble.org/documentation
- the BUGS manual – http://www.openbugs.net/Manuals/ModelSpecification.html

Basic regression example in BUGS

```
regrCode <- nimbleCode({
    ## priors
    intcpt ~ dnorm(0, sd = 1000)
    slope ~ dnorm(0, sd = 1000)
    ## Gelman (2006) recommended prior:
    sigma ~ dunif(0, 100)

    for(i in 1:4) {
        ## linear predictor (deterministic node)
        pred_y[i] <- intcpt + slope * x[i]
        ## likelihood
        y[i] ~ dnorm(pred_y[i], sd = sigma)
    }
})
```



Stochastic relationships are declared with $\sim$ and deterministic relationships with `<-`.

See Chapter 5 of the NIMBLE manual for available distributions, default and alternative parameterizations, and functions.

We don't actually need the `pred_y` line of code; could have:

```
y[i] ~ dnorm(intcpt + slope * x[i], sd = sigma)
```

GLMM example

- The litters model from the original 1995 BUGS software has data on rat pup survival for pups in 16 litters in each of two groups. Survival within a litter is governed by a survival probability specific to the litter.
- Probabilities for each group are treated as being exchangeable, coming from a common beta distribution to induce shrinkage.

```
glmmCode <- nimbleCode({
  for (i in 1:G) {
      ## priors for hyperparameters (from original example, not necessarily recommended)
     a[i] ~ dgamma(1, .001)
     b[i] ~ dgamma(1, .001)
     for (j in 1:N) {
        ## random effects ('latent process')
        p[i,j] ~ dbeta(a[i], b[i])
        ## likelihood
        r[i,j] ~ dbin(p[i,j], n[i,j])
     }
   }           })
```

- This is a GLMM structure (albeit without any explicit fixed effects, though they are implicit in the grouping).
- We could write an alternative version with a logistic link, normal random effects, and a group-based offset.

Models in NIMBLE

In WinBUGS and JAGS, the model and its MCMC are tied together. NIMBLE's approach is to separate the model and the algorithm(s) used with a model. This allows for:

- use of the model for other purposes, such as simulating from it in a simulation study,
- customization of MCMCs for a given model (e.g., changing samplers and blocking),
- use of other algorithms (e.g., SMC, MCEM, eventually INLA, VB, etc.) on a model, and
- programming new algorithms for use with any model.

Let's create a model from the BUGS code.

```
consts <- list(G = 2, N = 16,
     n = matrix(c(13,12,12,11,9,10,9,9,8,11,8,10,13,10,12,9,
          10,9,10,5,9,9,13,7,5,10,7,6,10,10,10,7), nrow = 2))
data = list(r = matrix(c(13,12,12,11,9,10,9,9,8,10,8,9, 12,9,11,8,9,8,9,
                    4,8,7,11,4,4,5,5,3,7,3,7,0), nrow = 2))
inits <- list(a = c(2, 2), b = c(2, 2))
## create a model object that can be used and modified
model <- nimbleModel(glmmCode, constants = consts, data = data, inits = inits)
```

Data vs. constants

- data: observations that contribute to the likelihood (and in some cases predictors/covariates)
- constants: values that never change and must include any indexing limits (e.g., `n` in `for(i in 1:n)`)
- inits: initial values for parameters (or in some cases predictors/covariates)

'Operating' the model

You can move values into and out of the model, simulate from the priors, and calculate prior densities ('logProbs'), as well as examining relationships in the model.

```
model$p
model$simulate('p')
model$p
model$p <- matrix(0.5, consts$G, consts$N)
## IMPORTANT - don't assume you know what child nodes do or
## don't need to be updated once you've changed values in the model:
model$calculate(model$getDependencies('p'))
model$calculate('r')

## plot(model£graph)
model$getDependencies('a[1]')
model$getDependencies('a[1]', determOnly = TRUE)
model$getDependencies('a[1]', dataOnly = TRUE)
model$getDependencies('a[1]', dataOnly = TRUE, downstream = TRUE)
```

Why simulation based methods: A motivating example

- To illustrate the power of simulation methods, consider the problem of estimating the diagnostic power of a medical test, defined as the probability of a positive result in the test for a diseased individual.

$$\eta = \Pr\left(D \mid T\right).$$

- η is not directly observable. However, we can easily obtain data on the prevalence θ, the false positive rate α and the false negative rate β,

$$\theta = \Pr(D), \qquad \alpha = \Pr\left(T \mid \bar{D}\right), \qquad \beta = \Pr\left(\bar{T} \mid D\right),$$

by sampling individuals from the general population, the population of healthy individuals, and the population of diseased individuals, respectively.

A motivating example

- Note that η is related to θ, α and β by the formula:

$$\eta(\theta, \alpha, \beta) = \frac{(1-\beta)\theta}{(1-\beta)\theta + \alpha(1-\theta)}.$$

(This is just Bayes theorem!)

- **Statistical model:** Assuming that individuals are sampled at random from the corresponding populations

$$x_1 = \left\{\begin{array}{l}\text{\# of diseased individuals in a sample} \\ \text{of size } n_1 \text{ of the general population}\end{array}\right\} \sim \mathrm{Bin}(n_1, \theta),$$

$$x_2 = \left\{\begin{array}{l}\text{\# of positive individuals in a sample} \\ \text{of size } n_2 \text{ of healthy individuals}\end{array}\right\} \sim \mathrm{Bin}(n_2, \alpha),$$

$$x_3 = \left\{\begin{array}{l}\text{\# of positive individuals in a sample} \\ \text{of size } n_3 \text{ of diseased individuals}\end{array}\right\} \sim \mathrm{Bin}(n_3, 1-\beta).$$

A motivating example

- A frequentist approach:
  - **Point estimation:** The MLEs for θ, α and β are

$$\hat{\theta} = \frac{x_1}{n_1}, \qquad \hat{\alpha} = \frac{x_2}{n_2}, \qquad \hat{\beta} = 1 - \frac{x_3}{n_3}.$$

    Hence, the MLE of η is simply $\hat{\eta} = \frac{(1-\hat{\beta})\hat{\theta}}{(1-\hat{\beta})\hat{\theta}+\hat{\alpha}(1-\hat{\theta})}$.
  - **Interval estimation:** Finding a pivot for η is hard. However, an asymptotic approximate interval can be constructed using the Delta method

$$\text{Var}(\hat{\eta}) \approx \left[\nabla \eta(\theta,\alpha,\beta)|_{(\hat{\theta},\hat{\alpha},\hat{\beta})} \right]^T \mathbf{\Sigma} \left[\nabla \eta(\theta,\alpha,\beta)|_{(\hat{\theta},\hat{\alpha},\hat{\beta})} \right]$$

    where $\mathbf{\Sigma} = \text{diag}\left\{ \text{Var}(\hat{\theta}), \text{Var}(\hat{\alpha}), \text{Var}(\hat{\beta}) \right\}$.

    For small samples, parametric bootstrap could be used (which is a simulation-based computational tool common in frequentist statistics).

A motivating example

- A frequentist approach (cont):
  - **Interval estimation (cont):** Parametric bootstrap: Once $\hat{\theta}$,$\hat{\alpha}$ and $\hat{\beta}$ have been computed, repeat the following steps for $b = 1, \ldots, B$ where B is "large":
    1. Sample imaginary samples $x_1^{(b)} \sim \text{Bin}(n_1, \hat{\theta})$, $x_2^{(b)} \sim \text{Bin}(n_2, \hat{\alpha})$ and $x_3^{(b)} \sim \text{Bin}(n_3, 1 - \hat{\beta})$.
    2. Compute $\tilde{\theta}^{(b)} = \frac{x_1^{(b)}}{n_1}$, $\tilde{\alpha}^{(b)} = \frac{x_2^{(b)}}{n_2}$ and $\tilde{\beta}^{(b)} = 1 - \frac{x_3^{(b)}}{n_3}$ and let $\tilde{\eta}^{(b)} = \frac{(1-\tilde{\beta}^{(b)})\tilde{\theta}^{(b)}}{(1-\tilde{\beta}^{(b)})\tilde{\theta}^{(b)}+\tilde{\alpha}^{(b)}(1-\tilde{\theta}^{(b)})}$.

    Then, an ϵ-coverage confidence interval for η is obtained by computing the $\epsilon/2$ and $1 - \epsilon/2$ quantiles of the sample $\tilde{\beta}^{(1)}, \ldots, \tilde{\beta}^{(B)}$.

A motivating example

- A Bayesian approach:
  - **Priors:** Natural non-informative priors for this problem are $\theta \sim \text{beta}\left(\frac{1}{2}, \frac{1}{2}\right)$, $\alpha \sim \text{beta}\left(\frac{1}{2}, \frac{1}{2}\right)$ and $\beta \sim \text{beta}\left(\frac{1}{2}, \frac{1}{2}\right)$.
  - **Posteriors:** Because of independence and conjugacy we have

$$p(\theta, \alpha, \beta \mid x_1, x_2, x_3) = \text{beta}\left(\theta \mid \frac{1}{2} + x_1, \frac{1}{2} + n_1 - x_1\right)$$
$$\text{beta}\left(\alpha \mid \frac{1}{2} + x_2, \frac{1}{2} + n_2 - x_2\right) \text{beta}\left(\beta \mid \frac{1}{2} + n_3 - x_3, \frac{1}{2} + x_3\right)$$

Since η is a function of (θ, α, β), its posterior can be obtained through transformations. However, in this case, this is extremely messy ... Try it yourself.

A motivating example

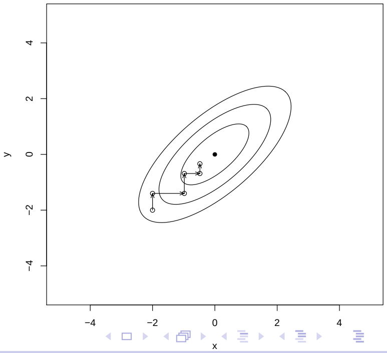
- A Bayesian approach:
  - **Using a simulation-based approach:** Instead of trying to find a closed-form expression for $p(\eta \mid x_1, x_2, x_3)$, sample!
    1. For $b = 1, \ldots, B$ generate $\theta^{(b)} \sim \text{beta}\left(\frac{1}{2} + x_1, \frac{1}{2} + n_1 - x_1\right)$, $\alpha^{(b)} \sim \text{beta}\left(\frac{1}{2} + x_2, \frac{1}{2} + n_2 - x_2\right)$ and $\beta^{(b)} \sim \text{beta}\left(\frac{1}{2} + n_3 - x_3, \frac{1}{2} + x_3\right)$.
    2. For $b = 1, \ldots, B$ let $\eta^{(b)} = \frac{(1-\beta^{(b)})\theta^{(b)}}{(1-\beta^{(b)})\theta^{(b)} + \alpha^{(b)}(1-\theta^{(b)})}$.
  - Density $\Rightarrow$ histogram or a kernel density estimator.
  - Posterior mean or median $\Rightarrow$ Empirical ean or median.
  - An ϵ-probability credible interval $\Rightarrow$ $\epsilon/2$ and $1 - \epsilon/2$ quantiles of $\eta^{(1)}, \ldots, \eta^{(B)}$.

Let's run the scripts in the file `simulationmotivation.R`.

Motivation for MCMC algorithms

- Direct sampling like in the previous example is not feasible for many models of interest, particularly those involving large numbers of parameters.
- In *optimization* problems, this “curse of dimensionality” is dealt with by using conditional gradient algorithms.

Conditional gradient algorithms repeatedly optimizes along one single dimension while keeping the value of the rest fixed.



Motivation for MCMC algorithms

- **Example (Gaussian distributions with unknown mean and variance):** Assume that data $y_1, \ldots, y_n$ are independent and identically distributed with $y_i \mid \theta, \sigma^2 \sim \mathsf{N}(\theta, \sigma^2)$ and that we use independent priors $\theta \sim \mathsf{N}(\mu, \tau^2)$ and $\sigma^2 \sim \mathsf{IGam}(a, c)$. After some algebra the posterior distribution reduces to

$$p(\theta, \sigma^2 \mid y_1, \ldots, y_n) \propto \left(\frac{1}{\sigma^2}\right)^{\frac{n}{2}+a+1}$$
$$\exp\left\{-\frac{n\theta^2 - 2n\theta\bar{y} + \sum_{i=1}^{n} y_i^2 + 2c}{2\sigma^2} - \frac{\theta^2 - 2\theta\mu + \mu^2}{2\tau^2}\right\}.$$

This posterior is intractable (by which I mean, computing means/variances and cumulative probabilities is difficult).

Motivation for MCMC algorithms

- **Example (Gaussian distributions with unknown mean and variance, cont):** However, we can try to find the posterior mode (which would provide us with a point estimate for (θ, σ^2) in the same spirit as the MLE) using iterative conditional modes.

  Starting with some initial guess $\left(\theta^{(0)}, \sigma^{2(0)}\right)$ (such as $\left(\theta^{(0)}, \sigma^{2(0)}\right) = (\bar{y}, \text{Var}(y))$), iterate through:

  1. Set $\theta^{(b)} = \frac{\frac{n\bar{y}}{\sigma^{2(b-1)}} + \frac{\mu}{\tau^2}}{\frac{n}{\sigma^{2(b-1)}} + \frac{1}{\tau^2}}$.
  2. Set $\sigma^{2(b)} = \frac{2b + \sum_{i=1}^{n}\left(y_i - \theta^{(b)}\right)^2}{n+2a+2}$.

  The algorithm is stopped when the value of the posterior "does not change" from one iteration to the next.

  See the file `simpleconjugategradient.R`.

Motivation for MCMC algorithms

- Iterative conditional modes only gives us a point estimator and does not allow for full exploration of the posterior distribution (e.g., to compute credible intervals).
- What would be an equivalent idea for *sampling* from a distribution $p(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$?
  - If we consider a single dimension at a time we end up working with the *full conditional* $p(\theta_i \mid \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p)$.
  - Because we are trying to sample, maybe we can get a new value of θ_i by simulating from $p(\theta_i \mid \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p)$ instead of by maximizing it.
- Does this work?
  - Generally speaking, the answer is yes, this is the Gibbs sampler!

Motivation for MCMC algorithms

- **Example (Gaussian distributions with unknown mean and variance, cont):** We do not know how to sample directly from the posterior distribution. However:

$$\theta \mid \sigma^2, y_1, \ldots, y_n \sim \mathrm{N}\left(\frac{\frac{n\bar{y}}{\sigma^2} + \frac{\mu}{\tau^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}}, \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}} \right),$$

and

$$\sigma^2 \mid \theta, y_1, \ldots, y_n \sim \mathrm{IGam}\left(a + \frac{n}{2}, c + \frac{1}{2}\sum_{i=1}^{n} (y_i - \theta)^2 \right).$$

(To find these full conditionals, just drop from the posterior all the – multiplicative – terms that do not involve the parameter of interest, and then try to recognize what is left as the kernel of a known distribution.)

Motivation for MCMC algorithms

- **Example (Gaussian distributions with unknown mean and variance, cont):** The iterative algorithm looks like this
  1. Initialize $\tilde{\theta}^{(0)}$ and $\tilde{\sigma}^{2(0)}$.
  2. For $b = 1, \ldots, B$ repeat
     1. Sample $\tilde{\theta}^{(b)} \sim \mathsf{N}\left(\frac{\frac{n\bar{y}}{\tilde{\sigma}^{2(b-1)}} + \frac{\mu}{\tau^2}}{\frac{n}{\tilde{\sigma}^{2(b-1)}} + \frac{1}{\tau^2}}, \frac{1}{\frac{n}{\tilde{\sigma}^{2(b-1)}} + \frac{1}{\tau^2}}\right)$.
     2. Sample $\tilde{\sigma}^{2(b)} \sim \mathsf{IGam}\left(a + \frac{n}{2}, c + \frac{1}{2}\sum_{i=1}^{n}\{y_i - \tilde{\theta}^{(b)}\}^2\right)$.

  An example of this implementation in R is available in the file `simpleGibbs.R`.
  - Run the algorithm first with $\theta^{(0)} = 98$ and $\sigma^{2(0)} = 1$ and then with $\theta^{(0)} = 0$ and $\sigma^{2(0)} = 1600$. What differences do you see in the trace plots?

Motivation for MCMC algorithms

- A couple of things to keep in mind:
  - Unlike iterative conditional modes, we care about all numbers generated by the sampler, not only about the last one (we are trying to explore the whole distribution, not just find where the mean/median/mode is).
  - Note that when the initial values are too far from where most of the mass of the posterior is located (as in the second set of initial values), the first few samples are clearly wrong and can seriously bias the results. Hence they need to be discarded.
- Because in the Gibbs sampler we are dealing with a distribution, we need to think about convergence in the probabilistic sense (almost surely/distribution/probability) rather than in terms of convergence in the sense we talk in analysis.

Motivation for MCMC algorithms

- Because the value of each pair $\left(\theta^{(b)}, \sigma^{2(b)}\right)$ depends on the value of $\left(\theta^{(b-1)}, \sigma^{2(b-1)}\right)$ (but NOT on $\left(\theta^{(b-2)}, \sigma^{2(b-2)}\right), \ldots, \left(\theta^{(1)}, \sigma^{2(1)}\right)$) we are actually dealing with a Markov chain!
- The Gibbs sampler is but one example of a class of algorithms that attempt to construct Markov chains whose limiting distributions correspond to the posterior distribution of interest, and then iterate the chain to generate (dependent) samples that can be used to approximate any integral of interest. Hence the name of Markov chain Monte Carlo methods!
- For a history of MCMC algorithms, see Robert et al. (2011).

A brief review of Markov chains

- A *Markov chain* is a sequence of random variables $X_1, X_2, X_3 \dots$ defined on a common probability space $(\Omega, \mathcal{B})$ such that the distribution of X_t given $X_{t-1}, \dots, X_1$ depends only on X_{t-1}, i.e.,

$$\Pr(X_t \in A \mid X_{t-1} = x_{t-1}, \dots, X_1 = x_1) = \Pr(X_t \in A \mid X_{t-1} = x_{t-1})$$

- It is convenient to think of t as representing time.

A brief review of Markov chains

- Markov chains are defined in terms of *transition kernels*.
- A *transition kernel* is a function K defined on $\Omega \times \mathcal{B}$ such that
  1. For all $x \in \Omega$, $K(x, \cdot)$ is a probability measure.
  2. For all $A \in \mathcal{B}$, $K(\cdot, A)$ is measurable.
- We will focus on *time-homogeneous chains* where the transition kernel is the same for all t.
  - When Ω is countable the transition kernel is a matrix such that $[\mathbf{P}]_{x,y} = \Pr(X_t = y \mid X_{t-1} = x)$ where $x, y \in \Omega$.
  - When Ω is uncountable K represents a conditional density, i.e., $\Pr(X_t \in A | X_{t-1} = x) = \int_A K(x, x')\mathrm{d}x'$.

A brief review of Markov chains

For simplicity we present the main concepts in the context of countable state spaces.

- Assuming that Ω contains exactly K points, the state of a chain at time t can be represented by a probability vector $\boldsymbol{\pi}_t = (\pi_{t,1}, \ldots, \pi_{t,K})^T$.
- The state of the chain at times t and $t+1$ are related by $\boldsymbol{\pi}_t^T \mathbf{P} = \boldsymbol{\pi}_{t+1}^T$.
- Similarly, the state of the chain at times t and $t+k$ are related by $\boldsymbol{\pi}_t^T \mathbf{P}^k = \boldsymbol{\pi}_{t+1}^T$.

Stationary distribution of a Markov chain

- A distribution $\tilde{\pi}$ is a *stationary distribution* if $\tilde{\pi}^T \mathbf{P} = \tilde{\pi}^T$.
- A Markov chain (on a countable probability space) has a unique stationary distribution if and only if it is *irreducible* and all of its states are *positive recurrent*.
  - A chain is irreducible if for every pair of states i, j exist finite constants $n_{i,j}$ and $n_{j,i}$ such that $\Pr(X_{n_{i,j}} = j \mid X_0 = i) > 0$ and $\Pr(X_{n_{j,i}} = i \mid X_0 = j) > 0$.
  - A state is positive recurrent if the time it takes to return to it once you leave it is finite, i.e., if

$$M_i = \sum_{t=1}^{\infty} t\, f_{ii}^t < \infty$$

where

$$f_{ii}^t = \Pr(T_i = t), \qquad T_i = \inf\left\{t \geq 1 : X_t = i \mid X_0 = i\right\}.$$

Limiting distribution of a Markov chain

- A distribution $\hat{\boldsymbol{\pi}}$ is called a *limiting distribution* (or *equilibrium distribution*) if for any $\boldsymbol{\pi}_0$ we have $\lim_{n\to\infty} \boldsymbol{\pi}_0^T \mathbf{P}^n = \hat{\boldsymbol{\pi}}$.
- A Markov chain has a limiting distribution if the chain is positive recurrent chain, irreducible and aperiodic.
  - A state i has period k if any return to state i must occur in multiples of k time steps, i.e., $k = \gcd\{n : \Pr(X_n = i \mid X_0 = i) > 0\}$
  - A state with period $k = 1$ is called aperiodic.
  - An aperiodic chain has all its states being aperiodic (additional requirement compared to the stationary distribution).
- If a chain has a limiting distribution then it is equal to the unique stationary distribution!

