

# Bayesian modelling

## Simulation-based inference

Léo Belzile  
2023

# Bayesian inference beyond conjugate models

How to circumvent the problem of intractable posteriors?

- simulation-based methods: accept-reject, Markov chain Monte Carlo, particle filters, etc.
- deterministic methods: (integrated nested) Laplace approximations, variational Bayes, expectation propagation, etc.

We focus on the Monte Carlo methods in the sequel.

## Objective of methods

Suppose we can simulate  $B$  i.i.d. variables with the same distribution,  $X_b \sim F$  ( $b = 1, \dots, B$ ).

We want to compute  $\mathbf{E}\{g(X)\} = \mu_g$  for some functional  $g(\cdot)$

- $g(x) = x$  (posterior mean)
- $g(x) = \mathbf{I}(x \in A)$  (probability of event)
- etc.

# Monte Carlo methods

We substitute expected value by sample average

$$\hat{\mu}_g = \frac{1}{B} \sum_{b=1}^B g(X_b), \quad X_b \sim F$$

- law of large number guarantees convergence of  $\hat{\mu}_g \rightarrow \mu_g$  if the latter is finite.
- central limit theorem applies:

$$\sqrt{B}(\hat{\mu}_g - \mu_g) \sim \text{No}(0, \sigma_g^2).$$

# Ordinary Monte Carlo

We want to have an estimator as precise as possible.

- but we can't control the variance of  $g(X)$ , say  $\sigma_g^2$
- the more simulations  $B$ , the lower the variance of the mean.
- sample average for i.i.d. data has variance  $\sigma_g^2 / B$
- to reduce the standard deviation by a factor 10, we need 100 times more draws!

Remember: the answer is **random**.

# Example: functionals of gamma distribution

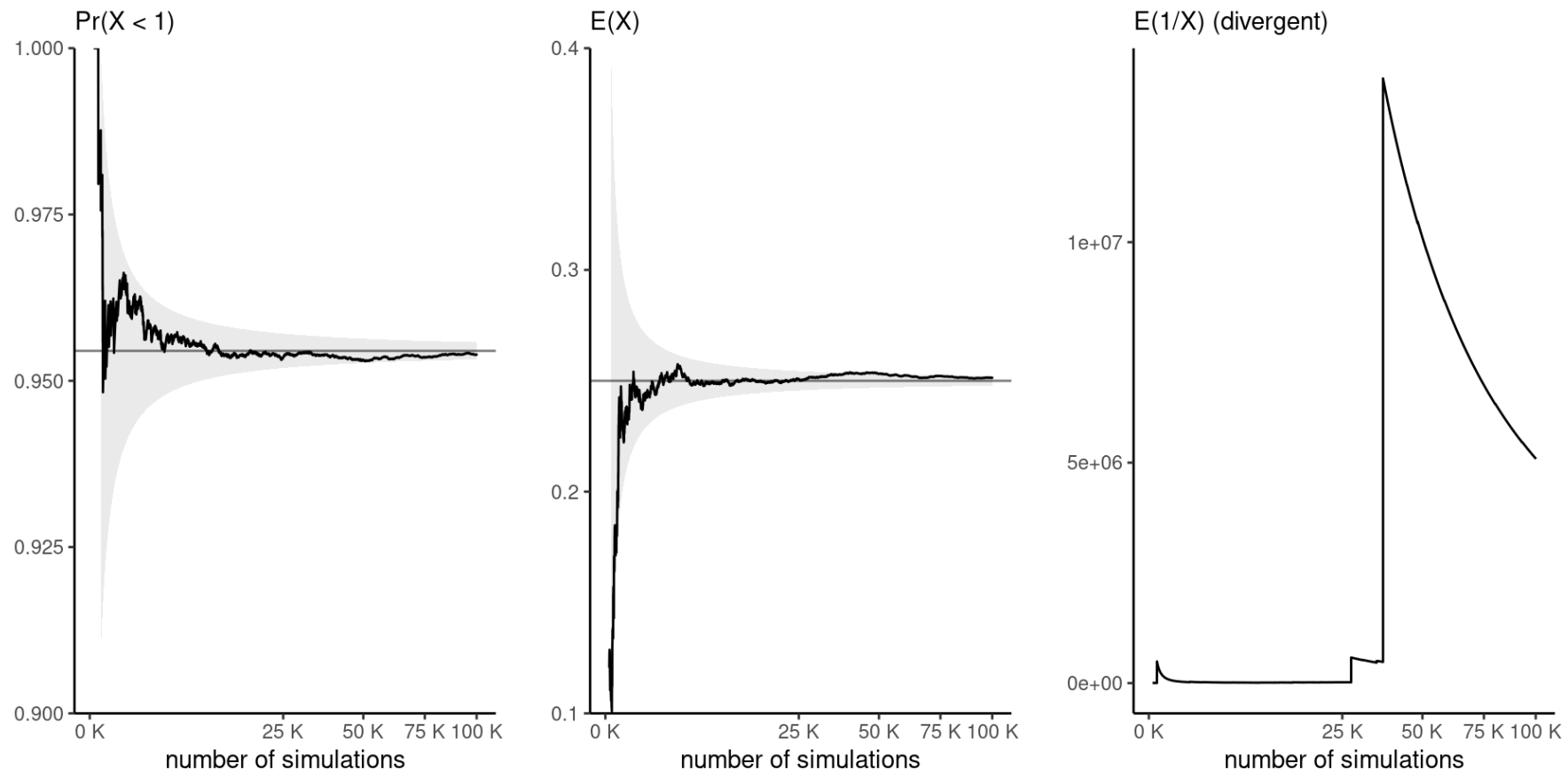


Figure 1: Running mean trace plots for  $g(x) = \mathbf{I}(x < 1)$  (left),  $g(x) = x$  (middle) and  $g(x) = 1/x$  (right) for a Gamma distribution with shape 0.5 and rate 2, as a function of the Monte Carlo sample size.

