Summer Institute in Statistical Genetics

Module 17:

MCMC for Genetics

Presenters

Dr Eric C. Anderson
NOAA - Southwest Fisheries Science Center, Santa Cruz, CA
Research Associate, Department of Applied Math & Statistics, UCSC

Eric develops methods for the analysis of genetic data from populations of salmon and other organisms. He has developed approaches using MCMC and other Monte Carlo methods for estimating N_e, classifying species hybrids, the inference of pedigrees, and analysis of massively parallel sequencing data.

Dr. Matthew Robinson
Assistant Professor, Complex Trait Genetics Group
Department of Computational Biology
University of Lausanne, Switzerland

Matthew began working on evolutionary genetics in wild populations and then changed fields to human medical genetics. His research areas include prediction of disease risk in personalised medicine, the genetics of aging, genotype-environment interactions, the role of selection in shaping human population differentiation, and assortative mating of human populations.

Schedule of Sessions

Wednesday

12:00-1:30pm	LUNCH
1:30-3:00pm	Session 1: Intro, Basic Theory, and the Bayesian Perspective
3:00-3:30pm	BREAK
3:30-5:00pm	Session 2: Monte Carlo and Markov Chains

Thursday

8:30-10:00am	Session 3: Simple MCMC Examples Using R
10:00-10:30am	BREAK
10:30-12noon	Session 4: More Complex MCMC Examples
12:00-1:30pm	LUNCH
1:30-3:00pm	Session 5: Case Study I: Inference of Population Structure
3:00-3:30pm	BREAK
3:30-5:00pm	Session 6: Practical on *structure*

Friday

8:30-10:00am	Session 7: Case Study II: Haplotype Inference
10:00-10:30am	BREAK
10:30-12noon	Session 8: Practical on PHASE
12:00-1:30pm	LUNCH
1:30-3:00pm	Session 9: Bayes Factors and Model Selection
3:00-3:30pm	BREAK
3:30-5:00pm	Session 10: Importance Sampling, MCMC Likelihood, Pedigrees, Metropolis-coupled MCMC

Personal Probability

If I take a shuffled pack of cards, what is the probability that the top card is the Ace of Spades?

Are you sure?

What would change your mind?

Personal Probability

If I toss a coin 10 times, and see 0 heads, what is your estimate for the long-run probability of a head?

What if I toss a coin once, and see 0 heads?

Personal Probability

In Classical statistics, probability is treated as a way of stating limiting long-run frequencies of independent events occuring.

In Bayesian statistics, probability is treated as a way (*the* way) of expressing uncertainty about unknown quantities.

This view is more flexible, since it allows one to use probability in treating uncertain events of many different kinds. For example, what is the probability that it will rain tomorrow? That Donald Trump will face impeachment in the next year? That Monrovia is the capital of Liberia?

Uncertainty is, intrinsically, personal. Individuals may differ in their uncertainty, and therefore in their personal probabilities. For this reason personal probabilities are also referred to as "subjective probabilities."

Personal Probability: Underlying Theory

The theory underlying personal probability is too vast for us to do justice here.

Fortunately you don't need to know it to apply or understand much of Bayesian statistics (although it helps).

Here's a one-line summary: "In the face of uncertainty, a logical person should act as if they are assigning probabilities to uncertain events, losses to outcomes, and act so as to minimize their expected loss."

And if I'm allowed another line: "Probabilities of uncertain events should be updated in the light of new information, using Bayes Theorem".

For more info (non-math): Lindley, "Understanding Uncertainty".

Classical Inference

Probability model $p(y|\theta)$ for data $y = (y_1, \ldots, y_n)$.

θ a vector of parameters *assumed fixed but unknown*.

Statistical inference is concerned with estimating the value of θ from the data y.

Statistical procedure such as maximum likelihood estimation (MLE) provides estimate $\widehat{\theta}$ of θ as a function of y.

MLE is the value of θ which maximises $p(y|\theta)$, regarded as a function of θ (the likelihood function).

Example: Estimating population allele frequency

Suppose n diploid individuals sampled from a population, typed at a locus with two alleles (A,a). y = number of A alleles observed. Want to estimate θ, frequency of A allele.

Model the alleles as being independent draws from $\Pr(A) = \theta$ (Hardy Weinberg Equilibrium).

$$L(\theta) = p\left(y|\,\theta\right) = \left(\begin{array}{c} 2n \\ y \end{array} \right) \theta^{y} \left(1-\theta\right)^{2n-y}, \qquad y = 0, 1, \ldots, 2n$$

$$\widehat{\theta} = y/2n$$

Uncertainty (eg confidence intervals) estimated from sampling distribution of $\widehat{\theta}$.

Bayesian Inference

In Bayesian inference, unknown parameters θ are regarded as random variables rather than fixed but unknown quantities.

Prior knowledge about θ expressed by a "prior" probability distribution $p(\theta)$.

Information in observed data y expressed via the Likelihood function $L(\theta) = p(y|\theta)$.

Two sources of information are combined using Bayes Theorem (or "Bayes Rule"):

$$p(\theta|y) = \frac{p(y|\theta)\,p(\theta)}{p(y)}$$

$p(\theta|y)$ called the "posterior distribution" of θ.

Bayes Rule: Derivation

The joint probability of y and θ can be expressed as either

$$p(y, \theta) = p(y|\theta)\, p(\theta)$$

or

$$p(\theta, y) = p(\theta|y)\, p(y)\,.$$

Equating these gives Bayes rule.

Bayes Rule: Convenient Form

$$p(\theta|y) = \frac{p(y|\theta)\,p(\theta)}{p(y)}$$

posterior $\propto$ likelihood $\times$ prior.

Example: Estimating allele frequency

n diploid individuals sampled from a population, typed at a locus with two alleles (A,a). y = number of A alleles. Want to estimate θ, frequency of A allele.

$$\text{posterior} \propto \text{likelihood} \times \text{prior}.$$

Likelihood: as before (Hardy–Weinberg Equilibrium).

Prior: For parameters like an allele frequency, constrained to lie in $[0, 1)$, a convenient prior is the beta distribution: $\theta \sim \text{Beta}(\alpha, \beta)$, with density

$$p(\theta; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}, \quad 0 < \theta < 1; \alpha > 0; \beta > 0$$

Beta Distribution: summary

$$p(\theta; \alpha, \beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

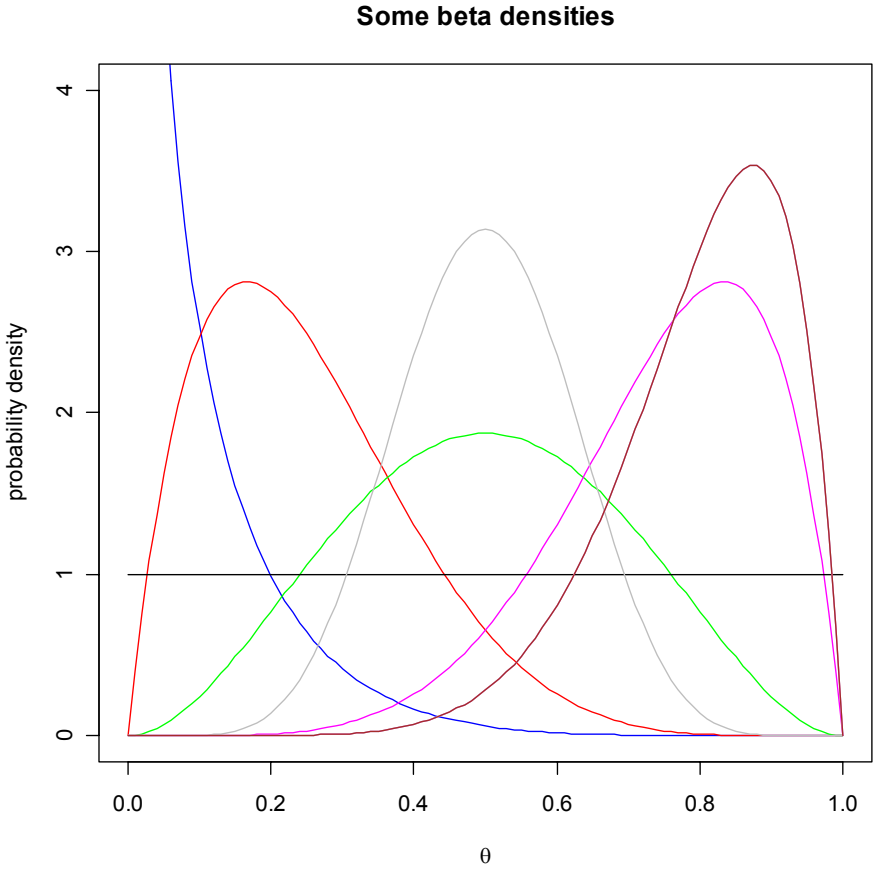
$E(\theta) = \frac{\alpha}{\alpha+\beta}$.

$\text{var}(\theta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

Large values of α, β give small variance: θ becomes peaked about its expectation.

Small values of α, β (< 1) give peaks at 0,1.

$\alpha = \beta = 1$ gives uniform distribution on [0,1).


Some beta densities
probability density
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
θ

Continuing with our example:

$$\text{posterior} \propto \text{likelihood} \times \text{prior}.$$

$$p\left(y|\,\theta\right) \propto \theta^{y}\left(1-\theta\right)^{2n-y}$$

$$p\left(\theta\right) \propto \theta^{\alpha-1}\left(1-\theta\right)^{\beta-1}.$$

The posterior is

$$p\left(\theta|\,y\right) \propto \theta^{\alpha+y-1}\left(1-\theta\right)^{\beta+2n-y-1},$$

the beta distribution with parameters $\alpha + y$ and $\beta + 2n - y$.

$$\theta|y \sim \text{Be}(\alpha + y, \beta + 2n - y).$$

Summarizing posteriors: Point Estimates

The posterior distribution of θ encapsulates our beliefs about θ after seeing the data (and in light of our prior beliefs).

It can be helpful to produce point estimates of θ.

The Bayesian approach to this involves specifying a "loss" function $L(\theta, \widehat{\theta})$ that says how much you "lose" by guessing $\widehat{\theta}$ if the truth is θ. Then choose the value of $\widehat{\theta}$ that minimizes the (posterior) expected loss:

$$E[L(\theta, \widehat{\theta})] = \sum_t L(t, \widehat{\theta}) p(\theta = t|y).$$

[Note, for continuous parameters the sum is replaced by an integral].

The estimate that minimizes this expected loss is called the *Bayes Estimate*.

Example: Posterior mean

If the loss function is "squared loss": $L(\theta, \widehat{\theta}) = (\theta - \widehat{\theta})^2$ then the Bayes Estimate is the *posterior mean*:

$$E(\theta|y) = \sum_t tp(\theta = t|y).$$

In our allele frequency example, the posterior mean is

$$E\left(\theta|\, y\right) = \frac{\alpha + y}{\alpha + \beta + 2n}$$

Compare with the prior mean $\frac{\alpha}{\alpha+\beta}$ and the mle $\frac{y}{2n}$.

Example: Posterior mode

If the loss function is "0-1 loss":

$$L(\theta, \widehat{\theta}) = 0 \text{ if } \theta = \widehat{\theta}; 1 \text{ otherwise}$$

then the Bayes Estimate is the *posterior mode* (the "most likely" posterior value).

The mode of the Beta (α, β) distribution is $\frac{\alpha-1}{\alpha+\beta-2}$.

Example

In our example the posterior mode is $\frac{\alpha+y-1}{\alpha+\beta+2n-2}$.

So if $\alpha = \beta = 1$, posterior mode is $\frac{y}{2n}$.

So, with a uniform prior, the posterior mode = MLE. (Follows from posterior $\propto$ likelihood $\times$ prior.)

A note about nuisance parameters

Consider an inference problem where you are interested in θ, and there is a nuisance parameter η. (e.g. estimating a normal mean, with an unknown variance).

The classical maximum likelihood approach maximizes the likelihood over both θ and η.

The Bayesian approach is to compute the posterior $p(\theta, \eta|y)$, and then to compute the posterior for θ by integrating out η:

$$p(\theta|y) = \int p(\theta, \eta|y) d\eta.$$

You then use this "marginal distribution" to get a point estimate of θ (e.g. posterior mean or mode.) Note that here a uniform prior on (θ, η) does not necessarily give posterior mode(θ) = mle.

Summarizing Posterior: interval estimates

A 95% "credible interval" for θ is an interval I with the property that $\Pr(\theta \in I|y) = 0.95$.

Such intervals are most easily formed by taking the 2.5% and 97.5% percentiles of the posterior density. This yields a "symmetric" or "central" CI.

An alternative (usually considered superior, but harder to do) is to the *highest posterior density* (HPD) region:

> The set of values of θ that contains 95% of the posterior probability, and such that the density within the region is always higher than the density outside.

Bayesian vs frequentist intervals

Strict interpretation of classical (frequentist) confidence interval is awkward:

> For *fixed* (unknown) parameter θ, 95% confidence limits are functions of the data y, say $a(y)$ and $b(y)$, satisfying
>
> $$P\left(a\left(y\right) < \theta < b\left(y\right) \middle| \theta\right) = 0.95.$$
>
> Here y is the random variable and we have to imagine infinite repetitions of the experiment to give meaning to the probability statement.

For the corresponding Bayesian credible interval, θ satisfies

$$P\left(a\left(y\right) < \theta < b\left(y\right) \middle| y\right) = 0.95$$

where θ is now the random variable.

Conjugate Priors

In our example, both the prior and posterior for θ were Beta distributions.

This made for a particularly simple analysis.

This property of a prior has a special name: the Beta distribution is the “conjugate” prior for this problem (binomial sampling).

Conjugate priors are often used, because of the simplicity they introduce. However, it is not necessary to restrict oneself to a conjugate prior.

Considerations for choice of Prior

Prior for θ should capture our knowledge before observing data. Possible sources:

- background scientific knowledge;
- previous studies;
- expert judgements (see e.g. O'Hagan, 1998, and other papers in the same issue).

When “testing” a large number of hypotheses, it can be helpful to think about how many total positive findings one might expect when trying to assess the prior probability that any one is true.

Example: in ongoing genome-wide association studies, a million genetic variants (SNPs) across the genome may be tested for association with a complex disease (eg diabetes).

Perhaps tens or hundreds of them will actually be causally linked to the disease.

Therefore a reasonable prior might be in the range $10/10^6$ to $100/10^6$.

Example: in a candidate gene study one might select, say, 10 genes to study in detail, to assess whether genetic variation in these genes are associated with disease.

Presumably the study would not have been done had it not been considered at least reasonably likely that one of the genes would be associated with the disease.

Based on this, a prior probability of 1/10 for each gene containing genetic variation associated with the disease, might be reasonable.

Example 2: Assignment Problem

Consider the problem of estimating the origin of an individual, based on genetic data.

One bi-allelic locus with alleles A, a.

Allele frequencies of A in two populations are f_1 and f_2.

What is the probability that an individual with genotype AA came from population 1 vs population 2?

That is, what is $\Pr(\text{Pop } 1|AA)$?

We need a prior probability that the individual came from population 1 vs 2.

Assume the two are equally likely *a priori*, so prior probability is 0.5.

Then apply Bayes Theorem:

$$\Pr(\text{Pop 1}|AA) \propto \Pr(AA|\text{Pop 1}) \Pr(\text{Pop 1}) = f_1^2 0.5$$

$$\Pr(\text{Pop 2}|AA) \propto \Pr(AA|\text{Pop 2}) \Pr(\text{Pop 2}) = f_2^2 0.5$$

These must sum to 1, so this gives us the constant of proportionality:

$$\Pr(\text{Pop 1}|AA) = 0.5 f_1^2 / (0.5 f_1^2 + 0.5 f_2^2)$$
$$\Pr(\text{Pop 2}|AA) = 0.5 f_2^2 / (0.5 f_1^2 + 0.5 f_2^2)$$

Example: Assignment Problem with estimated allele frequencies

Assume now that f_1 and f_2 are unknown, but are to be estimated from 10 "reference" individuals sampled from each population.

Suppose that in 10 individuals, we see 1 copy of A in population 1, and 0 copies of A in population 2.

What is the estimated probability that an AA individual came from population 1?

Naive Frequentist Analysis: answer = ?

Naive Bayes Analysis: answer = ?

Bayesian Inference: summary

Bayesian inference proceeds by

1. specifying what is known about θ before obtaining the data as the prior $p(\theta)$;
2. collecting the data y and specifying its distribution $p(y|\theta)$ (or the likelihood function);
3. calculating the posterior $p(\theta|y)$ from Bayes rule.

Monte Carlo (No Markov Chains. . . Yet)

Goals of this lecture:

- Define the Monte Carlo method generally
- Explain why Monte Carlo is useful
- Understand variance of Monte Carlo estimators

all without talking about Markov chains. . .

The origin of Monte Carlo:

First Reference:

METROPOLIS, N. AND S. ULAM. 1949. The Monte Carlo method. *Journal of the American Statistical Association*. 44:335–341.

The name “Monte Carlo” was coined originally by John von Neumann and Stanislaw Ulam who used the phrase as a code word at Los Alamos for their simulation-based computational methods used to solve the problem of initiating fusion in a thermonuclear bomb.

Earlier examples exist. (for example, the Comte de Buffon and his needle in the mid-1700’s).

With computer power increasing all the time, Monte Carlo methods are more widespread than ever, and are used in a wide range of applications, most of them of a more humanitarian bent than bomb design.

Journal of the AMERICAN STATISTICAL ASSOCIATION

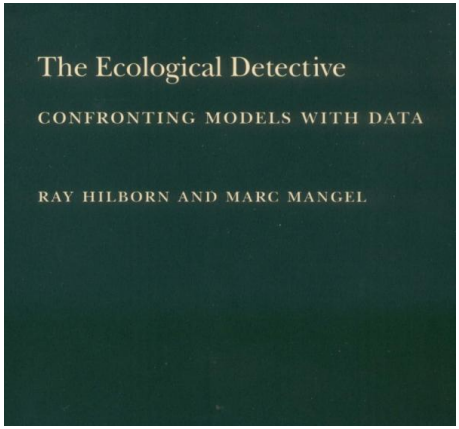
SEPTEMBER 1949

In search of a definition for Monte Carlo:

There are few concise, yet general definitions in the literature. Some, like Ripley (1987) reserve “Monte Carlo” only for “Monte Carlo Integration” and “Monte Carlo Tests,” and refer to all else as “stochastic simulation.”

Others use Monte Carlo to refer specifically to the generation of pseudo-random numbers:

“One way to increase our confidence in the methods we use is to test models and methods on sets of data in which we know exactly what is happening. . . A useful method for generating such data is called the Monte Carlo method. . . . The Monte Carlo method uses random number generators for the construction of data.”



A general definition of the Monte Carlo method:

DEFINITION: *Monte Carlo is the art of approximating an expectation by the sample mean of a function of simulated random variables.*

This definition is general enough to encompass everything that has been called "Monte Carlo," yet also makes clear its essence in familiar terms: Monte Carlo is about invoking laws of large numbers to approximate expectations.

REVIEW: *Expectations*.

$$\mathbb{E}[g(X)] = \int_{x \in \mathcal{X}} g(x) f_X(x) dx \quad \text{or} \quad \mathbb{E}[g(X)] = \sum_{x \in \mathcal{X}} g(x) f_X(x)$$

REVIEW: *Weak Law of Large Numbers*.
Let $X_1, X_2, \ldots, X_n$ be iid r.v.'s with $\mathbb{E}|X_i| < \infty$, and $\bar{X}_n = \frac{1}{n} \sum_1^n X_i$. Then

$$\lim_{n \to \infty} P(|\bar{X}_n - \mathbb{E}X_i| > \epsilon) = 0 \quad \text{for any } \epsilon > 0$$

A Suggestion from the weak law of large numbers::

Any expectation may be approximated by the sample mean of n random variables (and it will work better if n is large).

$$\mathbb{E}[g(X)] \approx \frac{1}{n} \sum_{i=1}^{n} g(x^{(i)}) \quad \text{with} \quad X^{(i)} \sim f_X$$

Why estimating expectations is useful:

Usually, any quantity of interest may be expressed as the expected value of a function of some random variable. Importantly:

Probabilities:

$$P(X \in \mathcal{A}) = \mathbb{E}[I_{\{\mathcal{A}\}}(X)]$$

where $I_{\{\mathcal{A}\}}(X)$ is the *indicator function* taking the value 1 when $X \in \mathcal{A}$ and 0 otherwise.

Integrals: For a simple example, let U be a uniform r.v. on the interval $[a, b)$ with pdf $f_U(u) = 1/(b-a)$. Hence

$$\int_a^b q(x)dx = (b-a)\int_a^b q(x)\frac{1}{b-a}dx = (b-a)\mathbb{E}[q(U)]$$

We see here a case where Monte Carlo applies to a purely deterministic problem.

Discrete Sums: In the same vein as above, just as any integral can be approximated by Monte Carlo, so can any sum. For another simple, uniform example, let W be a discrete random variable that takes all values w in the set $\mathcal{A}$ with equal probability p. Then, the sum $\sum_{x \in \mathcal{A}} q(x)$ is easily approximated by Monte Carlo:

$$\sum_{x \in \mathcal{A}} q(x) = \frac{1}{p} \sum_{x \in \mathcal{A}} q(x) p = \frac{1}{p} \mathbb{E}[q(W)].$$

This is particularly useful in statistical genetics, because many probabilities of interest may be expressed as an intractable sum over latent variables.

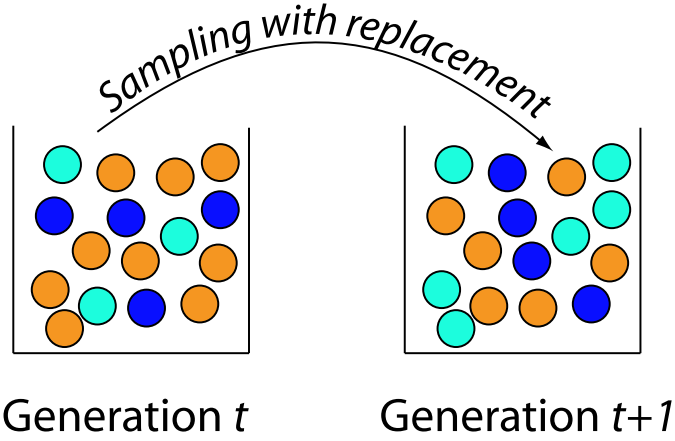
The bottom line is that, since you can cast any quantity as an expectation; you can (in theory) approximate any quantity by Monte Carlo. In the simplest case, when X is distributed according to f_X.

$$\mathbb{E}[g(X)] \approx \frac{1}{n} \sum_{i=1}^{n} g(x^{(i)}) \quad \text{with} \quad X^{(i)} \sim f_X$$

Genetics example I: Estimating probabilities in the Wright-Fisher model:

The Wright-Fisher model underlies most population genetics theory that deals with the descent of genes in a finite population from one generation to the next.

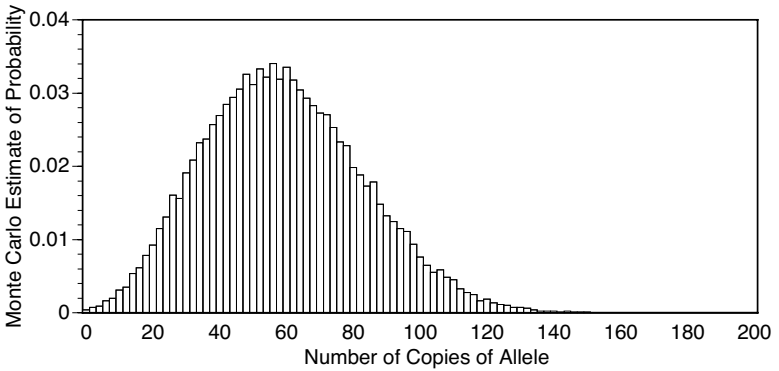
A schematic of its underlying assumptions:



Genetics example I: Estimating probabilities in the Wright-Fisher model:

Let X be the frequency of the A allele in a Wright-Fisher population of size $2N = 200$ after 14 generations of drift, having started from a frequency of 60 out of 200.

The distribution of X can be approximated by simulating $n = 50,000$ instances:



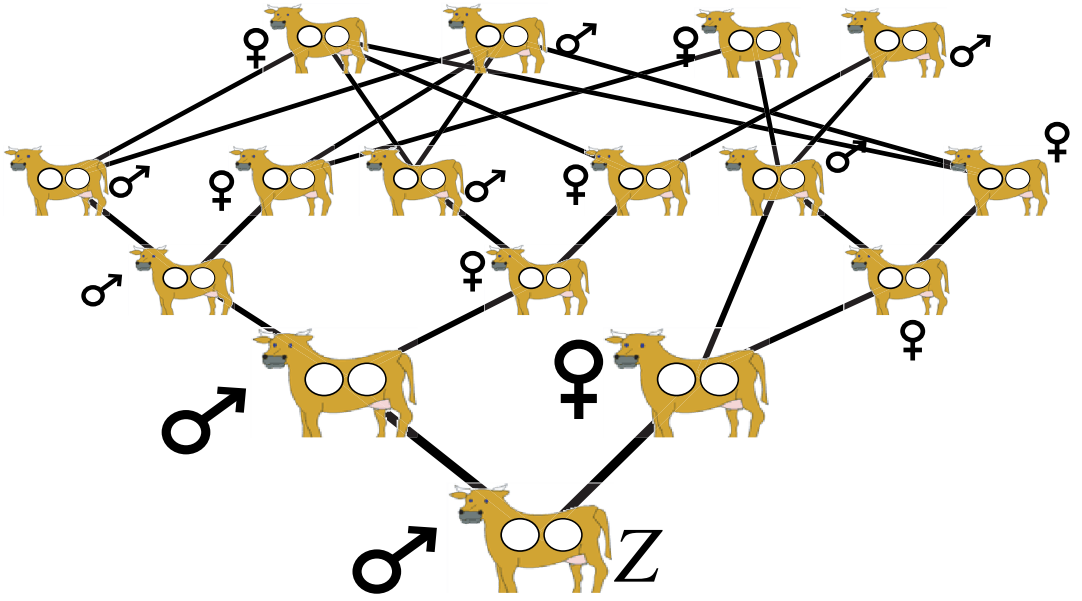
Each histogram column is an approximation of an expectation:

$$\begin{aligned} P(a \leq X < a+2) &= \mathbb{E}[I_{\{x:a \leq x < a+2\}}(X)] \\ &\approx \frac{1}{n} \sum_{i=1}^{n} I_{\{x:a \leq x < a+2\}}(x^{(i)}) \end{aligned}$$

for $a = 0, 2, \ldots, 200$, where each $x^{(i)}$ is an independent realization of the number of A alleles at $t = 14$ in the Wright-Fisher model.