A brief review of Markov chains

- If we can construct a Markov chain that has a unique stationary/limiting distribution that is identical to the one we are interested in (in our case, the posterior distribution), then we can generate samples from the posterior by using the following approach:
  - Pick an initial state from the chain x_0 according to some distribution (often the prior).
  - For $t = 1, \ldots, T$ iteratively sample x_t from a multinomial distribution with parameter $\left(p_{x_{t-1},1}, \ldots, p_{x_{t-1},K}\right)$.
  - Use the samples so obtained to approximate any posterior summary of interest.

A brief review of Markov chains

- Some caveats!
  - The samples generated by the chain will be (approximately) identically distributed. However, they will not be independent, so convergence of empirical averages to integrals cannot be argued using standard arguments (e.g., WLLN). Instead, the *ergodic theorem* needs to be used to justify the convergence of averages.
  - Additionally, because samples are not independent, the variance of the estimates will depend not only on how many samples are generated, but also on the autocorrelation! This means we might need more samples than if we had a direct sampler.
  - Unless the initial state is sampled from the stationary distribution (which will rarely be the case) the first few samples will follow a distribution that can be very different from the stationary distribution (and therefore need to be discarded!)

A brief review of Markov chains

- Given a transition matrix, finding the stationary/limiting distribution is not too difficult. However, in Bayesian statistics we are faced with the opposite problem: we know the stationary/limiting distribution and need to come up with (at least one) transition kernel that generates it.
- Reversibility is a concept that can help in the construction of such chains.
- A Markov chain with transition matrix $\mathbf{P}$ and stationary distribution $\tilde{\boldsymbol{\pi}}$ is said to be *reversible* if $\tilde{\pi}_i p_{i,j} = \tilde{\pi}_j p_{j,i}$. This equation is called the detailed balance equation.
- Roughly speaking, if we know what the stationary distribution should look like, the detailed balance equations tell us that we can choose $p_{i,j}$ for $i > j$ in whichever way we want as long as we then make $p_{j,i} = \frac{\pi_i}{\pi_j} p_{i,j}$.

The Metropolis-Hastings algorithm

- We move on now to discuss specific MCMC algorithms used in Bayesian computation. We start with the Metropolis-Hastings (MH) algorithm, which encompasses many others as special cases.
- The idea behind the MH algorithm is similar to that behind the rejection algorithm:
  - Pick a proposal distribution $q(\vartheta \mid \theta)$ (think of it as *almost* the transition kernel of your Markov chain) that will be used to generate potentially new states.
  - The chain either stays on the old value or moves to this new proposed values according to a certain probability, that is chosen to ensure that the chain is reversible and has the right stationary distribution!

The Metropolis-Hastings algorithm

Given $\theta^{(t)}$

1. Generate $\vartheta^{(t+1)} \sim q\left(\vartheta \mid \theta^{(t)}\right)$.
2. Take

$$\theta^{(t+1)} = \begin{cases} \vartheta^{(t+1)} & \text{with probability } \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \\ \theta^{(t)} & \text{with probability } 1 - \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \end{cases},$$

where

$$\rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) = \min\left\{1, \frac{p\left(\vartheta^{(t+1)} \mid \mathbf{y}\right)}{p\left(\theta^{(t)} \mid \mathbf{y}\right)} \frac{q\left(\theta^{(t)} \mid \vartheta^{(t+1)}\right)}{q\left(\vartheta^{(t+1)} \mid \theta^{(t)}\right)}\right\}.$$

The Metropolis-Hastings algorithm

- $\rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right)$ is called the Metropolis-Hastings acceptance probability.
- Note that the algorithm depends on the ratio $\frac{p\left(\vartheta^{(t+1)}|\mathbf{y}\right)}{p\left(\theta^{(t)}|\mathbf{y}\right)}$, so it works even if $p\left(\theta \mid \mathbf{y}\right)$ is known only up to a normalizing constant.
- There are a number of variants of the algorithm according to the form of the proposal!
  - Random walk Metropolis-Hastings.
  - Independent proposal Metropolis-Hastings.
  - Hamiltonian Monte Carlo.

The Metropolis-Hastings algorithm

In the random-walk Metropolis-Hastings, given $\theta^{(t)}$:

1. Generate $\vartheta^{(t+1)} \sim g\left(\left|\vartheta - \theta^{(t)}\right|\right)$, where g is a density.
2. Take

$$\theta^{(t+1)} = \begin{cases} \vartheta^{(t+1)} & \text{with probability } \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \\ \theta^{(t)} & \text{with probability } 1 - \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \end{cases},$$

where

$$\rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) = \min\left\{1, \frac{p\left(\vartheta^{(t+1)} \mid \mathbf{y}\right)}{p\left(\theta^{(t)} \mid \mathbf{y}\right)}\right\}.$$

Common choices for g include zero-mean Gaussian (my favorite) or uniform distributions!

The Metropolis-Hastings algorithm

- **Example (Gumbel likelihood):** Consider a setting where $y_1, \ldots, y_n$ corresponds to a random sample from a Gumbel distribution with location parameter θ, i.e.,

  $$p(y_i \mid \theta) = \exp\left\{-(y_i - \theta) - \exp\{-(y_i - \theta)\}\right\} \qquad y_i \in \mathbb{R}$$

  and assume that we let $\theta \sim \mathrm{N}(\xi, \kappa^2)$ a priori.

  The posterior distribution associated with this model is intractable. However, creating a RWMH algorithm to sample from it is straightforward. Because θ can in principle take any real value, a Gaussian random walk seems appropriate,

  $$q(\vartheta \mid \theta) = \frac{1}{\sqrt{2\pi}\tau} \exp\left\{-\frac{1}{2\tau^2}(\vartheta - \theta)^2\right\}$$

The Metropolis-Hastings algorithm

- **Example (Gumbel likelihood, cont):** The corresponding acceptance probability becomes

$$\rho(\vartheta, \theta) =$$

$$\min\left\{1, \frac{\exp\left\{-\sum_{i=1}^{n}(y_i - \vartheta) - \sum_{i=1}^{n}\exp\{-(y_i - \vartheta)\}\right\}}{\exp\left\{-\sum_{i=1}^{n}(y_i - \theta) - \sum_{i=1}^{n}\exp\{-(y_i - \theta)\}\right\}} \frac{\exp\left\{-\frac{(\vartheta-\xi)^2}{2\kappa^2}\right\}}{\exp\left\{-\frac{(\theta-\xi)^2}{2\kappa^2}\right\}}\right\}$$

(Note that, as we discussed before, the ratio of the proposals cancels out because of the symmetry!)

We implement the algorithm using NIMBLE in the file `mh-gumbel.R` and evaluate its performance using $\tau^2 = 0.001$ (too small!), $\tau^2 = 0.07$ (about right, roughly 40% acceptance rate), and $\tau^2 = 5$ (too large!).

The Metropolis-Hastings algorithm

Here's what a direct implementation in R would look like (see the code file for the definition of `dgumbel`).

```r
smpR <- rep(0, r)
acceptRate <- 0
thetaCurr <- thetaInit
for(s in 1:r){
    thetaProp <- rnorm(1, thetaCurr, sqrt(tau2))
    rho <- sum(dgumbel(x, thetaProp, log = TRUE)) -
        sum(dgumbel(x, thetaCurr, log = TRUE))
    ## likelihood and prior evaluated on log scale to avoid underflow
    rho <- rho + dnorm(thetaProp, xi, sqrt(kappa2), log = TRUE) -
        dnorm(thetaCurr, xi, sqrt(kappa2), log = TRUE)
    u <- runif(1, 0, 1)
    if(log(u) < rho){  # compare on log scale and no need for min()
      thetaCurr <- thetaProp
      acceptRate <- acceptRate + 1
    }
    smpR[s] <- thetaCurr
}
```

The Metropolis-Hastings algorithm

NIMBLE implements the algorithm in a model-generic fashion using a `nimbleFunction`. The `setup` code adapts the algorithm to the model and parameter of interest. The `run` code implements the algorithm generically.

```r
sampler_RW <- nimbleFunction(
    contains = sampler_BASE,
    setup = function(model, mvSaved, target, control) {
        ## control list extraction
        logScale      <- control$log
        scale         <- control$scale
        ## node list generation
        calcNodes  <- model$getDependencies(target)
    },
    run = function() {
        currentValue <- model[[target]]
        propLogScale <- 0
        if(logScale) {
            propLogScale <- rnorm(1, mean = 0, sd = scale)
            propValue <- currentValue * exp(propLogScale)
        } else propValue <- rnorm(1, mean = currentValue,  sd = scale)
        model[[target]] <<- propValue
        logMHR <- calculateDiff(model, calcNodes) + propLogScale
        jump <- decide(logMHR)
        if(jump) nimCopy(from = model, to = mvSaved, row = 1, nodes = calcNodes, logProb = TRUE)
        else     nimCopy(from = mvSaved, to = model, row = 1, nodes = calcNodes, logProb = TRUE)
    })
```

The Metropolis-Hastings algorithm

Here's the BUGS-based model specification for this dataset in NIMBLE.

```
gumbelModelCode <- nimbleCode({
    theta ~ dnorm(xi, sd = kappa)
    for(i in 1:n)
          x[i] ~ dgumbel(theta)
})

gumbelModel <- nimbleModel(gumbelModelCode,
          constants = list(n = n, xi = xi, kappa = sqrt(kappa2)),
          data = list(x = x), inits = list(theta = thetaInit))


## defining model...


## Registering the following user-provided distributions: dgumbel .
## Warning: random generation function for dgumbel is not available as a nimbleFunction without setup code
## NIMBLE has registered dgumbel as a distribution based on its use in BUGS code. Note that if you make cl


## building model...
## setting data and initial values...
## running calculate on model (any error reports that follow may simply reflect missing values in
model variables) ...
##
## checking model sizes and dimensions...
##
## model building finished.
```

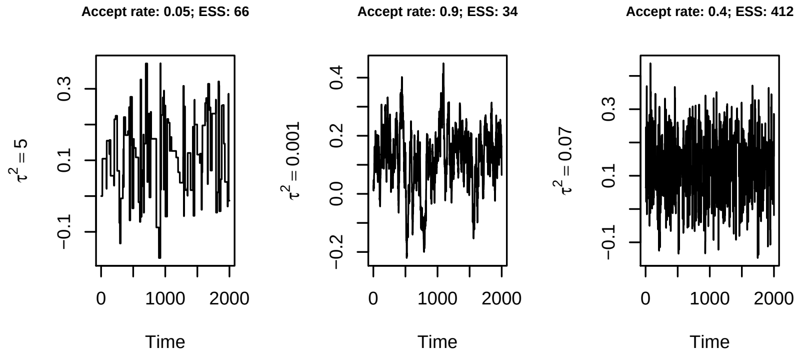
The Metropolis-Hastings algorithm

Here's how we set up and run the MCMC, with $\tau^2 = 5$.

```
### variance of proposal
tau2 <- 5
### empty MCMC configuration as default would use adaptive algorithm
conf <- configureMCMC(gumbelModel, nodes = NULL)
### add basic Metropolis sampler for the parameter
conf$addSampler('theta', 'RW', control = list(adaptive = FALSE,
                                             scale = sqrt(tau2)))
### create MCMC algorithm for the model, given the configuration
mcmc <- buildMCMC(conf)
### compiled version of the model
cGumbelModel <- compileNimble(gumbelModel)


## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compiler
details.
## compilation finished.

### compiled version of MCMC (model needs to be compiled first or at the same time)
cmcmc <- compileNimble(mcmc)


## compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compiler
details.
## compilation finished.

set.seed(1)
### run the MCMC
cmcmc$run(r)
```

The Metropolis-Hastings algorithm

Here are the results/MCMC diagnostics for various values of τ^2.



The Metropolis-Hastings algorithm

- **Example (Nonlinear binomial regression):** We'll consider next a binomial regression for data on the number of adult flour beetles killed after five hours of exposure to various levels of gaseous carbon disulphide (CS_2) (from Bliss (1935, Annals of Applied Biology).

$$P(\text{death} \mid w_i) \equiv h(w_i) = \left[\frac{\exp(x_i)}{1+\exp(x_i)}\right]^{m_1}$$

where $m_1 > 0$, w_i is the known covariate (dose), and $x_i = \frac{w_i - \mu}{\sigma}$, where $\mu \in R^1$ and $\sigma^2 > 0$. This problem is more complicated than the previous one in two ways: Now the unknown paramater vector $\boldsymbol{\theta} = (\mu, \sigma, m_1)$ is three-dimensional, and the last two parameters are restricted to be positive!

The Metropolis-Hastings algorithm

- **Example (binomial regression, cont):** For the positive parameters, we need to use non-negative priors:

$$\mu \sim N(c_0, d_0) \qquad \sigma^2 \sim IG(e_0, f_0) \qquad m_1 \sim Ga(a_0, b_0)$$

A Gaussian random walk directly on θ is not a great idea (you would be automatically rejecting every time you generate negative values for either σ or m_1).

However, a (trivariate!) Gaussian random walk for $(\mu, \log\sigma, \log m_1)$ is feasible and does not lead to automatic rejections.

(Another alternative is to use a reflecting Gaussian random walk (this is an option in NIMBLE), but I will not pursue that idea further.)

The Metropolis-Hastings algorithm

- **Example (binomial regression, cont):** Since we work with a transformation of the parameters, there are two ways to derive the algorithm (they lead to different formal expressions, but they are equivalent!):
  1. Do a transformation so that the problem is reformulated as sampling from $p(z_1, z_2, z_3 \mid \mathbf{y})$ where $z_1 = \mu$, $z_2 = \log \sigma$ and $z_3 = \log m_1$ instead of $p(\mu, \sigma, m_1 \mid \mathbf{y})$. Once samples $z_1^{(1)}, \ldots, z_1^{(B)}$ and $z_2^{(1)}, \ldots, z_2^{(B)}$ and $z_3^{(1)}, \ldots, z_3^{(B)}$ have been generated, samples $\sigma^{(1)}, \ldots, \sigma^{(B)}$ and $m_1^{(1)}, \ldots, m_1^{(B)}$ can be constructed by letting $\sigma^{(b)} = \exp\{z_2^{(b)}\}$ and $m_1^{(b)} = \exp\{z_3^{(b)}\}$
  2. Keep the original target $p(\mu, \sigma, m_1 \mid \mathbf{y})$ but think of your proposal as being a trivariate combination of a Gaussian and log-Gaussian distribution rather than a Gaussian (and recognize that your proposal is almost symmetric, but not quite!).

Note that, in both cases, a Jacobian is going to be involved.

The Metropolis-Hastings algorithm

- **Example (binomial regression, cont):** Taking the second route, the proposal distribution is

$$p(\vartheta_1, \vartheta_2, \vartheta_3 \mid \mu, \sigma, m_1) = \frac{1}{(2\pi)^{3/2}} \frac{1}{\vartheta_2 \vartheta_3} |\mathbf{\Omega}|^{-1/2}$$
$$\exp\left\{ -\frac{1}{2} \begin{pmatrix} \vartheta_1 - \mu \\ \log \vartheta_2 - \log \sigma \\ \log \vartheta_3 - \log m_1 \end{pmatrix}^T \mathbf{\Omega}^{-1} \begin{pmatrix} \vartheta_1 - \mu \\ \log \vartheta_2 - \log \sigma \\ \log \vartheta_3 - \log m_1 \end{pmatrix} \right\}$$

Note that $\mathbf{\Omega}$ does not have to be diagonal, so the moves could (and, turns out, should!) be correlated. The acceptance probability is

$$\rho(\vartheta_1, \vartheta_2, \vartheta_3, \mu, \sigma, m_1) = \min\left\{ 1, \frac{p(\mathbf{y}|\vartheta_1, \vartheta_2, \vartheta_3) p(\vartheta_1, \vartheta_2, \vartheta_3)}{p(\mathbf{y}|\mu, \sigma, m_1) p(\mu, \sigma, m_1)} \frac{\vartheta_2 \vartheta_3}{\sigma m_1} \right\}$$

The Metropolis-Hastings algorithm

- **Example (binomial regression, cont):** In the file `mh-bliss.R` we implement the first of these two approaches (because that allows us to use NIMBLE's provided MCMC algorithms).
  1. To tune the proposal variance, we run the algorithm once with a diagonal proposal variance/covariance matrix for the three parameters.
  2. We then rerun with the proposal covariance matrix for the trivariate Gaussian proposal equal to $= \frac{2.38^2}{d} \mathrm{Var}\left\{ (\mu, \log \sigma, \log m_1)^T \mid \mathbf{y} \right\}$. (In this case $d = 3$.)

  We'll revisit this when we talk about adaptive Metropolis-Hastings in which we learn the right proposal covariance.

The Metropolis-Hastings algorithm

- The magic numbers $\frac{2.38^2}{d}\text{Var}(\boldsymbol{\theta}|\mathbf{y})$ for proposals and 44% acceptance rates for unidimensional problems and 23% for multivariate problems comes from Gelman et al. (1996) Roberts et al. (1997) and Roberts et al. (2001), which derived general theoretical results and performed experiments on a Gaussian.
- Since, with enough data, most posteriors are approximately Gaussian, these are reasonable rules of thumb!
- Note that, because we are making two independent runs, the fact that we change the variance of the chain is not (conceptually) an issue as it does not invalidate the Markov property.
- If a good approximation for $\text{Var}(\boldsymbol{\theta}|\mathbf{y})$ is available (rarely the case) then the preliminary run can be skipped.

The Metropolis-Hastings algorithm

In the independent Metropolis-Hastings, given $\theta^{(t)}$:

1. Generate $\vartheta^{(t+1)} \sim g\left(\vartheta\right)$.
2. Take

$$\theta^{(t+1)} = \begin{cases} \vartheta^{(t+1)} & \text{with probability } \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \\ \theta^{(t)} & \text{with probability } 1 - \rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) \end{cases},$$

where

$$\rho\left(\vartheta^{(t+1)}, \theta^{(t)}\right) = \min\left\{1, \frac{p\left(\vartheta^{(t+1)} \mid \mathbf{y}\right)}{p\left(\theta^{(t)} \mid \mathbf{y}\right)} \frac{g\left(\theta^{(t)}\right)}{g\left(\vartheta^{(t+1)}\right)}\right\}.$$

You have to be careful to choose an instrumental distribution g that has heavier tails than the target.

The Metropolis-Hastings algorithm

- **Example (autoregressive models):** Consider a zero-mean autoregressive process of the form

$$y_t = \phi y_{t-1} + \epsilon_t \qquad\qquad \epsilon_t \sim \mathrm{N}(0, \sigma^2)$$

where σ^2 is known and ϕ is unknown. It is customary to assume that the process is stationary, which requires $y_1 \sim \mathrm{N}\left(0, \frac{\sigma^2}{1-\phi^2}\right)$ and a prior for ϕ with support only on $(-1, 1)$ (e.g., a uniform).

The Metropolis-Hastings algorithm

- **Example (autoregressive models, cont):** The posterior distribution for this problem is

$$p(\phi \mid \mathbf{y}) \propto \left(1-\phi^2\right)^{\frac{1}{2}} \exp\left\{-\frac{(1-\phi^2)}{\sigma^2} y_1^2\right\}$$
$$\exp\left\{-\frac{1}{2\sigma^2}\left[\phi^2 \sum_{t=2}^{T} y_{t-1}^2 - 2\phi \sum_{t=2}^{T} y_t y_{t-1}\right]\right\} \mathbb{I}_{(-1<\phi<1)}$$

  Note that this posterior IS NOT tractable. However, if you drop the first two terms (which come from $p(y_1 \mid \phi)$) then the remainder looks like the kernel of a Gaussian distribution.

The Metropolis-Hastings algorithm

- **Example (autoregressive models, cont):** This suggest using independent proposal for you MCMC of the form

$$\vartheta \sim \mathsf{N}\left(\vartheta \mid \frac{\sum_{t=2}^{T} y_t y_{t-1}}{\sum_{t=2}^{T} y_t^2}, \frac{\sigma^2}{\sum_{t=2}^{T} y_t^2}\right) \mathbb{I}_{(-1<\vartheta<1)}$$

Because this distribution contains most of the information in the data, we expect it to be very close to the true posterior. The acceptance rate in this case is simply

$$\rho(\vartheta, \phi) = \min\left\{1, \sqrt{\frac{1-\vartheta^2}{1-\phi^2}} \exp\left\{\frac{y_1^2(\vartheta^2 - \phi^2)}{2\sigma^2}\right\}\right\}$$

Which depends only on the term we dropped out when generating our proposal!

The Metropolis-Hastings algorithm

- The Metropolis-Hastings algorithm is very general and we have a lot of freedom in choosing proposals. However, there are some constraints:
  - For the algorithm to work, we need to ensure that the chain is irreducible. One way to ensure this is ensure that the support of the proposal $q(\vartheta|\theta)$ contains the support of f for every θ.
  - Positive recurrence is typically not an issue once you have ensured irreducibility.
  - Aperiodicity is also typically not a problem (for most situation you would actually need to do some work to get a periodic chain).

The Metropolis-Hastings algorithm

- Why does the Metropolis-Hastings work? Because of the reversibility of the chain. This ensures that the stationary distribution is the posterior.
- Note that the transition kernel associated with the Metropolis-Hastings algorithm is

$$K(\theta_{t-1}, \theta_t) = \rho(\theta_t, \theta_{t-1}) q(\theta_t, \theta_{t-1}) + \{1 - r(\theta_{t-1})\} \delta_{\theta_{t-1}(\theta_t)}$$

  where δ_x denotes the Dirac mass at x and

$$r(\theta) = \int \rho(\vartheta, \theta) q(\vartheta \mid \theta) \mathrm{d}\vartheta$$

- This kernel satisfies the detailed balance equations.