# Simulation algorithms: inversion method

If  $F$  is an absolutely continuous distribution function, then

$$F(X) \sim U(0, 1).$$

The inversion method consists in applying the quantile function  $F^{-1}$  to  $U \sim U(0, 1)$ , viz.

$$F^{-1}(U) \sim X.$$

## Inversion method for truncated distributions

Consider a random variable  $Y$  with distribution function  $F$ . If  $X$  follows the same distribution as  $Y$ , but restricted over the interval  $[a, b]$ , then

$$\Pr(X \leq x) = \frac{F(x) - F(a)}{F(b) - F(a)}, \quad a \leq x \leq b,$$

Therefore,

$$F^{-1}[F(a) + \{F(b) - F(a)\}U] \sim X$$



# Simulation algorithms: accept-reject

- **Target:** sample from density  $p(x)$  (hard to sample from)
- **Proposal:** find a density  $q(x)$  with nested support,  $\text{supp}(p) \subset \text{supp}(q)$ , such that

$$\frac{p(x)}{q(x)} \leq C, \quad C \geq 1.$$

# Rejection sampling algorithm

1. Generate  $X$  from proposal with density  $q(x)$ .
2. Compute the ratio  $R \leftarrow p(X)/q(X)$ .
3. If  $R \leq CU$  for  $U \sim \text{U}(0, 1)$ , return  $X$ , else go back to step 1.

# Remarks on rejection sampling

- Acceptance rate is  $1/C$ 
  - we need on average  $C$  draws from  $q$  to get one from  $p$
- $q$  must be more heavy-tailed than  $p$ 
  - e.g.,  $q(x)$  Student- $t$  for  $p(x)$  Gaussian
- $q$  should be cheap and easy to sample from!

# Designing a good proposal density

Good choices must satisfy the following constraints:

- pick a family  $q(x)$  so that

$$C = \sup_x \frac{p(x)}{q(x)}$$

is as close to 1 as possible.

- you can use numerical optimization with  $f(x) = \log p(x) - \log q(x)$  to find the mode  $x^*$  and the upper bound  $C = \exp f(x^*)$ .

## Example of accept-reject (1)

Consider sampling  $Y \sim \text{No}(\mu, \sigma^2)$ , but truncated in the interval  $(a, b)$ . The target density is

$$p(x; \mu, \sigma, a, b) = \frac{1}{\sigma} \frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi(\beta) - \Phi(\alpha)}.$$

for  $\alpha = (a - \mu)/\sigma$  and  $\beta = (b - \mu)/\sigma$ . where  $\phi(\cdot)$ ,  $\Phi(\cdot)$  are respectively the density and distribution function of the standard Gaussian distribution.

# Accept-reject (crude version)

1. Simulate  $X \sim \text{No}(\mu, \sigma^2)$
2. reject any draw if  $X < a$  or  $X > b$ .

The acceptance rate is  $C^{-1} = \{\Phi(\beta) - \Phi(\alpha)\}$

```
1 # Standard Gaussian truncated on [0,1]
2 candidate <- rnorm(1e5)
3 trunc_samp <- candidate[candidate >= 0 & candidate <= 1]
4 # Acceptance rate
5 length(trunc_samp)/1e5
```

```
[1] 0.34242
```

```
1 # Theoretical acceptance rate
2 pnorm(1)-pnorm(0)
```

```
[1] 0.3413447
```

# Accept-reject for truncated Gaussian

Since the Gaussian is a location scale family, the inversion method gives

$$X \sim \mu + \sigma \Phi^{-1} [\Phi(\alpha) + \{\Phi(\beta) - \Phi(\alpha)\}U]$$

We however need to evaluate  $\Phi$  numerically (no closed-form expression).

The method fails for *rare event* simulation because the computer returns

- $\Phi(x) = 0$  for  $x \leq -39$
- $\Phi(x) = 1$  for  $x \geq 8.3$ ,

implying that  $a \leq 8.3$  for this approach to work (Botev & L'Écuyer, 2017).

# Simulating tails of Gaussian variables

We consider simulation from a standard Gaussian truncated above  $a > 0$  (Devroye, 1986, p. 381).

Write the density of the truncated Gaussian as

$$f(x) = \frac{\exp(-x^2/2)}{\int_a^\infty \exp(-z^2/2)dz} = \frac{\exp(-x^2/2)}{c_1}.$$

Note that, for  $x \geq a$ ,

$$c_1 f(x) \leq \frac{x}{a} \exp\left(-\frac{x^2}{2}\right) = a^{-1} \exp\left(-\frac{a^2}{2}\right) g(x);$$

where  $g(x)$  is the density of a Rayleigh variable shifted by  $a$ .<sup>1</sup>

<sup>1</sup> The constant  $C = \exp(-a^2/2)(c_1 a)^{-1}$  approaches 1 quickly as  $a \rightarrow \infty$  (asymptotically optimality)



# Accept-reject: truncated Gaussian with Rayleigh

The shifted Rayleigh has distribution function

$$G(x) = 1 - \exp\{(a^2 - x^2)/2\}, x \geq a.$$

## ! Marsaglia algorithm

1. Generate a shifted Rayleigh above  $a$ ,  $X \leftarrow \{a^2 - 2 \log(U)\}^{1/2}$  for  $U \sim U(0, 1)$
2. Accept  $X$  if  $XV \leq a$ , where  $V \sim U(0, 1)$ .

For sampling on  $[a, b]$ , propose from a Rayleigh truncated above at  $b$  (Botev & L'Écuyer, 2017).

```
1 a <- 8.3
2 niter <- 1000L
3 X <- sqrt(a^2 + 2*rexp(niter))
4 samp <- X[runif(niter)*X <= a]
```

# Markov chains

Plain ordinary Monte Carlo is great, but few algorithms are generic enough to be useful in complex high-dimensional problems.

We will instead typically build Markov chains that target an invariant stationary distribution.

# Caveats?

Markov chain Monte Carlo methods generate **correlated** draws.

## Questions:

1. can we use them as ordinary independent samples?
2. what is the price to pay?

We need to do a little theoretical detour to answer these questions.

# Stationarity and Markov property

A stochastic process is **(weakly) stationary** if

- the distribution of  $\{X_1, \dots, X_t\}$  is the same as that of  $\{X_{n+1}, \dots, X_{t+n}\}$  for any value of  $n$  and given  $t$ .

