STAT8310 - Bayesian Data Analysis

Chi-Kuang Yeh

2026-01-20

Table of contents

Preface

Description

This course will cover the topics in the theory and practice of *Bayesian statistical inference*, ranging from a review of fundamentals to questions of current research interest. Motivation for the Bayesian approach. Bayesian computation, Monte Carlo methods, asymptotics. Model checking and comparison. A selection of examples and issues in modelling and data analysis. Discussion of advantages and difficulties of the Bayesian approach. This course will be computationally intensive through analysis of data sets using the R statistical computing language.

Prerequisites

MATH 4752/6752 – Mathematical Statistics II or equivalent, and the ability to program in a high-level language.

Instructor

Chi-Kuang Yeh, Assistant Professor in the Department of Mathematics and Statistics, Georgia State University.

- Office: Suite 1407, 25 Park Place.
- Email: cyeh@gsu.edu.

Office Hour

10:00–13:00 on Monday, or by appointment.

Grade Distribution

- Homework – 50%
- Exam – 30%
- Final – 20%

Assignment

☐ TBA

Midterm

☐ TBA

Topics and Corresponding Lectures

Those chapters are based on the lecture notes. This part will be updated frequently.

Status	Topic	Lecture Covered
	Welcome and Overview	1
	Intro to R Programming	2
	Probability and Exchangeability	3–

Recommended Textbooks

- Gelman, A., Carlin, J., Stern, H., Rubin, D., Dunson, D., and Vehtari, A. (2021). Bayesian Data Analysis, CRC Press, 3rd Ed.
- Hoff, P.D. (2009). A First Course in Bayesian Statistical Methods, Springer.
- McElreath, R. (2018). Statistical Rethinking: A Bayesian Course with Examples in R and Stan, CRC Press.

Side Readings

- TBA

1 Quick Overview

The posterior distribution is obtained from the prior distribution and sampling model via *Bayes' rule*:

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{\int_{\Theta} p(y \mid \theta')p(\theta')d\theta'}.$$

1.1 Why Bayesian?

- **Intuitive probability interpretation**: Directly quantifies uncertainty about parameters as probability distributions
- **Incorporates prior knowledge**: Systematically combines domain expertise with data through the prior distribution
- **Principled inference**: Bayes' rule provides a coherent framework for updating beliefs based on evidence
- **Natural handling of uncertainty**: Posterior distributions capture full uncertainty, not just point estimates
- **Sequential analysis**: Easily updates beliefs as new data arrives (posterior becomes new prior)
- **Small sample inference**: Performs well with limited data by leveraging prior information
- **Prediction with uncertainty**: Generates predictive distributions that quantify uncertainty in future observations
- **Decision-making**: Naturally incorporates loss functions for optimal decision rules
- **Model comparison**: Bayes factors provide a principled approach to comparing competing models

---

1.2 Some Bayesian Topics and their Computational Focus

Table 1.1: Some of the Bayesian Topics and its computational related focuses.

Topics	**Key Concepts / Readings**	**Computing Focus**
Introduction to Bayesian Thinking	Bayesian vs. Frequentist paradigms; Prior, likelihood, posterior	Review of R basics and reproducible workflows
Bayesian Inference for Simple Models	Conjugate priors, Beta-Binomial, Normal-Normal, Poisson-Gamma	Simulating posteriors, visualization
Prior Elicitation and Sensitivity	Informative vs. noninformative priors, Jeffreys prior	Prior sensitivity plots
Monte Carlo Integration	Law of large numbers, sampling-based inference	Random sampling and Monte Carlo approximation
Markov Chain Monte Carlo (MCMC)	Metropolis-Hastings, Gibbs sampler	Implementing MCMC in R
Convergence Diagnostics	Trace plots, autocorrelation, Gelman–Rubin statistic	`coda`, `rstan`, and `bayesplot` packages
Hierarchical Bayesian Models	Partial pooling, shrinkage, multilevel structures	`rstanarm` / `brms`
Midterm Project: Bayesian Linear Regression	Posterior inference for regression, model selection	`brms`, `rstanarm`, custom Gibbs samplers
Bayesian Model Comparison	Bayes factors, BIC, DIC, WAIC, LOO	Practical comparison via cross-validation
Model Checking and Diagnostics	Posterior predictive checks, residual analysis	`pp_check` in `brms`
Advanced Computation	Hamiltonian Monte Carlo (HMC), Variational Inference	Using `Stan` and `CmdStanR`
Bayesian Decision Theory	Utility functions, decision rules, loss minimization	Simple decision problems in R
Modern Bayesian Methods	Approximate Bayesian computation (ABC), Bayesian neural networks	Examples via `rstan` or `tensorflow-probability`
Student Project Presentations	Applications and case studies	Full workflow demonstration in R

1.3 Interesting Article:

- Goligher, E.C., Harhay, M.O. (2023). What Is the Point of Bayesian Analysis?, American Journal of Respiratory and Critical Care Medicine, 209, 485–487.

2 Belief function and Probability Review

Leading objectives:

be familar with the following concepts

- Belief Functions
- Probability
- Bayes' Rule
- Random Variables
- Exchangeability

2.1 Belief functions

Probability is a way to express rational beliefs.

A **belief function** Be$(\cdot)$ is a function that assigns number to statements such that the large the number, the higher the degree of belief.

Let F, G, and H be three possibly overlapping statements about the world.

For example:

- F = { a person owns a smartphone }
- G = { a person uses social media daily }
- H = { a person works remotely at least part of the time }

or

- F = { a person has a graduate degree }
- G = { a person works in a STEM field }
- H = { a person is employed in the private sector }

The perference over bets involving these statements can be used to define a belief function

- Be$(F) >$ Be(G) means you prefer a bet F is true over that G is true.

Also, we want Be$(\cdot)$ to describe our beliefs under certain conditions

- $\text{Be}(F \mid H) > \text{Be}(G \mid H)$ means that if we knew that H were true, then we would perfer to bet that F is also true over G is also true.
- $\text{Be}(F \mid G) > \text{Be}(F \mid H)$ means that if we bet on F, we would perfer to do it under the condition that G is true rather than H is true.

Some more notations:

- Let $\neg$ denote negation. That is, $\neg F$ is the statement that F is not true.
- Let $F \vee G$ denote the disjunction (or) of statements F and G, meaning that at least one of F or G is true.
- Let $F \wedge G$ denote the conjunction (and) of statements F and G, meaning that both F and G are true.

It has been argued by many that any function that is to numerically represent our beliefs should have the following properties:

- **B1**: $\text{Be}(\neg H \mid H) \leq \text{Be}(F \mid H) \leq \text{Be}(H \mid H)$
- **B2**: $\text{Be}(F \vee G \mid H) \geq \max\{\text{Be}(F \mid H), \text{Be}(G \mid H)\}$
- **B3**: $\text{Be}(F \wedge G \mid H)$ can be derived from $\text{Be}(G \mid H)$ and $\text{Be}(F \mid G \wedge H)$

How should we interpret these properties, and do they make sense?

- B1 means that the number we assign to $\text{Be}(F \mid H)$, our conditional belief in F given H, is bounded below and above by the numbers we assign to complete disbelief $\text{Be}(\neg H \mid H)$ and complete belief $\text{Be}(H \mid H)$.
- B2 says that our belief that the truth lies in a given set of possibilities should not decrease as we add to the set of possibilities.
- B3 is a bit trickier. To see why it makes sense, imagine you have to decide whether or not F and G are true, knowing that H is true. You could do this by first deciding whether or not G is true given H, and if so, then deciding whether or not F is true given G and H.

Recall the notation from (elementary) probability that, $F \cup G$ means F or G, and $F \cap G$ means F and G, and $\emptyset$ is the empty set.

- **P1**:
$$0 = \Pr(\neg H \mid H) \leq \Pr(F \mid H) \leq \Pr(H \mid H) = 1$$
- **P2**:
$$\Pr(F \cup G \mid H) = \Pr(F \mid H) + \Pr(G \mid H), \quad \text{if} \quad F \cap G = \emptyset$$
- **P3**:
$$\Pr(F \cap G \mid H) = \Pr(G \mid H)\Pr(F \mid G \cap H)$$

2.1.1 Conclusion

You can see that, a probability function satisfy P1–P3 also satisfies B1–B3. Therefore, probability functions are a special case of belief functions, and we can use it to describe our belief.

2.2 Events, Partitions and Bayes' Rule

A collectiion of sets $\{H_1, \dots, H_K\}$ is a partition of another set $\mathcal{H}$ if

1. $H_i \cap H_j = \emptyset$ for all $i \neq j$ (mutually exclusive);
2. $\bigcup_{i=1}^{K} H_i = \mathcal{H}$ (collectively exhaustive).

In the context of indetifying which of several statements is true, if $\mathcal{H}$ is the set of all possible truths and $\{H_1, \dots, H_K\}$ is a partition of $\mathcal{H}$, then exactly one set H_j contains the truth.

Let $\mathcal{H}$ be the status of a statistical model.

Valid partitions include:

- {correctly specified, misspecified}
- {underfitting, well-specified, overfitting}

2.2.1 Partition and Probability

Suppose $\{H_1, \dots, H_K\}$ is a partition of $\mathcal{H}$,$\Pr(\mathcal{H}) = 1$ and E is some specific event. Then, by the axioms of probability, we have

- Law of total probability

$$\sum_{k=1}^{K} \Pr(H_k) = \Pr\left(\bigcup_{k=1}^{K} H_k\right) = \Pr(\mathcal{H}) = 1$$

- Law of marginal probability

$$\Pr(E) = \sum_{k=1}^{K} \Pr(E \cap H_k) = \sum_{k=1}^{K} \Pr(E \mid H_k)\Pr(H_k)$$

- Bayes' rule

$$\Pr(H_j \mid E) = \frac{\Pr(E \mid H_j)\Pr(H_j)}{\Pr(E)} = \frac{\Pr(E \mid H_j)\Pr(H_j)}{\sum_{k=1}^{K} \Pr(E \mid H_k)\Pr(H_k)}$$

A subset of the 1996 General Social Survey includes data on the education level and income for a sample of males over 30 years of age. Let {H1,H2,H3,H4} be the events that a randomly selected person in this sample is in, respectively, the lower 25th percentile, the second 25th percentile, the third 25th percentile and the upper 25th percentile in terms of income. By definition,

$$\{\Pr(H_1), \Pr(H_2), \Pr(H_3), \Pr(H_4)\} = \{.25, .25, .25, .25\}.$$

Note that $\{H1, H2, H3, H4\}$ is a partition and so these probabilities sum to 1. Let E be the event that a randomly sampled person from the survey has a college education. From the survey data, we have

$$\{\Pr(E \mid H_1), \Pr(E \mid H_2), \Pr(E \mid H_3), \Pr(E \mid H_4)\} = \{.11, .19, .31, .53\}.$$

These probabilities do not sum to 1 - they represent the proportions of people with college degrees in the four different income subpopulations H_1, H_2, H_3 and H_4. Now let's consider the income distribution of the college-educated population. Using Bayes' rule we can obtain

$\{\Pr(H_1 \mid E), \Pr(H_2 \mid E), \Pr(H_3 \mid E), \Pr(H_4 \mid E)\} = \{0.09, 0.17, 0.27, 0.47\}$, and we see that the income distribution for people in the college-educated population differs markedly from $\{0.25, 0.25, 0.25, 0.25\}$, the distribution for the general population. Note that these probabilities do sum to 1 - they are the conditional probabilities of the events in the partition, given E.

In Bayesian inference, $H_1, ..., H_K$ often refer to disjoint hypotheses or states of nature and E refers to the outcome of a survey, study or experiment. To compare hypotheses *post-experimentally*, we often calculate the following ratio:

$$\begin{aligned}
\frac{\Pr(H_i \mid E)}{\Pr(H_j \mid E)} &= \frac{\Pr(E \mid H_i)\Pr(H_i) / \Pr(E)}{\Pr(E \mid H_j)\Pr(H_j) / \Pr(E)} \\
&= \frac{\Pr(E \mid H_i)\Pr(H_i)}{\Pr(E \mid H_j)\Pr(H_j)} \\
&= \frac{\Pr(E \mid H_i)}{\Pr(E \mid H_j)} \times \frac{\Pr(H_i)}{\Pr(H_j)} \\
&= \text{ "Bayes factor" } \times \text{ "prior beliefs".}
\end{aligned}$$

This calculation reminds us that Bayes' rule does not determine what our *beliefs should be* after seeing the data, it only tells us how they *should change after seeing the data.*

2.3 Independence

Two events F and G are conditionally independent, if given H, we have $\Pr(F \cap G \mid H) = \Pr(F \mid H)\Pr(G \mid H)$.

How do we interpret conditional independence? By Axiom P3, the following is always true:

$$\begin{aligned}
\Pr(G \mid H)\Pr(F \mid H \cap G) &\overset{\text{always}}{=} \Pr(F \cap G \mid H) &&\overset{\text{independence}}{=} \Pr(F \mid H)\Pr(G \mid H) \\
\Pr(G \mid H)\Pr(F \mid H \cap G) &= && \Pr(F \mid H)\Pr(G \mid H) \\
\Pr(F \mid H \cap G) &= && \Pr(F \mid H).
\end{aligned}$$

Thus, conditional independence implies that $\Pr(F \mid H \cap G) = \Pr(F \mid H)$. In other words, if we know H is true, and F and G are conditionally independent given H, then knowing G does not change our belief about F.

Let's consider the conditional dependence of F and G when H is assumed to be true in the following two situations:

Siutation 1:

- F = { a hospital patient is a smoker }
- G = { a hospital patient has lung cancer }
- H = { smoking causes lung cancer}

Situation 2:

- F = { a student studies regularly for an exam }
- G = { a student receives a high exam score }
- H = { studying improves exam performance }

Think: In both of these situations, H being true implies a relationship between F and G. What about when H is not true?

2.4 Random Variables

In Bayesian inference a random variable is defined as an unknown numerical quantity about which we make probability statements. For example, the quantitative outcome of a survey, experiment or study is a random variable before the study is performed. Additionally, a fixed but unknown population parameter is also a random variable

2.4.1 Discrete Ramdom variables

Let Y be a random variable and let $\mathcal{Y}$ be the set of all possible values that Y can take. If $\mathcal{Y}$ is countable, meaning that $\mathcal{Y} = \{y_1, y_2, ...\}$, then Y is a discrete random variable.

The event that the outcome Y of our survey has the value Y is expressed as $\{Y = y\}$. For each $y \in \mathcal{Y}$, the shorthand notation fofr $\Pr(Y = y)$ is $p(y)$, and $p(\cdot)$ is called the **probability mass function** of Y, and with two properties

1. $0 \leq p(y) \leq 1$ for all $y \in \mathcal{Y}$,
2. $\sum_{y \in \mathcal{Y}} p(y) = 1$.

General probability statements about Y can be derived from the pdf/pmf, for example, for any subset $A \subseteq \mathcal{Y}$, we have $\Pr(Y \in A) = \sum_{y \in A} p(y)$. When we have two disjoint subsets A and B of $\mathcal{Y}$, we have

$$\Pr(Y \in A \cup B) = \Pr(Y \in A) + \Pr(Y \in B) = \sum_{y \in A} p(y) + \sum_{y \in B} p(y).$$

Let Y be the number of successes in n independent Bernoulli trials, each with probability of success p. Then, Y follows a Binomial distribution with parameters n and p, denoted as $Y \sim \text{Binomial}(n, p)$. The probability mass function of Y is given by

$$p(y) = \Pr(Y = y) = \binom{n}{y} p^y (1-p)^{n-y}, \quad y = 0, 1, 2, \ldots, n.$$

If $\theta = 0.3$ and $n = 3$, then the probability of observing exactly 2 successes is

$$p(2) = \Pr(Y = 2 \mid \theta = 0.3) = \binom{3}{2} (0.3)^2 (0.7)^1 = 3 \cdot 0.09 \cdot 0.7 = 0.189.$$

2.4.2 Continuous random variables

If $\mathcal{Y}$ is uncountable, for example, $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = (0, 1)$, then Y is a continuous random variable. In this case, we cannot list all possible values of Y and assign probabilities to each value. Instead, we use a probability distribution to describe the distribution of Y. That is, the cummulative distribution function (cdf) defined as follows.

The **cummulative distribution function** (cdf) of a continuous random variable Y is defined as

$$F(y) = \Pr(Y \leq y), \quad y \in \mathcal{Y}.$$

Note that, for the cdf $F(y)$, we have the following properties:

- $0 \leq F(y) \leq 1$ for all $y \in \mathcal{Y}$,
- $F(y)$ is non-decreasing, meaning that if $y_1 < y_2$, then $F(y_1) \leq F(y_2)$,
- $\lim_{y \to -\infty} F(y) = 0$
- $\lim_{y \to \infty} F(y) = 1$.

Probability of various events can be derived from the cdf. For example, for any interval $A = (a, b] \subseteq \mathcal{Y}$, we have

$$\Pr(Y \in A) = \Pr(a < Y \leq b) = F(b) - F(a).$$

Also, $\Pr(Y \leq a) = F(a)$ and $\Pr(Y > a) = 1 - F(a)$.

2.4.3 Description of distributions through quantiles and moments

In this subsection, we discuss a few ways to describe probability distributions: quantiles and moments. They are used to describe the behaviour of the distribution compressing them into summary statistics.