Genetics example II: estimating a sum over latent variables:

Wright and McPhee (1925) estimating inbreeding of individuals within cattle pedigrees.



In terms of latent variables:

- The maternal (m) and paternal (p) genes in individual z are direct descendants from an *unobserved* lineage of ancestral gene copies, *i.e.*,
- $A_m = \{a_{m,1}, a_{m,2}, \ldots, a_{m,n}\}$ and $A_p = \{a_{p,1}, a_{p,2}, \ldots, a_{p,n}\}$
- The individual is inbred at a locus if the ancestors of the maternal gene and the paternal gene are the same at any point in their lineages, *i.e.*, if $A_m \cap A_p \neq \emptyset$.
- The probability that an individual is inbred at a locus is then the sum over latent variables:

$$P(\text{inbred}|\text{pedigree}) = \sum_{A_m, A_p} [1 - I_{\{\emptyset\}}(A_m \cap A_p)] P(A_m, A_p)$$

where $P(A_m, A_p)$ follow from Mendel's laws. Note that A_m and A_p are independent up until they intersect, then follow the same lineage back in time.

- This sum is an expectation of the indicator function, with respect to the joint distribution of A_m and A_p, given the pedigree:

$$P(\text{inbred}|\text{pedigree}) = \mathbb{E}[1 - I_{\{\emptyset\}}(A_m \cap A_p)]$$

so it may be estimated by Monte Carlo:

$$\begin{aligned} P(\text{inbred}|\text{pedigree}) &= \mathbb{E}[1 - I_{\{\emptyset\}}(A_m \cap A_p)] \\ &\approx \frac{1}{n}\sum_{i=1}^{n}[1 - I_{\{\emptyset\}}(A_m^{(i)} \cap A_p^{(i)})] \end{aligned}$$

where $A_m^{(i)}$ and $A_p^{(i)}$ are simulated from their respective distributions, which can be done by flipping a coin, until they intersect (and hence add 1 to the sum) or reach pedigree founders without intersecting (adding 0 to the sum).

Variance of Monte Carlo estimators—iid case:

A Monte Carlo estimator is simply a random variable itself—a sum of random variables:

$$G_n = \frac{1}{n}\sum_{i=1}^{n} g(X^{(i)})$$

So, if the X_i are independent[1] the variance of G_n is easily computed as the variance of a sum of independent R.V.'s:

$$\begin{aligned} \mathrm{Var}(G_n) &= \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} g(X^{(i)})\right) = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}[g(X^{(i)})] \\ &= \frac{\mathrm{Var}[g(X^{(i)})]}{n} \end{aligned}$$

$\mathrm{Var} \downarrow$ when $n \uparrow$ or if $\mathrm{Var}[g(X^{(i)})]$ can be reduced[2].

[1] Note that throughout most of the remainder of the course, we will deal with samples of correlated, non-independent X_i's. This is just a warm-up.

[2] We'll take this up later in our discussion importance sampling

Markov Chains (With Some MCMC at the End)

Goals of this lecture:

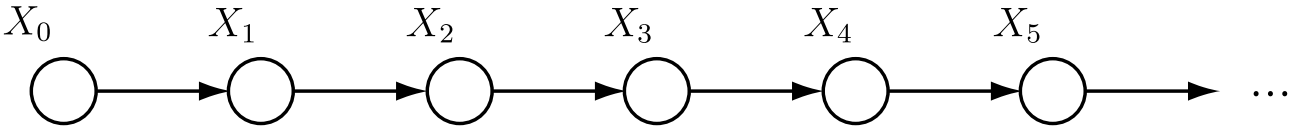
- Introduce Markov chains
- Highlight the properties of Markov chains needed to understand MCMC
- Discuss the weak law of large numbers for the number of passages through a recurrent state in an ergodic Markov chain
- Introduce the Metropolis-Hastings algorithm
- Explain why the M-H algorithm is useful

Definition of a Markov chain:

X_t , $t = 0, 1, 2 \ldots$, having a joint distribution such that

$$P(X_t|X_{t-1}, X_{t-2}, \ldots, X_0) = P(X_t|X_{t-1}) \ \forall \, t$$

Schematically:



And the variables X_t can be scalars or vectors.

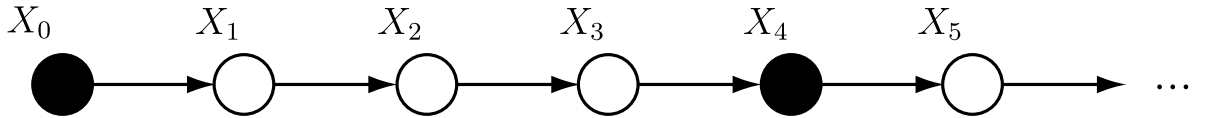
- For most of the *examples* in this lecture, X_t will be univariate.
- When X_t includes all the variables in an MCMC application, however, it is typically high-dimensional.

A genetics example—the Wright-Fisher population:

Let X_t denote the number of alleles of type A in a Wright-Fisher population of size N diploids. Then:

$$P(X_t = j | X_{t-1} = i) = \frac{2N!}{(2N-j)!j!} \left(\frac{i}{2N}\right)^j \left(\frac{2N-i}{2N}\right)^{2N-j}$$

- Important point: conditional independence $\neq$ independence. If you don't know X_{t-1}, then, of course X_t depends on X_{t-2}



- NB: Do not think that MCMC is useful in genetics because the problems in genetics involve Markov chains. The Markov chain underlying Markov chain Monte Carlo and any Markov chains involved in a statistical genetics model may be quite distinct entities.

Transition Probability Matrices:

We can write down the transition probabilities from all values of X_t to all values of X_{t+1} in a matrix:

$$\begin{array}{c} \nearrow \\ 0 \\ 1 \\ 2 \\ 3 \\ \vdots \\ 2N \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & & 1 & & 2 & \cdots & 2N \end{array} \\ \begin{pmatrix} P(0 \to 0) & P(0 \to 1) & P(0 \to 2) & \cdots & P(0 \to 2N) \\ P(1 \to 0) & P(1 \to 1) & P(1 \to 2) & \cdots & P(1 \to 2N) \\ P(2 \to 0) & P(2 \to 1) & P(2 \to 2) & \cdots & P(1 \to 2N) \\ P(3 \to 0) & P(3 \to 1) & P(3 \to 2) & \cdots & P(3 \to 2N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P(2N \to 0) & P(2N \to 1) & P(2N \to 2) & \cdots & P(2N \to 2N) \end{pmatrix} \end{array}$$

Here we use $P(i \to j)$ as a shorthand for $P(X_{t+1} = j | X_t = i)$.

Such *transition probability matrices* form the basis for much of the classical analysis of Markov chains.

The above t.p.m. is a bit unwieldy for examples, so consider another simple model. . .

Random walk with scattering boundaries:

Imagine a random walk on the integers from 1 to 5. From 1 or 5, the walk goes to any state with equal probability, and from the remaining states, the walk may take steps of size 0 or 1, but they are biased toward the center.

For example, consider the t.p.m:

$$\mathbf{P} = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} .2 & .2 & .2 & .2 & .2 \\ .2 & .3 & .5 & 0 & 0 \\ 0 & .3 & .4 & .3 & 0 \\ 0 & 0 & .5 & .3 & .2 \\ .2 & .2 & .2 & .2 & .2 \end{pmatrix} \end{array}$$

Computer Demo: `tpm`

Some things to note:

- The rows of $\mathbf{P}$ sum to one—they are probabilities that must sum to one
- The columns need not sum to one
- Probabilities after one step can be computed by matrix multiplication. *i.e.*, if: $\boldsymbol{v_0} = (0, 1, 0, 0, 0)$, then the probabilities of being in states 1,2,. . . ,5 after one step of the chain are given by:

$$\boldsymbol{v_0}\mathbf{P} = \boldsymbol{v_1} = (.2, .3, .5, 0, 0)$$

- Probabilities after two steps are:

$$\boldsymbol{v_1}\mathbf{P} = (\boldsymbol{v_0}\mathbf{P})\mathbf{P} = \boldsymbol{v_2}$$

- and probabilities after n steps are

$$\boldsymbol{v_n} = \boldsymbol{v_0}\mathbf{P}^n$$

  Vectors $\boldsymbol{v_t}$ are taken to be row vectors.

Limiting Distributions:

A class of Markov chains called *ergodic Markov chains*[3] have the property that

$$\lim_{n\to\infty} \mathbf{P}^n = \begin{pmatrix} \pi \\ \pi \\ \vdots \\ \pi \end{pmatrix}$$

where π is (in this discrete case) a vector referred to as the *limiting distribution* of the ergodic Markov chain.

Take our random walk example:

$$\mathbf{P}^1 = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} .2 & .2 & .2 & .2 & .2 \\ .2 & .3 & .5 & 0 & 0 \\ 0 & .3 & .4 & .3 & 0 \\ 0 & 0 & .5 & .3 & .2 \\ .2 & .2 & .2 & .2 & .2 \end{pmatrix} \end{array}, \quad \mathbf{P}^2 = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.12 & 0.20 & 0.36 & 0.20 & 0.12 \\ 0.10 & 0.28 & 0.39 & 0.19 & 0.04 \\ 0.06 & 0.21 & 0.46 & 0.21 & 0.06 \\ 0.04 & 0.19 & 0.39 & 0.28 & 0.10 \\ 0.12 & 0.20 & 0.36 & 0.20 & 0.12 \end{pmatrix} \end{array}$$

[3] Conditions conferring ergodicity will be discussed a little later.

$$\mathbf{P}^3 = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.088 & 0.216 & 0.392 & 0.216 & 0.088 \\ 0.084 & 0.229 & 0.419 & 0.202 & 0.066 \\ 0.066 & 0.225 & 0.418 & 0.225 & 0.066 \\ 0.066 & 0.202 & 0.419 & 0.229 & 0.084 \\ 0.088 & 0.216 & 0.392 & 0.216 & 0.088 \end{pmatrix} \end{array}$$

$$\mathbf{P}^5 = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.075 & 0.219 & 0.412 & 0.219 & 0.075 \\ 0.074 & 0.220 & 0.415 & 0.218 & 0.073 \\ 0.072 & 0.220 & 0.415 & 0.220 & 0.072 \\ 0.073 & 0.218 & 0.415 & 0.220 & 0.074 \\ 0.075 & 0.219 & 0.412 & 0.219 & 0.075 \end{pmatrix} \end{array}$$

$$\mathbf{P}^\infty = \begin{array}{c} \nearrow \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.07317 & 0.2195 & 0.4146 & 0.2195 & 0.07317 \\ 0.07317 & 0.2195 & 0.4146 & 0.2195 & 0.07317 \\ 0.07317 & 0.2195 & 0.4146 & 0.2195 & 0.07317 \\ 0.07317 & 0.2195 & 0.4146 & 0.2195 & 0.07317 \\ 0.07317 & 0.2195 & 0.4146 & 0.2195 & 0.07317 \end{pmatrix} \end{array}$$

The limiting distribution in this case is

$$\pi = (0.07317,\ 0.2195,\ 0.4146,\ 0.2195,\ 0.07317)$$

- This means that if you start the chain from any of the five states, after a sufficient (and not very many in this case) number of steps, the probability that it will be found in any of the five states is essentially independent of its starting state.
- And, as we have already seen, it means that a "time-average" over the chain converges to the limiting distribution.
- This second point is asserted by the weak law of large numbers for ergodic Markov chains which states that as the number of transitions (or steps) in an ergodic chain tends to infinity, the proportion of time the chain spends in a state tends to the limiting probability (*i.e.*, the appropriate component of π) of that state[4].
- Thus, the states visited by an ergodic Markov chain may be used to compute a Monte Carlo average. That is MCMC.

[4] See Feller (1957)

The use of Markov Chains in Monte Carlo:

Monte Carlo with a Markov chain is pretty much the same as before:

$$\mathbb{E}[g(X)] = \frac{1}{n}\sum_{i=1}^{n} g(x^{(i)})$$

except that now, the $x^{(i)}$ are states visited by a Markov chain having a limiting distribution that we wish to sample from.

To implement this we must be able to construct an appropriate Markov chain. It must:

- be ergodic
- have the right limiting distribution

The remainder of the lecture shows how this is done.

Important points to keep in mind:

1. If you can simulate *independent* samples, $x^{(1)}, \ldots, x^{(n)}$, then, by all means, do so, and avoid MCMC if you can.
2. MCMC is most useful when the desired distribution to be sampled from is "known only up to scale"—this means that the "shape" of the distribution is known but its normalizing constant is not.

   A germane example of a distribution known only up to scale is the Bayesian posterior distribution of θ given data Y:

   $$P(\theta|Y) = \frac{P(Y|\theta)P(\theta)}{\int_\theta P(Y|\theta)P(\theta)d\theta}$$

   The denominator is a constant w.r.t. θ. It is the (typically unknown) normalizing constant. However, the numerator is the joint density of the data Y and the parameters θ, which is often easy to compute. This is why MCMC is so useful to Bayesians.

Conditions ensuring a chain is ergodic:

A Markov chain is ergodic if:

1. *It has no transient states*, *i.e.*, there are no states with an expected time to recurrence of ∞. (This is not an issue with a finite state space).
2. *It is irreducible*, *i.e.*, any state in the state space is reachable from any other state in the state space in a finite number of steps.
3. *It is aperiodic*, *i.e.*, there exists no pair of states i and j such that the probability of reaching j from i is non-zero only if the number of steps is an integer multiple of some period τ.

In discrete "tree-like" state spaces, like pedigrees, reducibility can be an issue. We take this up briefly in Session 10.

General balance and the stationary distribution of a Markov chain:

Recall that we wish to construct an ergodic Markov chain with limiting distribution π equal to some distribution that we wish to sample from. How might we do that?

One possibility is via a theorem which tells us about a property of π:

If π is the limiting distribution of the ergodic Markov chain with t.p.m. $\mathbf{P}$, then π is the unique stationary distribution that satisfies the general balance equation

$$\pi \mathbf{P} = \pi$$

So, if we can find $\mathbf{P}$ such that its unique stationary distribution is π, and π is the distribution we wish to sample from, then we can use the chain defined by $\mathbf{P}$ in MCMC.

This is, however, not a tractable approach in any problem of consequence. In general it will be harder to (or just as difficult to) solve the general balance equation for π as it will be to draw independent samples from π.

The main problem with solving the general balance equation is that all possible states in the distribution π must be considered simultaneously. In most interesting problems, that number of states can be astronomically huge.

The solution to this conundrum is to not try to solve the general balance equation directly, but, instead satisfy a "locally-defined" balance condition (so that only two states in the state space need be considered simultaneously—rather than all of the states, simultaneously), AND do this in a way that ensures that general balance is satisfied.

This is quite a lovely thing.

Time-reversible Markov chains and detailed balance:

A special class of Markov chains are called “time-reversible” Markov chains, because if you were to watch them running “backward in time” they would look just the same as the chain running “forward in time.”

The salient feature of such chains is that they satisfy the *detailed balance* (also called the *local balance*) condition.

Detailed balance with respect to π between a pair of states i and j, is satisfied by a Markov chain with a stationary distribution π, and a t.p.m $\mathbf{P}$ having elements $P(x \to y)$ if

$$\pi(i)P(i \to j) = \pi(j)P(j \to i).$$

It is easy to show (by summing over all states i) that if detailed balance holds for every pair of states, then general balance is also satisfied.

The Metropolis-Hastings Algorithm[5]:

The M-H algorithm provides a way to perform steps in a Markov chain that satisfy detailed balance.

Imagine you wish to simulate a dependent sample from a target distribution f, and you are currently in state i. The recipe is:

- Propose changing the state from state i to a new state j. Draw the state j from a proposal distribution, $q(j|i)$ which may be conditional on i.
- Accept and move to the new state j with probability R equal to the lesser of 1 or the *Hastings ratio*:

$$R(i \rightarrow j) = \min \left\{ 1, \frac{f(j)}{f(i)} \times \frac{q(i|j)}{q(j|i)} \right\}$$

  If you don't accept the move to state j, then stay where you are.

[5] Metropolis et al. (1953) and Hastings (1970).

M-H algorithm satisfies detailed balance:

To show the M-H algorithm satisfies detailed balance w.r.t. f, we write down $P(i \to j)$ and $P(j \to i)$, for any i and j, under the M-H algorithm.

$P(i \to j)$ is the product of the probabilities that we:

1. Propose going from i to j. $\quad \text{Prob} = q(j|i)$
2. Accept that proposal. $\quad \text{Prob} = R(i \to j) = \min\{1, \frac{f(j)}{f(i)} \frac{q(i|j)}{q(j|i)}\}$

And $P(j \to i)$ can be found similarly [using $R(j \to i)$].

Now, there are two possible cases:

Case I: $f(j)q(i|j) > f(i)q(j|i)$, in which case
$R(i \to j) = 1 \quad$ and $\quad R(j \to i) = \frac{f(i)}{f(j)} \frac{q(j|i)}{q(i|j)}$

Case II: $f(j)q(i|j) \leq f(i)q(j|i)$, in which case
$R(i \to j) = \frac{f(j)}{f(i)} \frac{q(i|j)}{q(j|i)} \quad$ and $\quad R(j \to i) = 1.$

M-H algorithm satisfies detailed balance (cont'd):

So, $P(i \to j)$ and $P(j \to i)$ in those cases are:

Case I:

$$P(i \to j) = q(j|i) \qquad P(j \to i) = q(i|j) \times \frac{f(i)}{f(j)} \frac{q(j|i)}{q(i|j)}$$

Case II:

$$P(i \to j) = q(j|i) \times \frac{f(j)}{f(i)} \frac{q(i|j)}{q(j|i)} \qquad P(j \to i) = q(i|j)$$

Both of which, with a little rearranging, yield:

$$f(i)P(i \to j) = f(j)P(j \to i)$$

which is the detailed balance equation.

M-H algorithm example—beta distribution[6]:

Let θ be a variable with probability density function

$$f(\theta) \equiv \text{Beta}(74, 128) = \frac{\Gamma(74+128)}{\Gamma(74)\Gamma(128)}\theta^{73}(1-\theta)^{127}$$

Note that this is the posterior distribution of the frequency θ of allele A at a locus, given a uniform prior, and a sample of 100 diploids in which are found 73 copies of allele A.

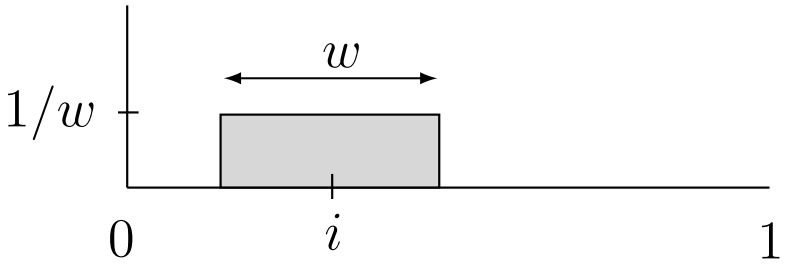
This distribution is known exactly, but for illustration, we will construct a Markov chain that has limiting distribution f, and sample from it.

[6]Up to this point we have been dealing with discrete Markov chains. The extension to continuous state spaces is straightforward. In such cases, π, P, q, and f may be taken as probability density functions, rather than as probability mass functions, and the general balance equation is expressed by a collection of integrals rather than in terms of matrix multiplication: $\int_x \pi(x)P(x \to y)dx = \pi(y)\ \forall y$

Step one: Choose a proposal distribution:

One is free to choose any proposal distribution q. The only property that it should satisfy is that if $q(i|j) > 0$, then $q(j|i)$ should also be positive[7]. Otherwise you end up wasting some time.

We will choose a uniform density for q with width w, centered on the current state i. It looks like:



Thus, $q(i|j) = 1/w$ for all i and j. (Note that if $j \geq 1$ or $j \leq 0$ then $f(j) = 0$ and we reject the proposal immediately.)

[7] And the proposal distributions used should fulfill irreducibility, *etc.*. Clearly, also, some proposal distributions will work better than others.

Step two: Compute the Hastings Ratio:

$$\frac{f(j)}{f(i)}\frac{q(i|j)}{q(j|i)} =$$

$$\frac{\Gamma(74+128)}{\Gamma(74)\Gamma(128)} \times \frac{\Gamma(74)\Gamma(128)}{\Gamma(74+128)} \times \frac{j^{73}(1-j)^{127}}{i^{73}(1-i)^{127}} \times \frac{1/w}{1/w}$$

$$= \left(\frac{j}{i}\right)^{73} \left(\frac{1-j}{1-i}\right)^{127}$$

Notice that the normalizing constant cancels out. (That *always* happens). And, in this case, q cancels out (*only because q is symmetrical*).

Computer Demo: `beta_sim`

Conclusions:

1. MCMC is just Monte Carlo with samples drawn from a Markov chain
2. The Markov chain in MCMC is constructed by concatenating moves together, each of which satisfies detailed balance
3. The rest of the module will explore methods for implementing MCMC in problems relevant to statistical genetics

```
# Session 3: Introduction to MCMC in R (Computing Practical)
#
# Note: anything following a # sign is a comment, and will be ignored by R
# (this should make it easier to cut and paste things)
#
#
# Example 1: sampling from an exponential distribution using MCMC
#
#
# first: a few example lines of R

R
x = seq(from = 0, to = 10, length=20)
?seq    # to get help on any command in R you can type ?command
x       # to see what an object looks like, just type its name
y = exp(-x)
y       # note that y is a vector, because x is a vector
plot(x,y)
lines(x,y) #lines adds lines to an existing plot

#
# We will learn more R as we go along: ask questions if there is
# something you don't understand!
#
```

```
# Now: set up a target function we want to sample from
# In this case we are going to use the density of the exponential distribution
# (What follows is not the best way to do things, but I have tried
# to make it easy to follow for novice users)
#

target = function(x){
if(x<0){
return(0)}
else {
return( exp(-x))
}
}

#Try computing a couple of values

target(1)
target(-1)
```

```
# Next, we will program a Metropolis--Hastings scheme to sample
# from a distribution proportional to the target

x = rep(0,1000)
x[1] = 3     #this is just a starting value, which I've set arbitrarily to 3
for(i in 2:1000){
currentx = x[i-1]
proposedx = currentx + rnorm(1,mean=0,sd=1)
A = target(proposedx)/target(currentx)
if(runif(1)<A){
x[i] = proposedx       # accept move with probabily min(1,A)
} else {
x[i] = currentx        # otherwise "reject" move, and stay where we are
}
}

# note that x is a realisation of a Markov Chain
# we can make a few plots of x:
plot(x)
hist(x)
```

```
# We can wrap this up in a function to make things a bit neater,
# and make it easy to try changing starting values and proposal distributions


easyMCMC = function(niter, startval, proposalsd){
x = rep(0,niter)
x[1] = startval
for(i in 2:niter){
currentx = x[i-1]
proposedx = rnorm(1,mean=currentx,sd=proposalsd)
A = target(proposedx)/target(currentx)
if(runif(1)<A){
x[i] = proposedx       # accept move with probabily min(1,A)
} else {
x[i] = currentx        # otherwise "reject" move, and stay where we are
}
}
return(x)
}
```

```
#Now we'll run the MCMC scheme 3 times,
#and look to see how similar the results are


z1=easyMCMC(1000,3,1)
z2=easyMCMC(1000,3,1)
z3=easyMCMC(1000,3,1)

plot(z1,type="l")
lines(z2,col=2)
lines(z3,col=3)


par(mfcol=c(3,1)) #rather odd command tells R to put 3 graphs on a single page
hist(z1)
hist(z2)
hist(z3)
```

```
# Exercises: use the function easyMCMC to explore the following:
#
# 1a) how do different starting values affect the MCMC scheme?
#
# 1b) what is the effect of having a bigger/smaller proposal standard deviation?
#
# 1c) try changing the target function to the following
#

target = function(x){
return((x>0 & x <1) + (x>2 & x<3))
}

# What does this target look like?
# What happens if the proposal sd is too small here? (try eg 1 and 0.1)
```

MCMC Convergence

Markov chain Monte Carlo relies on the fact that, asymptotically (i.e. after infinite iterations), simulated values from the Markov chain behave as if they were sampled from the joint posterior distribution we are interested in.

In practice we can't perform infinitely many iterations before we regard the simulated values as coming from the posterior distribution. Thus we hope that provided we do enough iterations our simulated values will be sufficiently close to coming from the posterior distribution. But how do we know when we are sufficiently close?

Assessing Convergence

Although “formal” methods exist for assessing convergence, I tend to prefer less formal methods.

Here are some guidelines for assessing problems:

- Always run the MCMC sampler multiple times from different starting points.
- Compare results from different starting points: do they differ enough to change your conclusions?
- Plot trace plots of different parameters against iteration number: do they look “sticky?”

