Chapter 3.4

Diagnosing and improving convergence

Tuning the MCMC algoritm

- MCMC is beautiful because it can handle virtually any statistical model and it is usually pretty easy to write functional code
- However, for hard problems great care must be taken to ensure that the algorithm has converged
- There are three main decisions:
  - Selecting the initial values
  - Determining if/when the chain(s) has converged
  - Selecting the number of samples needed to approximate the posterior

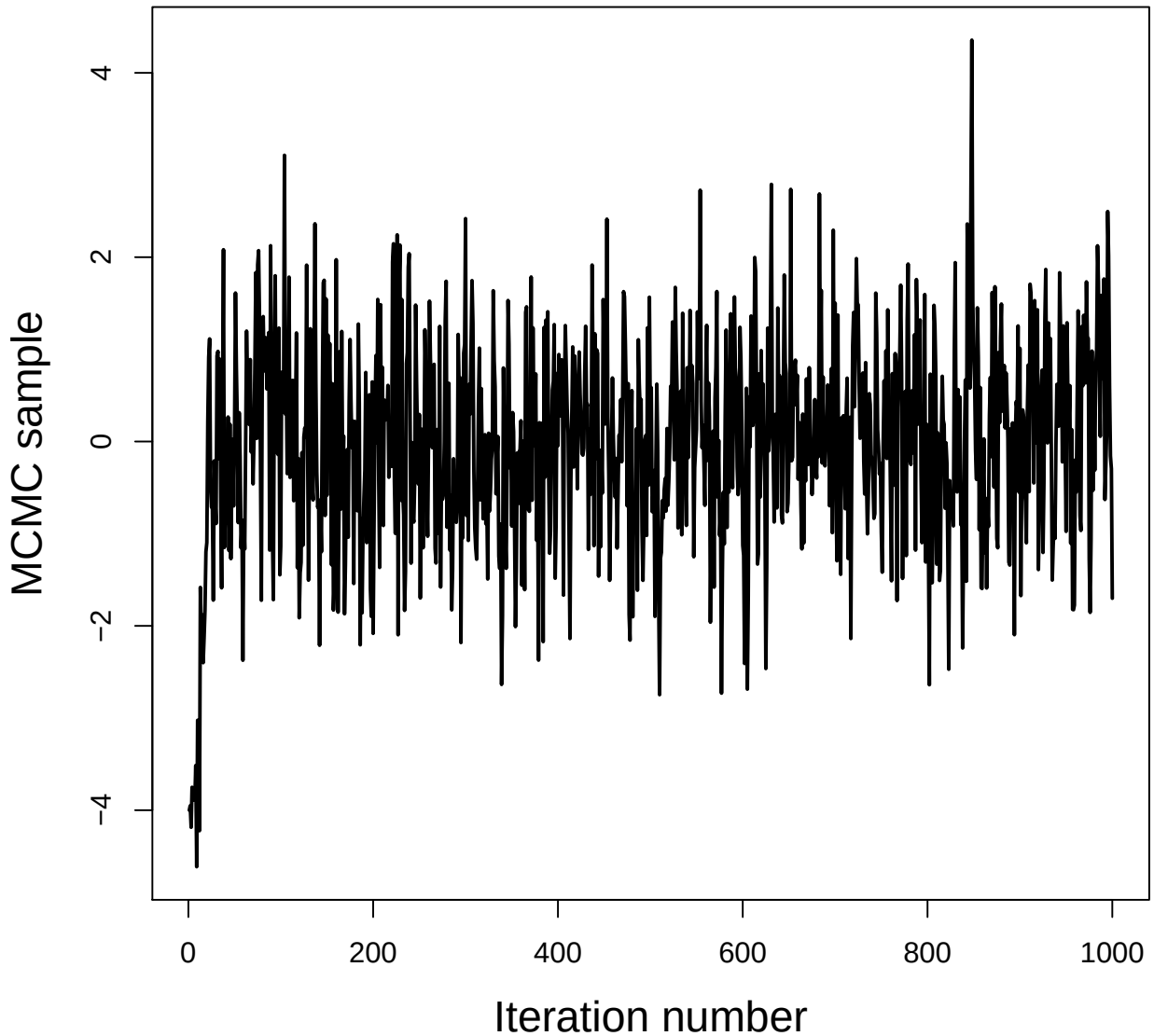
Initial values

- The algorithm will eventually converge no matter what initial values you select
- However taking time to select good initial values will speed up convergence
- It is important to try a few initial values to verify they all give the same result
- Usually 3-5 separate chains is sufficient
- **Option 1**: Select good initial values using method of moments or MLE
- **Option 2**: Purposely pick bad but different initial values for each chain to check convergence