The **expectation** or **mean** of a random variable Y can be thought as the centre of mass or the location of the distribution, which is defined as

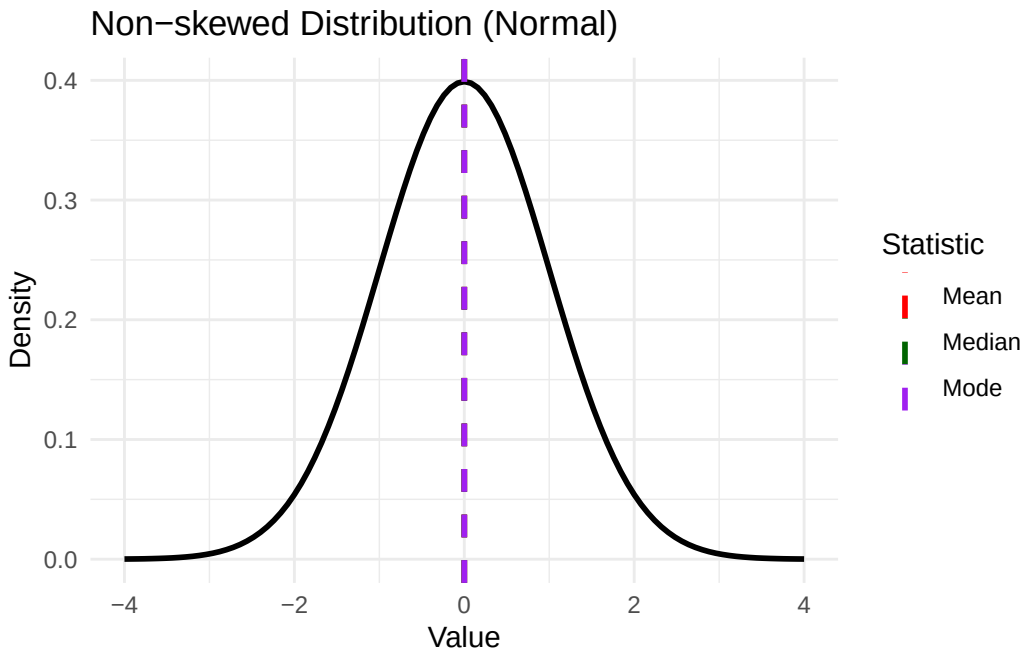
- For discrete random variable:
$$E(Y) = \sum_{y \in \mathcal{Y}} yp(y).$$
- For continuous random variable:
$$E(Y) = \int_{y} yf(y)dy.$$

> **i** Difference bewtwen mean, mode and median
>
> - Mean: the centre of mass of the distribution
> - Mode: The most probable value of Y
> - Median: The value of Y in the middle of the distribution.

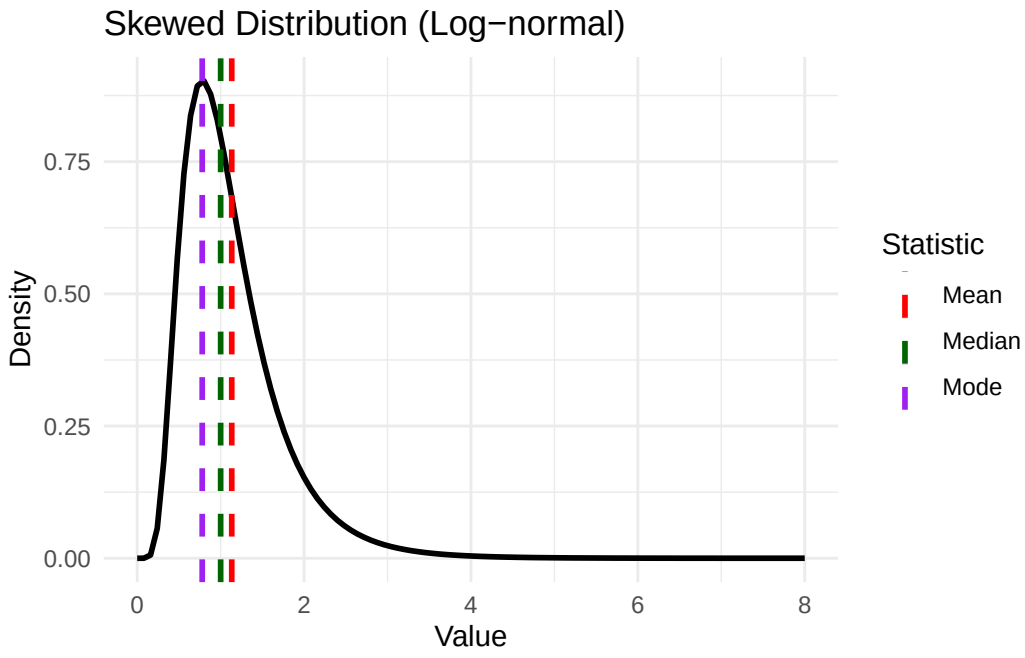
In skewed distribution, the three will not equal to each other.

```
library(ggplot2)

# ------------------------------
# Theoretical reference lines
# ------------------------------
lines_normal <- data.frame(
  value = c(0, 0, 0),
  Statistic = c("Mean", "Median", "Mode")
)

lines_lognormal <- data.frame(
  value = c(exp(1/8), 1, exp(-1/4)),
  Statistic = c("Mean", "Median", "Mode")
)

cols <- c("Mean" = "red", "Median" = "darkgreen", "Mode" = "purple")

# ------------------------------
# Normal distribution
```

```r
# ----------------------------
p1 <- ggplot() +
  stat_function(fun = dnorm, size = 1, color = "black") +
  geom_vline(
    data = lines_normal,
    aes(xintercept = value, color = Statistic),
    linetype = "dashed",
    size = 1
  ) +
  scale_color_manual(values = cols) +
  scale_x_continuous(limits = c(-4, 4)) +
  labs(
    title = "Non-skewed Distribution (Normal)",
    x = "Value", y = "Density", color = "Statistic"
  ) +
  theme_minimal()

# ----------------------------
# Log-normal distribution: LN(0, 0.5)
# ----------------------------
p2 <- ggplot() +
  stat_function(
    fun = function(x) dlnorm(x, meanlog = 0, sdlog = 0.5),
    size = 1,
    color = "black"
  ) +
  geom_vline(
    data = lines_lognormal,
    aes(xintercept = value, color = Statistic),
    linetype = "dashed",
    size = 1
  ) +
  scale_color_manual(values = cols) +
  scale_x_continuous(limits = c(0, 8)) +
  labs(
    title = "Skewed Distribution (Log-normal)",
    x = "Value", y = "Density", color = "Statistic"
  ) +
  theme_minimal()

p1
```


Non−skewed Distribution (Normal)
Density
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
4
Value
Statistic
Mean
Median
Mode

```
p2
```


Skewed Distribution (Log−normal)
Density
0.75
0.50
0.25
0.00
0
2
4
6
8
Value
Statistic
Mean
Median
Mode

Why use mean?

The mean is widely used in statistics and data analysis for several reasons:

1. Mathematical properties: The mean has desirable mathematical properties, such as linearity, which makes it easier to work with in various statistical analyses and models.
2. Sensitivity to all values: The mean takes into account all values in the dataset, providing a comprehensive measure of central tendency. It is also a scaled version of the total, which is often an interest
3. Foundation for other statistical measures: The mean serves as the basis for many other statistical measures, such as variance and standard deviation, which are essential for understanding the spread and variability of data.
4. Mean minimizes the sum of squared deviations: The mean is the value that minimizes the sum of squared deviations (i.e., the expected penalty by choosing one value) from itself, making it a natural choice for summarizing data.
5. May contains full information: In some distributions (e.g., bernoulli distribution), the mean contains all the information about the distribution, making it a sufficient statistic for inference.

The **variance** of a random variable Y measures the spread or dispersion of the distribution, and is defined as

$$\text{Var}(Y) = E\left[(Y - E(Y))^2\right] = E[Y^2] - E^2[Y].$$

The standard deviation is the square root of the variance, denoted as $\text{SD}(Y) = \sqrt{\text{Var}(Y)}$.

The **quantile** of order α of a random variable Y is defined as the value y_α such that

$$\Pr(Y \leq y_\alpha) = F(y_\alpha) = \alpha$$

for $0 < \alpha < 1$.

For example, the median is the quantile of order 0.5, denoted as $y_{0.5}$, which satisfies $\Pr(Y \leq y_{0.5}) = 0.5$. Also, $(y_{0.025}, y_{0.975})$ and $(y_{0.25}, y_{0.75})$ contains 95% and 50% of the mass of the distribution, respectively.

2.5 Joint Disitrubiton

2.5.1 Discrete random variables

Let Y_1 and Y_2 be two random variables with possible values in $\mathcal{Y}_1$ and $\mathcal{Y}_2$, respectively. The **joint distribution** of Y_1 and Y_2 describes the probability of various combinations of values

that (Y_1, Y_2) can take.

Joint beliefs about Y_1 and Y_2 can be represented with probabilities. For example, for subsets $A \subset \mathcal{Y}_1$ and $B \subset \mathcal{Y}_2$, $\Pr(\{Y_1 \in A\} \cap \{Y_2 \in B\})$ represents our belief that Y_1 takes a value in A and Y_2 takes a value in B. The *jointpdf* or *joint density* of Y_1 and Y_2 is defined as

$$p_{Y_1Y_2}(y_1, y_2) = \Pr(\{Y_1 = y_1\} \cap \{Y_2 = y_2\}), \text{ for } y_1 \in \mathcal{Y}_1, y_2 \in \mathcal{Y}_2.$$

The *marginal density* of Y_1 can be computed from the joint density:

$$\begin{aligned} p_{Y_1}(y_1) &\equiv \Pr(Y_1 = y_1) \\ &= \sum_{y_2 \in \mathcal{Y}_2} \Pr(\{Y_1 = y_1\} \cap \{Y_2 = y_2\}) \\ &\equiv \sum_{y_2 \in \mathcal{Y}_2} p_{Y_1Y_2}(y_1, y_2) \end{aligned}$$

The *conditional density* of Y_2 given $\{Y_1 = y_1\}$ can be computed from the joint density and the marginal density:

$$\begin{aligned} p_{Y_2|Y_1}(y_2 \mid y_1) &= \frac{\Pr(\{Y_1 = y_1\} \cap \{Y_2 = y_2\})}{\Pr(Y_1 = y_1)} \\ &= \frac{p_{Y_1Y_2}(y_1, y_2)}{p_{Y_1}(y_1)}. \end{aligned}$$

You should be able to see that

- $\left\{p_{Y_1}, p_{Y_2|Y_1}\right\}$ can be derived from $p_{Y_1Y_2}$,
- $\left\{p_{Y_2}, p_{Y_1|Y_2}\right\}$ can be derived from $p_{Y_1Y_2}$
- $p_{Y_1Y_2}$ can be derived from $\left\{p_{Y_1}, p_{Y_2|Y_1}\right\}$
- $p_{Y_1Y_2}$ can be derived from $\left\{p_{Y_2}, p_{Y_1|Y_2}\right\}$

BUT

- $p_{Y_1Y_2}$ cannot be derived from $\left\{p_{Y_1}, p_{Y_2}\right\}$.

The subscripts of density functions are often dropped, in which case the type of density function is determined by the arguments. For example,

- $p(y_1, y_2) = p_{Y_1Y_2}(y_1, y_2)$ is the joint density of Y_1 and Y_2,
- $p(y_1) = p_{Y_1}(y_1)$ is the marginal density of Y_1
- $p(y_2 \mid y_1) = p_{Y_2|Y_1}(y_2 \mid y_1)$ is the conditional density of Y_2 given $\{Y_1 = y_1\}$, and so on.

Suppose a sociological study reports the following joint distribution of parents' education level and children's income level in a population.

Joint distribution of education and income Suppose a sociological study reports the following **joint distribution** of **parents' education level** and **children's income level** in a population as shown in the Table below

Parent \ Child	Low Income	Middle Income	High Income
High School or Less	0.18	0.22	0.10
College	0.08	0.20	0.12
Graduate School	0.04	0.06	0.10

Suppose we randomly sample a **parent–child pair** from this population.

Let
- Y_1 be the parent's education level
- Y_2 be the child's income level

We are interested in the conditional probability that the child has **high income**, given that the parent has a **college education**.

We may answer this question using the conditional probability formula:

$$\Pr(Y_2 = \text{High Income} \mid Y_1 = \text{College}) = \frac{\Pr(Y_2 = \text{High Income} \cap Y_1 = \text{College})}{\Pr(Y_1 = \text{College})}$$

From the table,

$$\Pr(Y_2 = \text{High Income} \cap Y_1 = \text{College}) = 0.12$$

$$\Pr(Y_1 = \text{College}) = 0.08 + 0.20 + 0.12 = 0.40$$

Therefore,

$$\Pr(Y_2 = \text{High Income} \mid Y_1 = \text{College}) = \frac{0.12}{0.40} = 0.30$$

Thus, our conclusion from the table is, among children whose parents have a college education, **30%** attain high income.

2.5.2 Continuous random variables

Let Y_1 and Y_2 be two continuous random variables with possible values in $\mathcal{Y}_1$ and $\mathcal{Y}_2$, respectively. The **joint distribution** of Y_1 and Y_2 describes the probability of various combinations of values that (Y_1, Y_2) can take. We again work with the cumulative distribution function (cdf). The defintion is given as follows.

Given a continuous joint cdf $F_{Y_1Y_2}(y_1, y_2)$, there is a function p_{Y_1,Y_2} such that

$$F_{Y_1,Y_2}(a, b) = \int_{-\infty}^{a} \int_{-\infty}^{b} p_{Y_1,Y_2}(y_1, y_2) dy_2 dy_1,$$

and $p_{Y_1,Y_2}(y_1, y_2)$ is called the *joint density function* of Y_1 and Y_2.

Similar to the discrete case, we can derive marginal and conditional densities from the joint density as

- Marginal density of Y_1:
$$p_{Y_1}(y_1) = \int_{y_2} p_{Y_1,Y_2}(y_1, y_2) dy_2,$$
- Conditional density of Y_2 given $\{Y_1 = y_1\}$:
$$p_{Y_2|Y_1}(y_2 \mid y_1) = \frac{p_{Y_1,Y_2}(y_1, y_2)}{p_{Y_1}(y_1)}.$$

Think about why $p_{Y_2|Y_1}(y_2 \mid y_1)$ is an actual pdf.

2.5.3 Mixed continuous and discrete variables

It is possible to have joint distributions involving both discrete and continuous random variables. For example, let Y_1 be a discrete random variable taking values in $\mathcal{Y}_1$ and Y_2 be a continuous random variable taking values in $\mathcal{Y}_2$. The joint distribution of Y_1 and Y_2 can be described by the joint density function $p_{Y_1,Y_2}(y_1, y_2)$, which gives the probability that Y_1 takes the value y_1 and Y_2 takes a value in an infinitesimal interval around y_2. One such as example is that Y_1 is a binary variable indicating the presence or absence of a disease, and Y_2 is a continuous variable representing the severity of symptoms. Suppose we define

- Marginal density p_{Y_1} from our belief $\Pr(Y_1 = y_1)$
- a conditional density $p_{Y_2|Y_1}$ from $\Pr(Y_2 \leq y_2 \mid Y_1 = y_1) \doteq F_{Y_2|Y_1}(y_2 \mid y_1)$.

Then, the joint density can be derived as

$$p_{Y_1,Y_2}(y_1, y_2) = p_{Y_1}(y_1)p_{Y_2|Y_1}(y_2 \mid y_1),$$

and the probability can be calculated as

$$\Pr(Y_1 \in A, Y_2 \in B) = \int_{y_2 \in B} \left\{ \sum_{y_1 \in A} p_{Y_1,Y_2}(y_1, y_2) \right\} dy_2.$$

2.5.4 Bayes' rule and parameter estimation

Let

- θ: proportion of people in a large population who have a certain charactersitic.
- Y: number of people in a small random sample from the population who have the charactersitic

Then, in this case, we may threat θ as continuous random variable taking values in $\Theta = (0, 1)$, and Y as a discrete random variable taking values in $\mathcal{Y} = \{0, 1, 2, ..., n\}$, where n is the sample size. Bayesian estimation of the parameter θ derives from the calculate of $p(\theta \mid y)$ where y is the obeserxd value of Y. In bayesian, this calculation first requires that we have a joint density $p(y, \theta$ representing our belief about θ and the survey outcome Y. Often, it is natural to cosnt5ruct this joint density from

- $p(\theta)$: our prior belief about θ before seeing the data, and
- $p(y \mid \theta)$: belief about Y given θ, often called the likelihood function.

Once we observed $\{Y = y\}$, we need to compute our updated belief about θ, represented by the **posterior density** $p(\theta \mid y)$ as

$$p(\theta \mid y) = \frac{p(\theta, y)}{p(y)} = \frac{p(y \mid \theta)p(\theta)}{p(y)} = \frac{p(y \mid \theta)p(\theta)}{\int_\Theta p(y \mid \theta)p(\theta)d\theta}.$$

If we have two values θ_1 and θ_2 in Θ that may be true, then the ratio of their posterior densities is given by

$$\frac{p(\theta_1 \mid y)}{p(\theta_2 \mid y)} = \frac{p(y \mid \theta_1)p(\theta_1)/p(y)}{p(y \mid \theta_2)p(\theta_2)/p(y)} = \frac{p(y \mid \theta_1)p(\theta_1)}{p(y \mid \theta_2)p(\theta_2)}.$$

> **Note**
>
> From this calculation, we notice when we are calculating the relative posterior probability between two parameter values **we do not need** calculate $p(y)$ out.

Another way to think about this is, for a function of θ,

$$p(\theta \mid y) \propto p(y \mid \theta)p(\theta).$$

> **Note**
>
> We will see that the numerator is the important part, while the denominator is just a normalizing constant to make sure the posterior density integrates to 1.

2.6 Independence Random Variables

Let $Y_1, \dots, Y_n$ be random variables with joint density $p(y_1, \dots, y_n)$, and θ is the parameter describe the conditions under which the random variables are generated. We say that $Y_1, \dots, Y_n$ are conditionally independent given θ if every collection of n sets $\{A_1, \dots, A_n\}$ satisfies

$$\Pr(Y_1 \in A_1, \dots, Y_n \in A_n \mid \theta) = \prod_{i=1}^{n} \Pr(Y_i \in A_i \mid \theta).$$

If we have independence property, then

$$\Pr(Y_i \in A_i \mid \theta, Y_j \in A_j) = \Pr(Y_i \in A_i \mid \theta),$$

so the conditional indpenddence can be interpreted as meaning that Y_j gives no additional information about Y_i once we know θ. Also, under independence, the joint density can be factorized as

$$p(y_1, \dots, y_n \mid \theta) = \prod_{i=1}^{n} p_{Y_i}(y_i \mid \theta).$$

If the samples are also identically distributed, meaning that each Y_i has the same marginal density $p_Y(y \mid \theta)$, then the joint density can be further simplified as

$$p(y_1, \dots, y_n \mid \theta) = \prod_{i=1}^{n} p_Y(y_i \mid \theta).$$

In this case , we say that $Y_1, \dots, Y_n$ are **independent and identically distributed** (i.i.d.) given θ, with notation

$$Y_1, \dots, Y_n \mid \theta \overset{i.i.d.}{\sim} p_Y(y \mid \theta).$$

2.7 Exchangeability

A sequence of random variables $Y_1, Y_2, \dots, Y_n$ is **exchangeable** if for any permutation π of the indices $\{1, 2, \dots, n\}$, we have

$$p(y_1, y_2, \dots, y_n) = p(y_{\pi(1)}, y_{\pi(2)}, \dots, y_{\pi(n)}).$$

In other words, the joint density of an exchangeable sequence is invariant to the order of the random variables. That is, the labels contains no information about the outcome.