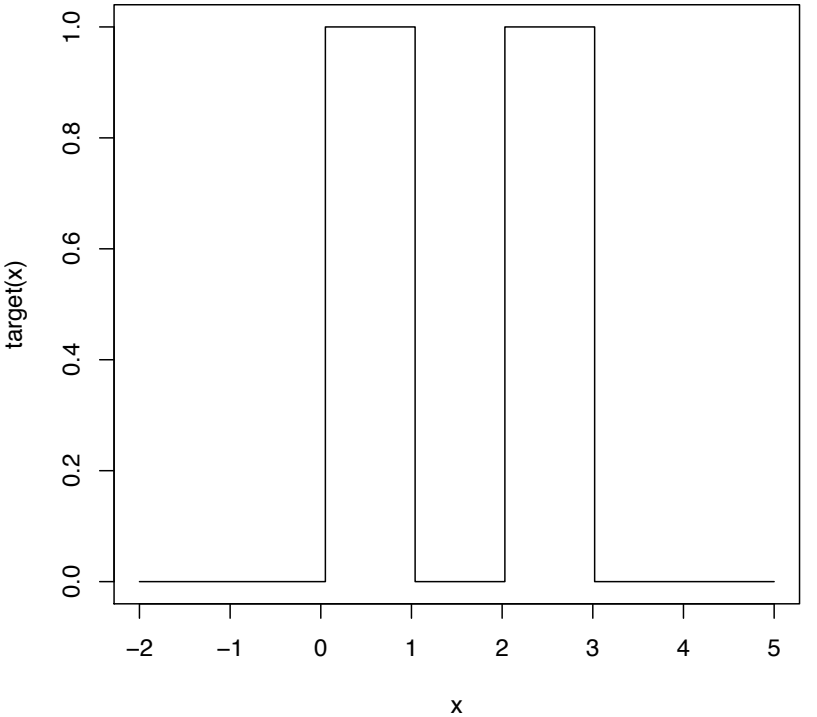
Improving Convergence

If convergence appears to be a problem, you can try:

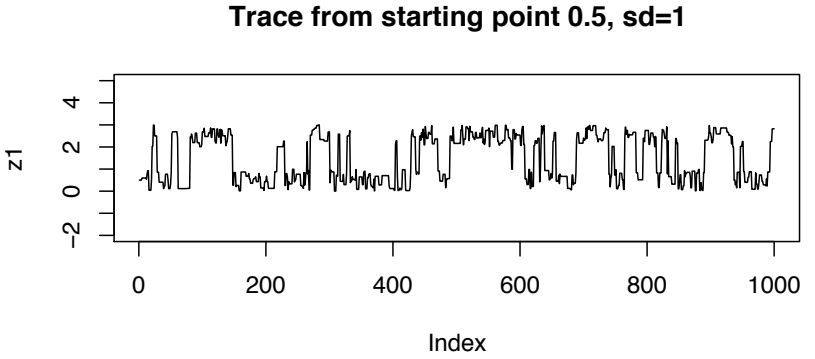
- Running the sampler longer!
- Changing the model (convergence problems tend to occur more often when the model is wrong).
- Tuning the MH proposal standard deviation.
- Redesigning the sampler.

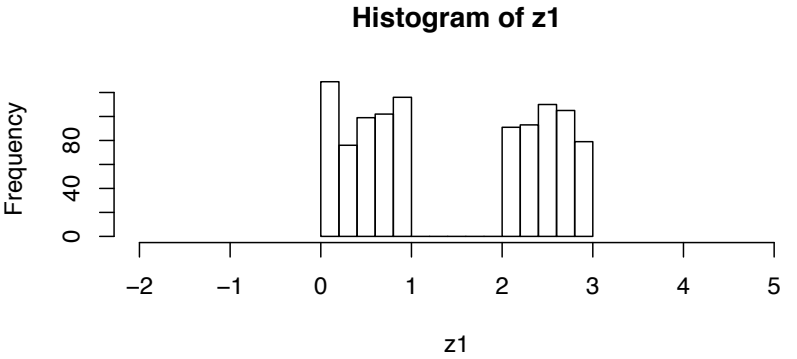
In some difficult problems, getting a sampler to converge is very difficult. In these cases MCMC can sometimes still provide useful output, if started from a “good” starting point. Think of it as a local exploration of a “good” solution.

Example: A bimodal target

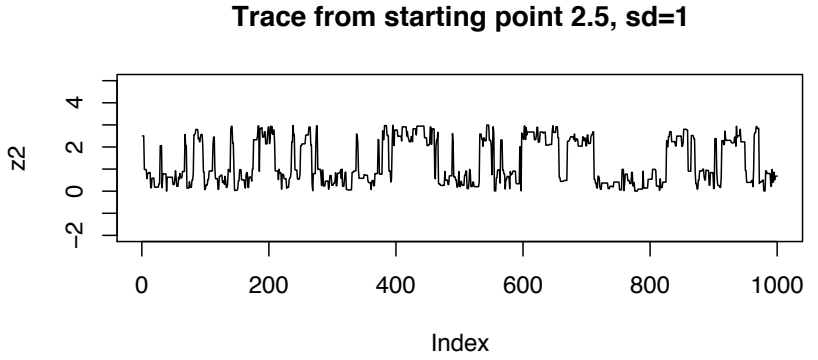


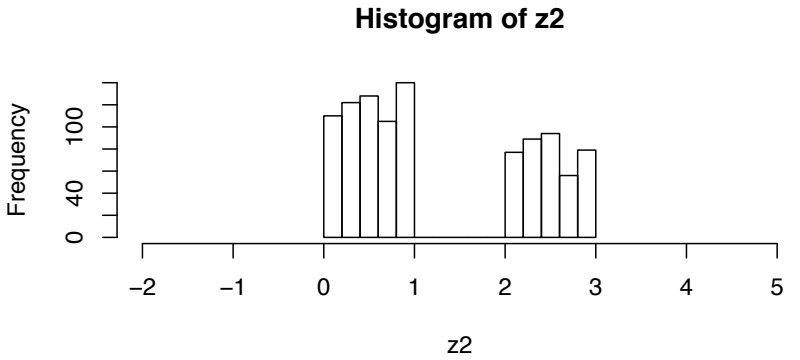
SD = 1 gives satisfactory results



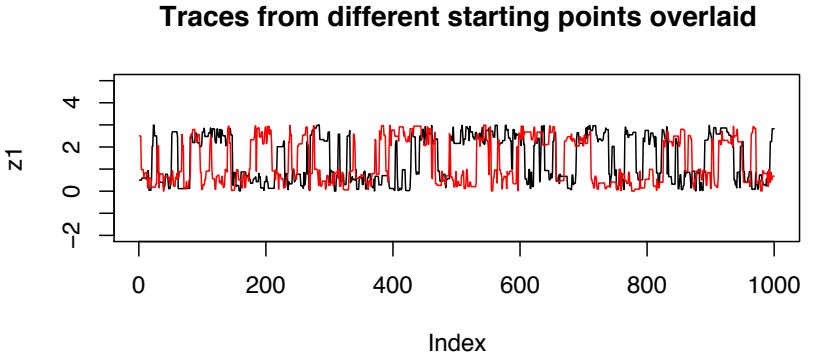


SD = 1: different starting points agree well

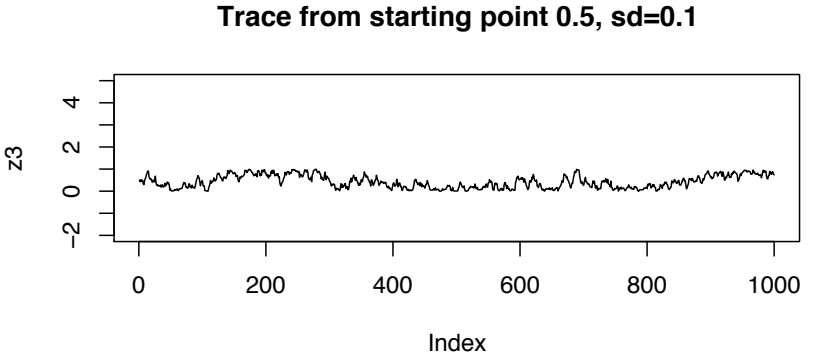


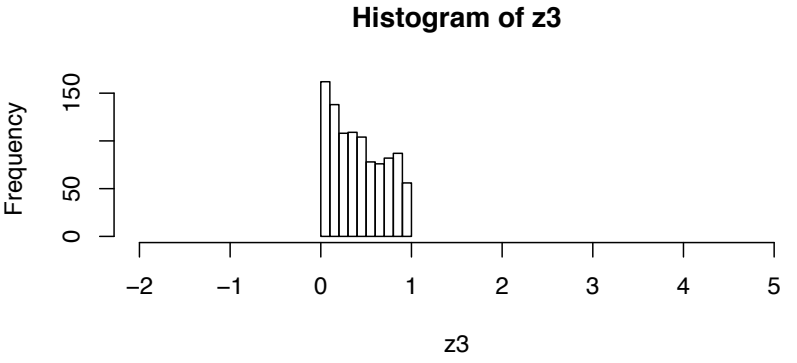


SD = 1: different starting points agree well

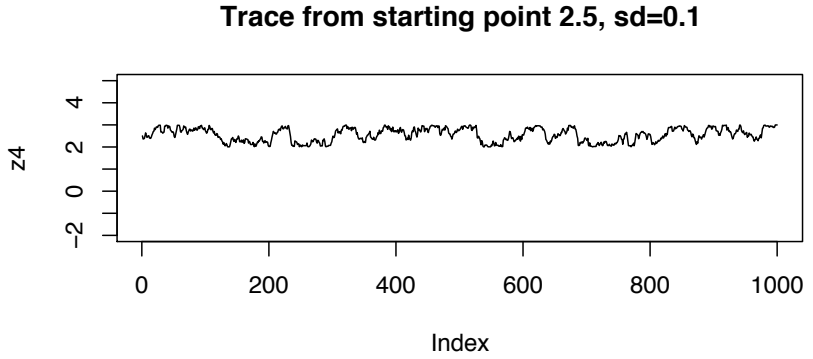


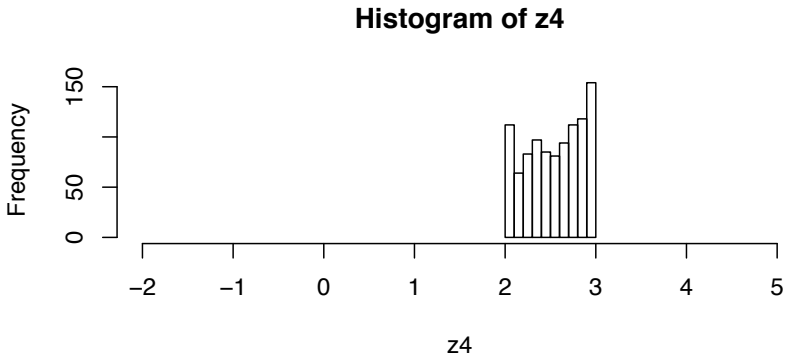
SD = 0.1 gives unsatisfactory results



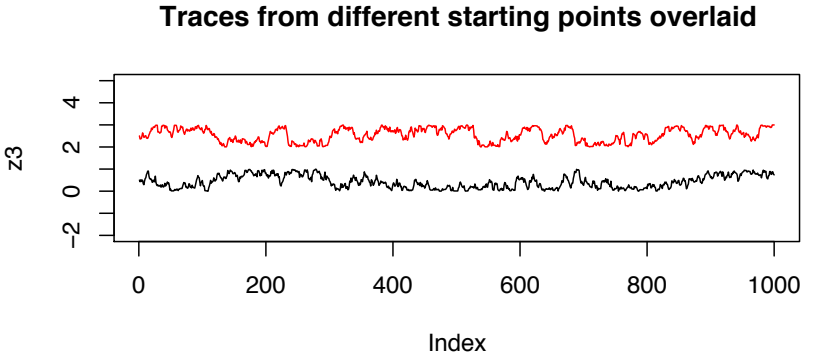


SD = 0.1: different starting points disagree

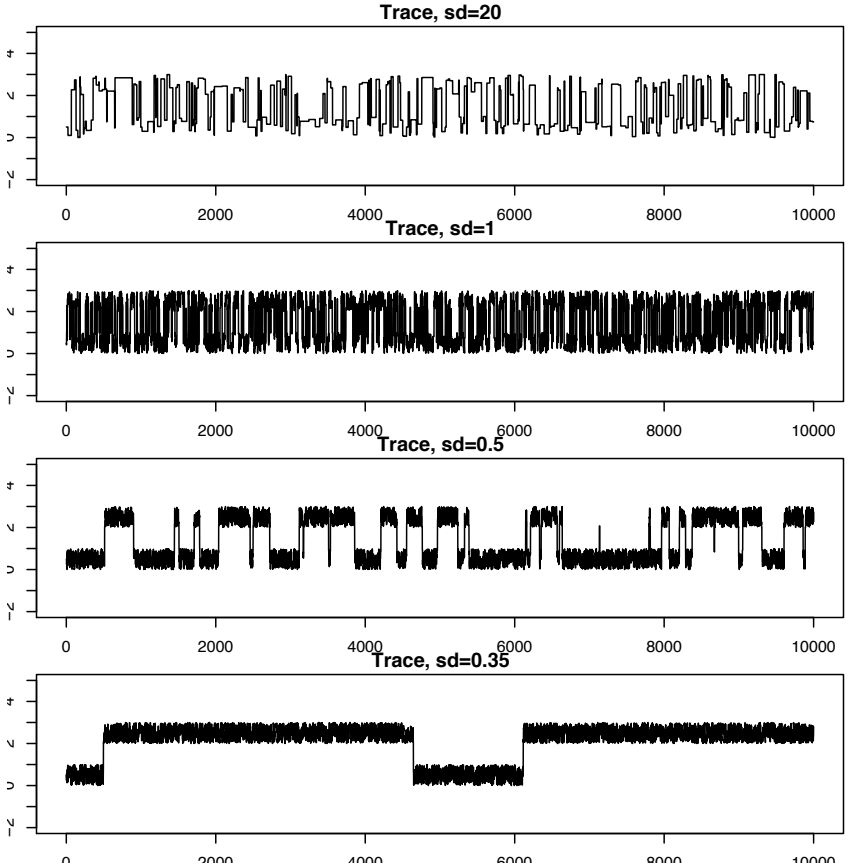




SD = 0.1: different starting points disagree



“Stickiness” of chain depends on SD



```
# Example 2: Estimating an allele frequency
#
# A standard assumption when modelling genotypes of
# bi-allelic loci (eg loci with alleles A and a) is
# that the population is "randomly mating". From this assumption
# it follows that the population will be in
# "Hardy Weinberg Equilibrium" (HWE), which means that
# if p is the frequency of the allele A then the
# genotypes AA, Aa and aa will have frequencies
# p*p, 2p(1-p) and (1-p)*(1-p)
#
# A simple prior for p is to assume it is uniform on [0,1].
# Suppose that we sample n individuals, and observe nAA with genotype AA,
# nAa with genotype Aa and naa with genotype aa.

# The following R code gives a short MCMC routine to sample from the posterior
# distribution of p. Try to go through the code to see how it works.

prior = function(p){
if((p<0) || (p>1)){  # || here means "or"
return(0)}
else{
return(1)}
}
```

```
likelihood = function(p, nAA, nAa, naa){
return(p^(2*nAA) * (2*p*(1-p))^nAa * (1-p)^(2*naa))
}

psampler = function(nAA, nAa, naa, niter, pstartval, pproposalsd){
p = rep(0,niter)
p[1] = pstartval
for(i in 2:niter){
currentp = p[i-1]
newp = currentp + rnorm(1,0,pproposalsd)
A = prior(newp)*likelihood(newp,nAA,nAa,naa)/(prior(currentp) * likelihood(currentp,nA
if(runif(1)<A){
p[i] = newp       # accept move with probabily min(1,A)
} else {
p[i] = currentp        # otherwise "reject" move, and stay where we are
}
}
return(p)
}

# running this sample for nAA = 50, nAa = 21, naa=29.

z=psampler(50,21,29,10000,0.5,0.01)
```

```
# Now some R code to compare the sample from the posterior
# with the theoretical posterior (which in this case is available analytically;
# from lecture 1, since we observed 121 As, and 79 as, out of 200, the posterior
# for p is beta(121+1,79+1). See notes from Session 1.

x=seq(0,1,length=1000)
hist(z,prob=T)
lines(x,dbeta(x,122, 80))  # overlays beta density on histogram

# You might also like to discard the first 5000 z's as "burnin"
# here's one way in R to select only the last 5000 zs

hist(z[5001:10000])

#
#
# Some things to try: how do the starting point and proposal sd affect things?
#
#
```

```
# Example 3: Estimating an allele frequency and inbreeding coefficient
# (If time allows!)
#
# A slightly more complex alternative than HWE is to assume that
# there is a tendency for people to mate with others who are
# slightly more closely-related than "random" (as might
# happen in a geographically-structured population, for example).
# This will result in an excess of homozygotes compared with
# HWE. A simple way to capture this is to introduce an extra parameter,
# the "inbreeding coefficient" f, and assume that the genotypes
# AA, Aa and aa have frequencies fp + (1-f)p*p, (1-f) 2p(1-p)
# and f(1-p) + (1-f)(1-p)(1-p)
#
# In most cases it would be natural to treat f as a feature of
# the population, and therefore assume f is constant across loci.
# For simplicity we will consider just a single locus.

# Note that both f and p are constrained to lie between 0 and 1
# (inclusive). A simple prior for each of these two parameters is to assume
# that they are independent, uniform on [0,1]. Suppose
# that we sample n individuals, and observe nAA with genotype AA,
# nAa with genotype Aa and naa with genotype aa.
#
# Exercise: write a short MCMC routine to sample from the joint
# distribution of f and p.
```

```
#
# Hint: here is a start; you'll need to fill in the ...
#
fpsampler = function(nAA, nAa, naa, niter, fstartval, pstartval,
                     fproposalsd, pproposalsd){
f = rep(0,niter)
p = rep(0,niter)
f[1] = fstartval
p[1] = pstartval
for(i in 2:niter){
currentf = f[i-1]
currentp = p[i-1]
newf = ...
newp = ...
...
...
}
return(list(f=f,p=p)) # return a "list" with two elements named f and p
}
```

Markov Chain Monte Carlo: more complex examples

Goals of this lecture:

- Consider MCMC in more than a single dimension
- Describe component-wise Metropolis-Hastings sampling
- Introduce the idea of latent variables and show how they lend themselves to Gibbs sampling—a special case of component-wise M-H sampling

Genotype Frequencies and Inbreeding:

Starting with the last exercise of Session 3: one locus with two alleles, A and a, at frequencies p and $1 - p$, respectively, and "inbreeding coefficient" f.

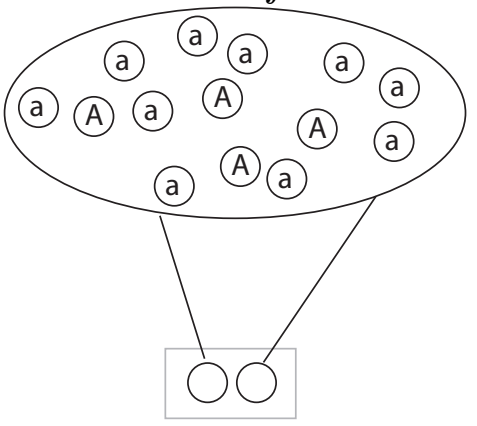
Probabilities of the three genotypes are:

- $P(AA) = fp + (1 - f)p^2$
- $P(Aa \text{ or } aA) = (1 - f)2p(1 - p)$
- $P(aa) = f(1 - p) + (1 - f)(1 - p)^2$

Since "inbreeding" is used to describe a lot of (related) things, let's briefly review what this model is saying...

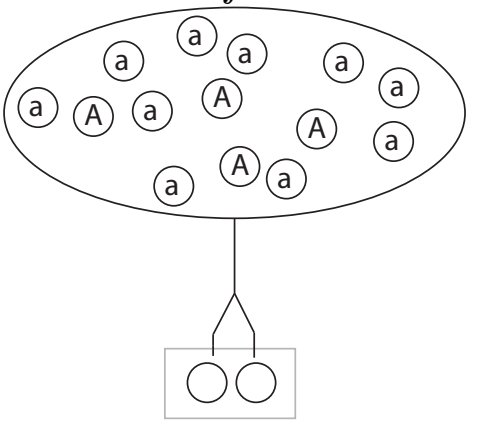
Inbreeding model:

Not-inbred with probability $1 - f$



- $P(AA) = p^2$
- $P(Aa \text{ or } aA) = 2p(1-p)$
- $P(aa) = (1-p)^2$

Inbred with probability f



- $P(AA) = p$
- $P(Aa \text{ or } aA) = 0$
- $P(aa) = (1-p)$

$$P(n_{AA}, n_{Aa}, n_{aa}|p, f) = C \times [fp + (1-f)p^2]^{n_{AA}} \times [(1-f)2p(1-p)]^{n_{Aa}} \times [f(1-p) + (1-f)(1-p)^2]^{n_{aa}}$$

What Our Data Would Look Like:

- We have a sample of n individuals total.
  - n_{AA} are homozygous for the A allele
  - n_{Aa} are heterorozygous
  - n_{aa} are homozygous for the a allele
- Clearly, $n = n_{AA} + n_{Aa} + n_{aa}$

A concrete example we will use throughout this lecture is $n = 50$ with:

$$n_{AA} = 30 \qquad n_{Aa} = 10 \qquad n_{aa} = 10$$

which are roughly the expected values if $p = .7$ and $f = .5$.

Bayesian Model for Estimating f and p:

To go about estimating f and p in a Bayesian fashion, we must fully specify the model. This means that we need

- Priors for f and p. We will choose uniform priors $f \sim \texttt{Beta}(1,1)$ and $p \sim \texttt{Beta}(1,1)$
- The data themselves. These are n_{AA}, n_{Aa}, and n_{aa}.
- The likelihood: $P(n_{AA}, n_{Aa}, n_{aa}|p, f)$, which is given on the previous page

The posterior distribution is then prior $\times$ likelihood, divided by the ("nasty") normalizing constant

$$P(p, f|n_{AA}, n_{Aa}, n_{aa}) = \frac{P(f)P(p)P(n_{AA}, n_{Aa}, n_{aa}|p, f)}{\int_{f,p} P(f)P(p)P(n_{AA}, n_{Aa}, n_{aa}|p, f)df\,dp}$$

Computing the normalizing constant is difficult, but with MCMC, *we don't have to!*

Recall, to perform MCMC it suffices to know the target distribution up to a constant of proportionality.

Our target distribution is

$$P(p, f|n_{AA}, n_{Aa}, n_{aa}) \propto P(f)P(p)P(n_{AA}, n_{Aa}, n_{aa}|p, f)$$

So long as $0 < f < 1$ and $0 < p < 1$.

Otherwise it is 0.

Two-Dimensional M-H Sampler:

We can compute the target distribution (up to a constant) easily for any f and p. So, to simulate from this posterior distribution we can just implement a Metropolis-Hastings sampler.

Following the examples of Session 3, for p we will choose a normal proposal distribution centered on the current value with standard deviation of s_p:

$$q(p^*|p) \equiv \text{Normal}(p, s_p)$$

And, we will use the same for f:

$$q(f^*|f) \equiv \text{Normal}(f, s_f)$$

Applying these proposal distributions in sequence gives us a simple way to simulate proposed values, (p^*, f^*), from the current values (p, f).

A "sweep" of our MCMC algorithm would look like:

1. Propose a new value, (p^*, f^*) for (p, f)
   - propose p^* from $\text{Normal}(p, s_p)$
   - propose f^* from $\text{Normal}(f, s_f)$
2. Accept or reject the proposed value (p^*, f^*) with probability R which is the minimum of 1 and the Hasting's Ratio:

$$R = \min\left\{1, \frac{q(p|p^*)q(f|f^*)}{q(p^*|p)q(f^*|f)} \times \frac{P(p^*)P(f^*)P(n_{AA}, n_{Aa}, n_{aa}|p^*, f^*)}{P(p)P(f)P(n_{AA}, n_{Aa}, n_{aa}|p, f)}\right\}$$

If you accept the proposed value, set the current value to the proposed value. Otherwise leave the current values unchanged.

Computer Demo: `inbred_p -n 30 10 10` (starts on "Jointly" (j))

Component-wise Metropolis Hastings Sampler :

- In any MCMC implementation, the proposal distribution *need not propose changes to* **every** *variable/parameter in the model*.
- In fact, there are few "real-world" problems requiring MCMC in which you would use a single proposal distribution in which changes were proposed to all the variables in the model.
- **Important Concept:**
  - Any proposal distribution, regardless of how many or how few variables it proposes changes to, is valid, so long as the proposal is accepted or rejected in a way that satisfies detailed balance w.r.t. the target distribution.
  - These different flavors of the "propose-reject/accept" step may be combined in series in whatever manner is desired, so long as they produce an irreducible, aperiodic chain[1].

[1] Nonetheless, some ways are better than others and will lead to a better-mixing chain

Simple Component-wise M-H Sampler for p and f:

Simplest scenario has a sweep as follows:

1. Do an update for p:
   (a) propose p^* from $\text{Normal}(p, s_p)$
   (b) Accept or reject the proposed value p^* with probability $\min\{1, \alpha\}$, where:

$$\alpha = \frac{q(p|p^*)}{q(p^*|p)} \times \frac{P(p^*)P(f)P(n_{AA}, n_{Aa}, n_{aa}|p^*, f)}{P(p)P(f)P(n_{AA}, n_{Aa}, n_{aa}|p, f)}$$

2. Do an update for f:
   (a) propose f^* from $\text{Normal}(f, s_f)$
   (b) Accept or reject the proposed value f^* with probability $\min\{1, \alpha\}$, where:

$$\alpha = \frac{q(f|f^*)}{q(f^*|f)} \times \frac{P(p)P(f^*)P(n_{AA}, n_{Aa}, n_{aa}|p, f^*)}{P(p)P(f)P(n_{AA}, n_{Aa}, n_{aa}|p, f)}$$

Simple Component-wise M-H Sampler for p and f, cont'd:

Some very important points:

- The proposals are each a little simpler (though just slightly...) than jointly proposing changes to (p, f)
- Neither step 1 nor step 2 of the sweep creates an irreducible chain (obviously, if you never update p, for example, your chain could never reach every possible value of p).
- However, taken together, steps 1 and 2 create an irreducible chain.

