MATH 680 Computer Intensive Statistics

Dr David A. Stephens

Department of Mathematics & Statistics
Room 1225, Burnside Hall

d.stephens@math.mcgill.ca
www.math.mcgill.ca/~dstephens/680/

Computational Methods in Statistics

Modern statistics is only possible because of advances in computational power. Virtually all (applied) statistics relies on

- Statistics packages with GUIs: Splus, SPSS, Stata ...
- Command line programming: R, SAS ...
- High-level language programming: C/C++ ...

to implement statistical methods.

Course content

This course will focus on two sorts of numerical computational methods:

- **Deterministic**
  - Optimization
  - Root-finding
- **Stochastic**
  - Random number generation
  - Bootstrap
  - Monte Carlo
  - Markov chain Monte Carlo

In addition, standard computational/numerical methods will be covered (solving linear systems, matrix inversion, Cholesky and other matrix decompositions etc.)

The need for computation

Estimation is a common statistical activity:

- Maximum likelihood (ML)
- Method of Moments (MM)
- (Generalized) Estimating Equations ((G)EE)
- Bayesian estimation

When analytical results are not available, the first three require either **optimization** or **root-finding**; the fourth typically requires **integration**.

Example: ML estimation

For an iid sample $\boldsymbol{x}$ from parametric density $f(x|\theta)$, the ML estimate is

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} L(\theta; \boldsymbol{x})$$

(i.e. obtained by optimization). However if

$$\ell_\theta(\theta; \boldsymbol{x}) = \frac{\partial}{\partial \theta} \log L(\theta; \boldsymbol{x})$$

under regularity conditions, we have that $\widehat{\theta}$ is the solution of the equation

$$\ell_\theta(\theta; \boldsymbol{x}) = \mathbf{0}$$

(i.e. obtained by root-finding).

If θ is a $(p \times 1)$ vector, then this is a p-dimensional optimization or equation system.

Notation

A note on notation: derivatives will be denoted by subscripts, that is, for function g, a function of d-dimensional vector t,

$$g_t(t) = \frac{\partial g(t)}{\partial t} \qquad (d \times 1 \text{ vector})$$

$$g_{tt}(t) = \frac{\partial^2 g(t)}{\partial t \partial t^\intercal} \qquad (d \times d \text{ matrix})$$

etc. The notation

$$\dot{g}(t) \qquad \ddot{g}(t)$$

etc. will also be used.

Example: MM estimation

For MM estimation, if

$$\mu_k(\theta) = \mathbb{E}_{f_{X|\theta}}[X^k]$$

then the MM estimate is the solution of the system

$$\sum_{i=1}^{n}(X_i^k - \mu_k(\theta)) = 0 \qquad k = 1, \ldots, p$$

(i.e. obtained by root-finding).

Example: Logistic regression

Suppose data comprise n data

- Response $Y \in \{0, 1\}$
- Covariate Predictors $X_1, \ldots, X_p$

where

$$\theta_i = \mathbb{P}[Y_i = 1 | \boldsymbol{X}_i = \boldsymbol{x}_i]$$

is modelled via

$$\eta_i = \log\left(\frac{\theta_i}{1-\theta_i}\right) = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} = \boldsymbol{X}_i^\intercal \beta$$

say; write

$$\theta_i \equiv \theta_i(\beta).$$

Example: Logistic regression

The corresponding likelihood is

$$L(\beta; \boldsymbol{x}) = \prod_{i=1}^{n} \left\{ \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}^{Y_i} \left\{ \frac{1}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}^{1-Y_i}$$

with log-likelihood

$$\ell(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} Y_i \boldsymbol{x}_i^\mathsf{T} \beta - \sum_{i=1}^{n} \log\left(1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}\right)$$

To compute the ML estimate, take partial derivatives wrt the components of β in turn: for $j = 1, \ldots, p$

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} Y_i X_{ij} - \sum_{i=1}^{n} \left\{ X_{ij} \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}$$

Example: Logistic regression

This simplifies to β: for $j = 1, \ldots, p$

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \left\{ \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\} \right)$$

or

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \theta_i(\beta) \right)$$

For β_0, the equation is

$$\ell_{\beta_0}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} \left(Y_i - \theta_i(\beta) \right)$$

Example: Logistic regression

Thus the system of equations that must be solved to find the ML estimate is the $(p+1) \times 1$ system

$$\sum_{i=1}^{n} \eta_{\beta i}(\beta)(Y_i - \theta_i(\beta)) = \mathbf{0}$$

where $\eta_{\beta i}(\beta)$ is the $(p+1) \times 1$ vector with jth component

$$\eta_{\beta ij}(\beta) = \frac{\partial \eta_i(\beta)}{\partial \beta_j}.$$

for $j = 0, 1, \ldots, p$.

The solution is not analytically available, and numerical methods are required.

Example: Logistic regression

Standard errors are typically also required. In regular problems, as $n \longrightarrow \infty$,

$$\sqrt{n}(\widehat{\beta} - \beta_0) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}(\beta_0)^{-1})$$

where β_0 is the true value of β, and

$$\mathcal{I}(\beta_0) = \mathbb{E}\left[S_\beta S_\beta^\intercal\right] = -\mathbb{E}\left[S_{\beta\beta}\right]$$

is the Fisher Information with

$$S_\beta \equiv S_\beta(X;\beta) = \frac{\partial \log f(X|\beta)}{\partial \beta} \qquad (p+1) \times 1$$

$$S_{\beta\beta} \equiv S_{\beta\beta}(X;\beta) = \frac{\partial^2 \log f(X|\beta)}{\partial \beta \partial \beta^\intercal} \qquad (p+1) \times (p+1)$$

and expectations are taken at $\beta = \beta_0$.

Example: Logistic regression

$\mathcal{I}(\beta_0)$ is typically estimated by

$$\widehat{I}(\widehat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} S_\beta(x_i;\widehat{\beta})S_\beta^\intercal(x_i;\widehat{\beta})$$

or

$$\widehat{I}(\widehat{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} S_{\beta\beta}(x_i;\widehat{\beta})$$

Thus the standard errors are available relatively easily once $\widehat{\beta}$ has been obtained.

Taylor approximation

Recall that, by a Taylor expansion of the log-likelihood around $\theta = \widehat{\theta}$

$$\begin{aligned} \ell(\theta) &\simeq \ell(\widehat{\theta}) + \ell_\theta(\widehat{\theta})(\theta - \widehat{\theta}) + \frac{1}{2}(\theta - \widehat{\theta})^\intercal \ell_{\theta\theta}(\widehat{\theta})(\theta - \widehat{\theta}) \\ &= \ell(\widehat{\theta}) + \frac{1}{2}(\theta - \widehat{\theta})^\intercal \ell_{\theta\theta}(\widehat{\theta})(\theta - \widehat{\theta}) \end{aligned}$$

as $\ell_\theta(\widehat{\theta}) = \mathbf{0}$. The matrix

$$\ell_{\theta\theta}(\theta)$$

is the matrix of second partial derivatives of ℓ sometimes termed the **Hessian** matrix for ℓ. We typically need $\ell_{\theta\theta}(\widehat{\theta})$; this is merely $-n\widehat{I}(\widehat{\theta})$.

Taylor approximation

As $L(\theta; \boldsymbol{x}) = \exp\{\ell(\theta)\}$, it follows that for large n

$$L(\theta; \boldsymbol{x}) \doteq c(\widehat{\theta}) \exp\left\{-\frac{n}{2}(\theta - \widehat{\theta})^{\intercal} \widehat{I}(\widehat{\theta})(\theta - \widehat{\theta})\right\}$$

Hence in a Bayesian calculation

$$\pi(\theta|\boldsymbol{x}) \propto L(\theta; \boldsymbol{x})\pi(\theta)$$

and $L(\theta; \boldsymbol{x})$ can be approximated by a Gaussian likelihood.

The Laplace Approximation

Using a similar strategy: for an arbitrary unnormalized multivariate density $f(\boldsymbol{z}) = \exp\{h(\boldsymbol{z})\}$, say, if $\boldsymbol{z}^* \in \mathbb{R}^d$ corresponds to the mode of f,

$$\begin{aligned} h(\boldsymbol{z}) &\doteq h(\boldsymbol{z}^*) + \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}^*)^\intercal \ddot{h}(\boldsymbol{z}^*)(\boldsymbol{z} - \boldsymbol{z}^*) \\ &= h(\boldsymbol{z}^*) - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}^*)^\intercal H(\boldsymbol{z}^*)(\boldsymbol{z} - \boldsymbol{z}^*) \end{aligned}$$

say. Therefore

$$f(\boldsymbol{z}) \doteq f(\boldsymbol{z}^*) \exp\left\{ -\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}^*)^\intercal H(\boldsymbol{z}^*)(\boldsymbol{z} - \boldsymbol{z}^*) \right\}$$

and

$$\int f(\boldsymbol{z})\, d\boldsymbol{z} \doteq f(\boldsymbol{z}^*) \frac{(2\pi)^{d/2}}{|\det H(\boldsymbol{z}^*)|^{1/2}}$$

The Laplace Approximation: Example

If $f(z) = z^{\alpha-1}e^{-z}$, then $h(z) = (\alpha-1)\log z - z$ and

$$\begin{aligned} \dot{h}(z) &= \frac{\alpha-1}{z} - 1 \qquad \therefore \qquad z^* = \alpha - 1 \\ H(z) &= -\ddot{h}(z) = \frac{(\alpha-1)}{z^2} \qquad \therefore \qquad H(z^*) = 1/(\alpha-1) \end{aligned}$$

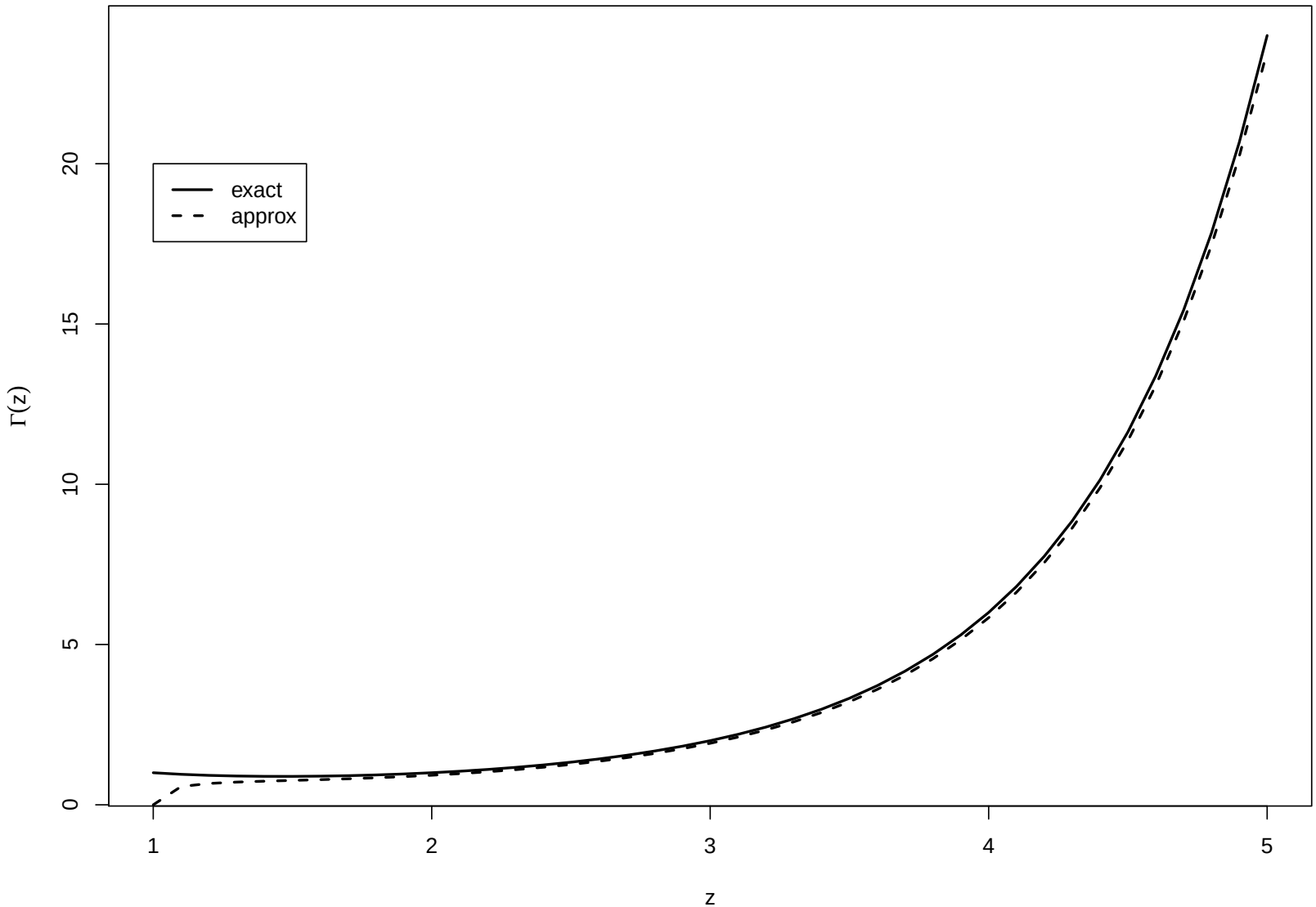
so therefore

$$\int f(\boldsymbol{z})\, d\boldsymbol{z} \doteq \sqrt{(2\pi)}(\alpha-1)^{\alpha-1/2}e^{-(\alpha-1)}$$

Here the normalizing constant is actually $\Gamma(\alpha)$.

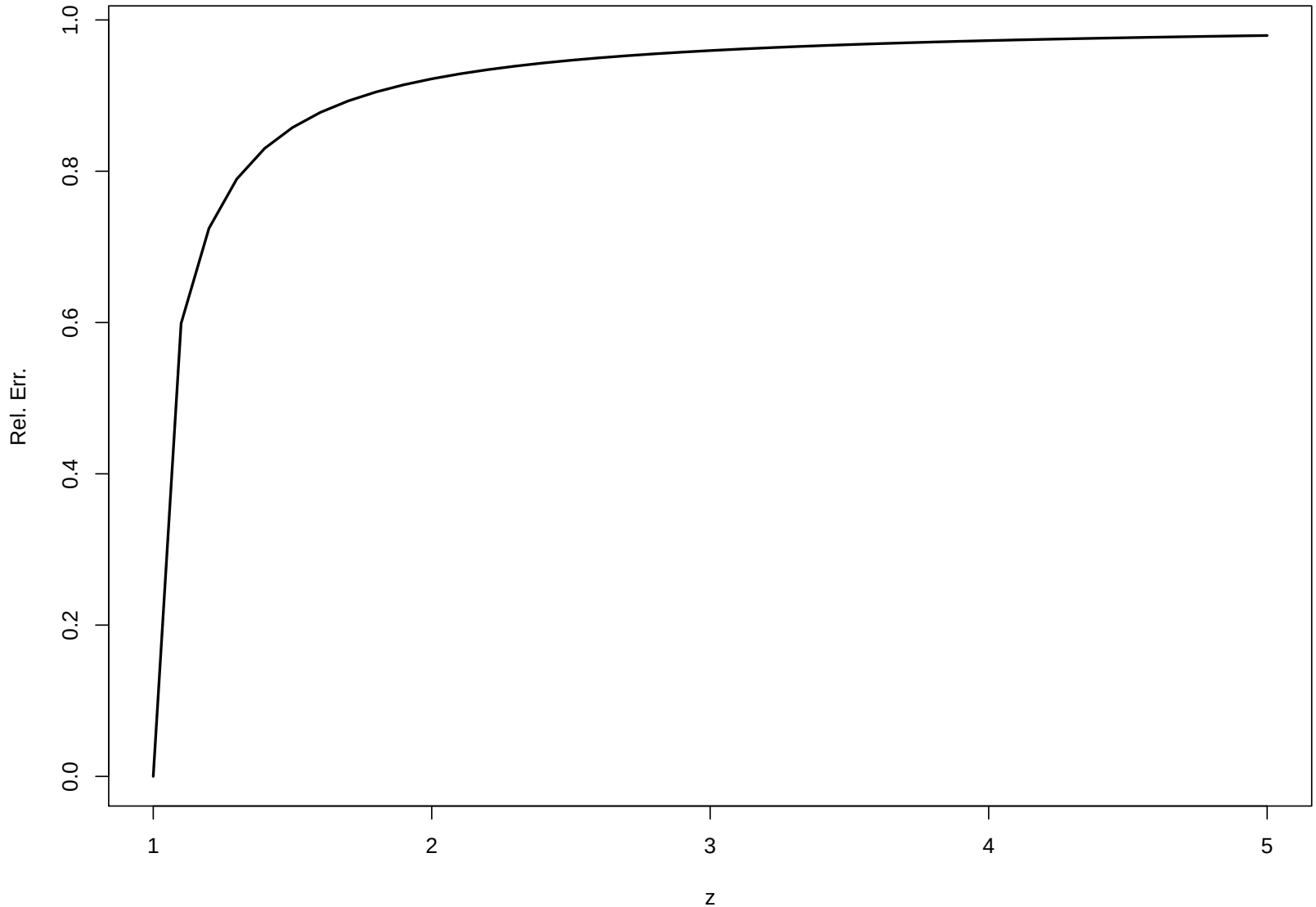
The Laplace Approximation: Example

Approximation:



The Laplace Approximation: Example

Relative error:



Linear Model

In the linear model

$$Y_i = \boldsymbol{X}_i^\intercal \beta + \epsilon_i$$

where $\epsilon_i, i = 1, \ldots, n$ are zero mean, variance σ^2 random variables that are mutually independent (and independent of $\boldsymbol{X}_i$).

Following the same derivation as above, to estimate β,

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \boldsymbol{x}_i^\intercal \beta \right) = 0$$

or equivalently, as a $(p+1) \times 1$ system

$$\boldsymbol{X}^\intercal \boldsymbol{X} \beta = \boldsymbol{X}^\intercal \boldsymbol{Y}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\intercal$, and $\boldsymbol{X}$ is the $n \times (p+1)$ matrix formed by binding the rows $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

Linear Model

Provided $\boldsymbol{X}^\intercal \boldsymbol{X}$ is non-singular (positive definite), this equation has a unique solution

$$\widehat{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

To compute this, we need

$$(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$$

i.e. we need to invert a $(p+1) \times (p+1)$ matrix. This is feasible (usually) using, say

`solve`

in R.

Cholesky decomposition

Alternatively:

- **Cholesky decomposition:** For a positive definite matrix A, write

$$A = LL^\intercal$$

  where L is a lower-triangular matrix. A^{-1} is more readily available compute as L^{-1} is easy to compute.

  We need to apply this to $\boldsymbol{X}^\intercal \boldsymbol{X}$.

Singular value decomposition

Suppose that A is an $m \times n$ real matrix. Then A can be decomposed

$$A = UDV^\intercal$$

where

- U is $m \times m$ such that $U^\intercal U = UU^\intercal = \boldsymbol{I}_m$
- D is $m \times n$ with non-negative elements only on the leading diagonal.
- V is $n \times n$ such that $V^\intercal V = VV^\intercal = \boldsymbol{I}_n$

Singular value decomposition

If A is $m \times m$ and **positive definite**, then D is square and has the m strictly positive **eigenvalues** of A on its diagonal.

