Bayesian modelling

Markov chain Monte Carlo

Léo Belzile

2023

HEC MONTRÉAL

Reminder: Metropolis–Hastings algorithm

Starting from an initial value $\boldsymbol{\theta}_0$:

1. draw a proposal value $\boldsymbol{\theta}_t^\star \sim q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{t-1})$.
2. Compute the acceptance ratio

$$R = \frac{p(\boldsymbol{\theta}_t^\star)}{p(\boldsymbol{\theta}_{t-1})} \frac{q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t^\star)}{q(\boldsymbol{\theta}_t^\star \mid \boldsymbol{\theta}_{t-1})}$$

3. With probability $\min\{R, 1\}$, accept the proposal and set $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t^\star$, otherwise set the value to the previous state, $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1}$.

Calculations

We compute the log of the acceptance ratio, $\ln R$, to avoid numerical overflow, with the log posterior difference

$$\ln\left\{\frac{p(\boldsymbol{\theta}_t^\star)}{p(\boldsymbol{\theta}_{t-1})}\right\} = \ell(\boldsymbol{\theta}_t^\star) + \ln p(\boldsymbol{\theta}_t^\star) - \ell(\boldsymbol{\theta}_{t-1}) - \ln p(\boldsymbol{\theta}_{t-1})$$

Compare the value of $\ln R$ (if less than zero) to $\log(U)$, where $U \sim \mathsf{U}(0,1)$.

What proposal?

The *independence* Metropolis–Hastings uses a global proposal q which does not depend on the current state (typically centered at the MAP)

This may be problematic with multimodal targets.

The Gaussian random walk takes $\boldsymbol{\theta}_t^\star = \boldsymbol{\theta}_{t-1} + \sigma_{\mathrm{p}} Z$, where $Z \sim \mathsf{Norm}(0, 1)$ and σ_{p} is the proposal standard deviation.

Random walks allow us to explore the space.

Burn in

We are guaranteed to reach stationarity with Metropolis–Hastings, but it may take a large number of iterations…

One should discard initial draws during a **burn in** or warmup period if the chain has not reached stationarity. Ideally, use good starting value to reduce waste.

We can also use the warmup period to adapt the variance of the proposal.

Goldilock principle and proposal variance

Mixing of the chain requires just the right variance (not too small nor too large).

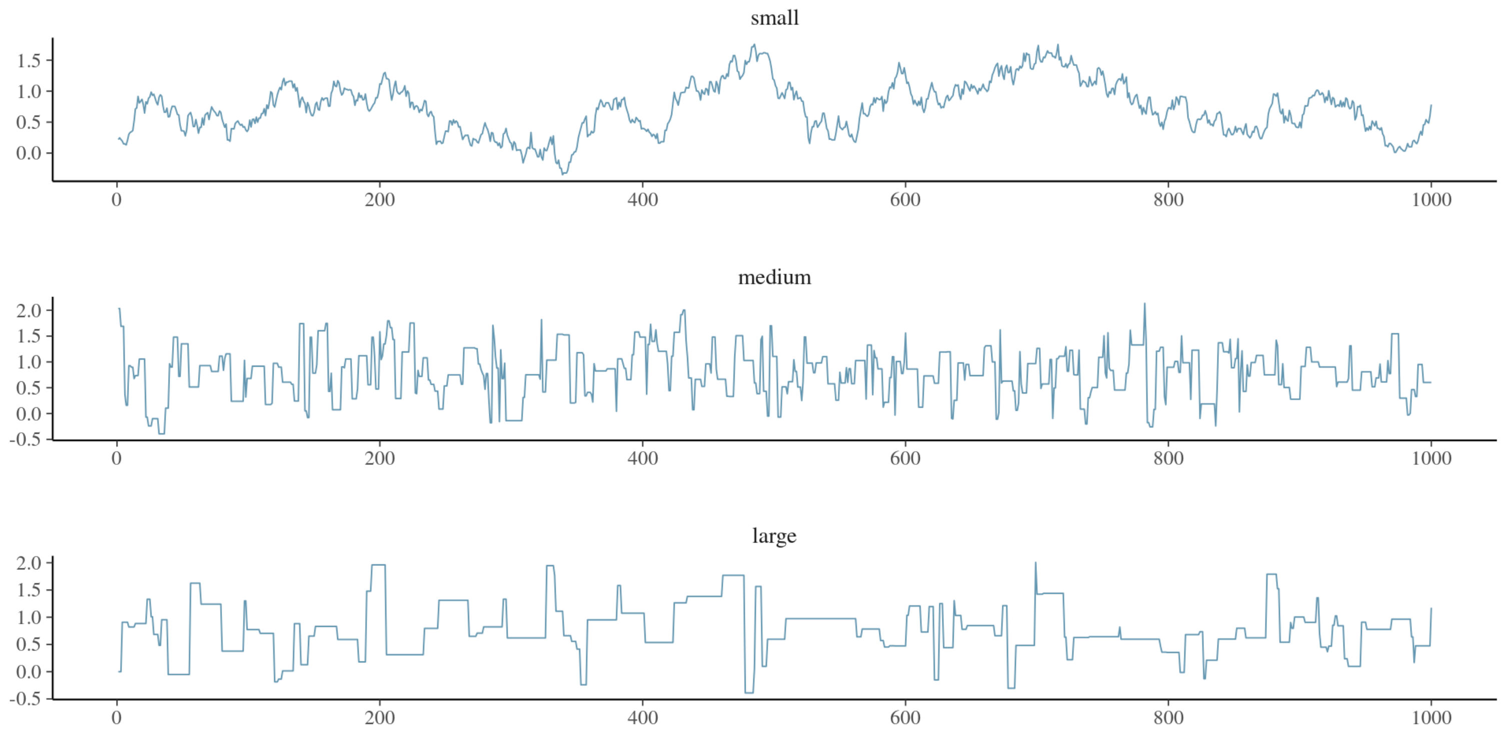


Figure 1: Example of traceplot with proposal variance that is too small (top), adequate (middle) and too large (bottom).