Computer Demo: `inbred_p` (Component-wise using (c) from info window)

Simple Component-wise M-H Sampler for p and f, cont'd:

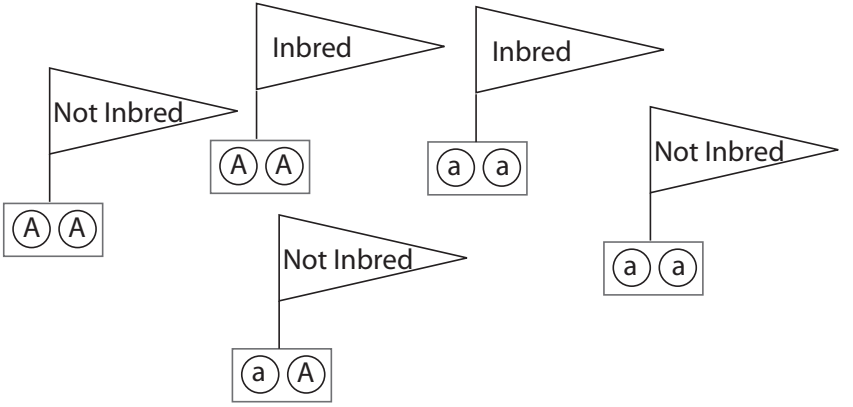
Two more important points

- The form of the Hastings ratio is a little simpler when we have proposed changing just a subset of the variables.
- However, in this case the target density remains just as complex, because it does not factorize into a separate part for f and a part for p.
- Since we are changing just a small part of the model at a time, it seems like we could spend some more energy on making each separate proposal distribution more "intelligent."

The final two points above get us to thinking about Gibbs sampling, which we will return to after a brief discussion of latent variables...

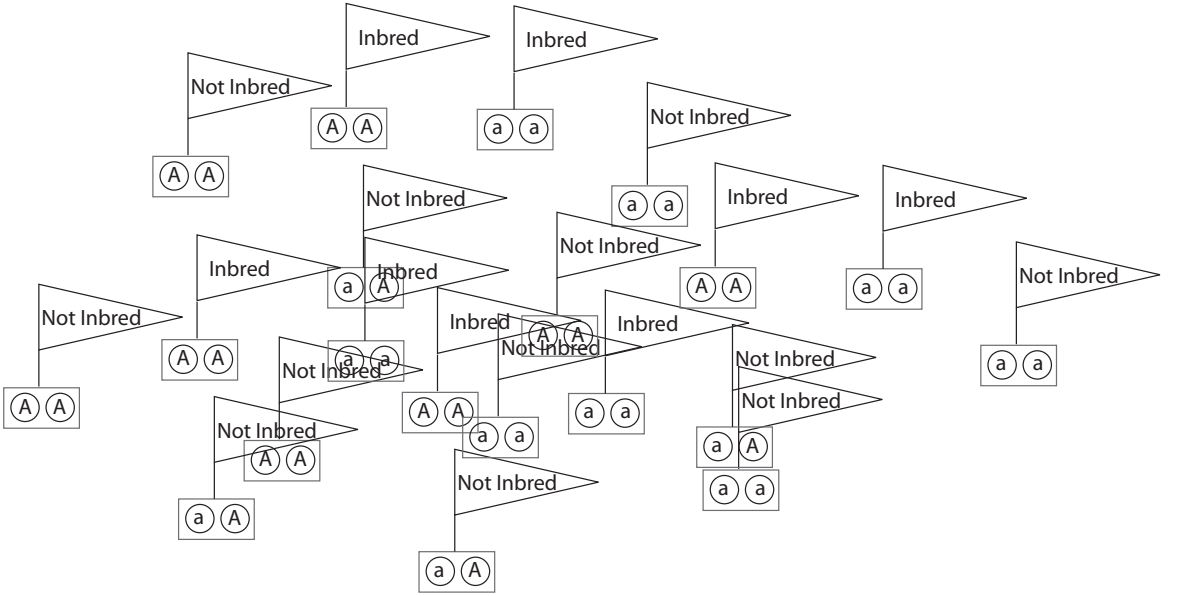
Formulating the Model with Latent Variables:

Just think how easy it would be to estimate f and p if we knew whether every individual we sampled was inbred or not:



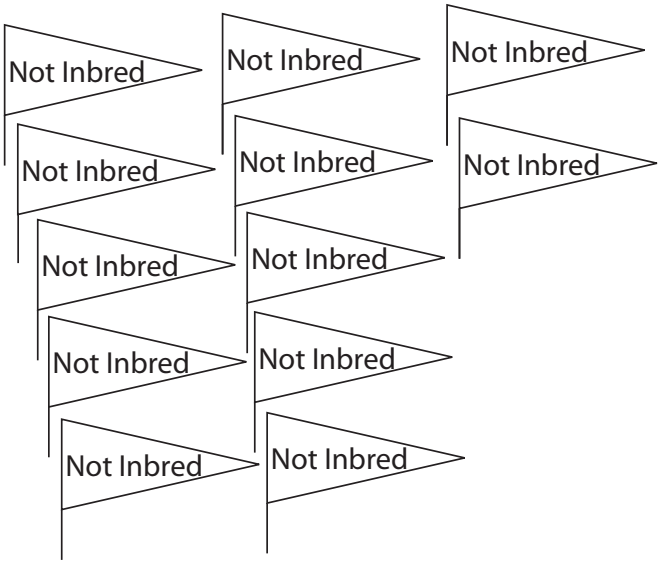
- Now, f is just a binomial proportion
- To estimate p you count both the alleles from Non-inbred individuals, and just one allele from Inbred individuals, and then it too is simply a binomial proportion.

In other words, if your data look like:

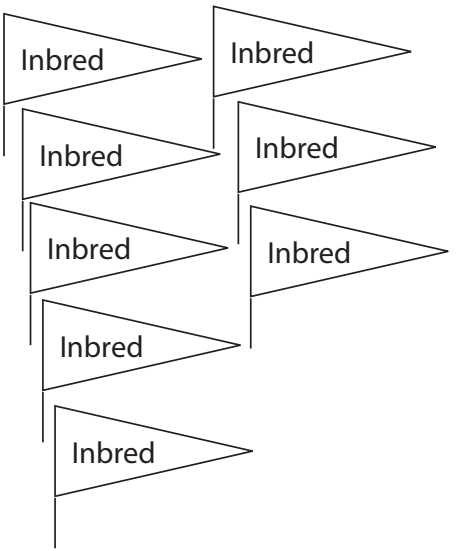


. . . then, to estimate f, you don't even need to think about the alleles carried by anyone. . .

. . . you just have to “count the flags”:

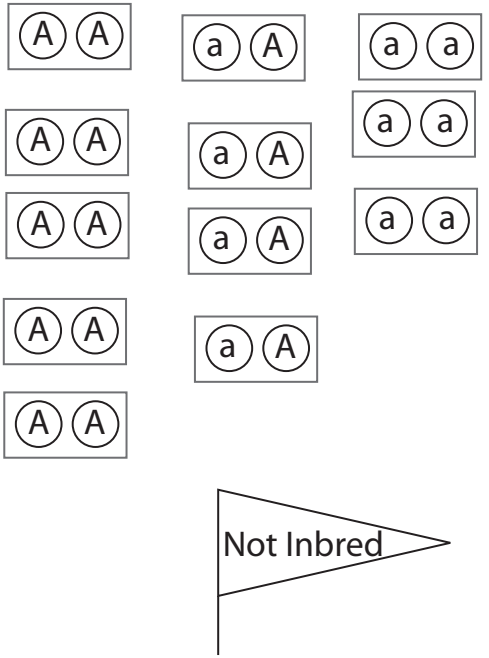


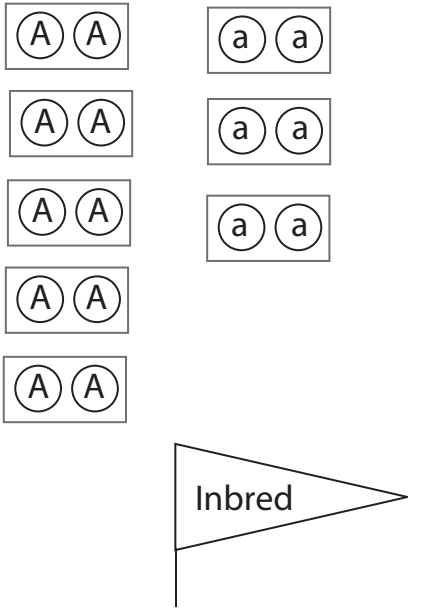
12



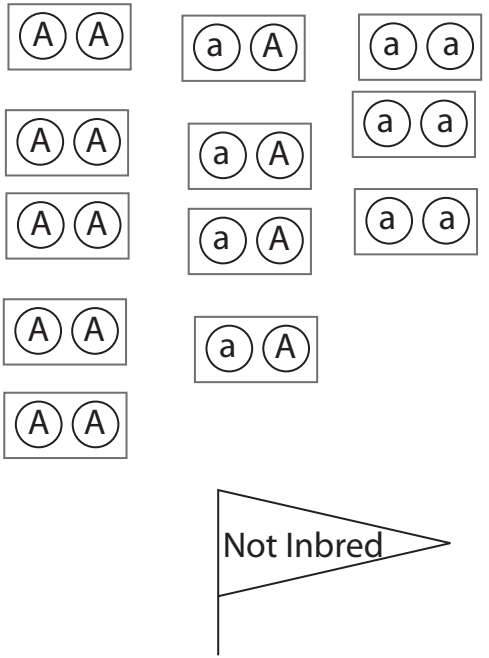
8

To estimate p you just have to count alleles. . .

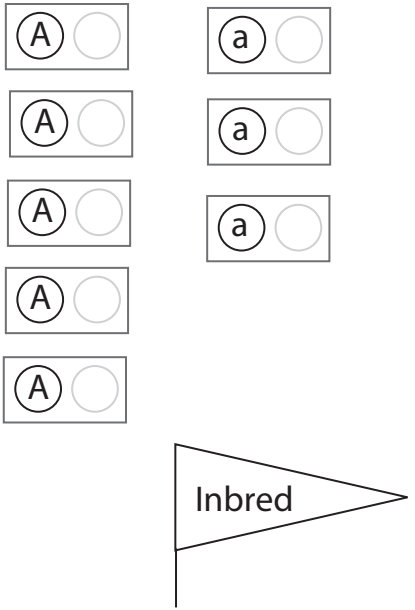




Keeping in mind each inbred individual contributes just a single gene copy. . .



14 A's 10 a's



5 A's 3 a's

Joint Density of p, f, and U:

- We denote the n latent variables as $\boldsymbol{U} = (U_1, \ldots, U_n)$.
- $u_i = 0$ indicates that i is Not Inbred. $u_i = 1$ indicates i is Inbred.
- Let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ denote the genotypes

So, the joint density is:

$$P(p, f, \boldsymbol{U}, \boldsymbol{Y}) = P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U}, p)$$

and, so, the posterior is:

$$P(p, f, \boldsymbol{U}|\boldsymbol{Y}) = \frac{P(p, f, \boldsymbol{U}, \boldsymbol{Y})}{\int_p \int_f \sum_{0 \leq U_1 \leq 1} \cdots \sum_{0 \leq U_n \leq 1} P(p, f, \boldsymbol{U}, \boldsymbol{Y}) dp df}$$

The normalizing constant is even nastier! It seems we have gained little, until we remember that we don't have to mess with the normalizing constant, ... but even then it's not immediately clear we have gained anything.

Generic MCMC for f and p with latent variables:

- Recall, we wish to simulate f and p from their posterior distributions and so learn about $P(f, p|\boldsymbol{Y})$.
- However, now, our chain is moving about the space of f, p, *and* $\boldsymbol{U}\ldots$
  - it visits a sequence of states of the form $(f^{(1)}, p^{(1)}, \boldsymbol{U}^{(1)})$, $(f^{(2)}, p^{(2)}, \boldsymbol{U}^{(2)})$, $(f^{(3)}, p^{(3)}, \boldsymbol{U}^{(3)}), \ldots$

- **Important Concept**: Obtaining the Marginal Posterior Distribution from MCMC output is simple:
  - From the sequence $(f^{(1)}, p^{(1)}, \boldsymbol{U}^{(1)})$, $(f^{(2)}, p^{(2)}, \boldsymbol{U}^{(2)}), \ldots$ you can just focus on collecting the values $(f^{(1)}, p^{(1)})$, $(f^{(2)}), \ldots$, discarding the $\boldsymbol{U}^{(\cdot)}$'s if you are not interested in their posterior distribution. (Note, though, that you might be interested in the individual U_i's!).

Naive Metropolis-Hastings for f, p, and U:

1. Do an update for p:
   (a) propose p^* from $\text{Normal}(p, s_p)$
   (b) accept or reject on the basis of

$$\frac{q(p|p^*)}{q(p^*|p)} \times \frac{P(p^*)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p^*)}{P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p)}$$

2. Do an update for f:
   (a) propose f^* from $\text{Normal}(f, s_f)$
   (b) accept or reject on the basis of

$$\frac{q(f|f^*)}{q(f^*|f)} \times \frac{P(p)P(f^*)P(\boldsymbol{U}|f^*)P(\boldsymbol{Y}|\boldsymbol{U},p)}{P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p)}$$

3. Do an update for $\boldsymbol{U}$ (n separate updates, one for each U_i).
   For i in $1, \ldots, n$:
   (a) Propose a new U_i^* by flipping a coin (heads=0, tails=1).
   (b) Accept or reject on the basis of

$$\frac{q(U_i|U_i^*)}{q(U_i^*|U_i)} \times \frac{P(p)P(f)P(\boldsymbol{U}^*|f)P(\boldsymbol{Y}|\boldsymbol{U}^*,p)}{P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p)}$$

$$= \frac{q(U_i|U_i^*)}{q(U_i^*|U_i)} \times \frac{P(p)P(f)P(U_i^*|f)P(\boldsymbol{Y}|U_i^*,p)}{P(p)P(f)P(U_i|f)P(\boldsymbol{Y}|U_i,p)}$$

Note the cancellations in all these Hastings Ratios.

Computer Demo: `inbred_p` ("Silly" using (S), (f), and (p)) [2]

[2] which will segue into Gibbs sampling

Gibbs Sampling:

Gibbs sampling is a special case of the component-wise M-H sampler in which the proposal distribution is the *full conditional distribution*.

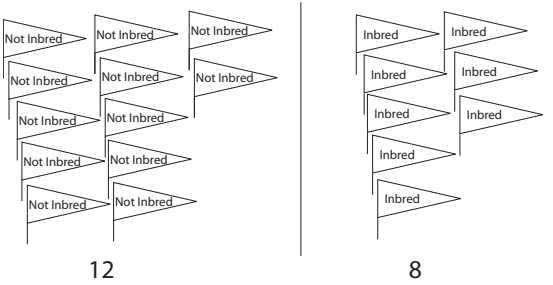
Gibbs Sampling Step for f:

- Recall the generic component-wise M-H update:

$$\frac{q(f|f^*)}{q(f^*|f)} \times \frac{P(p)P(f^*)P(\boldsymbol{U}|f^*)P(\boldsymbol{Y}|\boldsymbol{U},p)}{P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p)}$$

  The full conditional distribution for f is the distribution of f conditional upon the current values of all other variables in the model. This must be proportional to the product of all the factors in the joint density that have f in them: $P(f)P(\boldsymbol{U}|f)$

- The $P(\boldsymbol{U}|f)$ portion of the joint density pertains to "counting up our flags"

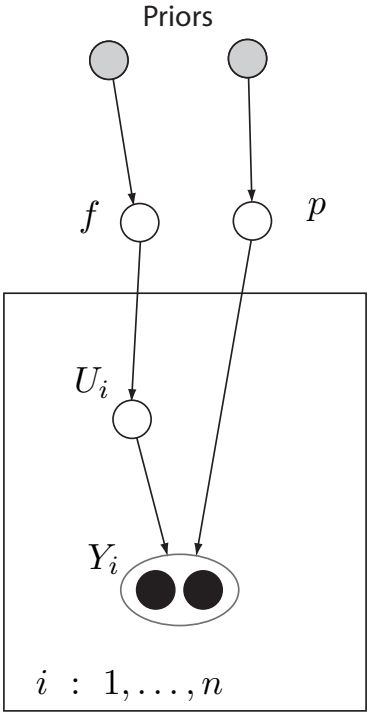


- $P(f)$ is the prior for f
- So the full conditional for f is a beta distribution—it's the "posterior" for f given the "data" $\boldsymbol{U}$
- Letting the proposal $q(\cdot|\cdot)$ be that full conditional gives us a Hastings Ratio of

$$\frac{P(f)P(\boldsymbol{U}|f)/C}{P(f^*)P(\boldsymbol{U}|f^*)/C} \times \frac{P(p)P(f^*)P(\boldsymbol{U}|f^*)P(\boldsymbol{Y}|\boldsymbol{U},p)}{P(p)P(f)P(\boldsymbol{U}|f)P(\boldsymbol{Y}|\boldsymbol{U},p)}$$

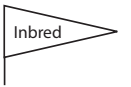
- Which is 1—This is always the case with Gibbs sampling—you always accept the proposal from the full conditional distribution.

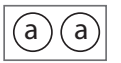
A Directed Graphical View of The Model. . . :



Values You Might See

$f = 0.2$ $p = 0.6$

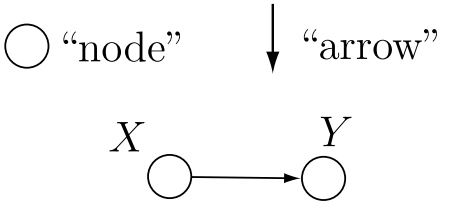




. . . makes it fairly easy to envision how Gibbs updates for p and $\boldsymbol{U}$ would proceed.

Computer Demo: `inbred_p` ("Gibbs" using (g))

DAG notation and terminology:



X is a parent of Y

Y is a daughter of X

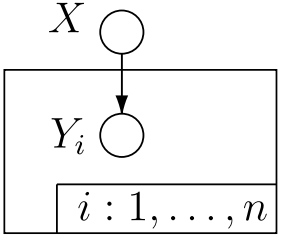
X is an unobserved variable

X is an observed variable

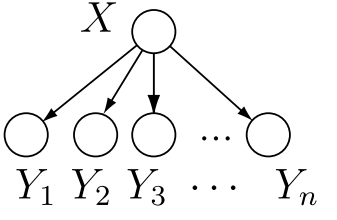
X is a variable with an assumed value (for a prior)

---

n-plates:



a shorthand for:



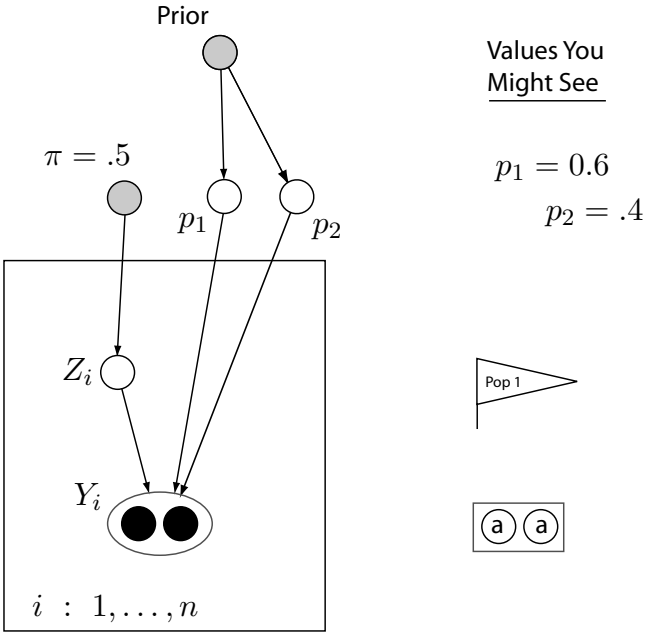
There are n variables Y_i, each conditionally independent given X.

Factorization:

$$P(\text{all}) = \prod_{\text{all nodes } i} P(\text{node } i | \text{parents of node } i)$$

A reminder that all inference is conditional upon the underlying model. . . and a small step toward *structure*:

We could have attributed the departure from Hardy-Weinberg proportions to a mixture (in the proportions of $\pi = .5$) of two populations with allele frequencies p_1 and $p2$, respectively.



This is, in fact, the *structure* model with no admixture, $K = 2$, and a single locus.

Gibbs sampling is straightforward. . . Computer Demo: `newhybs` via `RunNewHybs.sh`

Wrap-Up:

Main Points:

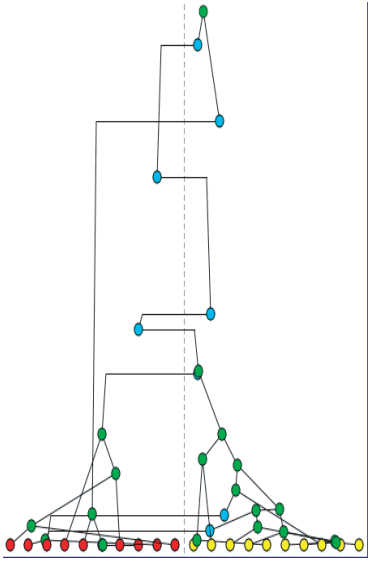
- In problems where it is useful, MCMC almost always proceeds by proposing changes to a small subset of the variables.
- There may be many different proposal types.
  - Each proposal type *must* satisfy detailed balance.
  - Each proposal type need not make an irreducible chain, BUT
  - All proposals taken together should form an irreducible chain.
- Latent variables may help in factorizing complex joint densities and lend themselves to Gibbs sampling.
- Simple proposals may be fast to implement, but may not lead to good mixing of the chain.
- Designing good proposal distributions is where one gets to perform "art" in implementing MCMC.