In this case U comprises the **eigenvectors** of A as its columns, and

$$U^{-1} = V^{\intercal}$$

and

$$A^{-1} = UD^{-1}V^{\intercal}$$

QR decomposition

- **QR decomposition:** Write

$$\boldsymbol{X} = QR$$

where

  - Q is a $n \times (p+1)$ orthonormal matrix, $Q^\intercal Q = \boldsymbol{I}$,
  - R is a $(p+1) \times (p+1)$ upper-triangular matrix

Then

$$\boldsymbol{X}^\intercal \boldsymbol{X}\beta = \boldsymbol{X}^\intercal \boldsymbol{Y} \qquad \therefore \qquad R^\intercal Q^\intercal QR\beta = R^\intercal Q^\intercal \boldsymbol{Y}$$

and if $Q^\intercal \boldsymbol{Y} = \boldsymbol{Y}^*$, say, then

$$R\beta = \boldsymbol{Y}^*$$

This is easy to solve, as R is triangular.

QR decomposition

Gram-Schmidt orthogonalization: For $n \times p$ matrix A, write

$$A = \left[A_{(1)}, \ldots, A_{(p)}\right]$$

Then set

- At step 1: $B_{(1)} = A_{(1)}$ then $Q_{(1)} = B_{(1)}/||B_{(1)}||$.
- At step 2:

$$B_{(2)} = A_{(2)} - \frac{A_{(2)}^\intercal Q_{(1)}}{Q_{(1)}^\intercal Q_{(1)}} Q_{(1)} \qquad \text{then} \qquad Q_{(2)} = \frac{B_{(2)}}{||B_{(2)}||}$$

- At step $k = 3, \ldots, p$:

$$B_{(k)} = A_{(k)} - \sum_{j=1}^{k-1} \frac{A_{(k)}^\intercal Q_{(j)}}{Q_{(j)}^\intercal Q_{(j)}} Q_{(j)} \qquad \text{then} \qquad Q_{(k)} = \frac{B_{(k)}}{||B_{(k)}||}$$

Then

$$Q = \left[Q_{(1)}, \ldots, Q_{(p)}\right]$$

is the desired matrix.

QR decomposition

The matrix R is the upper-triangular matrix with (j,k)th element

$$Q_{(j)}^{\intercal} A_{(k)} \qquad j \leq k.$$

Thus we can compute

$$(\boldsymbol{X}^{\intercal}\boldsymbol{X})^{-1}$$

without matrix inversion. This is often faster than using matrix inversion, and is the default approach used by R in the function `lm`.

Note: Other methods of orthogonalization are available.

Example: Ridge Regression

If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is ill-conditioned, or almost singular, the **ridge regression** estimator is sometimes used:

$$\widetilde{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

for scalar λ. More generally, could use

$$\widetilde{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \Lambda^\intercal \Lambda)^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

This is the solution to the regularized minimization problem

$$\min_{\beta} \|Y - \boldsymbol{X}\beta\|^2 + \|\Lambda\beta\|^2$$

and is termed **Tikhonov regularization**.

Matrix Inverse Identity

If $\boldsymbol{X}$ is $n \times p$, $\boldsymbol{X}^\intercal \boldsymbol{X}$ is $p \times p$:

- Typically $n \gg p$, and p is reasonably small, and inversion is relatively straightforward.
- Sometimes $p \gg n$, so inverting the large, sparse matrix

$$\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I}$$

  is more challenging.

Matrix Inverse Identity

Suppose

- A is $p \times p$
- U is $p \times n$
- C is $n \times n$
- V is $n \times p$

Then

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

For ridge regression: set

$$A = \lambda \boldsymbol{I} \qquad U = \boldsymbol{X}^\intercal \qquad C = \boldsymbol{I} \qquad V = \boldsymbol{X}$$

so that

$$(\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} = \lambda^{-1}\boldsymbol{I} - \lambda^{-2}\boldsymbol{X}^\intercal(\boldsymbol{I} + \lambda^{-1}\boldsymbol{X}\boldsymbol{X}^\intercal)^{-1}\boldsymbol{X}$$

Matrix Inverse Identity

This identity allows us to exchange an $p \times p$ inversion for an $n \times n$ inversion; it will be useful when p is large and n is small.

- Note that in the general version

$$A = \Lambda^\intercal \Lambda \qquad \text{and} \qquad C$$

must be inverted.

Other forms of regularization

In certain statistical estimation problems, the form of "regularization" is different

- LASSO

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- L_α

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|^\alpha$$

- Elastic net

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2$$

Minimization cannot be achieved analytically.

Example: Gaussian Markov Random Field

A (stationary) **Gaussian Markov Random Field** is a probabilistic model describing the co-variation of random variables having different temporal or spatial locations.

At points $x_1, \ldots, x_n$ in 1- or 2-dimensions, the joint distribution of random quantities $Z_1, \ldots, Z_n$ associated with these locations is multivariate Gaussian

$$\mathbb{C}ov[Z_i, Z_j] = \sigma(||x_i - x_j||) = \Sigma_{ij}$$

that is

$$\boldsymbol{Z} \sim \mathcal{N}_n(\boldsymbol{\mu}(\boldsymbol{x}), \Sigma)$$

The covariance function $\sigma(.)$ is often specified parametrically:

$$\sigma(d) = \sigma^2 \exp\{-(d/\lambda)^\alpha\}$$

Example: Gaussian Markov Random Field

This model is used in spatial modelling.

- risk surface analysis; using a binomial logistic model

$$Y_i \sim Binomial(n_i, \theta_i) \qquad \theta_i = \frac{\exp\{Z_i\}}{1 + \exp\{Z_i\}}$$

  or a log-linear Poisson model

$$Y_i \sim Poisson(\theta_i) \qquad \theta_i = \exp\{Z_i\}$$

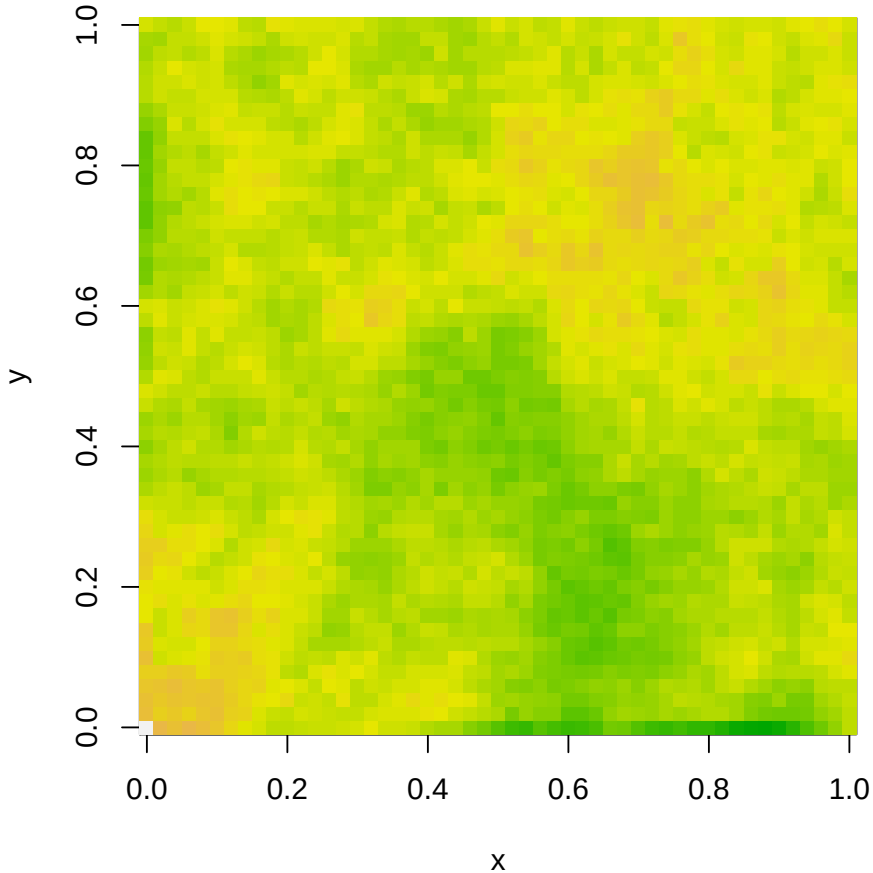
  can build a model for observed response data conditional on $\boldsymbol{Z}$.

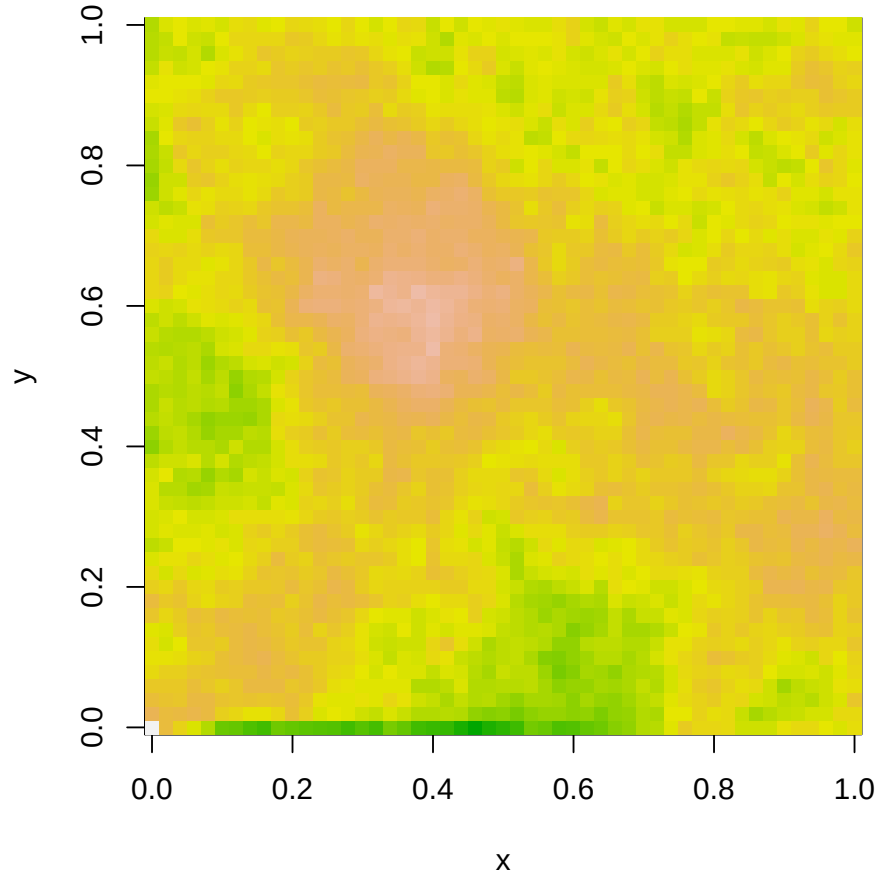
If n is fairly small, and the covariance function is reasonably well-behaved, then it is quite easy to study these models using simulation. However the nature of Σ can pose difficulties

- if n is large, and Σ is not sparse
- if Σ is almost singular

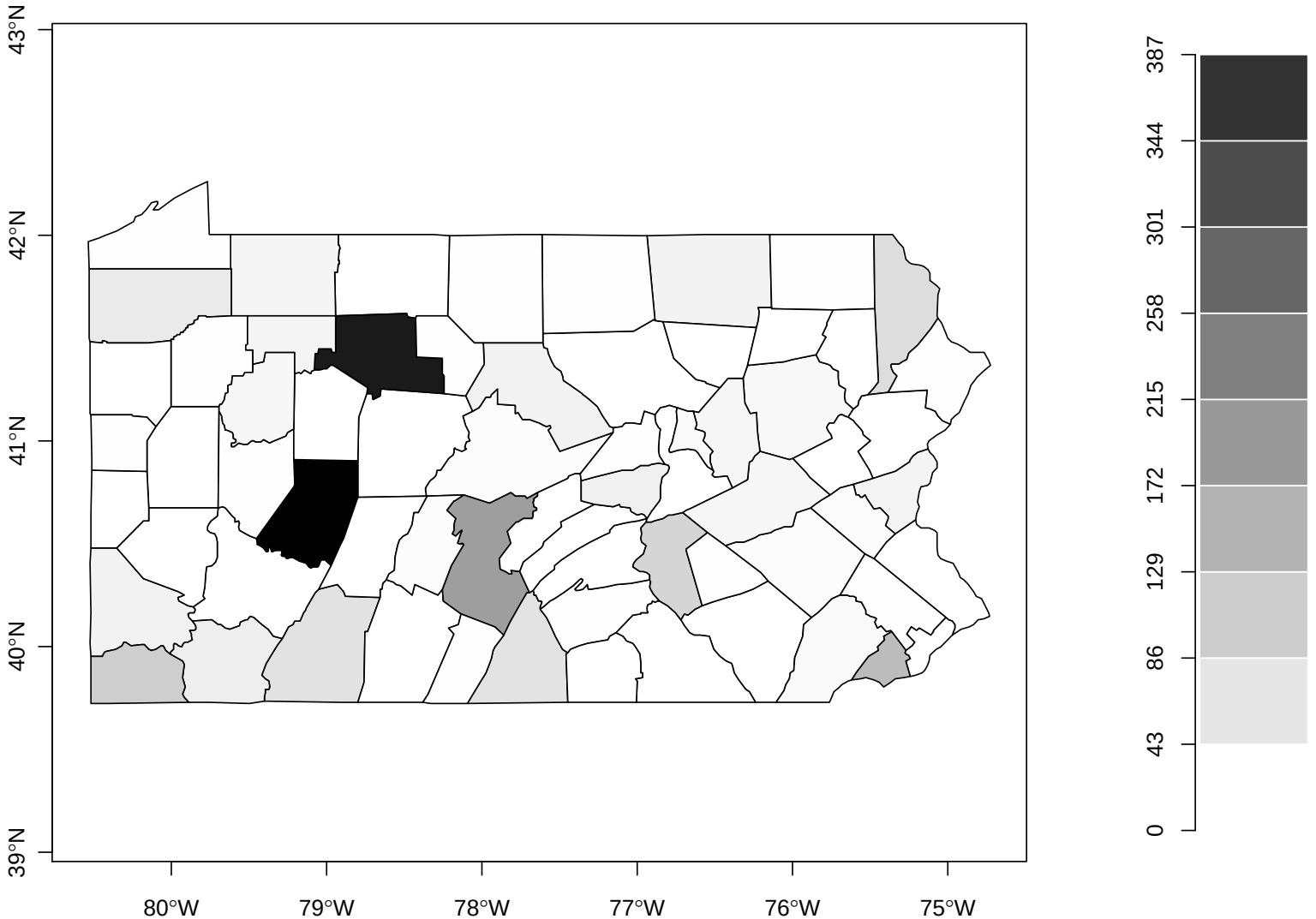
as a Cholesky decomposition of Σ is typically required.

Example: Gaussian Markov Random Field

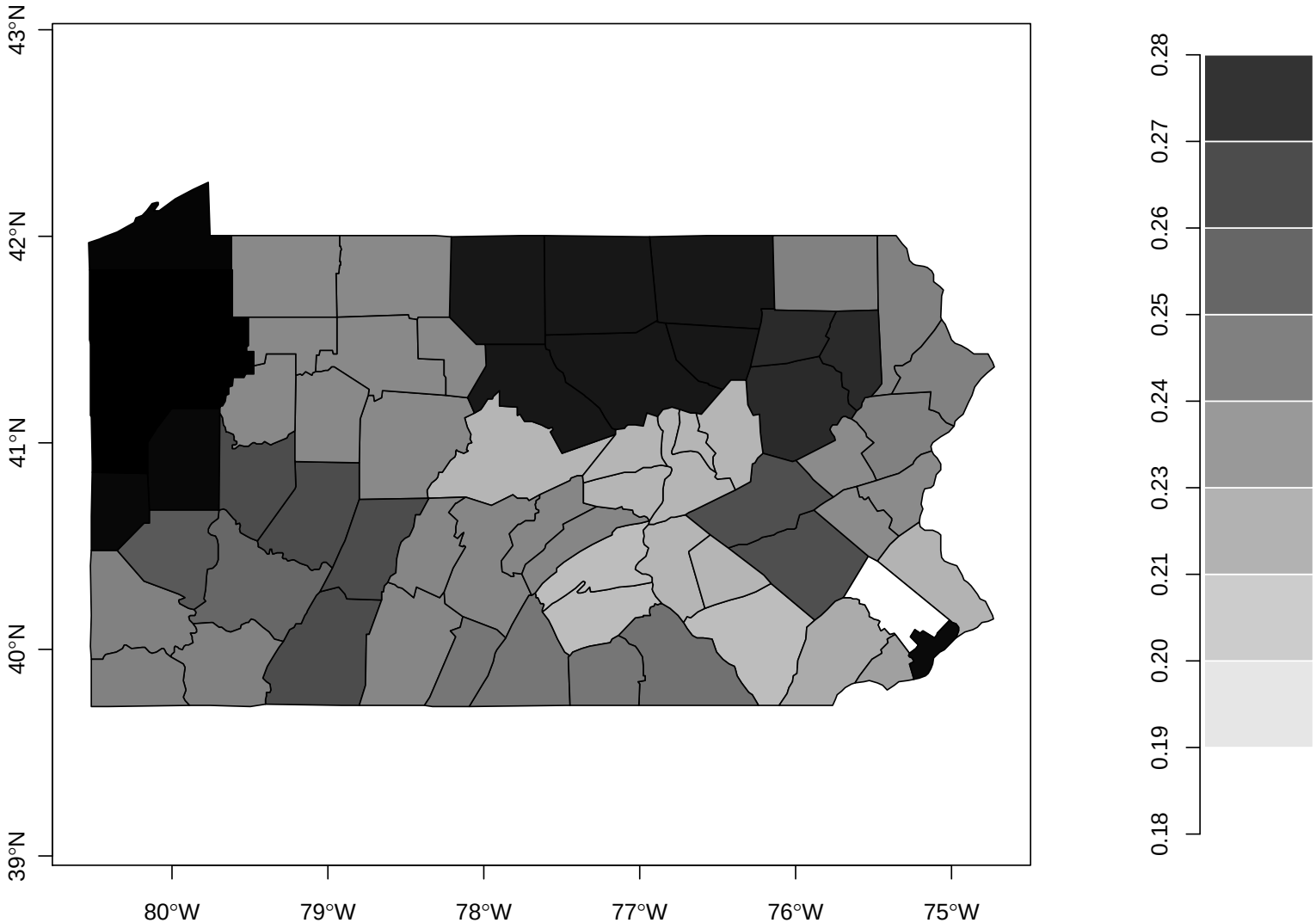




Example: Lung Cancer Cases in Pennsylvania



Example: Smoking rates in Pennsylvania



Example: Ising Model

Consider a collection of binary random variables $\{X_i\}$ placed on a rectangular $(N \times M)$ lattice, with joint mass function

$$f_{\mathbf{X}|\beta}(\boldsymbol{x}|\beta) = \frac{\exp\left\{\beta \sum_{i=1}^{NM} \sum_{j \neq i} \mathbb{1}_{x_i}(x_j)\right\}}{Z(\beta)}$$

where $Z(\beta)$ is the normalizing constant. The support of this mass function has

$$2^{NM}$$

elements. This model is used in statistical physics and image analysis; it is not analytically tractable, but its properties can be studied using **simulation** in a **Monte Carlo** approach.

