

Mathematical Statistics II

Maximum Likelihood Estimation

Jesse Wheeler

Introduction

- The next approach we will discuss is Maximum Likelihood Estimation (MLE).
- As we will see, the MLE has several desirable properties, and as a result is often favored over approaches like the method of moments.
- The material for this section largely comes from Chapter 8.5 of Rice (2007), and various sections in Pawitan (2001).

Likelihood: an introduction

What is likelihood?

- The term “likelihood” is often used colloquially to mean something analogous to probability. E.g., “What is the likelihood that it rains tomorrow?”
- When we use this term in statistics / mathematics, we mean something specific that isn’t the same thing as probability.
- The use of the term “likelihood” was first made by R. A. Fisher, who was the architect and primary proponent of “likelihood-based-inference”.
- We will start with the treatment of likelihood in the text “In all Likelihood” (Pawitan, 2001), which is a fantastic resource on the subject. (This will lead to some review...)

What is likelihood? II

Coin Flips

We will revisit this example, as it is a great starting point to connect with existing understanding.

Consider flipping a coin $N = 10$ times.

What is likelihood? III

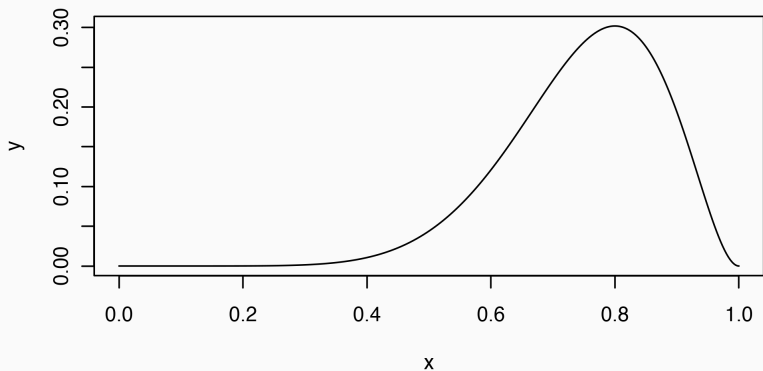
- For our specific coin-flipping example with $N = 10$, $X = 8$, the likelihood function is

$$L(\theta) = P_{\theta}(X = 8).$$

- This is plotted in the following way:

```
x <- seq(1e-8, 1-1e-8, length.out = 1000)
y <- dbinom(8, 10, x)
plot(x = x, y = y, type = 'l')
```

What is likelihood? IV



What is likelihood? V

- From the figure, we see that p is unlikely to be less than 0.5, or greater than 0.95.
- Given the data alone (no prior), we should prefer a value somewhere in the middle of these values.
- We still have some uncertainty about the value of p , but the likelihood gives us a numerical way to compare values of θ . Stochastic uncertainty as a result of sampling is captured in the likelihood function $L(\theta)$.

What is likelihood? VI

- The likelihood is not a probability. Though it came from a probability, the likelihood function (a function of θ) does not satisfy the requirements to be a probability. In our previous example, we have:

$$\int_0^1 L(\theta) d\theta = 1/11 \neq 1.$$

- For discrete probability, the likelihood was continuous. Discrete likelihoods are possible, arising when we want to select from a list $\{\theta_1, \theta_2, \dots\}$.

What is likelihood? VII

- The idea behind maximum likelihood estimation (MLE) is simple: our estimate is the value of θ that maximizes the likelihood function $L(\theta)$.
- The MLE is considered a *frequentist* approach. Why? It quantifies a maximum belief about a parameter, which is more Bayesian in nature than Frequentist.
- As we'll see later, the MLE has nice theoretical Frequentist *properties*, and as a result can be justified via the frequentist paradigm.
- Still, it has close connection to Bayesian estimation and interpretation. In fact, we'll discuss connections between the MLE and Bayesian statistics later.

What is likelihood? VIII

- Often, maximizing the likelihood directly is challenging, so we maximize the log-likelihood instead.
- Other times, the likelihood has to be maximized numerically.

MLE of coin toss problem

Suppose we have $N = 10$ total tosses, and n total heads. Find the MLE of p , the probability of heads.