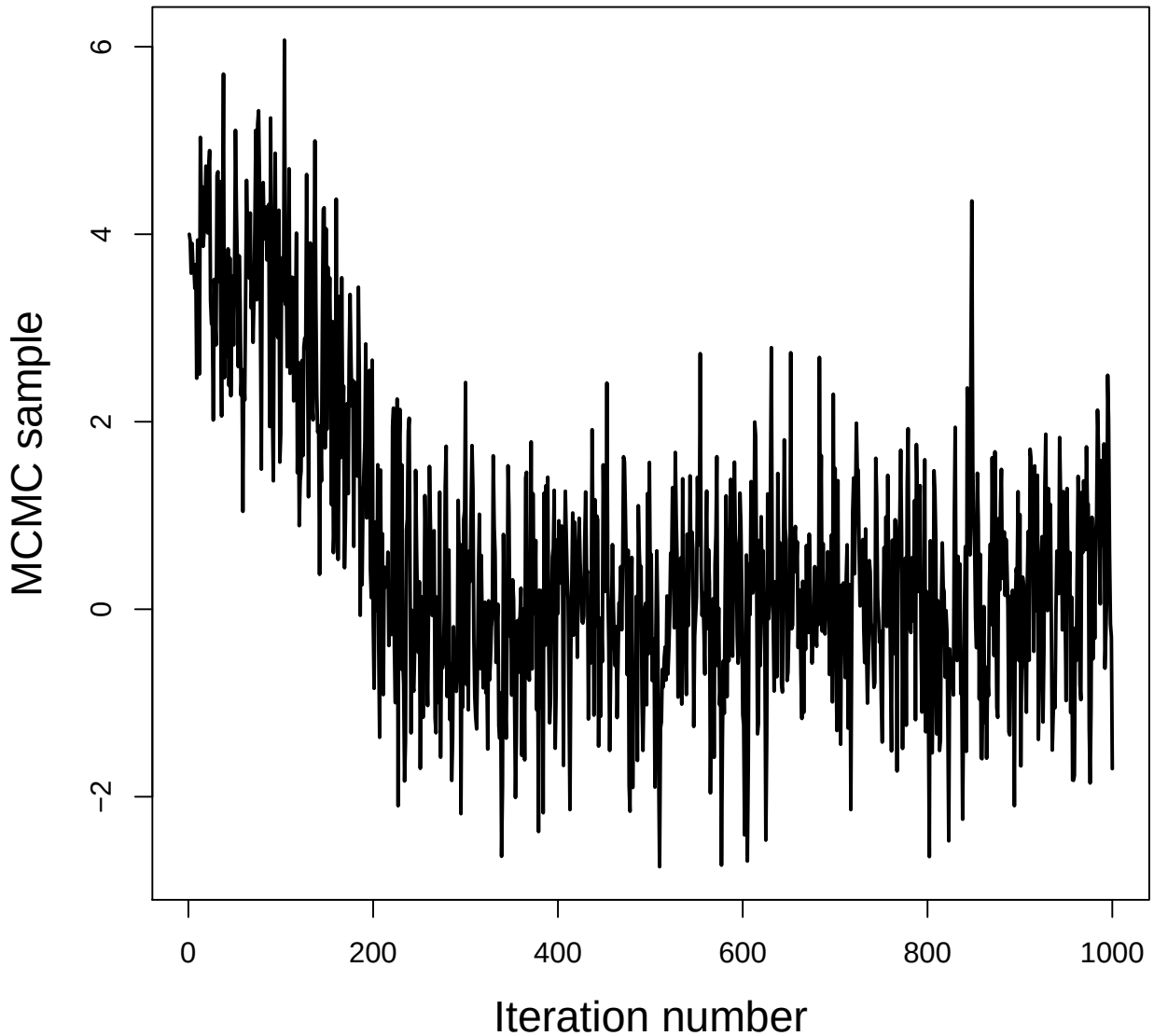
Convergence

- The first few samples are probably not draws from the posterior distribution
- It can take hundreds or even thousands of iterations to move from the initial values to the posterior
- When the sampler reaches the posterior this is called convergence
- Samples before convergence are discard as **burn-in**
- After convergence the samples should not converge to a single point!
- They should be draws from the posterior, and ideally look like a caterpillar or bar code