Diagnostics for MCMC algorithms

- Recall that, although Metropolis-Hastings algorithm work if we run them long enough, the fact that samples are dependent presents challenges:
  - Convergence.
  - Mixing.
- Mixing: As with every Monte Carlo algorithm an important question is how many samples should be generated in order to have estimates of the parameters with a reasonable level of accuracy.
  - However, because samples are (most usually positively) correlated, standard theory (standard central limit theorems, Chebyshev's inequality, etc.) does not apply here!
  - Mixing is related to how fast the chain explores the posterior distribution.

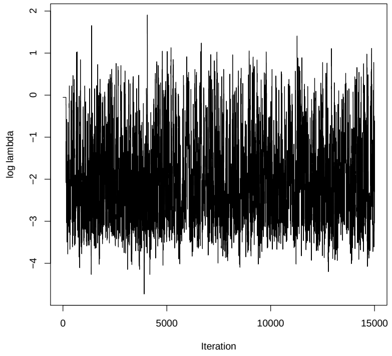
Diagnostics for MCMC algorithms

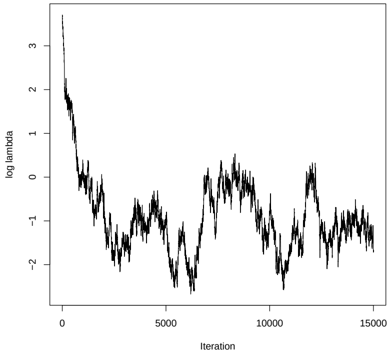
- Convergence: An additional question when using MCMC algorithms that is not a concern in standard Monte Carlo is how many of the initial observations need to be discarded before we can perform any inference.
  - Remember that, unless your initial state is sampled from the stationary distribution, the samples of your chain are only distributed as the stationary distribution when the number of iterations goes to infinity.
  - In practice, if your posterior is unimodal, you just want your initial state to be close to the high-probability areas.
  - When posterior is multimodal, convergence can be a huge issue.

Diagnostics for MCMC algorithms

- These two questions are interrelated, but are not identical. For example, you can accelerate convergence to the posterior by carefully selecting the initial state of your chain even if the chain mixes slowly. On the other hand, even if your initial state is really distributed from the stationary distribution (e.g., by using perfect sampling), you might need to run a very long chain to have an acceptable level of uncertainty in your estimates if the autocorrelation in the parameters of interest is very high.
- Time series plots of the realized chains (the so-called *trace* plots) provide visual intuition about these two concepts, and can be used as an informal check (which should be supplemented with some of the formal techniques discussed later).

Diagnostics for MCMC algorithms





Checking for convergence

- Generally speaking, determining *a priori* how many iterations are needed for burn-in in not possible. Indeed, except for some special cases, general theory about the behavior of the algorithm (e.g., the value of the second largest eigenvalue of the kernel function) is not easy to derive.
- For this reason, most practical algorithms to monitor convergence look at the time-series behavior of a small number of selected parameters (or functions of parameters).
  - Selecting which parameters need to be monitored is tricky and model dependent. In high-dimensional models I usually monitor either the likelihood or the posterior, a small (random) subset of the main first-stage parameters, and a couple of important hyperparameters.
  - All criteria here are implemented in the `R` package `CODA`.
- We typically cannot ensure convergence, just argue that there is no evidence of lack of convergence.

Checking for convergence (Multi-chain methods)

- As the name suggests multi-chain methods use $M > 1$ realizations of the algorithm (each of length B, typically started from "over dispersed" values) and compare the output of the chains. Once the statistical properties of the realizations appear to be similar, the chains are deemed to have converged.
- One example of this type of approach was introduced by Gelman & Rubin (1992) who propose to monitor a statistic that compares the average within-chain variability against the between-chain variability:

$$R = \left(\frac{B-1}{B} + \frac{M+1}{M} \frac{Z_B}{W_B} \right) \frac{\nu_B}{\nu_B - 2},$$

  where Z_B is an estimate of between-chain variability, W_B is an estimate of within-chain variability and ν_B is the approximate number of degrees of freedom.

Checking for convergence (Multi-chain methods)

If $\xi = h(\theta)$ is the quantity of interest being monitored, define

$$\bar{\xi}_m = \frac{1}{B}\sum_{b=1}^{B} \xi_m^{(b)} \qquad \bar{\xi} = \frac{1}{M}\sum_{m=1}^{M} \bar{\xi}_m \qquad s_m^2 = \frac{1}{B}\sum_{b=1}^{B}\left(\xi_m^{(b)} - \bar{\xi}\right)^2$$

Then

$$Z_B = \frac{1}{M}\sum_{m=1}^{M}\left(\bar{\xi}_m - \bar{\xi}\right)^2 \qquad W_B = \frac{1}{M}\sum_{m=1}^{M} s_m^2$$

and

$$\nu_B = \frac{2\left(\frac{B-1}{B}W_B + \frac{M+1}{M}Z_B\right)^2}{W_B}$$

Checking for convergence (Multi-chain methods)

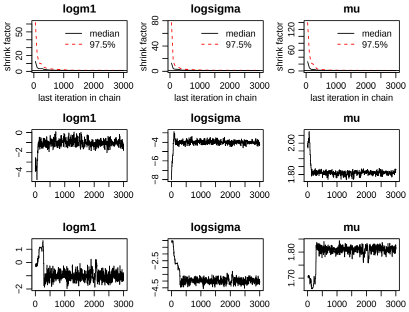
- We have $BZ_B/W_B \sim F_{M-1,2W_B^2/\varpi_B}$ approximately, where

$$\varpi_B = \frac{1}{M^2}\left[\sum_{m=1}^{M} s_m^4 - \frac{1}{M}\left(\sum_{m=1}^{M} s_m^2\right)^2\right]$$

- Once the chains have converged we expect $R \rightarrow 1$, so the approximate distribution can be used to test convergence at different values of B.
- Samples from different chains are then combined together for computing any expectation of interest.

Checking for convergence (Multi-chain methods)

```
library(coda, warn.conflicts = FALSE)

chains <- as.mcmc.list(
    list(as.mcmc(smp1), as.mcmc(smp2)))
gr = gelman.diag(chains, autoburnin = FALSE)
print(gr)


## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## logm1          1.27       2.02
## logsigma       1.13       1.46
## mu             1.28       1.89
##
## Multivariate psrf
##
## 1.17
```



Checking for convergence (Multi-chain methods)

- I tend to prefer multi-chain methods. One advantage is that they can be naively parallelized.
- However, finding over-dispersed initial points can be hard. For example, if the posterior is multimodal and you do not start the chain at least once on each mode you might never find out.
- For slow-mixing chains, inferences based on a single long chain might be more accurate than the inferences based on many small chains put together. (However, this is an issue mostly if computational resources are limited.)

Checking for convergence (Single-chain methods)

- An alternative to multi-chain methods is to use a single long chain and see if the properties of the chain have “stabilized”.
- More specifically, we can divide the original chain of length B into M sections of approximately equal length and compare their spectral densities.
- In particular Geweke (1992) proposes to compare the first and second half of the chain (i.e., use $M = 2$). The algorithm is implemented in the R function `geweke.diag` and `geweke.plot`.
- Interestingly, the most recent edition of Gelman et al. Bayesian Data Analysis suggests to use the Gelman-Rubin approach with each chain split into two to assess both mixing and convergence.

Mixing

- Again, it must be emphasized that mixing is a subtly different problem from convergence.
- Note that a chain that mixes slowly, if started on a low-probability region of the space, might take a long time to converge. However, if we start it in a high probability region convergence is much less of an issue.
- A simple way to decide how long the algorithm must be run *after* the burn-in period is to evaluate the variance of the estimators.

Mixing

- Let h be an integrable function. Recall that we aim at approximating $\mathrm{E}_{\boldsymbol{\theta}|\mathbf{y}}\left\{h(\boldsymbol{\theta})\right\} \approx \frac{1}{B}\sum_{b=1}^{B} h\left(\boldsymbol{\theta}^{(b)}\right)$
- If the samples $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(B)}$ were independent, the variance of the estimator would be equal to $\frac{\mathrm{Var}\{h(\boldsymbol{\theta})\}}{B}$. In practice a rough estimate of $\mathrm{Var}\left\{h(\boldsymbol{\theta})\right\}$ can also be obtained by Monte Carlo.
- From Markov chain theory, it is easy to show that for an MCMC that has converged the variance of the same estimator is $\tau_h \frac{\mathrm{Var} h(\boldsymbol{\theta})}{B}$ where τ_h is the integrated autocorrelation: $\tau_h = 1 + 2\sum_{t=1}^{\infty} \rho_t$, with ρ_t the autocorrelation at lag t.
- We call $\frac{B}{\tau_h}$ the equivalent sample size.

Mixing

- The equivalent sample size calculation is implemented in `CODA` through the function `effectiveSize`.
- Note that the equivalent sample size will be different for different parameters. That means that a given number of iterations might be enough for one parameter, but not for another.
- Some people like to “thin” their chain (retain every X number of iterations and drop the rest) as a way to decrease τ_h. However, this also reduces B!
  - MacEachern & Berliner (1994) proved the tradeoff is not favorable.
  - Unless storage (in memory or on disk) is an issue, thinning is not a good idea.

Beyond the Metropolis-Hasting algorithm

- Our description of the Metropolis-Hastings algorithm involved a single joint proposal for all entries of $\boldsymbol{\theta}$.
- However, even using adaptive algorithms as the ones described later, making good proposals in high dimensions is difficult. It would be helpful if we could work with one dimension (or a small number of dimensions) at a time and make low-dimensional updates.
- Such approach actually works, and can be justified by considering mixtures of kernels!

Mixtures of kernels

- You do not have to update parameters one at a time, you might decide to update subgroups of parameters.
- In addition to random sweeps, you can run through the kernels sequentially. Justification is based on a composition of kernels rather than a mixture.
- In sequential sweeps not all kernels need to be used with the same frequency (just like kernels do not have to be chosen with the same probability when doing random sweeps).
- It is important that all variables are updated at least once (but they could be updated multiple times).

The Gibbs sampler as a special case of the Metropolis

- We can take our argument one step further. Suppose that you are working with the component-wise (or block-wise!) MH algorithm and that you are able to directly sample from the full conditional, so that you decide to use independent proposal (rather than a random-walk proposals) with

$$q(\vartheta_i \mid \theta_i) = p(\vartheta_i \mid \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p).$$

- Note that in that case the acceptance probability is 1 for all samples (the Hastings ratio is the reciprocal of the ratio of posteriors), and your MH algorithm reduces to (sequentially) sampling from all full conditional distributions!
  - This is what we called the Gibbs sampler.

The Gibbs sampler as a special case of the Metropolis

- **Example (litters GLMM):** NIMBLE's default samplers use a Gibbs-based strategy of cycling through individual parameters with either Metropolis or conjugate samplers assigned individually:

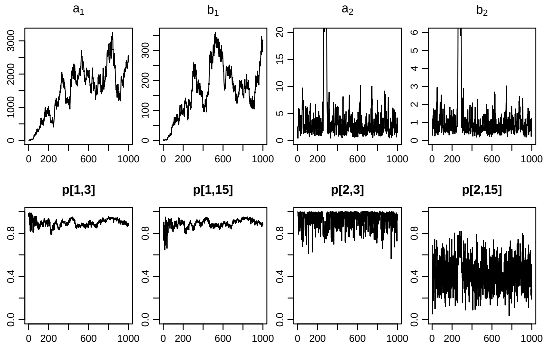
```
thin <- 10
# thinning _only_ to reduce time of plotting
conf <- configureMCMC(glmmModel, thin = thin)
conf$getSamplers(c('a','b','p[1,1]','p[2,1]'))
```

```
## [[1]]
## RW sampler: a[1]
## [[2]]
## RW sampler: a[2]
## [[3]]
## RW sampler: b[1]
## [[4]]
## RW sampler: b[2]
## [[5]]
## conjugate_dbeta_dbin sampler: p[1, 1]
## [[6]]
## conjugate_dbeta_dbin sampler: p[2, 1]
```

While Gibbs sampling is often effective, when there is strong posterior dependence, it can have bad convergence and bad mixing.

The Gibbs sampler as a special case of the Metropolis

- **Example (litters GLMM):** Here are traceplots (see `gibbs-litters.R` for the code).



Despite knowing the exact conditional distributions for p, the poor mixing of the hyperparameters in the priors for p produce bad mixing for p values for the first group.

General design rules for MH algorithms

- MH algorithms are highly modular but there is a trade-off between simplicity of implementation, speed, and mixing.
- Block highly-correlated parameters (but when there is not strong dependence individual samplers can move further than a blocked Gaussian).
- Reparameterizing the model to center it or to make parameters less correlated is usually helpful! This is easy to do in the context of linear model, but not always so in other case (computing information matrices might provide hints).
- As a rule of thumb, you want to “integrate out” as many parameters as you can before attempting to use any Metropolis/Gibbs steps. However, introducing latent variables can greatly simplify algorithms and in some cases improve mixing.

Blocking

- **Example (litters GLMM):** We saw that univariate sampling worked poorly on the litters GLMM model. This is common in GLMMs – the dependence between hyperparameters and random effects can impede mixing.

  Some strategies for random effects models:

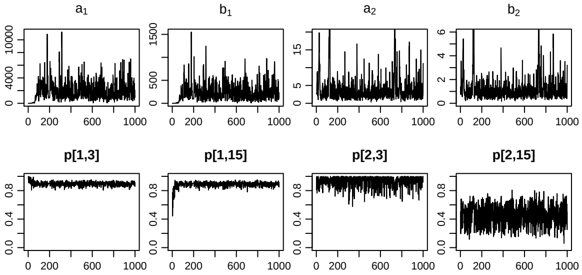
  - analytically integrate over random effects to reduce model dimensionality (not always feasible though feasible here)
  - approximately integrate over random effects (e.g., INLA)
  - expand the blocking to include the random effects

  In the litters model, we could integrate the p's out of the model and do block sampling on the hyperparameters.

  Alternatively, NIMBLE provides a *cross-level* sampler that blocks hyperparameters with random effects when the random effects have a conjugate structure. This blocking is equivalent to marginalizing the p's. This works much better than Metropolis-based blocking because the dependence is nonlinear.

Blocking

- **Example (litters GLMM):**
  See `blocked-litters.R` for NIMBLE code for the blocked sampler.



Centering

- To illustrate the role of centering, consider a simple linear mixed model (random-intercept model).

  $$y_{i,j} \mid \mu, \theta_i \sim \mathrm{N}(\eta + \theta_i, \sigma^2), \quad \eta \sim \mathrm{N}(0, \kappa^2 = \infty), \quad \theta_i \sim \mathrm{N}(0, \tau^2),$$

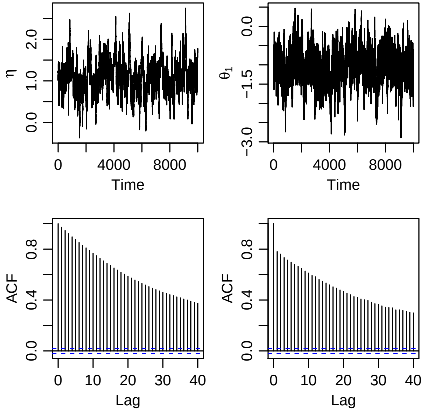
  for $i = 1, \ldots, I$ and $j = 1, \ldots, J$.
- The full conditional distributions are

$$\theta_i \mid \eta \sim \mathrm{N}\left(\frac{\sum_{j=1}^{J} (y_{i,j} - \eta)/\sigma^2}{\left(\frac{J}{\sigma^2} + \frac{1}{\tau^2}\right)}, \left(\frac{J}{\sigma^2} + \frac{1}{\tau^2}\right)^{-1} \right),$$

$$\eta \mid \theta_1, \ldots, \theta_I \sim \mathrm{N}\left(\frac{\sum_{i=1}^{I} \sum_{j=1}^{J} (y_{i,j} - \theta_i)/\sigma^2}{\left(\frac{IJ}{\sigma^2} + \frac{1}{\kappa^2}\right)}, \left(\frac{IJ}{\sigma^2} + \frac{1}{\kappa^2}\right)^{-1} \right).$$

Centering

- We simulated data with $\eta = 0.8$, $\sigma^2 = 1$ and $\tau^2 = 2$, and ran the algorithm.
- θ_1 and η show high (negative) autocorrelation and slow mixing.



Centering

- Instead, consider reparameterizing the model so that $\theta_i^* = \mu + \theta_i$ so that

$$y_{i,j} \mid \mu, \theta_i^* \sim \mathrm{N}(\theta_i^*, \sigma^2), \quad \theta_i^* \sim \mathrm{N}(\eta, \tau^2), \quad \eta \sim \mathrm{N}(0, \kappa^2).$$

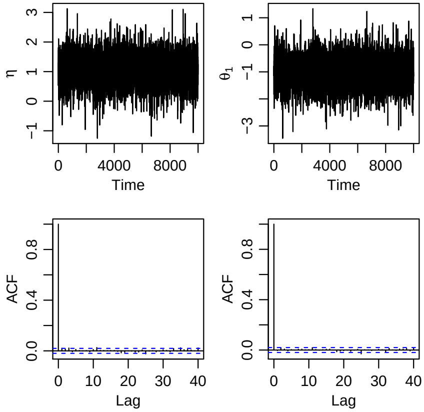
- The full conditional distributions are

$$\theta_i^* \mid \eta \sim \mathrm{N}\left(\frac{\frac{\sum_{j=1}^{J} y_{i,j}}{\sigma^2} + \frac{\eta}{\tau^2}}{\left(\frac{J}{\sigma^2} + \frac{1}{\tau^2}\right)}, \left(\frac{J}{\sigma^2} + \frac{1}{\tau^2}\right)^{-1} \right),$$

$$\eta \mid \theta_1^*, \ldots, \theta_I^* \sim \mathrm{N}\left(\frac{\frac{\sum_{i=1}^{I} \theta_i^*}{\tau^2}}{\left(\frac{I}{\tau^2} + \frac{1}{\kappa^2}\right)}, \left(\frac{I}{\tau^2} + \frac{1}{\kappa^2}\right)^{-1} \right).$$

Centering

- We ran this MCMC for the same dataset as before and backtransformed the samples to get $\theta_i = \theta_i^* - \eta$.
- Mixing for θ_1 and η is quite good now.
- See NIMBLE code in `mh-centering.R`.



Centering is not universally better, but it usually is! See (Yu & Meng, 2011) for how to combine centered and uncentered.

Auxiliary variables

- In spite of all the caveats mentioned so far, Gibbs steps are very often preferable because they are simpler to implement.
- Particularly in high-dimensional problems, making joint proposals is difficult, while univariate random-walk MH steps end up being very inefficient.
- In many cases computation can be simplified by adding auxiliary variables (we are not interested in them, but they are added for convenience!)
- This is sometimes called demarginalization: We look for a model such that, when we marginalize over the auxiliary variables, we recover the original one.
- This goes against our design criteria, but can dramatically simplify implementation with a moderate cost in terms of mixing. In some cases, it can even improve mixing!

Auxiliary variables

- **Example (Gaussian mixture models):** Assume that data is generated from a two-component mixture

$$y_i \mid \omega, \theta_1, \sigma_1^2, \theta_2, \sigma_2^2 \sim \omega \mathrm{N}(y_i \mid \theta_1, \sigma_1^2) + (1-\omega)\mathrm{N}(y_i \mid \theta_2, \sigma_2^2)$$

for $i = 1, \ldots, n$ with priors

$$\omega \sim \mathrm{beta}(1,1), \qquad \theta_i \sim \mathrm{N}(\mu, \tau^2), \qquad \sigma_i^2 \sim \mathrm{IGam}(a, c).$$

Sampling directly from this model is complicated. You could try a five-dimensional Gaussian random walk on $(\operatorname{logit} \omega, \theta_1, \log \sigma_1^2, \theta_2, \log \sigma_2^2)$, but it would be hard to tune. More importantly, such scheme would not scale very well to mixtures with more components.

Auxiliary variables

- **Example (Gaussian mixture models, cont):** Augment the model with indicator variables $\xi_1, \dots, \xi_n$ such that $\xi_i \in \{1, 2\}$ and
$$y_i \mid \xi_i, \theta_1, \sigma_1^2, \theta_2, \sigma_2^2 \sim \mathsf{N}(y_i \mid \theta_{\xi_i}, \sigma_{\xi_i}^2),$$
where the ξ_is are iid with
$$\Pr(\xi_i = 1 \mid \omega) = \omega = 1 - \Pr(\xi_i = 2 \mid \omega).$$
It is clear that if we integrate ξ_i out of the model we recover our original formulation.