A stochastic process is **Markov** if

- it satisfies the Markov property: given the current state of the chain, the future only depends on the current state and not on the past.

# Autoregressive process of order 1

Consider a first-order autoregressive process, or AR(1),

$$Y_t = \mu + \phi(Y_{t-1} - \mu) + \varepsilon_t,$$

where

- $\phi$  is the lag-one correlation,
- $\mu$  the global mean
- $\varepsilon_t$  is an iid innovation with mean zero and variance  $\sigma^2$

If  $|\phi| < 1$ , the process is stationary, otherwise variance increases with  $t$

# Variance of a stationary distribution

For a correlated sequence, the variance of the stationary distribution is

$$\tau^2 = \text{Va}(Y_t) + 2 \sum_{k=1}^{\infty} \text{Co}(Y_t, Y_{t-k}).$$

- for i.i.d. data,  $\tau^2 = \text{Va}(Y_t)$
- for stationary  $\text{AR}(1)$  process, we get  $\sigma^2 / (1 - \phi^2)$  (geometric series)

## Variance of sample average

Intuitively, a sample of correlated observations carries less information than an independent sample of draws.

We want the variance of the sample average, which is

$$\text{Va} \left( \bar{Y}_T \right) = \frac{1}{T} \sum_{t=1}^T \text{Va}(Y_t) + \frac{2}{T} \sum_{t=1}^{T-1} \sum_{s=t+1}^T \text{Co}(Y_t, Y_s).$$

If the process is stationary, the covariances at lag  $k$  are the same regardless of the time index and the unconditional variance is constant.

## Variance of sample average, redux

If a central limit theorem applies, the limiting variance of the sample mean simplifies to

$$\lim_{T \rightarrow \infty} T \text{Va} \left( \bar{Y}_T \right) = \tau^2 \left\{ 1 + 2 \sum_{t=1}^{\infty} \gamma_t \right\}.$$

which is a function of

- the unconditional variance  $\tau^2$
- the lag- $k$  autocorrelation  $\text{Cor}(Y_t, Y_{t+k}) = \gamma_k$



# Correlogram

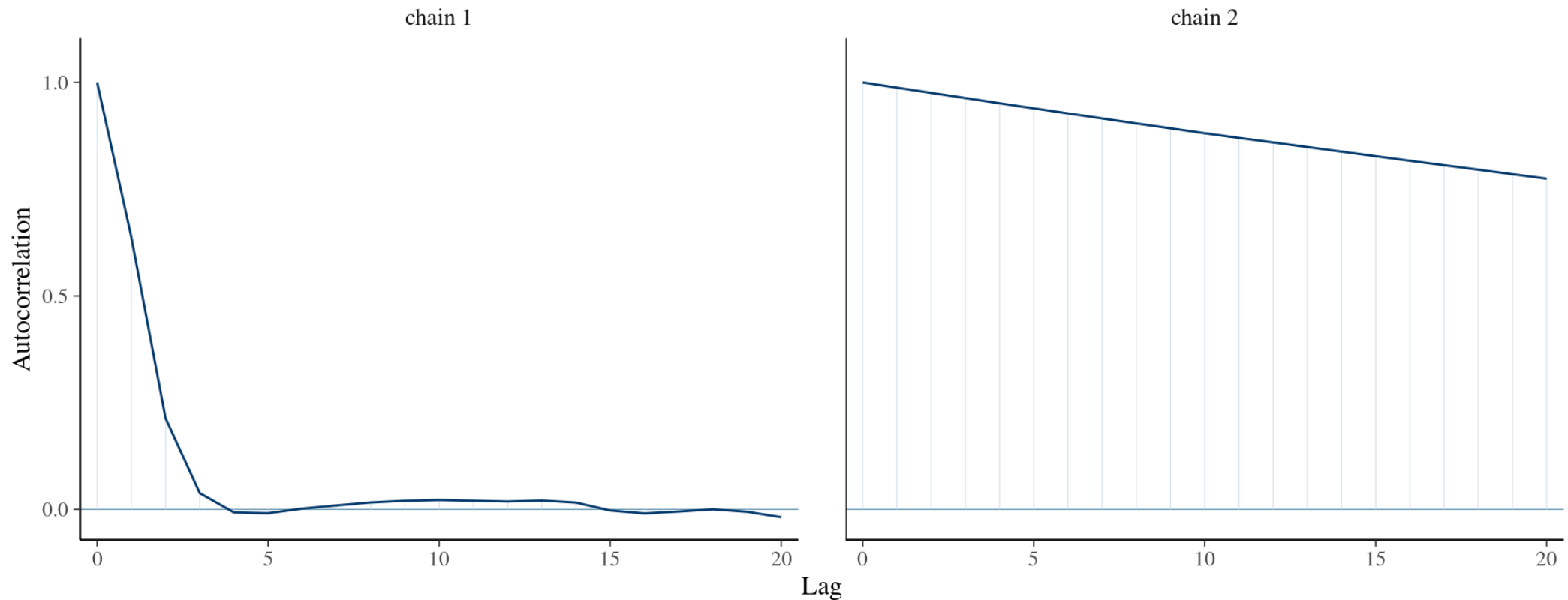


Figure 2: Correlogram of two two Markov chains. These plots, often called acf or autocorrelation functions, show the lag-k sample autocorrelation against lag number.

## Variance of sample mean of AR(1)

The lag- $k$  correlation of the stationary autoregressive process of order 1 is  $\phi^k$ , so

$$TVa \left( \bar{Y}_T \right) = \sigma^2 (1 + \phi) / (1 - \phi).$$

For an independent sample, we have

$$TVa \left( \bar{Y}_T \right) = \sigma^2 / (1 - \phi^2).$$

## Morale of the story

The price to pay for having correlated samples is

**inefficiency**

The higher the autocorrelation, the larger the variability of our estimators.