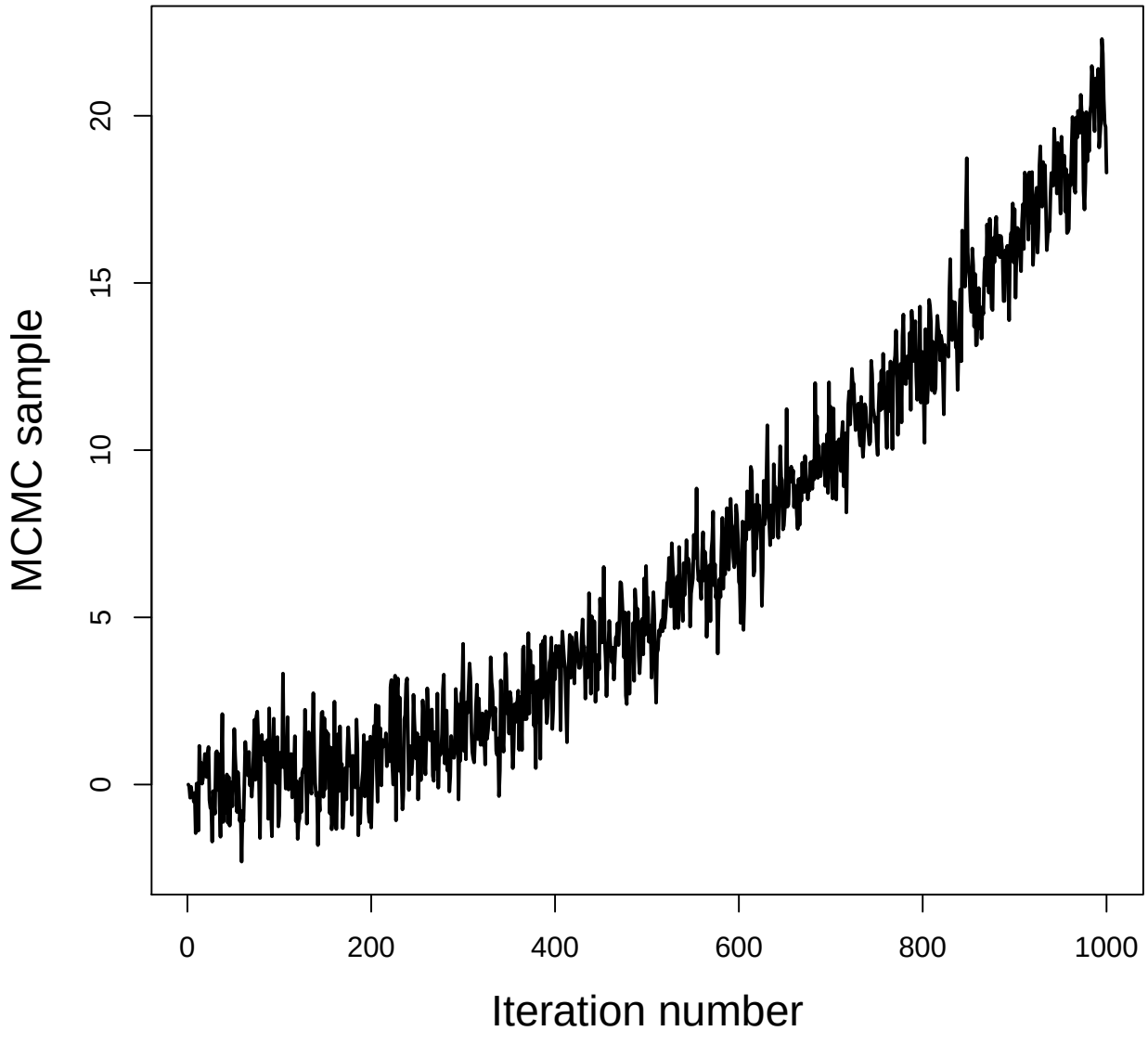
Convergence in a few iterations



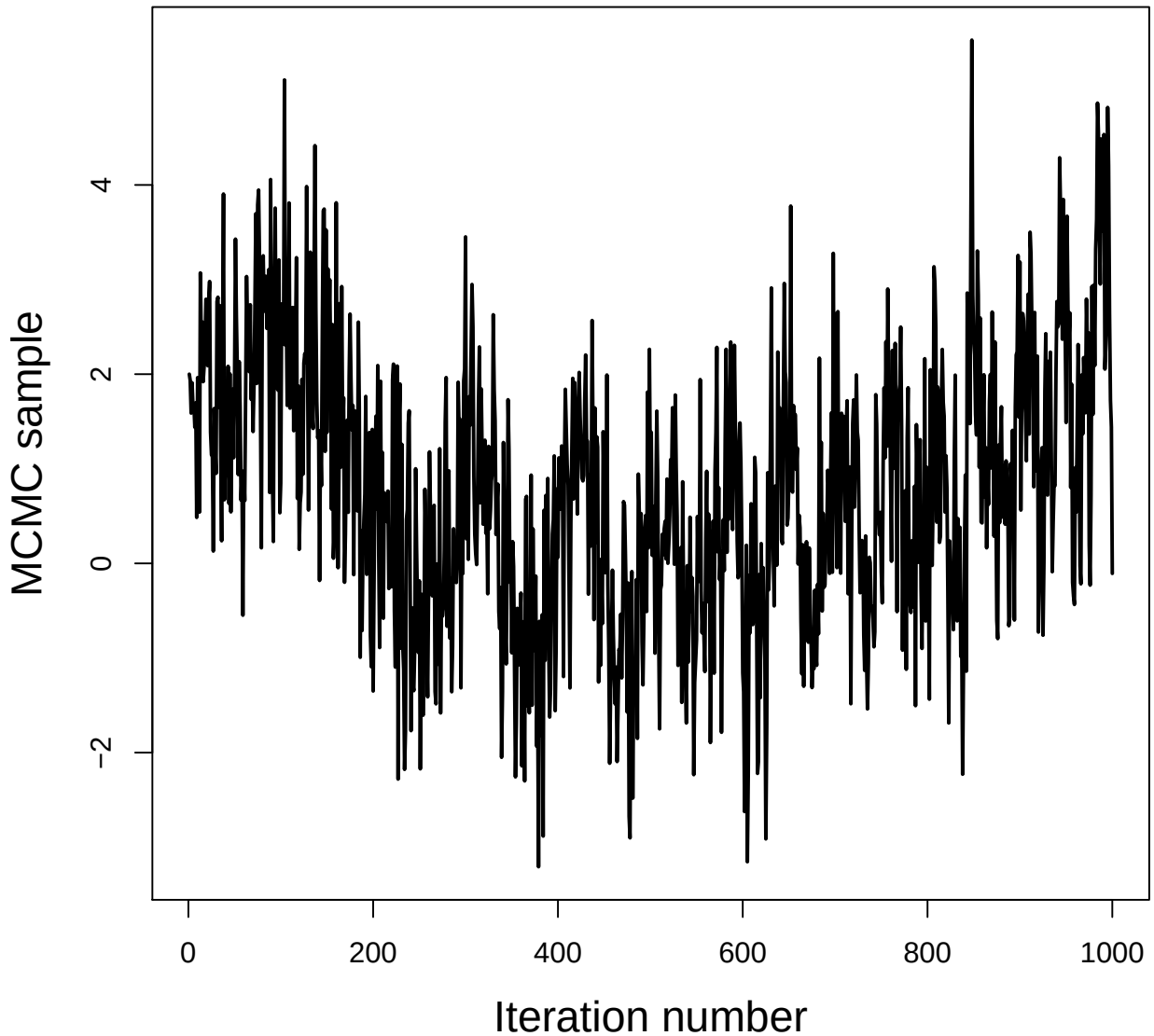
Convergence in a few hundred iterations



This one never converged



Convergence is questionable



Convergence diagnostics

- So far we have visually inspected the chains for convergence
- There are many formal diagnostics
- The `CODA` package in `R` has dozens of diagnostics
- Most give a measure of convergence for each parameter
- Checking convergence using these one-number summaries is more efficient and objective than visual inspection

Convergence diagnostics

- Did my chains converge?
  - Geweke
  - Gelman-Rubin
- Did I run the sampler long enough after convergence?
  - Effective sample size
  - Standard errors for the posterior mean estimate

Examples

- The JAGS function `coda.samples` returns sample is the format that can be passed to the `CODA` function which actually computes the diagnostics

- The course website uses CODA to access convergence for a best-case and a worst-case scenario

Geweke diagnostic

- Compares the mean in the beginning of the chain with the mean at the end of the chain
- Can we used for a single chain
- Done separately for each parameter
- The JAGS default is to compare the first 10% with the last 50%