Suppose a factory produces a large batch of items. Each item may be either **defective** or **non-defective**.

Let

$$Y_i = \begin{cases} 1, & \text{if the } i\text{th inspected item is defective,} \\ 0, & \text{otherwise.} \end{cases}$$

We inspect $n = 10$ items chosen at random from the batch and record $Y_1, Y_2, \dots, Y_{10}$.

Consider the following three observed sequences:

1. $p(1, 0, 1, 0, 1, 0, 0, 1, 0, 1)$
2. $p(0, 1, 0, 1, 0, 1, 1, 0, 0, 1)$
3. $p(1, 1, 0, 0, 1, 0, 1, 0, 0, 1)$

Each sequence contains **5 defective items** and **5 non-defective items**.

Question: Is there a reason to assign these three sequences *different probabilities*?

If the inspection order conveys no additional information about quality, then *only the number of defective items matters*, not their positions in the sequence. This motivates the concept of exchangeability.

2.7.1 Independence versus dependence

Consider the probability assignments

$$\begin{cases} \Pr(Y_{10} = 1) = a, \\ \Pr(Y_{10} = 1 \mid Y_1 = \cdots = Y_9 = 1) = b. \end{cases}$$

If $a \neq b$, then Y_{10} is **not independent** of $Y_1, \dots, Y_9$.

However, lack of independence does **not** imply lack of exchangeability.

Question: should we have $a = b$, $a > b$ or $a < b$?

2.7.2 A latent-parameter model

Suppose the defect rate θ of the factory is unknown.

Conditional on θ,

$$Y_1, \ldots, Y_{10} \mid \theta \sim \text{i.i.d. Bernoulli}(\theta).$$

Then

$$\Pr(Y_1 = y_1, \ldots, Y_{10} = y_{10} \mid \theta) = \theta^{\sum y_i}(1-\theta)^{10-\sum y_i}.$$

If our uncertainty about θ is described by a prior distribution $p(\theta)$, the marginal joint distribution is

$$p(y_1, \ldots, y_{10}) = \int \theta^{\sum y_i}(1-\theta)^{10-\sum y_i} p(\theta)\, d\theta.$$

This probability depends **only on the number of defective items**, not their order.

Thus, we have exchangeability, even though the Y_i are not independent under this model of belief.

Conditional i.i.d. given a latent parameter implies marginal exchangeability. That is, if $\theta \sim p(\theta)$ and $Y_1, \ldots, Y_n$ are conditionally i.i.d. given θ, then $Y_1, \ldots, Y_n$ (i.e., unconditional on θ) are exchangeable.

For the Proof, see page 28 in Hopf (2009).

2.8 de Finetti's Theorem

As of now, we have seen that conditional i.i.d. given a latent parameter implies marginal exchangeability. For example,

$$\begin{cases} Y_1, \ldots, Y_n \mid \theta \overset{i.i.d.}{\sim} \\ \theta \sim p(\theta) \end{cases} \implies Y_1, \ldots, Y_n \text{ are exchangeable.}$$

The converse is also true, as stated in de Finetti's theorem.

Let $Y_i \in \mathcal{Y}$ for all $i \in \{1, 2, \dots, n\}$ be an exchangeable sequence of random variables. Then, there exists a parameter space Θ and a prior distribution $p(\theta)$ on Θ such that the joint distribution of $Y_1, \dots, Y_n$ can be represented as

$$p(y_1, \dots, y_n) = \int_\Theta \left\{ \prod_{i=1}^{n} p_Y(y_i \mid \theta) \right\} p(\theta) d\theta,$$

where $p_Y(y \mid \theta)$ is a probability density function on $\mathcal{Y}$ for each $\theta \in \Theta$. The prior and sampling model depend on the form of the belief model $p(y_1, \dots, y_n)$.

The probability distribution $p(\theta)$ represents our belief about the outcomes $\{Y_1, Y_2, \dots, Y_n\}$, induced by our belief model $p(y_1, \dots, y_n)$. That is,

- $p(\theta)$ represents our belief about $\lim_{n\to\infty} \sum Y_i/n$ in the binary sense
- $p(\theta)$ represents our belief about $\lim_{n\to\infty} \sum (Y_i \le c)/n$ for each c in the general case.

The main idea ofthis and the previous section is as follows

$$\begin{aligned} Y_1, \dots, Y_n \mid \theta &\overset{\text{i.i.d.}}{\sim} p(\cdot \mid \theta), \\ \theta &\sim p(\theta) \end{aligned} \quad \Longleftrightarrow \quad Y_1, \dots, Y_n \text{ are exchangeable for all } n.$$

Question: When is the condition of "exchangeability for all n" reasonable?

- Have exchaneability and repeatability
  - Exchangeability holds if the labels convey no information
  - repeatability hold includes the follows
    1. $Y_1, \dots, Y_n$ are outcomes of a repeartable experiment
    2. $Y_1, \dots, Y_n$ are sampled from a finite population **with repleacement**
    3. $Y_1, \dots, Y_n$ are sampled from an infinite population without replacement.

> **i** In large finite population
>
> Note, if $Y_1, \dots, Y_n$ are exchangeable and sampled from a finite population of size N that is way bigger than n without replacement, then they can be modelled as *approximate* being conditional i.i.d.

This Chapter follows closely with Chapter 2 in Hoff (2009).

3 Week 2 — Conjugate Priors and Analytical Posteriors

3.1 Overview

This week focuses on **conjugate priors** — special priors that yield posteriors in the same family of distributions as the prior.
Students will learn why conjugacy simplifies Bayesian inference, how to identify conjugate pairs for common likelihoods, and how to perform analytical posterior updates without simulation. We will also introduce the concept of prior sensitivity analysis and noninformative (objective) priors.

3.2 Learning Goals

By the end of Week 2, you should be able to:

- Define and identify conjugate priors for standard likelihood models.
- Derive analytical posteriors for Binomial, Poisson, and Normal models.
- Compute posterior summaries and predictive distributions.
- Discuss the influence of priors on posterior inference.
- Perform prior sensitivity analysis in R.

3.3 Lecture 1: The Concept of Conjugacy

3.3.1 1.1 Definition

A **conjugate prior** for a likelihood $p(y \mid \theta)$ is a prior distribution $p(\theta)$ such that the posterior $p(\theta \mid y)$ belongs to the same family as the prior.

Formally:

$$p(\theta \mid y) \propto p(y \mid \theta)\, p(\theta)$$

If $p(\theta \mid y)$ has the same functional form as $p(\theta)$, then $p(\theta)$ is *conjugate* to the likelihood.

3.3.2 1.2 Why Conjugacy Matters

- Provides closed-form expressions for posterior means, variances, and credible intervals.
- Facilitates sequential updating — easy to update priors as new data arrive.
- Useful for educational and analytic illustration before moving to MCMC methods.

3.3.3 1.3 Examples of Conjugate Pairs

Likelihood	Conjugate Prior	Posterior Family
Binomial(n, θ)	Beta(α, β)	Beta$(\alpha + y, \beta + n - y)$
Poisson(λ)	Gamma(a, b)	Gamma$(a + \sum y_i, b + n)$
Normal(μ, σ^2) (known variance)	Normal(μ_0, τ_0^2)	Normal(μ_1, τ_1^2)
Exponential(λ)	Gamma(a, b)	Gamma$(a + n, b + \sum y_i)$
Normal mean/variance (unknown σ^2)	Normal–Inverse-Gamma	Normal–Inverse-Gamma

---

3.4 Lecture 2: Beta–Binomial and Gamma–Poisson Models

3.4.1 2.1 Beta–Binomial Model (Review and Generalization)

Let $y \mid \theta \sim \text{Binomial}(n, \theta)$ and $\theta \sim \text{Beta}(\alpha_0, \beta_0)$.
Then the posterior is:

$$\theta \mid y \sim \text{Beta}(\alpha_0 + y, \beta_0 + n - y).$$

Posterior Mean:

$$E[\theta \mid y] = \frac{\alpha_0 + y}{\alpha_0 + \beta_0 + n}.$$

Predictive Probability for a Future Success:

$$p(\tilde{y} = 1 \mid y) = E[\theta \mid y].$$

Interpretation:
Each observation updates the Beta prior by adding one success or failure to the corresponding shape parameter.

3.4.2 2.2 Gamma–Poisson Model (Counts)

Suppose we model count data as $y_i \sim \text{Poisson}(\lambda)$, with prior $\lambda \sim \text{Gamma}(a_0, b_0)$ (where the Gamma density is parameterized as $p(\lambda) \propto \lambda^{a_0 - 1} e^{-b_0 \lambda}$).

Posterior:

$$\lambda \mid y_1, \ldots, y_n \sim \text{Gamma}\left(a_0 + \sum_{i=1}^{n} y_i,\ b_0 + n\right).$$

Posterior Mean and Variance:

$$E[\lambda \mid y] = \frac{a_0 + \sum y_i}{b_0 + n}, \quad \text{Var}[\lambda \mid y] = \frac{a_0 + \sum y_i}{(b_0 + n)^2}.$$

Posterior Predictive:

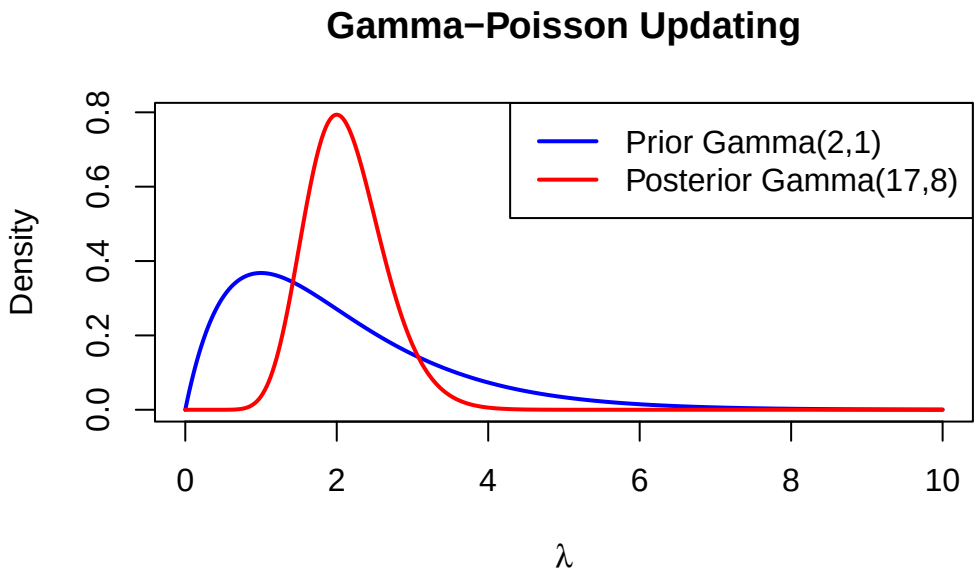
$$p(\tilde{y} \mid y) = \int \text{Poisson}(\tilde{y} \mid \lambda)\, p(\lambda \mid y)\, d\lambda,$$

which follows a **Negative Binomial** distribution.

Interpretation:
The Gamma prior acts as if we had observed $a_0 - 1$ pseudo-events over b_0 pseudo-trials.

3.4.3 2.3 R Example: Gamma–Poisson Updating

```r
# Posterior update for Gamma-Poisson model
y <- c(3, 2, 4, 1, 0, 2, 3)
a0 <- 2; b0 <- 1   # prior Gamma(2,1)
n <- length(y)

a1 <- a0 + sum(y)
b1 <- b0 + n

lambda <- seq(0, 10, length.out = 400)
prior <- dgamma(lambda, a0, b0)
posterior <- dgamma(lambda, a1, b1)

plot(lambda, prior, type="l", lwd=2, col="blue", ylim=c(0, max(posterior)),
     ylab="Density", xlab=expression(lambda),
     main="Gamma-Poisson Updating")
lines(lambda, posterior, col="red", lwd=2)
legend("topright",
       legend=c("Prior Gamma(2,1)", paste0("Posterior Gamma(", a1, ",", b1, ")")),
       col=c("blue", "red"), lwd=2)
```



4 Week 3 — Monte Carlo Integration and Simulation-Based Bayesian Inference

---

4.1 Overview

This week introduces **Monte Carlo methods**, which allow us to approximate Bayesian quantities when analytical solutions are unavailable.
We explore how random sampling can be used to estimate expectations, posterior summaries, and probabilities.
By the end of this week, students will understand how Monte Carlo simulation forms the foundation for modern Bayesian computation such as MCMC.

4.2 Learning Goals

By the end of Week 3, you should be able to:

- Explain the motivation for Monte Carlo methods in Bayesian inference.
- Approximate expectations, integrals, and posterior summaries using random sampling.
- Implement crude Monte Carlo and importance sampling in R.
- Assess the accuracy and variance of Monte Carlo estimators.
- Interpret Monte Carlo errors and convergence diagnostics.

4.3 Lecture 1: Motivation and Fundamentals of Monte Carlo

4.3.1 1.1 The Problem

Bayesian inference often requires evaluating integrals such as:

$$E[h(\theta) \mid y] = \int h(\theta)\, p(\theta \mid y)\, d\theta,$$

which are rarely available in closed form.

4.3.2 1.2 Monte Carlo Idea

If we can sample $\theta^{(1)}, \ldots, \theta^{(M)}$ from the posterior $p(\theta \mid y)$,
then we can approximate the expectation by:

$$\hat{E}[h(\theta)] = \frac{1}{M} \sum_{m=1}^{M} h(\theta^{(m)}).$$

This is called the **Monte Carlo estimator**.

By the **Law of Large Numbers**, $\hat{E}[h(\theta)] \to E[h(\theta)]$ as $M \to \infty$. The **Central Limit Theorem** gives:

$$\sqrt{M}(\hat{E} - E[h(\theta)]) \approx N(0, \mathrm{Var}[h(\theta)]).$$

4.3.3 1.3 Monte Carlo Error

We can estimate the simulation error by:

$$\mathrm{SE}(\hat{E}) \approx \sqrt{\frac{\mathrm{Var}(h(\theta))}{M}}.$$

Larger M gives more accurate approximations but increases computation time.

4.3.4 1.4 Simple Example

Compute $E[\theta]$ for $\theta \sim \mathrm{Beta}(2, 5)$ using Monte Carlo.

```
set.seed(1)
M <- 1e5
theta <- rbeta(M, 2, 5)
mean(theta)          # Monte Carlo estimate
```

```
[1] 0.2861808
```

```
var(theta) / M       # Monte Carlo variance
```

```
[1] 2.56548e-07
```

5 Week 4 — Markov Chain Monte Carlo (MCMC) Methods

5.1 Overview

This week introduces **Markov Chain Monte Carlo (MCMC)** — a powerful class of algorithms for simulating from complex posterior distributions that are difficult to sample from directly.
You will learn the logic of constructing Markov chains with a desired stationary distribution, how to implement the Metropolis–Hastings (MH) and Gibbs samplers, and how to assess convergence and mixing of MCMC chains.

5.2 Learning Goals

By the end of Week 4, you should be able to:

- Explain the intuition behind MCMC and why it works.
- Implement simple Metropolis–Hastings and Gibbs algorithms in R.
- Diagnose convergence using trace plots and summary statistics.
- Compute posterior means, variances, and credible intervals from MCMC samples.
- Understand practical issues such as burn-in, thinning, and autocorrelation.

5.3 Lecture 1: Introduction to MCMC

5.3.1 1.1 Motivation

For many posteriors, sampling directly is infeasible.
We instead build a *Markov chain* whose stationary distribution is the target posterior $p(\theta \mid y)$.
After sufficient iterations, the draws from the chain behave like samples from the true posterior.

5.3.2 1.2 Markov Chain Basics

A Markov chain $\{\theta^{(t)}\}$ has the **Markov property**:

$$p(\theta^{(t)} \mid \theta^{(t-1)}, \dots, \theta^{(1)}) = p(\theta^{(t)} \mid \theta^{(t-1)}).$$

If the chain is **ergodic**, the distribution of $\theta^{(t)}$ converges to a stationary distribution $\pi(\theta)$.

MCMC constructs such chains so that $\pi(\theta) = p(\theta \mid y)$.

5.3.3 1.3 Core Idea

Repeatedly propose a new value θ^* and decide whether to **accept** or **reject** it
based on how likely it is under the posterior.
This ensures that samples eventually represent the posterior distribution.

5.4 Lecture 2: The Metropolis–Hastings Algorithm

5.4.1 2.1 Algorithm Steps

1. Initialize with $\theta^{(0)}$.
2. For each iteration $t = 1, 2, \dots, T$:
   a. Propose $\theta^* \sim q(\theta^* \mid \theta^{(t-1)})$.
   b. Compute the **acceptance probability**:
   $$\alpha = \min\left(1,\ \frac{p(y \mid \theta^*)\ p(\theta^*)\ q(\theta^{(t-1)} \mid \theta^*)}{p(y \mid \theta^{(t-1)})\, p(\theta^{(t-1)})\, q(\theta^* \mid \theta^{(t-1)})}\right).$$
   c. Accept θ^* with probability α; otherwise, keep $\theta^{(t)} = \theta^{(t-1)}$.
3. After burn-in, the samples $\{\theta^{(t)}\}$ approximate draws from $p(\theta \mid y)$.