Case Study I: Inference of Population Structure

Goals of this lecture:

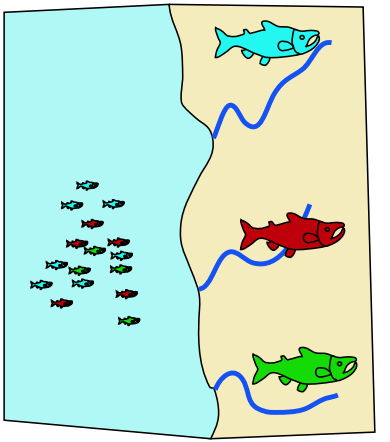
- Introduce the statistical model underlying *structure* as an elaboration of a simple finite mixture model.
- Introduce directed graphical model notation
- Present the two different "flavors" of *structure*'s underlying models—with and without prior population information.
- Introduce several related methods—*NewHybrids* and *BayesAss+*

Temporal and historical scales:



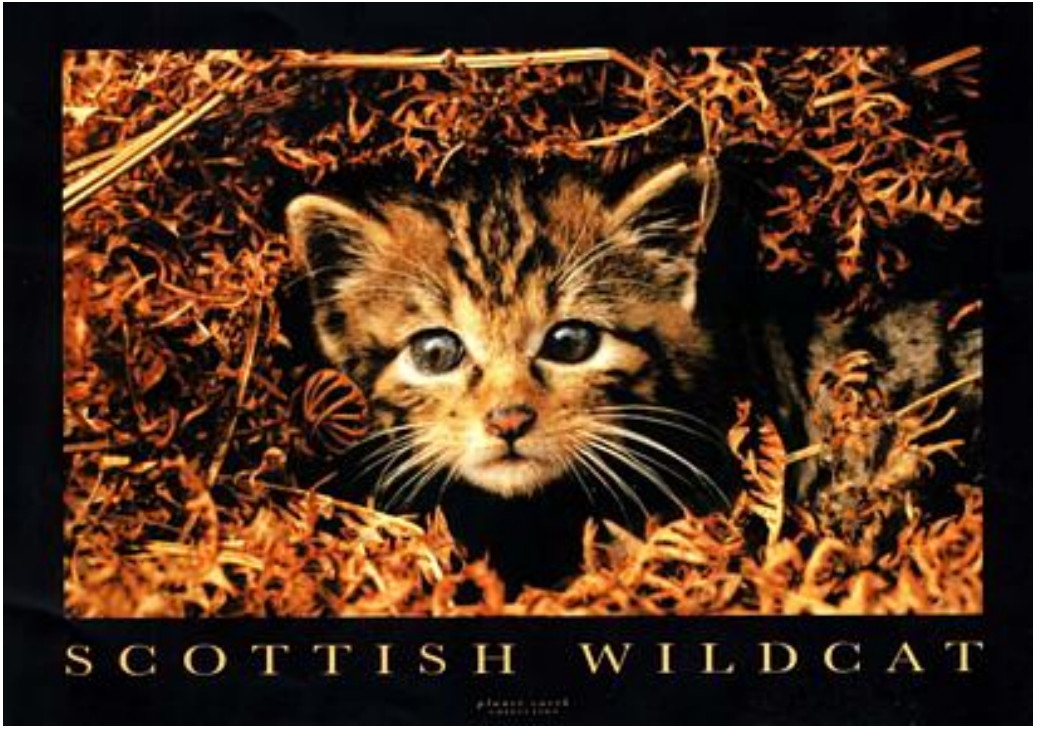
- Coalescent with migration
- Incorporates Mutation
- Non-multilocus

- Admixture Analysis
- Assumes No Mutation
- Non-multilocus

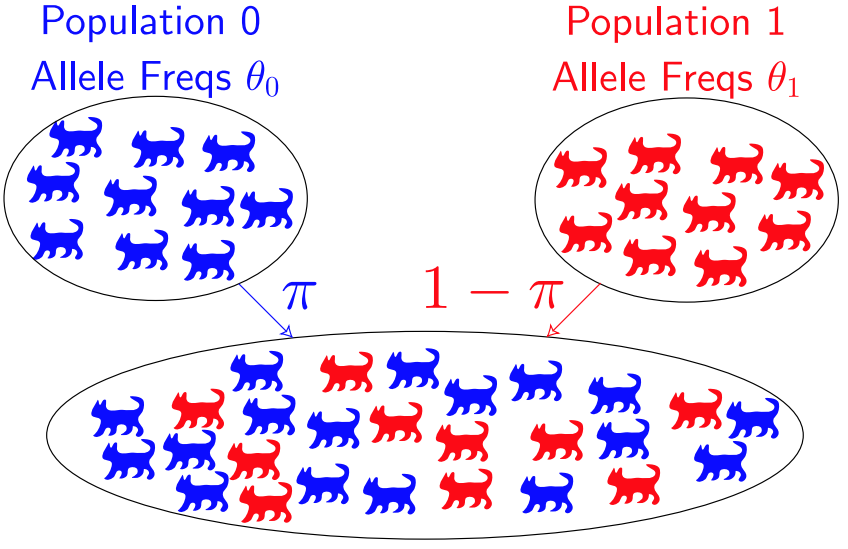


- Genetic Mixture/structure
- Assumes No Mutation
- Uses multilocus genotypes

Scottish wildcat:



Model For genetic mixture:

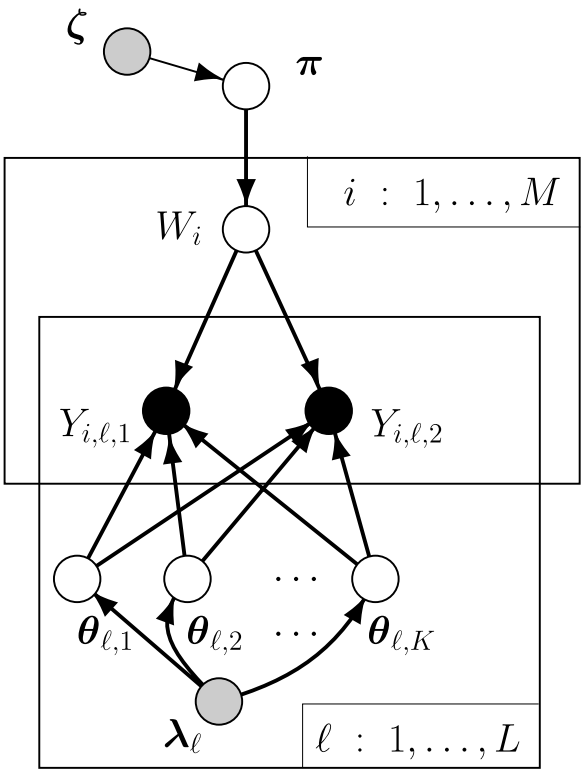


Using a sample from the mixture the goals are to:

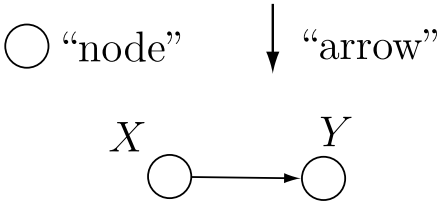
1. Estimate the allele frequencies in Populations 0 and 1
2. Estimate the mixing proportion π
3. For each individual in the sample, compute the posterior probability that it is from Population 0 or 1

Genetic mixture:

- π : the proportion of individuals from the K different sources in the mixture
- W_i : the source of individual i
- $Y_{i,\ell,1}$, $Y_{i,\ell,2}$: the two alleles at locus ℓ in individual i
- $\boldsymbol{\theta}_{\ell,1}, \ldots, \boldsymbol{\theta}_{\ell,K}$ allele frequencies at locus ℓ in the K sources
- ζ and $\boldsymbol{\lambda}_\ell$ represent Bayesian priors



DAG notation and terminology:



X is a parent of Y

Y is a daughter of X

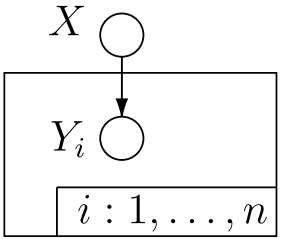
X is an unobserved variable

X is an observed variable

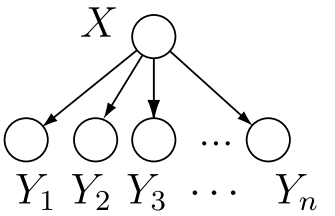
X is a variable with an assumed value (for a prior)

---

n-plates:



a shorthand for:



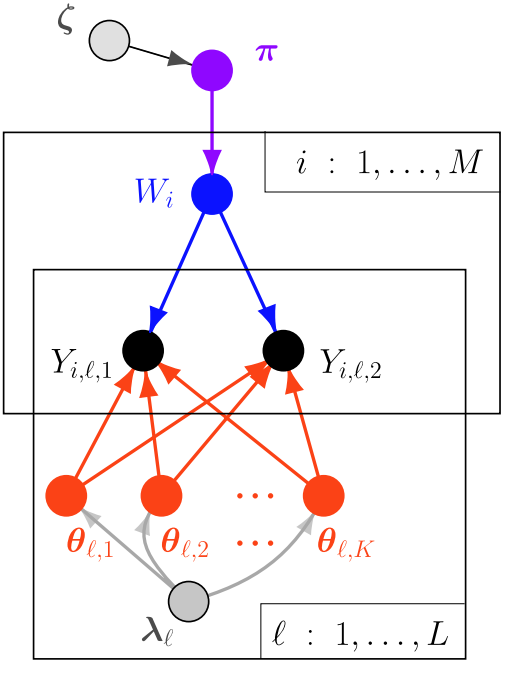
There are n variables Y_i, each conditionally independent given X.

Factorization:

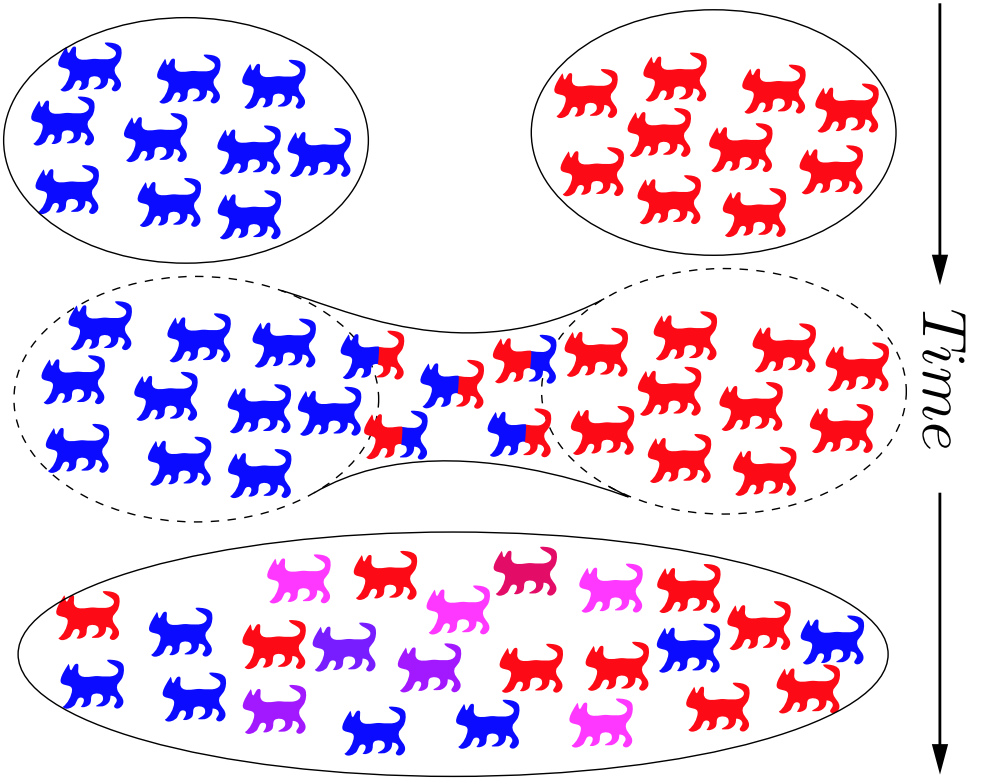
$$P(\text{all}) = \prod_{\text{all nodes } i} P(\text{node } i | \text{parents of node } i)$$

Factorization of the joint probability density:

$$
\begin{aligned}
&P(\boldsymbol{\pi}|\boldsymbol{\zeta})P(\boldsymbol{\zeta}) \times \\
&\prod_{i=1}^{M} \Bigg(P(W_i|\boldsymbol{\pi}) \times \\
&\prod_{\ell=1}^{L} P(Y_{i,\ell,1}|W_i, \boldsymbol{\theta}_{\ell,1}, \ldots, \boldsymbol{\theta}_{\ell,K}) \times \\
&\prod_{\ell=1}^{L} P(Y_{i,\ell,2}|W_i, \boldsymbol{\theta}_{\ell,1}, \ldots, \boldsymbol{\theta}_{\ell,K}) \Bigg) \times \\
&\prod_{\ell=1}^{L} P(\boldsymbol{\theta}_{\ell,1}, \ldots, \boldsymbol{\theta}_{\ell,K}|\boldsymbol{\lambda}_\ell)P(\boldsymbol{\lambda}_\ell)
\end{aligned}
$$



Interbreeding complicates matters:

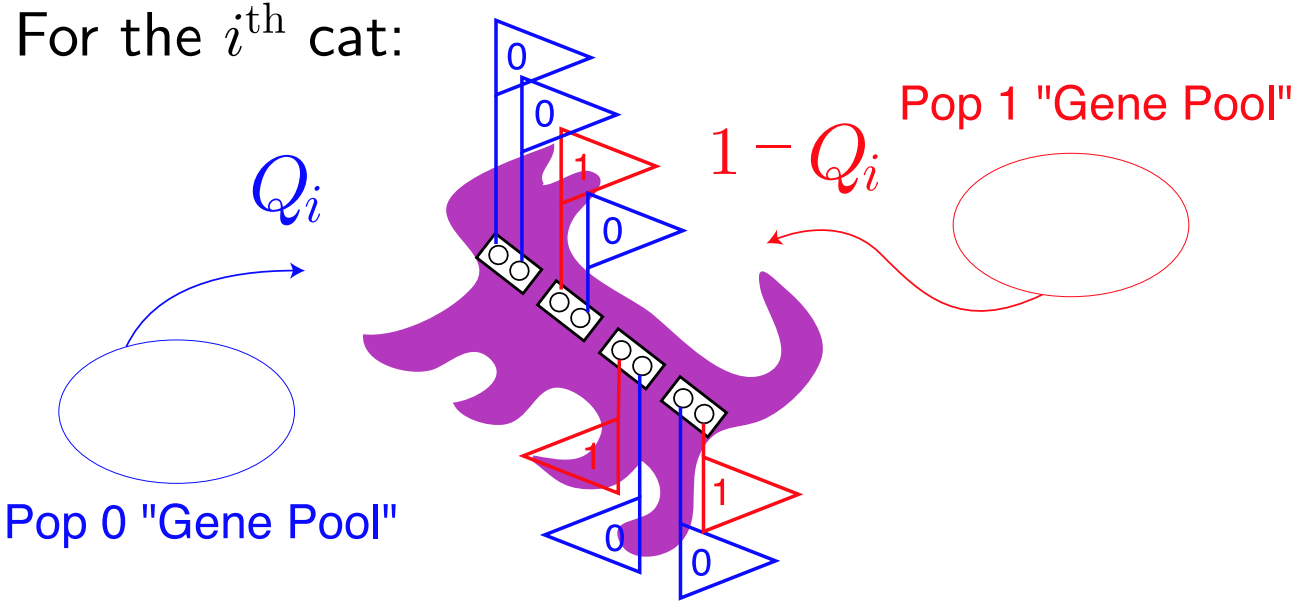


- This requires a different probability model

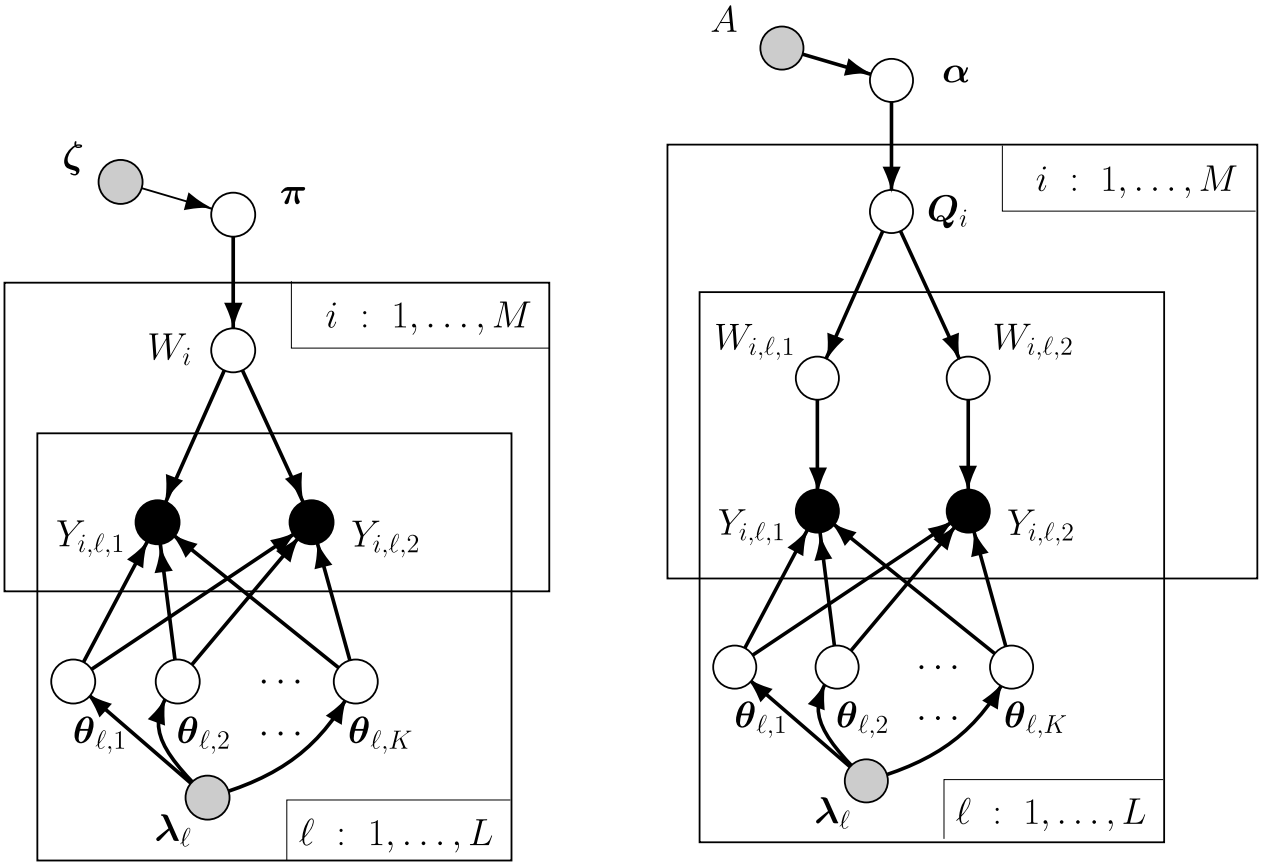
Features of *structure*—the admixture model (Pritchard et al. 2000) :

- Omnibus model—flexible and few parameters
- Applies generically to:
  - Hybrid zones
  - Recent gene flow
  - Population structure, cryptic or otherwise
  - Estimating the unkown number of subpopulations
- Data Requirement:
  - Multilocus genotypes
  - Learning samples are helpful but not necessary
- Assumes no LD or HWD in component subpopulations

The *structure* model:



Genetic mixture *vs.* *structure*[1]:



[1]Note that my notation departs somewhat from that in Pritchard et al.'s paper.

Other *structure* features:

There are several other bells and whistles that can be applied in *structure*.

A later version of structure (described in Falush et al. (2003)) includes a facility for modeling non-independence between the alleles at different loci. This allows for linkage between markers that are near to each other on the same chromosome.

The allele frequencies in the K subpopulations can also be modeled as having some correlation parameterized in terms of Wright's inbreeding parameter F. Can you write down the DAG for that model?

Further elaborations include model for dominant markers and null alleles (Falush et al. 2007) and a new model for incorporating sampling location information (Hubisz et al 2009).

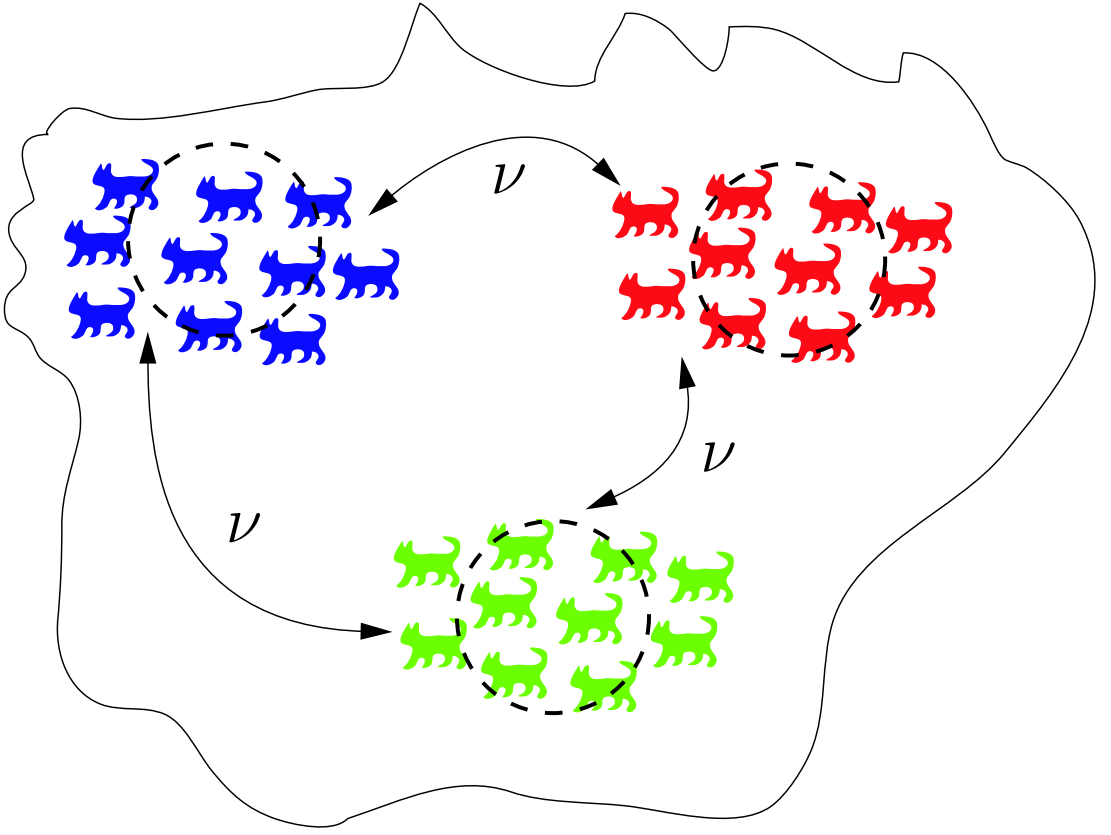
Findings of BEAUMONT ET AL. 2001:

- Evidence for two distinct gene pools
  - One inferred "gene pool" had allele frequencies very close to those in known domestic cats
- Some individuals had values of Q_i near 0 or 1
- Others appeared to be quite admixed, $Q_i \approx .5$
- Not possible to tell with *structure* whether there have been historical mixing events

Features of *structure* (prior pop. info model):

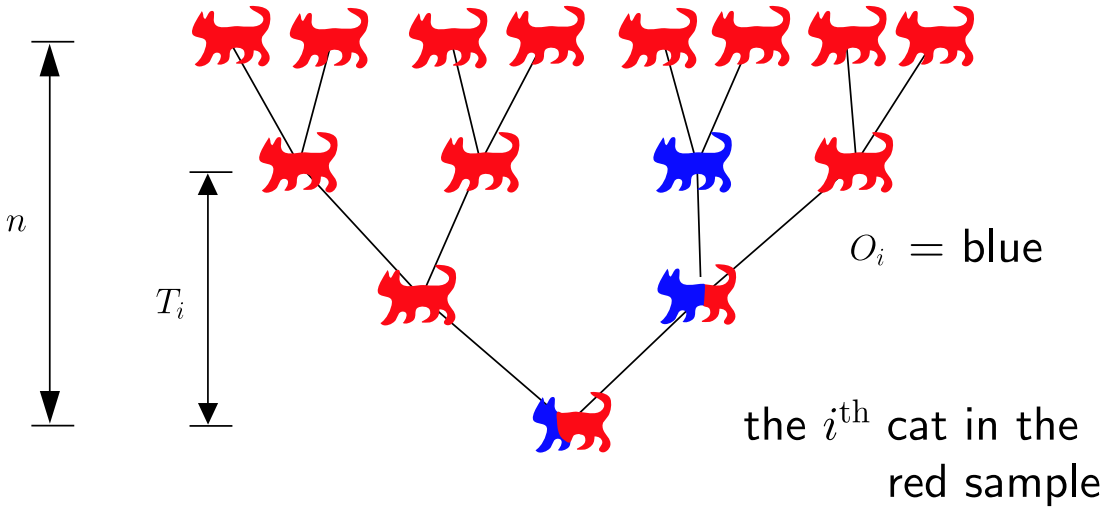
- Specialized model—more realistic in some scenarios
- Applies specifically to:
  - Detecting recent immigrants in populations
- Data Requirement:
  - Distinct subpopulation/sampling locales
  - Multilocus genotypes
- Assumptions:
  - No LD or H-WD in component subpopulations
  - Number of subpopulations is known
  - Migration occurs infrequently
  - Migration rate assumed known and equal in all directions
  - Migrant ancestry limited to the last n generations

Symmetrical immigrant probability ν:



The i^{th} individual is known to reside in subpopulation S_i.

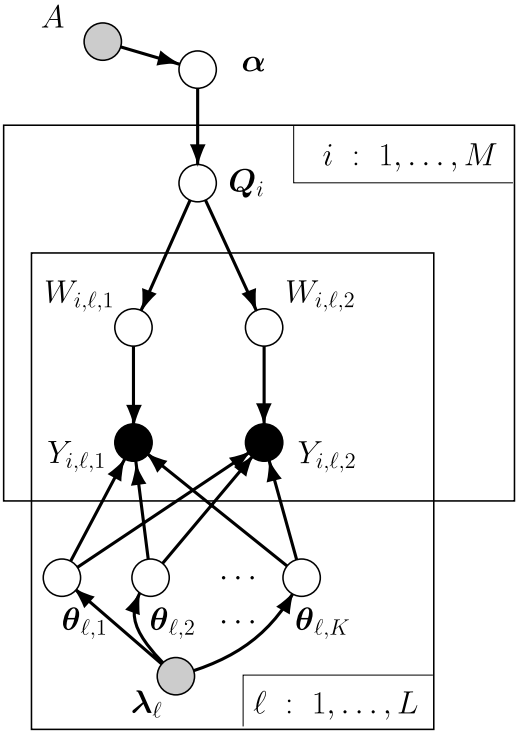
Immigrant ancestry with rare migration:

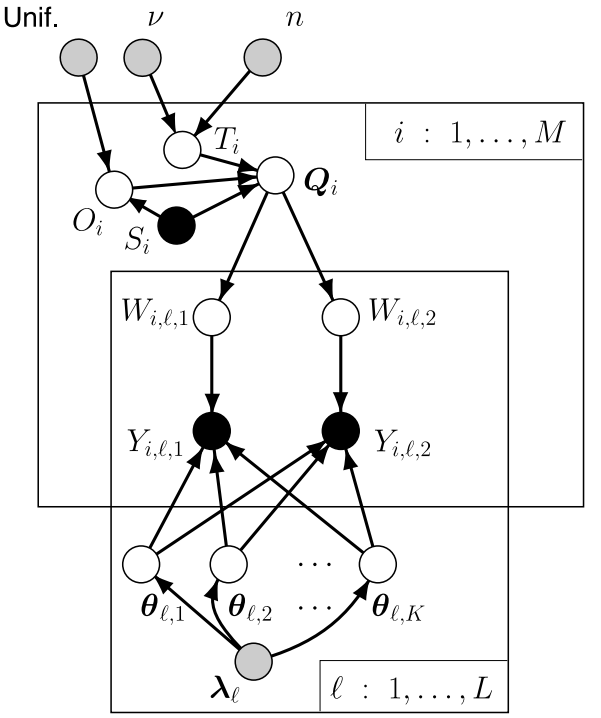


Each individual is assumed to have at most 1 migrant ancestor in the last n years.