Correlograms for Goldilock

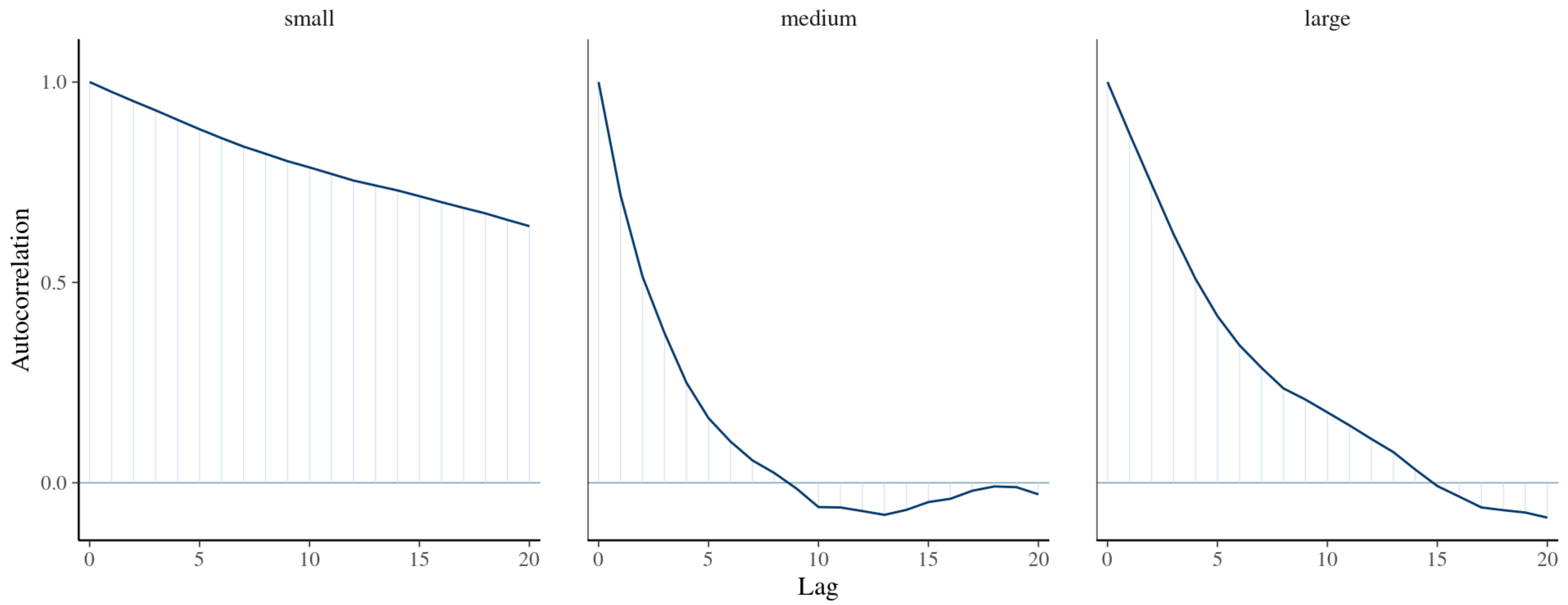


Figure 2: Correlogram for the three Markov chains.

Tuning Markov chain Monte Carlo

- Outside of starting values, the variance of the proposal has a huge impact on the asymptotic variance.
- We can adapt the variance during warmup by increasing/decreasing proposal variance (if acceptance rate is too large/small).
- We can check this via the acceptance rate (how many proposals are accepted).

Optimal acceptance rates

The following rules were derived for Gaussian targets under idealized situations.

- In 1D, rule of thumb is an acceptance rate of 0.44 is optimal, and this ratio decreases to 0.234 when $D \geq 2$ (Sherlock, 2013) for random walk Metropolis–Hastings.
- Proposals for D-variate update should have proposal variance of roughly $(2.38^2/d) \times \boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is the posterior variance.
- For MALA (see later), we get 0.574 rather than 0.234

Block update or one parameter at a time?

As with any accept-reject, proposals become inefficient when the dimension D increase.

This is the **curse of dimensionality**.

Updating parameters in turn

- increases acceptance rate (with clever proposals),
- but also leads to more autocorrelation between parameters

Solutions for strongly correlated coefficients

- Reparametrize the model to decorrelate variables (orthogonal parametrization).
- Block updates: draw correlated parameters together
  - using the chain history to learn the correlation, if necessary

Parameter transformation

Parameters may be bounded, e.g. $\theta_i \in [a, b]$.

- We can ignore this and simply discard proposals outside of the range, by setting the log posterior at $-\infty$ outside $[a, b]$
- We can do a transformation, e.g., $\log \theta_i$ if $\theta_i > 0$ and perform a random walk on the unconstrained space: don't forget Jacobians for $q(\cdot)$!
- Another alternative is to use truncated proposals (useful with more complex algorithms like MALA)

Efficient proposals: MALA

The Metropolis-adjusted Langevin algorithm (MALA) uses a Gaussian random walk proposal

$$\boldsymbol{\theta}_t^{\star} \sim \mathsf{Norm}\{\mu(\boldsymbol{\theta}_{t-1}), \tau^2\mathbf{A}\},$$

with mean

$$\mu(\boldsymbol{\theta}_{t-1}) = \boldsymbol{\theta}_{t-1} + \mathbf{A}\eta\nabla \log p(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{y}),$$

and variance $\tau^2\mathbf{A}$, for some mass matrix $\mathbf{A}$, tuning parameter $\tau > 0$.

The parameter $\eta < 1$ is a learning rate. This is akin to a Newton algorithm, so beware if you are far from the mode (where the gradient is typically large)!

Higher order proposals

For a single parameter update θ, a Taylor series expansion of the log posterior around the current value suggests using as proposal density a Gaussian approximation with (Rue & Held, 2005)

- mean $\mu_{t-1} = \theta_{t-1} - f'(\theta_{t-1})/f''(\theta_{t-1})$ and
- precision $\tau^{-2} = -f''(\theta_{t-1})$

We need $f''(\theta_{t-1})$ to be negative!

This gives **local adaption** relative to MALA (global variance)

Higher order and moves

For MALA and cie., we need to compute the density of the proposal also for the reverse move for the expansion starting from the proposal $\mu(\boldsymbol{\theta}_t^\star)$.

These methods are more efficient than random walk Metropolis–Hastings, but they require the gradient and the hessian (can be obtained analytically using autodiff, or numerically).

Modelling individual headlines of Upworthy example

The number of conversions `nclick` is binomial with sample size n_i =`nimpression`.

Since n_i is large, the sample average `nclick`/`nimpression` is approximately Gaussian, so write

[Math Processing Error]

MALA: data set-up

```r
data(upworthy_question, package = "hecbayes")
# Select data for a single question
qdata <- upworthy_question |>
  dplyr::filter(question == "yes") |>
  dplyr::mutate(y = clicks/impressions,
                no = impressions)
```

MALA: define functions

```r
# Create functions with the same signature (...) for the algorithm
logpost <- function(par, data, ...){
  mu <- par[1]; sigma <- par[2]
  no <- data$no
  y <- data$y
  if(isTRUE(any(sigma <= 0, mu < 0, mu > 1))){
    return(-Inf)
  }
  dnorm(x = mu, mean = 0.01, sd = 0.1, log = TRUE) +
  dexp(sigma, rate = 0.7, log = TRUE) +
  sum(dnorm(x = y, mean = mu, sd = sigma/sqrt(no), log = TRUE))
}
```

MALA: compute gradient of log posterior

```r
logpost_grad <- function(par, data, ...){
   no <- data$no
  y <- data$y
  mu <- par[1]; sigma <- par[2]
  c(sum(no*(y-mu))/sigma^2 -(mu - 0.01)/0.01,
    -length(y)/sigma + sum(no*(y-mu)^2)/sigma^3 -0.7
  )
}
```

MALA: compute maximum a posteriori

```r
# Starting values - MAP
map <- optim(
  par = c(mean(qdata$y), 0.5),
  fn = function(x){-logpost(x, data = qdata)},
  gr = function(x){-logpost_grad(x, data = qdata)},
  hessian = TRUE,
  method = "BFGS")
# Check convergence
logpost_grad(map$par, data = qdata)
```