Example: Ising Model

Simulation of variates from a **discrete** distribution $f(.)$ is in principle straightforward:

1. List the elements in the support of f: $x_1, x_2, \ldots$, with their associated probabilities $p_1, p_2, \ldots$, and cumulative probabilities $P_1, P_2, \ldots$

$$P_k = \sum_{j=1}^{k} p_j \qquad k = 1, 2, \ldots$$

2. Obtain a single random variate u from a $Uniform(0, 1)$ density.
3. Return the value $x = x_k$ as a simulated value from f, where

$$k = \max\{j : P_j \leq u\}$$

Example: Ising Model

This method is based on the inverse CDF transformation result

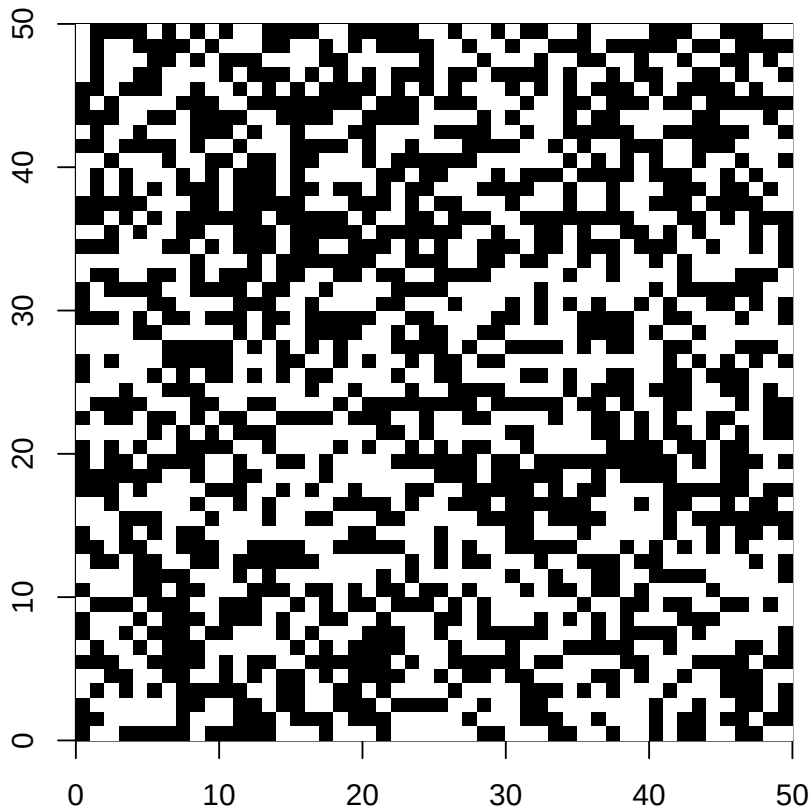
$$\text{If } X \text{ has CDF } F_X, \text{ then } U = F_X(X) \sim \mathit{Uniform}(0, 1).$$

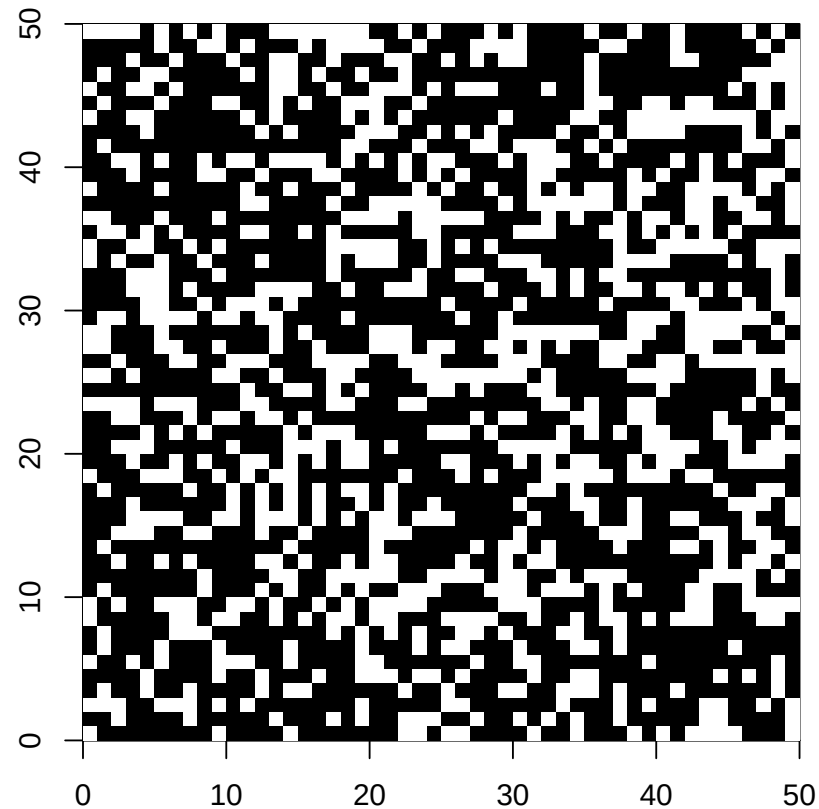
This Ising model mass function has a relatively complicated form; in particular the normalizing constant $Z(\beta)$ is difficult to compute, so that the probabilities $p_1, p_2, \ldots$ in the algorithm are only known up to proportionality.

Therefore, sampling from the Ising model is non-trivial.

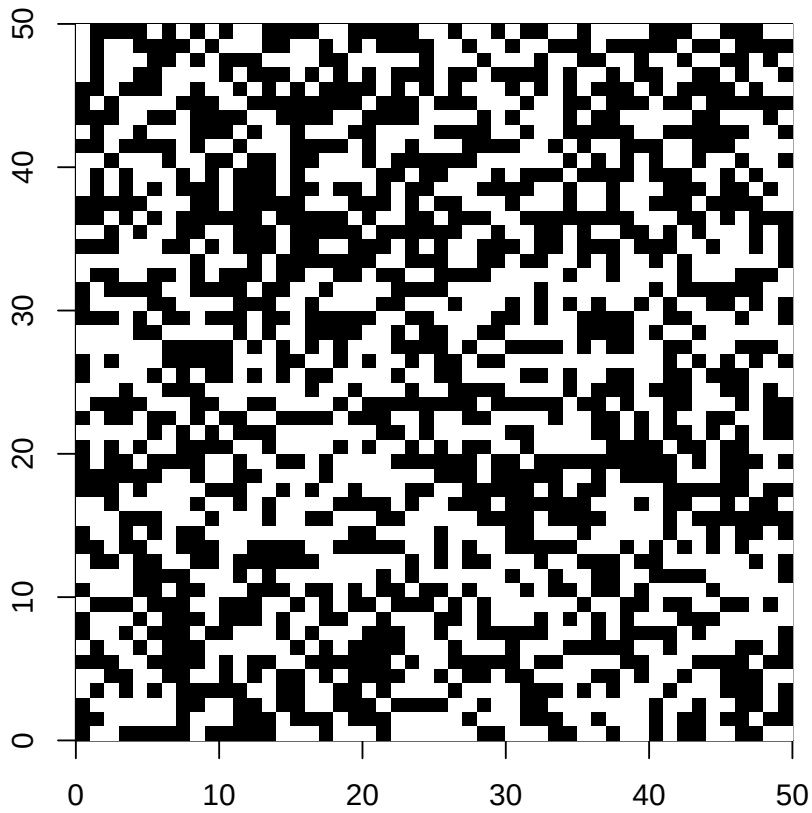
Also: how to achieve 2. ?