Auxiliary variables

- **Example (Gaussian mixture models, cont):** In the augmented model, the posterior distribution is proportional to

$$\left\{ \prod_{i=1}^{n} \mathsf{N}\left(y_i \mid \theta_{\xi_i}, \sigma^2_{\xi_i}\right) \right\} \left\{ \prod_{i=1}^{n} \omega^{\sum_{i=1}^{n} \mathbb{I}_{\{\xi_i = 1\}}} (1-\omega)^{\sum_{i=1}^{n} \mathbb{I}_{\{\xi_i = 2\}}} \right\}$$
$$\mathsf{beta}(\omega \mid 1, 1) \left\{ \prod_{i=1}^{2} \mathsf{N}(\theta_i \mid \mu, \tau^2) \right\} \left\{ \prod_{i=1}^{2} \mathsf{IGam}(\sigma_i^2 \mid a, c) \right\}$$

Auxiliary variables

- **Example (Gaussian mixture models, cont):** The full conditionals reduce to:
  1. $\Pr(\xi_i = 1 \mid \cdots) = \frac{\omega \frac{1}{\sigma_1} \exp\left\{ -\frac{(y_i - \theta_1)^2}{2\sigma_1^2} \right\}}{\omega \frac{1}{\sigma_1} \exp\left\{ -\frac{(y_i - \theta_1)^2}{2\sigma_1^2} \right\} + (1-\omega) \frac{1}{\sigma_2} \exp\left\{ -\frac{(y_i - \theta_2)^2}{2\sigma_2^2} \right\}}$, while $\Pr(\xi_i = 2 \mid \cdots) = 1 - \Pr(\xi_i = 1 \mid \cdots)$.
  2. $\omega \mid \cdots \sim \text{beta}(1 + n_1, 1 + n_2)$ where $m_k = \sum_{i=1}^n \mathbb{I}_{\{\xi_i = k\}}$.
  3. $\theta_k \mid \cdots \sim \text{N}\left(\frac{\frac{s_k}{\sigma_k^2} + \frac{\mu}{\tau^2}}{\frac{m_k}{\sigma_k^2} + \frac{1}{\tau^2}}, \frac{1}{\frac{m_k}{\sigma_k^2} + \frac{1}{\tau^2}} \right)$ where $m_k = \sum_{i=1}^n \mathbb{I}_{\{\xi_i = k\}}$ as before and $s_k = \sum_{\{i:\xi_i = k\}} y_i$.
  4. $\sigma_k^2 \mid \cdots \sim \text{IGam}\left(a + \frac{m_k}{2}, c + \frac{1}{2} \sum_{\{i:\xi_i = k\}} \{y_i - \theta_k\}^2 \right)$.

  Note that the ξ_is have a nice interpretation, particularly if you are using the mixture model for clustering!

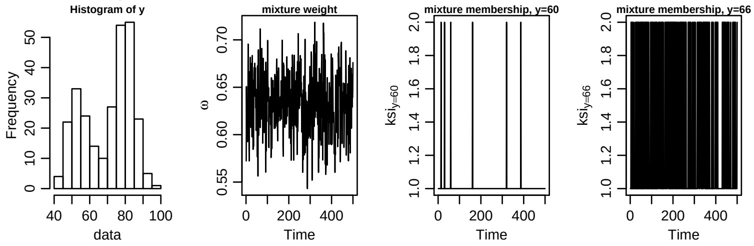
Auxiliary variables

- **Example (Gaussian mixture models, cont):** Here is the NIMBLE code for such an augmented mixture model.

```
## use of non-constant indexes requires NIMBLE version 0.6-6
if(packageVersion(nimble) < '0.6.6')
    stop("NIMBLE version 0.6-6 or greater required for non-constant indexe

nimbleOptions(allowDynamicIndexing = TRUE) ## it's a beta feature in 0.6-6

mixCode <- nimbleCode({
    for(i in 1:n) {
        y[i] ~ dnorm(theta[ksi[i]+1], var = sigma2[ksi[i]+1])
        ksi[i] ~ dbern(omega)
    }
    omega ~ dbeta(1, 1)
    for(j in 1:2) {
        theta[j] ~ dnorm(mu, tau2)
        sigma2[j] ~ dinvgamma(a, c)
    }
})
```

Auxiliary variables

- **Example (Gaussian mixture models, cont):** The code in `mixture-faithful.R` shows both the original and augmented versions of the two-component mixture model for R's Old Faithful dataset. For the original version, we need a user-defined mixture density.
  Here's the posterior inference for the mixture weights and membership.



Auxiliary variables

- **Example (Probit regression, Albert & Chib, 1993):** For $i = 1, \ldots, n$ let $y_i \mid \theta_i \sim \text{Ber}(\theta_i)$ where $\Phi^{-1}(\theta_i) = \mathbf{x}_i^T \boldsymbol{\beta}$, $\mathbf{x}_i$ is a vector of known regressors and $\boldsymbol{\beta}$ is a vector of regression coefficients with $\boldsymbol{\beta} \sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma})$.

  In this case introduce auxiliary variables $z_1, \ldots, z_n$ such that

$$z_i \mid \boldsymbol{\beta} \sim \text{N}(\mathbf{x}_i^T \boldsymbol{\beta}, 1) \qquad y_i = \begin{cases} 1 & z_i \geq 0 \\ 0 & z_i < 0 \end{cases}$$

  The full conditional for $\boldsymbol{\beta}$ is Gaussian, while the full conditional of z_i is a truncated normal. A slight improvement can be obtained if instead of working with $z_i \mid \boldsymbol{\beta}$ you work with $z_i | z_{-i}$ (this is an example of marginalization, see Holmes et al., 2006).

Auxiliary variables

- **Example (Robust regression):** Consider the robust regression model

$$y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \sigma \epsilon_i, \qquad \epsilon \sim t_\nu, \qquad i = 1, \ldots, n.$$

where $\boldsymbol{\beta} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}^2)$, $\sigma^2 \sim \mathrm{IGam}(a, c)$ and $\nu \sim p(\nu)$.

It is well known that the Student t distribution can be written as a scale mixture of normals. Accordingly, introduce auxiliary variables $\lambda_1, \ldots, \lambda_n$ such that

$$y_i \mid \boldsymbol{\beta}, \sigma^2, \lambda_i \sim \mathrm{N}\left(\mathbf{x}_i^T \boldsymbol{\beta}, \frac{\sigma^2}{\lambda_i}\right), \qquad \lambda_i \sim \mathrm{Gam}\left(\frac{\nu}{2}, \frac{\nu}{2}\right).$$

Including $\lambda_1, \ldots, \lambda_n$ in the model makes the full conditional for $\boldsymbol{\beta}$ a Gaussian and the full conditional for σ^2 an inverse Gamma. On the other hand, the full conditional for λ_i is another Gamma, while ν needs to be sampled using a MH step (preferably from the unaugmented model!).

Adaptive MCMC algorithms

- As our Metropolis examples illustrate, one of the main challenges associated with designing MH algorithms is selecting the proposal distribution.
- In particular, concentrate on (Gaussian) random-walk MH algorithms, where the problem is how to select the variance-covariance matrix. What we have done so far requires a lot of back and forth!
- Is there a way in which we can create a "self-tuning" algorithm? This is the main thrust behind adaptive MCMC algorithms!!!

Adaptive MCMC algorithms

- A simple yet powerful idea (at least for elliptical posteriors): start with a potentially very bad covariance matrix and, after the algorithm has been running for a while, switch to different proposal that uses the previous values of the chain to estimate $\text{Cov}(\boldsymbol{\theta} \mid \mathbf{y})$ (just as we did when we manually tuned the chain).
- This is particularly useful when $\dim\{\boldsymbol{\theta}\} = d$ is large!
- The main conceptual issue with this approach is that using the whole history of the chain to create a proposal means that we are not working with a Markov chain anymore (or, at least, not with a time-homogenous one), so the theory we discussed so far cannot be used to ensure that the algorithm actually converges to the desired equilibrium distribution.

Adaptive MCMC algorithms

- **Example (taken from Roberts & Rosenthal, 2009 and Haario et al., 2001):** Consider a target distribution $\mathrm{N}(\mathbf{0}, \mathbf{M}\mathbf{M}^T)$ where the entries of $\mathbf{M}$ have been randomly generated from standard normal distribution.

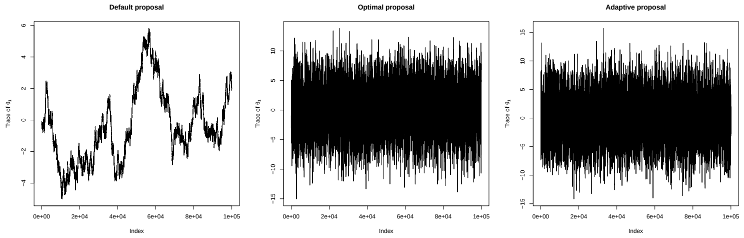
  We consider the case $d = \dim\{\boldsymbol{\theta}\} = 10$. For the first 20,000 iterations, the proposal distribution is fixed and corresponds to a normal distribution, $q(\vartheta \mid \theta) = \mathrm{N}\left(\vartheta \mid \theta, \frac{0.1^2}{d}\mathbf{I}\right)$. After that point, the proposal becomes a mixture

$$q(\vartheta \mid \theta) = (1-\beta)\mathrm{N}\left(\vartheta \mid \theta, \frac{2.38^2}{d}\boldsymbol{\Sigma}_b\right) + \beta\mathrm{N}\left(\vartheta \mid \theta, \frac{0.1^2}{d}\mathbf{I}\right)$$

  where $\boldsymbol{\Sigma}_b$ is the current empirical estimate of the covariance of the target distribution (computed on the basis of the previous $b-1$ iterations of the chain), and $\beta > 0$ is a (small) constant.

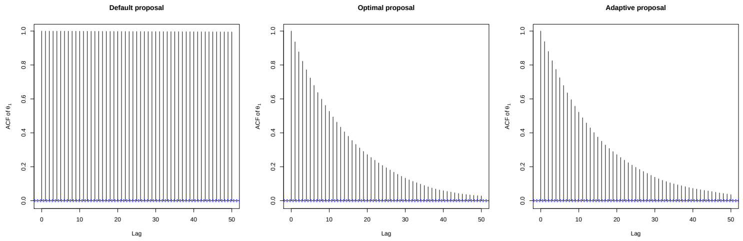
Adaptive MCMC algorithms

- **Example (taken from Roberts & Rosenthal, 2009 and Haario et al., 2001, cont):** The algorithm is implemented in the file `adaptive1.R`, which allows you to run the algorithm using only the "default", only the "optimal", and the adaptive proposal. The graphs below show the trace plots for the first component of $\boldsymbol{\theta}$.



Adaptive MCMC algorithms

- **Example (taken from Roberts & Rosenthal, 2009 and Haario et al., 2001, cont):** The graphs below show the acf functions for the first component of $\boldsymbol{\theta}$.



Acceptance rates are 0.964, 0.264, and 0.260. The effective sample sizes in the last two cases are 3096 and 3200.

Adaptive MCMC algorithms

- **Example (taken from Roberts & Rosenthal, 2009 and Haario et al., 2001, cont):**
  Some things to consider:
  - The use of $20,000$ "preliminary" samples and a "default" variance $\frac{0.1^2}{d}\mathbf{I}$ are totally ad-hoc.
  - More recent schemes (including that in NIMBLE) adapt on the fly (e.g., every 100 iterations), averaging the current proposal covariance with the estimated posterior covariance from recent iterations.
  - Adaptive univariate schemes simply adjust the proposal scale.

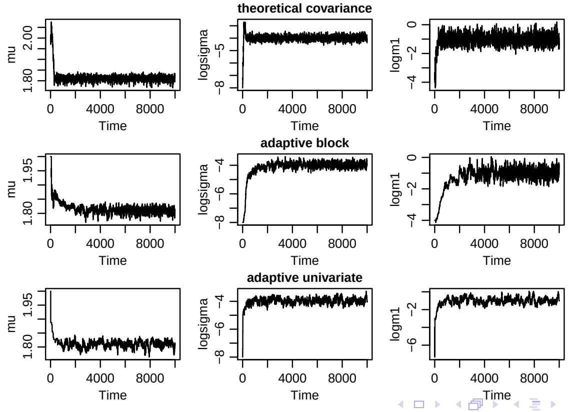
Adaptive MCMC algorithms

NIMBLE's adaptive MCMC

- NIMBLE's default scalar and block Metropolis samplers use adaptation.
- Users can set the initial proposal scale / covariance.
- Current scheme can sometimes perform very poorly when the scales of initial proposal covariance and the posterior covariance are very different.
  - We plan to roll out a fix for this in the next release.

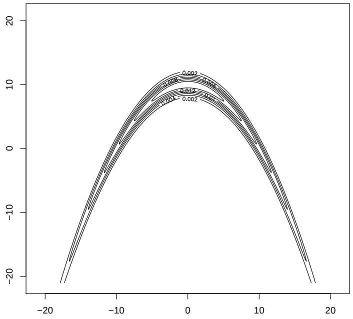
Adaptive MCMC algorithms

- **Example (Nonlinear binomial regression):** in `mh-bliss.R` we have NIMBLE code for non-adaptive (theoretical proposal covariance) and adaptive blocked and univariate Metropolis sampling.



Hamiltonian Monte Carlo

- One of the main challenges for all the previous MCMC methods is dealing with non-elliptical posteriors.
- We would like to exploit the local geometry of the distribution.
- HMC methods allow us to do precisely that!

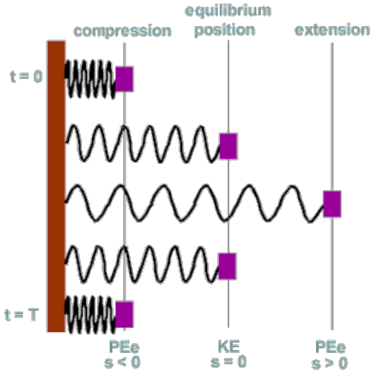


Hamiltonian Dynamics

- The Hamiltonian function H describes the time evolution of a dynamical system (e.g., a spring).
- For Bayesian inference we are interested mainly in Hamiltonians of the form

$$H(\boldsymbol{\theta}, \phi) = U(\boldsymbol{\theta}) + K(\phi)$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the "position", $\phi \in \mathbb{R}^d$ is the "momentum", U is the potential energy and K is the kinetic energy.



Hamiltonian Dynamics

- For our purpose, the "position" $\boldsymbol{\theta}$ corresponds to our parameters of interest and

$$U(\boldsymbol{\theta}) = -\log\left\{p(\boldsymbol{\theta} \mid \mathbf{y})\right\}.$$

- The momemtun ϕ corresponds to auxiliary variables, and the form of the kinetic energy is arbitrary (as long as it is the negative of the logarithm of a well-defined density). Often $\phi \sim \mathrm{N}(\mathbf{0}, \mathbf{M})$ with $\mathbf{M} = \mathrm{diag}\{m_1, \ldots, m_K\}$, so

$$K(\phi) = \frac{1}{2}\phi^T \mathbf{M}\phi = \frac{1}{2}\sum_{k=1}^{d} m_k \phi_k^2.$$

Can be interpreted as the kinetic energy available to d springs (with constants $m_1, \ldots, m_K$) that have been stretched $\phi_1, \ldots, \phi_K$ units away from their equilibrium position.

Hamiltonian Dynamics

- The time evolution of the system is uniquely defined by Hamilton's equations,

$$\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{\phi}}, \qquad \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{\theta}}$$

- The Hamiltonian remains invariant as time evolves, i.e.,

$$\frac{\mathrm{d}H}{\mathrm{d}t} = 0.$$

  (easy to show using Hamilton's equations and the chain rule).

- The dynamics induced by the Hamiltonian equations are reversible. They also preserve volume in the $(\boldsymbol{\theta}, \boldsymbol{\phi})$ space (which implies a unit Jacobian).

Hamiltonian Dynamics and MCMC

The path forward should be clear

- We want to build an MCMC on the augmented space $(\boldsymbol{\theta}, \phi)$ with stationary distribution

$$p(\boldsymbol{\theta}, \phi) \propto \exp\left\{-U(\boldsymbol{\theta}) - K(\phi)\right\}$$

  where $U(\boldsymbol{\theta}) = p(\boldsymbol{\theta} \mid \mathbf{y})$ that uses Hamilton's dynamics (which are reversible and volume-preserving) to generate proposals.
- If we can simulate the dynamic over time exactly, then this is a valid MH proposal, and will always be accepted (because of the time invariance of the Hamiltonian).
- In practice we can only simulate the dynamics approximately by discretizing time. If we do it carefully, the discretized dynamics will still be reversible and volume-preserving (although only be approximately invariant ...).

Discretizing the Hamiltonian Dynamics

- Euler's method (step size ϵ):

$$\phi_i(t+\epsilon) = \phi_i(t) + \epsilon \frac{\mathrm{d}\phi_i}{\mathrm{d}t}(t) = \phi_i(t) - \epsilon \frac{\partial U}{\partial \theta_i}(\boldsymbol{\theta}(t))$$

$$\theta_i(t+\epsilon) = \theta_i(t) + \epsilon \frac{\mathrm{d}\theta_i}{\mathrm{d}t}(t) = \theta_i(t) + \epsilon \frac{\partial K}{\partial \phi_i}(\boldsymbol{\phi}(t))$$

  Discretized dynamics are not necessarily reversible.
- Better: "Leapfrog method" (step size ϵ):

$$\phi_i(t+\epsilon/2) = \phi_i(t) - \frac{\epsilon}{2}\frac{\partial U}{\partial \theta_i}(\boldsymbol{\theta}(t))$$

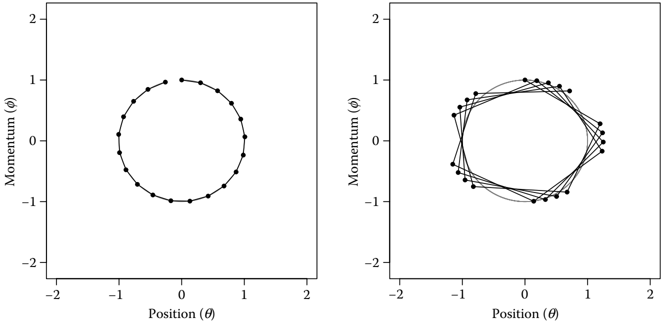
$$\theta_i(t+\epsilon) = \theta_i(t) + \epsilon \frac{\partial K}{\partial \phi_i}(\boldsymbol{\phi}(t+\epsilon/2))$$

$$\phi_i(t+\epsilon) = \phi_i(t+\epsilon/2) - \frac{\epsilon}{2}\frac{\partial U}{\partial \theta_i}(\boldsymbol{\theta}(t+\epsilon))$$

- Even Better: Multiple steps of "Leapfrog".

The Hamiltonian Monte Carlo algorithm

Discretized Hamiltonian trajectories for $H(\theta, \phi) = \theta^2 + \phi^2$ using the leapfrog method. Left panel corresponds to $\epsilon = 0.3$ and the right to $\epsilon = 1.2$.



Taken from Neal (2011).

The Hamiltonian Monte Carlo algorithm

1. Generate $\phi^{(s)} \sim q$, where $q \propto \exp\{-K(\phi)\}$.
2. Initialize $\boldsymbol{\varphi}^{(s)} = \boldsymbol{\phi}^{(s)}$ and $\boldsymbol{\vartheta}^{(s)} = \boldsymbol{\theta}^{(s)}$
3. Update $\boldsymbol{\varphi}^{(s+1/(L+1))} = \boldsymbol{\varphi}^{(s)} - \frac{\epsilon}{2}\frac{\partial U}{\partial \theta_i}(\boldsymbol{\theta}^{(s)})$.
4. For $l = 1, \dots, L-1$
   1. Update $\boldsymbol{\vartheta}^{(s+l/L)} = \boldsymbol{\vartheta}^{(s+(l-1)/L)} + \epsilon\frac{\partial K}{\partial \phi_i}(\boldsymbol{\varphi}^{(s+l/(L+1))})$.
   2. Update $\boldsymbol{\varphi}^{(s+(l+1)/(L+1))} = \boldsymbol{\varphi}^{(s+l/(L+1))} - \epsilon\frac{\partial U}{\partial \theta_i}(\boldsymbol{\vartheta}^{(s+l/L)})$.
5. Update $\boldsymbol{\vartheta}^{(s+1)} = \boldsymbol{\vartheta}^{(s+(L-1)/L)} + \epsilon\frac{\partial K}{\partial \phi_i}(\boldsymbol{\varphi}^{(s+L/(L+1))})$.
6. Update $\boldsymbol{\varphi}^{(s+1)} = \boldsymbol{\varphi}^{(s-L/(L+1))} - \frac{\epsilon}{2}\frac{\partial U}{\partial \theta_i}(\boldsymbol{\vartheta}^{(s+1)})$.
7. Set
$$\boldsymbol{\theta}^{(s+1)} = \begin{cases} \boldsymbol{\vartheta}^{(s+1)} & \text{with prob. } \rho(\boldsymbol{\theta}^{(s)}, \boldsymbol{\phi}^{(s)}, \boldsymbol{\vartheta}^{(s+1)}, \boldsymbol{\varphi}^{(s+1)}) \\ \boldsymbol{\theta}^{(s)} & \text{with prob. } 1-\rho(\boldsymbol{\theta}^{(s)}, \boldsymbol{\phi}^{(s)}, \boldsymbol{\vartheta}^{(s+1)}, \boldsymbol{\varphi}^{(s+1)}) \end{cases}$$

   where

$$\rho(\boldsymbol{\theta}^{(s)}, \boldsymbol{\phi}^{(s)}, \boldsymbol{\vartheta}^{(s+1)}, \boldsymbol{\varphi}^{(s+1)}) = \\ \min\left\{1, \exp\left\{U\left(\boldsymbol{\theta}^{(s)}\right) - U\left(\boldsymbol{\vartheta}^{(s+1)}\right) + K\left(\boldsymbol{\phi}^{(s)}\right) - K\left(-\boldsymbol{\varphi}^{(s+1)}\right)\right\}\right\}$$

The Hamiltonian Monte Carlo algorithm

- **Example (sampling from the Cauchy distribution):** Consider using a Hamiltonian Monte Carlo algorithm to sample from a Cauchy distribution $p(\theta) = \frac{1}{\pi(1+\theta^2)}$. We use a Gaussian distribution for the momentum, so

$$U(\theta) = -\log p(\theta) = \log\left\{1+\theta^2\right\}, \qquad K(\phi) = \frac{\phi^2}{2}.$$

- Hence, we have

$$\frac{\partial U}{\partial \theta} = \frac{2\theta}{1+\theta^2}, \qquad \frac{\partial K}{\partial \phi} = \phi.$$

The Hamiltonian Monte Carlo algorithm

- **Example (sampling from the Cauchy distribution, cont):** If we take a single step ($L = 1$) of length ϵ

$$\theta^{(p)} = \theta^{(c)} \left\{ 1 - \frac{\epsilon^2}{1 + \left(\theta^{(p)}\right)^2} \right\} + \epsilon \phi^{(c)}$$

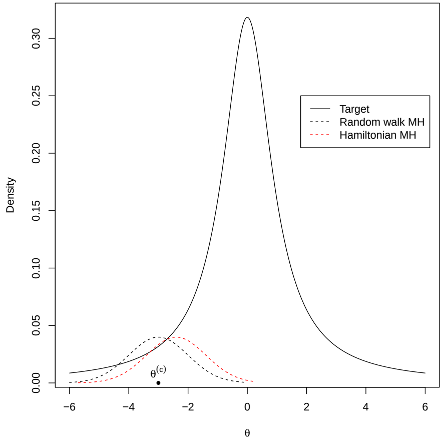
  where $\phi^{(c)} \sim \mathsf{N}(0, 1)$.
- This looks similar to what you would get from a random walk proposal with standard deviation ϵ, $\theta^{(p)} = \theta^{(c)} + \epsilon\omega$ where $\omega \sim \mathsf{N}(0, 1)$.
- However, there are some important differences!