MALA: starting values and mass matrix

```r
# Set initial parameter values
curr <- map$par
# Compute a mass matrix
Amat <- solve(map$hessian)
# Cholesky root - for random number generation
cholA <- chol(Amat)
```

MALA: containers and setup

```r
# Create containers for MCMC
B <- 1e4L # number of iterations
warmup <- 1e3L # adaptation period
npar <- 2L
prop_sd <- rep(1, npar) # tuning parameter
chains <- matrix(nrow = B, ncol = npar)
damping <- 0.8
acceptance <- attempts <- 0
colnames(chains) <- names(curr) <- c("mu","sigma")
# Proposal variance proportional to inverse hessian at MAP
prop_var <- diag(prop_sd) %*% Amat %*% diag(prop_sd)
```

MALA: sample proposal with Newton step

```r
for(i in seq_len(B + warmup)){
  ind <- pmax(1, i - warmup)
  # Compute the proposal mean for the Newton step
  prop_mean <- c(curr + damping *
     Amat %*% logpost_grad(curr, data = qdata))
  # prop <- prop_sd * c(rnorm(npar) %*% cholA) + prop_mean
  prop <- c(mvtnorm::rmvnorm(
    n = 1,
    mean = prop_mean,
    sigma = prop_var))
#  [...]
```

MALA: reverse step

```r
# Compute the reverse step
curr_mean <- c(prop + damping *
    Amat %*% logpost_grad(prop, data = qdata))
# log of ratio of bivariate Gaussian densities
logmh <- mvtnorm::dmvnorm(
  x = curr, mean = prop_mean,
  sigma = prop_var,
  log = TRUE) -
  mvtnorm::dmvnorm(
    x = prop,
    mean = curr_mean,
    sigma = prop_var,
    log = TRUE) +
logpost(prop, data = qdata) -
  logpost(curr, data = qdata)
```

MALA: Metropolis–Hastings ratio

```r
if(logmh > log(runif(1))){
  curr <- prop
  acceptance <- acceptance + 1L
}
attempts <- attempts + 1L
# Save current value
chains[ind,] <- curr
```

MALA: adaptation

```r
  if(i %% 100 & i < warmup){
    # Check acceptance rate and increase/decrease variance
    out <- hecbayes::adaptive(
      attempts = attempts, # counter for number of attempts
      acceptance = acceptance,
      sd.p = prop_sd, #current proposal standard deviation
      target = 0.574) # target acceptance rate
    prop_sd <- out$sd # overwrite current std.dev
    acceptance <- out$acc # if we change std. dev, this is set to zero
    attempts <- out$att # idem, otherwise unchanged
    prop_var <- diag(prop_sd) %*% Amat %*% diag(prop_sd)
  }
} # End of MCMC for loop
```

Gibbs sampling

The Gibbs sampling algorithm builds a Markov chain by iterating through a sequence of conditional distributions.

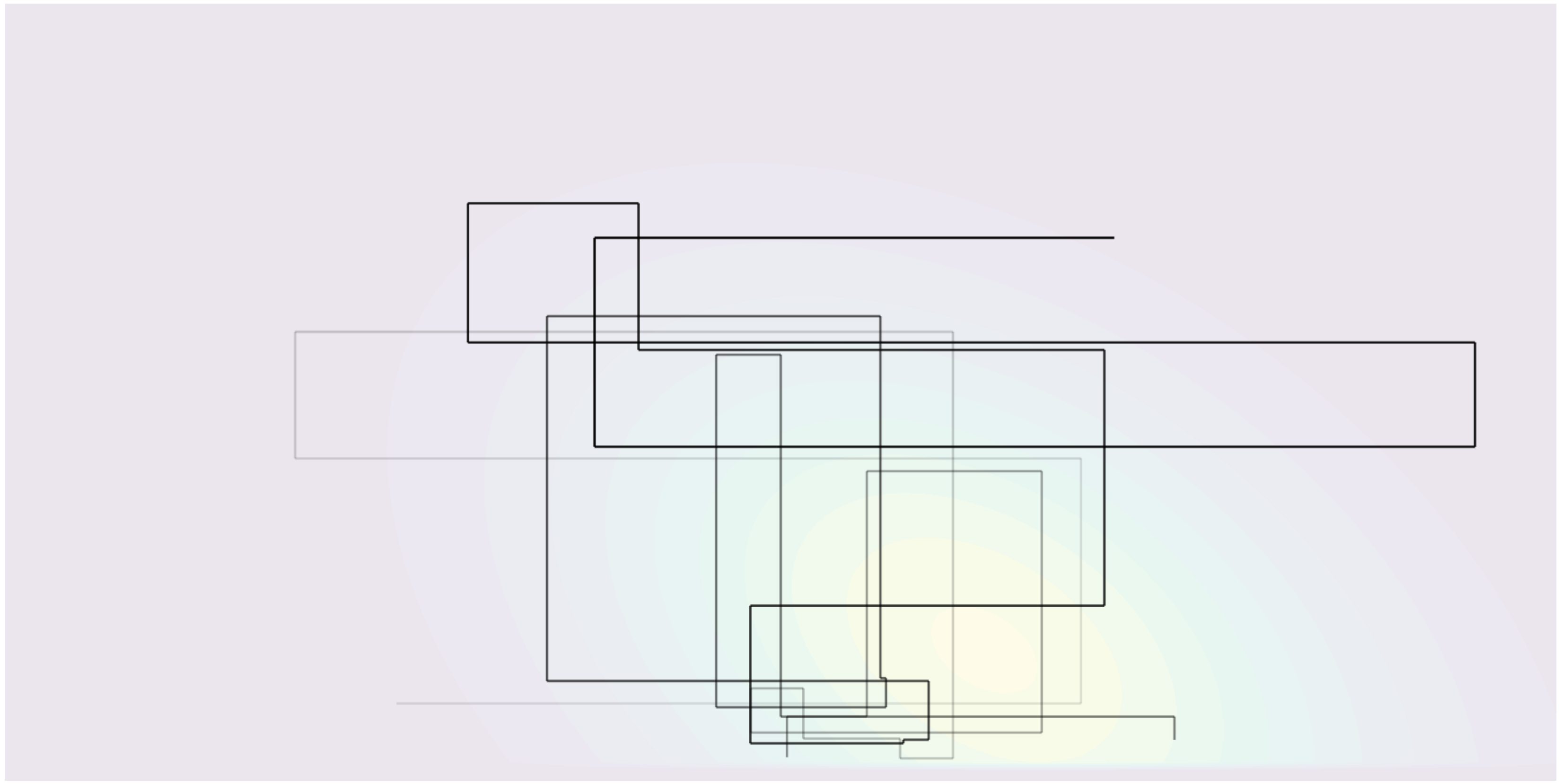


Figure 3: Sampling trajectory for a bivariate target using Gibbs sampling.

Gibbs sampler

Split the parameter vector $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$ into $m \leq p$ blocks,

$$\boldsymbol{\theta}^{[j]} \quad j = 1, \ldots, m$$

such that, conditional on the remaining components of the parameter vector $\boldsymbol{\theta}^{-[j]}$, the conditional posterior

$$p(\boldsymbol{\theta}^{[j]} \mid \boldsymbol{\theta}^{-[j]}, \boldsymbol{y})$$

is from a known distribution from which we can easily simulate.