Continuous models

- The interpretation of the likelihood function as the “the probability of the observed data x^* , considered as a function of θ ” makes perfect sense in the discrete model case.
- For continuous models, the technical issue arises that the probability of any point value x is zero.
- We resolve the problem similar to what was done in Math 4450 and the John Rice text: approximate the probability by discretizing into small, discrete intervals:

$$x^* \in (x^* - \epsilon/2, x^* + \epsilon/2),$$

Continuous models II

thus, the probability of observing something ϵ -close to the data is:

$$\begin{aligned} L(\theta) &= P_{\theta}(X \in (x^* - \epsilon/2, x^* + \epsilon/2)) \\ &= \int_{x^* - \epsilon/2}^{x^* + \epsilon/2} f(x; \theta) d\theta \approx \epsilon f(x^*; \theta). \end{aligned}$$

- Then, since the likelihood is only meaningful up to a constant (we will discuss likelihood ratios later), then this has the same behavior as $L(\theta) = f(x^*; \theta)$.
- There are more advanced approaches to this problem, but this simple argument justifies the use of the pdf of a continuous random variable as the likelihood $L(\theta)$.

Continuous models III

- **Going forward:** we once again will generalize a model $f(x; \theta)$ to mean either the pmf or pdf of a random variable. I will often say “density” as a blanket term, even if this corresponds to a pmf, not a density.
- Further, when we “integrate” a density, this means either:

$$\int f(x; \theta) dx, \quad \text{If continuous}$$

or

$$\sum_x f(x; \theta), \quad \text{If discrete.}$$

Joint Probabilities

Likelihood with multiple observations

- Often the data we observe is multi-dimensional, rather than summarized as a single observation.
- In this case, the likelihood θ is still determined via the joint model:

$$L(\theta) = f(x^*; \theta) = f_{X_{1:N}}(x_1^*, x_2^*, \dots, x_N^*; \theta).$$

- We are mostly focused in this class in the case where the observations are independent, meaning the likelihood factors:

$$L(\theta) = \prod_{i=1}^N f_{X_i}(x_i^*; \theta).$$

Likelihood with multiple observations II

- We often further simplify this by assuming the data are identically distributed:

$$L(\theta) = \prod_{i=1}^N f_{X_1}(x_i^*; \theta).$$

- As we've seen, it's generally easier to maximize the log-likelihood. In the IID case:

$$\ell(\theta) = \log \prod_{i=1}^N f_{X_1}(x_i^*; \theta) = \sum_{i=1}^n \log f_{X_1}(x_i^*; \theta).$$

Examples

Examples of finding the MLE

Traffic data: Poisson Model

Returning to a motivating example, suppose we model traffic accidents in a given week as X_1, X_2, \dots, X_N , where the data are iid $\text{Poisson}(\lambda)$. Obtain the MLE for λ .

Examples of finding the MLE II

Two parameter model: Gaussian model

Suppose we model observations X_1, \dots, X_N as IID $N(\mu, \sigma^2)$ random variables. Find the MLE of $\theta = (\mu, \sigma^2)$.

Plotting Normal Likelihood

- The likelihood function (not just to point estimate) will be used to measure uncertainty.
- For models with a single parameter, we often plot the likelihood curve.
- With more than one parameter, however, we have a **likelihood surface**.
- For the iid Normal(μ, σ^2) model, code for plotting this surface is available with course source-code.