# Inefficiency curve

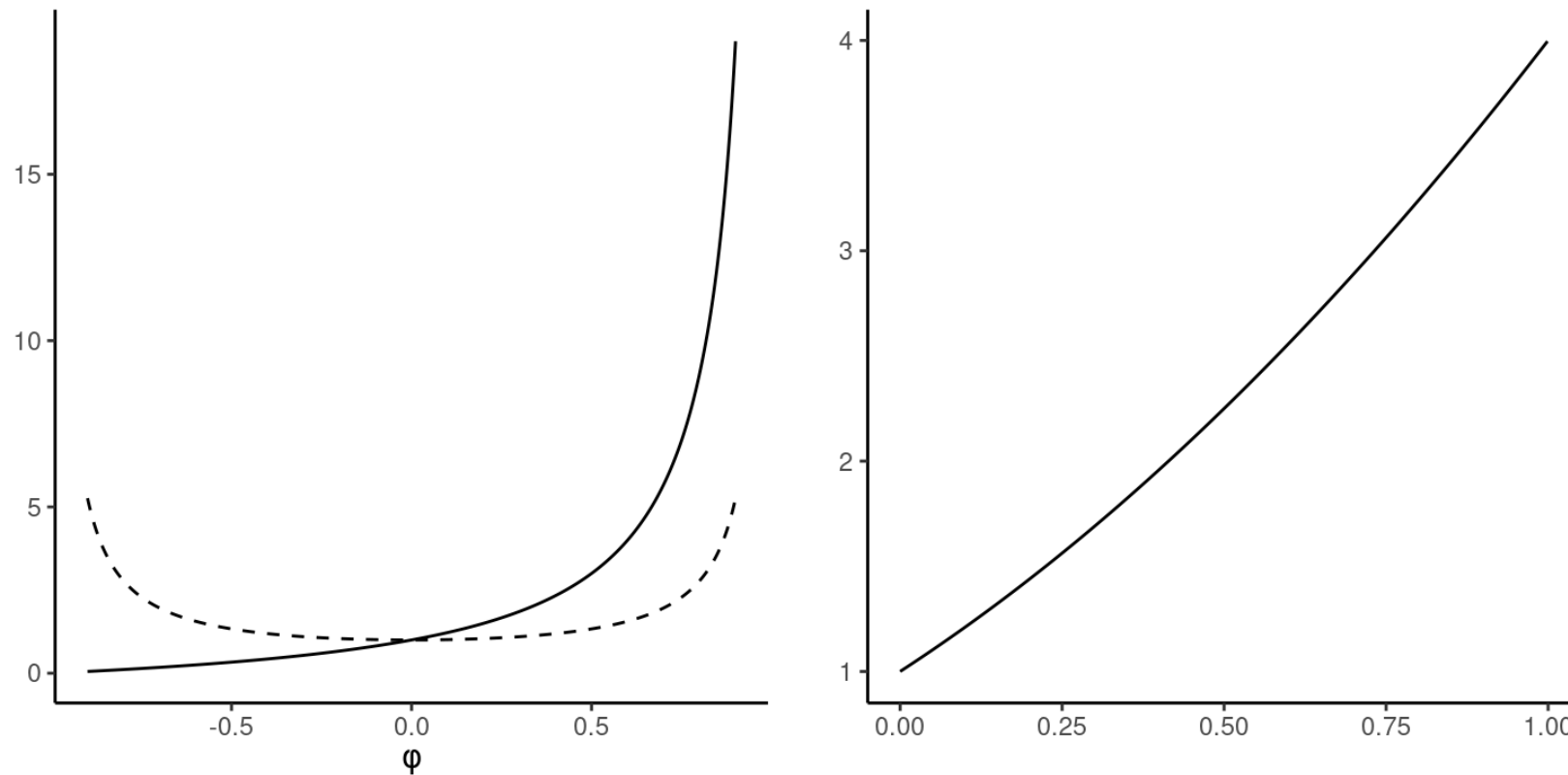


Figure 3: Scaled asymptotic variance of the sample mean for AR(1) (dashed) and independent observations with the same marginal variance (full line). The plot on the right gives the variance ratio for positive correlations.

To get the same precision for the mean of **AR(1)** process with  $\phi \approx 0.75$  than with i.i.d. data, we would need 9 times as many observations.

# When can we use Markov chains?

If a Markov chain is irreducible and aperiodic, it has a unique stationary distribution.

- irreducibility: means that the chain can move from anywhere to anywhere, so it doesn't get stuck in part of the space forever.
- acyclic: cyclical chains loop around and visit periodically a state

Ergodic theorem is our guarantee of convergence.

# Examples

Consider discrete Markov chains over the integers 1, 2, 3 with transition matrices

$$P_1 = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0 & 0.4 & 0.6 \\ 0 & 0.5 & 0.5 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Chain 1 is reducible to  $\{2, 3\}$ , chain 2 is cyclical.