The Hamiltonian Monte Carlo algorithm

- **Example (sampling from the Cauchy distribution, cont):** Suppose $\theta^{(p)} = -3$.
- The random walk proposal has the same probability of proposing a value to the right of $\theta^{(p)}$ as it does a value to the left.
- For small ϵ we have $\left\{1 - \frac{\epsilon^2}{1+\left(\theta^{(p)}\right)^2}\right\} \in [0,1]$. Hence, the Hamiltonian MH algorithm has a higher probability of proposing value to the right of -1 than values to its left. This is what you would intuitively would like to do!
- A similar argument applies for positive values of $\theta^{(p)}$.

The Hamiltonian Monte Carlo algorithm

- **Example (sampling from the Cauchy distribution, cont):** Suppose $\theta^{(c)} = -3$ and $\epsilon = 1$

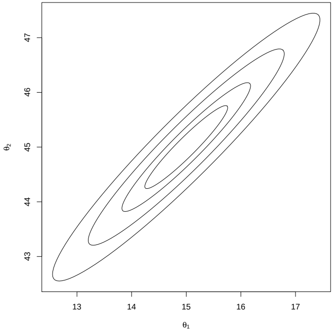


The Hamiltonian Monte Carlo algorithm

Example (bivariate normal target): Consider sampling from the bivariate normal distribution

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} 15 \\ 45 \end{pmatrix}, \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix} \right)$$

using both a Hamiltonian Monte Carlo and a bivariate random walk.



The Hamiltonian Monte Carlo algorithm

- **Example (bivariate normal target, cont):** For the Hamiltonian algorithm we use independent Gaussian distributions for the momentum variables ϕ_1 and ϕ_2. The gradients are

$$\nabla U(\boldsymbol{\theta}) = \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix}^{-1} \boldsymbol{\theta} \qquad \nabla K(\boldsymbol{\phi}) = \boldsymbol{\phi}$$

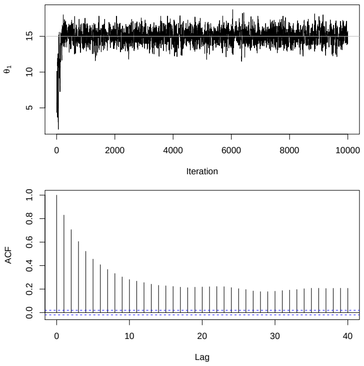
- For the random walk Metropolis algorithm, we use an optimal bivariate Gaussian proposal

$$\boldsymbol{\vartheta}^{(s+1)} \mid \boldsymbol{\theta}^{(s)} \sim \mathsf{N}\left(\boldsymbol{\theta}^{(s)}, \frac{2.38^2}{2} \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix} \right)$$

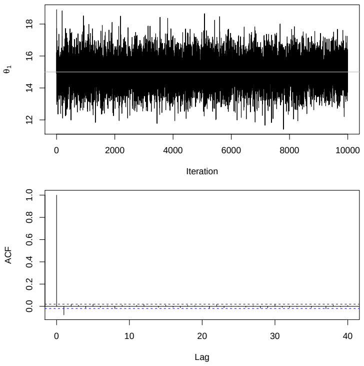
- Code in `hamiltonian_bivariatenormal.R`.

The Hamiltonian Monte Carlo algorithm

Random walk
Acceptance rate 36%



Hamiltonian
Acceptance rate 89%

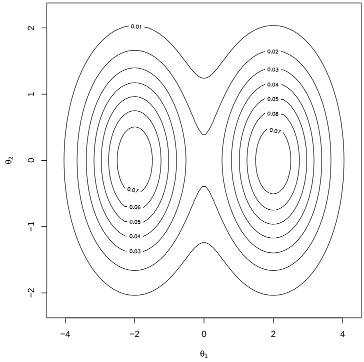


Hamiltonian Dynamics

- Hamiltonian algorithms can be useful in moving you across modes because proposals happen (approximately) along constant-energy contours.
- For example, consider sampling from a mixture of two normals

$$0.5\mathrm{N}(-2,1) + 0.5\mathrm{N}(2,1)$$

using a standard Gaussian distribution for the momentum variables.



Tuning

- Hamiltonian Monte Carlo algorithms are not totally automatic, as you still need to decide the size and number of steps, as well as (at least) the covariance matrix associated with the momentum variables.

- As with RWMH, tuning HMC algorithms usually requires pilot runs for different values of ϵ and L. Be careful, optimal values might be different before and after convergence!
- The value of ϵL roughly controls how far you move from the current point. However, a large value of ϵL does not necessarily lead to longer-distance moves (recall the graphs for the Hamiltonian trajectories in $(\boldsymbol{\theta}, \phi)$).
- Indeed, the chain produced by a Hamiltonian algorithm could potentially be periodic, but that is rarely the case in practice.

Tuning

- Assuming that ϵL is kept roughly constant, the smaller ϵ is, the closest the discretized trajectory is to the continuous time one, but more computations are required.
- The rejection rate of the algorithm is controlled mainly by ϵ (i.e., how good the discrete-time approximation is). Values of ϵ that are too big might lead to unstable trajectories and very low acceptance rates, and the problem might not be obvious in pilot runs because the rejection rate might be very different in different regions of the space.
- As a rule of thump, the optimal acceptance rate for HMC is around 65% for high-dimensional problems (against the $\sim$ 23% for high-dimensional RWMH).

Tuning

- If an estimate of $\text{Var}(\boldsymbol{\theta} \mid \mathbf{y})$ is available, applying a linear transformation for the position variables so that they have (approximately) variance 1 and no correlation and using independent standard Gaussian momentum variables leads to good performance with a small number of steps
- Note that under Gaussian momentum, performance of the algorithm is invariant to linear transformations $\mathbf{B}\theta$ if the momentum variables are multiplied by $(\mathbf{B}')^{-1}$. Hence, alternatively we leave the positions untransformed and use correlated Gaussian momentum variables.
- In any case, it is a good idea to use different scales for the each momentum variable if the position variables have very different variances.

Implementation

- Tuning HMC algorithms can be tricky, but the Stan software was developed for just this purpose. Stan uses its own language for specifying the statistical model.
- `hmc-cox.R` provides a Stan implementation for a Cox survival analysis model. See Appendix for more details on Stan and the Cox example.
- NIMBLE doesn't provide HMC because we're just now implementing automatic differentation and because it's Stan's sweet spot.

Special cases and extensions

- A single step HMC (i.e., $L = 1$) corresponds to a pre-conditioned Langevin diffusion as used in a Metropolis-Adjusted Langevin Algorithm (MALA).
- Note that HMC cannot deal with discrete parameters (but see new work from David Dunson's group).
- HMC can be seen as just another MH kernel, so it can be used for subsets of parameters and combined with other kernels (gives you a way to separate discrete parameters).
- A number of useful extensions, examples include
  - Riemmanian HMC (non-constant $\mathbf{M}$).
  - "Splitting" the Hamiltonian
  - Partial momentum refreshment.

Model selection and hypothesis testing

- Most of the discussion so far has focused on Morte Carlo methods for point and interval estimation.
- We move now to discuss MCMC methods in the context of model selection and hypothesis testing.
- The approach we will focus on turns the model selection problem into an estimation problem for a discrete random variable (the model indicator).

Variable selection

- **Example (variable selection in linear models):** Consider a linear model of the form $\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\dim\{\mathbf{y}\} = n$, and $\dim\{\boldsymbol{\theta}\} = p$. We are interested in selecting a subset of the columns of $\mathbf{X}$ that explain the response $\mathbf{y}$.

  Hence, since we have a total of p variables, we need to select among 2^p models. Each of these models can be represented using a vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ where $\gamma_k = 1$ if variable k is in the model, and $\gamma_k = 0$ otherwise.

  To address this problem from a Bayesian perspective we need to compute the probability associated with each model. To do so we need a prior distribution $p(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma})$ which we will conveniently breakdown as $p(\boldsymbol{\theta} \mid \sigma^2, \gamma)p(\sigma^2 \mid \gamma)p(\gamma)$.

Variable selection

- **Example (variable selection in linear models, cont):** For the regression coefficients, we focus on priors of the form

$$\boldsymbol{\theta} \mid \sigma^2, \boldsymbol{\gamma} \sim \mathrm{N}\left(\boldsymbol{\theta}_\gamma \mid \mathbf{0}, c\sigma^2 \left\{\mathbf{X}_\gamma^T \mathbf{X}_\gamma\right\}^{-1}\right) \prod_{k:\gamma_k=0} \delta_0(\theta_k),$$

  $\mathbf{X}_\gamma$ denotes the matrix composed of the columns of $\mathbf{X}$ for which associated with variables that have γ_k, and $\delta_0(\cdot)$ denotes the degenerate measure at 0.
  For the variance we have $\sigma^2 \mid \boldsymbol{\gamma} \sim \mathrm{IGam}(a, b)$ (which is independent of $\boldsymbol{\gamma}$).
  Set $p(\boldsymbol{\gamma}) = \frac{\Gamma(1+\sum_{k=1}^{p} \gamma_i)\Gamma(1+p-\sum_{k=1}^{p} \gamma_i)}{\Gamma(p+2)}$, (Beta-Bernoulli prior).
  Set $c = n$ (more genrally, a prior centered on n.)

Variable selection

- **Example (variable selection in linear models, cont):** Note that in this analysis we are really interested in the marginal posterior $p(\gamma \mid \mathbf{y})$, and we can think of $\boldsymbol{\theta}$ and σ^2 are "nuisance" parameters. Integrating them out we have

$$p(\mathbf{y} \mid \gamma) = \left(\frac{1}{2\pi}\right)^{\frac{n}{2}} \left| \mathbf{I} + c\mathbf{X}_\gamma \left(\mathbf{X}_\gamma^T \mathbf{X}_\gamma\right)^{-1} \mathbf{X}_\gamma^T \right|^{-\frac{1}{2}} \frac{b^a}{\Gamma(a)} \frac{\Gamma\left(a + \frac{n}{2}\right)}{\left(b + \frac{\mathbf{y}^T \left\{\mathbf{I} + c\mathbf{X}_\gamma \left(\mathbf{X}_\gamma^T \mathbf{X}_\gamma\right)^{-1} \mathbf{X}_\gamma^T\right\}^{-1} \mathbf{y}}{2}\right)^{a+\frac{n}{2}}},$$

and the posterior distribution is simply

$$p(\gamma \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \gamma) p(\gamma)}{\sum_{\gamma'} p(\mathbf{y} \mid \gamma') p(\gamma')}.$$

Variable selection

- **Example (variable selection in linear models, cont):** Because γ can only take a finite number of values, getting these posterior probabilities should be straightforward:
  - This is true when p is relatively small (say $p \leq 20$). In that case the number of models is manageable (with $p = 20$ we have 1,048,576 models), and we can actually compute (and store) the probabilities for each model with the resources available in regular computers.
  - However, when p is large we cannot do that anymore (roughly, every 10 extra variables you add multiplies the number of models by 1,000, so you can easily run out of atoms in the universe in a typical problem in genetics ...)

  Hence, even in discrete problems with an "analytical" solutions you might want to use simulations to make the problem more manageable.

Variable selection: A Metropolis-Hastings algorithm

- **Example (variable selection in linear models, cont):** A couple of options for constructing a Metropolis-Hasting algorithm:
    - A simple random walk proposal can be constructed by choosing one entry γ_k of $\boldsymbol{\gamma}$ uniformly at random and generating a new model by setting
    $$\boldsymbol{\vartheta} = (\gamma_1, \ldots, \gamma_{k-1}, 1 - \gamma_k, \gamma_{k+1}, \ldots, \gamma_p)$$
    - An independent proposal could be constructed by generating a new model by randomly sampling from a uniform distribution over model, i.e., by letting $\vartheta_k \sim \text{Ber}(1/2)$ independently for each k.

  The acceptance probability in both cases is
  $$\rho(\boldsymbol{\vartheta}, \boldsymbol{\gamma}) = \min\left\{1, \frac{p(\boldsymbol{\vartheta} \mid \mathbf{y})}{p(\boldsymbol{\gamma} \mid \mathbf{y})}\right\}.$$

Variable selction

- **Example (variable selection in a GLMM):** Gelman & Hill (2007) analyzes a dataset of support provided by unmarried fathers to their children's mothers. The model features are:
  - observations of binary outcomes for each mother/father pair of informal support to the mother
  - observations grouped within 20 cities (with random effects to account for correlation)
  - individual-level predictors
  - city-level predictors including the intensity of child support enforcement by the city

  Scientific question: does enforcement by the city impact whether support is provided?

  Model: probit regression with variable selection.

Reversible Jump

- In the case of the linear model we could integrate out the regression coefficients and work exclusively with the vector γ. Hence, although the size of $\boldsymbol{\theta}$ changes with γ, we did not really have to deal with the varying dimension.
- In the case of the generalized linear model, it is not obvious how we can integrate out the coefficients explicitly. Hence, our algorithm needs to deal with the fact that the number of coefficients can change.
- A slight generalization of Metropolis Hastings algorithms called reversible jump MCMC (Green, 1995; Brooks et al., 2003) can be used in this context.

Reversible Jump

- Let's start by establishing some notation.
  - Let K index the model of interest. Associate with such model a prior $p(K)$.
  - Each model $\mathcal{M}_K$ has a set of distinct set of parameters $\boldsymbol{\theta}_K$. The associated prior is $p(\boldsymbol{\theta}_K \mid K)$ which depends (explicitly or implicitly) on the dimension K.
  - The model $\mathcal{M}_K$ and the parameters $\boldsymbol{\theta}_K$, data is generated according to a likelihood $p(\mathbf{y} \mid K, \boldsymbol{\theta}_K)$.
  - We denote the corresponding posterior distribution as $p(K, \boldsymbol{\theta}_K \mid \mathbf{y})$ which can be factorized as

$$p(K, \boldsymbol{\theta}_K \mid \mathbf{y}) = p(K \mid \mathbf{y}) p(\boldsymbol{\theta}_K \mid K, \mathbf{y})$$

- Hence, if we could compute $p(K \mid \mathbf{y})$ for all K, then model selection would be straightforward. However, this involves integrating over all other parameters of model!

Reversible Jump

- To derive the algorithm:
    - First, augment each vector $\boldsymbol{\theta}_K$ with auxiliary variables $\mathbf{u}_K$, so that the dimension of $(\boldsymbol{\theta}_K, \mathbf{u}_K)$ *is the same* for all K.
    - Second, define a (consistent) collection of deterministic transformations $T_{K,K'}\left\{(\boldsymbol{\theta}_K, \mathbf{u}_K), (\boldsymbol{\theta}_{K'}, \mathbf{u}_{K'})\right\}$ that map the augmented parameter spaces of different models (this is typically done by defining transformation only between "neighboring" models).
- Note that, in some sense, the proposals are *deterministic* because the transformation is deterministic. (However, the presence of the auxiliary variables $\mathbf{u}_K$ introduces randomness).

Reversible Jump

- The acceptance probability becomes

$$\left\{1, \frac{p\left(K^{(p)}, \theta^{(p)}_{K^{(p)}} \mid \mathbf{y}\right)}{p\left(K^{(c)}, \theta^{(c)}_{K^{(c)}} \mid \mathbf{y}\right)} \frac{g_{K^{(p)}}\left(\mathbf{u}^{(p)}_{K^{(p)}}\right)}{g_{K^{(c)}}\left(\mathbf{u}^{(c)}_{K^{(c)}}\right)} \frac{d_{K^{(p)},K^{(c)}}}{d_{K^{(c)},K^{(p)}}} \left|\frac{\partial T_{K^{(c)},K^{(p)}}\left(\boldsymbol{\theta}^{(c)}_{K^{(c)}}, \mathbf{u}^{K^{(c)}}_{(c)}\right)}{\partial\left(\boldsymbol{\theta}^{(p)}_{K^{(p)}}, \mathbf{u}^{K^{(p)}}_{(p)}\right)}\right|\right\}$$

  where $g_K(\mathbf{u}_K)$ is the density of the auxiliary variables and $d_{K,K'}$ is the probability of proposing a move to state K' when you are currently in state K.

- Note that the auxiliary variables $\mathbf{u}_K$ do not need to be stored and can be simulated on the fly ...

Reversible Jump

- With these preliminaries, and assuming the current state of your chain is $\left(K^{(c)}, \boldsymbol{\theta}^{(c)}_{K^{(c)}}\right)$ the RJMCMC algorithm is:
  1. Propose a move to model $\mathcal{M}_{K^{(p)}}$ with probability $d_{K^{(c)},K^{(p)}}$.
  2. Generate $\mathbf{u}^{(p)} \sim g_{K^{(p)}}$ and $\mathbf{u}^{(c)} \sim g_{K^{(c)}}$.
  3. Set $\left(\boldsymbol{\theta}^{(p)}_{K^{(p)}}, \mathbf{u}^{(p)}_{K^{(p)}}\right) = T_{K^{(c)},K^{(p)}}\left(\boldsymbol{\theta}^{(c)}_{K^{(c)}}, \mathbf{u}^{(c)}_{K^{(c)}}\right)$.
  4. Accept this proposal with probability

$$\left\{1, \frac{p\left(K^{(p)}, \theta^{(p)}_{K^{(p)}} \mid \mathbf{y}\right)}{p\left(K^{(c)}, \theta^{(c)}_{K^{(c)}} \mid \mathbf{y}\right)} \frac{g_{K^{(p)}}\left(\mathbf{u}^{(p)}_{K^{(p)}}\right)}{g_{K^{(c)}}\left(\mathbf{u}^{(c)}_{K^{(c)}}\right)} \frac{d_{K^{(p)},K^{(c)}}}{d_{K^{(c)},K^{(p)}}} \left|\frac{\partial T_{K^{(c)},K^{(p)}}\left(\boldsymbol{\theta}^{(c)}_{K^{(c)}}, \mathbf{u}^{K^{(c)}}_{(c)}\right)}{\partial\left(\boldsymbol{\theta}^{(p)}_{K^{(p)}}, \mathbf{u}^{K^{(p)}}_{(p)}\right)}\right|\right\}$$

Reversible Jump

- The deterministic move can be justified as the limit of random proposals whose variability goes to zero.
- Remember that the proposals need to be reversible!
- Note that there are potentially a number of simplifications in the previous acceptance probability. For example, you typically do not need to generate all the entries of $\mathbf{u}^{(c)}_{K^{(c)}}$ and $\mathbf{u}^{(p)}_{K^{(p)}}$.
- The transitions above need to be combined with "regular" transition kernels within each dimension!

Reversible Jump

- **Example (variable selection in a GLMM):** Reversible jump here is quite simple. At each iteration propose to add or remove the variable depending on whether it is currently in the model.
  The deterministic transformation between the full and reduced models is $T_{0,1}(\theta, u) = (\theta, \beta_e)$, so $\beta_e^{(p)} = u^{(c)}$, where θ are the parameters always in the model, β_e is the enforcement coefficient, and u is the single auxiliary variable.
  $d_{0,1} = d_{1,0} = 1$ and the Jacobian is one, so the proposals involve:
  - Addition: propose $\beta_e^{(p)} = u^{(c)}$ from $u^{(c)}$ $g_0(u)$ with term $1/g_0(u^{(c)})$ in the acceptance ratio
  - Removal: propose removal $(u^{(p)} = \beta_e^{(c)})$ with term $g_0(\beta_e^{(c)})$ in the acceptance ratio

  Note that auxiliary variable density $g_0(u)$ plays the role of a constant proposal distribution for β_e so should be well-chosen.
  See `rj-childSupport.R` for the implementation using a user-defined sampler in NIMBLE.