5.4.2 2.2 Special Case: Symmetric Proposal

If $q(\theta^* \mid \theta^{(t-1)}) = q(\theta^{(t-1)} \mid \theta^*)$, then

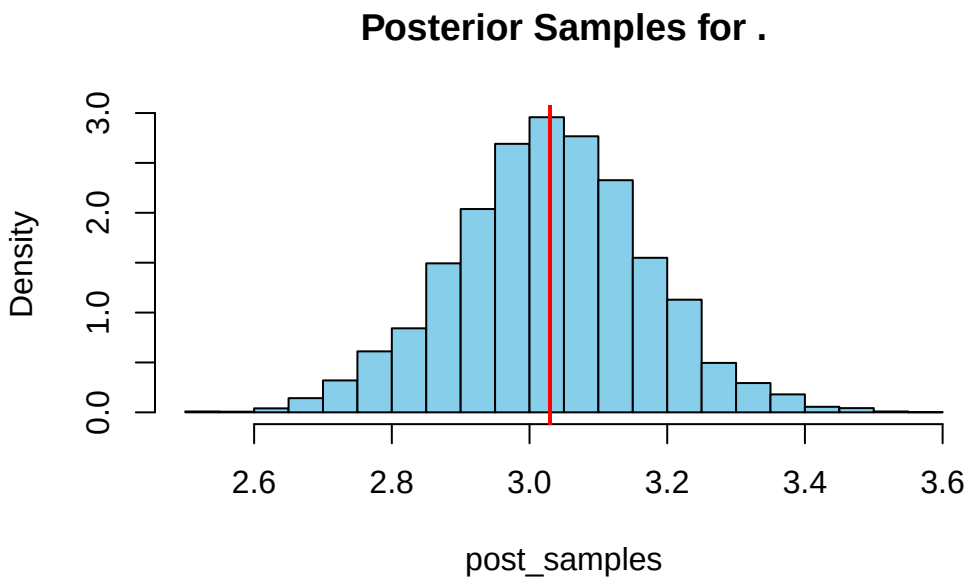
$$\alpha = \min\left(1,\ \frac{p(y \mid \theta^*)\ p(\theta^*)}{p(y \mid \theta^{(t-1)})\, p(\theta^{(t-1)})}\right).$$

This is the **Metropolis algorithm**.

5.4.3 2.3 Example: Posterior for a Normal Mean (Unknown Mean, Known Variance)

Let $y_i \sim N(\mu, 1)$ for $i = 1, \ldots, n$ and prior $\mu \sim N(0, 10^2)$.

```
set.seed(123)
# Data
y <- rnorm(50, mean = 3, sd = 1)
n <- length(y)
post_log <- function(mu) {
  sum(dnorm(y, mu, 1, log=TRUE)) + dnorm(mu, 0, 10, log=TRUE)
}

# Metropolis sampler
T <- 10000
mu <- numeric(T)
mu[1] <- 0
proposal_sd <- 0.5

for(t in 2:T) {
  mu_star <- rnorm(1, mu[t-1], proposal_sd)
  log_alpha <- post_log(mu_star) - post_log(mu[t-1])
  if(log(runif(1)) < log_alpha) mu[t] <- mu_star else mu[t] <- mu[t-1]
}

burnin <- 1000
post_samples <- mu[(burnin+1):T]

hist(post_samples, prob=TRUE, col="skyblue", main="Posterior Samples for ")
abline(v = mean(post_samples), col="red", lwd=2)
```


Posterior Samples for .
Density
0.0
1.0
2.0
3.0
2.6
2.8
3.0
3.2
3.4
3.6
post_samples

6 Week 5 — Model Checking and Comparison

This week introduces methods for evaluating Bayesian model adequacy and comparing models. We focus on **Posterior Predictive Checking (PPC)** and **Bayesian Model Comparison** via Bayes factors, WAIC, and LOO.
Students will diagnose model fit using replicated data and compare predictive accuracy among competing models.

6.1 Learning Goals

By the end of this week, you should be able to:

- Generate and interpret posterior predictive distributions.
- Use posterior predictive checks to detect model misspecification.
- Compute and interpret WAIC, LOO, and Bayes factors.
- Evaluate model adequacy visually and numerically in R.

6.2 Lecture 1 — Posterior Predictive Checking

6.2.1 Posterior Predictive Distribution

For data y and parameters θ,

$$p(\tilde{y} \mid y) = \int p(\tilde{y} \mid \theta)\, p(\theta \mid y)\, d\theta.$$

If a model is adequate, the observed data y should look typical among replicated datasets $\tilde{y}$ simulated from this distribution.

6.2.2 Implementation Steps

1. Choose a **discrepancy statistic** $T(y, \theta)$ capturing an aspect of fit.
2. For each posterior draw $\theta^{(m)}$:
   - Simulate $\tilde{y}^{(m)} \sim p(\tilde{y} \mid \theta^{(m)})$.
   - Compute $T(y, \theta^{(m)})$ and $T(\tilde{y}^{(m)}, \theta^{(m)})$.
3. Compute posterior predictive p-value:

$$p_{\text{ppc}} = P(T(\tilde{y}, \theta) > T(y, \theta) \mid y)\,.$$

Values near 0 or 1 suggest lack of fit.

6.2.3 Example A — Binomial Model

```
set.seed(5)
M <- 5000
y_obs <- 7; n <- 10

theta <- rbeta(M, 2 + y_obs, 2 + n - y_obs)   # posterior draws
y_rep <- rbinom(M, n, theta)

ppc_p <- mean(y_rep >= y_obs)
ppc_p
```

```
[1] 0.509
```

```
hist(y_rep, breaks=seq(-0.5, n+0.5, by=1),
     col="skyblue", main="Posterior Predictive Distribution", xlab="Replicated ỹ")
abline(v=y_obs, col="red", lwd=2)
legend("topright", legend=c("Observed y"), col="red", lwd=2, bty="n")
```

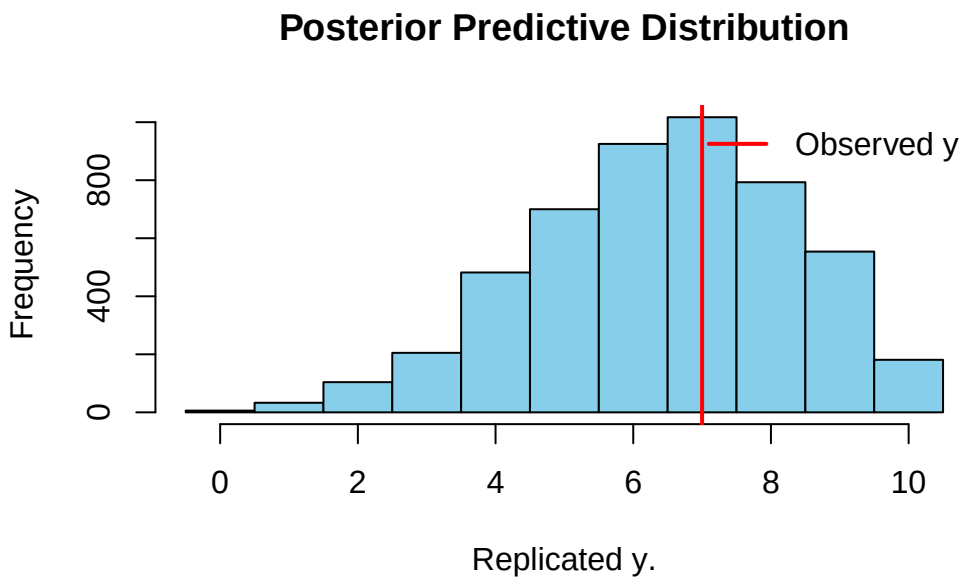


Figure 6.1: Posterior predictive distribution of $\tilde{y}$

6.2.4 Example B — Normal Model (Standard Deviation Check)

```
set.seed(6)
y <- rnorm(30, mean=0, sd=1)
mu_draw <- rnorm(1000, 0, 1)
y_rep_mat <- replicate(200, rnorm(length(y), sample(mu_draw,1), 1))

T_obs <- sd(y)
T_rep <- apply(y_rep_mat, 2, sd)

mean(T_rep >= T_obs)
```

```
[1] 0.145
```

```
hist(T_rep, col="lightgray", main="Posterior Predictive Check: SD",
     xlab="Replicated sd(ỹ)")
abline(v=T_obs, col="red", lwd=2)
legend("topright", legend=c("Observed sd(y)"), col="red", lwd=2, bty="n")
```

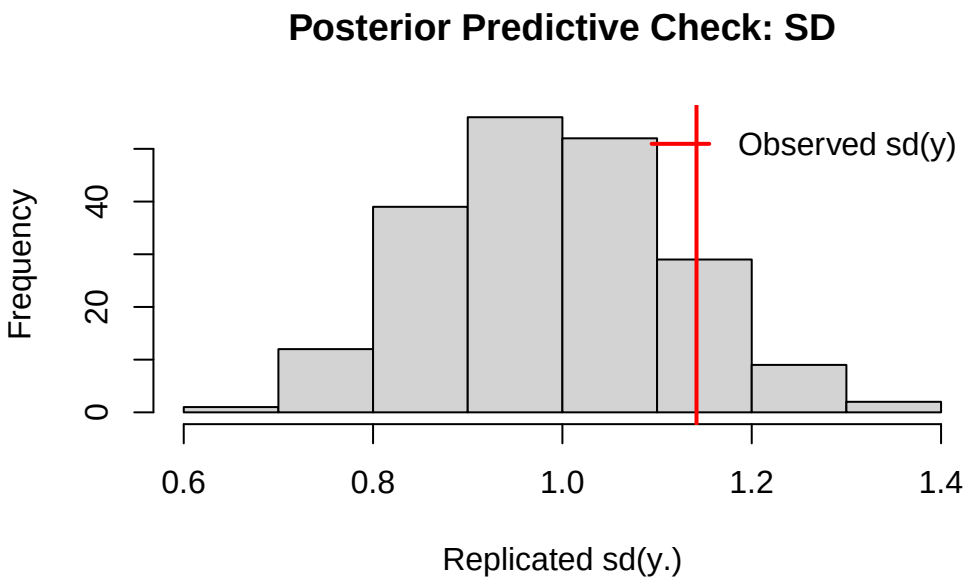


Figure 6.2: PPC for sample standard deviation

6.2.5 Practical Tips

- Use **multiple** statistics $(T_1, T_2, ...)$.
- Visual checks often outperform a single numeric p_{ppc}.
- Integrate PPC with subject-matter knowledge and residual plots.

6.3 Lecture 2 — Bayesian Model Comparison

6.3.1 Motivation

Competing Bayesian models are compared by their fit and complexity. Common approaches include **Bayes factors**, **WAIC**, and **LOO**.

6.3.2 Bayes Factors

Given two models M_1 and M_2,

$$\mathrm{BF}_{12} = \frac{p(y \mid M_1)}{p(y \mid M_2)}, \qquad p(y \mid M) = \int p(y \mid \theta_M, M)\, p(\theta_M \mid M)\, d\theta_M.$$

- $\mathrm{BF}_{12} > 1$ favors M_1.
- Express in $\log_{10}$ scale for interpretation.

$\log_{10} \mathrm{BF}_{12}$	Evidence for M_1
0–0.5	Barely worth mentioning
0.5–1	Substantial
1–2	Strong
>2	Decisive

6.3.3 WAIC and LOO (Predictive Criteria)

When computing marginal likelihoods is infeasible, we use predictive criteria:

- **WAIC (Watanabe–Akaike Information Criterion)**

  $$\mathrm{WAIC} = -2(\mathrm{lppd} - p_{\mathrm{WAIC}}),$$

  where $\mathrm{lppd} = \sum_i \log\Big(\frac{1}{S}\sum_{s=1}^{S} p(y_i \mid \theta^{(s)})\Big)$.

- **LOO (Leave-One-Out Cross-Validation)**
  Approximates the out-of-sample predictive performance:

  $$\mathrm{LOO} = \sum_i \log p(y_i \mid y_{-i}),$$

  usually estimated via Pareto-smoothed importance sampling (PSIS-LOO).

Lower WAIC (or higher elpd_loo) indicates better predictive performance.

6.3.4 Example A — Comparing Two Regression Models with `brms` (optional heavy computation)

```r
# Uncomment and install if needed
# install.packages(c("brms", "loo"))

library(brms)
library(loo)

set.seed(7)
dat <- data.frame(x = rnorm(200))
dat$y <- 2 + 3*dat$x + 1.5*dat$x^2 + rnorm(200, sd = 1)  # true quadratic

# Fit linear vs quadratic models
m1 <- brm(y ~ x, data = dat, family = gaussian(), refresh = 0)
m2 <- brm(y ~ x + I(x^2), data = dat, family = gaussian(), refresh = 0)

loo1 <- loo(m1)
loo2 <- loo(m2)
loo_compare(loo1, loo2)

pp_check(m1)
pp_check(m2)
```

6.3.5 Example B — Quick WAIC Comparison via Frequentist Approximation

```r
set.seed(42)
N <- 150
x <- rnorm(N)
y <- 1.5 + 2.2*x + rnorm(N, sd=1.2)

m1 <- lm(y ~ x)
m2 <- lm(y ~ poly(x, 2, raw = TRUE))

sigma1 <- summary(m1)$sigma
sigma2 <- summary(m2)$sigma

# Pseudo-WAIC: approximate lppd - 2p using residual sum of squares
RSS1 <- sum(resid(m1)^2)
RSS2 <- sum(resid(m2)^2)
```

```r
npar1 <- length(coef(m1))
npar2 <- length(coef(m2))

WAIC1 <- -2 * (sum(dnorm(y, predict(m1), sigma1, log=TRUE)) - npar1)
WAIC2 <- -2 * (sum(dnorm(y, predict(m2), sigma2, log=TRUE)) - npar2)

data.frame(Model=c("Linear","Quadratic"), WAIC=c(WAIC1,WAIC2))
```

```
      Model     WAIC
1    Linear 473.9530
2 Quadratic 475.7823
```

6.3.6 Visual Predictive Comparison

```r
plot(x, y, pch=19, col="#00000055", main="Observed Data with Fitted Models")
xs <- seq(min(x), max(x), length.out=200)
lines(xs, predict(m1, newdata=data.frame(x=xs)), col="steelblue", lwd=2)
lines(xs, predict(m2, newdata=data.frame(x=xs)), col="firebrick", lwd=2)
legend("topleft", lwd=2, col=c("steelblue","firebrick"),
       legend=c("Linear","Quadratic"), bty="n")
```

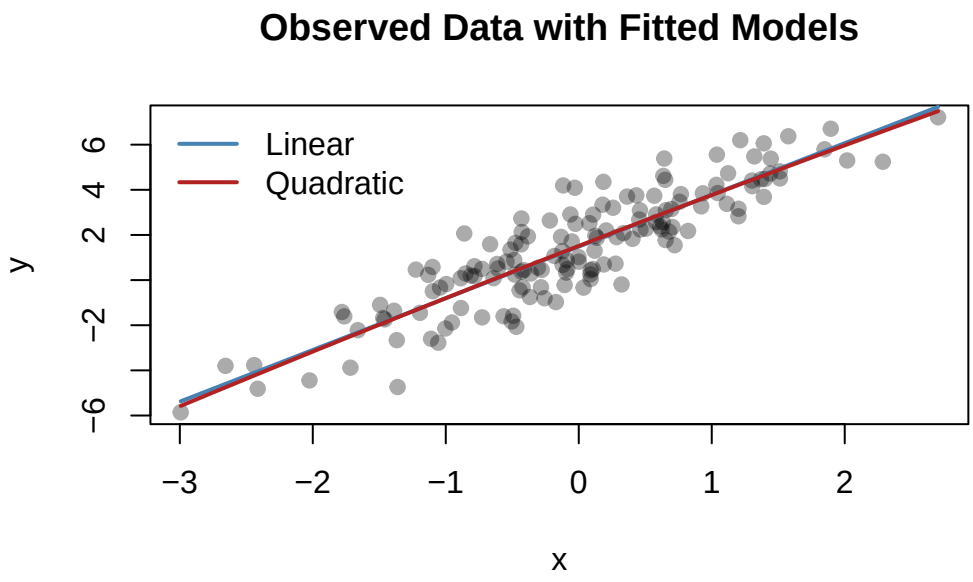


Figure 6.3: Overlay of linear and quadratic model fits

6.3.7 Practical Summary

Method	Strength	Limitation
Posterior Predictive Check	Diagnoses lack of fit to **observed** data	Not inherently comparative
Bayes Factor	Theoretically coherent model evidence	Sensitive to priors; hard integration
WAIC / LOO	Out-of-sample predictive performance	Approximate; needs posterior draws

---

6.4 Lab 5 — Model Checking and Comparison

Objectives

1. Perform PPC using both numerical and visual methods.
2. Compute and interpret WAIC and LOO for model selection.
3. Visualize predictive differences among models.

Packages

`brms`, `loo`, `bayesplot`, `ggplot2`

Tasks

- Fit two Bayesian regression models on the same dataset.
- Conduct posterior predictive checks and compare simulated vs. observed data.
- Compute WAIC and LOO; summarize which model performs better.

6.5 Homework 5

1. **Conceptual**
   - Explain the purpose of posterior predictive checks.
   - Compare WAIC and LOO conceptually.
2. **Computational**
   - Simulate data from a known model. Fit two Bayesian models in R.
   - Use PPC, WAIC, and LOO to assess fit.
   - Discuss how model choice depends on criterion used.
3. **Reflection**
   - Why might visual checks and numerical metrics disagree?
   - Which model would you report and why?

6.6 Key Takeaways

Concept	Summary
Posterior Predictive Check	Compares observed data to replicated draws under the posterior.
Posterior Predictive p-Value	Quantifies fit; extremes suggest model misfit.
WAIC / LOO	Predictive performance measures for Bayesian models.
Bayes Factor	Ratio of marginal likelihoods for model comparison.
Combined Evaluation	Use graphical and numerical criteria together.

Next Week: Hierarchical Bayesian Models — introducing partial pooling and shrinkage.

7 Week 6 — Hierarchical Bayesian Models

This week introduces **hierarchical (multilevel) Bayesian models**, which allow parameters to vary across groups while sharing information through higher-level priors.

We study partial pooling, shrinkage, and their implementation for normal and regression models.

7.1 Learning Goals

By the end of this week, you should be able to:

- Explain the motivation for hierarchical modeling.
- Formulate hierarchical models with group-level parameters.
- Interpret partial pooling and shrinkage.
- Implement a two-level Bayesian model in R using simulation or `brms`.
- Compare complete, no-pooling, and partial-pooling approaches.

7.2 Lecture 1 — Motivation and Structure of Hierarchical Models

7.2.1 1.1 Why Hierarchical Models?

Hierarchical models capture **structured variability** among related groups or units:

- Repeated measures within individuals
- Students within classrooms
- Machines within factories

They balance **within-group** and **between-group** information by introducing group-specific parameters drawn from a common population distribution.