# Convergence of Markov chains

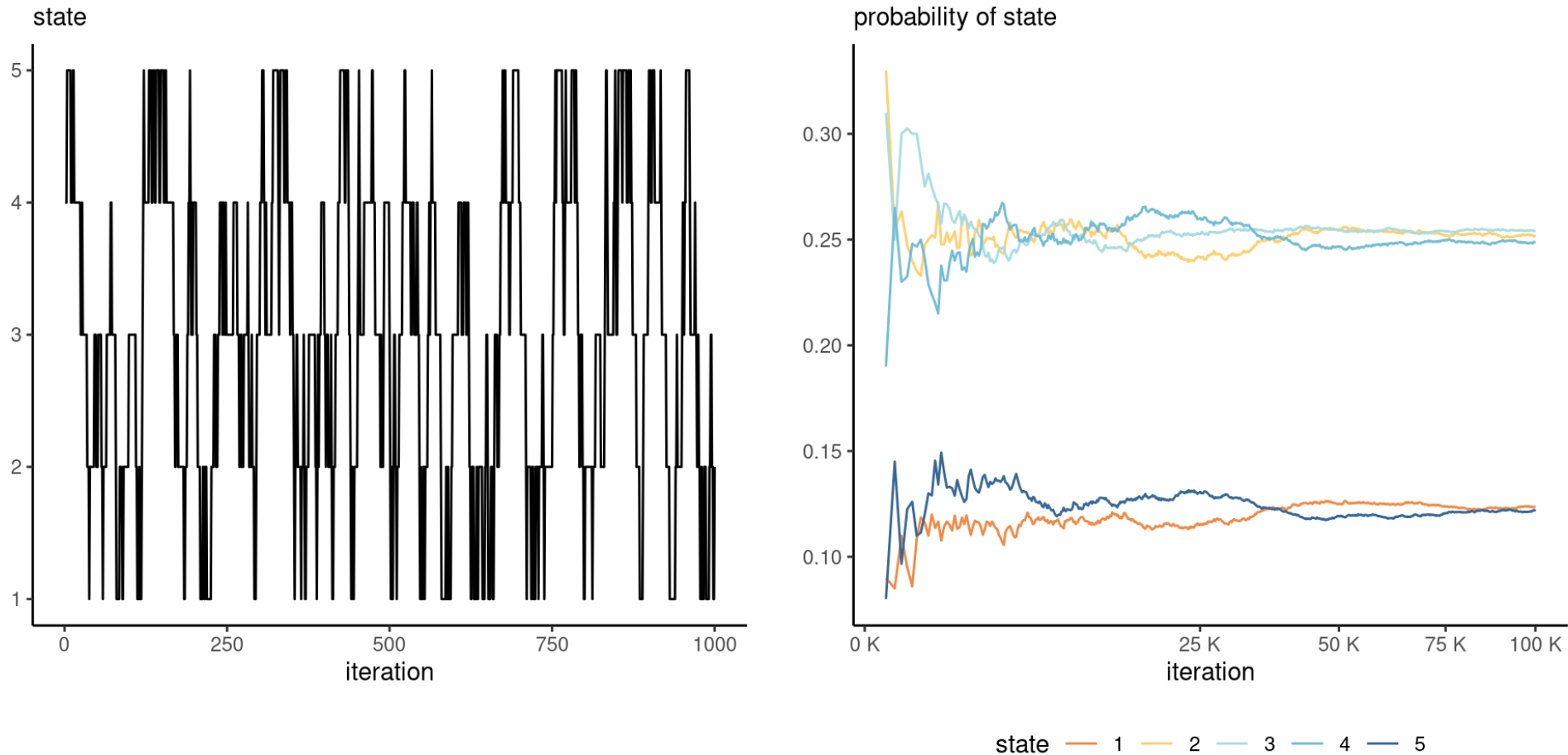


Figure 4: Discrete Markov chain on integers from 1 to 5, with traceplot of 1000 first iterations (left) and running mean plots of sample proportion of each state visited (right).



# Markov chain Monte Carlo

We consider simulating from a distribution with associated density function  $\propto p(\boldsymbol{\theta})$ .

- known up to a normalizing factor not depending on  $\boldsymbol{\theta}$ .

We use  $q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)$  as transition kernel to generate proposals.

# Metropolis–Hastings algorithm

Starting from an initial value  $\theta_0$ :

1. draw a proposal value  $\theta_t^* \sim q(\theta \mid \theta_{t-1})$ .
2. Compute the acceptance ratio

$$R = \frac{p(\theta_t^*)}{p(\theta_{t-1})} \frac{q(\theta_{t-1} \mid \theta_t^*)}{q(\theta_t^* \mid \theta_{t-1})}$$

3. With probability  $\min\{R, 1\}$ , accept the proposal and set  $\theta_t \leftarrow \theta_t^*$ , otherwise set the value to the previous state,  $\theta_t \leftarrow \theta_{t-1}$ .



# Interpretation

- If  $R > 1$ , the proposal has higher density and we always accept the move.
- If we reject the move, the Markov chain stays at the current value, which induces autocorrelation.
- Since the acceptance probability depends only on the density through ratios, normalizing factors of  $p$  and  $q$  cancel out.

## Symmetric proposals and random walk

If the proposal is symmetric, the ratio of proposal densities is

$$q(\boldsymbol{\theta}_{t-1} \mid \boldsymbol{\theta}_t^*) / q(\boldsymbol{\theta}_t^* \mid \boldsymbol{\theta}_{t-1}) = 1.$$

Common examples include random walk proposals

$$\boldsymbol{\theta}_t^* \leftarrow \boldsymbol{\theta}_{t-1} + \tau Z, \quad Z$$

where  $Z$  is a mean zero, variance one random variable.

# Independent proposals

- If we pick instead a global proposal, we must ensure that  $q$  samples in far regions (recall rejection sampling), otherwise ...
- Good proposals include heavy tailed distribution such as Student- $t$  with small degrees of freedom, centered at the maximum a posteriori  $\hat{\boldsymbol{\theta}}$  and with scale matrix  $-\mathbf{H}^{-1}(\boldsymbol{\theta}_t^*)$ , where  $\mathbf{H}(\cdot)$  is the Hessian of the log posterior.

## Upworthy data example

We model the Poisson rates for headlines with questions or not. Our model is *[Math Processing Error]*

# Implementation details: data and containers

In regression models, scale inputs if possible.

```
1 data(upworthy_question, package = "hecbayes")
2 # Compute sufficient statistics
3 data <- upworthy_question |>
4   dplyr::group_by(question) |>
5   dplyr::summarize(ntot = sum(impressions),
6                     y = sum(clicks))
7 # Create containers for MCMC
8 niter <- 1e4L
9 chain <- matrix(0, nrow = niter, ncol = 2L)
10 colnames(chain) <- c("beta", "kappa")
```



# Implementation details: log posterior function

Perform all calculations on the log scale to avoid numerical overflow!

```
1 # Code log posterior as sum of log likelihood and log prior
2 loglik <- function(par, counts = data$y, offset = data$ntot, ...){
3   lambda <- exp(c(par[1] + log(offset[1]), par[1] + par[2] + log(offset[2])))
4   sum(dpois(x = counts, lambda = lambda, log = TRUE))
5 }
6 # Note common signature of function
7 logprior <- function(par, ...){
8   dnorm(x = par[1], mean = log(0.01), sd = 1.5, log = TRUE) +
9     dnorm(x = par[2], log = TRUE)
10 }
11 logpost <- function(par, ...){
12   loglik(par, ...) + logprior(par, ...)
13 }
```