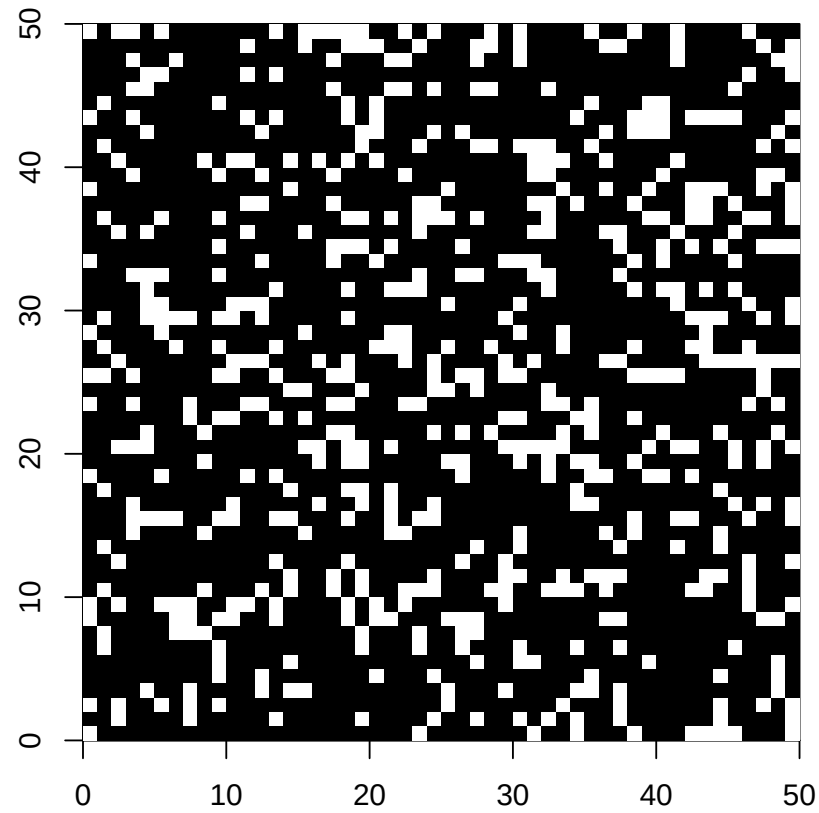
Ising Model: $\beta = 0.1$



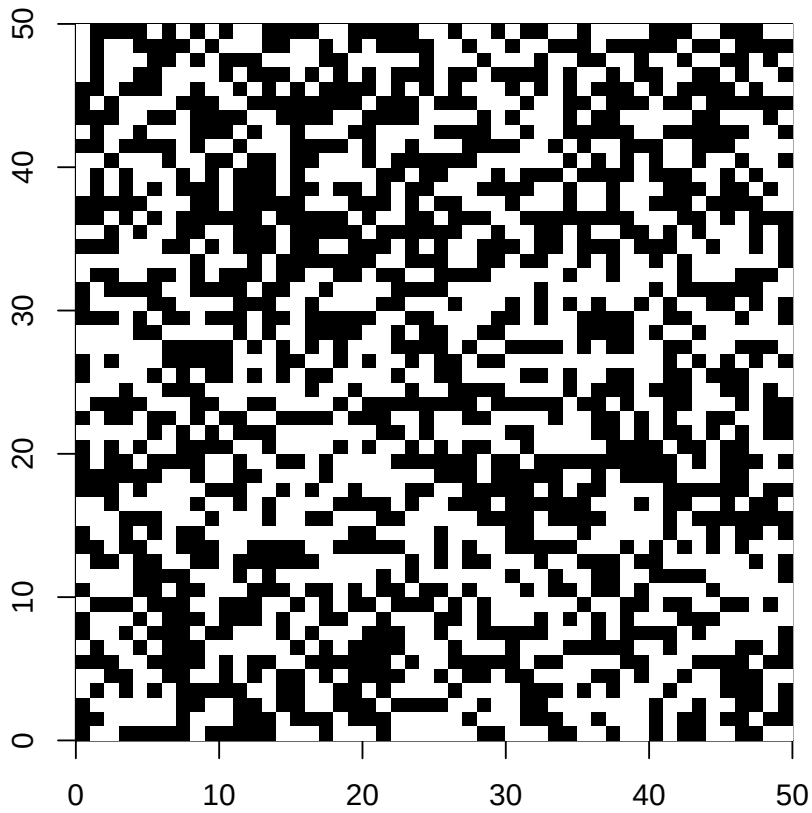


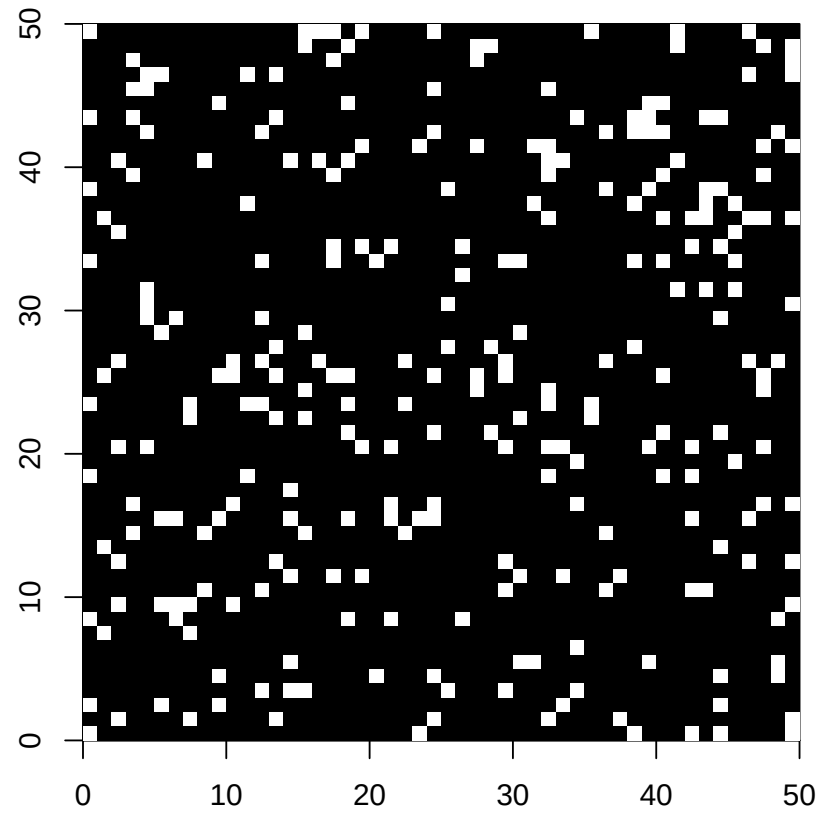
Ising Model: $\beta = 0.2$



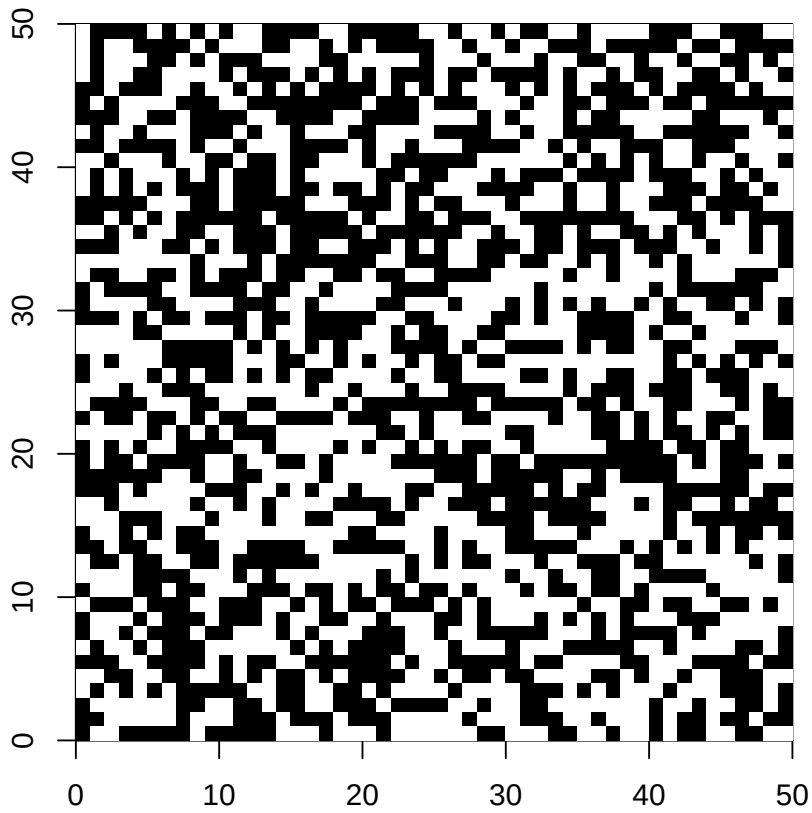


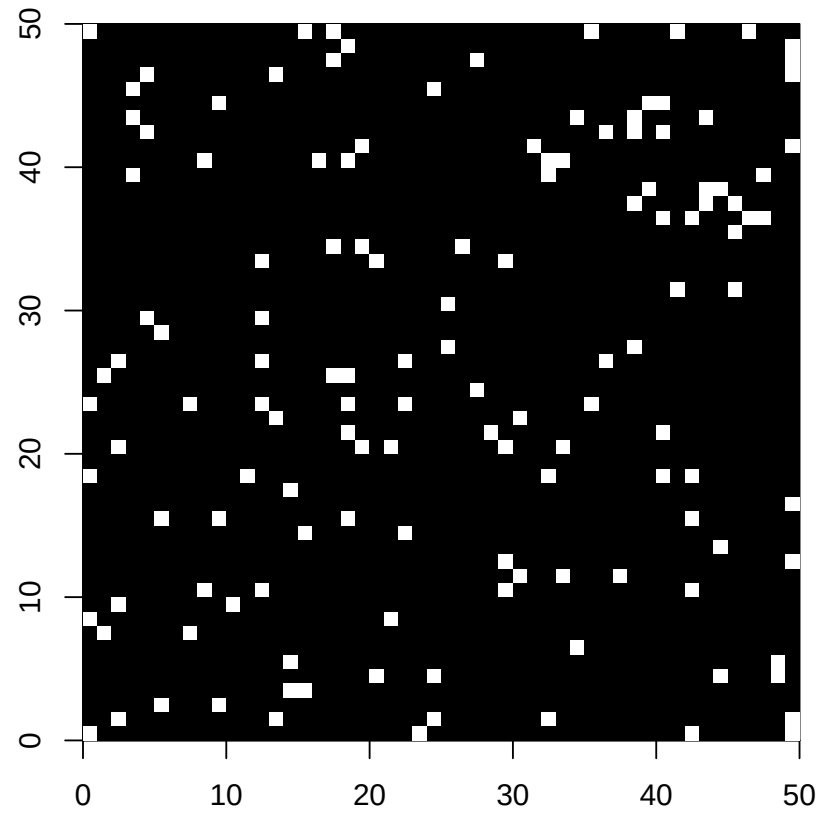
Ising Model: $\beta = 0.3$



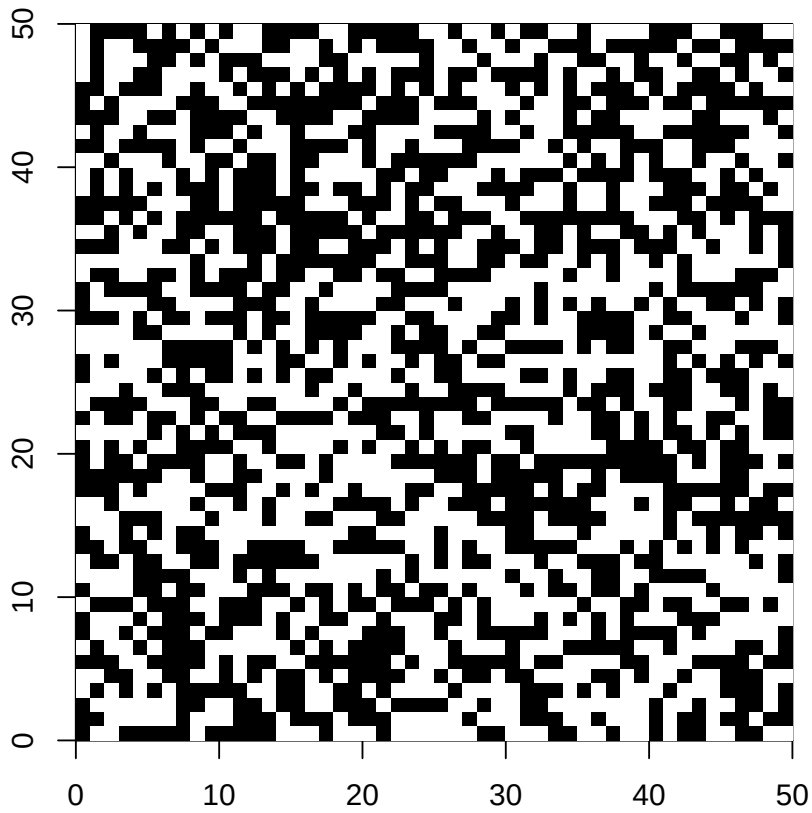


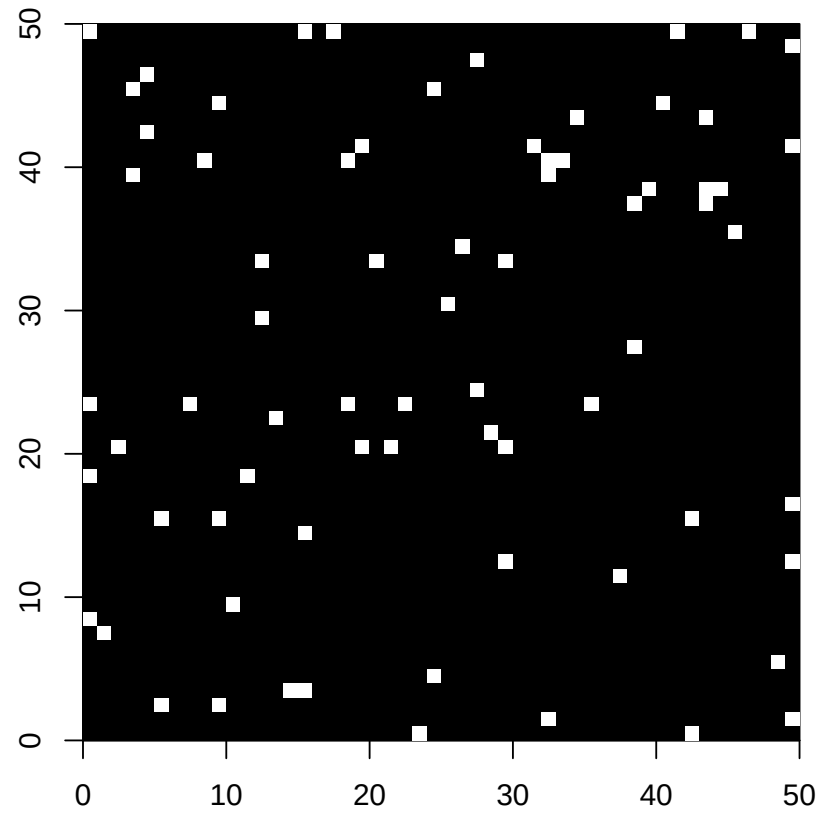
Ising Model: $\beta = 0.4$





Ising Model: $\beta = 0.5$





Example: The sampling distribution of a statistic

Suppose $X_1, \ldots, X_n$ is a random sample from a probability distribution with cdf F_X. Let $T = T(X_1, \ldots, X_n)$ be a statistic used to

- estimate some functional μ_{F_X} of F_X,
- test a hypothesis concerning F_X

The **sampling distribution** of T (perhaps under restrictions imposed by the hypothesis) is required:

- this is merely the probability distribution of T
- in a frequentist sense, this is the distribution of observed values of T over repeated samples of size n in presumed data collection situation.

Example: The sampling distribution of a statistic

If the form of F_X, and the value of the parameters in a parametric representation of F_X are known, we may examine the sampling distribution using **Monte Carlo**

1. Draw a sample $x_1, \ldots, x_n$ from F_X.
2. Compute and store $t = T(x_1, \ldots, x_n)$.
3. Repeat 1. and 2. a large number (N) of times, collecting the observed values
$$t_1, \ldots, t_N$$
4. Approximate f_T by a histogram estimator based on $t_1, \ldots, t_N$.

Example: The sampling distribution of a statistic

If the form of F_X is unknown, we may examine the sampling distribution using the **bootstrap**

1. **Resample** $x_1^{(b)}, \ldots, x_n^{(b)}$ from the original collection $x_1, \ldots, x_n$
   - For $i = 1, \ldots, n$, pick $x_i^{(b)}$ uniformly from $x_1, \ldots, x_n$.
   - Repeat this independently and with replacement.
   - This resampled collection will be similar to, but potentially slightly different from, the original collection; some original values may be repeated in, or omitted from, the resampled collection.
2. Compute the statistic $t^{(b)} = T(x_1^{(b)}, \ldots, x_n^{(b)})$
3. Repeat 1. and 2. a large number (B) of times, collecting the observed values $t_1^{(b)}, \ldots, t_B^{(b)}$.
4. Approximate f_T by a histogram estimator based on $t_1^{(b)}, \ldots, t_B^{(b)}$.

Example: The sampling distribution of a statistic

Example (Bootstrap for Gamma distribution median)

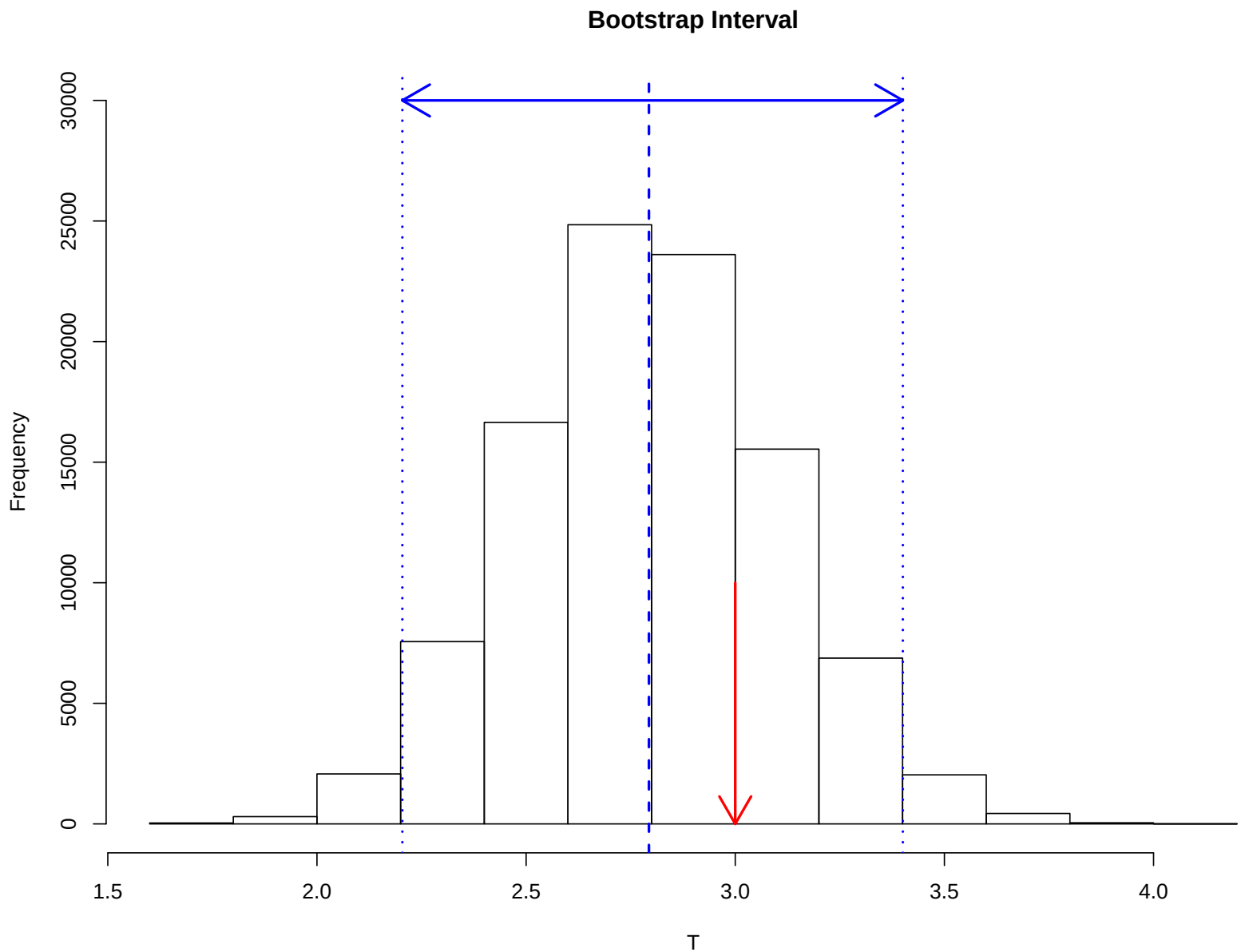
Suppose data

0.931	3.989	2.576	4.084	5.165
1.198	4.859	1.315	2.706	2.223
2.557	2.497	2.768	5.132	4.317
1.499	2.186	1.905	3.371	0.595

are believed to come from a Gamma distribution whose **mean** is the focus of interest. The natural estimator is the sample mean, 2.794.

We can construct a 95% confidence interval using the bootstrap.

Example: Bootstrap interval with $B = 100000$



Example: Bootstrap interval

We can tell if this is a good way to construct such an interval by carrying out a Monte Carlo simulation:

1. For a specific value (or collection of values) for α, generate a sample of size n from $Gamma(\alpha, 1)$.
2. Compute the sample mean.
3. Use the bootstrap to construct a 95% interval for α.
4. Check whether the true value of α lies in the interval.
5. Repeat 1. to 4. a large number of times, and compute the proportion of the repeats in which the interval **does** contain the true value - if the interval construction procedure is good, the proportion should be close to 0.95.

R Code

```
set.seed(423)
n<-20
al<-3
Nrep<-1000
nboot<-10000
tvec<-rep(0,nboot)
ico<-0
for(irep in 1:Nrep){
        X<-rgamma(n,al,1)
        for(i in 1:nboot){
                tvec[i]<-mean(sample(X,size=n,rep=T))
        }
        qv<-quantile(tvec,prob=c(0.025,0.975))
        if(al >= qv[1] & al <= qv[2]){ico<-ico+1}
}
print(ico/Nrep)
```

Example: State-space models

Consider the following model: for $t = 1, \ldots, T$

$$\begin{aligned} y_t &= g(x_t; \theta) + \epsilon_t \\ x_{t+1} &= h(x_t; \theta) + \xi_t \end{aligned}$$

where only the series $y_1, \ldots, y_T$, is observed. Sequences $\{\epsilon_t\}$ and $\{\xi_t\}$ are independent zero mean random variables.

This is termed a **state-space model** with state sequence $x_1, \ldots, x_T$. The state space sequence is a focus of interest, along with model parameters θ.

Example: State-space models

Let $\overline{y}_t = (y_1, \ldots, y_t)$, $\overline{x}_t = (x_1, \ldots, x_t)$. For the moment, suppress dependence on θ.

- **Filtering:** Compute $p(x_t|\overline{y}_t)$
- **Prediction:** Compute $p(x_{t+1}|\overline{y}_t)$
- **Smoothing:** Compute $p(x_t|\overline{y}_T)$

The first two steps carry out a sequential calculation **forwards** in time, utilizing basic probability results.

Example: State-space models

Filtering: By Bayes theorem

$$\begin{aligned} p(x_t|\overline{y}_t) &= \frac{p(y_t|x_t)p(x_t|\overline{y}_{t-1})}{p(y_t|\overline{y}_{t-1})} \\ &\propto p(y_t|x_t)p(x_t|\overline{y}_{t-1}) \end{aligned}$$

- The function $p(y_t|x_t)$ is usually a straightforward conditional density.
- The function $p(x_t|\overline{y}_{t-1})$ is the density computed at the previous time point **prediction** step.
- Note that the denominator (normalizing constant) is $p(y_t|\overline{y}_{t-1})$
- Recall that all of these densities potentially depend on θ.

Example: State-space models

Prediction: By marginalization

$$p(x_{t+1}|\overline{y}_t) \quad = \quad \int p(x_{t+1}|x_t)p(x_t|\overline{y}_t)\, dx_t$$

- The function $p(x_{t+1}|x_t)$ is usually a straightforward conditional density derived from the state equation.
- The function $p(x_t|\overline{y}_t)$ is the posterior density computed at the time point **filtering** step.
- This step requires an integration

Example: State-space models

Smoothing: Consider computing **backwards** from $t = T, T-1, \ldots, 1$:

$$\begin{aligned} p(x_t|\overline{y}_T) &= \int p(x_t, x_{t+1}|\overline{y}_T)\, dx_{t+1} \\ &= \int p(x_t|x_{t+1}, \overline{y}_T) p(x_{t+1}|\overline{y}_T)\, dx_{t+1} \\ &= \int p(x_t|x_{t+1}, \overline{y}_t) p(x_{t+1}|\overline{y}_T)\, dx_{t+1} \end{aligned}$$

where

$$p(x_t|x_{t+1}, \overline{y}_t) = \frac{p(x_{t+1}|x_t) p(x_t|\overline{y}_t)}{p(x_{t+1}|\overline{y}_t)}$$

Example: State-space models

- The function $p(x_{t+1}|x_t)$ is usually a straightforward conditional density derived from the state equation.
- The function $p(x_t|\overline{y}_t)$ is the posterior density computed at the relevant *filtering* step in the forward calculation.
- The function $p(x_{t+1}|\overline{y}_t)$ is the the density computed at the relevant *prediction* step in the forward calculation.
- The function $p(x_{t+1}|\overline{y}_T)$ is the density from the previous *smoothing* step in the recursion.
- This step requires an integration.

Example: State-space models

If the error terms ϵ_t and ξ_t are **Gaussian**, and the functions g and h are each **linear** in x_t, then all of the relevant densities

$$p(x_t|\overline{y}_t) \qquad p(x_{t+1}|\overline{y}_t) \qquad p(x_t|\overline{y}_T) \qquad p(x_{t+1}|\overline{y}_T)$$

can be computed analytically, and are Gaussian, so that the distributions are characterized by mean and variance.

If the error terms ϵ_t and ξ_t are **non-Gaussian**, or the functions g and h are **non-linear** in x_t, then the densities can typically only be computed numerically.

A typical approach uses **Sequential Monte Carlo**, in which sample-based approximations to the densities are used.

Example: State-space models

The above calculations presume that θ is known. Note that the likelihood $L(\theta; \overline{y}_T)$ can be expressed

$$L(\theta; \overline{y}_T) = p(y_1; \theta) \prod_{t=1}^{T-1} p(y_{t+1} | \overline{y}_t; \theta)$$

that is, in terms of quantities that are computed using the filtering and prediction steps.

Therefore the filtering approach can be used to facilitate inference about θ.

Part I

Optimization

Optimization in statistical problems

Optimization arises in many statistical problems:

- *minimization* in least-squares, penalized least-squares, model selection etc.
- *maximization* in maximum likelihood estimation, REML, optimal decision making etc.

We will focus on optimization of continuous and differentiable functions of vector parameters. Discrete optimization problems do arise, but are less common.

Optimization in statistical problems

We seek to optimize a scalar function $g(\theta)$ of vector parameter θ

$$\theta = (\theta_1, \ldots, \theta_p)^\intercal.$$

The components of θ may be

- *unconstrained*: all $\theta_j \in \mathbb{R}$
- *constrained*: $\theta_j \in [a_j, b_j]$, or $\varphi(\theta) \in [a, b]$ etc.

Recommendation:

It is usually easier to solve the **unconstrained** version, so where possible try to reparameterize onto $\mathbb{R}^p$.

Some basic facts about optimization

Consider $g(\theta)$ defined on the domain Θ. Usually, g will be continuous and bounded on Θ. Then

- **either** $g(\theta)$ attains its minimium/maximum value at value where

  $$\dot{g}(\theta) = \mathbf{0}$$

  where $\dot{g}(\theta)$ is the $p \times 1$ vector with jth component

  $$\frac{\partial g(\theta)}{\partial \theta_j}$$

- **or** $g(\theta)$ attains its minimium/maximum value on the boundary of Θ.

Some basic facts about optimization

The function $g(\theta)$ is **convex** if, for all θ, the $p \times p$ matrix

$$\ddot{g}(\theta) = \frac{\partial^2 g(\theta)}{\partial\theta\partial\theta^\intercal} = \left[\frac{\partial^2 g(\theta)}{\partial\theta_j\partial\theta_k}\right]_{jk}$$

is positive definite. Another characterization of convexity is that for $\theta_1, \theta_2 \in \Theta$ and $0 \leq \lambda \leq 1$.

$$g(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda g(\theta_1) + (1-\lambda)g(\theta_2)$$

A convex function has a **unique** minimum (if the minimum exists).

Strategies for Numerical Optimization

Basic components of a numerical optimization procedure:

- *approximation*: approximate a complicated function via a simpler and more tractable one.
- *recursion*: refine an initial guess at the optimizing parameter by using repeated adjustment.
- *stopping*: decide when the refinement process can stop.

We focus on estimation of parameter θ via a parametric log likelihood $\ell(\theta)$.

Newton's Method

A second-order Taylor expansion of the log-likelihood around a "current" (at step n) point, $\theta = \theta_n$ yields

$$\ell(\theta) \doteq \ell(\theta_n) + \dot{\ell}(\theta_n)(\theta - \theta_n) + \frac{1}{2}(\theta - \theta_n)^\intercal \ddot{\ell}(\theta_n)(\theta - \theta_n)$$

Using this approximation, consider trying to maximize the right-hand side; we would differentiate wrt θ, and equate to zero. Therefore, to generate the "next" point

$$\dot{\ell}(\theta) \doteq \dot{\ell}(\theta_n) + \ddot{\ell}(\theta_n)(\theta - \theta_n) = \mathbf{0}$$

can be solved for θ to yield the next point in the recursion, θ_{n+1},

$$\theta_{n+1} = \theta_n - \{\ddot{\ell}(\theta_n)\}^{-1}\dot{\ell}(\theta_n)$$

(recall, $\dot{\ell}$ is $p \times 1$, $\ddot{\ell}$ is $p \times p$).

Newton's Method

The recursion continues until $\|\theta_{n+1} - \theta_n\|$ is small enough.

A stationary point, θ^*, of this algorithm, which satisfies

$$\theta^* = \theta^* - \{\ddot{\ell}(\theta^*)\}^{-1}\dot{\ell}(\theta^*)$$

is a local optimum of ℓ. The algorithm requires computation of

$$\{\ddot{\ell}(\theta^*)\}^{-1}$$

and also may be slow to converge (if it converges at all).

In addition, there is no guarantee that

$$\ell(\theta_{n+1}) > \ell(\theta_n)$$

(i.e. it is not an *ascent* algorithm).

Newton's Method

Replacing $-\ddot{\ell}(\theta^*)$ by an approximating non-singular matrix H_n say can produce an ascent algorithm. From the previous expansion

$$\triangle\theta_n = \theta_{n+1} - \theta_n = H_n^{-1}\dot{\ell}(\theta_n)$$

and hence, for $s > 0$, by another Taylor expansion

$$\ell(\theta_n + s\triangle\theta_n) - \ell(\theta_n) \doteq s\dot{\ell}^{\intercal}(\theta_n)H_n^{-1}\dot{\ell}(\theta_n) \geq 0$$

as H_n is positive definite, provided s is made small enough.