Plotting Normal Likelihood II

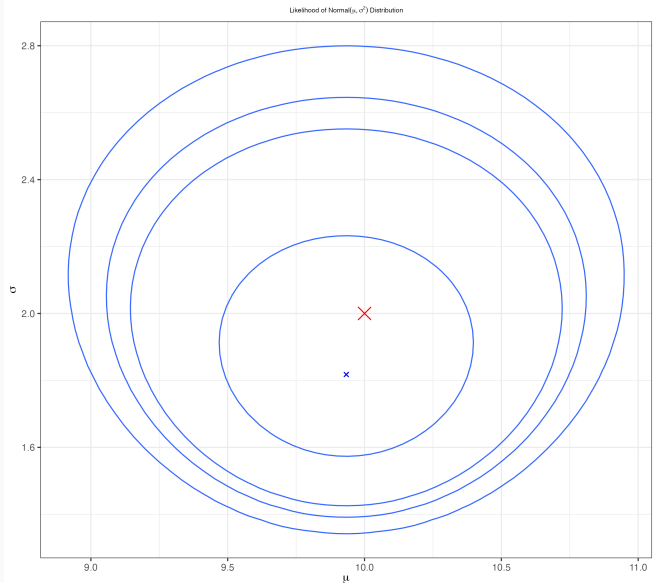


Figure 1: Likelihoods of data generated from a normal distribution

Calculating the likelihood in R

- For standard distributions, the likelihood is easy to calculate in R. Recall the likelihood is:

$$L(\theta) = f(x_i^*; \theta).$$

- For X_1, \dots, X_n iid normal, the likelihood can be calculated using `dnorm`:

```
# Synthetic data
x_star <- rnorm(n = 100, mean = 4, sd = 3)
theta <- c(0, 1) # value of theta
Likelihood <- prod(dnorm(x, mean = theta[1], sd = theta[2]))
loglik <- sum(
  dnorm(x, mean = theta[1], sd = theta[2], log = TRUE)
)
```


Numeric Optimization

Numeric Optimization

- In the previous examples, the MLE was available *analytically*.
- In many cases, however, there is no closed-form solution for the MLE, and it must be computed numerically.
- The next example demonstrates this, and then we will discuss optimization strategies.

Example: Gamma likelihood

Suppose we want to model data X_1, X_2, \dots, X_n as iid $\text{Gamma}(\alpha, \lambda)$, which has the density function:

$$f(x; \alpha, \lambda) = \frac{1}{\Gamma(\alpha)} \lambda^\alpha x^{\alpha-1} e^{-\lambda x}, \quad 0 \leq x < \infty.$$

Find the MLE of $\theta = (\alpha, \lambda)$.

Numeric Optimization III

- The previous example leads us to consider numeric techniques for optimization and root finding.
- Note that this class is **not** an optimization course, so we'll only cover some of the most basic ideas.
- More modern and efficient numeric optimization techniques are readily available in R (or any other statistical software).
- For this class, we'll introduce some basic ideas like the Newton-Raphson approach for root finding and optimization, as well as some other basic methods.

Newton-Raphsom root-finding algorithm

- Idea: start at a point θ_0 , and approximate find the tangent line of $f(\theta)$ at the point θ_0 :

$$y - f(\theta_0) = f'(\theta_0)(\theta - \theta_0)$$

- Then, find the root of the tangent line by setting $y = 0$, and solving for θ :

$$\theta = \theta_0 - \frac{f(\theta_0)}{f'(\theta_0)}.$$

Newton-Raphsom root-finding algorithm II

- This root of the tangent line will be closer than our original guess θ_0 , so we set:

$$\theta_1 = \theta_0 - \frac{f(\theta_0)}{f'(\theta_0)},$$

and repeat:

$$\theta_{n+1} = \theta_n - \frac{f(\theta_n)}{f'(\theta_n)}.$$

- We stop based on some convergence criteria, often something like $|\theta_{n+1} - \theta_n| < \epsilon$, for a small choice of ϵ .
- (In class, check out wikipedia or draw a picture).

Newton-Raphsom root-finding algorithm III

- We need now a starting point θ_0 .
- Really we can pick anything, but it's best if we are close to the MLE.
- For our current problem (Gamma distribution), we could use the MoM estimator:

$$\hat{\alpha}_{\text{MoM}} = \theta_0 = \frac{(\bar{x}_n^*)^2}{\frac{1}{n} \sum_{i=1}^n (x_i^* - \bar{x}_n^*)^2}.$$

Newton-Raphsom root-finding algorithm IV

```
NR_root <- function(theta0, fn, deriv, tol = 1e-8, maxiter = 100) {  
  iter <- 0  
  theta_old <- theta0  
  theta_new <- theta_old + 10 * tol  
  
  while(abs(theta_old - theta_new) > tol && iter < maxiter) {  
    iter <- iter + 1  
    theta_old <- theta_new  
    theta_new <- theta_old - fn(theta_old) / deriv(theta_new)  
  }  
  cat("iters: ", iter, "\n")  
  theta_new  
}
```