Gibbs sampling update

At iteration t, we can update each block in turn: note that the kth block uses the partially updated state *[Math Processing Error]* which corresponds to the current value of the parameter vector after the updates.

Notes on Gibbs sampling

- Special case of Metropolis–Hastings with conditional density as proposal q.
- The benefit is that all proposals get accepted, $R = 1$!
- No tuning parameter, but parametrization matters.
- Automatic acceptance does not equal efficiency.

To check the validity of the Gibbs sampler, see the methods proposed in Geweke (2004).

Efficiency of Gibbs sampling

As the dimension of the parameter space increases, and as the correlation between components becomes larger, the efficiency of the Gibbs sampler degrades

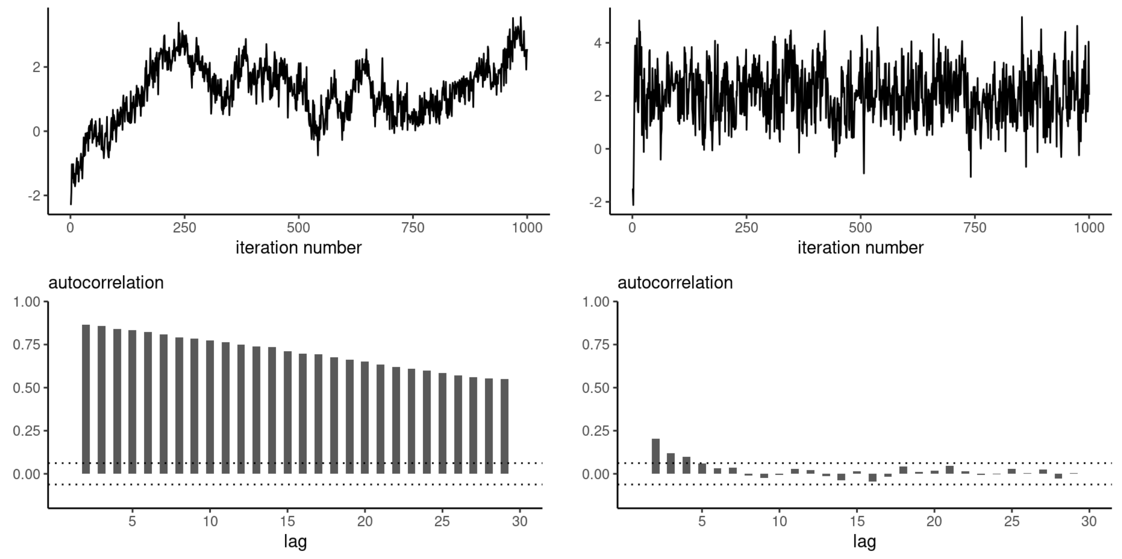


Figure 4: Trace plots (top) and correlograms (bottom) for the first component of a Gibbs sampler with $d = 20$ equicorrelated Gaussian variates with correlation $\rho = 0.9$ (left) and $d = 3$ with equicorrelation $\rho = 0.5$ (right).

Gibbs sampling requires work!

- You need to determine all of the relevant conditional distributions, which often relies on setting conditionally conjugate priors.
- In large models with multiple layers, full conditionals may only depend on a handful of parameters (via directed acyclic graph and moral graph of the model; not covered).

Example of Gibbs sampling

Consider independent and identically distributed observations, with *[Math Processing Error]*

The joint posterior is not available in closed form, but the independent priors for the mean and variance of the observations are conditionally conjugate.

Joint posterior for Gibbs sample

Write the posterior density as usual, *[Math Processing Error]*

Recognizing distributions from posterior

Consider the conditional densities of each parameter in turn (up to proportionality): *[Math Processing Error]*

Gibs sample

We can simulate in turn *[Math Processing Error]*

Data augmentation and auxiliary variables

When the likelihood $p(\boldsymbol{y}; \boldsymbol{\theta})$ is intractable or costly to evaluate (e.g., mixtures, missing data, censoring), auxiliary variables are introduced to simplify calculations.

Consider auxiliary variables $\boldsymbol{u} \in \mathbb{R}^k$ such that

$$\int_{\mathbb{R}^k} p(\boldsymbol{u}, \boldsymbol{\theta} \mid \boldsymbol{y}) \mathrm{d}\boldsymbol{u} = p(\boldsymbol{\theta} \mid \boldsymbol{y}),$$

i.e., the marginal distribution is that of interest, but evaluation of $p(\boldsymbol{u}, \boldsymbol{\theta}; \boldsymbol{y})$ is cheaper.

Bayesian augmentation

The data augmentation algorithm (Tanner & Wong, 1987) consists in running a Markov chain on the augmented state space $(\Theta, \mathbb{R}^k)$, simulating in turn from the conditionals

- $p(\boldsymbol{u} \mid \boldsymbol{\theta}, \boldsymbol{y})$ and
- $p(\boldsymbol{\theta} \mid \boldsymbol{u}, \boldsymbol{y})$

For more details and examples, see Dyk & Meng (2001) and Hobert (2011).

Data augmentation: probit example

Consider independent binary responses $\boldsymbol{Y}_i$, with *[Math Processing Error]* where Φ is the distribution function of the standard Gaussian distribution. The likelihood of the probit model is

$$L(\boldsymbol{\beta}; \boldsymbol{y}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i},$$

and this prevents easy simulation.

Probit augmentation

We can consider a data augmentation scheme where $Y_i = \mathsf{I}(Z_i > 0)$, where $Z_i \sim \mathsf{Norm}(\mathbf{x}_i\boldsymbol{\beta}, 1)$, where $\mathbf{x}_i$ is the ith row of the design matrix.

The augmented data likelihood is *[Math Processing Error]*

Conditional distributions for probit regression

[Math Processing Error] with $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{z}$ the ordinary least square estimator.

Data augmentation with scale mixture of Gaussian

The Laplace distribution with mean μ and scale σ, which has density [Math Processing Error] can be expressed as a scale mixture of Gaussians, where $Y \mid \tau \sim \mathsf{La}(\mu, \tau)$ is equivalent to $Z \mid \tau \sim \mathsf{Norm}(\mu, \tau)$ and $\tau \sim \mathsf{Exp}\{(2\sigma)^{-1}\}$.

Joint posterior for Laplace model

With $p(\mu, \sigma) \propto \sigma^{-1}$, the joint posterior for the i.i.d. sample is *[Math Processing Error]*

Conditional distributions

The conditionals for $\mu \mid \cdots$ and $\sigma \mid \cdots$ are, as usual, Gaussian and inverse gamma, respectively. The variances, τ_j, are conditionally independent of one another, with *[Math Processing Error]*

Inverse transformation

With the change of variable $\xi_j = 1/\tau_j$, we have *[Math Processing Error]* and we recognize the Wald (or inverse Gaussian) density, where $\xi_i \sim \mathsf{Wald}(\nu_i, \lambda)$ with $\nu_i = \{\sigma/(y_i-\mu)^2\}^{1/2}$ and $\lambda = \sigma^{-1}$.