Importance sampling

- Let's go back to the problem of approximating integrals of the form
$$\int h(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}) \mathrm{d}\boldsymbol{\theta},$$
where h is an integrable function and $p(\boldsymbol{\theta} \mid \mathbf{y})$ is the posterior distribution induced by our model.
- We return to the idea of "replacing" $p(\boldsymbol{\theta} \mid \mathbf{y})$ with $g(\boldsymbol{\theta})$ from which it is easy to generate, we can write
$$\begin{aligned}\int h(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}) \mathrm{d}\boldsymbol{\theta} &= \int h(\boldsymbol{\theta}) \frac{p(\boldsymbol{\theta} \mid \mathbf{y})}{g(\boldsymbol{\theta})} g(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \\ &= \int h(\boldsymbol{\theta}) w(\boldsymbol{\theta}, \mathbf{y}) g(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta},\end{aligned}$$
where $w(\boldsymbol{\theta}, \mathbf{y}) = \frac{p(\boldsymbol{\theta}|\mathbf{y})}{g(\boldsymbol{\theta})}$ is a "weight" function.

Importance sampling

- As long as the support of g contains the support of $p(\boldsymbol{\theta} \mid \mathbf{y})$, the WLLN ensures that if $\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_1^{(B)}$ are independent and identically distributed samples from g then

$$\frac{\sum_{b=1}^{B} h\left(\boldsymbol{\theta}^{(b)}\right) w\left(\boldsymbol{\theta}^{(b)}, \mathbf{y}\right)}{\sum_{b=1}^{B} w\left(\boldsymbol{\theta}^{(b)}, \mathbf{y}\right)} \xrightarrow[B\to\infty]{} \mathrm{E}\left\{h\left(\boldsymbol{\theta}^{(b)}\right)\right\}$$

- Almost any g works in principle. However, if you want your estimator to have a finite variance, then you need g such that

$$\int h^2(\boldsymbol{\theta}) \frac{p^2(\boldsymbol{\theta} \mid \mathbf{y})}{g(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} < \infty.$$

A necessary condition for this to be satisfied is that g has heavier tails than $p(\boldsymbol{\theta} \mid y)$!

Importance sampling

- The optimal importance sampling distribution should be a good approximation to $h(\boldsymbol{\theta})p(\boldsymbol{\theta} \mid \mathbf{y})$ rather than a good approximation to $p(\boldsymbol{\theta} \mid \mathbf{y})$. For example, consider $p = \Pr(X > 2)$ where $X \sim \text{Cauchy}(0, 1)$.
  - The naive estimator is given by

$$\hat{p}_1 = \frac{1}{B} \sum_{b=1}^{B} \mathbb{I}_{\{x^{(b)} > 2\}},$$

    where $x^{(b)} \sim \text{Cauchy}(0, 1)$. Its variance $0.127/B$.
  - A much better alternative is to estimate p using

$$\hat{p}_2 = \frac{1}{B} \sum_{b=1}^{B} \frac{\left(x^{(b)}\right)^{-2}}{2\pi \left\{1 + \left(x^{(b)}\right)^{-2}\right\}},$$

    where $x^{(b)} \sim \text{Uni}[0, 1/2]$. Its variance $0.095/B$.

Importance sampling

- A way to evaluate the efficiency of the algorithm is through the equivalent sample size (ESS):

$$ESS = \frac{\left\{\sum_{b=1}^{B} w\left(\boldsymbol{\theta}^{(b)}, \mathbf{y}\right)\right\}^2}{\sum_{b=1}^{B}\left\{w\left(\boldsymbol{\theta}^{(b)}, \mathbf{y}\right)\right\}^2}.$$

  The interpretation of this ESS is analogous to that of the ESS we discussed for MCMC methods. In particular, if all the weights are identical then $ESS = B$, while if all weight is concentrated on a single particle then $ESS = 1$.

Importance sampling

- **Example (unknown Gaussian mean with Cauchy priors):** Consider a simple model where $y_1, \ldots, y_n$ are iid with $y_i \sim \mathrm{N}(\theta, 1)$, and where we use a "robust" prior $\theta \sim \mathrm{Cauchy}(0, 1)$. The posterior distribution is proportional to

$$\left(\frac{1}{1+\theta^2} \right) \exp\left\{ -\frac{n}{2} \left(\theta^2 - 2\theta\bar{y} \right) \right\}.$$

We are interested in computing the posterior mean:

$$\mathrm{E}\left\{ \theta \mid \mathbf{y} \right\} = \frac{\int_{-\infty}^{\infty} \left(\frac{\theta}{1+\theta^2} \right) \exp\left\{ -\frac{n}{2} \left(\theta^2 - 2\theta\bar{y} \right) \right\} \mathrm{d}\theta}{\int_{-\infty}^{\infty} \left(\frac{1}{1+\theta^2} \right) \exp\left\{ -\frac{n}{2} \left(\theta^2 - 2\theta\bar{y} \right) \right\} \mathrm{d}\theta}.$$

An analytic solution for this expectation is not available.

Importance sampling

- **Example (unknown Gaussian mean with Cauchy priors):** You could use the prior as the importance distribution:

$$\mathrm{E}\left\{\theta \mid \mathbf{y}\right\} \approx \frac{\sum_{b=1}^{B} \theta^{(b)} \exp\left\{-\frac{n}{2}\left(\theta^{(b)2} - 2\theta^{(b)}\bar{y}\right)\right\}}{\sum_{b=1}^{B} \exp\left\{-\frac{n}{2}\left(\theta^{(b)2} - 2\theta^{(b)}\bar{y}\right)\right\}}.$$

  This estimator is implemented in `impsampex_v1.R`.
  A more reasonable importance function use knowledge about the shape of $p(\theta \mid \mathbf{y})$

$$\mathrm{E}\left\{\theta \mid \mathbf{y}\right\} \approx \frac{\sum_{b=1}^{B} \theta^{(b)} \left(\frac{1+n\{\theta^{(b)}-\bar{y}\}^2}{1+\theta^{2(b)}}\right) \exp\left\{-\frac{n}{2}\left(\theta^{(b)2} - 2\theta^{(b)}\bar{y}\right)\right\}}{\sum_{b=1}^{B} \left(\frac{1+n\{\theta^{(b)}-\bar{y}\}^2}{1+\theta^{2(b)}}\right) \exp\left\{-\frac{n}{2}\left(\theta^{(b)2} - 2\theta^{(b)}\bar{y}\right)\right\}}.$$

  where $\theta^{(b)} \sim \mathrm{Cauchy}(\bar{y}, 1/n)$. See `impsampex_v2.R`.

Importance sampling

- Importance sampling schemes can also be interpreted as providing an approximation to $p(\boldsymbol{\theta} \mid \mathbf{y})$:

$$\hat{p}_B(\boldsymbol{\theta} \mid \mathbf{y}) = \sum_{b=1}^{B} \omega\left(\boldsymbol{\theta}^{(b)}, \mathbf{y}\right) \delta_{\boldsymbol{\theta}^{(b)}}(\boldsymbol{\theta}),$$

  where $\delta_x(\boldsymbol{\theta})$ denotes the Dirac point mass at x.
- Then, a fancy way to write the importance sampling estimator is as

$$\int h(\boldsymbol{\theta})\hat{p}_B(\boldsymbol{\theta} \mid y) \mathrm{d}\boldsymbol{\theta} \to \int h(\boldsymbol{\theta}) p_B(\boldsymbol{\theta} \mid y) \mathrm{d}\boldsymbol{\theta}.$$

  Since this is true for any integrable function h, it implies that

$$\hat{p}_B(\boldsymbol{\theta} \mid y) \to p_B(\boldsymbol{\theta} \mid y) \qquad \text{in distribution.}$$

Sequential Importance Sampling

- When $\boldsymbol{\theta}$ is high dimensional, constructing a good instrumental distribution can be very hard.
- One option is to construct it in pieces by (for example) writing

$$\begin{aligned} g(\boldsymbol{\theta} \mid \mathbf{y}) = g(\theta_T \mid \theta_{T-1}, \theta_{T-2}, \ldots, \theta_1, \mathbf{y}) \\ g(\theta_{T-1} \mid \theta_{T-2}, \ldots, \theta_1, \mathbf{y}) \cdots g(\theta_2 \mid \theta_1, \mathbf{y}) g(\theta_1 \mid \mathbf{y}) \end{aligned}$$

- Particularly useful for models of the form:

$$p(\boldsymbol{\theta}_{1:T} \mid \mathbf{y}_{1:T}) \propto \prod_{t=1}^{T} p\left(\mathbf{y}_t \mid \mathbf{y}_{1:(t-1)}, \boldsymbol{\theta}_{1:t}\right) p\left(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{1:(t-1)}\right)$$

where $\mathbf{y}_{1:t} = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$ and $\boldsymbol{\theta}_{1:t} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t)$.The key is that $p\left(\mathbf{y}_t \mid \mathbf{y}_{1:(t-1)}, \boldsymbol{\theta}_{1:t}\right)$ does not depend on $\boldsymbol{\theta}_{(t+1):T}$.

Sequential Importance Sampling

- The SIS algorithm proceeds by repeating the following steps for $t = 1, \dots, T$:
  - Given a representation $\hat{p}\left(\boldsymbol{\theta}_{1:t} \mid \mathbf{y}_{1:t}\right) = \sum_{b=1}^{B} \omega_t^{(b)} \delta_{\hat{\boldsymbol{\theta}}_{1:t}^{(b)}}$
    1. For $b = 1, \dots, B$ sample $\hat{\boldsymbol{\theta}}_{t+1}^{(b)} \sim g\left(\boldsymbol{\theta}_{t+1}^{(b)} \mid \boldsymbol{\theta}_{1:t}, \mathbf{y}\right)$ and let $\hat{\boldsymbol{\theta}}_{1:(t+1)}^{(b)} = \left(\hat{\boldsymbol{\theta}}_{1:t}^{(b)}, \hat{\boldsymbol{\theta}}_{t+1}^{(b)}\right)$.
    2. Construct $\hat{p}\left(\boldsymbol{\theta}_{1:(t+1)} \mid \mathbf{y}_{1:(t+1)}\right) = \sum_{b=1}^{B} \omega_{t+1}^{(b)} \delta_{\hat{\boldsymbol{\theta}}_{1:(t+1)}^{(b)}}$ where

$$\omega_{t+1}^{(b)} \propto \omega_t^{(b)} \frac{p\left(\mathbf{y}_{1:(t+1)} \mid \mathbf{y}_{1:t}, \hat{\boldsymbol{\theta}}_{1:(t+1)}^{(b)}\right) p\left(\hat{\boldsymbol{\theta}}_{t+1}^{(b)} \mid \hat{\boldsymbol{\theta}}_{1:t}^{(b)}\right)}{g\left(\hat{\boldsymbol{\theta}}_{t+1}^{(b)} \mid \hat{\boldsymbol{\theta}}_{1:t}^{(b)} \mathbf{y}\right)}.$$

- Note that the end of this process we have a particle representation for $p(\boldsymbol{\theta}_{1:T} \mid \mathbf{y}_{1:T})$ (the joint posterior).

Sequential Importance Sampling

- The main issue with SIS is degeneracy. As the algorithm proceeds, a single particle tends to receive all the weight $\Rightarrow$ ESS tends to be very small.
- This is true even if the instrumental distributions are chosen optimally (which is very rarely the case!)
- Hence, for large T, estimates derived from the algorithm can have enormous variances.
- Although degeneracy is an issue with every sequential Monte Carlo scheme, but it can often be alleviated by focusing on $p(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t})$ (the marginal for the parameters at time t) rather than $p(\boldsymbol{\theta}_{1:t} \mid \mathbf{y}_{1:t})$.

State-space models

- State space models, which are particularly amenable to computation using variations of importance sample, are defined recursively through an *observational equation*

$$\mathbf{y}_t \mid \boldsymbol{\theta}_t \sim p(\mathbf{y}_t \mid \boldsymbol{\theta}_t), \qquad t = 1, \dots, T,$$

  and an *evolution equation*

$$\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1} \sim p(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}), \qquad t = 1, \dots, T,$$

  (together with an *initial distribution* $\boldsymbol{\theta}_0 \sim p(\boldsymbol{\theta}_0)$).
- We typically interpret the index t as representing "time", but it could represent other things (e.g., position in the genome).
- State-space models include a number of well known models (hidden Markov models, dynamic linear models, ARMA models, etc.)

State-space models

- With state-space models there is typically two types of tasks that we are interested in carrying out:
  - Computing the posterior distribution $p(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t})$ for each $t = 1, \ldots, T$. Typically we want to do this by somehow "updating" $p(\boldsymbol{\theta}_{t-1} \mid \mathbf{y}_{1:(t-1)})$ rather than recomputing from scratch. We call this process *filtering*.
  - Computing the posterior distribution $p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T \mid \mathbf{y}_{1:T})$, preferably by reusing information available from the computation of the filtering distributions. We call this process *smoothing*.
- Note that we usually do not care about $p(\boldsymbol{\theta}_{1:t} \mid \mathbf{y}_{1:t})$.
- Also, if we can carry out the filtering step, we can also easily make predictions.

Filtering: Sampling Importance Resampling

- Given the Markovian definition of state-space models, it is very natural to employ proposal distributions of the form $g(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \mathbf{y}_t)$ (i.e., one that depends only on the previous state and the current observation).
- Furthermore, although using the prior as your instrumental distribution is typically a poor idea, the simplest algorithms further assume that $g(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \mathbf{y}_t) = p(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1})$. This leads to simplifications in the form of the weights
- Finally, to increase the diversity of the particles, rather than just updating the weights consider adding a resampling step in which a subset of the particle at time t are selected to create the particles at $t+1$.
- These modifications to SIS lead to the sampling importance resampling (SIR) algorithm.

Filtering: Sampling Importance Resampling

- The SIR algorithm proceeds by repeating the following steps for $t = 1, \ldots, T$:
  - Given a particle representation $\hat{p}\left(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t}\right) = \sum_{b=1}^{B} \omega_t^{(b)} \delta_{\hat{\boldsymbol{\theta}}_t^{(b)}}$
    1. For $b = 1, \ldots, B$ sample $\tilde{\boldsymbol{\theta}}_t^{(b)} \sim \hat{p}\left(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t}\right)$.
    2. For $b = 1, \ldots, B$ sample $\hat{\boldsymbol{\theta}}_{t+1}^{(b)} \mid \tilde{\boldsymbol{\theta}}_t^{(b)} \sim p\left(\boldsymbol{\theta}_{t+1} \mid \tilde{\boldsymbol{\theta}}_t^{(b)}\right)$.
    3. Construct $\hat{p}\left(\boldsymbol{\theta}_{t+1} \mid \mathbf{y}_{1:(t+1)}\right) = \sum_{b=1}^{B} \omega_{t+1}^{(b)} \delta_{\hat{\boldsymbol{\theta}}_{t+1}^{(b)}}$ where

$$\omega_{t+1}^{(b)} = \frac{p\left(\mathbf{y}_{t+1} \mid \hat{\boldsymbol{\theta}}_{t+1}^{(b)}\right)}{\sum_{\ell=1}^{B} p\left(\mathbf{y}_{t+1} \mid \hat{\boldsymbol{\theta}}_{t+1}^{(\ell)}\right)}.$$

- It is very important to note that because of the resampling step in (1) the dependence between $\boldsymbol{\theta}_t^{(b)}$ and $\boldsymbol{\theta}_{t+1}^{(b)}$ is broken and the SIR algorithm only gives you a representation for $p\left(\boldsymbol{\theta}_{t+1} \mid \mathbf{y}_{1:(t+1)}\right)$ rather than for $p\left(\boldsymbol{\theta}_{1:(t+1)} \mid \mathbf{y}_{1:(t+1)}\right)$

Filtering: Sampling Importance Resampling

- **Example (a simple linear Gaussian system):** Consider the linear Gaussian state-space model with

$$y_t \mid \theta_t \sim \mathrm{N}(\theta_t, \sigma^2), \quad \theta_t \mid \theta_{t-1} \sim \mathrm{N}(\theta_{t-1}, \tau^2), \quad \theta_0 \sim \mathrm{N}(m_0, C_0^2).$$

If we assume that σ^2 and τ^2 are known, then the filtering distributions can be obtained in closed form. In particular, $\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t} \sim \mathrm{N}\left(m_t, C_t^2\right)$ where

$$m_t = \frac{\frac{y_t}{\sigma^2} + \frac{m_{t-1}}{\tau^2 + C_{t-1}^2}}{\frac{1}{\sigma^2} + \frac{1}{\tau^2 + C_{t-1}^2}}, \qquad C_t^2 = \frac{1}{\frac{1}{\sigma^2} + \frac{1}{\tau^2 + C_{t-1}^2}}.$$

These recursions are known as the Kalman filter in engineering.

Filtering: Sampling Importance Resampling

- **Example (a simple linear Gaussian system, cont):** The SIR algorithm for this problem takes the form

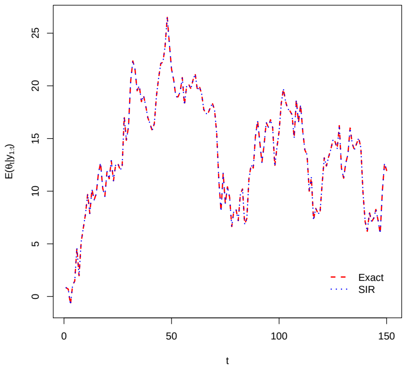
  1. For $b = 1, \ldots, B$ sample $\tilde{\theta}_t^{(b)} \sim \hat{p}\left(\theta_t \mid \mathbf{y}_{1:t}\right)$.
  2. For $b = 1, \ldots, B$ sample $\hat{\theta}_{t+1}^{(b)} \mid \tilde{\theta}_t^{(b)} \sim \mathsf{N}\left(\theta_{t+1} \mid \tilde{\theta}_t^{(b)}, \tau^2\right)$.
  3. Construct $\hat{p}\left(\theta_{t+1} \mid \mathbf{y}_{1:(t+1)}\right) = \sum_{b=1}^{B} \omega_{t+1}^{(b)} \delta_{\hat{\theta}_{t+1}^{(b)}}$ where

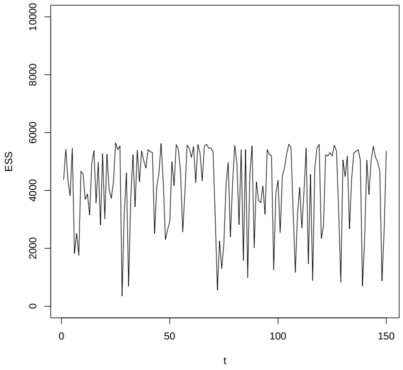
$$\omega_{t+1}^{(b)} = \frac{\mathsf{N}\left(y_{t+1} \mid \hat{\theta}_{t+1}^{(b)}, \sigma^2\right)}{\sum_{\ell=1}^{B} \mathsf{N}\left(y_{t+1} \mid \hat{\theta}_{t+1}^{(\ell)}, \sigma^2\right)}.$$

  The file `GaussianSIR.R` implements this SIR algorithm using NIMBLE and compares it against the closed-form solution.

Filtering: Sampling Importance Resampling

- **Example (a simple linear Gaussian system, cont):** Comparing estimates based on the closed-form recursions and the SIR filter



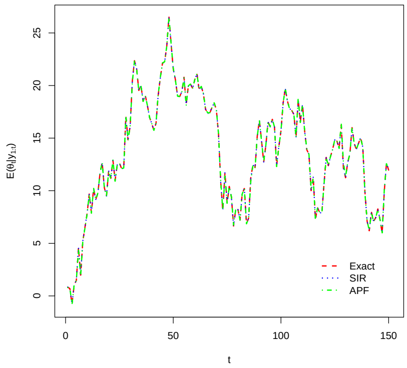


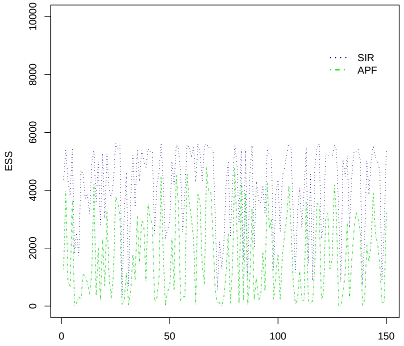
Filtering: Auxiliary particle filters

- APFs reverse the order of resampling and reweighting:
  - Given a particle representation $\hat{p}\left(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t}\right) = \sum_{b=1}^{B} \omega_t^{(b)} \delta_{\hat{\boldsymbol{\theta}}_t^{(b)}}$
    1. For $b = 1, \ldots, B$ construct a first-stage weight
    $$\varpi_t^{(b)} = \frac{\omega_t^{(b)} p\left(\mathbf{y}_{t+1} \mid \boldsymbol{\mu}_t^{(b)}\right)}{\sum_{\ell=1}^{B} \omega_t^{(\ell)} p\left(\mathbf{y}_{t+1} \mid \boldsymbol{\mu}_t^{(\ell)}\right)}.$$
    where $\boldsymbol{\mu}_t^{(b)} = \mathrm{E}\left(\boldsymbol{\theta}_{t+1} \mid \hat{\boldsymbol{\theta}}_t^{(b)}\right)$.
    2. For $b = 1, \ldots, B$ sample $\xi_t^{(b)} \sim \sum_{\ell=1}^{B} \varpi_t^{(\ell)} \delta_{\ell}$.
    3. For $b = 1, \ldots, B$ sample $\hat{\boldsymbol{\theta}}_{t+1}^{(b)} \mid \hat{\boldsymbol{\theta}}_t^{(\xi_t^{(b)})} \sim p\left(\boldsymbol{\theta}_{t+1} \mid \hat{\boldsymbol{\theta}}_t^{(\xi_t^{(b)})}\right)$.
    4. Construct $\hat{p}\left(\boldsymbol{\theta}_{t+1} \mid \mathbf{y}_{1:(t+1)}\right) = \sum_{b=1}^{B} \omega_{t+1}^{(b)} \delta_{\hat{\boldsymbol{\theta}}_{t+1}^{(b)}}$ where
    $$\omega_{t+1}^{(b)} \propto \frac{p\left(\mathbf{y}_{t+1} \mid \hat{\boldsymbol{\theta}}_{t+1}^{(\xi_t^{(b)})}\right)}{p\left(\mathbf{y}_{t+1} \mid \boldsymbol{\mu}_t^{(\xi_t^{(b)})}\right)}.$$

Filtering: Auxiliary particle filters

- **Example (a simple linear Gaussian system, cont):** The APF for the linear Gaussian system we described in our previous example (see file `GaussianAPF.R`) gives a lower average ESS (1687 vs. SIR's 4117).