7.2.2 1.2 Model Structure

For group $j = 1, \ldots, J$ and observations $i = 1, \ldots, n_j$:

$$y_{ij} \mid \theta_j, \sigma^2 \sim \mathcal{N}(\theta_j, \sigma^2), \qquad \theta_j \mid \mu, \tau^2 \sim \mathcal{N}(\mu, \tau^2).$$

Top-level priors:

$$\mu \sim \mathcal{N}(0, 10^2), \quad \tau \sim \text{Half-Cauchy}(0, 5).$$

- μ: overall population mean

- τ: between-group standard deviation (pooling strength)
- σ: within-group standard deviation

7.2.3 1.3 Three Extremes of Pooling

Model Type	Description	Behavior
No pooling	Estimate each θ_j separately	Ignores commonality across groups
Complete pooling	Force all groups to share one parameter	Ignores group differences
Partial pooling	Combine information via hierarchical prior	Balances both; default Bayesian choice

---

7.2.4 1.4 Shrinkage Intuition

Posterior for each group mean \$ _j\$ shrinks toward the global mean μ:

$$E[\theta_j \mid y] = w_j \bar{y}_j + (1 - w_j)\mu,$$

where

$$w_j = \frac{n_j/\sigma^2}{n_j/\sigma^2 + 1/\tau^2}.$$

- Large n_j (lots of data): $w_j \to 1 \to$ less shrinkage.
- Small n_j: $w_j \to 0 \to$ stronger shrinkage toward μ.

7.2.5 1.5 Example — Simulated Group Means

```
set.seed(6)
J <- 8; n_j <- rep(10, J)
mu_true <- 5; tau_true <- 2; sigma_true <- 1

theta_true <- rnorm(J, mu_true, tau_true)
y <- sapply(theta_true, function(tj) rnorm(10, tj, sigma_true))
ybar <- colMeans(y)

# No pooling (group means)
no_pool <- ybar

# Complete pooling (global mean)
complete_pool <- mean(y)

# Partial pooling: simple empirical Bayes shrinkage
tau_hat <- sd(ybar)
sigma_hat <- sd(as.vector(y))
w <- (n_j/sigma_hat^2) / (n_j/sigma_hat^2 + 1/tau_hat^2)
partial_pool <- w*ybar + (1-w)*complete_pool

data.frame(Group=1:J,
           ybar=round(ybar,2),
           NoPool=round(no_pool,2),
           Partial=round(partial_pool,2))
```

```
  Group ybar NoPool Partial
1     1 5.53   5.53    5.53
2     2 4.06   4.06    4.21
3     3 6.90   6.90    6.76
4     4 8.19   8.19    7.91
5     5 4.86   4.86    4.93
6     6 5.42   5.42    5.43
7     7 2.20   2.20    2.55
8     8 6.99   6.99    6.84
```

Observe how partial-pool estimates move small-sample groups toward the global mean.

7.2.6 1.6 Advantages of Hierarchical Models

- Borrow strength across groups.
- Naturally incorporate uncertainty at multiple levels.
- Handle unbalanced data and missingness elegantly.
- Allow group-level predictors and complex dependence structures.

7.3 Lecture 2 — Hierarchical Regression and Implementation

7.3.1 2.1 Hierarchical Linear Regression

General form:

$$y_{ij} = \alpha_j + \beta_j x_{ij} + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2),$$
$$\alpha_j \sim \mathcal{N}(\mu_\alpha, \tau_\alpha^2), \quad \beta_j \sim \mathcal{N}(\mu_\beta, \tau_\beta^2).$$

This allows both intercepts and slopes to vary by group.

7.3.2 2.2 Example — Hierarchical Regression with `brms`

```
library(brms)
set.seed(7)

J <- 10
n_j <- 20
group <- rep(1:J, each=n_j)
x <- rnorm(J*n_j, 0, 1)

alpha_true <- rnorm(J, 2, 1)
beta_true  <- rnorm(J, 3, 0.5)
sigma_true <- 0.8
```

```r
y <- alpha_true[group] + beta_true[group]*x + rnorm(J*n_j, 0, sigma_true)
dat <- data.frame(y, x, group=factor(group))

# Hierarchical model (random intercept and slope)
m_hier <- brm(y ~ 1 + x + (1 + x | group),
              data=dat, family=gaussian(),
              chains=2, iter=2000, refresh=0)

summary(m_hier)
plot(m_hier)
```

The `(1 + x | group)` formula defines a **varying intercept and slope** for each group.

7.3.3 2.3 Interpretation

Posterior summaries provide:

- Group-level means α_j, β_j.
- Population-level means μ_α, μ_β.
- Variability estimates τ_α, τ_β showing degree of pooling.

Visualize partial pooling by comparing group-specific fits to the global regression line.

```r
pp_check(m_hier)
plot(conditional_effects(m_hier), points=TRUE)
```

7.3.4 2.4 Practical Considerations

- Choose weakly informative hyperpriors for scale parameters (e.g., Half-Cauchy or Exponential).
- Inspect group-level posterior intervals to assess pooling.
- Center predictors for numerical stability.
- Use hierarchical models as the default when groups share a common process.

7.3.5 2.5 Summary of Hierarchical Modeling Benefits

Feature	Description
Partial pooling	Shares strength across groups while retaining group differences.
Shrinkage	Stabilizes small-sample estimates toward population mean.
Interpretability	Captures multi-level variation naturally.
Predictive accuracy	Usually superior to separate or fully pooled models.

7.4 Homework 6

1. **Conceptual**
   - Explain why hierarchical modeling is often superior to analyzing groups separately.
   - Distinguish between complete pooling, no pooling, and partial pooling.
2. **Computational**
   - Simulate a small dataset with several groups and fit:
     a. Separate regressions (no pooling).
     b. A single pooled regression.
     c. A hierarchical model (partial pooling).
   - Compare estimates for each group and interpret shrinkage behavior.
3. **Reflection**
   - In what situations would you *not* use a hierarchical model?
   - How does the hierarchical prior act as a regularizer?

7.5 Key Takeaways

Concept	Summary
Hierarchical Model	Combines group-level and population-level inference.
Partial Pooling	Balances within- and between-group information.
Shrinkage	Moves noisy group estimates toward a global mean.
Hierarchical Regression	Extends pooling to both intercepts and slopes.
Practical Insight	Default choice when analyzing grouped or multilevel data.

Next Week: Bayesian Decision Theory — introducing utilities, losses, and optimal decision rules under uncertainty.

8 Week 7 — Bayesian Decision Theory

This week introduces the **decision-theoretic foundation** of Bayesian inference.
We study how posterior distributions lead naturally to optimal decisions when losses or utilities are specified, and apply the theory to point estimation and hypothesis testing.

8.1 Learning Goals

By the end of this week, you should be able to:

- Describe the Bayesian decision-theoretic framework.
- Define loss functions and posterior expected loss.
- Derive Bayes rules for common loss functions.
- Apply Bayesian decision principles to estimation and classification.
- Distinguish between point estimation, interval estimation, and decision-making contexts.

8.2 Lecture 1 — Principles of Bayesian Decision Theory

8.2.1 1.1 Motivation

Statistical inference often involves making decisions under uncertainty:
select an action $a$based on observed data y.
Each action has a **loss** (or **utility**) depending on the true parameter value $ $.

8.2.2 1.2 The Decision-Theoretic Setup

- Parameter: $\theta \in \Theta$
- Data: y
- Action space: $\mathcal{A}$
- Loss function: $L(a, \theta)$

After observing y, the Bayesian chooses an action $a(y)$minimizing **posterior expected loss**:

$$\rho(a \mid y) = E[L(a, \theta) \mid y] = \int L(a, \theta)\, p(\theta \mid y)\, d\theta.$$

Bayes rule:

$$a^*(y) = \arg\min_a \rho(a \mid y).$$

8.2.3 1.3 Common Loss Functions and Bayes Rules

Loss Function	Form	Bayes Action
Squared Error	$L(a, \theta) = (a - \theta)^2$	Posterior mean $E[\theta \mid y]$
Absolute Error	$L(a, \theta) = \|a - \theta\|$	Posterior median
0–1 Loss	$L(a, \theta) = \mathbb{1}\{a \neq \theta\}$	Posterior mode (MAP)

These connect the **posterior mean, median, and mode** to optimal decisions under different losses.

8.2.4 1.4 Example — Estimation under Quadratic Loss

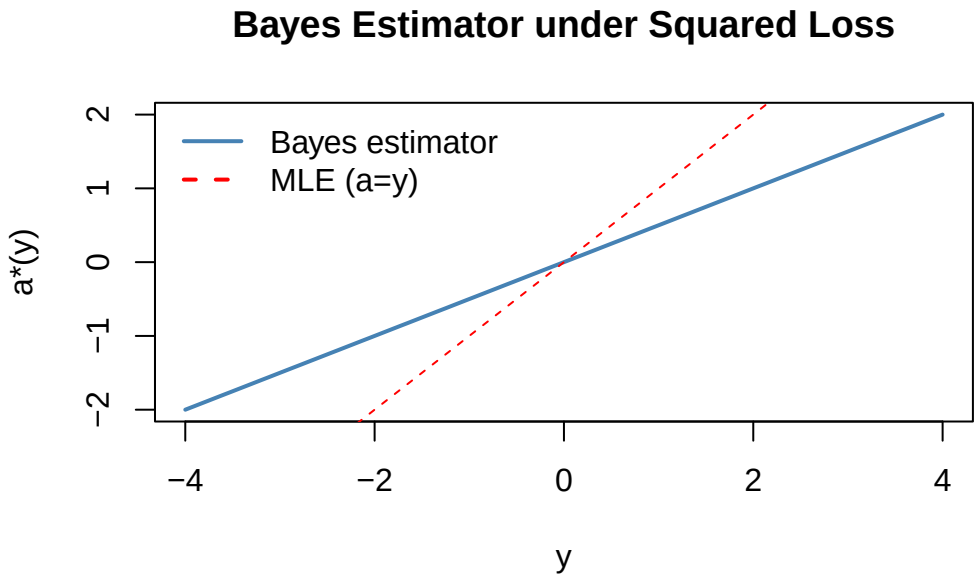
Suppose $y \mid \theta \sim N(\theta, 1)$with prior $\theta \sim N(0, 1)$.

Posterior: $\theta \mid y \sim N\big(\frac{y}{2}, \frac{1}{2}\big)$.

Bayes estimator under squared loss:

$$a^*(y) = E[\theta \mid y] = \frac{y}{2}.$$

```r
set.seed(7)
y <- seq(-4, 4, length=100)
bayes_est <- y/2
plot(y, bayes_est, type="l", lwd=2, col="steelblue",
     main="Bayes Estimator under Squared Loss", xlab="y", ylab="a*(y)")
abline(a=0, b=1, col="red", lty=2)
legend("topleft", legend=c("Bayes estimator","MLE (a=y)"),
       col=c("steelblue","red"), lwd=2, lty=c(1,2), bty="n")
```



Interpretation: The Bayes rule shrinks the estimate toward zero (the prior mean), especially for small |y|.

8.2.5 1.5 Decision Rules and Risk

The **Bayes risk** is the expected loss averaged over data and parameters:

$$r(a) = E[L(a(Y), \Theta)] = \iint L(a(y), \theta)\, p(y, \theta)\, dy\, d\theta.$$

A decision rule minimizing Bayes risk across all priors is **admissible** (cannot be uniformly improved).

8.2.6 1.6 Example — Hypothesis Testing with 0–1 Loss

We test $H_0 : \theta \leq 0$vs $H_1 : \theta\, 0$.

Loss: $L(a, \theta) = \begin{cases} 0 & \text{if correct,} \\ 1 & \text{if wrong.} \end{cases}$

Posterior decision rule:

$$\text{Accept } H_1 \text{ if } P(\theta > 0 \mid y) > 0.5.$$

```
set.seed(8)
theta_draws <- rnorm(5000, mean=1, sd=1)
mean(theta_draws > 0)   # posterior probability of H1
```

```
[1] 0.8406
```

8.3 Lecture 2 — Applications and Extensions

8.3.1 2.1 Bayesian Credible Intervals as Decision Regions

For a given loss that penalizes excluding the true parameter,
a **credible interval** minimizing posterior expected loss corresponds to the shortest interval containing a fixed posterior probability (e.g. 95%).

```
theta_post <- rnorm(5000, mean=2, sd=1)
quantile(theta_post, c(0.025, 0.975))
```

```
       2.5%        97.5%
-0.004383415  4.024907476
```

8.3.2 2.2 Decision Theory for Classification

For a two-class problem with class probabilities $p_1 = P(Y = 1 \mid x)$and $p_0 = 1 - p_1$:
Minimize expected loss $L(a, y)$using a **loss matrix**.

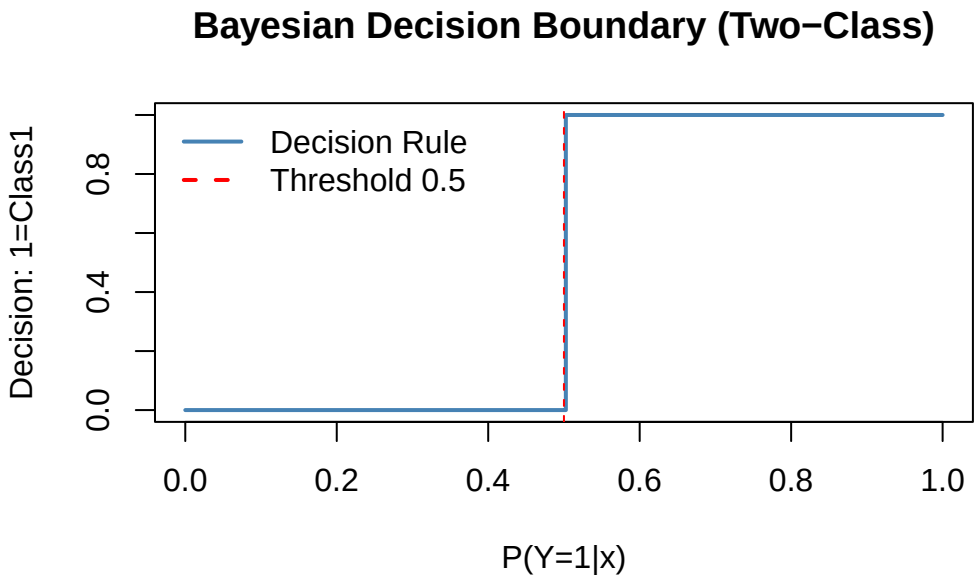
True Class	Predict 0	Predict 1
0	0	c_{10}
1	c_{01}	0

Bayes rule: choose class 1 if

$$\frac{p_1}{p_0} > \frac{c_{10}}{c_{01}}.$$

The usual 0–1 loss corresponds to $c\_10 = c\_01 = 1$, threshold $= 0.5$.

```
p1 <- seq(0,1,length=200)
threshold <- 0.5
plot(p1, ifelse(p1>threshold,1,0), type="s", col="steelblue", lwd=2,
     main="Bayesian Decision Boundary (Two-Class)", xlab="P(Y=1|x)", ylab="Decision: 1=Class
abline(v=threshold, col="red", lty=2)
legend("topleft", legend=c("Decision Rule","Threshold 0.5"),
       col=c("steelblue","red"), lwd=2, lty=c(1,2), bty="n")
```



8.3.3 2.3 Loss vs Utility

Utility $U(a, \theta)$is simply the negative of loss.
Maximizing expected utility is equivalent to minimizing expected loss:

$$a^*(y) = \arg\max_a E[U(a, \theta) \mid y].$$

This framing is often used in economics and decision analysis.

8.3.4 2.4 Connection to Frequentist Estimation

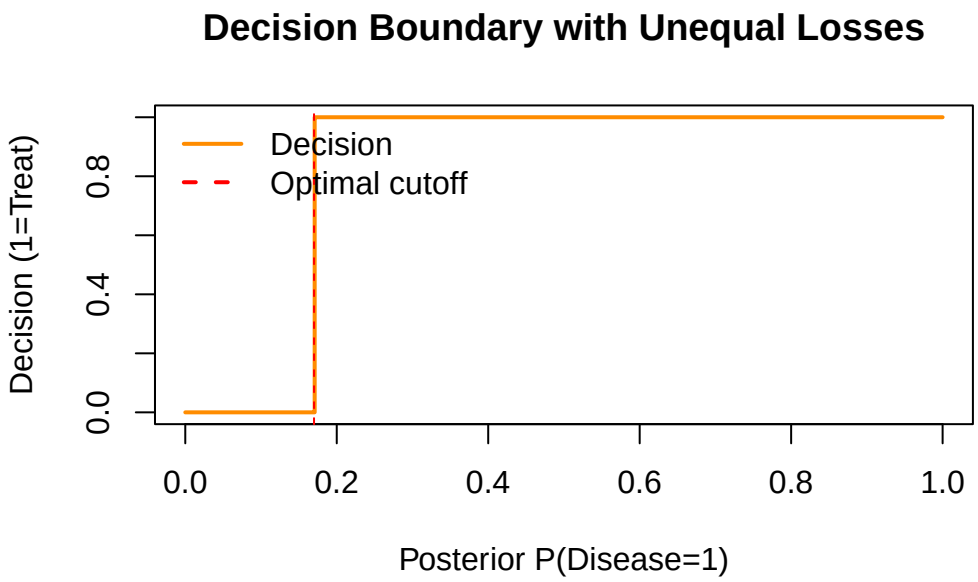
Under certain priors and symmetric losses, Bayes rules coincide with frequentist estimators (e.g. posterior mean = MLE for flat priors).
Bayesian decision theory thus **generalizes** classical estimation.

8.3.5 2.5 Example — Optimal Cutoff for a Diagnostic Test

Let θdenote disease presence (1 = disease).
If false negatives cost 5× more than false positives, the optimal threshold satisfies

$$\frac{p_1}{p_0} > \frac{1}{5} \Rightarrow p_1 > 0.17.$$

```
p <- seq(0,1,length=200)
decision <- ifelse(p > 0.17, 1, 0)
plot(p, decision, type="s", col="darkorange", lwd=2,
     main="Decision Boundary with Unequal Losses", xlab="Posterior P(Disease=1)", ylab="Decis
abline(v=0.17, col="red", lty=2)
legend("topleft", legend=c("Decision","Optimal cutoff"), col=c("darkorange","red"),
       lwd=2, lty=c(1,2), bty="n")
```



8.3.6 2.6 Summary of Bayesian Decision Theory

Concept	Description
Loss function	Quantifies cost of wrong decisions
Posterior expected loss	Average loss given observed data
Bayes rule	Action minimizing posterior expected loss
Common losses	Squared, absolute, 0–1
Applications	Estimation, hypothesis testing, classification, decision support

---

8.4 Homework 7

1. **Conceptual**
   - Define loss and risk in the Bayesian framework.