Bayesian LASSO

Park & Casella (2008) use this hierarchical construction to defined the Bayesian LASSO. With a model matrix $\mathbf{X}$ whose columns are standardized to have mean zero and unit standard deviation, we may write *[Math Processing Error]*

Comment about Bayesian LASSO

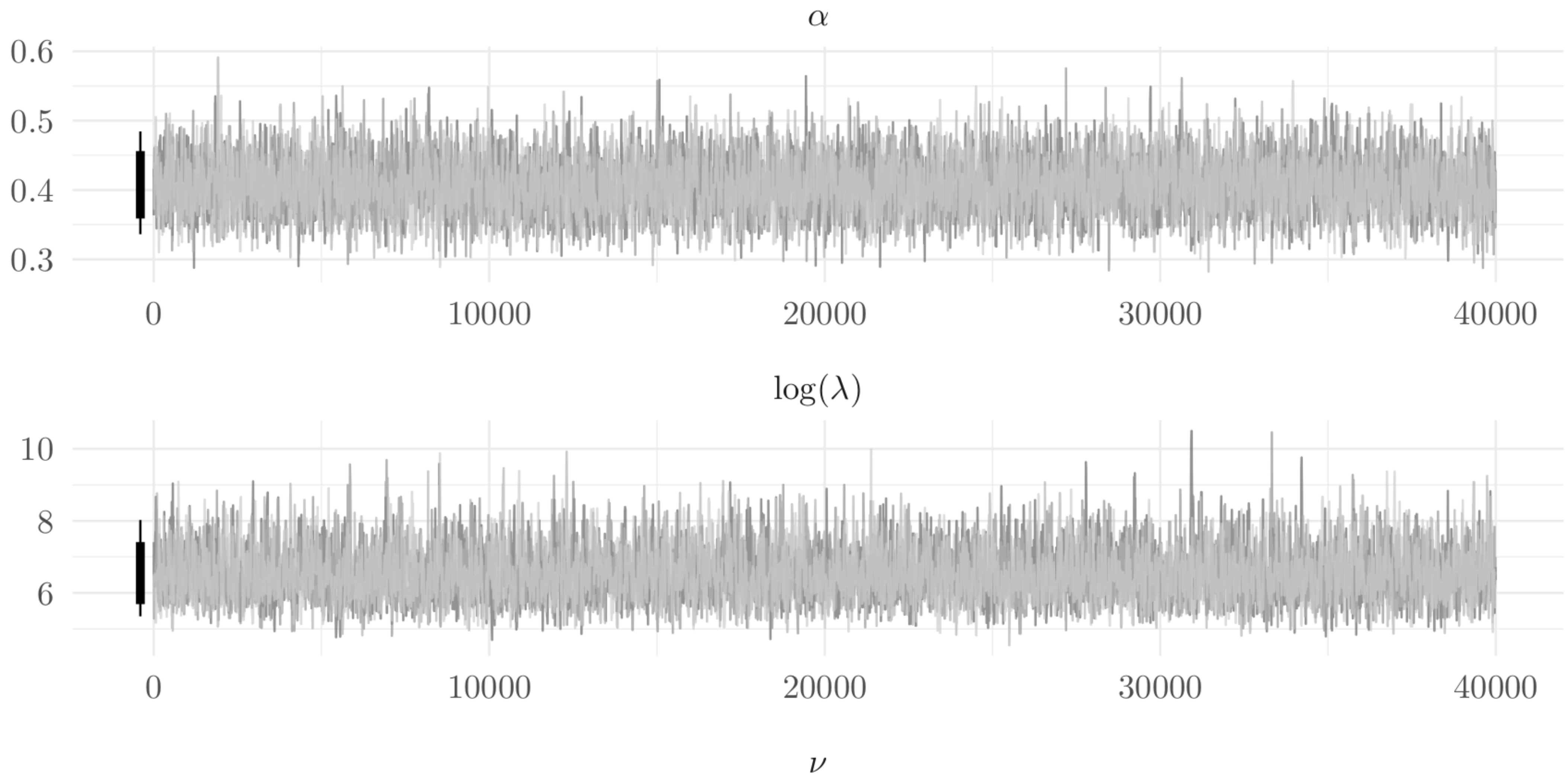
- If we set an improper prior $p(\mu, \sigma) \propto \sigma^{-1}$, the resulting conditional distributions are all available and thus the model is amenable to Gibbs sampling.
- The Bayesian LASSO places a Laplace penalty on the regression coefficients, with lower values of λ yielding more shrinkage.
- Contrary to the frequentist setting, none of the posterior draws of $\boldsymbol{\beta}$ are exactly zero.

Visual diagnostic: trace plots

Display the Markov chain sample path as a function of the number of iterations.

- Run multiple chains to see if they converge to the same target.
  - if not, check starting values (compare log posterior) or parameter identifiability!
- Markov chains should look like a fat hairy caterpillar!
- `bayesplot` and `coda` have functionalities for plots (trace plot, trace rank, correlograms, marginal densities, etc.)

Checking convergence with multiple chains



Four healthy parallel chains for parameters.

Effective sample size

Are my chains long enough to compute reliable summaries?

Compute the sample size we would have with independent draws by taking [Math Processing Error] where γ_t is the lag t autocorrelation.

The relative effective sample size is simply ESS/B: small values indicate pathological or inefficient samplers.

How many samples?

We want our average estimate to be reliable!

- We probably need ESS to be several hundred
- We can estimate the variance of the target to know the precision
- (related question: how many significant digits to report?)

In **R**, via `coda::effectiveSize()`

Estimating the variance (block method)

1. Break the chain of length B (after burn in) in K blocks of size $\approx K/B$.
2. Compute the sample mean of each segment. These values form a Markov chain and should be approximately uncorrelated.
3. Compute the standard deviation of the segments mean. Rescale by $K^{-1/2}$ to get standard error of the global mean.

More efficient methods using overlapping blocks exists.