Newton-Raphsom root-finding algorithm V

```
alpha_fn <- function(alpha, data) {  
  n <- length(data)  
  n * log(alpha) - n * log(mean(data)) + sum(log(data)) - n * di  
}  
  
alpha_deriv <- function(alpha, data) {  
  n <- length(data)  
  (n/alpha) - n * psigamma(alpha, 1)  
}  
  
set.seed(123)  
data <- rgamma(n = 23, shape = 1, rate = 2)
```

Newton-Raphsom root-finding algorithm VI

```
# Not the exact MoM estimate, but close enough:
```

```
alpha_mom <- (mean(data)^2) / sd(data)
```

```
NR_root(  
  theta0 = alpha_mom,  
  fn = function(x) alpha_fn(x, data = data),  
  deriv = function(x) alpha_deriv(x, data = data),  
  tol = 1e-10  
)
```

```
iters: 7
```

```
[1] 0.9728019
```

Newton-Raphsom root-finding algorithm VII

- Once the estimate of α is found, we can then plug it in to get the estimate of λ :

$$\hat{\lambda} = \frac{\hat{\alpha}}{\bar{x}_n^*} = 1.981.$$

Newton-Raphsom root-finding algorithm VIII

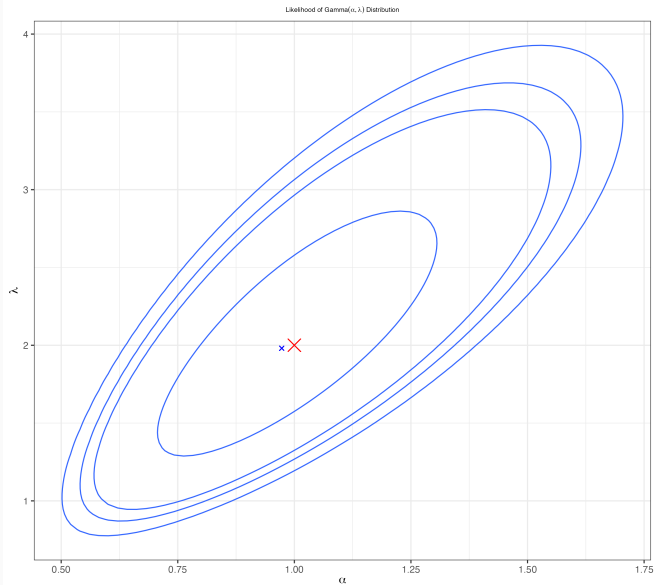


Figure 2: Likelihood surface of data generated from Gamma distribution

Root-find considerations

- The function that we built for solving $f(\theta) = 0$ requires the derivative $f'(\theta)$.
- In some cases, the derivative is not readily available. Instead, we can approximate using the definition:

$$f'(\theta) = \lim_{h \rightarrow 0} \frac{f(\theta + h) - f(\theta)}{h} \approx \frac{f(\theta + \Delta) - f(\theta)}{\Delta}.$$

- We just pick Δ to be small, and we get a very good approximation of the derivative.
- Thus, we don't really need to derivative for uni-variate root-finding.
- The same mathematical approach can readily be extended into higher dimensional θ , replacing the derivative with a **gradient**.

Root-find considerations II

- An important consideration, however, is that the results may depend on your starting parameter θ_0 . In practice, you may want to try multiple values of θ_0 .

Root-find considerations III

- In most programming languages, there will be pre-built methods for root-finding and optimization.
- Our function we built works, but it doesn't do careful error checking, and it isn't optimized for speed.
- For univariate-root finding $f(\theta) = 0$ in \mathbb{R} , we can use the `uniroot` function, which doesn't require a derivative.
- This function is very efficient for solving roots if $\theta \in \mathbb{R}$. For higher dimensions, we need to import a different package.