Trying $s = 1/2, 1/4, \ldots$ usually can produce an increase in ℓ.

Weighted Least-Squares

Example (Weighted Least-Squares)

Consider estimating parameters θ appearing in the possibly non-linear regression model

$$Y_i = \mu_i(\theta) + \epsilon_i \qquad \mathbb{E}[\epsilon_i] = 0$$

by weighted least-squares. We seek to minimize the function

$$g(\theta) = \frac{1}{2} \sum_{i=1}^{n} w_i (y_i - \mu_i(\theta))^2$$

Weighted Least-Squares

Example (Weighted Least-Squares)

Therefore

$$\begin{aligned}
\dot{g}(\theta) &= -\sum_{i=1}^{n} w_i(y_i - \mu_i(\theta))\dot{\mu}_i(\theta) \\
\ddot{g}(\theta) &= \sum_{i=1}^{n} w_i\mu_i(\theta)\dot{\mu}_i(\theta)\dot{\mu}_i(\theta)^\intercal - \sum_{i=1}^{n} w_i(y_i - \mu_i(\theta))\ddot{\mu}_i(\theta) \\
&\doteq \sum_{i=1}^{n} w_i\mu_i(\theta)\dot{\mu}_i(\theta)\dot{\mu}_i(\theta)^\intercal
\end{aligned}$$

if either

(i) $w_i(y_i - \mu_i(\theta)) \approx 0$ or

(ii) $\ddot{\mu}_i(\theta) \approx \mathbf{0}$

Weighted Least-Squares

Example (Weighted Least-Squares)

In a recursion, given an initial fit, it is quite likely that

$$w_i(y_i - \mu_i(\theta)) \approx 0$$

so (i) holds. Alternatively, (ii) is true if the function $\mu(\theta)$ is approximately linear in θ.

Therefore the approximation

$$\ddot{g}(\theta) \doteq H(\theta) = \sum_{i=1}^{n} w_i \mu_i(\theta) \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal$$

in the algorithm seems like a good one; note that $H(\theta)$ is positive (semi) definite as it is formed from outer products of the $\dot{\mu}_i(\theta)$.

The Gauss-Newton Algorithm

The approximation method above is a form of the **Gauss-Newton** algorithm. It avoids the calculation of second derivatives of g, and does not rely on specification of a step-size s.

This approximation is useful for estimation in GLMs.

Poisson GLM

Example (Poisson GLM)

Suppose $Y_i \sim Poisson(\mu_i(\theta))$ for $i = 1, \ldots, n$.

$$\ell(\theta) = \sum_{i=1}^{n} \{y_i \log \mu_i(\theta) - \mu_i(\theta)\} + \text{ constant}$$

so that

$$\begin{aligned}
\dot{\ell}(\theta) &= \sum_{i=1}^{n} \left\{ \frac{y_i}{\mu_i(\theta)} \dot{\mu}_i(\theta) - \dot{\mu}_i(\theta) \right\} \\
\ddot{\ell}(\theta) &= \sum_{i=1}^{n} \left\{ -\frac{y_i}{\mu_i(\theta)^2} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal + \frac{y_i}{\mu_i(\theta)} \ddot{\mu}_i(\theta) - \ddot{\mu}_i(\theta) \right\}
\end{aligned}$$

Poisson GLM

Example (Poisson GLM)

As $\mathbb{E}[Y_i] = \mu_i(\theta)$, approximating $y_i/\mu_i(\theta) \approx 1$, we have

$$\ddot{\ell}(\theta) \doteq \sum_{i=1}^{n} \left\{ -\frac{y_i}{\mu_i(\theta)^2} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal \right\} \doteq \sum_{i=1}^{n} \left\{ -\frac{1}{\mu_i(\theta)} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal \right\}$$

which yields the required positive definite approximation to $\ddot{\ell}(\theta)$.

Expected Information

The matrix $\ddot{\ell}(\theta)$ yields $-n\widehat{I}(\theta)$, where $\widehat{I}$ is the observed (unit) information. If available, the Fisher or expected information

$$\mathcal{I}(\theta) = \mathbb{E}[S_\theta(\theta)S_\theta(\theta)^\intercal] = \mathbb{V}ar[S_\theta(\theta)]$$

or

$$\mathcal{I}(\theta) = -\mathbb{E}[S_{\theta\theta}(\theta)]$$

can be utilized in the algorithm instead; this matrix is automatically positive (semi) definite.

Expected Information

In the Exponential Family

$$\ell(\theta) = g(\boldsymbol{y}) + \beta(\theta) + \boldsymbol{\gamma}(\theta)^{\mathsf{T}}\boldsymbol{h}(\boldsymbol{y})$$

so therefore

$$\dot{\ell}(\theta) = \dot{\beta}(\theta) + \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}\boldsymbol{h}(\boldsymbol{y})$$

and as $\mathbb{E}[\dot{\ell}(\theta)] = \mathbb{E}[S_\theta(\theta)] = \mathbf{0}$, if $\mathbb{E}[\boldsymbol{h}(\boldsymbol{y})] = \boldsymbol{\mu}(\theta)$, so therefore

$$\dot{\beta}(\theta) + \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}\boldsymbol{\mu}(\theta) = \mathbf{0}$$

Expected Information

Hence

$$\dot{\ell}(\theta) = \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}(\theta))$$

so therefore, by the matrix variance result

$$\mathbb{V}ar[\dot{\ell}(\theta)] = \mathbb{V}ar[\dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}(\boldsymbol{h}(\boldsymbol{Y}) - \boldsymbol{\mu}(\theta))] = \dot{\boldsymbol{\gamma}}(\theta)\Sigma(\theta)\dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}$$

where

$$\Sigma(\theta) = \mathbb{V}ar[\boldsymbol{h}(\boldsymbol{Y})].$$

Here the $p \times p$ matrix $\dot{\boldsymbol{\gamma}}(\theta)$ has (j,k)th element

$$\frac{\partial \gamma_j(\theta)}{\partial \theta_k}$$

Expected Information

Now, by further differentiation

$$\dot{\boldsymbol{\mu}}(\theta) = \Sigma(\theta)\dot{\boldsymbol{\gamma}}(\theta)$$

so that, provided $\Sigma(\theta)$ is non-singular,

$$\dot{\boldsymbol{\gamma}}(\theta) = \Sigma(\theta)^{-1}\dot{\boldsymbol{\mu}}(\theta)$$

and hence

$$\dot{\ell}(\theta) = \dot{\boldsymbol{\mu}}(\theta)^{\intercal}\Sigma(\theta)^{-1}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}(\theta))$$

so that, finally

$$\mathcal{J}(\theta) = \mathbb{V}ar[\dot{\ell}(\theta)] = \dot{\boldsymbol{\mu}}(\theta)\Sigma(\theta)\dot{\boldsymbol{\mu}}(\theta)^{\intercal}$$

Expected Information and Scoring

The matrix $\mathcal{J}(\theta)$ can replace the $\ddot{\ell}(\theta)$ term in the Newton's method formula, for example,

$$\theta_{n+1} = \theta_n + \{\mathcal{J}(\theta_n)\}^{-1}\dot{\ell}(\theta_n)$$

etc. This is termed **scoring** or **Fisher scoring**.

Quasi-Newton methods

The approximating matrix H_n can also be updated during the recursion. Using a second order Taylor approximation

$$\dot{\ell}(\theta_{n+1}) - \dot{\ell}(\theta_n) \doteq \ddot{\ell}(\theta_{n+1})(\theta_n - \theta_{n+1})$$

or

$$g_n \doteq \ddot{\ell}(\theta_{n+1}) s_n$$

say. Thus the positive definite approximating matrix H should satisfy

$$g_n = -H s_n$$

that is, for each step n,

$$g_n = -H_{n+1} s_n$$

Quasi-Newton methods

To update H_n to H_{n+1}, use the recursion

$$H_{n+1} = H_n - c_n v_n v_n^\intercal$$

where

$$c_n = [(g_n + H_n s_n)^\intercal s_n]^{-1}$$

and

$$v_n = g_n + H_n s_n.$$

This is termed a **rank-one update** because of the $v_n v_n^\intercal$ term; higher-rank updates exist (for example, the **BFGS** algorithm).

Note: can consider H_n^{-1} instead of H_n.

Iteratively Reweighted Least Squares

For a model using the one parameter exponential-dispersion family, the log-likelihood for data $y_1, \ldots, y_n$ takes the form

$$\ell(\theta, \phi) = \sum_{i=1}^{n} \frac{w_i (y_i \theta + c(\theta))}{\phi} + \sum_{i=1}^{n} d\left(y_i, \frac{\phi}{w_i}\right) + const. \qquad (1)$$

and the expectation and variance of observable Y_i are

$$\begin{aligned} \mathbb{E}[Y_i] &= -\frac{\partial c(\theta)}{\partial \theta} = -\dot{c}(\theta) = \mu \\ \mathbb{V}ar[Y_i] &= -\frac{\phi}{w_i}\frac{\partial^2 c(\theta)}{\partial \theta^2} = -\frac{\phi}{w_i}\ddot{c}(\theta) = \frac{\phi}{w_i} V(\mu) \qquad (2) \end{aligned}$$

say.

Iteratively Reweighted Least Squares

The GLM with link function l has $\mu = \mu_i$, and sets

$$l(\mu_i) = \boldsymbol{x}_i^{\mathsf{T}} \beta = \eta_i \tag{3}$$

say, for a model with parameters $\beta = \left(\beta_1, ..., \beta_p\right)$. With the canonical link, we have merely that

$$l(\mu_i) = \theta_i$$

so that $\theta_i = \boldsymbol{x}_i^{\mathsf{T}} \beta$.

η_i is the *linear predictor*.

Iteratively Reweighted Least Squares

A set of p **score equations** is obtained by differentiation partially with respect to the elements of β and equating to zero. For example, for β_j, by the chain rule

$$\begin{aligned}\frac{\partial}{\partial \beta_j}\left\{\ell\left(\beta,\phi\right)\right\} &= \sum_{i=1}^{n}\frac{w_i}{\phi}\left(y_i\frac{\partial\theta_i}{\partial\beta_j}+\frac{\partial c(\theta_i)}{\partial\beta_j}\right)\\ &= \sum_{i=1}^{n}\frac{w_i}{\phi}\left(y_i+\frac{\partial c(\theta_i)}{\partial\theta_i}\right)\frac{\partial\theta_i}{\partial\beta_j}=0 \qquad (4)\end{aligned}$$

Iteratively Reweighted Least Squares

However, differentiating the LHS of (3), denoting by $\dot{c}$ and $\ddot{c}$ the first and second derivatives of c wrt θ, we have

$$\frac{\partial}{\partial \beta_j}\left\{l\left(-\dot{c}\left(\theta_i\right)\right)\right\} = \dot{l}\left(-\dot{c}\left(\theta_i\right)\right)\left(-\ddot{c}\left(\theta_i\right)\right)\frac{\partial \theta_i}{\partial \beta_j} = \dot{l}\left(\mu_i\right)V\left(\mu_i\right)\frac{\partial \theta_i}{\partial \beta_j}$$

by (2). Hence, differentiating the RHS of (3) and substituting back

$$\frac{\partial \theta_i}{\partial \beta_j} = \frac{1}{\dot{l}\left(\mu_i\right)V\left(\mu_i\right)}\frac{\partial \eta_i}{\partial \beta_j}.$$

Iteratively Reweighted Least Squares

In addition,

$$\frac{\partial c(\theta_i)}{\partial \theta_i} = \dot{c}(\theta_i) = -\mu_i \qquad\qquad \frac{\partial \eta_i}{\partial \beta_j} = x_{ij}$$

so that, finally, (4) becomes

$$\sum_{i=1}^{n} \frac{w_i}{\phi} \frac{(y_i - \mu_i)\, x_{ij}}{\dot{l}(\mu_i)\, V(\mu_i)} = 0. \tag{5}$$

Removing the dispersion constant ϕ and re-arranging gives the set of score equations

$$\sum_{i=1}^{n} \frac{w_i (y_i - \mu_i)}{V(\mu_i)} \frac{x_{ij}}{\dot{l}(\mu_i)} = 0 \qquad\qquad j = 1, ..., p \tag{6}$$

Iteratively Reweighted Least Squares

If the canonical link is used

$$\theta_i = x_i^\intercal \beta \Longrightarrow \frac{\partial \theta_i}{\partial \beta_k} = x_{ij}$$

and the equations simplify to

$$\sum_{i=1}^{n} w_i \left(y_i - \mu_i\right) x_{ij} = 0 \qquad \therefore \qquad \sum_{i=1}^{n} w_i y_i = \sum_{i=1}^{n} w_i \mu_i x_{ij} \tag{7}$$

for $j = 1, ..., p$, to be solved simultaneously for $\beta_1, ..., \beta_p$ which appear in (7) through $\mu_i = l^{-1}\left(x_i^\intercal \beta\right)$.

Iteratively Reweighted Least Squares

The system (6) cannot be solved analytically, but can be solved recursively. Let

$$\begin{aligned} Z_i &= \eta_i + (Y_i - \mu_i)\,\dot{l}\,(\mu_i) \qquad (8) \\ &= x_i^{\mathsf{T}}\beta + (Y_i - \mu_i)\,\dot{l}\,(\mu_i) \end{aligned}$$

then as $\mathbb{E}[Y_i] = \mu_i$ the expected value of Z_i is

$$\mathbb{E}[Z_i] = \eta_i + (\mathbb{E}[Y_i] - \mu_i)\,\dot{l}\,(\mu_i) = \eta_i + 0 \times \dot{l}\,(\mu_i) = \eta_i.$$

and the variance of Z_i

$$\mathbb{V}ar[Z_i] = \mathbb{V}ar[Y_i]\left\{\dot{l}\,(\mu_i)\right\}^2 = \frac{\left\{\dot{l}\,(\mu_i)\right\}^2 \phi V(\mu_i)}{w_i}.$$

Iteratively Reweighted Least Squares

Hence, from (8) the $\{Z_i\}$ form a **linear model** in β conditional on $\{\eta_i\}$ and $\{\mu_i\}$, and if observed values $\{z_i\}$ were available, the parameters in $\mu_i = l^{-1}\left(x_i^{\intercal}\beta\right)$ could be estimated using a weighted least squares procedure, that is, by $\widehat{\beta}$ where,

$$\widehat{\beta} = \left(\mathbf{X}^{\intercal}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\intercal}\mathbf{W}z \tag{9}$$

and where, if $x_i = \left(x_{i1}, \ldots x_{ip}\right)^{\intercal}$ is the row vector of predictors, $\mathbf{X}$ is the $n \times n$ matrix whose n rows are $x_1, \ldots, x_n$, and

$$\mathbf{W} = \operatorname{diag}\left(\frac{w_1}{\left\{\dot{l}\left(\mu_1\right)\right\}^2 V\left(\mu_1\right)}, \ldots, \frac{w_n}{\left\{\dot{l}\left(\mu_n\right)\right\}^2 V\left(\mu_n\right)}\right)$$

Iteratively Reweighted Least Squares

The recursive procedure for finding the estimates of β in the GLM then proceeds as follows:

1. Choose any values for $\widehat{\mu}_i^{(0)}$ and $\widehat{\eta}_i^{(0)} = l\left(\widehat{\mu}_i^{(0)}\right)$ for $i = 1, ..., n$
2. For $i = 1, ..., n$ set

$$\widehat{z}_i^{(1)} = \widehat{\eta}_i^{(0)} + \left(y_i - \widehat{\mu}_i^{(0)}\right) \dot{l}\left(\widehat{\mu}_i^{(0)}\right)$$

and

$$\widehat{w}_i^{(1)} = \frac{w_i}{\left\{\dot{l}\left(\widehat{\mu}_i^{(0)}\right)\right\}^2 V\left(\widehat{\mu}_i^{(0)}\right)} \quad \Longrightarrow \quad \mathbf{W}^{(1)} = \text{diag}\left(\widehat{w}_1^{(1)}, ..., \widehat{w}_n^{(1)}\right)$$

3. Compute estimates

$$\hat{\beta}^{(1)} = \left(\mathbf{X}^\intercal \mathbf{W}^{(1)} \mathbf{X}\right)^{-1} \mathbf{X}^\intercal \mathbf{W}^{(1)} \widehat{z}^{(1)}$$

Iteratively Reweighted Least Squares

4. Define vectors

$$\widehat{\eta}^{(1)} = \mathbf{X}\hat{\beta}^{(1)} \qquad\qquad \widehat{\mu}^{(1)} = l^{-1}\left(\widehat{\eta}^{(1)}\right)$$

5. Iterate through 1. to 4. using $\widehat{\mu}^{(1)}$ and $\widehat{\eta}^{(1)}$ as the updated starting values, eventually to obtain $\widehat{\mu}^{(2)}$ and $\widehat{\eta}^{(2)}$
6. Repeat 1. to 5. until $\widehat{\mu}^{(t)}$ and $\widehat{\eta}^{(t)}$ satisfy

$$\left|\widehat{\mu}^{(t)} - \widehat{\mu}^{(t-1)}\right| < \epsilon_\mu \qquad\qquad \left|\widehat{\eta}^{(t)} - \widehat{\eta}^{(t-1)}\right| < \epsilon_\eta$$

   for tolerances ϵ_μ and ϵ_η.

Iteratively Reweighted Least Squares

This procedure is known as **Iteratively Reweighted Least Squares**. It can be shown that the resulting estimator, $\widetilde{\beta}$, has the correct asymptotic behaviour, that is, as $n \to \infty$,

$$\sqrt{n}(\widetilde{\beta} - \beta) \xrightarrow{d} \mathcal{N}\left(0, \mathcal{I}\left(\beta\right)^{-1}\right)$$

where the asymptotic variance may be estimated by the inverse of $\widehat{I}_n$

$$\widehat{I}_n = \frac{1}{\widehat{\phi}}\mathbf{X}^\intercal\mathbf{W}\mathbf{X} \qquad \text{where} \qquad [\mathbf{W}]_{ii} = \frac{w_i}{V\left(\widetilde{\mu}_i\right)\left\{\dot{l}(\widetilde{\mu}_i)\right\}^2}$$

and $\widetilde{\mu}_i$ is the estimated μ_i formed by plugging in $\widetilde{\beta}$ to the linear predictor.

Iteratively Reweighted Least Squares

Equation (8) is justified by noting that, by a first order Taylor approximation, expanding around $\mu_i = \mathbb{E}[Y_i]$,

$$\begin{aligned} l(Y_i) &\doteq l(\mu_i) + (Y_i - \mu_i)\dot{l}(\mu_i) \\ &= \eta_i + (Y_i - \mu_i)\dot{l}(\mu_i) \end{aligned}$$

This indicates that the IRLS procedure may work poorly in finite samples when this first order approximation is not adequate.

Conjugate Gradient Methods

A typical minimization problem relates to the quadratic form

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\intercal A\boldsymbol{x} - \boldsymbol{b}^\intercal \boldsymbol{x} + c$$

If A is positive definite and symmetric, minimizing this function is equivalent to solving the system

$$A\boldsymbol{x} = \boldsymbol{b}$$

as

$$\dot{g}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$$

This shows the equivalence of certain **optimization** and **root-finding** tasks.

Conjugate Gradient Methods

Steepest descent methods are recursive procedures that utilize the first derivative (**gradient**) to indicate how the next point in the recursion should be generated.