- What is the relationship between posterior mean, median, and mode under different losses?

2. **Computational**
   - Simulate data from $N(\theta, 1)$with prior $N(0, 1)$.
   - Compute the Bayes estimator under squared loss and compare it with the MLE.
   - Repeat using absolute loss and report the posterior median.
3. **Reflection**
   - How does changing the loss function alter your decision?
   - Give a real-world example where asymmetric losses are important.

8.5 Key Takeaways

Concept	Summary
Decision Theory	Provides a unified framework linking inference to action.
Bayes Rule	Minimizes posterior expected loss.
Common Losses	Squared → mean; absolute → median; 0–1 → mode.
Applications	Estimation, testing, classification, optimal thresholds.
Perspective	Inference as a special case of decision-making under uncertainty.

Next Week: Advanced Bayesian Computation — Hamiltonian Monte Carlo (HMC) and Variational Inference.

9 Week 8 — Advanced Bayesian Computation

This week explores two major developments that enable scalable Bayesian inference for complex or high-dimensional models:
Hamiltonian Monte Carlo (HMC) and **Variational Inference (VI)**.
We study their principles, intuition, and practical use in software such as `Stan` and `brms`.

9.1 Learning Goals

By the end of this week, you should be able to:

- Explain the motivation for advanced sampling and approximation methods.
- Describe the mechanics and intuition of Hamiltonian Monte Carlo.
- Understand the trade-offs between exact (MCMC) and approximate (VI) inference.
- Run basic HMC and VI fits using modern R interfaces.
- Interpret diagnostics for both approaches.

9.2 Lecture 1 — Hamiltonian Monte Carlo (HMC)

9.2.1 1.1 Motivation

Traditional MCMC (e.g., Metropolis–Hastings, Gibbs) can mix slowly in high dimensions.
Hamiltonian Monte Carlo accelerates exploration by using gradient information from the log posterior to simulate physical motion through parameter space.

9.2.2 1.2 Hamiltonian Dynamics

We introduce an auxiliary "momentum" variable p and define the **Hamiltonian**:

$$H(\theta, p) = U(\theta) + K(p),$$

where
- $U(\theta) = -\log p(\theta \mid y)$ (potential energy = negative log posterior),
- $K(p) = \frac{1}{2} p^\top M^{-1} p$ (kinetic energy, with mass matrix M).

The system evolves via Hamilton's equations:

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial p}, \qquad \frac{dp}{dt} = -\frac{\partial H}{\partial \theta}.$$

9.2.3 1.3 Leapfrog Integration

To approximate continuous motion, HMC uses a **leapfrog integrator** with step size ϵ and L steps:

1. $p\_t + \epsilon/2 = p_t - \frac{\epsilon}{2} \nabla_\theta U(\theta_t)$
2. $\theta\_t + \epsilon = \theta_t + \epsilon M^{-1} p_{t+\epsilon/2}$
3. $p_{t+\epsilon} = p_{t+\epsilon/2} - \frac{\epsilon}{2} \nabla_\theta U(\theta_{t+\epsilon})$

After simulating this path, we apply a **Metropolis acceptance step** using the change in H.

9.2.4 Intuition

- The gradient $\nabla_\theta U(\theta)$ guides proposals along high-density regions, avoiding random walk behavior.
- Proper tuning of step size ϵ and number of steps L yields efficient exploration.
- Modern implementations (e.g., *Stan*) adapt these automatically via the **No-U-Turn Sampler (NUTS)**.

9.2.5 1.5 Example — Logistic Regression with HMC (Stan)

```
library(brms)
set.seed(8)

# Simulated logistic data
N <- 200
x <- rnorm(N)
y <- rbinom(N, 1, plogis(-1 + 2*x))
dat <- data.frame(x, y)

# Fit via Hamiltonian Monte Carlo (NUTS)
fit_hmc <- brm(y ~ x, data=dat, family=bernoulli(), chains=2, iter=2000, refresh=0)
summary(fit_hmc)
plot(fit_hmc)
```

`Stan`'s NUTS algorithm performs automatic adaptation of step size and trajectory length.

9.2.6 1.6 Diagnosing HMC Performance

Key diagnostics: - **Divergent transitions** → numerical instability (reduce step size or re-scale parameters).
- **Energy Bayesian Fraction of Missing Information (E-BFMI)** → low values (< 0.3) indicate poor exploration.
- $\widehat{R}$ and effective sample size → check convergence and mixing.

```
library(bayesplot)
mcmc_nuts_divergence(fit_hmc)
mcmc_trace(fit_hmc, pars=c("b_Intercept","b_x"))
```

9.2.7 1.7 Advantages of HMC

Feature	Benefit
Gradient-based proposals	Rapid movement through high-density regions
Higher acceptance rates	Fewer rejections than random-walk MH
Fewer tuning parameters	Automatic adaptation (NUTS)
Robust for high-dimensional models	Used in most modern Bayesian software

---

9.3 Lecture 2 — Variational Inference (VI)

9.3.1 2.1 Motivation

When exact sampling is too costly (e.g., massive datasets, deep models),
Variational Inference (VI) approximates the posterior by a simpler distribution $q\_\lambda(\theta)$ within a parameterized family.

9.3.2 2.2 Objective Function

We minimize the **Kullback–Leibler (KL) divergence**:

$$\mathrm{KL}(q_\lambda(\theta) \,\|\, p(\theta \mid y)) = \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{p(\theta \mid y)} \, d\theta.$$

Equivalently, we **maximize the Evidence Lower Bound (ELBO):**

$$\mathrm{ELBO}(\lambda) = E_{q_\lambda}[\log p(y, \theta)] - E_{q_\lambda}[\log q_\lambda(\theta)].$$

The higher the ELBO, the closer $q\_\lambda(\theta)$ is to the true posterior.

9.3.3 2.3 Mean-Field Approximation

A common simplification assumes factorization:

$$q_\lambda(\theta) = \prod_j q_{\lambda_j}(\theta_j),$$

which allows coordinate-wise optimization of each factor.

9.3.4 2.4 Example — Variational Bayes for a Normal Mean

Assume $y_i \sim N(\theta, 1)$ with prior $\theta \sim N(0, 1)$.

Analytically, the posterior is $N\left(\frac{n\bar{y}}{n+1}, \frac{1}{n+1}\right)$.
We approximate it variationally by another normal $q(\theta) = N(m, s^2)$,
and find m, s^2 maximizing ELBO.

```
set.seed(9)
y <- rnorm(50, mean=1)
n <- length(y)
log_joint <- function(theta) sum(dnorm(y, theta, 1, log=TRUE)) + dnorm(theta, 0, 1, log=TRUE)

# Closed-form optimal q is Normal(m,s^2) with same moments as true posterior:
m_vi <- n*mean(y)/(n+1)
s2_vi <- 1/(n+1)
c(mean=m_vi, sd=sqrt(s2_vi))
```

```
     mean        sd
0.8884051 0.1400280
```

9.3.5 2.5 Automatic VI with `brms`

```
library(brms)
set.seed(10)
N <- 1000
x <- rnorm(N)
y <- 2 + 1.5*x + rnorm(N)
dat <- data.frame(x,y)

fit_vi <- brm(y ~ x, data=dat, family=gaussian(),
              algorithm="meanfield", iter=5000, refresh=0)
summary(fit_vi)
```

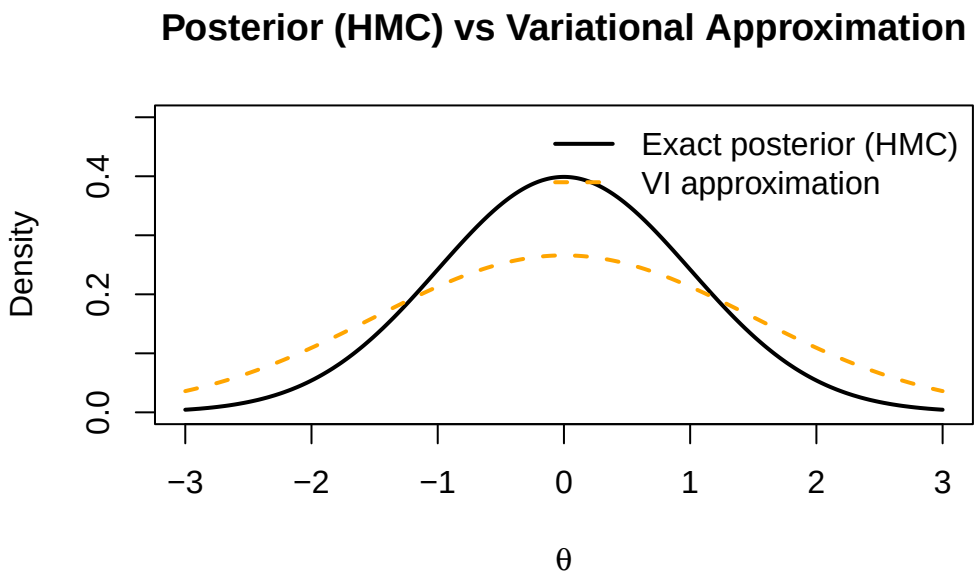
VI provides a fast deterministic approximation, trading off accuracy for scalability.

9.3.6 2.6 Comparison: HMC vs VI

Feature	HMC (NUTS)	Variational Inference
Type	Sampling (asymptotically exact)	Optimization (approximate)
Accuracy	Very high	Depends on variational family
Speed	Slower	Very fast
Diagnostics	Convergence via $\widehat{R}$, ESS	ELBO convergence
Use case	Complex or small data	Massive or real-time problems

9.3.7 2.7 Visual Comparison (Conceptual)

```
theta <- seq(-3,3,length=400)
posterior <- dnorm(theta, 0, 1)              # true posterior
vi_approx <- dnorm(theta, 0, 1.5)            # wider variational approx
plot(theta, posterior, type="l", lwd=2, col="black", ylim=c(0,0.5),
     main="Posterior (HMC) vs Variational Approximation",
     ylab="Density", xlab=expression(theta))
lines(theta, vi_approx, col="orange", lwd=2, lty=2)
legend("topright", legend=c("Exact posterior (HMC)","VI approximation"),
       col=c("black","orange"), lwd=2, lty=c(1,2), bty="n")
```



9.3.8 2.8 Practical Advice

- Use **HMC (NUTS)** as the default for accuracy and diagnostics.
- Use **VI** for large-scale models, initialization, or quick exploration.
- Compare results: if VI and HMC differ substantially, favor HMC.

9.4 Homework 8

1. **Conceptual**
   - Explain the difference between sampling-based and optimization-based inference.
   - What role does the ELBO play in VI?
2. **Computational**

- Fit a simple linear regression using both HMC and VI in `brms`.
- Compare posterior means, standard deviations, and computation time.

3. **Reflection**
   - In what types of real-world problems might VI be preferred over HMC?
   - How would you check whether your VI approximation is adequate?

9.5 Key Takeaways

Concept	Summary
Hamiltonian Monte Carlo	Uses gradients to propose efficient moves through parameter space.
No-U-Turn Sampler (NUTS)	Adapts step size and trajectory automatically.
Variational Inference	Optimizes a tractable approximation to the posterior.
ELBO	Objective function for VI; measures closeness to the true posterior.
Trade-off	HMC = accuracy, VI = speed; choice depends on model and data size.

Next Week: Bayesian Model Averaging and Ensemble Learning — combining multiple Bayesian models for improved predictive performance.

10 Week 9 — Bayesian Model Averaging and Ensemble Learning

This week introduces **Bayesian Model Averaging (BMA)**, a principled framework to combine inferences from multiple Bayesian models, and contrasts it with **ensemble methods** common in machine learning.
We discuss model uncertainty, predictive averaging, and practical implementations for linear regression and classification.

10.1 Learning Goals

By the end of this week, you should be able to:

- Explain the motivation for Bayesian Model Averaging.
- Derive model-averaged predictions using posterior model probabilities.
- Compare BMA with frequentist model selection and ML ensembles.
- Implement BMA for simple regression models in R.
- Discuss advantages and limitations of Bayesian model combination.

10.2 Lecture 1 — Bayesian Model Averaging (BMA)

10.2.1 1.1 Model Uncertainty

Model selection often ignores uncertainty about which model is true.
BMA accounts for this by averaging over all candidate models weighted by their posterior probabilities.

For models $M_1, \ldots, M_K$:

$$p(M_k \mid y) = \frac{p(y \mid M_k)\, p(M_k)}{\sum_{j=1}^{K} p(y \mid M_j)\, p(M_j)}.$$

Here: - $p(y$
$midM_k)$ = marginal likelihood under model M_k.
- $p(M_k)$ = prior model probability.
- $p(M_k$
$midy)$ = posterior model probability.

10.2.2 1.2 Model-Averaged Posterior and Predictions

Posterior distribution for parameter :

$$p(\theta \mid y) = \sum_{k=1}^{K} p(\theta \mid y, M_k)\, p(M_k \mid y).$$

Posterior predictive distribution:

$$p(\tilde{y} \mid y) = \sum_{k=1}^{K} p(\tilde{y} \mid y, M_k)\, p(M_k \mid y).$$

BMA integrates out model uncertainty rather than conditioning on a single "best" model.

10.2.3 1.3 Comparison with Model Selection

Approach	Key Idea	Limitation
Model Selection	Choose one best model (e.g., by AIC, WAIC, LOO)	Ignores model uncertainty
Model Averaging	Combine all models weighted by posterior probability	Computationally heavier, prior sensitive

---

10.2.4 1.4 Example — Two Competing Linear Models

```
set.seed(9)
n <- 100
x <- rnorm(n)
y <- 1 + 2*x + 0.5*x^2 + rnorm(n, sd=1)

m1 <- lm(y ~ x)
m2 <- lm(y ~ x + I(x^2))

log_marglik1 <- -AIC(m1)/2
log_marglik2 <- -AIC(m2)/2
p_m1 <- exp(log_marglik1)
p_m2 <- exp(log_marglik2)

w1 <- p_m1 / (p_m1 + p_m2)
w2 <- p_m2 / (p_m1 + p_m2)

pred1 <- predict(m1)
pred2 <- predict(m2)
bma_pred <- w1*pred1 + w2*pred2

c(weights=c(M1=w1, M2=w2)[1:2])
```

```
  weights.M1   weights.M2
1.426496e-08 1.000000e+00
```

```
plot(x, y, pch=19, col="#00000055", main="Bayesian Model Averaging (Linear vs Quadratic)",
     xlab="x", ylab="y")
xs <- seq(min(x), max(x), length.out=200)
lines(xs, predict(m1, newdata=data.frame(x=xs)), col="steelblue", lwd=2)
lines(xs, predict(m2, newdata=data.frame(x=xs)), col="firebrick", lwd=2)
lines(xs, w1*predict(m1, newdata=data.frame(x=xs)) +
          w2*predict(m2, newdata=data.frame(x=xs)),
      col="darkgreen", lwd=3, lty=2)
legend("topleft", legend=c("Model 1 (linear)","Model 2 (quadratic)","BMA prediction"),
       col=c("steelblue","firebrick","darkgreen"), lwd=c(2,2,3), lty=c(1,1,2), bty="n")
```

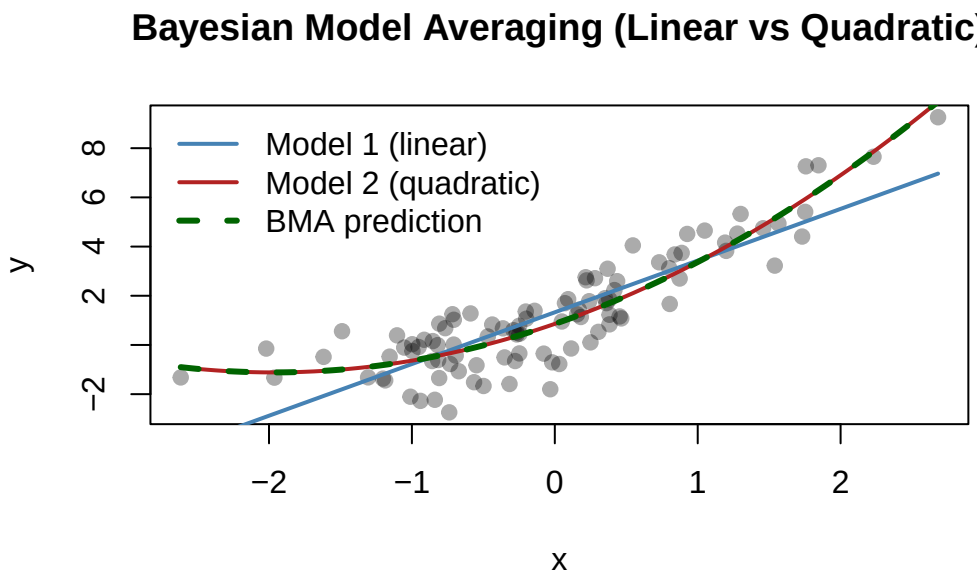


Figure 10.1: Model-averaged predictions vs. data

Interpretation: The model-averaged prediction blends the strengths of both models, weighted by their posterior support.

10.2.5 1.5 Advantages of BMA

- Incorporates model uncertainty directly.
- Avoids overconfidence from single-model conditioning.
- Improves predictive performance, especially in small samples.
- Provides model weights interpretable as probabilities.

10.2.6 1.6 Limitations

- Requires marginal likelihoods (often hard to compute).
- Sensitive to model priors and parameter priors.
- Computationally expensive for many models.

10.3 Lecture 2 — Bayesian Ensembles and Predictive Stacking

10.3.1 2.1 Beyond BMA: Ensemble Learning

Machine learning often uses **ensembles** (e.g., bagging, boosting, stacking) to improve prediction. Bayesian analogues combine predictive distributions rather than point estimates.

10.3.2 2.2 Predictive Stacking

Rather than using posterior model probabilities, stacking optimizes weights to **maximize predictive performance** under cross-validation:

$$w^* = \arg\max_w \sum_{i=1}^{n} \log \left(\sum_k w_k \, p(y_i \mid y_{-i}, M_k) \right),$$

subject to w_k
$ge0$ and
$sum_k w_k = 1$.

This yields **stacking weights** that combine models for best out-of-sample prediction.