Block means in pictures

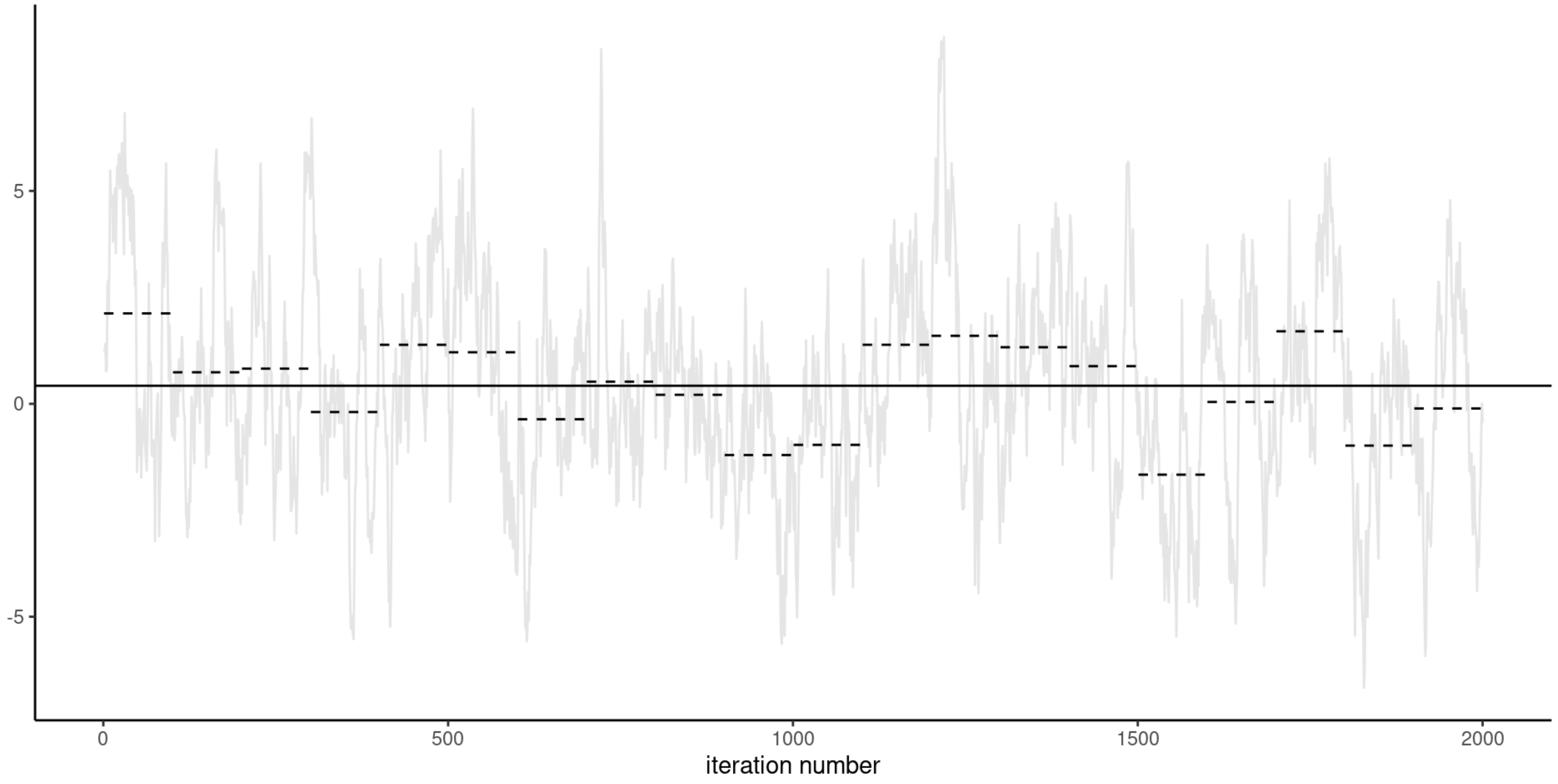


Figure 5: Calculation of the standard error of the posterior mean using the batch method.

Cautionary warning about stationarity

Batch means only works if the chain is sampling from the stationary distribution!

The previous result (and any estimate) will be unreliable and biased if the chain is not (yet) sampling from the posterior.

Lack of stationarity

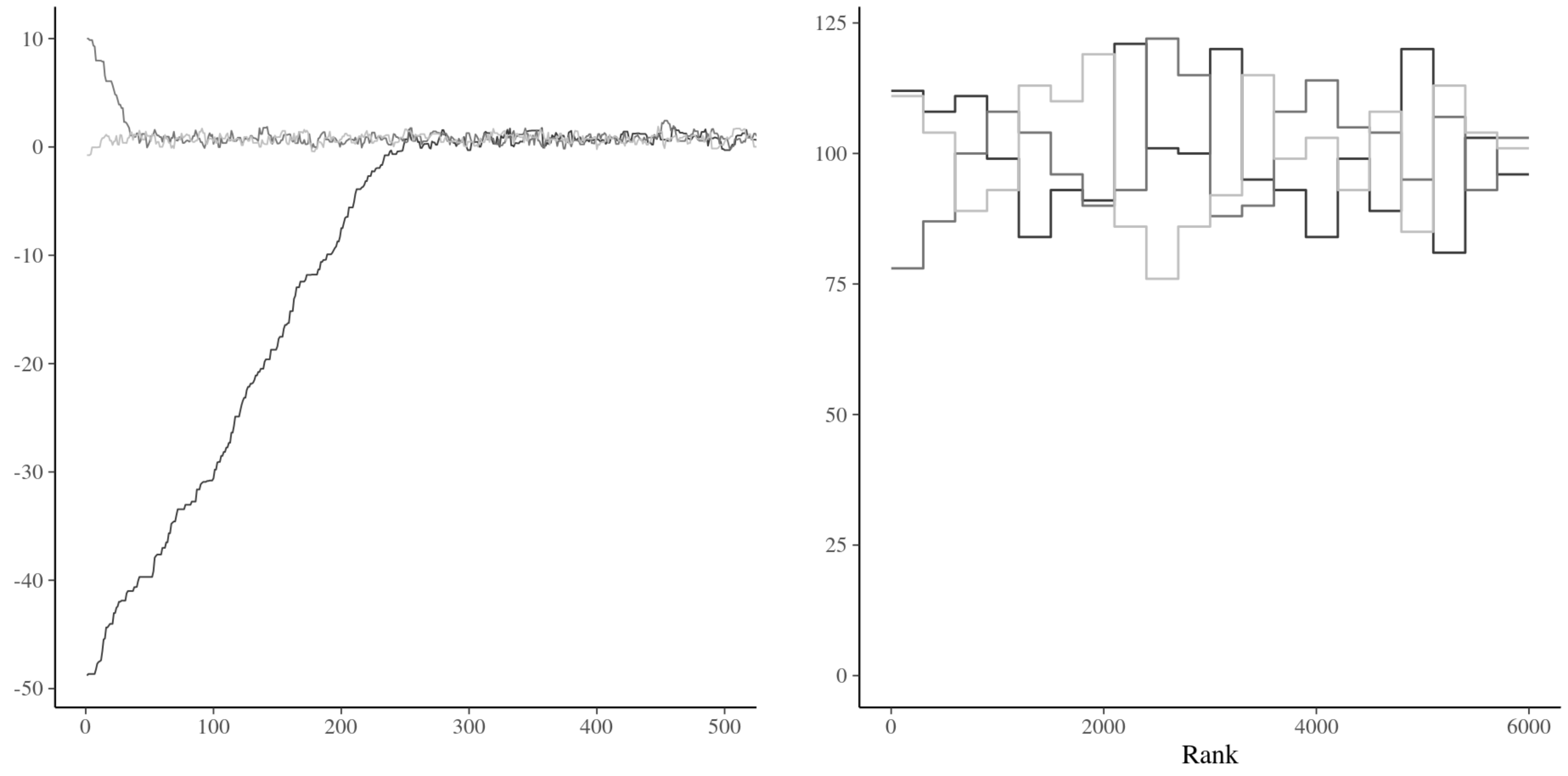


Figure 6: Traceplots of three Markov chains for the same target with different initial values for the first 500 iterations (left) and trace rank plot after discarding these (right). The latter is indicative of the speed of mixing.