If the i^{th} individual has a migrant ancestor, it occured T_i generations ago, and was a migrant from population O_i.

structure **vs.** *structure with prior pop. info*:





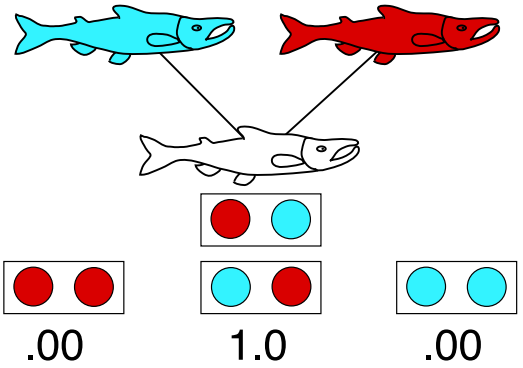
- Disadvantages: must assume ν, independence within loci, only one migrant ancestor

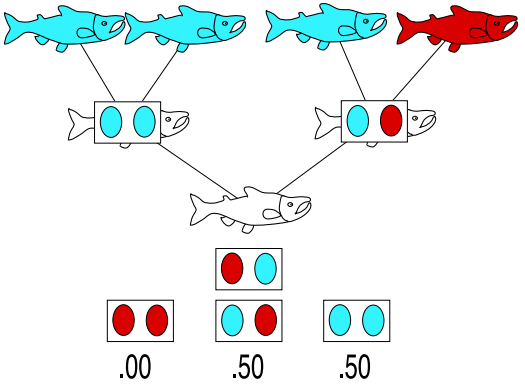
Two other methods similar to *structure* with prior population information:

- NewHybrids (Anderson and Thompson 2002)
- BayesAss+ (Wilson and Rannala 2003)

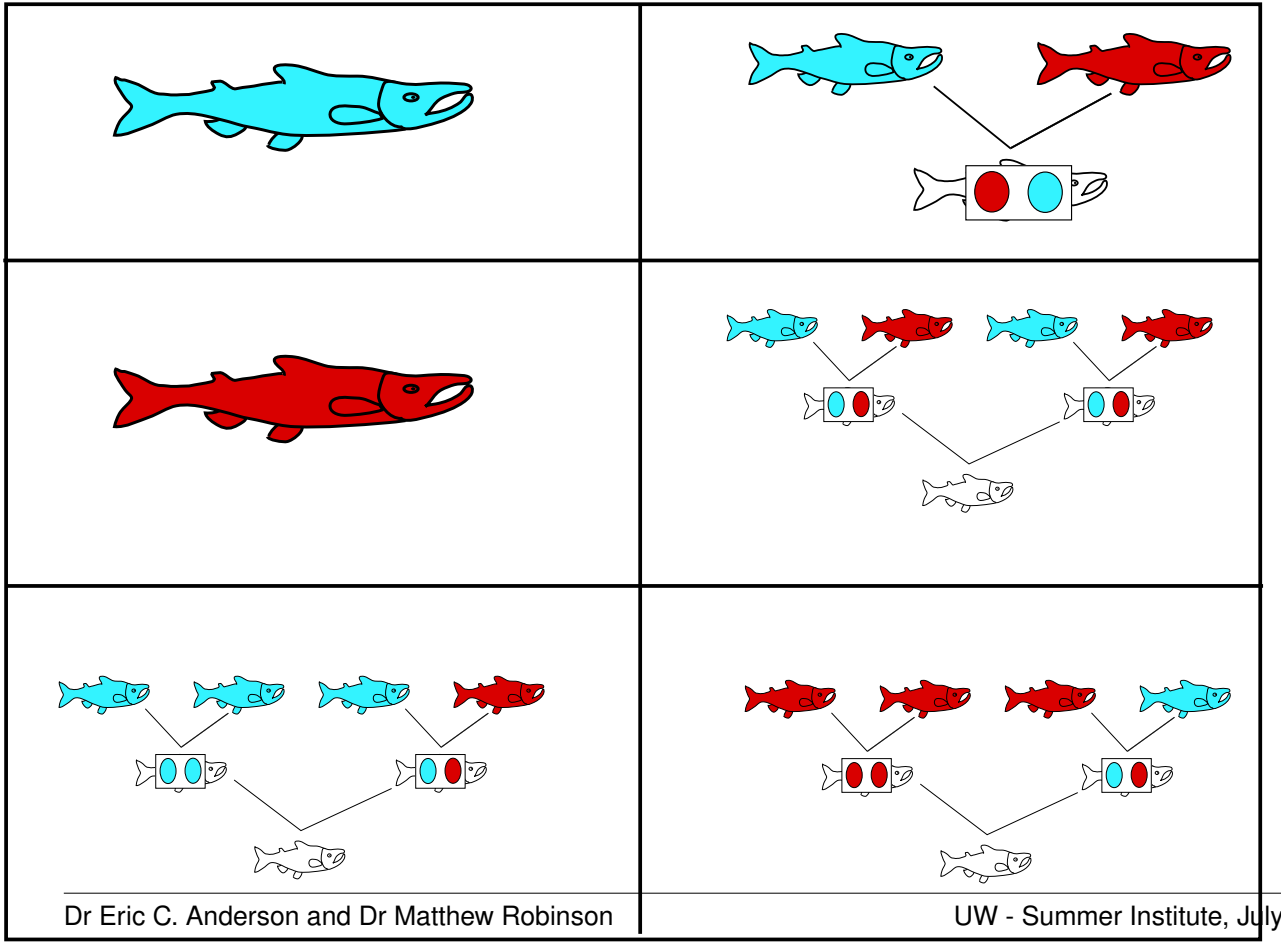
Features of *NewHybrids*:

1. Does not require sampling from known, separate locations
2. Allows more than one “migrant ancestor” (but only 2 sources)
3. Allows non-symmetrical migration/backcrossing
4. Dependence within loci is explicitly modeled

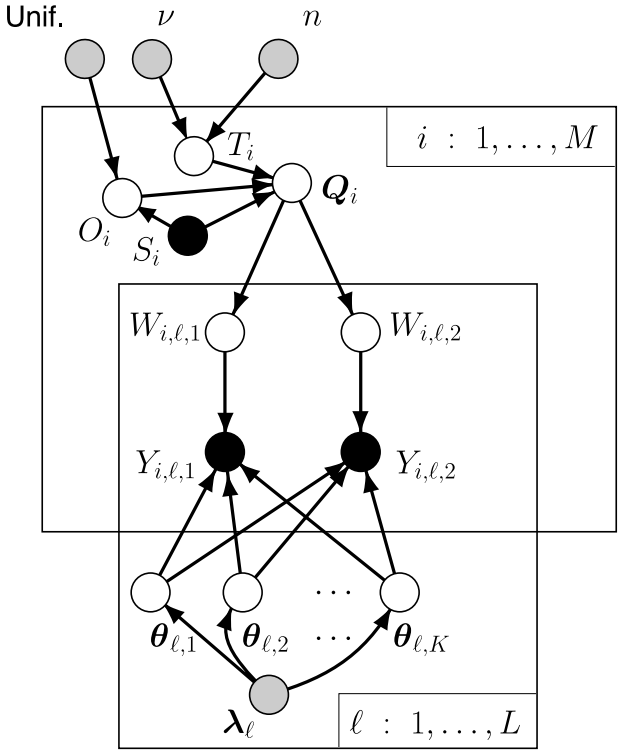


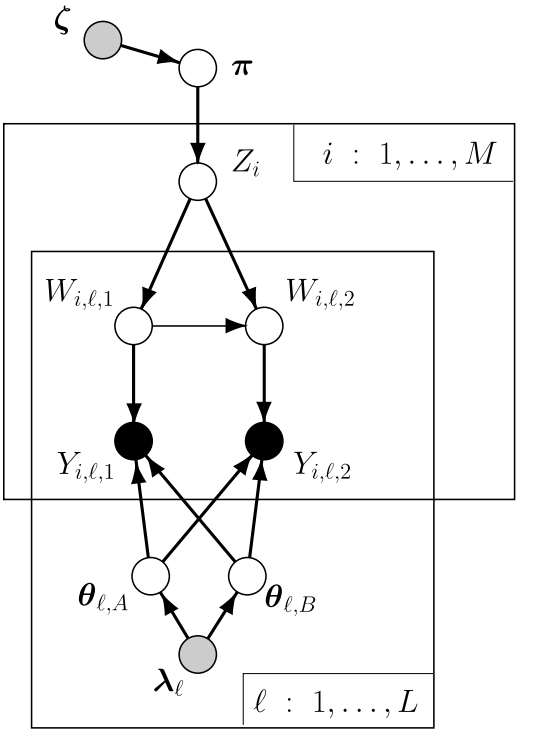


2 Generations = 6 hybrid categories:



structure with prior pop. info **vs.** *NewHybrids* **:**

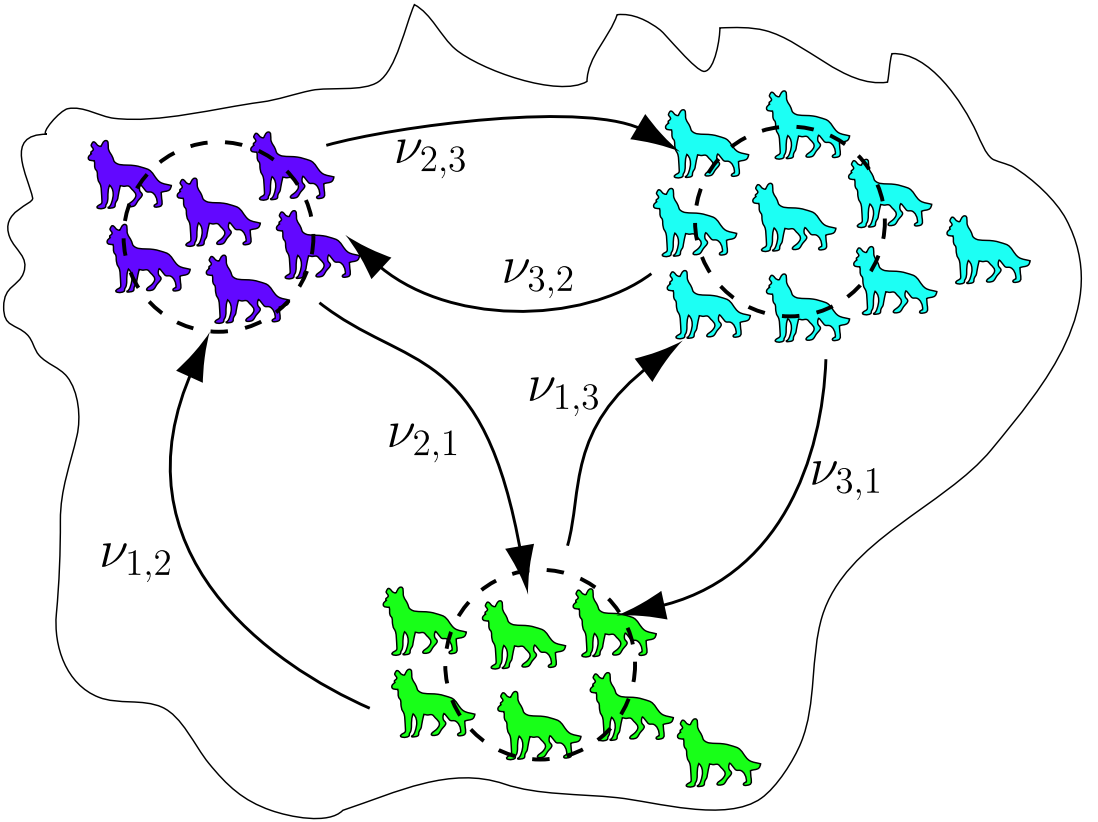




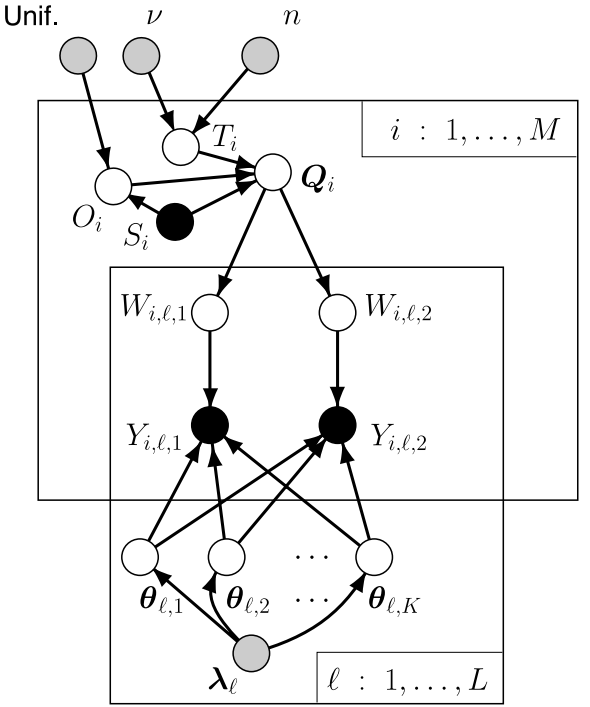
Features of *BayesAss+*:

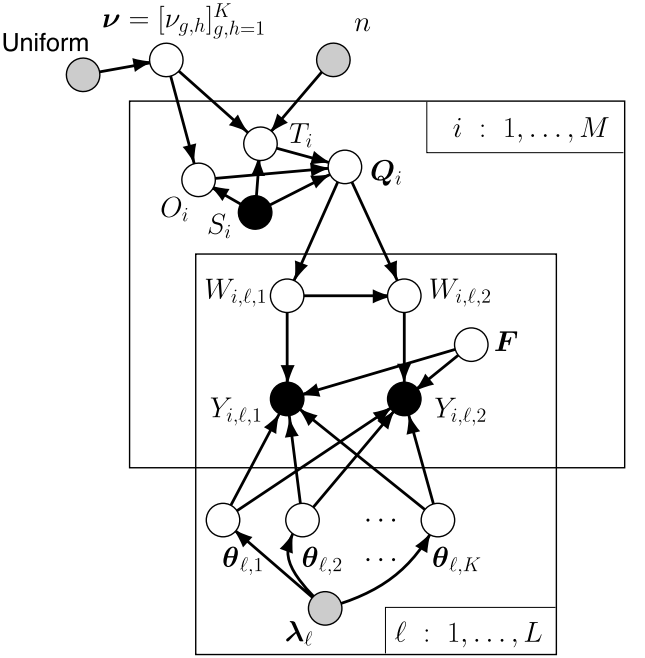
- Specialized model—more realistic in some scenarios
- Applies specifically to:
  - Detecting recent immigrants in populations
  - Estimating separate migration rates between populations
  - Can deal with multiple locales/subpopulations
- Data Requirement:
  - Distinct subpopulation/sampling locales
  - Multilocus genotypes
- Assumptions:
  - No LD in component subpopulations (F for H-WD)
  - Number of subpopulations is known
  - Migration occurs infrequently
  - Migrant ancestry limited to the last n generations

BayesAss+ sampling scenario and model:



structure with prior pop. info **vs.** *BayesAss+* **:**





Markov chain Monte Carlo in *structure*

Goals of this lecture:

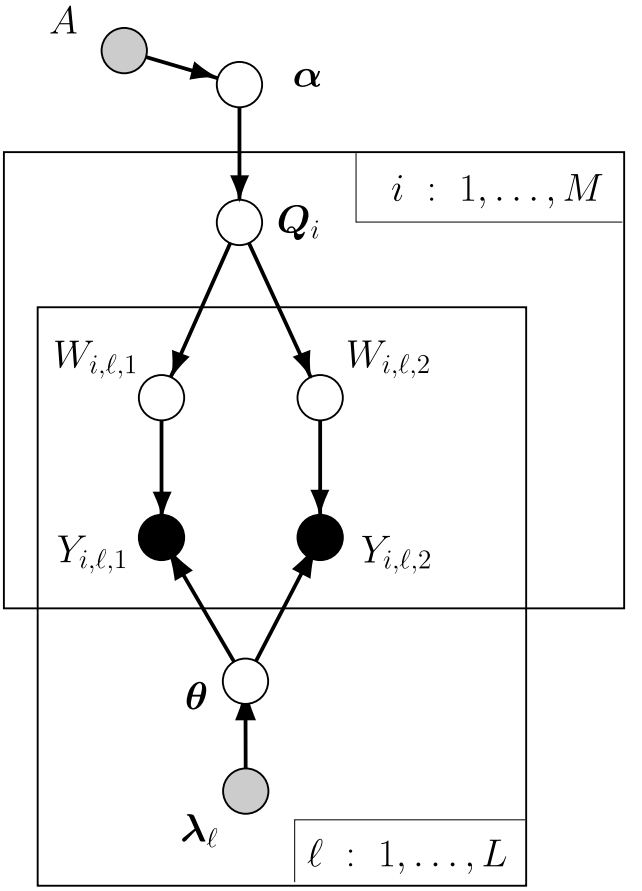
- Discuss component-wise Metropolis-Hastings sampling
- Demonstrate how MCMC is carried out in the *structure* model
- Introduce Gibbs sampling as a special case of Metropolis-Hastings sampling
- Visualize MCMC in *structure* in real time

Inference in the *structure* model:

We might be interested in the posterior mean of the admixture proportions $\boldsymbol{Q}_i$ for each individual in the sample, *i.e.*,

$$\mathbb{E}(\boldsymbol{Q}_i|\mathbf{Y}, A, \boldsymbol{\lambda})$$

This is an expectation over the posterior distribution of $\boldsymbol{Q}_i$.
For most real scenarios, this can't be computed exactly.



However, if we could simulate values of $\boldsymbol{\alpha}$, $\boldsymbol{Q}_i$ $\mathbf{W}$, and $\boldsymbol{\theta}$ from their distribution conditional on $\mathbf{Y}$ and the priors, then we could use that sample in Monte Carlo.

In other words, imagine we have a sample of all the latent variables $(\boldsymbol{\alpha}, \boldsymbol{Q}, \mathbf{W}, \boldsymbol{\theta})^{(1)}$, $(\boldsymbol{\alpha}, \boldsymbol{Q}, \mathbf{W}, \boldsymbol{\theta})^{(2)}$, $\ldots$, $(\boldsymbol{\alpha}, \boldsymbol{Q}, \mathbf{W}, \boldsymbol{\theta})^{(n)}$. Then we could focus only on the simulated $\boldsymbol{Q}_i$'s, for any individuals i, and estimate their posterior mean by

$$\mathbb{E}(\boldsymbol{Q}_i | \mathbf{Y}, A, \boldsymbol{\lambda}) \approx \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{Q}_i^{(j)}$$

where $\boldsymbol{Q}_i^{(j)}$ is just one little component of the multidimensional random variable $(\boldsymbol{\alpha}, \boldsymbol{Q}, \mathbf{W}, \boldsymbol{\theta})^{(j)}$ which is simulated from its conditional distriubution given the genotypic data $\mathbf{Y}$.

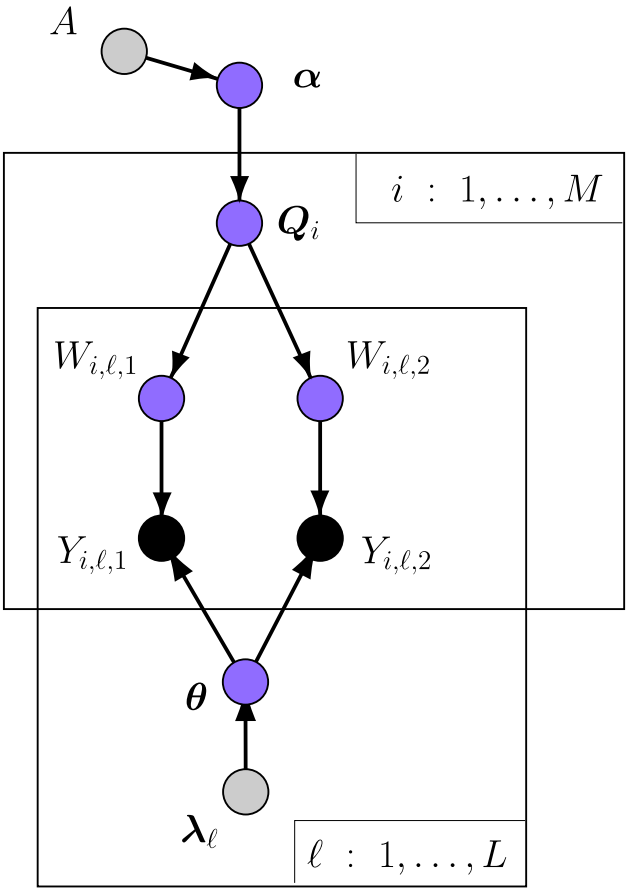
Simulating this "mega-variable" can be done via a *component-wise* Metropolis-Hastings algorithm.

Component-wise Metropolis-Hastings sampling I:

Pretend that you knew the values of all the latent variables (shaded purple here) which we will collectively call $\mathbf{X} = (\boldsymbol{\alpha}, \boldsymbol{Q}, \mathbf{W}, \boldsymbol{\theta})$.

AND

the value of these latent variables was randomly drawn from their conditional distribution given $\mathbf{Y}$ and the priors.

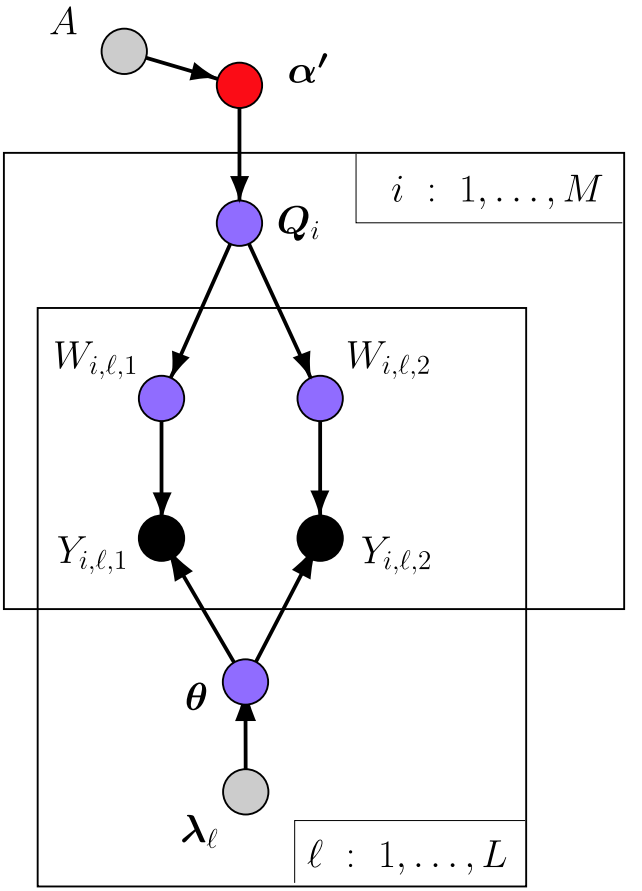


Component-wise Metropolis-Hastings sampling II:

Then, select a variable ($\boldsymbol{\alpha}$ here), and update it using the Metropolis-Hastings algorithm:

1. propose changing it to $\boldsymbol{\alpha}'$
2. accept or reject the proposal according to the M-H ratio

We know by the properties of the M-H algorithm, that the new state of all the variables $\mathbf{X}'$ can also be considered a random draw from $P(\mathbf{X}|\mathbf{Y}, \boldsymbol{\lambda}, A)$.



Performing this process repeatedly, focusing on all the different components of $\mathbf{X}$ in series or in random order will lead to series of states $\mathbf{X}$ that form a Markov chain $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots$ and those values can be used to approximate any expectations of interest.

A happy consequence of the factorization:

The $P(\mathbf{X}'|\mathbf{Y}, \boldsymbol{\lambda}, A)/P(\mathbf{X}|\mathbf{Y}, \boldsymbol{\lambda}, A)$ term in the Hastings ratio mostly cancels out. For example, when $\mathbf{X}'$ involves an altered state of $\boldsymbol{\alpha}$:

$$\frac{P(\mathbf{X}'|\mathbf{Y}, \boldsymbol{\lambda}, A)}{P(\mathbf{X}|\mathbf{Y}, \boldsymbol{\lambda}, A)} = \frac{P(\boldsymbol{\alpha}'|A)}{P(\boldsymbol{\alpha}|A)} \cdot \frac{\prod_{i=1}^{M} P(Q_i|\boldsymbol{\alpha}') \prod_{\ell=1}^{L} P(W_{i,\ell,1}, W_{i,\ell,2}|Q_i) \cdots}{\prod_{i=1}^{M} P(Q_i|\boldsymbol{\alpha}) \prod_{\ell=1}^{L} P(W_{i,\ell,1}, W_{i,\ell,2}|Q_i) \cdots}$$

$$= \frac{P(\boldsymbol{\alpha}'|A)}{P(\boldsymbol{\alpha}|A)} \cdot \frac{\prod_{i=1}^{M} P(Q_i|\boldsymbol{\alpha}')}{\prod_{i=1}^{M} P(Q_i|\boldsymbol{\alpha})}$$

Thus, only a portion of the complete-data, joint distribution must be considered for each update.

The details for an update of α:

The conditional distributions with K subpopulations are as follows:

$$\alpha_k \sim \text{Uniform}(0, A) \ , \ \text{indep. for each } k = 1, \ldots, K$$

$$Q_i|\boldsymbol{\alpha} \sim \text{Dirichlet}(\boldsymbol{\alpha})$$

New values are proposed for a single component k of $\boldsymbol{\alpha}$ from a normal distribution with variance σ^2 centered at the current value α_k:

$$q(\alpha'_k|\alpha_k) \equiv \text{Normal}(\alpha_k, \sigma^2) \ \text{Note : this is symmetrical}$$

So the Hastings ratio for component $k = 2$ looks like:

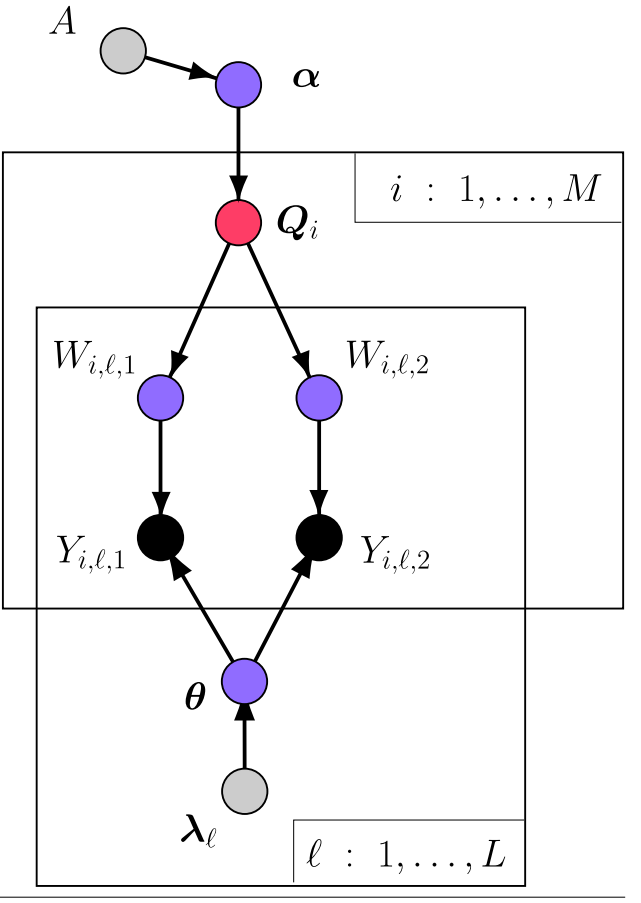
$$\prod_{i=1}^{M} \frac{\Gamma(\alpha_1 + \alpha'_2 + \cdots \alpha_K)}{\Gamma(\alpha_1 + \alpha_2 + \cdots \alpha_K)} \frac{\Gamma(\alpha_2)}{\Gamma(\alpha'_2)} \frac{Q_{i,2}^{\alpha'_2}}{Q_{i,2}^{\alpha_2}}$$

How about updating Q_i?:

Q_i, the admixture proportion for individual i is updated by a special kind of componentwise M-H algorithm called "Gibbs Sampling."