# Implementation details: proposal variance and starting values.

Use good starting values for your Markov chains, such as maximum a posteriori.

```
1 # Compute maximum a posteriori (MAP)
2 map <- optim(
3   par = c(-4, 0.07),
4   fn = logpost,
5   control = list(fnscale = -1),
6   offset = data$ntot,
7   counts = data$y,
8   hessian = TRUE)
9 # Use MAP as starting value
10 cur <- map$par
11 # Compute logpost_cur - we can keep track of this to reduce calculations
12 logpost_cur <- logpost(cur)
13 # Proposal covariance
14 cov_map <- -2*solve(map$hessian)
15 chol <- chol(cov_map)
```

# Implementation details: Metropolis–Hastings algorithm

Use seed for reproducibility, do not compute posterior twice, compute log of acceptance ratio.

```
1  set.seed(80601)
2  naccept <- 0L
3  for(i in seq_len(niter)){
4    # Multivariate normal proposal - symmetric random walk
5    prop <- c(rnorm(n = 2) %*% chol + cur)
6    logpost_prop <- logpost(prop)
7    logR <- logpost_prop - logpost_cur
8    if(logR > -rexp(1)){
9      cur <- prop
10     logpost_cur <- logpost_prop
11     naccept <- naccept + 1L
12   }
13   chain[i,] <- cur
14 }
```

# Implementation details: analysis of output

Need specialized methods to compute standard errors of the posterior mean.

```
1 # Posterior summaries
2 summary(coda::as.mcmc(chain))
3 # Computing standard errors using batch means
4 sqrt(diag(mcmc::olbm(chain, batch.length = niter/40)))
```

1. Empirical mean and standard deviation for each variable, plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
beta	-4.51268	0.001697	1.697e-05	6.176e-05
kappa	0.07075	0.002033	2.033e-05	9.741e-05

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
beta	-4.51591	-4.51385	-4.51273	-4.51154	-4.50929
kappa	0.06673	0.06933	0.07077	0.07212	0.07463

## Standard errors of posterior mean

A method recommended by Geyer (2011) consists in running a long chain and

1. Break the chain of length  $B$  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 global chain mean.

## Pro tips: initial values

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 sampling from the posterior.

This happens often with poor starting value. We are guaranteed to reach stationarity, but this may take more time than the number of steps  $B$  we fixed.

One can discard initial draws during a **burn in** or warmup period, but you better use good starting value.

# Visual diagnostic: trace plots

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

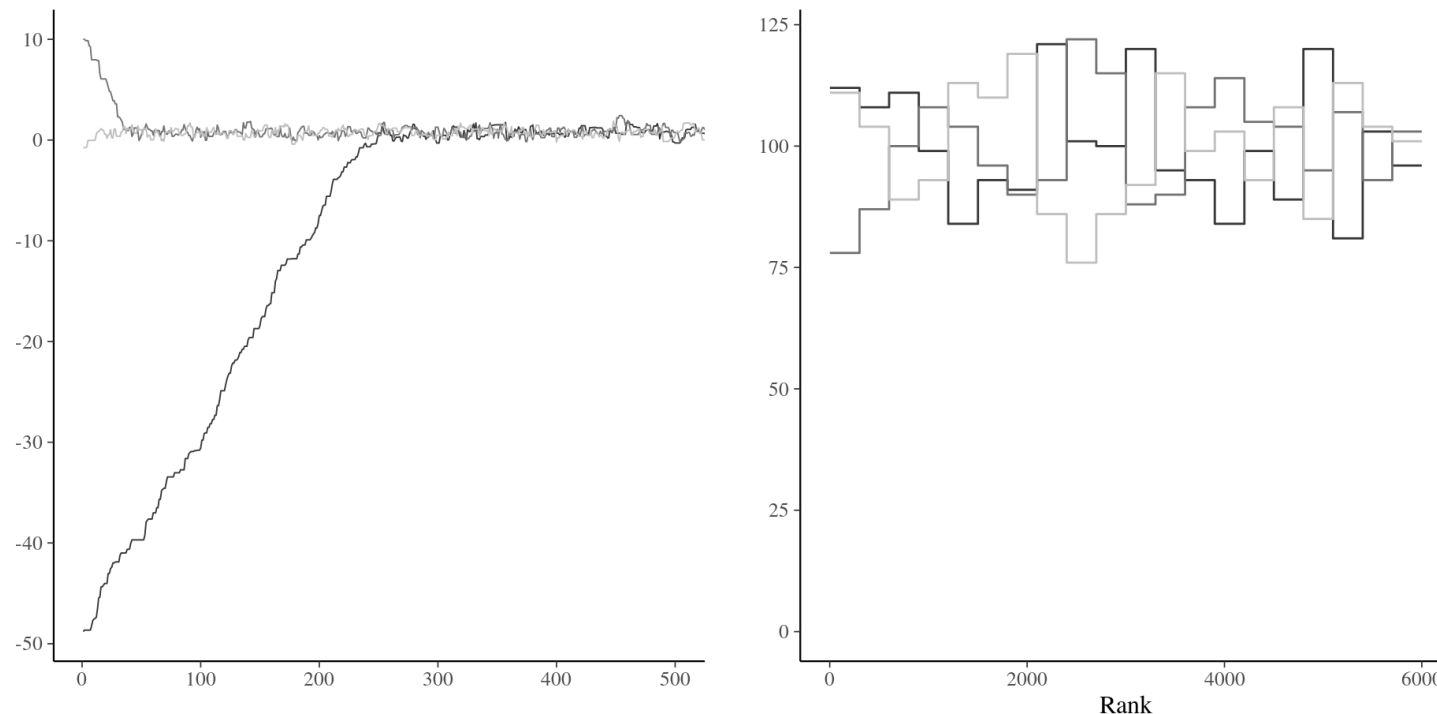


Figure 5: Traceplots of three Markov chains for the same target with different initial values for the first 500 iterations (left) and trace rank plot after discarding these (right).