Smoothing

- Particle smoothing refers to the process of generating samples from $p(\boldsymbol{\theta}_{1:T} \mid \mathbf{y}_{1:T})$ using the particles associated with the approximations $\hat{p}(\boldsymbol{\theta}_t \mid \mathbf{y}_{1:T}) = \sum_{b=1}^{B} \omega_t^{(b)} \delta_{\hat{\boldsymbol{\theta}}_t^{(b)}}$.
- The simplest smoothing algorithm was proposed by Godsill et al. (2004). It generates a sample trajectory $\left(\bar{\theta}^{(1)}, \ldots, \bar{\theta}^{(B)}\right)$ through the following steps:
  1. Sample $\bar{\theta}_{(T)} \sim \hat{p}(\boldsymbol{\theta}_T \mid \mathbf{y}_{1:T})$.
  2. For $t = T-1, T-2, \ldots, 1$, repeat:
     1. Calculate $\varpi_{t|t+1}^{(b)} \propto \omega_t^{(b)} p\left(\bar{\boldsymbol{\theta}}_{t+1} \mid \hat{\boldsymbol{\theta}}_t^{(b)}\right)$ for each $b = 1, \ldots, B$.
     2. Sample $\bar{\boldsymbol{\theta}}_t \sim \sum_{b=1}^{B} \varpi_{t|t+1}^{(b)} \delta_{\hat{\boldsymbol{\theta}}_t^{(b)}}$.

Some important topics not treated here

- We just scratched the surface of SMC algorithms.
- A particularly interesting area is algorithms that allow for joint estimation of dynamic and static parameters.
- SMC algorithms are very popular in the engineering literature.
- For comprehensive review see Cappé et al. (2007) (my favorite) and Doucet & Johansen (2009).

Variational algorithms

- The idea behind variational algorithms is to replace the true (intractable) posterior $p(\boldsymbol{\theta} \mid \mathbf{y})$ by a (tractable) approximation $q_{\boldsymbol{\eta}}$ "close" to $p(\boldsymbol{\theta} \mid \mathbf{y})$ and dependent on a set of tunable parameters $\boldsymbol{\eta}$.
- Different ways to meassure how "close" $p(\boldsymbol{\theta} \mid \mathbf{y})$ and $q_{\boldsymbol{\eta}}(\boldsymbol{\theta})$ are. KullbackLeibler divergence is popular:
    - $K(q||p) = \mathrm{E}_q \left[\log \left\{ \frac{q_{\boldsymbol{\eta}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{y})} \right] \right\} \Rightarrow$ Variational.
    - $K(p||q) = \mathrm{E}_p \left[\log \left\{ \frac{p(\boldsymbol{\theta}|\mathbf{y})}{q_{\boldsymbol{\eta}}(\boldsymbol{\theta})} \right] \right\} \Rightarrow$ Expectation-propagation.
- Approximation is given by $q_{\hat{\boldsymbol{\eta}}}(\boldsymbol{\theta})$ where

$$\hat{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{\eta}} \mathrm{E}_q \left\{ \log \left[\frac{q_{\boldsymbol{\eta}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta} \mid \mathbf{y})} \right] \right\}$$

Variational algorithms

- In principle, we have a lot of freedom in choosing the functional form of $q_{\boldsymbol{\eta}}$. However, in practice we are limited by the need to have a tractable approximation, to compute the expectations $\mathrm{E}_q\left\{\log\left[q_{\boldsymbol{\eta}}(\boldsymbol{\theta})\right]\right\}$ and $\mathrm{E}_q\left\{\log\left[p_{\boldsymbol{\eta}}(\boldsymbol{\theta} \mid \mathbf{y})\right]\right\}$ and solve the associated maximization problem.
- So-called *mean field* variational algorithms, which use a fully factorized q,

$$q_{\boldsymbol{\eta}}(\boldsymbol{\theta}) = \prod_{k=1}^{p} q_{k,\boldsymbol{\eta}_k}(\theta_k)$$

  are extremely popular because of their tractability.
- As with the Gibbs sampler, we might not fully factorize $q_{\boldsymbol{\eta}}(\boldsymbol{\theta})$ and instead work with blocks of parameters.

Variational algorithms in exponential families

- Within the exponential family, variational algorithms are quite straightforward and look like Gibbs samplers. Let the full conditional distribution for θ_k be of the form

$$p(\theta_k \mid \boldsymbol{\theta}_{-k}, \mathbf{y}) \propto \exp\left\{ \sum_{l=1}^{L} g_{k,l}\left(\boldsymbol{\theta}_{-k}, \mathbf{y}\right) t_l(\theta_k) - h(\theta_k) \right\}$$

where $\boldsymbol{\theta}_{-k} = (\theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_p)$ and $g_{k,1}\left(\boldsymbol{\theta}_{-k}, \mathbf{y}\right), \ldots, g_{k,L}\left(\boldsymbol{\theta}_{-k}, \mathbf{y}\right)$ are conditional sufficient statistics, and let the variational approximation be

$$q_{k,\eta_k}(\theta_k) \propto \exp\left\{ \sum_{l=1}^{L} \eta_{k,l} t_l(\theta_k) - a(\theta_k) \right\},$$

then, $\hat{\eta}_{k,l} = \mathrm{E}_{q_{-k}}\left\{ g_{k,l}\left(\boldsymbol{\theta}_{-k}, \mathbf{y}\right) \right\}$. By iterating updates of this form we can develop a very fast computational algorithm!

Variational algorithms in exponential families

- **Example (Gaussian distributions with unknown mean and variance):** Assume that data $y_1, \dots, y_n$ are independent and identically distributed with $y_i \mid \theta, \sigma^2 \sim \mathrm{N}(\theta, \sigma^2)$ and that we use independent priors $\theta \sim \mathrm{N}(\mu, \tau^2)$ and $\sigma^2 \sim \mathrm{IGam}(a, c)$. We derived a Gibbs sampler for this model earlier in this course:
  1. Initialize $\tilde{\theta}^{(0)}$ and $\tilde{\sigma}^{2(0)}$.
  2. For $b = 1, \dots, B$ repeat
     1. Sample $\tilde{\theta}^{(b)} \sim \mathrm{N}\left(\frac{\frac{n\bar{y}}{\tilde{\sigma}^{2(b-1)}} + \frac{\mu}{\tau^2}}{\frac{n}{\tilde{\sigma}^{2(b-1)}} + \frac{1}{\tau^2}}, \frac{1}{\frac{n}{\tilde{\sigma}^{2(b-1)}} + \frac{1}{\tau^2}} \right)$.
     2. Sample $\tilde{\sigma}^{2(b)} \sim \mathrm{IGam}\left(a + \frac{n}{2}, c + \frac{1}{2} \sum_{i=1}^{n} \{y_i - \tilde{\theta}^{(b)}\}^2 \right)$.

Variational algorithms in exponential families

- **Example (Gaussian distributions with unknown mean and variance):** Consider now a variational algorithm for exactly the same problem. A mean field variational approximation using tractable distributions that belong to the same families as the priors would be

$$q(\theta, \sigma^2) = q_{(\eta_{\theta,1}, \eta_{\theta,2})}(\theta) q_{(\eta_{\sigma^2,1}, \eta_{\sigma^2,2})}(\sigma^2),$$

where

$$q_{(\eta_{\theta,1}, \eta_{\theta,2})}(\theta) = \mathsf{N}(\theta \mid \eta_{\theta,1}, \eta_{\theta,2})$$

and

$$q_{(\eta_{\sigma^2,1}, \eta_{\sigma^2,2})}(\sigma^2) = \mathsf{IGam}(\sigma^2 \mid \eta_{\sigma^2,1}, \eta_{\sigma^2,2}).$$

Variational algorithms in exponential families

- **Example (Gaussian distributions with unknown mean and variance):** Under these choices, the variational parameters can be iteratively updated by letting

$$\eta_{\theta,1}^{(b)} = \frac{n\bar{y}\frac{\eta_{\sigma^2,1}^{(b-1)}}{\eta_{\sigma^2,2}^{(b-1)}} + \frac{\mu}{\tau^2}}{n\frac{\eta_{\sigma^2,1}^{(b-1)}}{\eta_{\sigma^2,2}^{(b-1)}} + \frac{1}{\tau^2}} \qquad \eta_{\theta,2}^{(b)} = \frac{1}{n\frac{\eta_{\sigma^2,1}^{(b-1)}}{\eta_{\sigma^2,2}^{(b-1)}} + \frac{1}{\tau^2}}$$

  and

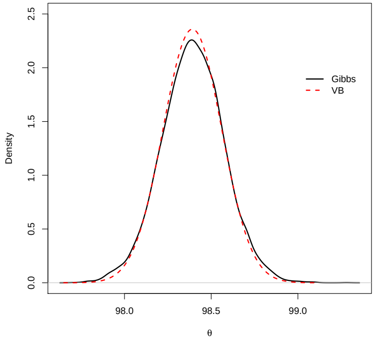
$$\eta_{\sigma^2,1}^{(b)} = a + \frac{n}{2}$$

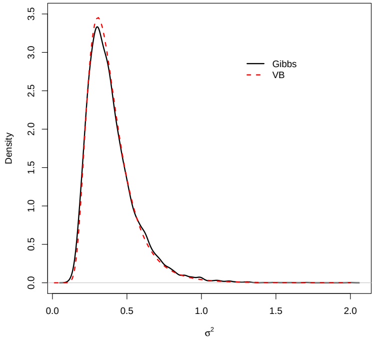
$$\eta_{\sigma^2,2}^{(b)} = c + \frac{1}{2}\sum_{i=1}^{n}\left\{y_i^2 - 2y_i\eta_{\theta,1}^{(b)} + \eta_{\theta,2}^{(b)} + \eta_{\theta,1}^{2(b)}\right\}$$

  Compare against the Gibbs sampler updates!

Variational algorithms in exponential families

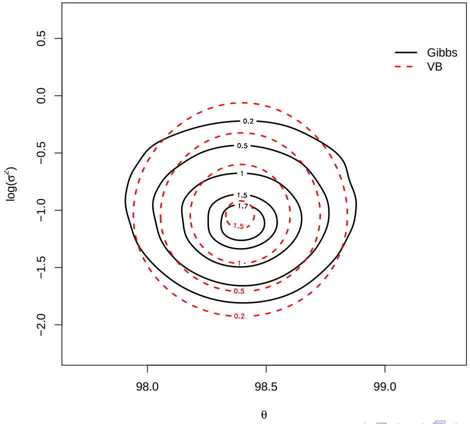
- **Example (Gaussian distributions with unknown mean and variance):** The file `simpleVB.R` implements this algorithm. Below is a comparison of the marginal posterior distributions.





Variational algorithms

- **Example (Gaussian distributions with unknown mean and variance):** Comparing the joint distributions.



Variational algorithms in exponential families

- **Example (Mixture models):** Recall the two-component Gaussian mixture model where

$$y_i \mid \xi_i, \theta_1, \sigma_1^2, \theta_2, \sigma_2^2 \sim \mathsf{N}(y_i \mid \theta_{\xi_i}, \sigma_{\xi_i}^2)$$

with $\Pr(\xi_i = 1 \mid \omega) = \omega = 1 - \Pr(\xi_i = 2 \mid \omega)$, $\omega \sim \mathsf{beta}(1,1)$, $\theta_i \sim \mathsf{N}(\mu, \tau^2)$ and $\sigma_i^2 \sim \mathsf{IGam}(a, c)$.

For the purpose of deriving the variational algorithm, the best way to write the joint distribution of data and parameters is

$$\left[\prod_{i=1}^{n}\prod_{k=1}^{2}\left\{p(y_i \mid \theta_k, \sigma_k^2\right\}^{\mathbb{I}_{(\xi_i = k)}}\right]$$
$$\left[\omega^{1+\sum_{i=1}^{n}\mathbb{I}_{(\xi_i=1)}}(1-\omega)^{1+\sum_{i=1}^{n}\mathbb{I}_{(\xi_i=2)}}\right]$$
$$\left[\prod_{k=1}^{2}\mathsf{N}(\theta_k \mid \mu, \tau^2)\mathsf{IGam}(\sigma_k^2 \mid a, c)\right]$$

Variational algorithms in exponential families

- **Example (Mixture models, cont):** The variational approximation in this case is given by

$$q_{\gamma_1,\gamma_2}(\omega) \prod_{i=1}^{n} q_{\varpi_i}(\xi_i) \prod_{k=1}^{2} q_{\eta_{k,1},\eta_{k,2}}(\theta_k) \prod_{k=1}^{2} q_{\nu_{k,1},\nu_{k,2}}\left(\sigma_k^2\right),$$

where

$$\begin{aligned} q_{\gamma_1,\gamma_2}(\omega) &= \text{beta}(\omega \mid \gamma_1, \gamma_2) \\ q_{\varpi_i}(\xi_i = 1) &= 1 - q_{\varpi_i}(\xi_i = 2) = \varpi_i, \\ q_{\eta_{k,1},\eta_{k,2}}(\theta_k) &= \text{N}(\theta_k \mid \eta_{k,1}, \eta_{k,2}), \\ q_{\nu_{k,1},\nu_{k,2}}\left(\sigma_k^2\right) &= \text{IGam}\left(\sigma_k^2 \mid \nu_{k,1}, \nu_{k,2}\right). \end{aligned}$$

Variational algorithms in exponential families

- **Example (Mixture models, cont):** The corresponding variational updates become
  - For ω,

$$\gamma_1^{(b)} = 1 + \sum_{i=1}^{n} \varpi_i^{(b-1)}, \qquad \gamma_2^{(b)} = 1 + n - \sum_{i=1}^{n} \varpi_i^{(b-1)}.$$

  - For θ_1,

$$\eta_{1,1}^{(b)} = \frac{\frac{\nu_{1,1}^{(b-1)}}{\nu_{1,2}^{(b-1)}} \sum_{i=1}^{n} y_i \varpi_i^{(b-1)} + \frac{\mu}{\tau^2}}{\frac{\nu_{1,1}^{(b-1)}}{\nu_{1,2}^{(b-1)}} \sum_{i=1}^{n} \varpi_i^{(b-1)} + \frac{1}{\tau^2}}, \quad \eta_{1,2}^{(b)} = \frac{1}{\frac{\nu_{1,1}^{(b-1)}}{\nu_{1,2}^{(b-1)}} \sum_{i=1}^{n} \varpi_i^{(b-1)} + \frac{1}{\tau^2}}.$$

  The formulas for θ_2 are analogous but replace ϖ_i with $1 - \varpi_i$.

Variational algorithms in exponential families

- **Example (Mixture models, cont):**
  - For σ_1^2,

$$\nu_{1,1} = a + \frac{1}{2}\sum_{i=1}^{n} \varpi_i^{(b-1)}$$

$$\nu_{1,2} = c + \frac{1}{2}\sum_{i=1}^{n} \varpi_i^{(b-1)} \left\{ y_i^2 - 2y_i\eta_{1,1}^{(b)} + \eta_{1,2}^{(b)} + \eta_{1,1}^{2(b)} \right\}$$

    The formulas for σ_2^2 are analogous but replace ϖ_i with $1 - \varpi_i$.
  - For ξ_i,

$$\varpi_i^{(b)} \propto \exp\left\{ \Psi\left(\gamma_1^{(b)}\right) - \Psi\left(\gamma_1^{(b)} + \gamma_2^{(b)}\right) + \frac{1}{2}\left[\Psi\left(\nu_{k,1}^{(b)}\right) - \log \nu_{k,2}^{(b)}\right] \right.$$
$$\left. - \frac{\nu_{k,1}^{(b)}}{2\nu_{k,2}^{(b)}} \left[y_i^2 - 2y_i\eta_{k,1}^{(b)} + \eta_{k,2}^{(b)} + \eta_{k,1}^{2(b)} \right] \right\}$$

    where Ψ denotes the digamma function.

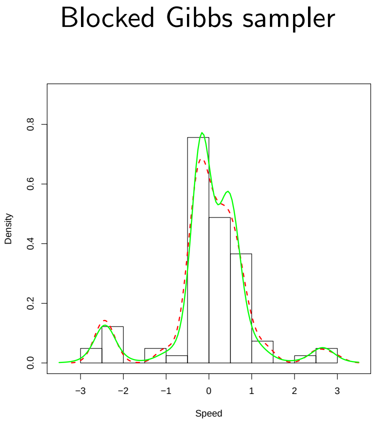
Variational algorithms in exponential families

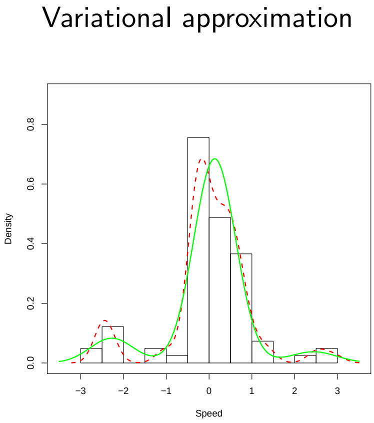
- **Example (Mixture models, cont):** The algorithm can be naturally extended to mixtures with a larger number of components, and even to nonparametric mixtures.

  The following two slides show results for a slightly more convoluted location scale mixture under a finite Dirichlet process mixture model Blei & Jordan (2006) for the `galaxy` dataset Roeder (1990) (available in the `DPpackage` library for `R`). They illustrate some of the disadvantages of variational approximation when compared to "equivalent" Gibbs sampling algorithms.

Gibbs vs. variational algorithms for location mixtures of normals

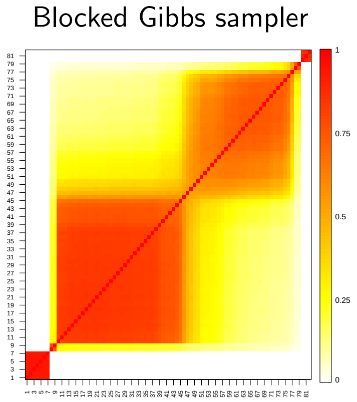
Density estimates

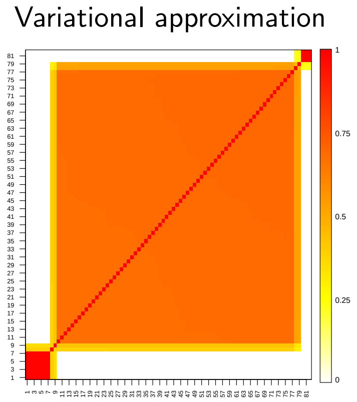




Gibbs vs. variational algorithms for location mixtures of normals

Pairwise clustering probabilities





Some concluding remarks about variational algorithms

- The minimization of $K(q||p)$ can also be justified as maximizing the bound on the log marginal likelihood:
$$\log p(\mathbf{y}) \geq \mathrm{E}_q \left\{\log p(\boldsymbol{\theta}, \mathbf{y})\right\} - \mathrm{E}_q \left\{\log q_{\boldsymbol{\eta}}(\boldsymbol{\theta})\right\}$$
The gap in the bound is precisely the divergence between $q_{\boldsymbol{\eta}}$ and the true posterior. This value of this gap is usually monitored to assess convergence of the algorithm.
- Variational Bayes works well at approximating the marginal posteriors of the parameters on which it is derived, but typically fails for joint distributions/transformations involving multiple parameters.
- For complex models variational algorithms tend to get stuck in local modes. It is very important to perform multiple runs from *overdispersed* initial values and verify that the algorithm converges to the same set of values.

The Laplace approximation

- An alternative to variational approximations are Laplace approximations.
- The Laplace approximation is a simple yet powerful asymptotic expansion of the posterior distribution that goes back to Laplace in 1774!
- There are different variants of the technique, but the main argument is similar in all cases: Start by assuming that $y_1, \ldots, y_n$ are independent and identically distributed with $y_i \mid \boldsymbol{\theta} \sim p(y_i \mid \boldsymbol{\theta})$ so that the posterior takes the form

$$p(\boldsymbol{\theta} \mid \mathbf{y}) \propto \exp\left\{-n h_n(\boldsymbol{\theta})\right\},$$

where $h_n(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i \mid \boldsymbol{\theta}) - \frac{p(\boldsymbol{\theta})}{n}$.

The Laplace approximation

- Now, do a second order expansion of $h_n(\boldsymbol{\theta})$ around a point $\boldsymbol{\theta}_0$ and write

$$\begin{aligned} h_n(\boldsymbol{\theta}) = h_n(\boldsymbol{\theta}_0) &+ \left[\nabla h_n\left(\boldsymbol{\theta}_0\right)\right]^T\left[\boldsymbol{\theta} - \boldsymbol{\theta}_0\right] \\ &+ \frac{1}{2}\left[\boldsymbol{\theta} - \boldsymbol{\theta}_0\right]^T\left[\nabla\nabla h_n\left(\boldsymbol{\theta}_0\right)\right]\left[\boldsymbol{\theta} - \boldsymbol{\theta}_0\right] + R_n(\boldsymbol{\theta}, \boldsymbol{\theta}_0), \end{aligned}$$

  where $\nabla h_n\left(\boldsymbol{\theta}\right)$ is the gradient of h_n and $\nabla\nabla h_n\left(\boldsymbol{\theta}\right)$ is the Hessian matrix of h.
- It is not difficult to show that, under standard regularity conditions, $\lim_{n\to\infty} R_n(\boldsymbol{\theta}, \boldsymbol{\theta}_0) = 0$.