Potential scale reduction statistic

The Gelman–Rubin diagnostic, denoted $\widehat{R}$, is obtained by running multiple chains and considering the difference between within-chain and between-chains variances,

[Math Processing Error]

Any value of $\widehat{R}$ larger 1 is indicative of problems of convergence.

Bad chains

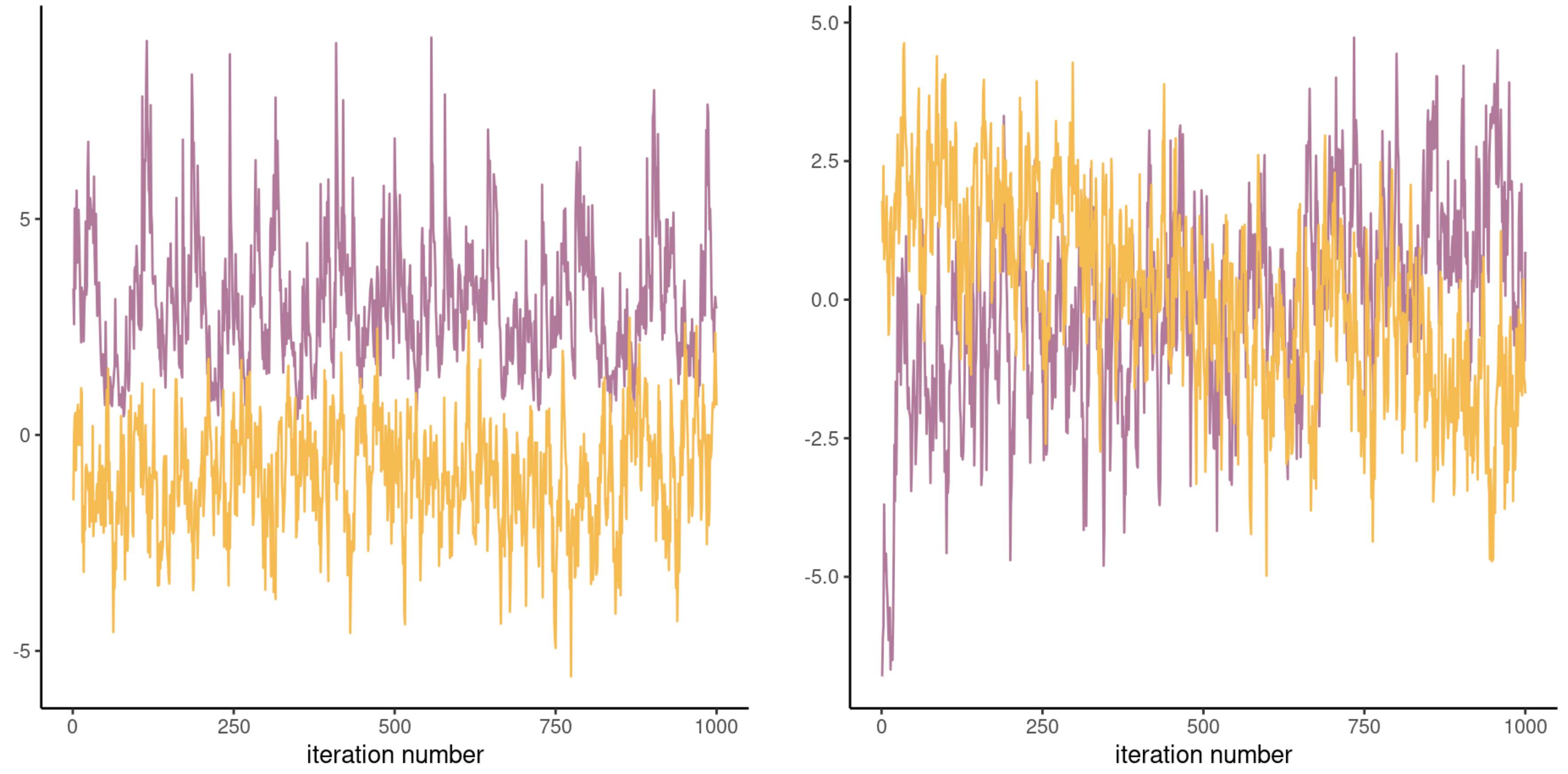


Figure 7: Two pairs of Markov chains: the top ones seem stationary, but with different modes and $\widehat{R} \approx 3.4$. The chains on the right hover around zero, but do not appear stable, with $\widehat{R} \approx 1.6$.

Posterior predictive checks

1. For each of the B draws from the posterior, simulate n observations from the posterior predictive $p(\tilde{\boldsymbol{y}} \mid \boldsymbol{y})$
2. For each replicate, compute a summary statistics (median, quantiles, std. dev., etc.)
3. Compare it with the same summary computed for the sample $\boldsymbol{y}$.