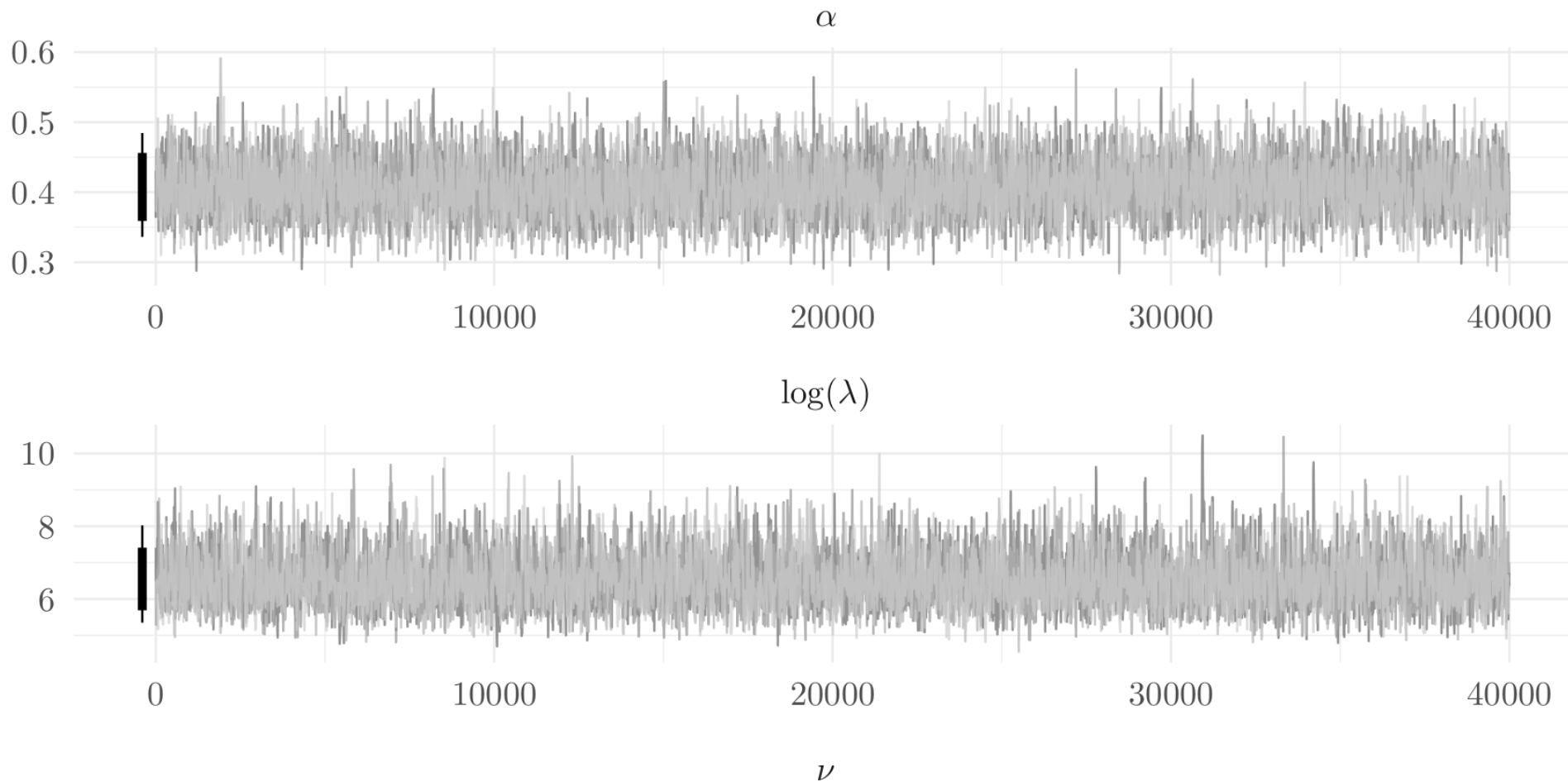
# Checking convergence with multiple chains

- 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.)



# Healthy Markov chains on display

Four chains run in parallel.