10.3.3 2.3 Example — Predictive Stacking with `loo`

```r
library(brms)
library(loo)

set.seed(10)
dat <- data.frame(x = rnorm(200))
dat$y <- 1 + 2*dat$x + 0.5*dat$x^2 + rnorm(200)

m1 <- brm(y ~ x, data=dat, refresh=0)
m2 <- brm(y ~ x + I(x^2), data=dat, refresh=0)

loo1 <- loo(m1)
loo2 <- loo(m2)

# Stacking weights based on LOO predictive densities
w_stack <- loo_model_weights(list(m1,m2), method="stacking")
w_pseudo <- loo_model_weights(list(m1,m2), method="pseudobma")

w_stack
w_pseudo
```

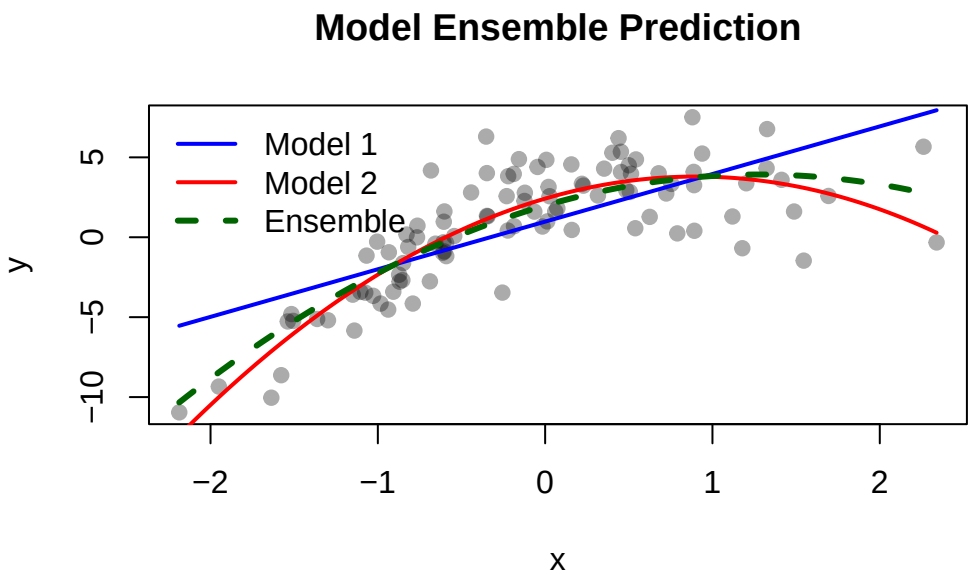
Interpretation:
- *Stacking weights* directly optimize predictive log-likelihood.
- *Pseudo-BMA* provides a simpler (WAIC/LOO-based) approximation.

10.3.4 2.4 Comparison: BMA vs Stacking

Feature	Bayesian Model Averaging	Predictive Stacking
Weights	Posterior model probabilities	Optimized predictive weights
Goal	Represent model uncertainty	Maximize predictive performance
Computation	Needs marginal likelihoods	Uses cross-validation
Prior dependence	Sensitive	Weak or none
Typical use	Theoretical coherence	Practical prediction

10.3.5 2.5 Ensemble Prediction Example

```
set.seed(11)
n <- 100
x <- rnorm(n)
y_true <- 2 + 3*x - 1.5*x^2
y <- y_true + rnorm(n, sd=2)

m1 <- lm(y ~ x)
m2 <- lm(y ~ poly(x, 2, raw=TRUE))

pred_grid <- seq(min(x), max(x), length=200)
p1 <- predict(m1, newdata=data.frame(x=pred_grid))
p2 <- predict(m2, newdata=data.frame(x=pred_grid))

# Ensemble weighting (ad hoc stacking weights)
w1 <- 0.3; w2 <- 0.7
p_ens <- w1*p1 + w2*p2

plot(x, y, pch=19, col="#00000055", main="Model Ensemble Prediction",
     xlab="x", ylab="y")
lines(pred_grid, p1, col="blue", lwd=2)
lines(pred_grid, p2, col="red", lwd=2)
lines(pred_grid, p_ens, col="darkgreen", lwd=3, lty=2)
legend("topleft", legend=c("Model 1","Model 2","Ensemble"),
       col=c("blue","red","darkgreen"), lwd=c(2,2,3), lty=c(1,1,2), bty="n")
```



10.3.6 2.6 Practical Guidance

- Use **BMA** when posterior model probabilities are available (few models, interpretable priors).
- Use **stacking or ensemble averaging** when prediction accuracy is the goal.
- Avoid double counting data — always base weights on held-out or cross-validation predictive performance.

10.4 Homework 9

1. **Conceptual**
   - Explain how BMA differs from model selection.
   - Why does stacking avoid prior sensitivity found in BMA?

2. **Computational**
    - Simulate data where two Bayesian regression models compete.
    - Fit both models in R (e.g., using `brms` or `lm`).
    - Compute stacking and pseudo-BMA weights using `loo_model_weights()`.
    - Compare model-averaged predictions to the true curve.
3. **Reflection**
    - Discuss when BMA and stacking might give very different results.
    - How can model averaging improve scientific interpretability?

10.5 Key Takeaways

Concept	Summary
Bayesian Model Averaging	Combines models weighted by posterior probabilities.
Predictive Stacking	Chooses weights that maximize predictive accuracy via cross-validation.
Model Uncertainty	Accounted for rather than ignored.
Practical Use	BMA for interpretability; stacking for prediction.
Modern Tools	`loo_model_weights()` in R provides both stacking and pseudo-BMA weights.

Next Week: Bayesian Nonparametrics — infinite-dimensional models such as Dirichlet processes and Gaussian processes for flexible Bayesian modeling.

11 Week 10 — Bayesian Nonparametrics

This week introduces **Bayesian Nonparametric (BNP)** models, which allow infinite flexibility in representing uncertainty about functions, densities, or model structure.
We focus on two cornerstone approaches: **Dirichlet Processes** for mixture modeling and clustering, and **Gaussian Processes** for regression on functions.

11.1 Learning Goals

By the end of this week, you should be able to:

- Explain the motivation for nonparametric Bayesian models.
- Describe the Dirichlet Process and its stick-breaking and Chinese Restaurant representations.
- Explain how DP mixture models perform clustering automatically.
- Understand Gaussian Processes for function estimation.
- Implement simple DP and GP examples in R using available packages.

11.2 Lecture 1 — Dirichlet Process Models

11.2.1 1.1 Motivation

Classical parametric models fix the number of parameters (e.g., number of clusters).
Dirichlet Processes (DPs) let the data decide model complexity by placing a prior on infinite mixture components.

11.2.2 1.2 The Dirichlet Process

A **Dirichlet Process** $\mathrm{DP}(\alpha, G_0)$ is a distribution over distributions such that, for any partition $A_1, \ldots, A_k$ of the space,

$$(G(A_1), \ldots, G(A_k)) \sim \mathrm{Dirichlet}(\alpha G_0(A_1), \ldots, \alpha G_0(A_k)).$$

- G_0: base (prior mean) distribution.
- α: concentration parameter controlling clustering strength.

As $\alpha \to 0$: few clusters (more sharing).
As $\alpha \to \infty$: many clusters (approaches G_0).

11.2.3 1.3 Stick-Breaking Representation

A constructive definition (Sethuraman, 1994):

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}, \quad \theta_k \sim G_0, \quad \pi_k = v_k \prod_{l<k} (1 - v_l), \quad v_k \sim \mathrm{Beta}(1, \alpha).$$

The weights π_k form an infinite sequence summing to 1 — conceptually "breaking a stick" into random lengths.

11.2.4 1.4 DP Mixture Model

For data $y_1, \ldots, y_n$:

$$y_i \mid \theta_i \sim F(\theta_i), \qquad \theta_i \mid G \sim G, \qquad G \sim \mathrm{DP}(\alpha, G_0).$$

Marginally, this induces **clustering** because multiple y_i can share the same θ_i.

11.2.5 1.5 Chinese Restaurant Process (CRP)

Equivalent generative process:

1. Customer 1 starts a new table.

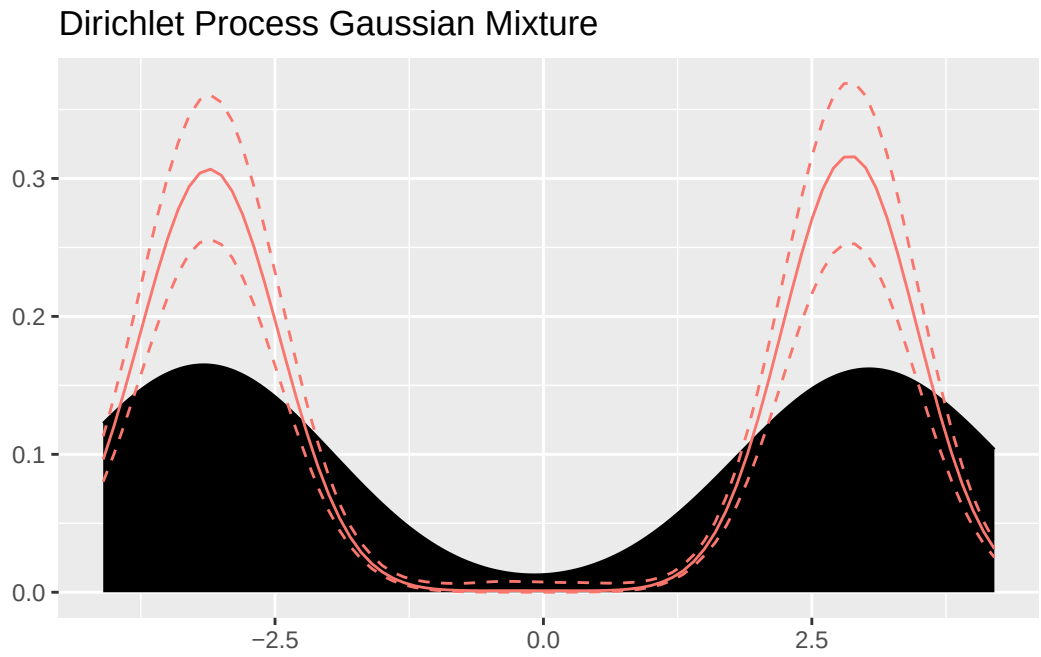
2. Customer i joins an existing table k with probability
   $n_k/(\alpha + i - 1)$,
   or starts a new one with probability $\alpha/(\alpha + i - 1)$.

This describes how clusters grow adaptively as data arrive.

11.2.6 1.6 Example — Simulated DP Mixture of Normals

```
set.seed(10)
library(dirichletprocess)

# Generate mixture data
y <- c(rnorm(50, -3, 0.5), rnorm(50, 3, 0.5))

# Fit a Dirichlet Process Gaussian Mixture
dp <- DirichletProcessGaussian(y)
dp <- Fit(dp, its = 2000)

# Cluster assignments
plot(dp) + ggplot2::ggtitle("Dirichlet Process Gaussian Mixture")
```

```
Warning: `aes_()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with `aes()`
i The deprecated feature was likely used in the dirichletprocess package.
  Please report the issue at
  <https://github.com/dm13450/dirichletprocess/issues>.
```



Interpretation: the DP mixture automatically discovers clusters without specifying their number in advance.

11.2.7 1.7 Practical Notes

- The posterior number of clusters depends on α and data separation.
- Inference via Gibbs sampling or variational truncation.
- Extensions: Hierarchical DP, DP regression, and DP topic models.

11.3 Lecture 2 — Gaussian Processes for Regression

11.3.1 2.1 Motivation

A **Gaussian Process (GP)** defines a prior directly over functions, enabling flexible nonlinear regression without specifying a parametric form.

11.3.2 2.2 Definition

A GP is a collection of random variables $f(x)$ such that every finite subset has a joint multivariate normal distribution:

$$f(x) \sim \mathrm{GP}(m(x), k(x, x')),$$

where

- $m(x) = E[f(x)]$: mean function,
- $k(x, x') = \mathrm{Cov}(f(x), f(x'))$: covariance (kernel) function.

11.3.3 GP Regression Model

For data (x_i, y_i):

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Posterior of $f(x)$ given y:

$$f_* \mid y, X, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{Cov}(f_*)),$$

where the mean and covariance are computed using kernel matrices.

11.3.4 2.4 Common Kernels

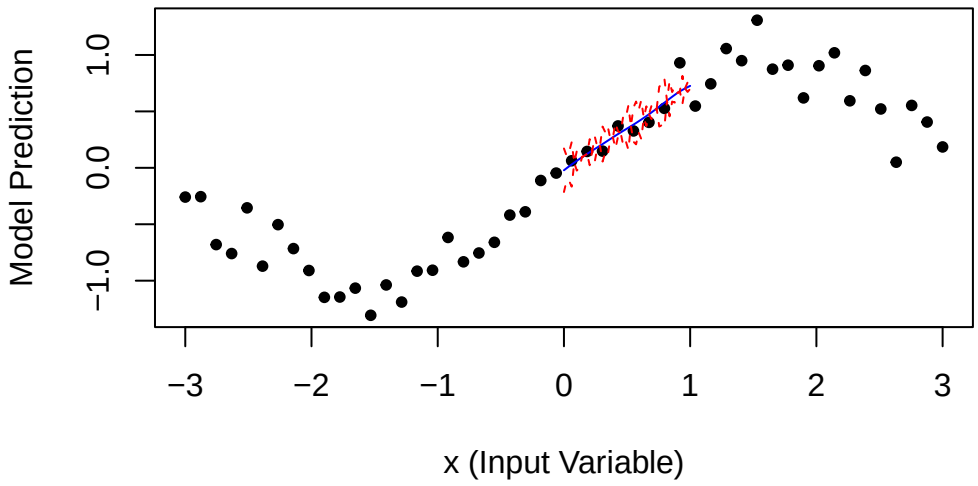
Kernel	Formula	Property
Squared Exponential	$k(x, x') = \tau^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	Smooth, infinitely differentiable
Matérn	$k(x, x') = \tau 2 \frac{21-\nu}{\Gamma(\nu)}(\sqrt{2\nu})$	x-x'
Periodic	$k(x, x') = \tau^2 \exp(-2\sin^2(\pi))$	x-x'

11.3.5 2.5 Example — Gaussian Process Regression in R

```r
library(GPfit)
set.seed(11)

x <- seq(-3, 3, length.out = 50)
y <- sin(x) + rnorm(50, sd = 0.2)

# Make X a column matrix (good practice for GPfit)
gp_model <- GP_fit(X = matrix(x, ncol = 1), Y = y)
```

```
Warning in GP_fit(X = matrix(x, ncol = 1), Y = y): X should be in range (0, 1)
```

```r
plot(gp_model)
```




```r
pred_grid <- seq(-3, 3, length.out = 200)
pred <- predict(gp_model, xnew = matrix(pred_grid, ncol = 1))

yhat <- as.numeric(pred$Y_hat)
se   <- sqrt(as.numeric(pred$MSE))  # MSE is a variance vector; no diag()

plot(x, y, pch = 19, col = "#00000055",
     main = "Gaussian Process Regression", xlab = "x", ylab = "y")
```

```r
lines(pred_grid, yhat, lwd = 2, col = "darkorange")
lines(pred_grid, yhat + 2*se, lty = 2, col = "gray40")
lines(pred_grid, yhat - 2*se, lty = 2, col = "gray40")
```

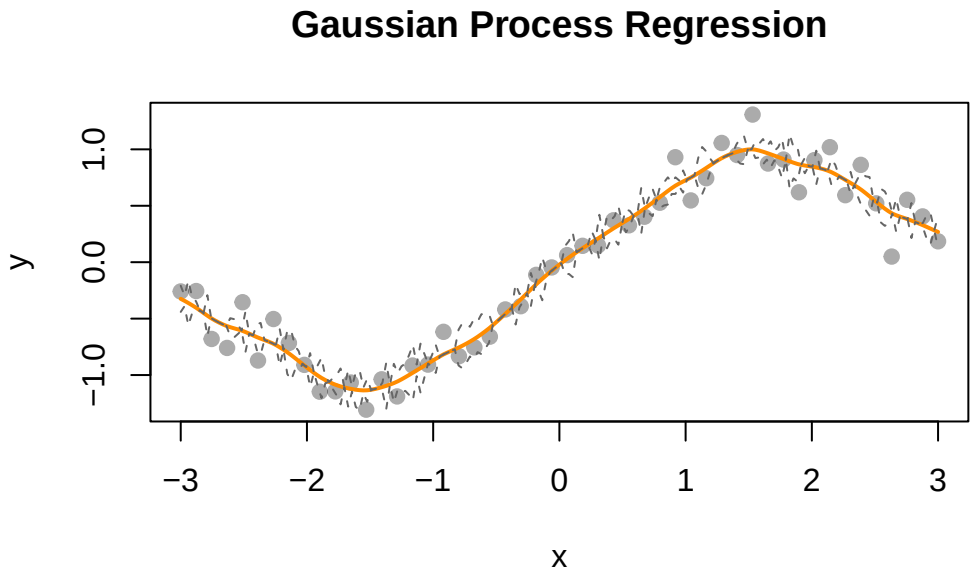


Figure 11.1: Gaussian Process Regression Fit

Interpretation: the GP posterior mean tracks the true function smoothly, with uncertainty quantified by the shaded region.

11.3.6 2.6 GP vs DP: Comparison

Aspect	Dirichlet Process	Gaussian Process
Domain	Distributions / Clusters	Functions
Output	Discrete clustering	Continuous regression
Flexibility	Unknown number of components	Infinite function space
Typical Use	Density estimation, mixture modeling	Nonlinear regression, spatial data

---

11.3.7 2.7 Practical Considerations

- GP computational cost is $O(n^3)$; use sparse or inducing-point approximations for large n.
- Choice of kernel determines function smoothness and inductive bias.
- In practice, hyperparameters (e.g., $\ell,\ \tau$) are learned via marginal likelihood maximization.

11.4 Homework 10

1. **Conceptual**
   - Compare the roles of α in DP and the kernel parameters in GP.
   - Explain the difference between parametric and nonparametric Bayesian models.
2. **Computational**
   - Simulate data from two Gaussian clusters and fit a DP mixture model using `dirichletprocess`.
   - Fit a GP regression to noisy sinusoidal data using `GPfit`.
   - Plot both model fits and discuss flexibility.
3. **Reflection**
   - When might nonparametric models be overkill?
   - How could hierarchical extensions of DP or GP handle grouped data?