Gibbs sampling (Geman and Geman 1984) is Metropolis-Hastings sampling in which the proposal distribution is the *full conditional* distribution of the variable being updated.

The full conditional distribution is the distribution conditional on all variables that are not being changed in the update.

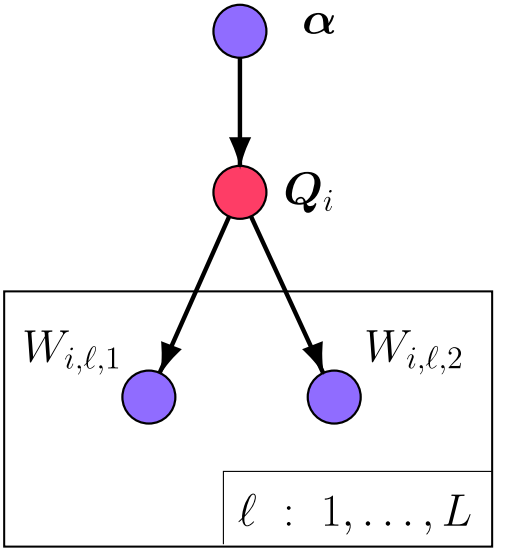


First, focus only on the relevant variables:

The factorization of the joint distribution, and the conditional independencies thereof, again ensure that we only have to worry about variables in a neighborhood around $\boldsymbol{Q}_i$

Also, we can focus on the $\boldsymbol{Q}_i$ for a single individual i at a time, rather than dealing with all M individuals at once.

The relevant variables appear in the pruned graph to the right.



The full conditional distribution is quite simple:

$$P(\boldsymbol{Q}_i|\text{all other variables}) = P(\boldsymbol{Q}|\boldsymbol{\alpha}, \boldsymbol{W})$$

$W_{i,\ell,1}$ is a multinomial variable (of only one trial) indicating which subpopulation the first gene copy, at the ℓ^{th} locus in the i^{th} individual came from.

$$\boldsymbol{Q}_i|\boldsymbol{\alpha} \sim \text{Dirichlet}(\boldsymbol{\alpha})$$

$$W_{i,\ell,a}|\boldsymbol{Q}_i \sim \text{Multinomial}_K(1, \boldsymbol{Q}_i) \quad \text{iid} \quad \forall \ell \text{ and } a = 1, 2$$

So, it is immediately apparent that the full conditional distribution $P(\boldsymbol{Q}|\boldsymbol{\alpha}, \boldsymbol{W})$ is just the posterior distribution for the cell probabilities of multinomial data—the multivariate generalization of the Beta distribution discussed in Session 1.

If $P(\boldsymbol{Q}|\boldsymbol{\alpha}, \boldsymbol{W})$ is used as the proposal distribution $q(\boldsymbol{Q'}|\boldsymbol{Q})$, then it is easy to show that the acceptance probability, computed using the Hastings ratio, is always equal to 1—a convenient feature of Gibbs sampling.

How about the other variables—W and θ?:

The updates for these variables proceed in a similar fashion using Gibbs sampling.

Computer Demo: `pritch` run on `ScottishKittiesForNewHybs.txt`

Computing Practical Session with *structure*

The goals of this practical session—Allow students to:

- Familiarize themselves with *structure*'s data format
- Use the *structure* front end
- Understand how the structure front end prepares input files
- Analyze the Scottish cats data
- Examine the output from *structure* to gain familiarity with visually assessing convergence of different MCMC chains.

Structure data format:

- *structure* expects data in white-space delimited rows
- The genotypes of individuals can be recorded in a single row, or in multiple rows. The cat data uses a single row for each individual.
- The first row holds the marker names (*e.g.*, "fca8 fca23 . . . ").
- Each individual's row contains the individual's ID (in the first column—*e.g.*, "Z12"), then an integer specifying its population or sampling location information (in the second column—*e.g.*, "3"). The remaining columns are the genotypes (two columns for each locus, *e.g.*, "123 143 . . . ").
- These are *microsatellite* genotypes. Each diploid individual has two gene copies, and the different alleles are different lengths of simple repeat DNA segments between conserved primer regions.
- Such data are not used extensively in human genetics any longer, but are still used in molecular ecology. In this case they provide a

compact data set that *structure* can analyze quickly enough to do multiple runs in the course of a short practical session.

- The first five lines of *cats.dat* are below:

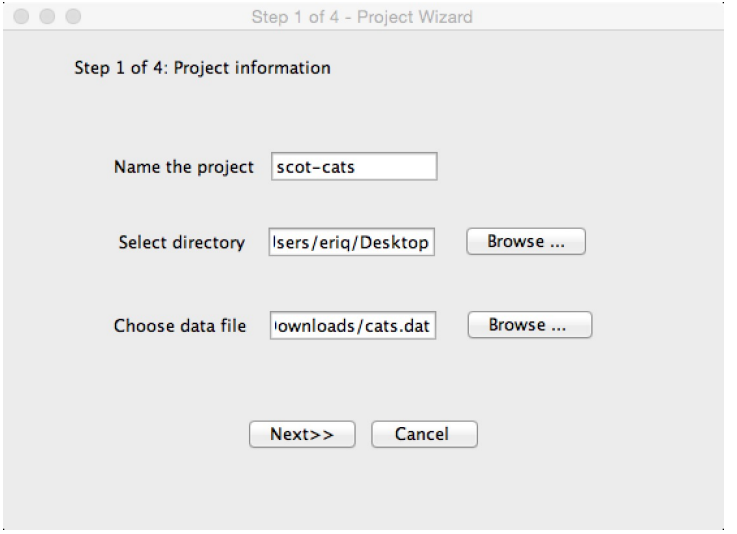
```
fca8      fca23  fca35  fca43  fca45  fca77  fca90  fca96  fca126
Z12 3   123 143 136 140 144 144 120 122 157 158 144 144 107 107 210 212 142 144
Z13 3   135 135 134 140 144 144 120 122 161 161 144 144 93  105 210 220 144 144
Z14 3   115 121 132 150 144 152 118 122 152 158 144 144 107 113 214 220 142 142
Z15 3   121 123 140 140 144 152 122 128 152 152 144 144 93  107 212 210 136 142
```

- Sampling localities 1–6 refer to wild-living cats in Scotland, and localities 7, 8, and 9 refer to house cats sampled from veterinary centers in three regions in the southern UK.

Making a new *structure* project file:

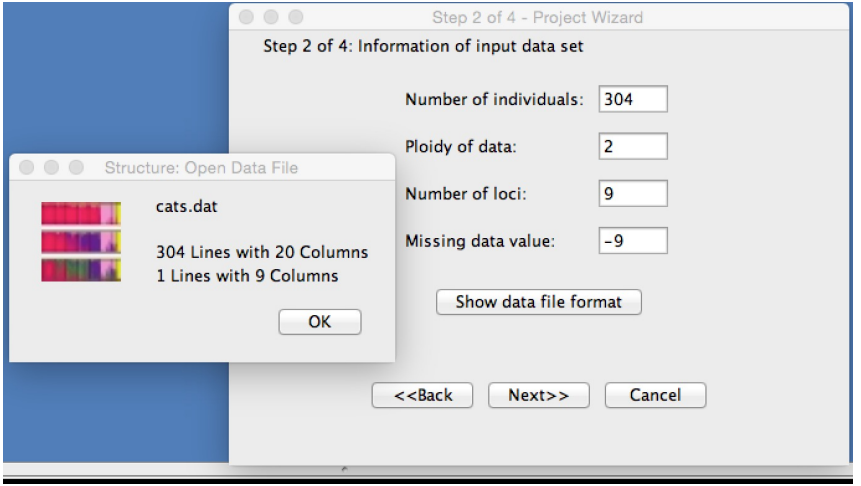
- The Java-based front end of *structure* organizes MCMC runs in the form of projects.
- Each run in a project uses the same data set, but can use different parameter settings for different runs.
- The front end accomplishes this by preparing appropriate input files for *structure* then issuing the appropriate commands to run structure with those input files. It is convenient for people who are not very comfortable in a command line environment.
- The front end also has some useful data visualization capabilities.
- To make a new *structure* project to analyze the *cats.dat* data set:
  1. Launch *structure* (usually by double-clicking its icon or its short-cut). This launches the front end.
  2. Go back to *structure* and choose [File]$\rightarrow$[New Project...] to launch the "Project Wizard". Then give the project a name (I would recommend not putting spaces in the name). In the rest

of this, we will assume we have named the project "scot-cats". Select the directory you wish to put it in (I just put it on my Desktop) and then select the data file *cats.dat* which is part of the SISG module's data. Or, if you wish to download it directly, you can get it from http://tinyurl.com/scot-cats. Once you download it, you have to direct *structure* to where the file resides in your file system.



3. In Step 2 of the "Wizard" we have to report how many indi-

viduals and loci there are in the files. Choose “Show data file format” to get a little information about the file:



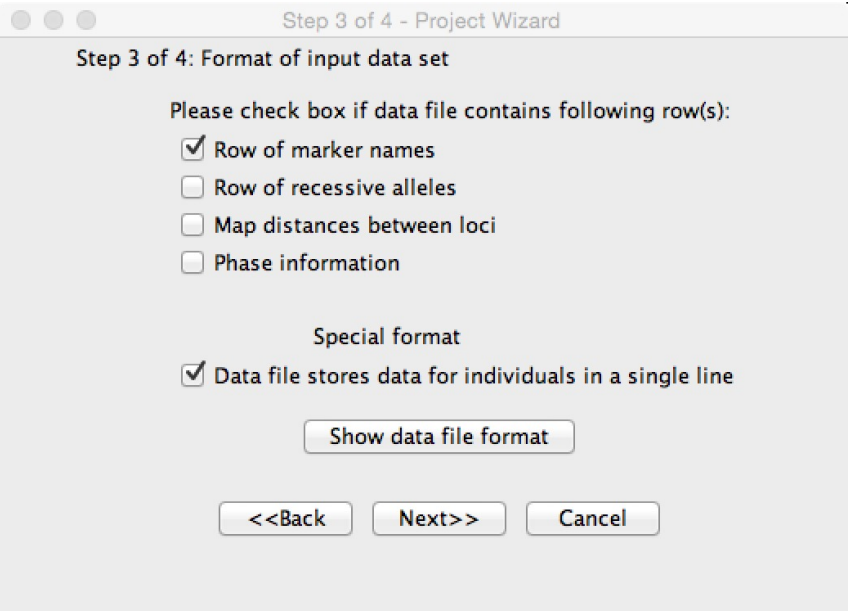
One can see there is one row with 9 locus names then 304 rows of individuals, with 20 columns (2×9 for the loci, plus one for the ID, and one for the population label). So fill in these values:

- Number of individuals: 304
- Ploidy of data: 2

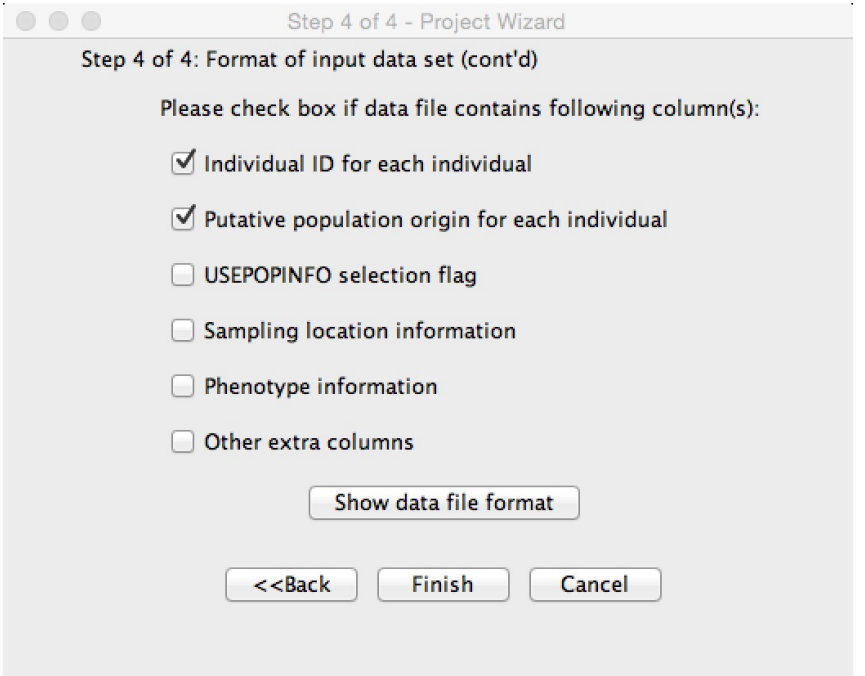
- Number of loci: 9
- Missing data value -9

(If you look through the file you see that missing data values are coded as -9's). Then hit next. . . .

4. Step 3 of the "Wizard": More data file formatting setup. Check the boxes for "Row of marker names" (because the file starts with a row of locus names) and also for "Data file stores data for individuals in a single line."



5. Step 4 of the “Wizard”: Here we have to alert *structure* to any other columns that exist for each individual in a row. Recall that we have an ID for each indiv, and also an integer expressing sampling location. So check the boxes next to “Individual ID for each individual” and “Putative population origin for each individual.”



Then hit Finish. Then Proceed.

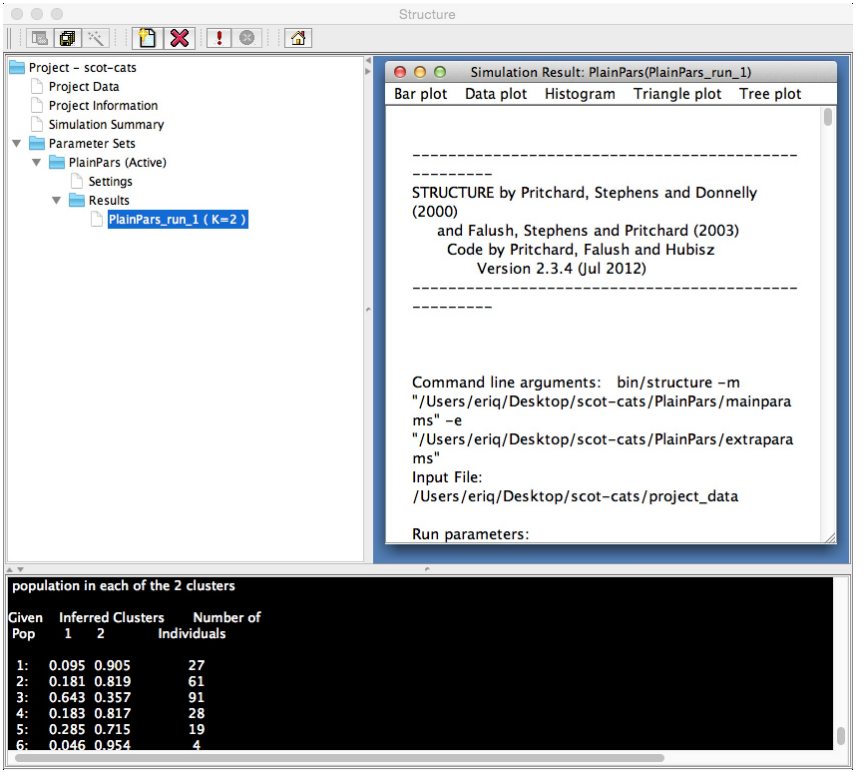
6. That should work, and you should see the data in a spreadsheet window. Now we must specify parameters for a run.

Creating parameter sets:

1. Choose [Parameter Set]→[New. . .]
2. Note that there are tabs to set the Run Length, Ancestry Model, Allele Frequency Model, and “Advanced” options.
3. Let’s do a simple parameter set with mostly default options. We want to get things done pretty quickly, so we will do shorter runs than are typically recommended. Set Burn In at 2000 and MCMC reps after burn-in to 4000.
4. We will use the admixture model with correlated frequencies, etc. These are all the default options.
5. Hit OK, and then name the parameter set. For this example I will name it “PlainPars”. Hit OK. You should now see a summary of the parameter set.
6. Try clicking around in the left hand panel to see the different views of data and parameter sets that are now available.

Doing a *structure* run:

- We can now do a run of *structure* using the parameters in PlainPars. Do this by choosing [Parameter Set]→[Run]. Set the number of assumed populations (K) to 2 and hit OK.
- While this is running, you can view traces of some of the important parameters. For example, choose [Plotting]→[Run-time]→[Alpha] to view a trace of values of α inferred by *structure*.
- Be sure to kill the trace plot windows when you are done with them because (at least on older versions I used) they slow down the progress immensely.
- When the run has finished, click on "Results" under "PlainPars" in the left panel, so it expands out and you can click on "PlainPars_run_1 (K=2)". With that highlighted, the right panel should give the text summary of the *structure* output, and the right panel also has pull-down menus to plot the output in various ways:


Structure
Project - scot-cats
Project Data
Project Information
Simulation Summary
Parameter Sets
PlainPars (Active)
Settings
Results
PlainPars_run_1 (K=2)
Simulation Result: PlainPars(PlainPars_run_1)
Bar plot
Data plot
Histogram
Triangle plot
Tree plot
STRUCTURE by Pritchard, Stephens and Donnelly (2000)
and Falush, Stephens and Pritchard (2003)
Code by Pritchard, Falush and Hubisz
Version 2.3.4 (Jul 2012)
Command line arguments: bin/structure -m "/Users/eriq/Desktop/scot-cats/PlainPars/mainparams" -e "/Users/eriq/Desktop/scot-cats/PlainPars/extraparams"
Input File: /Users/eriq/Desktop/scot-cats/project_data
Run parameters:
population in each of the 2 clusters
Given Pop | Inferred Clusters 1 | 2 | Number of Individuals
1: 0.095 0.905 27
2: 0.181 0.819 61
3: 0.643 0.357 91
4: 0.183 0.817 28
5: 0.285 0.715 19
6: 0.046 0.954 4

Plotting results using the front end:

From the right hand window when results of any run are selected you can

- Make a Bar Plot that shows each individual's Q (admixture proportions). To get this plot, choose [Bar plot]$\rightarrow$[Show]. To see it up closer, you can select the "Plot in multiple lines" radio button. The result should look like:

Scroll through this plot and note that the individuals from sampling locales 7, 8, and 9, (the house cats) consistently have a large proportion of ancestry from one of the subpopulations. That subpopulation presumably represents domestic cat ancestry. Interestingly, some of the wild-living cats have just as much “domestic cat ancestry” as do

the cats from the vet clinics.

- You can also plot traces of variables like α, and the complete data log likelihood. Choose [Data plot] for these. These can give you some idea of how well the MCMC is mixing.
- Try creating some histogram plots, as well.

The underlying parameter files:

When the front end executes a run of *structure*, it creates a *mainparams* and an *extraparams* file. These are the files that you would use to execute the same run on the command line.

To see what these files look like, you can find these two files in the PlainPars directory in your *structure* project folder, and open them up with any suitable text editor

mainparams

File Path : ~/Desktop/scot-cats/PlainPars/mainparams

mainparams

```
#define OUTFILE /Users/eriq/Desktop/scot-cats/PlainPars/Results/PlainPars_run_1
#define INFILE /Users/eriq/Desktop/scot-cats/project_data
#define NUMINDS 304
#define NUMLOCI 9
#define LABEL 1
#define POPDATA 1
#define POPFLAG 0
#define LOCDATA 0
#define PHENOTYPE 0
#define MARKERNAMES 1
#define MAPDISTANCES 0
#define ONEROWPERIND 1
#define PHASEINFO 0
#define PHASED 0
#define RECESSIVEALLELES 0
#define EXTRACOLS 0
#define MISSING -9
#define PLOIDY 2
#define MAXPOPS 2
#define BURNIN 2000
#define NUMREPS 4000


#define NOADMIX 0
#define LINKAGE 0
#define USEPOPINFO 0

#define LOCPRIOR 0
#define INFERALPHA 1
#define ALPHA 1.0
#define POPALPHAS 0
#define UNIFPRIORALPHA 1
#define ALPHAMAX 10.0
#define ALPHAPROPSD 0.025


#define FREQSCORR 1
#define ONEFST 0
#define FPRIORMEAN 0.01
#define FPRIORSD 0.05


#define INFERLAMBDA 0
#define LAMBDA 1.0
#define COMPUTEPROB 1
#define PFROMPOPFLAGONLY 0
#define ANCESTDIST 0
```

Line 54 Col 1 | (none) | Unix (LF) | Last saved: 7/2/15, 11:29:55 AM | 1,012 / 141 / 54

Scheduling a series of runs:

One very useful feature of the front end is that it allows you to set up a series of runs. This is especially useful for investigating different values of K.

We will use this feature to investigate model selection for different values of K:

1. Select [Project]$\rightarrow$[Start a Job]
2. Then select "PlainPars" from the Structure Scheduler
3. Set K from 1 to 4
4. Set the number of iterations to 2 (or more if your computer crunches through each run in little time.)
5. Hit start.

That should finish in a few minutes. Once it is done, note that all the

results are neatly stored in the Results directory of the left panel.

If you did more than one iteration for each K, compare the values obtained for $\mathrm{Ln}P(D)$ in the different runs at the same value of K. Especially for $K = 4$ there may be marked disparities due to mixing anomalies.

Look at trace plots of the Ln Likelihood values for runs that look like they may be a little anomalous.

Remember that inference for K using *structure* is quite *ad hoc*.

Assessing Convergence By Examining Traces:

One of the unfortunate deficiencies of the *structure* front end is that it does not print out all the standard output of the program structure. That output includes current values of a number of variables in the MCMC every sweep. Instead, those data are stored within Java objects by the *structure* front end.

While that approach allows you to make trace plots for single runs in the front end, it is hard to compare the results from multiple runs.

Since one of the themes of this course is that it is essential to compare the results from multiple MCMC chains, we've prepared a little R package that makes it easy to do multiple runs of *structure* and then compare the runs using the R package `ggplot2`.

Instructions on using it are available in an R notebook. Find a link to it under "Section 6" at: http://eriqande.github.io/sisg_mcmc_course/

Taking it further:

That was an introductory analysis using *structure*. Here are some suggestions for different analyses that you might want to experiment with if you have more time:

1. Try analyzing the data using the model with prior population information and 9 different populations. (This is a little silly, but a good exercise nonetheless). Do this by creating a new parameter set in which the Ancestry Model is "Use population information" and set the migration rate parameter, ν, at a value around .01 to .05. Repeat the analysis with a ν of about .15 or .25 and see how sensitive the results are to the assumed migration rate.
2. Note that the first analysis we did was a little naive, because, in terms of estimating α, it essentially assumes that all individuals were randomly sampled from a big grab-bag of potentially-admixed wild-living cats. In reality, the cats from the vet clinics should not be used to estimate α. Rather we might desire that

α would reflect the degree of admixture amongst *wild-living* cats. This problem can be rectified by specifying that the cats from the vet clinic are a "learning sample." Do this in the following way:

(a) Create a new data set that has both a PopData and a PopFlag column.

(b) Make the PopData column hold 1's for all the wild-living cats, and 2's for all the vet-clinic cats (those that originally had 7, 8, or 9). (This—and the next task—is probably most easily done with a spreadsheet program).

(c) Make the next column the PopFlag column. Give all the vet-clinic cats 1's, and everything else 0's.

(d) Make a new project using this data set, and run the analysis with prior population information with a negligible migration rate.

3. Finally, we may wonder how well *structure* would perform if it didn't have the learning sample of vet-clinic cats. So, make a new data set in which you have deleted all cats with PopData of 7, 8, or 9.

Then analyze the remaining cats using the admixture model with no prior population information. Do this for $K = 2$ and compare the allele frequencies estimated for the two subpopulations with the results obtained from our original analysis. Investigate how inference for K is affected.

:

The Haplotyping Problem

A Bayesian Approach

:

The Haplotype Problem

- Suppose we genotype individuals at a number of tightly linked SNPs.

A C G C C T T T G C G C

G A A C C C C C A G G C

:

The Haplotype Problem

- Suppose we genotype individuals at a number of tightly linked SNPs.

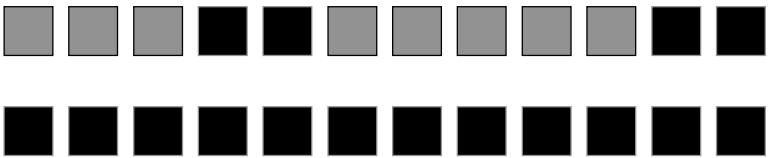
A C G C C T T T G C G C

G A A C C C C C A G G C

:

The Haplotype Problem

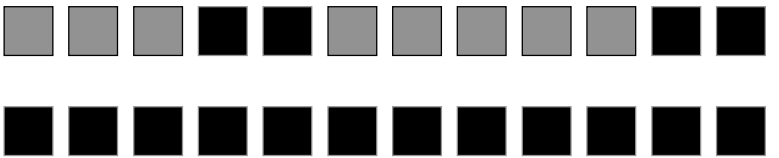
- Suppose we genotype individuals at a number of tightly linked SNPs.