# Goldilock principle and proposal variance

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

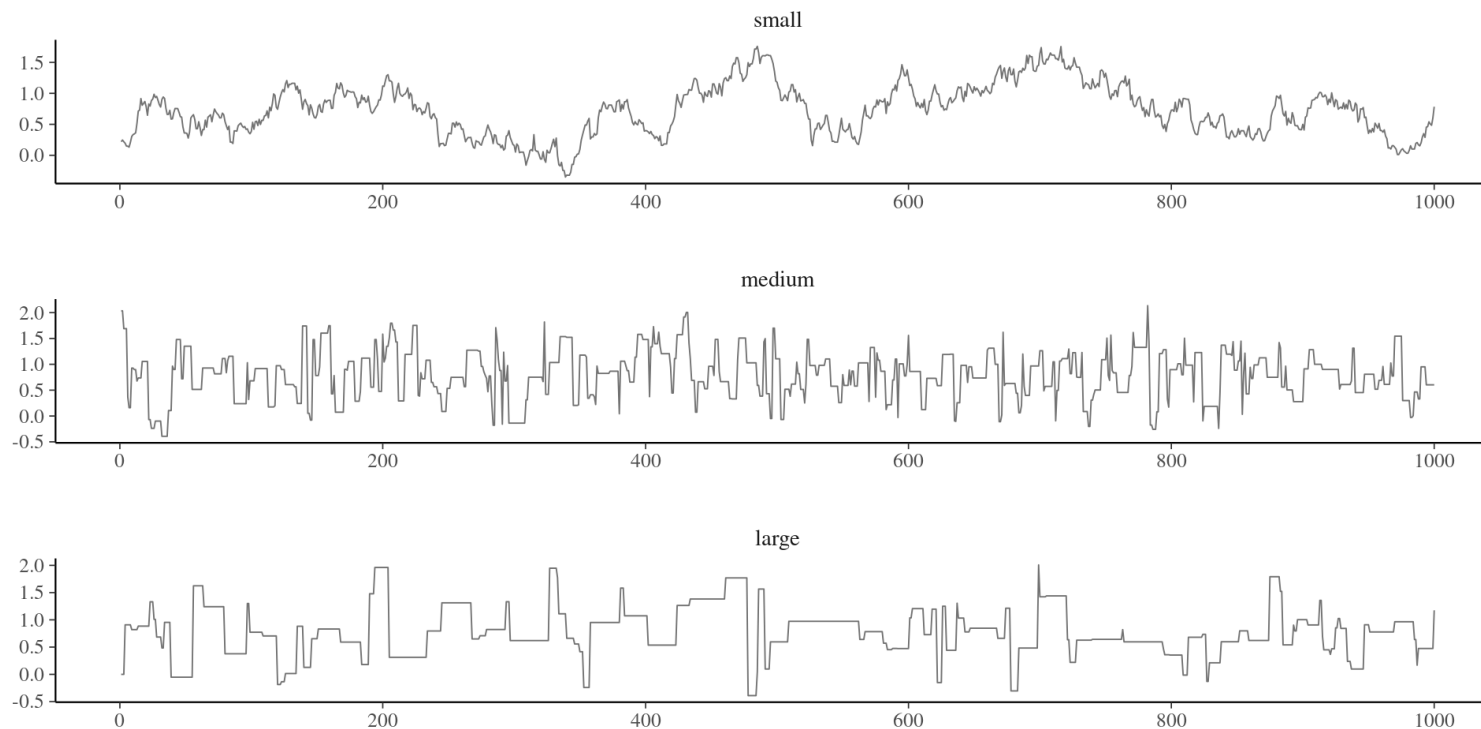


Figure 6: Example of traceplot with proposal variance that is too small (top), adequate (middle) and too large (bottom).

# Correlograms for Goldilock

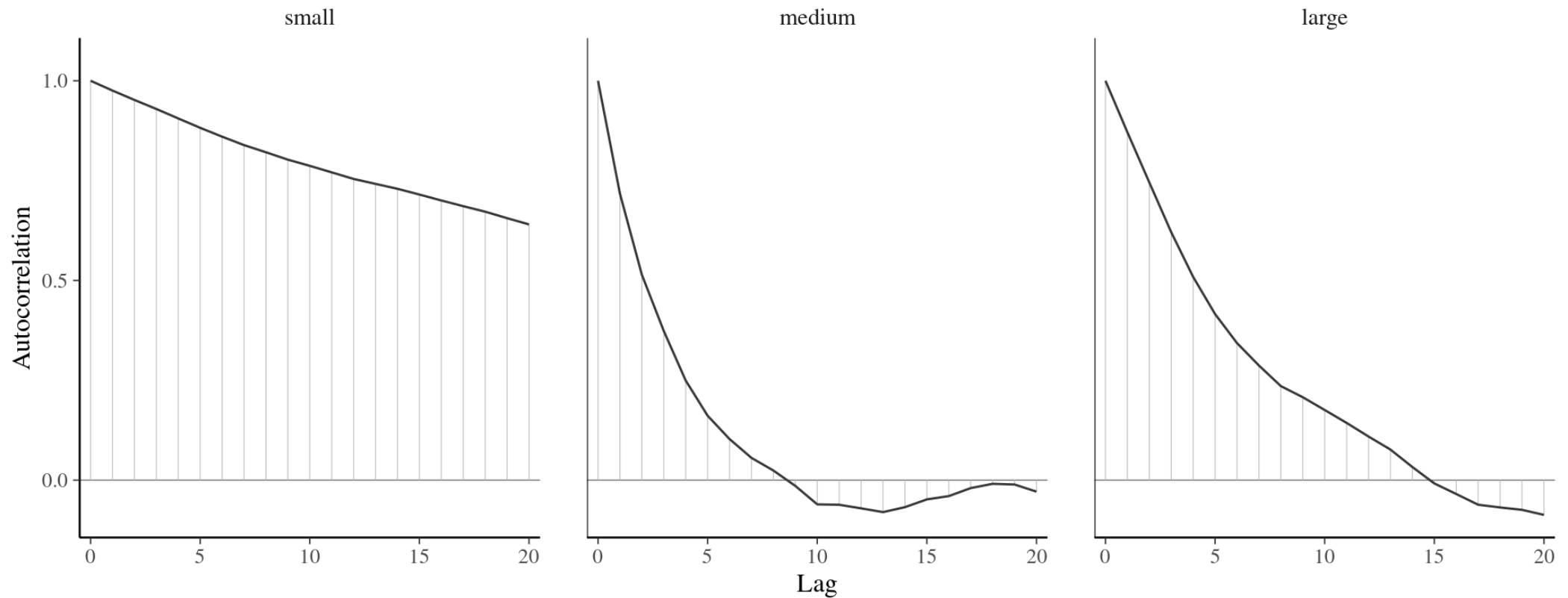


Figure 7: Correlogram for the three Markov chains.

# Tuning Markov chain Monte Carlo

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

## 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
- using orthogonal parametrization whenever possible (reparametrize the model to decorrelate parameters)
- if you cannot, draw correlated proposals (using the chain history to inform 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: need Jacobians!
- Another alternative is to use truncated proposals (especially with more complex algorithms like MALA)

# Efficient proposals: MALA

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

$$\boldsymbol{\theta}_t^* \sim \text{No}\{\mu(\boldsymbol{\theta}_{t-1}), \tau^2 \mathbf{A}\},$$

where

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

and with variance  $\tau^2 \mathbf{A}$ , for some mass matrix  $\mathbf{A}$ , tuning parameter  $\tau > 0$  and a damping factor  $\eta < 1$ .

We need to compute the density for the reverse move with mean  $\mu(\boldsymbol{\theta}_t^*)$ .<sup>1</sup>

1. This is akin to a Newton algorithm, so beware if far from the mode (where gradient can be large)!

# References

- Botev, Z., & L'Écuyer, P. (2017). Simulation from the normal distribution truncated to an interval in the tail. *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, 23–29.  
<https://doi.org/10.4108/eai.25-10-2016.2266879>
- Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer.  
<http://www.nrbook.com/devroye/>
- Geyer, C. J. (2011). Introduction to Markov chain Monte Carlo. In S. Brooks, A. Gelman, G. Jones, & X. L. Meng (Eds.), *Handbook of Markov chain Monte Carlo* (pp. 3–48). CRC Press. <https://doi.org/10.1201/b10905>
- 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>