In the quadratic form problem, the direction of steepest descent is

$$-\dot{g}(\boldsymbol{x}) = \boldsymbol{b} - A\boldsymbol{x}$$

At step n, define the *residual*, $\boldsymbol{r}_n$, by

$$\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n.$$

Denote the true minimizing solution by $\boldsymbol{x}_*$, and the error at step n, $\boldsymbol{e}_n$, by

$$\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}_*.$$

Then $\boldsymbol{r}_n = -A\boldsymbol{e}_n$.

Conjugate Gradient Methods

We need to move in the direction $-\dot{g}(\boldsymbol{x})$ - but how far ? Starting at $\boldsymbol{x}_0$, consider the line

$$\boldsymbol{x} = \boldsymbol{x}_0 + \alpha \boldsymbol{r}_0 = \boldsymbol{x}_0 + \alpha(\boldsymbol{b} - A\boldsymbol{x}_0)$$

Pick α to minimize the function along this line:

$$\alpha_0 = \arg\min_{\alpha} g(\boldsymbol{x}_0 + \alpha \boldsymbol{r}_0)$$

to generate the next point

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha_0 \boldsymbol{r}_0 = \boldsymbol{x}_0 + \alpha_0(\boldsymbol{b} - A\boldsymbol{x}_0)$$

Minimizing wrt α: Note that

$$\frac{dg(\boldsymbol{x}_1)}{d\alpha} = \dot{g}(\boldsymbol{x}_1)^\mathsf{T} \frac{d\boldsymbol{x}_1}{d\alpha} = \dot{g}(\boldsymbol{x}_1)^\mathsf{T} \boldsymbol{r}_0$$

and so the derivative is zero if $\dot{g}(\boldsymbol{x}_1)$ and $\boldsymbol{r}_0$ are orthogonal.

Now, from above,

$$\dot{g}(\boldsymbol{x}_1) = -\boldsymbol{r}_1$$

so therefore at the minimizing α

$$\boldsymbol{r}_1^\mathsf{T} \boldsymbol{r}_0 = \boldsymbol{0}.$$

Conjugate Gradient Methods

Therefore

$$(\boldsymbol{b} - A\boldsymbol{x}_1)^\intercal \boldsymbol{r}_0 = (\boldsymbol{b} - A(\boldsymbol{x}_0 + \alpha \boldsymbol{r}_0))^\intercal \boldsymbol{r}_0 = \boldsymbol{0}$$

Rearranging, this means choosing

$$\alpha_0 = \frac{\boldsymbol{r}_0^\intercal \boldsymbol{r}_0}{\boldsymbol{r}_0^\intercal A \boldsymbol{r}_0}$$

Thus the recursion involves repeatedly computing

- $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$,
- $\alpha_n = \dfrac{\boldsymbol{r}_n^\intercal \boldsymbol{r}_n}{\boldsymbol{r}_n^\intercal A \boldsymbol{r}_n}$,
- $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{r}_n$.

Note that

$$\boldsymbol{r}_{n+1} = (\boldsymbol{I} - \alpha_n A)\boldsymbol{r}_n$$

Conjugate Gradient Methods

A modification to the steepest descent method is as follows: consider a set of **orthogonal** search directions

$$\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{(p-1)}$$

with step sizes $\alpha_0, \ldots, \alpha_{(p-1)}$ chosen to end up precisely at $\boldsymbol{x}_*$ from a given starting point.

In this strategy, we have

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{d}_n$$

Following through in the style of the previous calculations, this implies we need $\boldsymbol{d}_n^\mathsf{T} \boldsymbol{e}_{n+1} = 0$, or

$$\alpha_n = -\frac{\boldsymbol{d}_n^\mathsf{T} \boldsymbol{e}_n}{\boldsymbol{d}_n^\mathsf{T} \boldsymbol{d}_n}$$

Conjugate Gradient Methods

Of course, we cannot compute $\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}_*$. Instead, use the requirement that

$$\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_m = \boldsymbol{0} \qquad \forall m, n$$

that is, the directions are *conjugate* (or *A-orthogonal*). With the new (minimization) requirement that

$$\frac{dg(\boldsymbol{x}_{(n+1)})}{d\alpha} = 0 \qquad \Longrightarrow \qquad \boldsymbol{d}_n^\mathsf{T} A \boldsymbol{e}_{n+1} = 0$$

this means we should set

$$\alpha_n = -\frac{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{e}_n}{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_n}$$

which may be calculated, and the algorithm can proceed.

Conjugate Gradient Methods

A Gram-Schmidt type procedure can be used to construct the search direction vectors.

1. Take p linearly independent vectors $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$.
2. Set $\boldsymbol{d}_0 = \boldsymbol{u}_0$.
3. For $n = 1, 2, \ldots,$ set

$$\boldsymbol{d}_n = \boldsymbol{u}_n - \sum_{l=1}^{n-1} \frac{\boldsymbol{u}_n^\mathsf{T} A \boldsymbol{d}_l}{\boldsymbol{u}_l^\mathsf{T} A \boldsymbol{d}_l} \boldsymbol{d}_l$$

A common choice is to set

$$\boldsymbol{u}_n = \boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$$

that is, based on the successive residuals. This renders the computation more straightforward as Step 3. simplifies to

$$\boldsymbol{d}_n = \boldsymbol{r}_n + \frac{\boldsymbol{r}_n^\mathsf{T} \boldsymbol{r}_n}{\boldsymbol{r}_{(n-1)}^\mathsf{T} \boldsymbol{r}_{(n-1)}} \boldsymbol{d}_{(n-1)}$$

Conjugate Gradient Methods

Thus the recursion becomes

1. Set $\boldsymbol{d}_0 = \boldsymbol{r}_0 = b - A\boldsymbol{x}_0$.
2. For $n = 1, 2, \ldots,$ set

$$\begin{aligned}
\alpha_n &= \frac{\boldsymbol{r}_n^\mathsf{T}\boldsymbol{r}_n}{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_n} \\
\boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \alpha_n \boldsymbol{d}_n \\
\boldsymbol{r}_{n+1} &= \boldsymbol{r}_n - \alpha_n A \boldsymbol{d}_n \\
\boldsymbol{d}_{n+1} &= \boldsymbol{r}_{n+1} + \frac{\boldsymbol{r}_{n+1}^\mathsf{T}\boldsymbol{r}_{n+1}}{\boldsymbol{r}_n^\mathsf{T}\boldsymbol{r}_n}\boldsymbol{d}_n
\end{aligned}$$

Non-linear Conjugate Gradient

To minimize a general function g of θ, we proceed in a similar fashion:

1. Pick a starting value θ_0, and choose α_0 to minimize the function
$$g(\theta_0 - \alpha \dot{g}(\theta_0))$$
over α using a grid-based search. Define
$$\theta_1 = \theta_0 - \alpha_0 \dot{g}(\theta_0).$$
and $d(x_0) = -\dot{g}(\theta_0)$.
2. For the $n+1$st step, starting at θ_n: compute β_n using a standard formula dependent on $\dot{g}(\theta_n)$ and $\dot{g}(\theta_{n-1})$.
3. Set
$$d(\theta_n) = -\dot{g}(\theta_n) + \beta_n d(\theta_{n-1})$$

Non-linear Conjugate Gradient

4. Choose α_n to minimize

$$g(\theta_n + \alpha d(\theta_n))$$

over α.

5. Set

$$\theta_{n+1} = \theta_n + \alpha_n d(\theta_n).$$

and return to 2. Repeat until convergence.

Non-linear Conjugate Gradient

The logic behind this algorithm is that $-\dot{g}(\theta_n)$ gives a good indication of which direction to move in from θ_n in order to achieve minimization.

A typical choice for β_n is

$$\beta_n = \frac{\dot{g}(\theta_n)^\intercal \dot{g}(\theta_n)}{\dot{g}(\theta_{n-1})^\intercal \dot{g}(\theta_{n-1})}$$

Optimization with constraints

Optimization problems of the form

$$\text{minimize } g(\theta) \quad \text{subject to} \qquad \varphi_k(\theta) \leq c_k, k = 1, \ldots, K$$

are not straightforward to solve if the constraints, specified by $(\varphi_k, c_k), k = 1, \ldots, K$ are not simple interval constraints.

Lagrange multipliers can be used to reformulate the problem. Define the function $g^*(\theta, \lambda)$

$$g^*(\theta, \lambda) = g(\theta) + \sum_{k=1}^{K} \lambda_k(\varphi_k(\theta) - c_k),$$

differentiate with respect to (θ, λ), and find the zeros of $\dot{g}^*(\theta, \lambda) = 0$. The modification to the usual methods is that the solutions for λ should occur at saddlepoints, not minima.

Nelder-Mead Simplex algorithm

The Nelder-Mead algorithm is another form of recursive optimization procedure. It is **derivative-free** so is simpler to implement, although can be slower to converge.

The basic algorithm proceeds as follows: to minimize a function g in p dimensions

1. Pick $p+1$ points $x_1, \ldots, x_{p+1}$ indexed such that

$$g(x_1) \leq g(x_2) \leq \cdots \leq g(x_p) \leq g(x_{p+1})$$

   and compute the centre of mass $\overline{x}_p$ of $x_1, \ldots, x_p$
2. For a suitably chosen $\alpha \geq 1$, compute new point, x^*,

$$x^* = \overline{x}_p + \alpha(\overline{x}_p - x_{p+1})$$

Nelder-Mead Simplex algorithm

3. (i) If $g(x_1) \leq g(x^*) \leq g(x_p)$, replace x_{p+1} by x^*.
   (ii) If $g(x^*) < g(x_1)$, for suitably chosen $\beta \geq 1$, compute new point, $x^\dagger$,
   $$x^\dagger = \overline{x}_p + \beta(\overline{x}_p - x_{p+1})$$
   If $g(x^\dagger) < g(x^*)$, replace x_{p+1} by $x^\dagger$ and go to 1., else replace x_{p+1} by x^* and go to 1.
4. For a suitably chosen $\gamma < 1$, compute new point, $x^\ddagger$,
   $$x^\ddagger = x_{p+1} + \gamma(\overline{x}_p - x_{p+1})$$
   If $g(x^\ddagger) < g(x_{p+1})$, replace x_{p+1} by $x^\ddagger$, and go to 1.
5. For $i = 2, \ldots, n+1$, for a suitably chosen $\delta < 1$, replace x_i by
   $$x_1 + \delta(x_i - x_1)$$
   and go to 1.

Nelder-Mead Simplex algorithm

Steps 1-5 are repeated until the function value $f(x_1)$ ceases to change over time. The algorithm is somewhat *ad hoc*

- Need to specify $\alpha, \beta, \gamma, \delta$
- No proof of convergence to the global minimum

and can be slow to reach a stationary value.

However, it is remarkable effective.

R Code for optimization

All of these methods are available via the R function `optim`:

```
optim(par, fn, gr = NULL,  method = c("Nelder-Mead",
    "BFGS", "CG", "L-BFGS-B", "SANN"),
    lower = -Inf, upper = Inf,
    control = list(), hessian = FALSE, ...)
```

`par` provides the parameters, `fn` is the objective function, defined as a function elsewhere, `gr` is the gradient (first derivative) function vector. Arguments to these two functions can be passed via the `...` arguments. `lower` and `upper` specify the parameter constraints.

`hessian` is a logical specifying whether the Hessian (second derivative) matrix evaluated at the minimum value should be returned.

R Code for optimization

`control` is a list that controls the parameters of the chosen algorithm. Key settings are

- `fnscale` sets the scale of the objective function, useful in diagnosing convergence; setting this negative turns the (default) minimization into a maximization
- `maxit` sets the maximum number of iterations
- tolerance parameters `abstol` and `reltol`

Example (Pharmacokinetics Model)

A model typically used in *pharmacokinetics* (the study of how drugs are processed through the body through time), where given a dose D at time $x = 0$, the amount of drug left in the body at time x is

$$\mu(x; \theta_1, \theta_2, \theta_3) = D\theta_1(\exp\{-\theta_2 x\} - \exp\{-(\theta_2 + \theta_3)x\})$$

where $\theta_1, \theta_2, \theta_3 > 0$.

The parameters can be estimated by measuring concentration through time.

Non-Linear Least Squares

Example (Pharmacokinetics Model)

Let $\phi_j = \log \theta_j$. Least squares estimation of finds the minimum of the function

$$g(\phi_1, \phi_2, \phi_3) = \sum_{i=1}^{n} (y_i - \mu(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3}))^2$$

with respect to ϕ_1, ϕ_2, ϕ_3.

We have

$$\dot{g}(\phi) = 2 \sum_{i=1}^{m} (y_i - \mu(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3}))\dot{\mu}(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3})$$

where the computation of $\dot{\mu}_i$ involves partial derivatives with respect to ϕ_1, ϕ_2, ϕ_3.

Example (Pharmacokinetics Model)

For example

$$\mu_{\phi_1}(x;\phi) = De^{\phi_1}(\exp\{-e^{\phi_2}x\} - \exp\{-(e^{\phi_2}+e^{\phi_3})\}) = \mu(x;\phi)$$

and

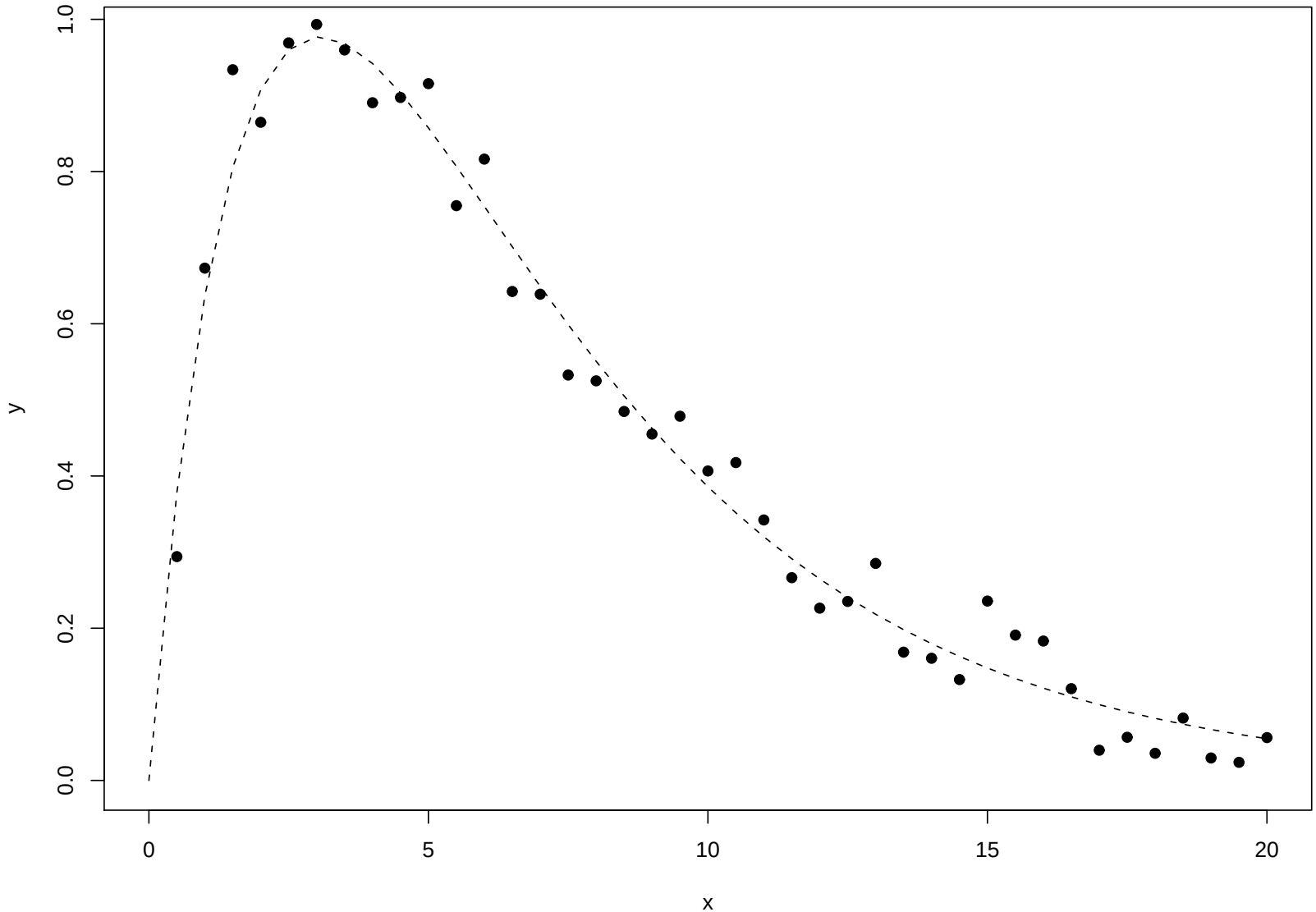
$$\begin{aligned}\mu_{\phi_2}(x;\phi) &= -De^{\phi_1+\phi_2}x(\exp\{-e^{\phi_2}x\} - \exp\{-(e^{\phi_2}+e^{\phi_3})x\}) \\ &= -xe^{\phi_2}\mu(x;\phi)\end{aligned}$$

and

$$\mu_{\phi_3}(x;\phi) = De^{\phi_1+\phi_3}\exp\{-(e^{\phi_2}+e^{\phi_3})x\}.$$

Second derivatives are also available in a straightforward fashion.

Pharmacokinetics data



R Code for Non-Linear LS

```
1  mu.funct<-function(thv,xv){
2
3          aval<-exp(thv[1])
4          lam1<-exp(thv[2])
5          lam2<-exp(thv[3])
6          fvec<-aval*(exp(-lam1*xv)-exp(-(lam1+lam2)*xv))
7          return(fvec)
8  }
9
10 g.funct<-function(thv,xv,yv){
11
12         rv<-yv-mu.funct(thv,xv)
13         return(sum(rv*rv))
14 }
15
16 fit.result<-optim(c(0,0,0),fn=g.funct,xv=X,yv=Y)
```

if the data are in vectors `X` and `Y`.

Using `control=list(trace=1)`

Example

```
  Nelder-Mead direct search function minimizer
function value for initial parameters = 10.485159
  Scaled convergence tolerance is 1.56241e-07
  Stepsize computed as 0.100000

BUILD              4 10.683248 10.379035
EXTENSION          6 10.4851599.809242
LO-REDUCTION       8 10.450614 9.809242
EXTENSION         10 10.379035 8.949165
EXTENSION         12 9.943463 7.170710
```

Using `control=list(trace=1)`

Example

```
LO-REDUCTION     184 0.088658 0.088656
HI-REDUCTION     186 0.088658 0.088656
HI-REDUCTION     188 0.088657 0.088656
HI-REDUCTION     190 0.088656 0.088656
HI-REDUCTION     192 0.088656 0.088656
HI-REDUCTION     194 0.088656 0.088656
Exiting from Nelder Mead minimizer
    196 function evaluations used
```

The EM Algorithm

The EM Algorithm is a method for producing the maximum likelihood estimates in **incomplete data** problems, that is, models formulated for data that are only partially observed.

Suppose that random variables to be modelled can be partitioned $(\boldsymbol{X}, \boldsymbol{Y})$ where

- $\boldsymbol{X} = (X_1, \ldots, X_m)^\intercal$ are **unobserved**, termed the **latent** or **missing** data.
- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\intercal$ are **observed**, termed the **incomplete data**
- $(\boldsymbol{X}, \boldsymbol{Y})$ are termed the **complete data**

where

$$f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta) = \int f_{\mathbf{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)\, d\boldsymbol{x}$$

The EM Algorithm: Terminology

$$L_Y(\theta; \boldsymbol{y}) = f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)$$

is termed the **incomplete data** likelihood, and

$$L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) = f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)$$

is termed the **complete data** likelihood.