Posterior predictive checks

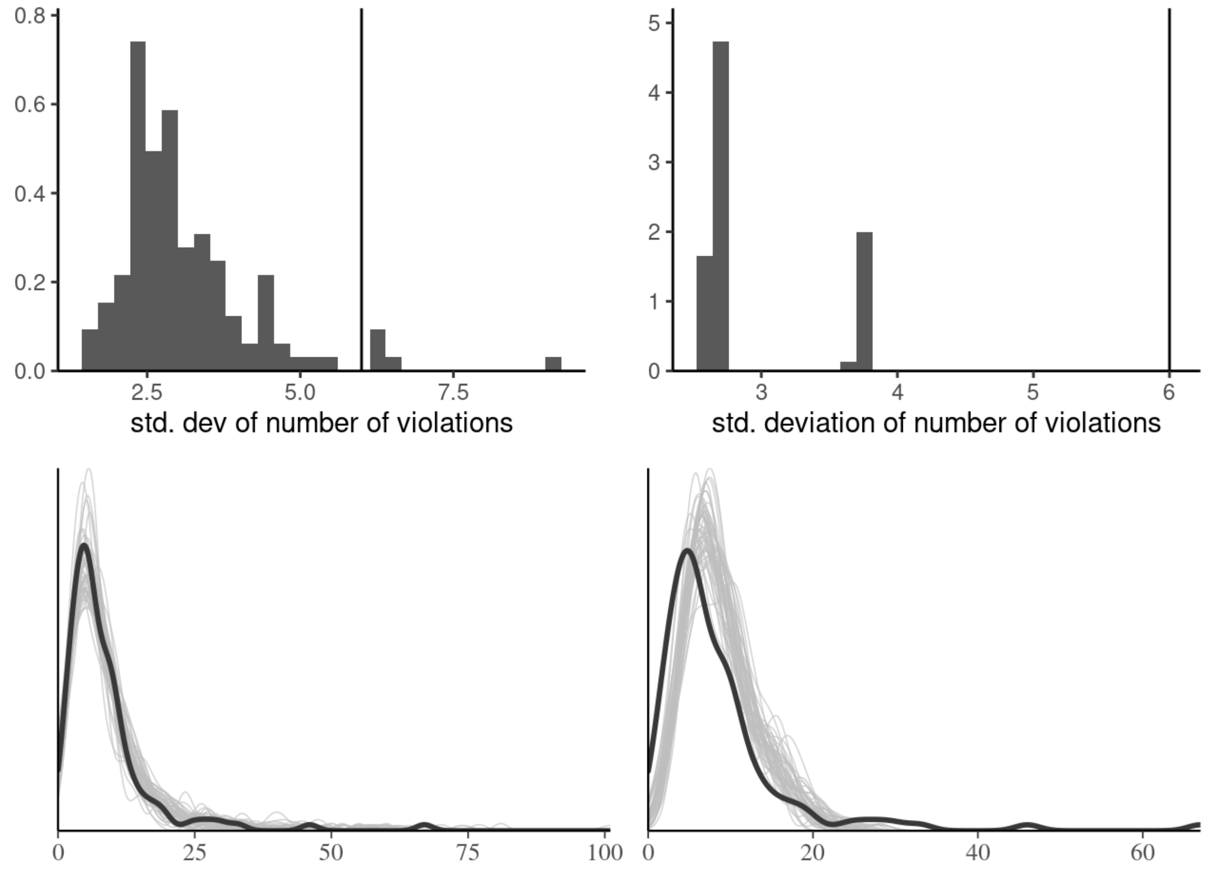


Figure 8: Posterior predictive checks for the standard deviation (top) and density of posterior draws (bottom) for hierarchical Poisson model with individual effects (left) and simpler model with only conditions (right).

Log pointwise predictive density

Consider the expected value of the log observation-wise log posterior with respect to the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{y})$,

[Math Processing Error]

We can approximate this quantity pointwise by averaging the log posterior of each sample observation over the posterior samples via Monte Carlo.

The higher the value of LPPD, the better the fit.

Widely available information criterion

To build an information criterion, we add a penalization factor that approximates the effective number of parameters in the model, with *[Math Processing Error]* where we use again the empirical variance to compute the rightmost term.

Smaller values of WAIC are better.

Bayesian leave-one-out cross validation

In Bayesian setting, we can use the leave-one-out predictive density

$$p(y_i \mid \boldsymbol{y}_{-i})$$

as a measure of predictive accuracy. the

We can use importance sampling to approximate the latter.

Requirement: need to keep track of the log likelihood of each observation for each posterior draw ($B \times n$ values).

LOO-CV diagnostics

We can draw B samples from $p(\tilde{y} \mid \boldsymbol{y}_{-i})$ and compute the rank of y_i.

Under perfect calibration, ranks should be uniform.

Leave-one-out with quantile-quantile plots

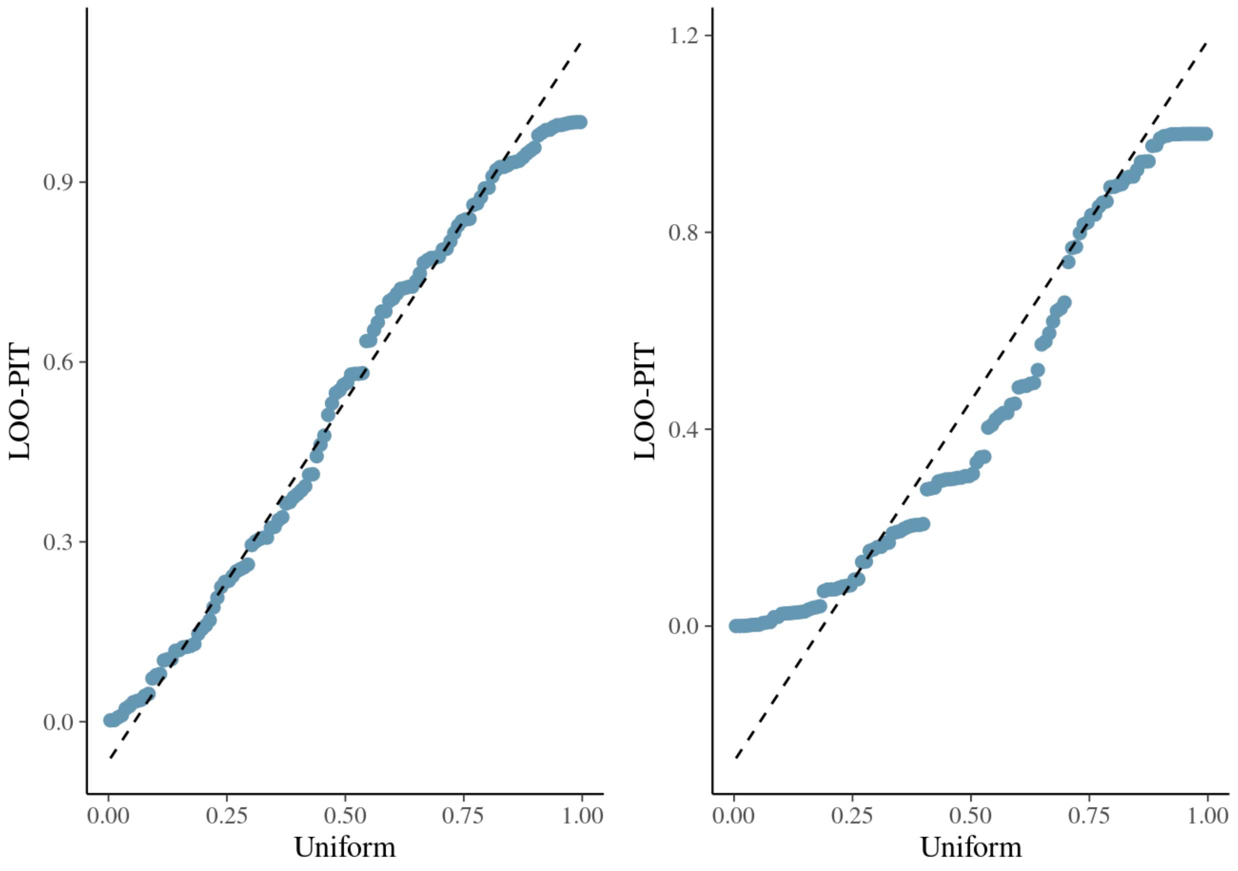


Figure 9: Quantile-quantile plots based on leave-one-out cross validation for model for the hierarchical Poisson model fitted to the Upworthy data with the individual random effects (left) and without (right).

References

Dyk, D. A. van, & Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, *10*(1), 1–50. https://doi.org/10.1198/10618600152418584

Geweke, J. (2004). Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, *99*(467), 799–804. https://doi.org/10.1198/016214504000001132

Hobert, J. P. (2011). The data augmentation algorithm: Theory and methodology. In S. Brooks, A. Gelman, G. Jones, & X. L. Meng (Eds.), *Handbook of Markov chain Monte Carlo* (pp. 253–294). CRC Press. https://doi.org/10.1201/b10905

Park, T., & Casella, G. (2008). The Bayesian Lasso. *Journal of the American Statistical Association*, *103*(482), 681–686. https://doi.org/10.1198/016214508000000337

Rue, H., & Held, L. (2005). *Gaussian Markov random fields: Theory and applications* (p. 280). CRC Press.

Sherlock, C. (2013). Optimal scaling of the random walk Metropolis: General criteria for the 0.234 acceptance rule. *Journal of Applied Probability*, *50*(1), 1–15. https://doi.org/10.1239/jap/1363784420