Root-find considerations IV

```
uniroot(  
  function(x) alpha_fn(x, data = data),  
  interval = c(0.5, 2)  
)
```

```
$root
```

```
[1] 0.9728068
```

```
$f.root
```

```
[1] -7.754612e-05
```

```
$iter
```

```
[1] 6
```

```
$init.it
```

```
[1] NA
```


Direct Numeric Optimization

- Based on our last example, we solved for one parameter λ , and numerically found the other, α , via root-finding.
- In many cases, this is not possible. Instead, we might want to directly optimize both parameters.
- We will first introduce this by extending the Newton-Raphson to perform optimization rather than root-finding.

Direct Numeric Optimization II

- The idea is simple: local maximum are just the zeros (roots) of the derivative function.
- Thus, we still perform Newton-Raphson, but on the derivative rather than the original function.
- Starting with initial estimate θ_0 , we update following the equations until convergence:

$$\theta_{n+1} = \theta_n - \frac{f'(\theta_n)}{f''(\theta_n)}$$

Direct Numeric Optimization III

- If θ is multivariate, then again we use the **gradient**: $\nabla f(\theta)$, which is a vector, and Hessian: $\nabla^2 f(\theta)$, which is a matrix:

$$\theta_{n+1} = \theta_n - (\nabla^2 f(\theta_n))^{-1} \nabla (f(\theta_n)).$$

- This is the basic approach of many traditional machine learning algorithms:
 - Pick a model that depends on θ , and a loss-function $f(\theta)$ that depends on the data and model (here, our loss is the log-likelihood function).
 - If possible, calculate the derivative of the loss function with respect to θ . If not, approximate numerically.
 - If possible, calculate the Hessian of the loss function with respect to θ . If not, approximate numerically.

Direct Numeric Optimization IV

- Start with initial guess for θ , take steps the size of the (approximate) hessian of $f(\theta)$, in the direction of the gradient $f(\theta)$.
- Many optimization strategies are variants of this basic approach: In practice, Hessians / Gradients may be expensive to compute.
- We will primarily focus our attention on pre-existing solutions.
- In R, the function we will use is called `optim`.
- There are several optimization methods available within this function

Direct Numeric Optimization V

- Nelder-Mead: a heuristic, direct search method. Does not require differentiability. Slow, but robust.
- BFGS / L-BFGS-B: A Quasi-Newton approach, approximates either (or both) the Hessian and Gradient numerically. Extremely effective for (twice) differentiable functions, but slow as θ grows in dimension.
- SANN: A stochastic approach. Hard to tune, but effective on “rough” surfaces
- CG: Conjugate Gradients. More “fragile” than BFGS, but require less memory storage so can be useful for large θ .
- Brent: Only univariate θ . In these cases, approximates the gradient and hessian, similar to what we proposed. Very effective, but requires univariate θ .

Direct Numeric Optimization VI

- How it works: Get function f , method you want to use, and starting point θ_0 . Example:

```
gamma_negloglik <- function(theta) {  
  -dgamma(  
    x = data,  
    shape = theta[1], rate = theta[2],  
    log = TRUE  
  ) |> sum()  
}
```

Direct Numeric Optimization VII

```
optim(  
  c(2, 5),  
  fn = gamma_negloglik,  
  method = 'L-BFGS-B', # Constrained theta>0  
  lower = c(1e-8, 1e-8) # Constrained theta>0  
)
```

\$par

```
[1] 0.9728026 1.9806150
```

\$value

```
[1] 6.641716
```

\$counts

function gradient

16

16

Gradient Descent and Variations

- In Newton-Raphson, the Hessian just tells us the *size* of the step.
- Often the gradient is available, but not the Hessian (or it is expensive to compute the inverse, since it is a matrix).
- In these cases, we often use a different approach called **gradient descent**.
- Idea: still step in the direction of the (negative) gradient, but the step size will just be small δ_n , and let δ_n shrink over-time:

$$\theta_{n+1} = \theta_n - \delta_n \nabla f(\theta_n).$$

Gradient Descent and Variations II

- The idea is simple, but a lot of modern optimization techniques and theory are based on this idea, finding various strategies for specifying δ_n .
- Ex: BFGS approximates the Hessian, but scales poorly with dimension. In deep learning, θ has dimension in millions / billions, so BFGS won't work (nor does `optim`).
- The success of deep-learning and AI is largely due to the advent of **automatic differentiation**: software that calculates the exact gradient of $f(\theta_n)$, after simple calculations.
- Auto-diff libraries: PyTorch, TensorFlow, JAX, etc. Mostly available in Python, though some are available in R.