The EM Algorithm facilitates maximization of the incomplete data likelihood $L_Y(\theta; \boldsymbol{y})$ by working with the complete data likelihood $L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y})$ and the conditional distribution

$$f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta) = \frac{f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)}{f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)} = \frac{L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y})}{L_Y(\theta; \boldsymbol{y})} = K(\theta; \boldsymbol{x}, \boldsymbol{y}) \quad (10)$$

$$\therefore \quad \ell_Y(\theta; \boldsymbol{y}) = \ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) - k(\theta; \boldsymbol{x}, \boldsymbol{y}) \quad (11)$$

say.

The EM Algorithm

Consider replacing the right-hand side of equation (11) by the expectations with respect to the conditional density $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y},\theta')$, for some $\theta' \in \Theta$. This yields

$$\begin{aligned}\ell_Y(\theta;\boldsymbol{y}) &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] - \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\log K(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] \\ &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] - \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[k(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] \quad (12)\end{aligned}$$

that is, we condition on a specific (but as yet unspecified) value of θ' when computing the expectations of $\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{y})$ and $k(\theta;\boldsymbol{X},\boldsymbol{Y})$ at the θ at which the likelihood on the left-hand side of equation (12) is being computed.

The EM Algorithm

The EM algorithm is an iterative algorithm that produces a sequence of estimates that converges to the (incomplete data) maximum likelihood estimate. Generically, starting from an initial value $\theta = \theta^{(0)}$, the $(r+1)$st value in the sequence, $\theta^{(r+1)}$, is constructed given the rth value , $\theta^{(r)}$,

$$\theta^{(r+1)} = \underset{\theta \in \Theta}{\text{argmax}}\, \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}] = \underset{\theta \in \Theta}{\text{argmax}}\, Q(\theta; \theta^{(r)})$$

say. The two components of this calculation are

- **E-step** : compute the expected conditional log-likelihood
- **M-step** : carry out the maximization of the expectation.

We wish to show that the sequence of estimates $\{\theta^{(r+1)}\}$ converges to the maximum likelihood estimate.

The EM Algorithm

For two pdfs f_1 and f_2 for random variable Z, we have by the properties of the *Kullback-Leibler divergence* that

$$\begin{aligned}
\mathbb{E}_{f_1}[\log f_1(Z)] - \mathbb{E}_{f_1}[\log f_2(Z)] &= -\mathbb{E}_{f_1}[\log\{f_2(Z)/f_1(Z)\}] \\
&\geq -\log \mathbb{E}_{f_1}[\{f_2(Z)/f_1(Z)\}] \\
&= -\log \int_{\mathcal{Z}} \{f_2(z)/f_1(z)\} f_1(z)\, dz \\
&= -\log \int_{\mathcal{Z}} f_2(z)\, dz = 0
\end{aligned}$$

so therefore $\mathbb{E}_{f_1}[\log f_1(Z)] \geq \mathbb{E}_{f_1}[\log f_2(Z)]$, with equality if and only if $f_1 \equiv f_2$.

The EM Algorithm

For $\theta \in \Theta$, as

$$K(\theta; \boldsymbol{X}, \boldsymbol{Y}) = \frac{L_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})}{L_Y(\theta; \boldsymbol{Y})} = f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta)$$

is itself a (conditional) pdf for all $\theta \in \Theta$, and hence we have, for any θ,

$$\begin{aligned} Q(\theta; \theta^{(r)}) - \ell_Y(\theta; \boldsymbol{y}) &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}] - \ell_{X,Y}(\theta; \boldsymbol{y}) \\ &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}\left[k(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}\right] \\ &\leq \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}\left[k(\theta^{(r)}; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}\right] \\ &= Q(\theta^{(r)}|\theta^{(r)}) - \ell_Y(\theta^{(r)}|\boldsymbol{y}). \end{aligned}$$

The EM Algorithm

Thus $\ell_Y(\theta; \boldsymbol{y}) - Q(\theta; \theta^{(r)})$ achieves its **minimum** value when $\theta = \theta^{(r)}$. Now suppose that $\theta^{(r+1)}$ is the value that maximizes $Q(\theta; \theta^{(r)})$ over Θ; we have that

$$\begin{aligned}
\ell_Y(\theta^{(r+1)}|\boldsymbol{y}) &\equiv Q(\theta^{(r+1)}|\theta^{(r)}) + \Big(\ell_Y(\theta^{(r+1)}|\boldsymbol{y}) - Q(\theta^{(r+1)}|\theta^{(r)})\Big) \\
&\geq Q(\theta^{(r)}|\theta^{(r)}) + \Big(\ell_Y(\theta^{(r)}|\boldsymbol{y}) - Q(\theta^{(r)}|\theta^{(r)})\Big) \\
&= \ell_Y(\theta^{(r)}|\boldsymbol{y})
\end{aligned}$$

and the likelihood attained is **increasing** with the sequence $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots$.

Thus, provided $L_Y(\theta; \boldsymbol{y})$ is bounded, we have that the sequence of $\ell_Y(\theta^{(r)}|\boldsymbol{y})$ converges to a (local) maximum of $L_Y(\theta; \boldsymbol{y})$.

Example: Finite Mixture Model

Suppose that $Y_1 \ldots, Y_n$ are a random sample from the K component finite mixture model for $y \in \mathbb{R}$

$$f_{Y|\theta}(y|\theta) = \sum_{k=1}^{K} \omega_k f_k(y|\theta_k) \qquad \sum_{k=1}^{K} \omega_k = 1$$

for $0 < \omega_k < 1$ where $f_1, \ldots, f_K$ are component densities.

Estimation of $\theta = (\theta_1, \ldots, \theta_K)^\intercal$ from the likelihood is in general difficult. However, consider the latent data $X_1, \ldots, X_n$, where $\mathbb{P}[X_i = k] = \omega_k$ are independent random variables. Then

$$\begin{aligned} L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \prod_{i=1}^{n} \prod_{k=1}^{K} \{\omega_k f_k(y_i|\theta_k)\}^{\mathbb{1}_{\{k\}}(X_i)} \\ \ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}_{\{k\}}(X_i) \left(\log \omega_k + \log f_k(y_i|\theta_k)\right). \end{aligned}$$

Example: Finite Mixture Model

The conditional distribution $f_{X|Y,\theta}(x|y,\theta)$ is a discrete distribution on the set $\{1, 2, \ldots, K\}$ where for each $i = 1, \ldots, n$

$$\mathbb{P}[X_i = k|\boldsymbol{y}, \omega, \theta] = \frac{\omega_k f_k(y_i|\theta_k)}{\sum_{j=1}^{K} \omega_j f_j(y_i|\theta_j)} = \varpi_k(y_i, \theta) \qquad k = 1, \ldots, K$$

where $X_1, \ldots, X_n$ are conditionally independent. Thus

$$\mathbb{E}_{f_{X_i|Y_i,\theta,\omega}}[\mathbb{1}_{\{k\}}(X_i)|y_i, \theta, \omega] = \varpi_k(y_i, \theta)$$

Example: Finite Mixture Model

Hence

$$
\begin{aligned}
Q(\theta, \omega | \theta^{(r)}, \omega^{(r)}) &= \mathbb{E}_{f_{\mathbf{X}|\mathbf{Y},\theta,\omega}}[\ell_{X,Y}(\theta; \mathbf{X}, \mathbf{Y}) | \boldsymbol{y}, \theta^{(r)}, \omega^{(r)}] \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} \varpi_k^{(r)}(y_i, \theta^{(r)}) \left(\log \omega_k + \log f_k(y_i|\theta_k)\right) \\
&= \sum_{k=1}^{K} \left\{ \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \right\} \log \omega_k \qquad (13) \\
&\quad + \sum_{k=1}^{K} \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \log f_k(y_i|\theta_k) \qquad (14)
\end{aligned}
$$

Example: Finite Mixture Model

We seek to maximize over (θ, ω) to obtain $(\theta^{(r+1)}, \omega^{(r+1)})$ presuming that the values $\varpi_k^{(r)}(y_i, \theta^{(r)})$ are fixed.

From the form of equations (13) and (14), it is evident that the function is sum of two parts, the first only depending on ω, the second only dependent on θ.

We can therefore maximize the two parts separately to obtain $(\theta^{(r+1)}, \omega^{(r+1)})$.

Example: Finite Mixture Model

Equation (13) is of the form of a multinomial likelihood in ω, therefore, by previous results, it follows that

$$\omega_k^{(r+1)} = \frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})}{\sum_{j=1}^{K} \sum_{i=1}^{n} \varpi_j^{(r)}(y_i, \theta^{(r)})} \qquad k = 1, \ldots, K$$

Equation (14) is the sum of K log-likelihoods for the K mixture components which can be maximized separately

$$\theta_k^{(r+1)} = \underset{\theta_k}{\operatorname{argmax}} \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \log f_k(y_i|\theta_k) \qquad (15)$$

Example: Finite Mixture Model

For certain choices of the component densities, this maximization can be carried out analytically. For example, if $f_k(y|\theta_k)$ is the normal density with expectation μ_k and variance σ_k^2, it follows that the new maximizing value equals $\theta_k^{(r+1)} = (\mu_k^{(r+1)}, \sigma_k^{(r+1)})$ where

$$\mu_k^{(r+1)} = \frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) y_i}{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})} \qquad \sigma_k^{(r+1)} = \sqrt{\frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})(y_i - \mu_k^{(r+1)})^2}{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})}}$$

In the normal model the terms in (15) correspond to likelihood components of the form

$$\{f_k(y_i|\theta_k)\}^{\varpi_k^{(r)}} = \left(\frac{1}{2\pi\sigma_k^2}\right)^{\varpi_k^{(r)}/2} \exp\left\{-\frac{\varpi_k^{(r)}}{2\sigma_k^2}(y_i - \mu_k^{(r)})^2\right\}$$

so the terms $\varpi_k^{(r)} \equiv \varpi_k^{(r)}(y_i, \theta^{(r)})$ are acting as weighting factors.

Example: Genetics of Human Blood Groups

In human genetics, the **genotype** at a genomic locus is a pair of **alleles** corresponding to small segments of DNA lying on the two chromosomal strands. The **phenotype** is the physical presentation or trait arising from the genotype. At a certain locus that determines the phenotype of blood group, the relationship between genotype and phenotype is somewhat complex; there are

- three alleles (A, B and O) yielding **six** possible genotypes (ordering is not important)
- only **four** phenotypes (A,B,AB and O).

Example: Genetics of Human Blood Groups

The relationship between phenotype and genotype in this case is determined by the following table. The third column, headed X, denotes a label for the genotype class. However, only the phenotype may be observed; let $Y_1, \ldots, Y_n$ denote the recorded phenotype for each of the n data.

Genotype	Phenotype	X	Y
AA	A	1	1
AB	AB	2	3
AO	A	3	1
BB	B	4	2
BO	B	5	2
OO	O	6	4

Example: Genetics of Human Blood Groups

Suppose that inference about the proportions of the three alleles A,B and O, denoted $\theta_A, \theta_B, \theta_O$ is required from a sample of size n of phenotype data.

One independence assumption (based on so-called *Hardy-Weinberg equilibrium*) is needed; we assume that the probability of observing a genotype is the product of the individual allele probabilities. For example

$$\mathrm{P(AA)} = \theta_A \times \theta_A \qquad\qquad \mathrm{P(AB)} = \theta_A \times \theta_B$$

and so on.

Example: Genetics of Human Blood Groups

Define the augmented data $X_1, \ldots, X_m$, where for $i = 1 \ldots, n$,

$$\mathbb{P}[X_i = j] = \mathbb{P}[i\text{th genotype is in class j}] \qquad j = 1, \ldots, 6$$

that is

$$\mathbb{P}[X_i = j] = \begin{cases} \theta_A^2 & j = 1 \\ \theta_A\theta_B & j = 2 \\ \theta_A\theta_O & j = 3 \\ \theta_B^2 & j = 4 \\ \theta_B\theta_O & j = 5 \\ \theta_O^2 & j = 6 \end{cases} \qquad \text{for } i = 1, \ldots, n$$

with $X_1, \ldots, X_m$ a random sample.

Example: Genetics of Human Blood Groups

This simplification yields a complete data likelihood

$$\begin{aligned} L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &\equiv L_X(\theta; \boldsymbol{x}) \\ &= \prod_{i=1}^{n} \left\{ \theta_A^{2\mathbb{1}_{\{1\}}(x_i)+\mathbb{1}_{\{2\}}(x_i)+\mathbb{1}_{\{3\}}(x_i)} \theta_B^{\mathbb{1}_{\{2\}}(x_i)+2\mathbb{1}_{\{4\}}(x_i)+\mathbb{1}_{\{5\}}(x_i)} \right. \\ &\qquad \left. \theta_O^{\mathbb{1}_{\{3\}}(x_i)+\mathbb{1}_{\{5\}}(x_i)+2\mathbb{1}_{\{6\}}(x_i)} \right\} \\ &= \theta_A^{2n_1+n_2+n_3} \theta_B^{n_2+2n_4+n_5} \theta_O^{n_3+n_5+2n_6} \end{aligned}$$

say, where

$$n_j = \sum_{i=1}^{n} \mathbb{1}_{\{j\}}(x_i) \qquad j = 1, \ldots, 6$$

that is, a multinomial-type likelihood in θ.

Example: Genetics of Human Blood Groups

In the standard notation, for the EM steps, we have to

- E-step: compute

$$Q(\theta; \theta^{(r)}) = \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}]$$

  taking the expectation over $X_1, \ldots, X_m$ etc.

- M-step: maximize $Q(\theta; \theta^{(r)})$ to get $\theta^{(r+1)}$.

Here the M-step is straightforward due to the multinomial likelihood. The E-step is also quite straightforward, but some steps need clarification.

Example: Genetics of Human Blood Groups

The log complete data likelihood takes the form

$$\begin{aligned}\ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \left(2\mathbb{1}_{\{1\}}(x_i) + \mathbb{1}_{\{2\}}(x_i) + \mathbb{1}_{\{3\}}(x_i)\right) \log \theta_A \\ &\quad + \sum_{i=1}^{n} \left(\mathbb{1}_{\{2\}}(x_i) + 2\mathbb{1}_{\{4\}}(x_i) + \mathbb{1}_{\{5\}}(x_i)\right) \log \theta_B \\ &\quad + \sum_{i=1}^{n} \left(\mathbb{1}_{\{3\}}(x_i) + \mathbb{1}_{\{5\}}(x_i) + 2\mathbb{1}_{\{6\}}(x_i)\right) \log \theta_O\end{aligned}$$

which is linear and additive in the indicator functions.

Example: Genetics of Human Blood Groups

Conditional on $\boldsymbol{Y}$ and θ, some expectations can be written down automatically. For example

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 3, \theta] = \begin{cases} 1 & j = 2 \\ 0 & j \neq 2 \end{cases}$$

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 4, \theta] = \begin{cases} 1 & j = 6 \\ 0 & j \neq 6 \end{cases}$$

as by definition $Y = 3 \Longrightarrow X = 2$ and $Y = 4 \Longrightarrow X = 6$.

Example: Genetics of Human Blood Groups

For the remaining conditional expectations, we have by Bayes theorem

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 1, \theta] = \begin{cases} \dfrac{\theta_A^2}{\theta_A^2 + 2\theta_A\theta_O} & j = 1 \\ \dfrac{2\theta_A\theta_O}{\theta_A^2 + 2\theta_A\theta_O} & j = 3 \\ 0 & \text{otherwise} \end{cases}$$

as if $Y = 1$, then either $X = 1$ or $X = 3$, with conditional probability for each determined by noting that

$$\mathbb{P}[X = 1|Y = 1] = \frac{\mathbb{P}[X = 1, Y = 1]}{\mathbb{P}[Y = 1]} = \frac{\mathbb{P}[X = 1, Y = 1]}{\mathbb{P}[X = 1, Y = 1] + \mathbb{P}[X = 3, Y = 1]}$$

Example: Genetics of Human Blood Groups

Similarly,

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 2, \theta] = \begin{cases} \dfrac{\theta_B^2}{\theta_B^2 + 2\theta_B\theta_O} & j = 4 \\ \dfrac{2\theta_B\theta_O}{\theta_B^2 + 2\theta_B\theta_O} & j = 5 \\ 0 & \text{otherwise} \end{cases}$$

Example: Genetics of Human Blood Groups

Thus $Q(\theta;\theta^{(r)})$ takes the form

$$Q(\theta;\theta^{(r)}) = \alpha_A^{(r)} \log \theta_A + \alpha_B^{(r)} \log \theta_B + \alpha_O^{(r)} \log \theta_O$$

where

$$\begin{aligned}
\alpha_A^{(r)} &= \frac{2n_1\theta_A^{(r)2}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} + n_3 + \frac{2n_1\theta_A^{(r)}\theta_O^{(r)}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} \\
\alpha_B^{(r)} &= n_3 + \frac{2n_2\theta_B^{(r)2}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} + \frac{2n_2\theta_B^{(r)}\theta_O^{(r)}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} \\
\alpha_O^{(r)} &= \frac{2n_1\theta_A^{(r)}\theta_O^{(r)}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} + \frac{2n_2\theta_B^{(r)}\theta_O^{(r)}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} + 2n_4.
\end{aligned}$$

and $n_1, \ldots, n_4$ are the observed counts for phenotypes A,B,AB and O.

Example: Genetics of Human Blood Groups

By the results for the multinomial likelihood, we can maximize $Q(\theta; \theta^{(r)})$ analytically to get

$$\begin{aligned}
\theta_A^{(r+1)} &= \frac{\alpha_A^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}} \\
\theta_B^{(r+1)} &= \frac{\alpha_B^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}} \\
\theta_O^{(r+1)} &= \frac{\alpha_O^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}}
\end{aligned}$$

Example: Genetics of Human Blood Groups

Real Data Example: Data from Clarke *et. al.* (1959)
We have $n_1 = 186$, $n_2 = 38$, $n_3 = 13$ and $n_4 = 284$ for the numbers of A,B,AB and O phenotypes in a sample of $n = 521$. Starting the iterative procedure at $\theta^{(0)} = (1/3, 1/3, 1/3)^\intercal$ yields the following first ten iterations:

r	$\theta_A^{(r)}$	$\theta_B^{(r)}$	$\theta_O^{(r)}$
1	0.25047985	0.06110045	0.68841971
2	0.21845436	0.05049394	0.73105170
3	0.21418233	0.05016173	0.73565593
4	0.21366195	0.05014667	0.73619139
5	0.21359944	0.05014547	0.73625508
6	0.21359196	0.05014535	0.73626270
7	0.21359106	0.05014533	0.73626361
8	0.21359095	0.05014533	0.73626372
9	0.21359094	0.05014533	0.73626373
10	0.21359094	0.05014533	0.73626373

indicating that convergence to the maximum value is fairly rapid.

The EM Algorithm

The EM algorithm

- can often be slow to converge, although speed-up methods exist.
- is only guaranteed to find a local maximum of the likelihood; typically, multiple starting values are used
- does not readily yield standard errors, as it does not utilize (or differentiate) the likelihood directly

The EM Algorithm: Variance Estimates

Let

$$\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) = -\mathbb{E}_{X|Y,\theta}[\ddot{k}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]$$

and

$$\mathcal{I}_{X,Y}(\theta) = -\mathbb{E}_{X,Y|\theta}[\ddot{\ell}(\theta; \boldsymbol{X}, \boldsymbol{Y})]$$

denote the **missing** and **complete** information matrices respectively, with corresponding score functions

$$\dot{k}(\theta; \boldsymbol{X}, \boldsymbol{y}) \qquad\qquad \dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})$$

respectively.