The Laplace approximation

- The previous argument suggests the approximation

$$p(\boldsymbol{\theta} \mid \mathbf{y}) \dot{\propto} \exp\left\{-n\left[\nabla h_n(\boldsymbol{\theta}_0)\right]^T[\boldsymbol{\theta}-\boldsymbol{\theta}_0] \right.$$
$$\left. -\frac{n}{2}[\boldsymbol{\theta}-\boldsymbol{\theta}_0]^T\left[\nabla\nabla h_n(\boldsymbol{\theta}_0)\right][\boldsymbol{\theta}-\boldsymbol{\theta}_0]\right\}$$

  which corresponds to a Gaussian kernel!

- Note that the contribution of the prior drops out faster than the contribution from the likelihood. Therefore, alternatively,

$$p(\boldsymbol{\theta} \mid \mathbf{y}) \dot{\propto} \exp\left\{\left[\nabla \log p(\mathbf{y} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}\right]^T[\boldsymbol{\theta}-\boldsymbol{\theta}_0] \right.$$
$$\left. -\frac{1}{2}[\boldsymbol{\theta}-\boldsymbol{\theta}_0]^T\left[-\nabla\nabla \log p(\mathbf{y} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}\right][\boldsymbol{\theta}-\boldsymbol{\theta}_0]\right\}.$$

The Laplace approximation

- Typically $\boldsymbol{\theta}_0$ is taken to be either the MLE of $\boldsymbol{\theta}$ or the maximum a posteriori, leading to simplifications.
  - For example, discarding the contribution of the likelihood and letting $\boldsymbol{\theta}_0 = \hat{\boldsymbol{\theta}}$ we have

$$p(\boldsymbol{\theta} \mid \mathbf{y}) \dot{\propto} \exp\left\{ -\frac{1}{2} \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right]^T \left[-\nabla\nabla \log p\left(\mathbf{y} \mid \boldsymbol{\theta}\right)\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \right] \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right] \right\}$$

  where the matrix $-\nabla\nabla \log p\left(\mathbf{y} \mid \boldsymbol{\theta}\right)\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$ is called the *observed* information matrix. That is

$$\boldsymbol{\theta} \mid \mathbf{y} \dot{\sim} \mathsf{N}\left(\hat{\boldsymbol{\theta}}, \left[-\nabla\nabla \log p\left(\mathbf{y} \mid \boldsymbol{\theta}\right)\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \right]^{-1}\right)$$

- This last version of the approximation is the basis for "Bayesian CLT", as well as for the derivation of the BIC.

The Laplace approximation

- Although technically the result is valid for any parameterization, maing appropriate transformations (e.g., logarithm, logit) might improve the accuracy.
- The Laplace approximation can be very accurate for large n, but in complicated hierarchical models with a large number of hyperparameters the approximation can be very poor!
- When the posterior moments are available in closed form, it might simply be better to use the posterior mean and variance as the moments for the normal approximation.
- Even if they are not very accurate, Laplace approximations are sometimes used to guide the construction of proposal distributions in some of the Monte Carlo schemes discussed before.

The Laplace approximation

- **Example (Poisson-Gamma models):** Consider data $y_1, \ldots, y_n$ where $y_i \sim \text{Poi}(\lambda)$ and $\lambda \sim \text{Gam}(a, b)$ The posterior distribution (which is known to be a $\text{Gam}\left(a + \sum_{i=1}^n y_i, n + b\right)$ distribution) can be written as $p(\lambda \mid \mathbf{y}) \propto \{-nh_n(\lambda)\}$ where

$$h_n(\lambda) = -\lambda \frac{n+b}{n} + \frac{a - 1 + \sum_{i=1}^n y_i}{n} \log \lambda.$$

It is easy to se that the posterior mode in this case is $\tilde{\lambda} = \frac{a-1+\sum_{i=1}^n y_i}{n+b}$. Also $-\frac{\partial^2 h_n}{\partial \lambda^2} = \frac{a-1+\sum_{i=1}^n y_i}{n} \frac{1}{\lambda^2}$, which leads

$$\lambda \mid \mathbf{y} \mathrel{\dot{\sim}} \text{N}\left(\frac{a - 1 + \sum_{i=1}^n y_i}{n+b}, \frac{\{a - 1 + \sum_{i=1}^n y_i\}}{(n+b)^2} \right)$$

The Laplace approximation

- **Example (Poisson-Gamma models, cont):** Alternatively, if we discard the prior and center the approximation on the MLE we get the familiar expression

$$\lambda \mid \mathbf{y} \mathrel{\dot\sim} \mathrm{N}\left(\frac{1}{n}\sum_{i=1}^{n} y_i, \sum_{i=1}^{n} \frac{y_i}{n^2}\right).$$

(Note that this is equivalent to setting $a = 1$ and $b = 0$ in the first approximation).

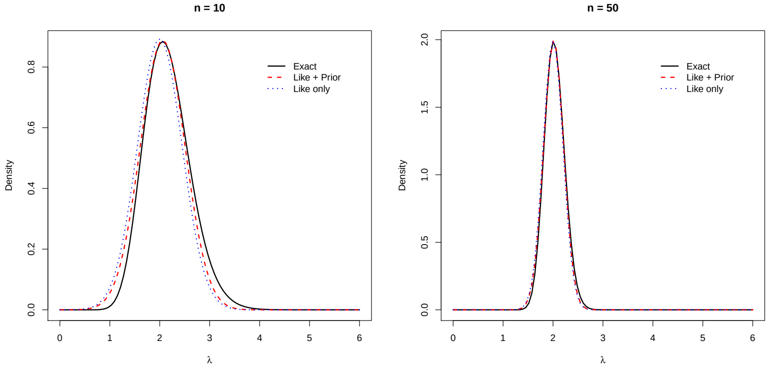
If we are interested in approximating the posterior mean then we have

Exact	$\frac{a+\sum_{i=1}^{n} y_i}{b+n}$
Like + Prior	$\frac{a-1+\sum_{i=1}^{n} y_i}{b+n}$
Like only	$\frac{\sum_{i=1}^{n} y_i}{n}$

If a is large and n small, the differences can be substantial!

The Laplace approximation

- **Example (Poisson-Gamma models):** The following graphs show a comparison of the two approximations against the true posterior for $a = 2$, $b = 1/5$ and $\bar{y} = 2$.



The Laplace approximation

- **Example (Poisson-Gamma models):** The following table compares the exact posterior probability of the interval (l, u) against these two approximation for different intervals and sample sizes.

(l, u)	n	Exact	Like + Prior	Like only
$(1.5, 2.8)$		0.849	0.844	0.831
$(1.8, 2.3)$	10	0.420	0.422	0.421
$(2.5, \infty)$		0.218	0.163	0.132
$(1.5, 2.8)$		0.998	0.995	0.994
$(1.8, 2.3)$	50	0.783	0.780	0.775
$(2.5, \infty)$		0.014	0.007	0.006

Integrated Nested Laplace Approximations

- INLA is designed for hierarchical models:

$$y_i \mid \theta_i, \boldsymbol{\eta}_1 \sim p(y_i \mid \theta_i, \boldsymbol{\eta}_1) \qquad \boldsymbol{\theta} \mid \boldsymbol{\eta}_2 \sim \mathsf{N}\left(\mathbf{0}, \mathbf{Q}(\boldsymbol{\eta}_2)\right),$$

  where $\boldsymbol{\eta} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2) \sim p(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ and $m = \dim(\boldsymbol{\eta}_1) + \dim(\boldsymbol{\eta}_2)$ is small (usually ≤ 6).
- Particularly useful if $\mathbf{Q}(\boldsymbol{\eta}_2)$ is sparse.
- Goal is to provide approximations to

$$p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y}) = \int p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y}) p(\boldsymbol{\eta} \mid \mathbf{y}) \mathrm{d}\boldsymbol{\eta}$$

  and

$$p(\eta_j \mid \mathbf{y}) = \int p(\eta \mid \mathbf{y}) \mathrm{d}\boldsymbol{\eta}_{-j}.$$

- Uses Laplace approximations twice (hence the name).

Approximating $p(\boldsymbol{\eta} \mid \mathbf{y})$

- Start by approximating $p(\boldsymbol{\eta} \mid \mathbf{y})$ by

$$\tilde{p}(\boldsymbol{\eta} \mid \mathbf{y}) \propto \left. \frac{p(\boldsymbol{\theta}, \boldsymbol{\eta}, \mathbf{y})}{\tilde{p}_G(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})} \right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}(\boldsymbol{\eta})}$$

where $\tilde{p}_G(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})$ is a Gaussian approximation to the full conditional of $\boldsymbol{\theta}$

$$p(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y}) \propto \exp\left\{ -\frac{1}{2}\boldsymbol{\theta}^T \mathbf{Q}\left(\eta_2\right)\boldsymbol{\theta} + \sum_i \log\left\{p(y_i \mid \theta_i, \boldsymbol{\eta}_1)\right\} \right\}$$

and $\hat{\boldsymbol{\theta}}(\boldsymbol{\eta}) = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})$. This is just the Laplace approximation around the posterior mode.

Approximating $p(\boldsymbol{\eta} \mid \mathbf{y})$

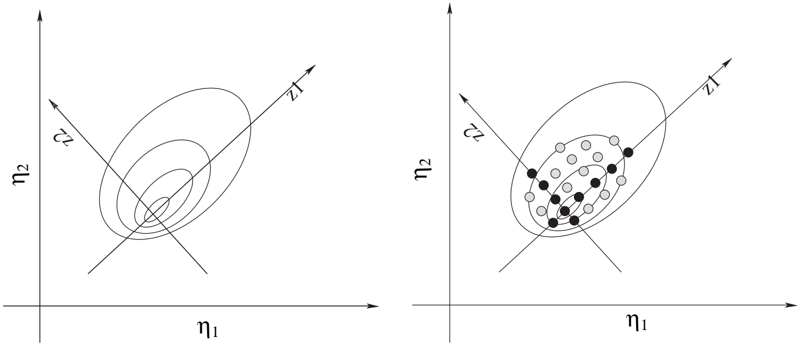
- Generally speaking $\tilde{p}(\boldsymbol{\eta} \mid \mathbf{y})$ will not be tractable even if $\tilde{p}_G(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})$ is. Hence, this approximation by itself is not too helpful for computing an approximation to $p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$.
- We go one step forward and approximate the approximation using a discrete set of points,

  $$\tilde{p}^*(\boldsymbol{\eta} \mid \mathbf{y}) = \sum_{i_1 \in \mathcal{I}_1} \cdots \sum_{i_m \in \mathcal{I}_m} w_{i_1,\cdots,i_m} \delta_{(\tilde{\eta}_{i_1},\ldots,\tilde{\eta}_{i_m})}(\boldsymbol{\eta})$$

  where the points $(\tilde{\eta}_{i_1}, \ldots, \tilde{\eta}_{i_m})$ for $(i_1, \ldots, i_m) \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_m$ form a regular grid and

  $$w_{i_1,\cdots,i_m} \propto \tilde{p}^*(\tilde{\eta}_{i_1}, \ldots, \tilde{\eta}_{i_m} \mid \mathbf{y})$$
- To select the grid points we use a second Laplace expansion (in this case of $\tilde{p}(\boldsymbol{\eta} \mid \mathbf{y})$). See next slide.

Approximating $p(\boldsymbol{\eta} \mid \mathbf{y})$



From Rue et al. (2009).

Approximating $p(\eta_i \mid \mathbf{y})$

- It is natural to approximate $p(\eta_j \mid \mathbf{y})$ by

$$\tilde{p}(\eta_j \mid \mathbf{y}) = \int \tilde{p}(\eta \mid \mathbf{y}) \mathrm{d}\boldsymbol{\eta}_{-j}.$$

- The integral can be computed using standard numerical integration techniques.
- However, it is much more efficient to construct the numerical integrator by reusing the grid of values used in the definition $\tilde{p}^*(\boldsymbol{\eta} \mid \mathbf{y})$ (which is aligned with the principal axes of the distribution) rather than a completely new grid aligned to the original axes.

Approximating $p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$

- If we had access to $p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$ in closed form, the approximation $\tilde{p}^*(\boldsymbol{\eta} \mid \mathbf{y})$ could be used to compute $\hat{p}(\theta_i \mid \mathbf{y})$ as

$$\begin{aligned}\hat{p}(\theta_i \mid \mathbf{y}) &= \int p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})\tilde{p}^*(\boldsymbol{\eta} \mid \mathbf{y})\mathrm{d}\boldsymbol{\eta} \\ &= \sum_{i_1 \in \mathcal{I}_1} \cdots \sum_{i_m \in \mathcal{I}_m} w_{i_1,\cdots,i_m} p(\theta_i \mid (\tilde{\eta}_{i_1}, \ldots, \tilde{\eta}_{i_m}), \mathbf{y})\end{aligned}$$

- However, in $p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$ is not available in closed form so we need to first approximate by $\hat{p}(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$. One option is to reuse the Laplace approximation $\tilde{p}_G(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})$ we already computed for getting $\hat{p}(\theta_i \mid \mathbf{y})$. Then

$$\hat{p}^*(\theta_i \mid \mathbf{y}) = \sum_{i_1 \in \mathcal{I}_1} \cdots \sum_{i_m \in \mathcal{I}_m} w_{i_1,\cdots,i_m} \tilde{p}_G(\theta_i \mid (\tilde{\eta}_{i_1}, \ldots, \tilde{\eta}_{i_m}), \mathbf{y})$$

Approximating $p(\theta_i \mid \boldsymbol{\eta}, \mathbf{y})$

- Reusing $\tilde{p}_G(\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y})$ is expedient, but does not account for skewness and other features of the data.
- An alternative is to use another Laplace expansion

$$\tilde{p}_{LA}(\theta_i \mid \boldsymbol{\eta}, \mathbf{y}) \propto \left. \frac{p(\boldsymbol{\theta}, \boldsymbol{\eta}, \mathbf{y})}{\tilde{p}_{GG}(\boldsymbol{\theta}_{-i} \mid \theta_i, \boldsymbol{\eta}, \mathbf{y})} \right|_{\boldsymbol{\theta}_{-i} = \hat{\boldsymbol{\theta}}_{-i}(\theta_i, \boldsymbol{\eta})}$$

where $\tilde{p}_{GG}(\boldsymbol{\theta}_{-i} \mid \theta_i, \boldsymbol{\theta}, \mathbf{y})$ is a Gaussian approximation to $(\boldsymbol{\theta}_{-i} \mid \theta_i, \boldsymbol{\eta}, \mathbf{y})$ (which is different from the full conditional obtained from $\boldsymbol{\theta} \mid \boldsymbol{\eta}, \mathbf{y}$).

Laplace-based methods and NIMBLE

- Currently your software options are R-INLA and the related TMB software.
- NIMBLE is very close to having automatic differentiation, which is important for Laplace-based methods as they need gradients.
- At that point, one of our target algorithms will be INLA so we hope to see that in NIMBLE within a year.
- Our hope is that by providing in NIMBLE, users can:
  - use BUGS code to specify their model,
  - use distributions and model structures not provided in R-INLA, and
  - more readily compare results with other algorithms.

Learning objectives

- Understand the landscape of current computational methods for Bayesian inference;
- Understand the statistical and probabilistic principles behind the methods;
- Understand the various MCMC alternatives and how to assess MCMC performance;
- Be able to choose a method and software tool for a real-world problem; and
- Be able to implement Bayesian inference using NIMBLE and Stan.

Albert, J. & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association* **88**, 669–679.

Blei, D. M. & Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis* **1**, 121–144.

Brooks, S. P., Giudici, P. & Roberts, G. O. (2003). Efficient construction of reversible jump markov chain monte carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **65**, 3–39.

Cappé, O., Godsill, S. J. & Moulines, E. (2007). An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE* **95**, 899–924.

Doucet, A. & Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering* **12**, 656–704.

Gelman, A. & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press New York, NY, USA.

Gelman, A., Roberts, G. & Gilks, W. (1996). Efficient metropolis jumping hules. In *Bayesian statistics*, Eds. J. M. Bernardo, J. O. Berger, A. P. Dawid & A. F. M. Smith, volume 5, pp. 599–608. Oxford University Press.

Gelman, A. & Rubin, D. (1992). Inferences from iterative simulation using multiple sequences. *Statistical Science* **7**, 457–472.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *Bayesian Statistics 4*, Eds. J. M. Bernardo, J. Berger, A. P. Dawid & A. F. M. Smith, pp. 169–193. Oxford: Oxford University Press.

Godsill, S. J., Doucet, A. & West, M. (2004). Monte carlo smoothing for nonlinear time series. *Journal of the American Statistical Association* **99**.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**.

Haario, H., Saksman, E. & Tamminen, J. (2001). An adaptive metropolis algorithm. *Bernoulli* pp. 223–242.

Holmes, C. C., Held, L. et al. (2006). Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis* **1**, 145–168.

MacEachern, S. N. & Berliner, L. M. (1994). Subsampling the Gibbs sampler. *The American Statistician* **48**, 188–190.

NEAL, R. M. (2011). *MCMC using Hamiltonian dynamics*, volume 2, pp. 113–162. Chapman and Hall/CRC.

ROBERT, C., CASELLA, G. et al. (2011). A short history of markov chain monte carlo: subjective recollections from incomplete data. *Statistical Science* **26**, 102–115.

ROBERTS, G. O., GELMAN, A., GILKS, W. R. et al. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability* **7**, 110–120.

ROBERTS, G. O. & ROSENTHAL, J. S. (2009). Examples of Adaptive MCMC. *Journal of Computational and Graphical Statistics* **18**, 349–367.

ROBERTS, G. O., ROSENTHAL, J. S. et al. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical science* **16**, 351–367.

ROEDER, K. (1990). Density estimation with confidence sets exemplified by superclusters and voids in the galaxies. *Journal of the American Statistical Association* **85**, 617–624.

RUE, H., MARTINO, S. & CHOPIN, N. (2009). Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)* **71**, 319–392.

YU, Y. & MENG, X.-L. (2011). To center or not to center: That is not the question – an ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC efficiency. *Journal of Computational and Graphical Statistics* **20**, 531–570.

Stan example: Cox regression

- **Example (Cox regression):** Recall that standard Cox regression specifies the hazard function as:

$$\lambda(t|z) = \lambda_0(t)\exp(z^\top\beta)$$

for an unspecified baseline hazard, $\lambda_0(t)$.
With right-censoring, one has the likelihood (for one patient):

$$(\lambda_0(w_i)\exp(z_i^\top\beta))^{v_i}\exp\left(-\exp(z_i^\top\beta)\int_0^{w_i}\lambda_0(u)du\right)$$

where w_i is the minimum of the failure time and the censoring time for the patient and v_i is 1 if the person is not censored. It turns out this likelihood can be written as a Poisson distribution when one assumes piecewise constant baseline hazard – we'll use this in the implementation.

Stan example: Bayesian Cox regression

- **Example (Cox regression):** A Bayesian treatment requires either a prior for $\lambda_0(t)$ or use of the Kalbfleisch (1978) result that Cox's partial likelihood can be used as an approximation to an actual likelihood, with a prior placed simply on the regression coefficients.
  Considering the fully Bayesian approach, a canonical solution has been to assume a piecewise constant hazard model with independent jumps at the failure times. The size of the jumps is centered on those in a parametric hazard model, $\lambda_0^*(t)$ (such as a Weibull) using a gamma distribution for conjugacy.

$$\lambda_0(t)dt \sim \text{Gamma}(c\lambda_0^*(t)dt, c)$$

  This construction is called a gamma process.

Stan example: Stan overview

Stan focuses on HMC, so its primary computation is of the full log posterior density.

- `model` block of code uses imperative syntax (order matters) to encode calculation of log posterior.
- BUGS-style distributional statements are allowed.
- Stan also has its own language for encoding computation, including full linear algebra.

Stan generates C++ from the model code to implement HMC for the model and then compiles it to an executable, similar to NIMBLE.

Note: Stan cannot handle discrete parameters (discrete data are fine), so any implementation would require marginalization over such parameters.

Stan example: Stan overview

Other blocks of Stan code for a model describe:

- `data`: input data and what NIMBLE calls constants
- `transformed data`: fixed hyperparameters and transformations of input data
- `parameters`: list of unknown parameters
- `generated quantities`: posterior functionals of parameters and/or data

Stan example: Stan code for Cox regression

Here's the core `model` block for Cox regression for the BUGS leukemia example.

```
model {
  beta ~ normal(0, 1000);
  for(j in 1:NT) {  ## unique times
    ## gamma process prior for baseline hazard, centered on exponential
    dL0[j] ~ gamma(r * (t[j + 1] - t[j]) * c, c);
    for(i in 1:N) { ## patients
      if (Y[i, j] != 0)
         ## Poisson representation of likelihood with piecewise
         ## baseline hazard; Y[i,j] is observation process:
         ## Y[i,j]=1 if patient i is observed and 0 if censored
         dN[i, j] ~ poisson(Y[i, j] * exp(beta * Z[i]) * dL0[j]);
    }
  }
}
```

See `hmc-cox.R` for full Stan implementation.