Gradient Descent and Variations III

- Final approach we will discuss is **stochastic** gradient descent, which is effectively randomly sampling the data to compute an approximate gradient.
- The idea is that, even with auto-diff, gradients (with entire data) are expensive to compute.
- Thus, randomly sample data to get a stochastic approximation of the gradient; this is faster to compute, so we get more update-steps.
- This is the primary technique used in most deep-learning frameworks.

Maximum Likelihood: continued examples

Muon Decay

Returning to the Muon-Decay example we used MoM to solve.

The density of iid observations X_1, \dots, X_n is:

$$f(x; \alpha) = \frac{1 + \alpha x}{2}, \quad -1 \leq x \leq 1 \quad -1 \leq \alpha \leq 1.$$

Find the MLE of α .

Constraint Optimization

Constrained optimization

- In all of our previous examples, we want to maximize the log-likelihood function $f(x_i^*; \theta)$, and there is some natural constraints of θ .
- Examples: In $\text{Gamma}(\alpha, \lambda)$, $\theta = (\alpha, \lambda)$, and we are constrained by $\alpha, \lambda > 0$.
- In the Gaussian model, $\theta = (\mu, \sigma)$. μ is unconstrained, but $\sigma \geq 0$.
- Similarly, in the Muon-decay example: $\theta = \alpha \in (0, 1)$.
- This leads to the issue of *constrained* optimization, and is typically a requirement for maximum likelihood estimation.
- There are a few common strategies:

Constrained optimization II

- Limit search region to be within constraints: This is the basic approach used by the `uniroot` function. It's also used sometimes in `optim`, e.g., the L-BFGS-B method.
- Variable transformations. Modify the function so that the search algorithm can try all possible values, but apply a transformation that keeps it in the desired range.
 - Example: if $\theta > 0$ is a constraint, optimize over α with $\theta = e^\alpha$. Thus, $\alpha < 0$ is no problem, since $\theta > 0$. (Example on next slide)
 - Another common example is a logit transformation, which ensures $0 \leq \theta \leq 1$, via $\log(\theta/(1 - \theta)) = \alpha$.
- In some cases, the constraint leads to an equation $g(\theta) = 0$, in which case we can apply something like Lagrange-multipliers. We will look at one of these cases as well.

Example: Gamma likelihood

MLE of Gamma likelihood using transformations

Revisit the numeric optimization of the Gamma likelihood, using variable transformations to deal with constraints.

```
gamma_negloglik2 <- function(alpha) {  
  theta <- exp(alpha)  
  -dgamma(  
    x = data,  
    shape = theta[1], rate = theta[2],  
    log = TRUE  
  ) |> sum()  
}
```

Example: Gamma likelihood II

```
results <- optim(  
  c(log(2), log(5)),  
  fn = gamma_negloglik2,  
  method = 'BFGS' # unconstrained  
)  
exp(results$par) # Theta  
  
[1] 0.9728029 1.9806131
```


Multinomial Cell probabilities

- In this example, we are bound by a curve, not a region. Lagrange Multipliers are a good analytic approach.
- Consider fitting a multinomial distribution to a frequency table.
- We let X_1, \dots, X_m be the counts in cells $1, \dots, m$, and we assume that (X_1, \dots, X_m) follows a multinomial distribution.
- The distribution has parameters n : total counts, and p_i probability of cell i .
- The total count $n = \sum_{i=1}^m X_i$, and p_i being the
- In this case, the X_i are *not* independent, so we don't just take product of IID observations.