- The test accounts for autocorrelation
- The test statistic is a z-score, so $|Z| > 2$ indicates poor convergence

Gelman-Rubin statistic

- If we run multiple chains, we hope that all chains give same result
- The Gelman-Rubin statistics measures agreement between chains
- Is it essentially an ANOVA test of whether the chains have the same mean
- It is scaled so that 1 is perfect and 1.1 is decent but not great convergence
- JAGS plots the statistic over iteration
- When the statistic reaches one this indicates convergence

Autocorrelation

- Ideally the samples would be independent across iteration
- The autocorrelation function $\rho(h)$ is the correlation between samples h iterations apart
- JAGS plots the autocorrelation as a function of h
- Lower values are better, but if the chains are long enough even large values can be OK
- **Thinning**: If autocorrelation is zero after lag h you can thin the samples by h to achieve independence
- This is always less efficient than using all samples, but can save memory

Effective sample size

- Highly correlated samples have less information than independent samples
- Say S is the actual number of MCMC samples
- The **effective samples size** is

$$ESS = \frac{S}{1 + 2\sum_{h=1}^{\infty} \rho(h)}$$

- The correlated MCMC sample of length S has the same information as ESS independent samples
- ESS should be at least a few thousand for all parameters

Standard errors of posterior mean estimates

- The sample mean of the MCMC draws is an estimate of the posterior mean
- The standard error of this estimate as another diagnostic
- Assuming independence the standard error is

$$\text{Naive SE} = \frac{s}{\sqrt{S}}$$

where s is the sample SD and S is the number of samples

- A more realistic standard error is

$$\text{Times-series SE} = \frac{s}{\sqrt{ESS}}$$

What to do if the chains haven't converged?

- Determining if chains have converged is not that difficult
- Improving converge is challenging
- We will discuss options in lab
- Hopefully we can get a list of 10 or so

What to do for massive datsets?

- MAP estimation
- Bayesian CLT
- Variational Bayes: Approximates the posterior by assuming the posterior is independent across parameters (fast, but questionable statistical properties)
- Parallel computing: MCMC is inherently sequential, but often some steps can be done in parallel, e.g., onerous likelihood computations
- Divide and Conquer: Split the data into batches and analyze them in parallel, and then carefully combine the result of the batch analyses