The EM Algorithm: Variance Estimates

By definition

$$\dot{\ell}(\theta; \boldsymbol{x}, \boldsymbol{y}) = \frac{\partial}{\partial \theta}\{\log f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)\} = \frac{\partial}{\partial \theta}\{\log f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta) + \log f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)\}$$

so therefore

$$\begin{aligned} \dot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \dot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) + \dot{\ell}_Y(\theta; \boldsymbol{y}) \\ \ddot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \ddot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) + \ddot{\ell}_Y(\theta; \boldsymbol{y}) \end{aligned}$$

Replacing $\boldsymbol{x}$ by $\boldsymbol{X}$, and taking expectation with respect to the distribution $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}$ at $\boldsymbol{Y} = \boldsymbol{y}$ yields

$$\mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) = \mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) - \ddot{\ell}_Y(\theta; \boldsymbol{y})$$

say, where $\mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) = -\mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}[\ddot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]$.

The EM Algorithm: Variance Estimates

Hence

$$-\ddot{\ell}_Y(\theta; \boldsymbol{y}) = \mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) - \mathcal{I}_{X|Y}(\theta; \boldsymbol{y})$$

This can be interpreted as

Observed data information at $\boldsymbol{y}$

=

Expected complete data information evaluated at $\boldsymbol{y}$

-

Expected latent data information evaluated at $\boldsymbol{y}$.

Replacing $\boldsymbol{y}$ by $\boldsymbol{Y}$, and taking a further expectation over $f_{\boldsymbol{Y}}$, we have

$$\mathcal{I}_Y(\theta) = \mathcal{I}_{X,Y}(\theta) - \mathbb{E}_{f_{\boldsymbol{Y}|\theta}}[\mathcal{I}_{X|Y}(\theta; \boldsymbol{Y})]$$

The EM Algorithm: Variance Estimates

Now

$$\dot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) = \dot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) - \dot{\ell}_Y(\theta; \boldsymbol{y})$$

so for any fixed $\boldsymbol{y}$ the final term is a constant, so replacing $\boldsymbol{x}$ by $\boldsymbol{X}$ and taking conditional expectations wrt $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}$, we have

$$\begin{aligned}
\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) &= \mathbb{V}ar_{f_{X|Y,\theta}}[\dot{k}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] \\
&= \mathbb{V}ar_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] \\
&= \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})^\intercal|\boldsymbol{y}] \\
&\qquad -\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]^\intercal
\end{aligned}$$

The EM Algorithm: Variance Estimates

By the usual expansion

$$\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] = \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X|Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] + \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{Y}(\theta; \boldsymbol{Y})|\boldsymbol{y}]$$

and in this expression, the first expectation on the RHS is zero by the usual properties of (conditional) score functions. Therefore

$$\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) = \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})^{\mathsf{T}}|\boldsymbol{y}] - \ell_{Y}(\theta; \boldsymbol{y})\ell_{Y}(\theta; \boldsymbol{y})^{\mathsf{T}}$$

Now from the previous expression

$$-\ddot{\ell}_{Y}(\theta; \boldsymbol{y}) = \mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) - \mathcal{I}_{X|Y}(\theta; \boldsymbol{y})$$

At the ML estimate, $\ell_{Y}(\widehat{\theta}; \boldsymbol{y}) = 0$.

The EM Algorithm: Variance Estimates

Therefore, at the ML estimate

$$-\ddot{\ell}_Y(\widehat{\theta};\boldsymbol{y}) \quad = \quad \mathcal{I}_{X,Y}(\widehat{\theta};\boldsymbol{y}) - \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\widehat{\theta};\boldsymbol{X},\boldsymbol{Y})\dot{\ell}_{X,Y}(\widehat{\theta};\boldsymbol{X},\boldsymbol{Y})^{\mathsf{T}}|\boldsymbol{y}]$$

Both quantities on the right hand side are computed during the EM algorithm steps, as functions of the complete data, so the left hand side can be estimated without recourse to other methods.

MM Algorithms

A critical step in the EM algorithm derivation is the realization that the iterations produce a monotone increasing sequence of likelihood values; this guarantees convergence if the likelihood is bounded above.

This motivates a general objective: to produce an iterative scheme to maximize or function $g(\theta)$ such that for the sequence of points generated, $\{\theta_n\}$, for all n,

$$g(\theta_{n+1}) \geq g(\theta_n)$$

for maximization, or

$$g(\theta_{n+1}) \leq g(\theta_n)$$

for minimization.

MM Algorithms

An MM algorithm

- for minimization: "majorize/minimize"
- for maximization: "minorize/maximize"

produces such a sequence. A function $h(x; x_n)$ *majorizes* g at x_n if $g(x_n) = h(x_n; x_n)$ and

$$g(x) \leq h(x; x_n) \qquad \forall x \neq x_n$$

Therefore if next value x_{n+1} is chosen such that

$$x_{n+1} = \operatorname*{argmin}_x h(x; x_n) \qquad \text{or} \qquad h(x_{n+1}; x_n) \leq h(x_n; x_n)$$

then $g(x_{n+1}) \leq g(x_n)$. The function h might be easier to work with than g.

At an interior point of the support of g, as $g(x_n) = h(x_n; x_n)$,

$$\dot{g}(x_n) = \dot{h}(x_n; x_n)$$

and

$$\dot{h}(x_n; x_n) - \dot{g}(x_n)$$

is positive definite.

MM Algorithms: Convexity methods

For **convex** functions, a method of majorization follows from

$$g(\alpha^\intercal t) \leq \sum_{j=1}^{p} \alpha_j g(t_j)$$

which is useful if we choose for any two positive points x and y, and vector of positive constants c,

$$\alpha_j = \frac{c_j y_j}{c^\intercal y} \qquad\qquad t_j = (c^\intercal y)\frac{x_j}{y_j}$$

then

$$g(c^\intercal x) \leq \sum_{j=1}^{p} \frac{c_j y_j}{c^\intercal y} g((c^\intercal y)x_j/y_j) = h(x;y)$$

say. This leads to the potential for a *componentwise* majorization.

MM Algorithms: Convexity methods

To relax positivity: choose α_j according to

$$0 \leq \alpha_j \leq 1 \qquad \sum_{j=1}^{p} \alpha_j = 1$$

such that

$$g(c^\intercal x) \leq \sum_{j=1}^{p} \alpha_j g\left(\frac{\alpha_j}{c_j}(x_j - y_j) + c^\intercal y\right) = h(x; y)$$

provided $c_j \neq 0$ for all j. Could choose

$$\alpha_j = \frac{|c_j|^\gamma}{\sum_{k=1}^{p} |c_k|^\gamma}$$

for $\gamma \geq 0$.

MM Algorithms: Derivative methods

First order method: for a **concave** function

$$g(x) \leq g(y) + \dot{g}(y)^\intercal (x - y) = h(x; y)$$

Second order method: by a Taylor expansion, for intermediate point z

$$\begin{aligned} g(x) &= g(y) + \dot{g}(y)^\intercal (x - y) + \frac{1}{2}(x - y)^\intercal \ddot{g}(z)(x - y) \\ &\leq g(y) + \dot{g}(y)^\intercal (x - y) + \frac{1}{2}(x - y)^\intercal A(x - y) \end{aligned}$$

for some matrix A, provided $A - \ddot{g}(z)$ is non-negative definite for all z. The right-hand side of this inequality defines $h(x; y)$.

MM Algorithms: Example

Example (Linear Least Squares)

In the linear least squares problem, we aim to minimize the function

$$g(\theta) = \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\intercal \theta)^2$$

for data $(y_i, \boldsymbol{x}_i), i = 1, \ldots, n$. Note that the terms in the summand are convex in (the elements of) θ, so consider the majorization at current point $\theta^{(r)}$

$$g(\theta) \leq \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij} \left(y_i - \frac{x_{ij}}{\alpha_{ij}} (\theta_j - \theta_j^{(r)}) - \boldsymbol{x}_i^\intercal \theta^{(r)} \right)^2$$

for which the RHS defines $h(\theta; \theta^{(r)})$.

MM Algorithms: Example

Example (Linear Least Squares)

The minimum of this $h(\theta; \theta^{(r)})$ is achieved at

$$\theta_j^{(r+1)} = \theta_j^{(r)} + \frac{\sum_{i=1}^{n} x_{ij}(y_i - \boldsymbol{x}_i^\mathsf{T}\theta^{(r)})}{\sum_{i=1}^{n} \frac{x_{ij}^2}{\alpha_{ij}}}$$

which defines the recursion. Again define

$$\alpha_{ij} = \frac{|x_{ij}|^\gamma}{\sum_{k=1}^{p} |x_{ik}|^\gamma}$$

for some appropriate γ. This procedure avoids matrix inversion.

Block Relaxation and Coordinate Methods

Block relaxation optimization methods update different (possibly disjoint) subsets of the components of the parameter vector at each step of the recursion, cycling through the subsets to complete an update. This strategy can improve convergence, or be easier to implement.

When each block just involves a single parameter, the approach is termed a **cyclic coordinate** method, that is

- *cyclic coordinate descent* for minimization
- *cyclic coordinate ascent* for maximization

Example (Poisson log-linear model)

A Poisson log-linear model is often used to model data in a multi-way contingency table. For a three-way table, defined by three factors A, B and C, it is assumed that $Y_{ijk} \sim Poisson(\mu_{ijk})$ where the saturated model can be written

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} + \lambda_{ij}^{(AB)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)} + \lambda_{ijk}^{(ABC)}$$

and all other log-linear models are nested inside the saturated model. If the factors take I, J and K levels, then there are a maximum of IJK parameters. The log likelihood is

$$\ell(\lambda) = \sum_i \sum_j \sum_k (y_{ijk} \log \mu_{ijk} - \mu_{ijk}) + \text{const.}$$

Example (Poisson log-linear model)

Identifiability: consider the model conventionally written $Y \sim A$, that is, the main effect A only model. In this model, the mean parameter is explained only as a function of the first factor index, that is, for all j, k,

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} \qquad i = 1, \ldots, I$$

The model in this form has $I + 1$ parameters, where only I are identifiable. Thus constraints for identifiability must be introduced; typically, these constraints take the form

$$\sum_{i=1}^{I} \lambda_i^{(A)} = 0.$$

Example (Poisson log-linear model)

For interaction terms, the constraints are of the form

$$\sum_{j=1}^{J} \lambda_{ij}^{(AB)} = 0$$

or

$$\sum_{j=1}^{J} \sum_{k=1}^{K} \lambda_{ijk}^{(ABC)} = 0$$

for each i.

Example (Poisson log-linear model)

To find the ML estimates, note that solving $\ell_{\lambda_0}(\lambda) = 0$ reduces to

$$\ell_{\lambda_0}(\lambda) = \sum_i \sum_j \sum_k (y_{ijk} - \mu_{ijk}) = 0$$

But note that μ_{ijk} can be written

$$\mu_{ijk} = \exp\{\lambda_0\}\omega_{ijk}$$

so therefore

$$\exp\{\lambda_0\} = \frac{\sum_i \sum_j \sum_k y_{ijk}}{\sum_i \sum_j \sum_k \omega_{ijk}} = \frac{N}{\omega_{...}}$$

say, where N is the total count, whatever the ω values are.

Example (Poisson log-linear model)

Therefore the log likelihood using this reparameterization is

$$\begin{aligned}\ell(\omega) &= \sum_i \sum_j \sum_k y_{ijk} \log \frac{N\omega_{ijk}}{\omega_{\cdots}} - N \\ &= \sum_i \sum_j \sum_k y_{ijk} \log \frac{\omega_{ijk}}{\omega_{\cdots}} + N \log N - N\end{aligned}$$

This is a complicated function of the ω parameters, so differentiating and solving to compute the ML estimates is not possible analytically (the exception to this is the case of the saturated model).

Example (Poisson log-linear model)

For a non-saturated model, the Poisson log likelihood is not amenable to direct maximization, but a block-based method can be used. Consider the model written in traditional notation as

$$A + B + C + A.B + A.C + B.C$$

where

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} + \lambda_{ij}^{(AB)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)}$$

and the identifiability constraints are

$$\sum_i \lambda_i^{(A)} = 0 \qquad \sum_j \lambda_{ij}^{(AB)} = 0 \qquad \sum_k \lambda_{ik}^{(AC)} = 0$$

and similarly for the other factors.

Example (Poisson log-linear model)

Write

$$\begin{aligned} \mu_{ij} &= \exp\{\lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_{ij}^{(AB)}\} \\ \alpha_{ijk} &= \exp\{\lambda_k^{(C)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)}\} \end{aligned}$$

and consider the score equation for a single parameter, $\lambda_{ij}^{(AB)}$ say. Note first that for **any** of the parameters ϕ that appear in the form of μ_{ijk},

$$\frac{\partial \mu_{ijk}}{\partial \phi} = \mu_{ijk}$$

so therefore, using the . symbol to indicate summation, we have

$$\frac{\partial \ell(\lambda)}{\partial \lambda_{ij}^{(AB)}} = \sum_k (y_{ijk} - \mu_{ijk}) = y_{ij.} - \mu_{ij.}$$

Example (Poisson log-linear model)

But

$$\mu_{ij.} = \mu_{ij}\alpha_{ij.}$$

so therefore setting the score equal to zero yields

$$\mu_{ij} = \frac{y_{ij.}}{\alpha_{ij.}}$$

Now, if

$$\sum_k (y_{ijk} - \mu_{ijk}) = 0$$

then

$$\frac{\partial \ell(\lambda)}{\partial \lambda_0} = \frac{\partial \ell(\lambda)}{\partial \lambda_i^{(A)}} = \frac{\partial \ell(\lambda)}{\partial \lambda_j^{(B)}} = 0.$$

Example (Poisson log-linear model)

Assuming the α_{ijk} (i.e. $\lambda_k^{(C)}$, $\lambda_{ik}^{(AC)}$, and $\lambda_{jk}^{(BC)}$) are held constant, the remaining score equations can be solved analytically

$$\begin{aligned}
\lambda_0 &= \frac{1}{IJ}\sum_i\sum_j \log\mu_{ij} = \frac{1}{IJ}\sum_i\sum_j \log\left\{\frac{y_{ij.}}{\alpha_{ij.}}\right\} \\
\lambda_i^{(A)} &= \frac{1}{J}\sum_j \log\mu_{ij} - \lambda_0 \\
\lambda_j^{(B)} &= \frac{1}{I}\sum_i \log\mu_{ij} - \lambda_0 \\
\lambda_{ij}^{(AB)} &= \log\mu_{ij} - \lambda_0 - \lambda_i^{(A)} - \lambda_j^{(B)}
\end{aligned}$$

These parameters are thus updated together according to these formulae, and identify the identifiability constraints.

Example (Poisson log-linear model)

The subsequent steps in the iteration update

- $\lambda_0, \lambda_i^{(A)}, \lambda_k^{(C)}, \lambda_{ik}^{(AC)}$
- $\lambda_0, \lambda_j^{(B)}, \lambda_k^{(C)}, \lambda_{ik}^{(BC)}$

and then all three updates are carried out repeatedly until convergence. Note that the parameter blocks here are not disjoint.

Note also that this algorithm leads to a non-decreasing sequence of log likelihood values. The algorithm is termed *Iterative Proportional Fitting*.

Example (The LASSO)

The LASSO (Least Absolute Shrinkage and Selection Operator) is a form of penalized loss estimation procedure, where the minimum of

$$g(\theta) = \mathcal{L}(\boldsymbol{x}; \theta) + \lambda \sum_{j=1}^{p} |\theta_j|$$

is sought, where $\mathcal{L}$ is a data-dependent loss function, and $\lambda > 0$ is a tuning parameter at the control of the user. Most often the LASSO in the context of (L_2) regression

$$g(\beta) = \|\boldsymbol{y} - \boldsymbol{x}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

but $\mathcal{L}$ could be any negative log likelihood.

Example (The LASSO)

The form of the penalty

$$\sum_{j=1}^{p} |\beta_j|$$

renders this a relatively challenging minimization problem, as the penalty is not differentiable at $\beta_j = 0$.

A common method of solution is cyclic coordinate descent, where the parameters are updated in turn to decrease the objective function.

Example (The LASSO)

Assuming $\mathcal{L}$ is differentiable, taking the derivative of g from the right yields

$$g^+_{\theta_j}(\theta) = \dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \begin{cases} \lambda & \theta_j \geq 0 \\ -\lambda & \theta_j < 0 \end{cases}$$

and from the left yields

$$g^-_{\theta_j}(\theta) = \dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \begin{cases} -\lambda & \theta_j > 0 \\ \lambda & \theta_j \leq 0 \end{cases}$$

where

$$g^\pm_\theta(\theta) = \lim_{t \downarrow 0} \frac{g(\theta \pm t\epsilon_\theta) - g(\theta)}{t}$$

Example (The LASSO)

Cyclic coordinate descent proceeds as follows:

1. Select starting values for $\theta_1^{(0)}, \ldots, \theta_p^{(0)}$.
2. For $j = 1, \ldots, p$, evaluate $g_{\theta_j}^+(\theta)$ and $g_{\theta_j}^-(\theta)$ at the current parameter value $\theta_j^{(0)}$:
   - If both derivatives are **non-negative**, do not update θ_j
   - If one derivative is **negative** (indicating the potential for reducing the value of the objective function), update θ_j by moving in the direction of greatest possible decrease.
3. Complete the cycle to generate $\theta_1^{(1)}, \ldots, \theta_p^{(1)}$.
4. Cycle through 1. to 3. until the updated vector $\theta_1^{(r)}, \ldots, \theta_p^{(r)}$ ceases to decrease the objective function.

Example (The LASSO)

Implementing the update: If $\theta_j^{(r)} = 0$

$$g_{\theta_j}^{+}(\theta) = \dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \lambda \qquad\qquad g_{\theta_j}^{-}(\theta) = -\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \lambda$$

so

- if $\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) < -\lambda$, **increase** θ_j,
- else if $\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) > \lambda$, **decrease** θ_j.
- else do not update θ_j, set $\theta_j^{(r+1)} = \theta_j^{(r)}$

Example (The LASSO)

In penalized linear (L_2) regression with an intercept β_0:

$$g(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \boldsymbol{x}_i^\intercal\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|.$$

The update for β_0 is straightforward

$$\beta_0^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\intercal\beta^{(r)}) = \beta_0^{(r)} - \frac{1}{n}\dot{\mathcal{L}}_{\beta_0}(\beta^{(r)})$$

say.

Example (The LASSO)

The update for the other β parameters can be computed using the ideas above: either

$$\beta_j^{(r+1)} = \min \left\{ 0, \beta_j^{(r)} - \frac{\dot{\mathcal{L}}_{\beta_j}(\beta^{(r)}) - \lambda}{\sum_{i=1}^{n} x_{ij}^2} \right\}$$

or

$$\beta_j^{(r+1)} = \max \left\{ 0, \beta_j^{(r)} - \frac{\dot{\mathcal{L}}_{\beta_j}(\beta^{(r)}) + \lambda}{\sum_{i=1}^{n} x_{ij}^2} \right\}$$

as only one of these is non-zero.