Multinomial Cell probabilities II

- However, the likelihood can be calculated via the pmf of the multinomial distribution:

$$L(\theta) = f(x_i^*; \theta) = \frac{n!}{\prod_{i=1}^m x_i!} \prod_{i=1}^m p_i^{x_i^*}.$$

- Given X_1, \dots, X_m , the number of trials n is known. The parameter vector of interest is then:

$$\theta = (p_1, \dots, p_m).$$

- The log-likelihood is given by:

$$\ell(\theta) = \log n! - \sum_{i=1}^m \log x_i! + \sum_{i=1}^m x_i \log p_i.$$

Multinomial Cell probabilities III

- Note that we have the constraint $\sum_i p_i = 1$. Thus, we will solve this using the Lagrange-multiplier technique.
- Reminder: Suppose we are trying to maximize (or minimize) a function $f(\theta)$, with the constraint $g(\theta) = 0$. That is,

$$\begin{aligned} & \max_{\theta} f(\theta) \\ & \text{subject to } g(\theta) = 0 \end{aligned}$$

- This optimization problem can be solved by introducing the Lagrange function:

$$\mathcal{L}(\theta, \lambda) = f(\theta) + \lambda g(\theta),$$

Multinomial Cell probabilities IV

- And solving the system of equations that arises from:

$$\nabla_{\theta, \lambda} \mathcal{L}(\theta, \lambda) = 0.$$

- Recall the partial derivative $\frac{\partial}{\partial \lambda} \mathcal{L}(\theta, \lambda) = g(\theta)$, which must be zero, giving the necessary constraint.
- Then, the condition $\nabla_{\theta} \mathcal{L}(\theta, \lambda) = 0$ ensures that the gradients of f and g are parallel. That is, geometrically, the constraint $g(\theta) = 0$ defines a curve or surface that we are constrained to; on this surface, the max/min value of f occurs when a level-set of f is tangent to the constraint curve $g(\theta) = 0$. This occurs only where $\nabla f(\theta) = \lambda \nabla g(\theta)$, which gives rise to the set of equations defined by $\nabla_{\theta} \mathcal{L}(\theta, \lambda) = 0$.

Multinomial Cell probabilities V

MLE of multinomial cell probabilities

Use Lagrange multipliers to find the MLE of a multinomial distribution, with X_1, \dots, X_m .

Multinomial Cell probabilities VI

- Often, we can describe parameters of a model as functions of other parameters.
- Example: in the multinomial model, we have parameters p_i representing probabilities for each cell. We might want to make these a function of θ , such that $p_i(\theta)$; then, we will still want to estimate θ using the data.
- In the case above the log-likelihood is then:

$$\ell(\theta) = \log n! - \sum_{i=1}^m \log x_i^*! + \sum_i i = 1^m x_i \log p_i(\theta).$$

- TODO: Finished 8.5.1 Example A, RICE.

Hardy-Weinberg Equilibrium

- Consider alleles expressed as dominant (A) and recessive (a), and they are expressed with rates $1 - \theta$ and θ , respectively.
- The Hardy-Weinberg principle states genotypes are expressed in large, randomly mixing populations following the Punnett square in Table 1.

	A	a
A	$(1 - p)^2$	$p(1 - p)$
a	$p(1 - p)$	p^2

Table 1: Genotype frequencies in large, randomly mixing population.

Hardy-Weinberg Equilibrium II

- Since the order of Aa or aA doesn't matter, this results in frequencies for AA , Aa , and aa : $(1 - \theta)^2$, $2\theta(1 - \theta)$, θ^2 .
- Supposing that we observe a population with these three possible gene expressions. Then, using population counts, we want to fit a multinomial model with $m = 3$ categories, and probabilities defined as:

$$p_1 = (1 - \theta)^2, \quad p_2 = 2\theta(1 - \theta), \quad p_3 = \theta^2.$$

Hardy-Weinberg Equilibrium III

Blood Types

In a Chinese population of Hong Kong in 1937, blood types occurred with the following frequencies, where M and N are erythrocyte antigens: M (342), MN (500), N (187), for a total of 1029 samples. Use a multinomial model to estimate the frequency of alleles and genotypes.

Example: logistic regression

- TODO.

References and Acknowledgements

Pawitan Y (2001). *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press.

Rice JA (2007). *Mathematical statistics and data analysis*, volume 371. 3 edition. Thomson/Brooks/Cole Belmont, CA.

- Compiled on January 25, 2026 using R version 4.5.1.
- Licensed under the [Creative Commons Attribution-NonCommercial](#) license. Please share and remix non-commercially, mentioning its origin.



References and Acknowledgements II

- We acknowledge [students and instructors for previous versions of this course / slides](#).