11.5 Key Takeaways

Concept	Summary
Dirichlet Process	Prior over distributions enabling infinite mixture models.
Stick-Breaking Construction	Represents DP as weighted infinite discrete components.
Chinese Restaurant Process	Intuitive clustering interpretation of DP.
Gaussian Process	Defines prior over functions for regression and smoothing.
Common Feature	Both model complexity grows with data — no fixed parameter dimension.

Next Week: Bayesian Time Series and State-Space Models — dynamic modeling and sequential inference using Bayesian methods.

12 Week 11 — Bayesian Time Series and State-Space Models

This week introduces **Bayesian approaches to time series analysis** and **state-space modeling**, which unify filtering, forecasting, and dynamic parameter estimation under a probabilistic framework.
We study both classical dynamic linear models (DLMs) and modern Bayesian filtering methods.

12.1 Learning Goals

By the end of this week, you should be able to:

- Formulate Bayesian dynamic models for time series data.
- Understand latent (state) variable representations.
- Apply Bayesian updating and filtering for sequential data.
- Implement simple state-space and autoregressive models in R.
- Interpret uncertainty propagation over time.

12.2 Lecture 1 — Dynamic Linear Models (DLMs)

12.2.1 1.1 Motivation

Time series exhibit temporal dependence.
Dynamic models describe how latent states evolve over time and how observations depend on

those states:

$$\text{State equation: } \theta_t = G_t\theta_{t-1} + \omega_t, \quad \omega_t \sim N(0, W_t),$$
$$\text{Observation equation: } y_t = F_t^\top \theta_t + \nu_t, \quad \nu_t \sim N(0, V_t).$$

Here,

- $theta_t$: latent state vector,
- y_t : observed data,
- G_t, F_t : known system matrices,
- W_t, V_t : process and observation noise covariances.

12.2.2 1.2 Bayesian Updating

Given data up to time $t-1$, the **prior for** $theta_t$ is:

$$p(\theta_t \mid y_{1:t-1}) = N(a_t, R_t),$$

where

$$a_t = G_t m\_t - 1$$

,

$$R_t = G_t C\_t - 1 G_t top + W_t$$

.

After observing y_t

:

$$p(\theta_t \mid y_{1:t}) = N(m_t, C_t),$$

where

$$m_t = a_t + A_t(y_t - F_t topa_t)$$

,

$$C_t = R_t - A_t F_t topR_t$$

,
and

$$A_t = R_t F_t (F_t topR_t F_t + V_t) - 1$$

is the **Kalman gain**.

12.2.3 1.3 Example — Local Level Model

Simplest DLM:

$$y_t = \theta_t + \nu_t, \quad \theta_t = \theta_{t-1} + \omega_t,$$

with

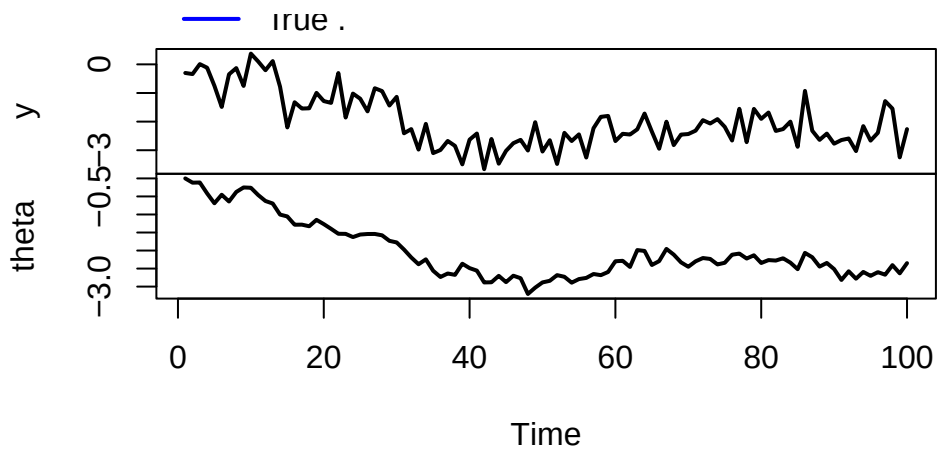
$$nu_t simN(0, V)$$

,

$$omega_t simN(0, W)$$

.

```r
set.seed(11)
n <- 100
theta <- numeric(n); y <- numeric(n)
theta[1] <- 0
for (t in 2:n) theta[t] <- theta[t-1] + rnorm(1, 0, 0.2)
y <- theta + rnorm(n, 0, 0.5)
plot.ts(cbind(y, theta), col=c("black","blue"), lwd=2,
        main="Local Level Model: True State vs Observed y", ylab="")
legend("topleft", legend=c("Observed y","True  "), col=c("black","blue"), lwd=2, bty="n")
```



12.2.4 1.4 Filtering with the Kalman Algorithm

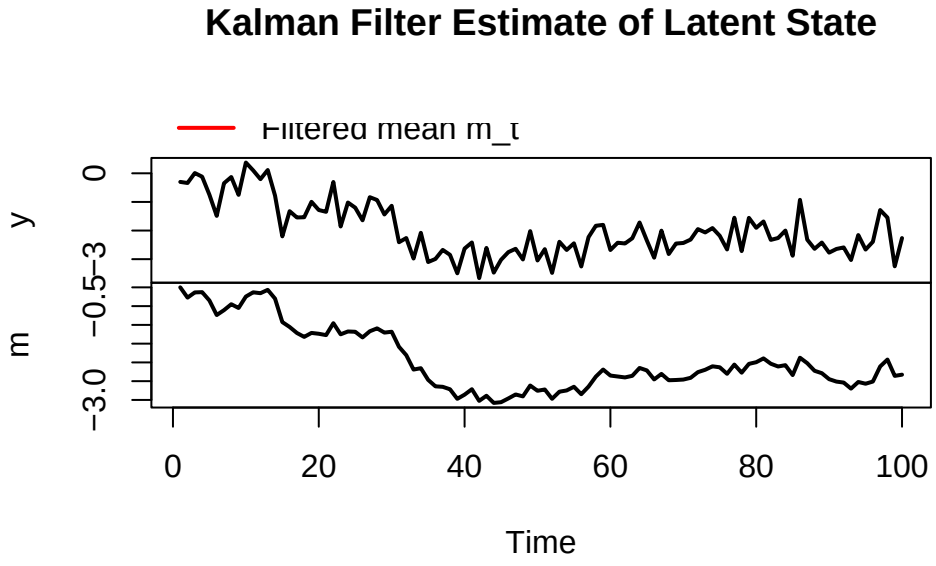
We estimate the evolving state mean

$$m_t$$

recursively:

```
m <- numeric(n); C <- numeric(n)
m[1] <- 0; C[1] <- 1
V <- 0.5^2; W <- 0.2^2

for (t in 2:n) {
  a <- m[t-1]
  R <- C[t-1] + W
  A <- R / (R + V)
  m[t] <- a + A * (y[t] - a)
  C[t] <- (1 - A) * R
}

plot.ts(cbind(y, m), col=c("black","red"), lwd=2,
```

```
       main="Kalman Filter Estimate of Latent State", ylab="")
legend("topleft", legend=c("Observed y","Filtered mean m_t"),
       col=c("black","red"), lwd=2, bty="n")
```



The red line tracks the smoothed latent process inferred from noisy data.

12.2.5 1.5 Forecasting and Uncertainty

Predictive distribution for the next observation:

$$y_{t+1} \mid y_{1:t} \sim N(F_{t+1}^{\top} a_{t+1}, F_{t+1}^{\top} R_{t+1} F_{t+1} + V_{t+1}).$$

Forecast variance increases as the state uncertainty grows over time.

12.2.6 1.6 Advantages of Bayesian DLMs

- Naturally handle missing observations.
- Flexible hierarchical extensions (time-varying parameters).
- Probabilistic forecasting with credible intervals.
- Online updating suitable for real-time applications.

12.3 Lecture 2 — State-Space Models and Bayesian Filtering

12.3.1 2.1 General State-Space Models

General form:

$$x_t = f(x_{t-1}) + \omega_t, \qquad y_t = g(x_t) + \nu_t,$$

where

$$f$$

and

$$g$$

may be nonlinear or non-Gaussian.
Examples: stochastic volatility, epidemic dynamics, tracking models.

12.3.2 2.2 Bayesian Filtering

We recursively compute:

$$p(x_t \mid y_{1:t}) \propto p(y_t \mid x_t) \int p(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{1:t-1})\, dx_{t-1}.$$

Closed forms exist only for linear-Gaussian models (Kalman).
Otherwise, approximate methods are required: - **Particle Filtering** (Sequential Monte Carlo).
- **Extended Kalman Filter** (linearization).
- **Unscented Kalman Filter** (deterministic sampling).

12.3.3 2.3 Particle Filtering (Sequential Monte Carlo)

Idea: Represent

$$p(x_t midy\_1:t)$$

by weighted particles

$$x_t(i), w_t(i)$$

.

1. **Prediction:** Sample

   $$x_t(i) simp(x_t midx\_t-1(i))$$

   $.

2. **Weighting:**

   $$w_t(i) proptop(y_t midx_t(i))$$

   .

3. **Resampling:** Normalize and resample particles based on

   $$w_t(i)$$

   .

As

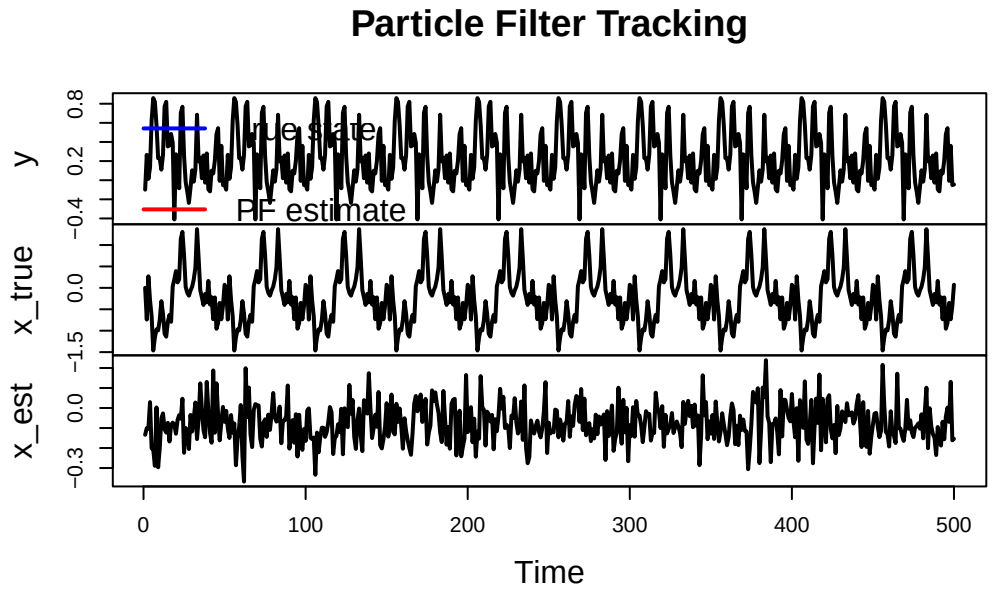
$$Ntoinfty$$

, the empirical distribution approximates the true posterior.

12.3.4 2.4 Example — Particle Filter for a Simple Nonlinear Model

```
set.seed(12)
n <- 50; Np <- 500
x_true <- numeric(n); y <- numeric(n)
x_true[1] <- 0
for (t in 2:n) x_true[t] <- 0.7*x_true[t-1] + rnorm(1,0,0.5)
y <- x_true^2/2 + rnorm(n,0,0.2)

# Initialize particles
x_pf <- matrix(0, nrow=n, ncol=Np)
```

```r
w <- matrix(1/Np, nrow=n, ncol=Np)
x_pf[1,] <- rnorm(Np, 0, 1)

for (t in 2:n) {
  # Propagate
  x_pf[t,] <- 0.7*x_pf[t-1,] + rnorm(Np,0,0.5)
  # Weight
  w[t,] <- dnorm(y[t], mean=x_pf[t,]^2/2, sd=0.2)
  w[t,] <- w[t,]/sum(w[t,])
  # Resample
  idx <- sample(1:Np, Np, replace=TRUE, prob=w[t,])
  x_pf[t,] <- x_pf[t,idx]
}

x_est <- colMeans(x_pf)
plot.ts(cbind(y, x_true, x_est), col=c("black","blue","red"), lwd=2,
        main="Particle Filter Tracking", ylab="")
legend("topleft", legend=c("Observed y","True state","PF estimate"),
       col=c("black","blue","red"), lwd=2, bty="n")
```



The particle filter captures nonlinear state dynamics unavailable to standard Kalman filtering.

12.3.5 2.5 Extensions and Modern Bayesian State Models

- **Dynamic GLMs** for count or binary data.
- **Stochastic Volatility Models** in finance.
- **Dynamic Factor Models** for multivariate time series.
- **Bayesian Structural Time Series (BSTS)** — trend + seasonality decomposition.

```r
library(bsts)
data(Seatbelts)
y <- Seatbelts[,"drivers"]
ss <- AddLocalLevel(list(), y)
bsts_fit <- bsts(y ~ 1, state.specification = ss, niter = 2000)
plot(bsts_fit)
```

12.3.6 2.6 Practical Considerations

- Choose priors for
$$V_t, W_t$$
that balance smoothness and responsiveness.
- Use HMC or SMC samplers for full Bayesian inference.
- Check convergence and predictive calibration through one-step-ahead forecasts.

12.4 Homework 11

1. **Conceptual**
    - Explain the difference between filtering and smoothing in Bayesian time series analysis.
    - When does the Kalman filter provide exact inference?

2. **Computational**
    - Simulate a local-level model and implement a Kalman filter in R.
    - Extend it with time-varying variance or drift.
    - Compare filtered estimates to true latent states.
3. **Reflection**
    - How does particle filtering generalize the Kalman filter?
    - Discuss a potential application of Bayesian state-space modeling in your field.

12.5 Key Takeaways

Concept	Summary
Dynamic Linear Model (DLM)	Linear-Gaussian state-space model solved by Kalman filter.
Kalman Filter	Sequential Bayesian updating for latent states.
Particle Filter	Simulation-based approach for nonlinear or non-Gaussian models.
Forecasting	Naturally derived from predictive posterior distribution.
Extensions	BSTS, stochastic volatility, dynamic regression, multivariate models.

Next Week: Hierarchical Bayesian Inference for Complex Systems — multi-level time series and dynamic hierarchical modeling.

13 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

Part I

Appendix

14 Appendix: Introduction to R

14.1 R

For conducting analyses with data sets of hundreds to thousands of observations, calculating by hand is not feasible and you will need a statistical software. **R** is one of those. **R** can also be thought of as a high-level programming language. In fact, **R** is one of the top languages to be used by data analysts and data scientists. There are a lot of analysis packages in **R** that are currently developed and maintained by researchers around the world to deal with different data problems. Most importantly, **R** is free! In this section, we will learn how to use **R** to conduct basic statistical analyses.

14.2 IDE

14.2.1 Rstudio

RStudio is an integrated development environment (IDE) designed specifically for working with the **R** programming language. It provides a user-friendly interface that includes a source editor, console, environment pane, and tools for plotting, debugging, version control, and package management. RStudio supports both **R** and Python and is widely used for data analysis, statistical modeling, and reproducible research. It also integrates seamlessly with tools like **R** Markdown, Shiny, and Quarto, making it popular among data scientists, statisticians, and educators.

14.2.2 Visual Studio Code (VS Code)

VS Code is a versatile code editor that supports multiple programming languages, including **R**. With the **R** extension for VS Code, users can write and execute **R** code, access **R**'s console, and utilize features like syntax highlighting, code completion, and debugging. While not as specialized as RStudio for **R** development, VS Code offers a lightweight alternative with extensive customization options and support for various programming tasks.

14.2.3 Positron

Positron IDE is the next-generation integrated development environment developed by Posit, the company behind RStudio. Designed to be a modern, extensible, and language-agnostic IDE, Positron builds on the strengths of RStudio while supporting a broader range of languages and workflows, including **R**, Python, and Quarto.

14.3 RStudio Layout

RStudio consists of several panes: - **Source**: Where you write scripts and markdown documents. - **Console**: Where you type and execute **R** commands. - **Environment/History**: Shows your variables and command history. - **Files/Plots/Packages/Help/Viewer**: For file management, viewing plots, managing packages, accessing help, and viewing web content.

14.4 R Scripts

R scripts are plain text files containing **R** code. You can create a new script in RStudio by clicking `File > New File > R Script`.

14.5 R Help

Use `?function_name` or `help(function_name)` to access help for any **R** function. For example:

```
?mean
help(mean)
```

14.6 R Packages

Packages extend **R**'s functionality. There are thousands of packages available in **R** ecosystem. You may install them from different sources.

14.6.1 With Comprehensive R Archive Network (CRAN)

CRAN is the primary repository for **R** packages. It contains thousands of packages that can be easily installed and updated.

Install a package with:

```
install.packages("package_name")
```

14.6.2 With Bioconductor

Bioconductor is a repository for bioinformatics packages in **R**. It provides tools for the analysis and comprehension of high-throughput genomic data.

Install Bioconductor packages using the `BiocManager` package:

```
BiocManager::install("package_name")
```

14.6.3 From GitHub

Many of the authors of **R** packages host their work on GitHub. You can install these packages using the `devtools` package:

```
devtools::install_github("username/package_name")
```

14.6.4 Load a package

Once a package is installed, you need to load it into your **R** session to use its functions:

```
library(package_name)
```

Alternatively, you may use a function in the package with `package_name::function_name()` without loading the entire package.

14.7 R Markdown

R Markdown allows you to combine text, code, and output in a single document. Create a new **R** Markdown file in RStudio via `File > New File > R Markdown...`.

Recently, the posit team has developed a new version of the **R** Markdown called quarto document, with the file extension `.qmd`. It is still under rapid development.

14.8 Vectors

Vectors are the most basic data structure in **R**.

```
x <- c(1, 2, 3, 4, 5)
x
```

```
[1] 1 2 3 4 5
```

You can perform operations on vectors:

```
x * 2
```

```
[1]  2  4  6  8 10
```

14.9 Data Sets

Data frames are used for storing data tables. Create a data frame:

```
df <- data.frame(Name = c("Alice", "Bob"), Score = c(90, 85))
df
```

```
   Name Score
1 Alice    90
2   Bob    85
```

You can import data from files using `read.csv()` or `read.table()`.

This appendix is adapted from Why R?.

References