:

The Haplotype Problem

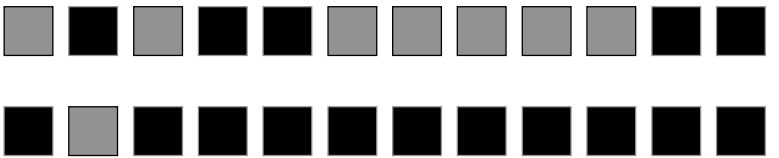
- What do the types on the two chromosomes look like?



:

The Haplotype Problem

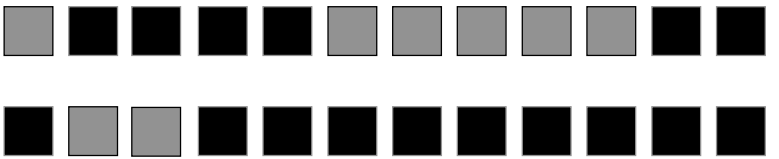
- What do the types on the two chromosomes look like?



:

The Haplotype Problem

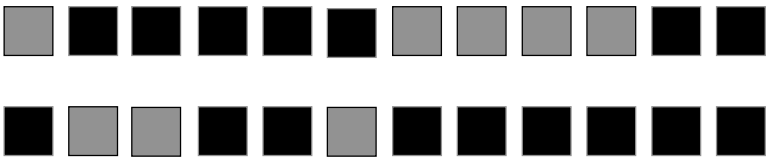
- What do the types on the two chromosomes look like?



:

The Haplotype Problem

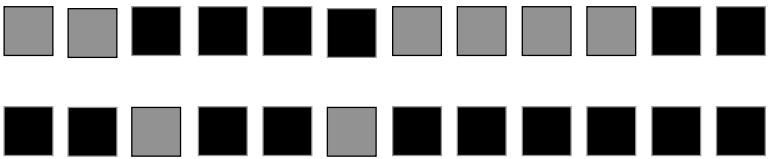
- What do the types on the two chromosomes look like?



:

The Haplotype Problem

- What do the types on the two chromosomes look like?



:

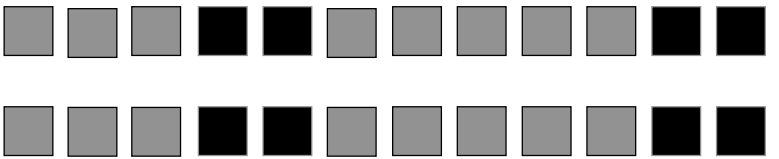
The Haplotype Problem – potential solutions

- Molecular methods
- Collect family data
- Statistical methods

:

The Simplest Case

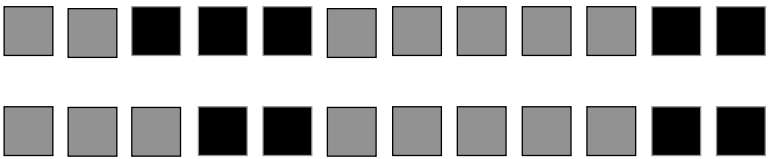
- What do the types on the two chromosomes look like?



:

The Next Simplest Case

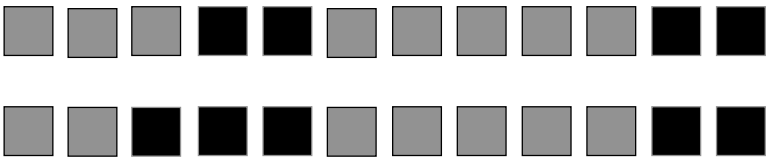
- What do the types on the two chromosomes look like?



:

The Next Simplest Case

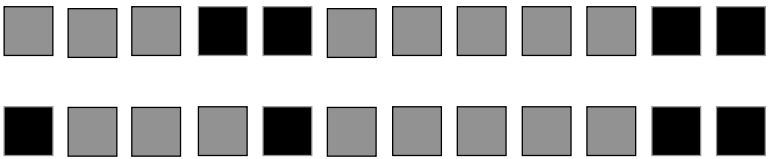
- What do the types on the two chromosomes look like?



:

The first difficult case…

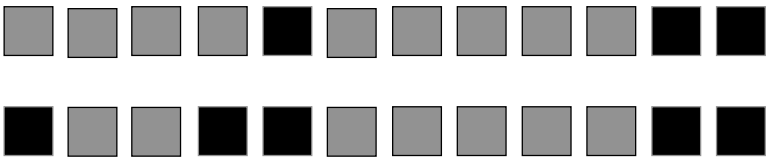
- What do the types on the two chromosomes look like?



:

The first difficult case…

- What do the types on the two chromosomes look like?

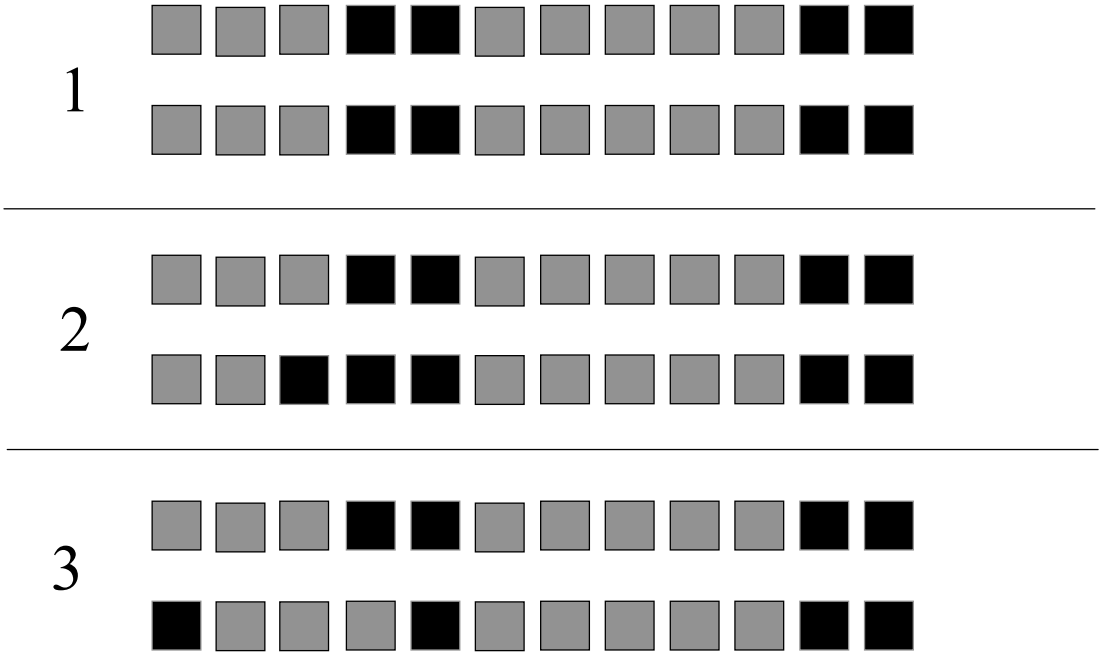


:

Clark's Method (1990)

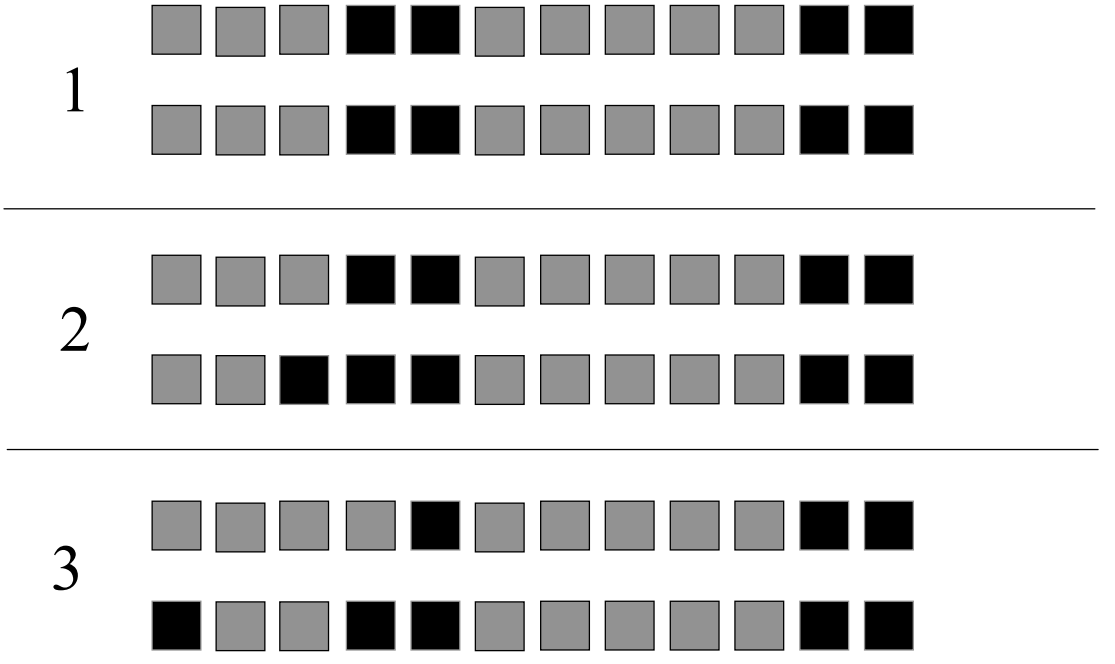
- Idea: use information obtained from other individuals in the population to determine the most probable haplotype pair.

:

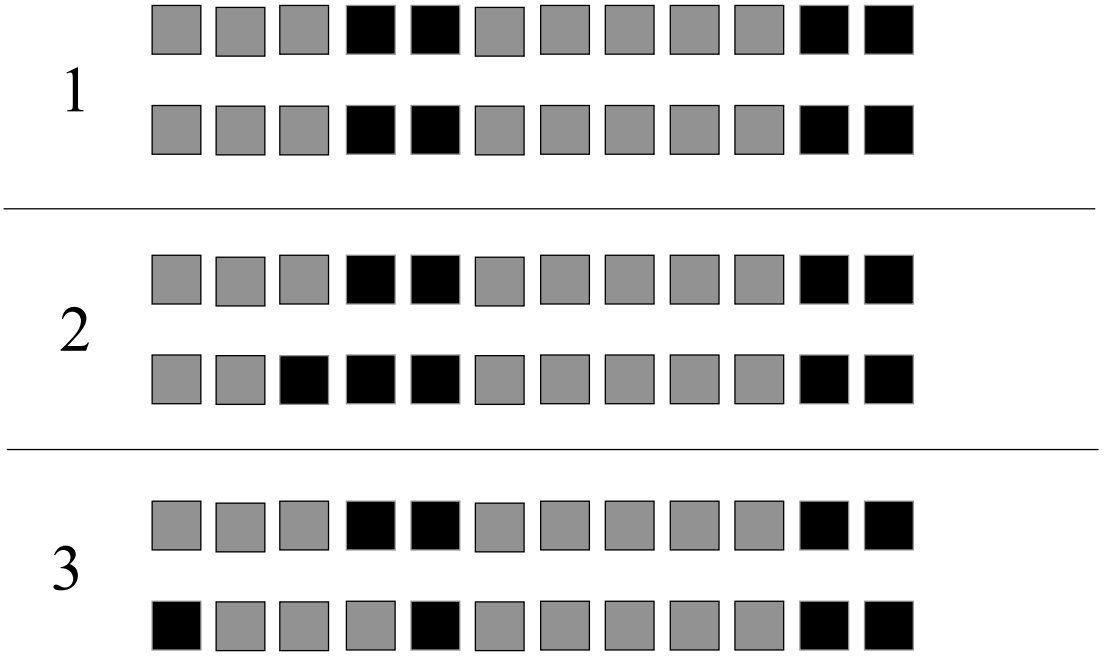


Is it this configuration?

:


1
2
3
…or this one?

:



This one is more probable.

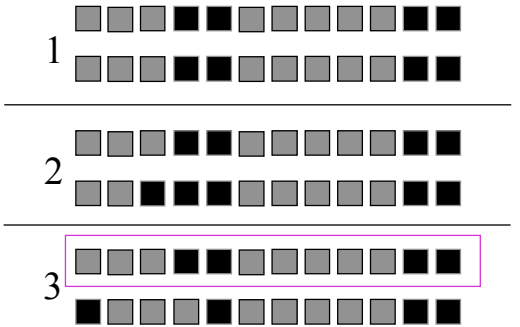
:

Clark's Method (Clark, 1990)

- Identify the unambiguous individuals.
- Make a list of "known" haplotypes.
- Go through list, and see whether ambiguous individuals can be made up from a "known" haplotype plus another "complementary" haplotype. If so, add the complementary haplotype to the list of "known" haplotypes.

:

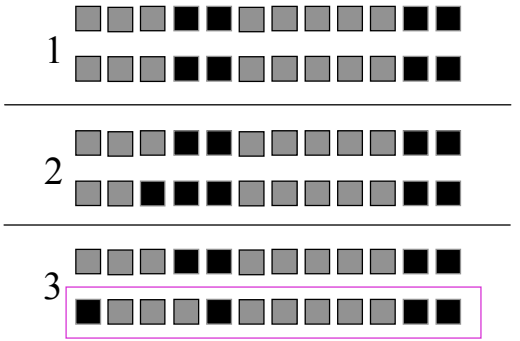
Clark's Method



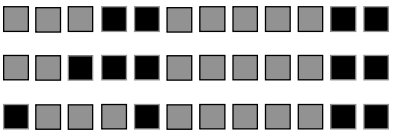
List of known haps.

:

Clark's Method

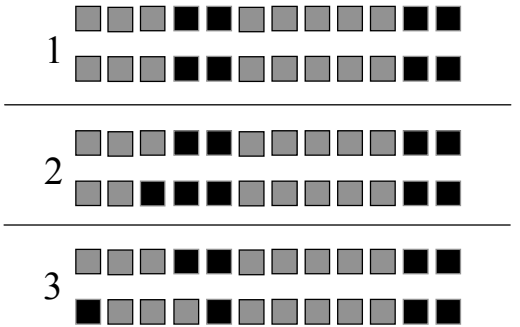


List of known haps.



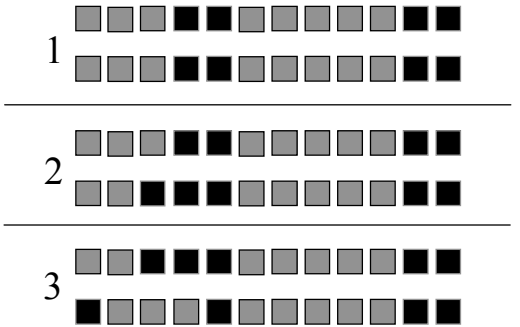
:

Clark's Method: Problem 1



:

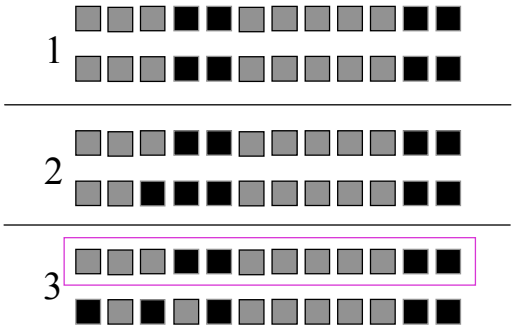
Clark's Method: Problem 1



List of known haps.

:

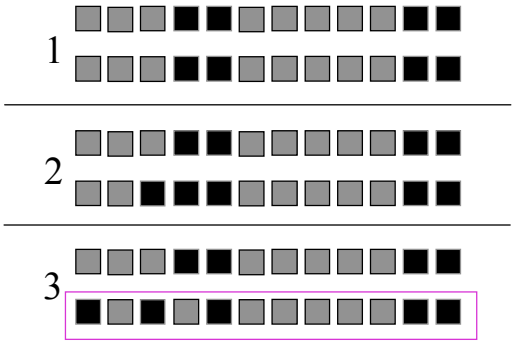
Clark's Method: Problem 1



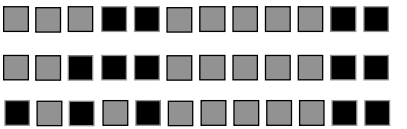
List of known haps.

:

Clark's Method: Problem 1

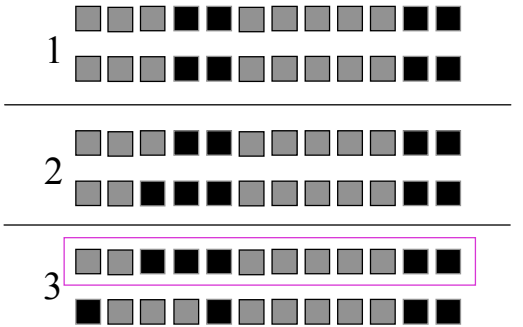


List of known haps.



:

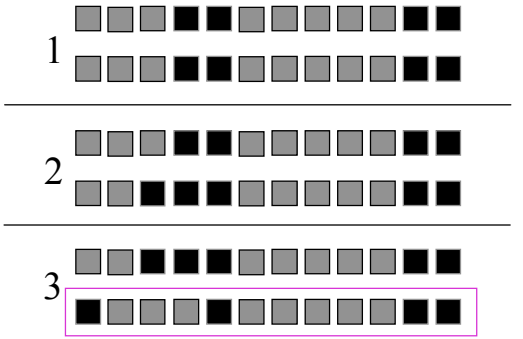
Clark's Method: Problem 1



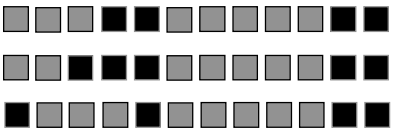
List of known haps.

:

Clark's Method: Problem 1



List of known haps.

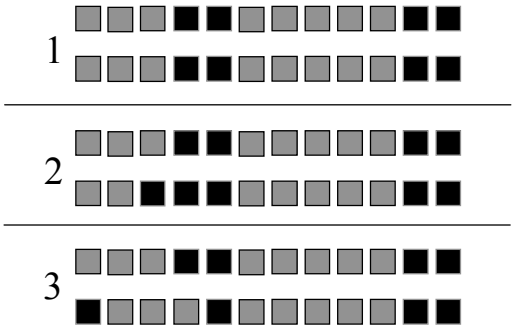


Answer depends on order list is considered….

… and frequency information is ignored

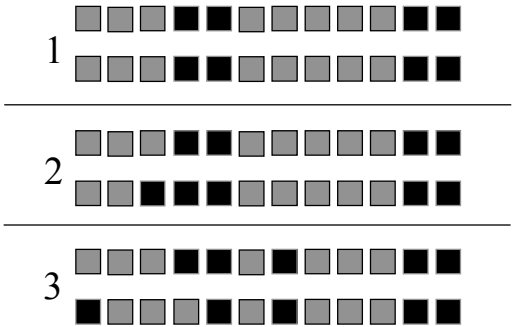
:

Clark's Method: Problem 2



:

Clark's Method: Problem 2



List of known haps.

Algorithm can fail to resolve all haplotypes…

… because looks only for exact matches

:

Clark's Algorithm: Summary

- Results may depend on order individuals are considered.
- Frequency information is ignored.
- Looks only for exact matches.
- May fail to resolve all haplotypes.
- Fails to assess uncertainty.

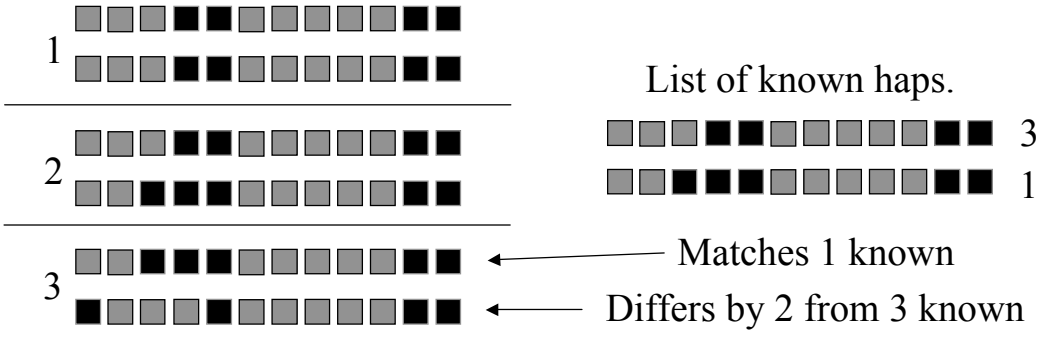
:

Strategies for improvement

- Replace single pass through data, with iterative scheme (MCMC).
- Use frequency information.
- Take into account “near misses”, as well as exact matches.

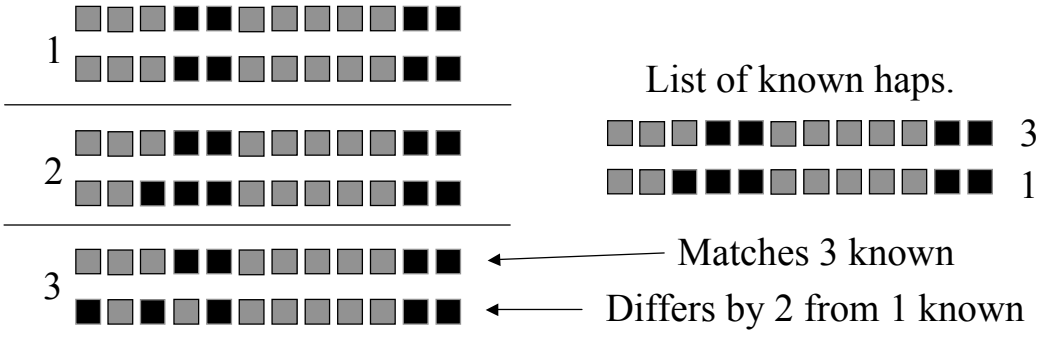
:

Example


1
2
3
List of known haps.
3
1
Matches 1 known
Differs by 2 from 3 known

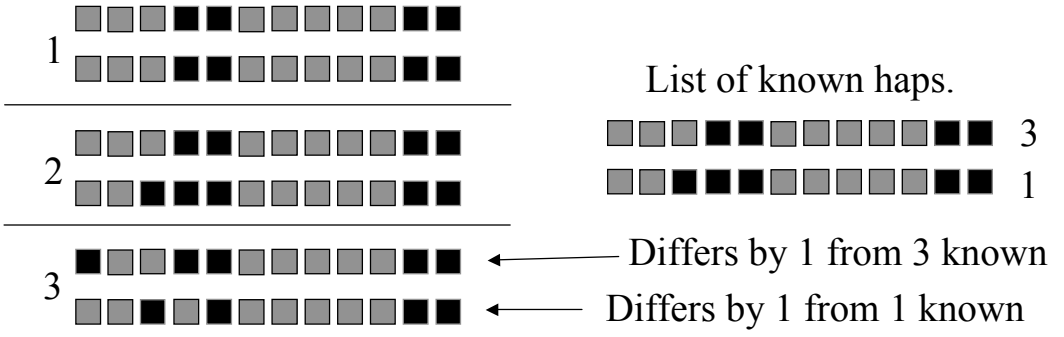
:

Example


1
2
3
List of known haps.
3
1
Matches 3 known
Differs by 2 from 1 known

:

Example



How to balance these possibilities?

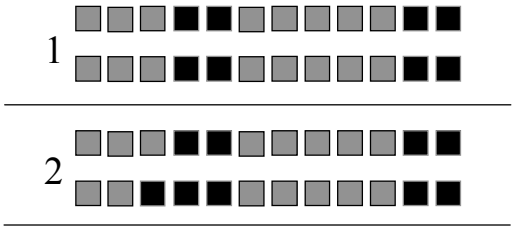
:

The key question

- What is the conditional distribution of the next haplotype, given a set of known haplotypes?
- Answer to this question allows a Gibbs sampler to be implemented.

:

Example



Given the above haplotypes, what would you expect the next haplotype to look like?

:

A Naïve Gibbs Sampler

- Put standard conjugate prior on the N_H haplotype frequencies:

$$\text{Dirichlet}(\alpha, \ldots, \alpha)$$

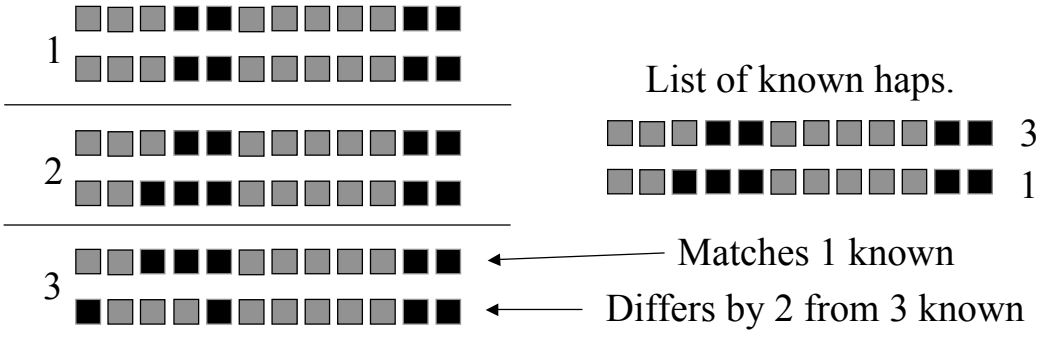
- Gives simple conditional distribution:

$$\Pr(H_{n+1} \mid H_1, H_2, \ldots, H_n) = \frac{\#\{i : H_i = H_{n+1}\} + \alpha}{n + N_H \alpha}$$

- Uses frequencies, but ignores near misses.

:

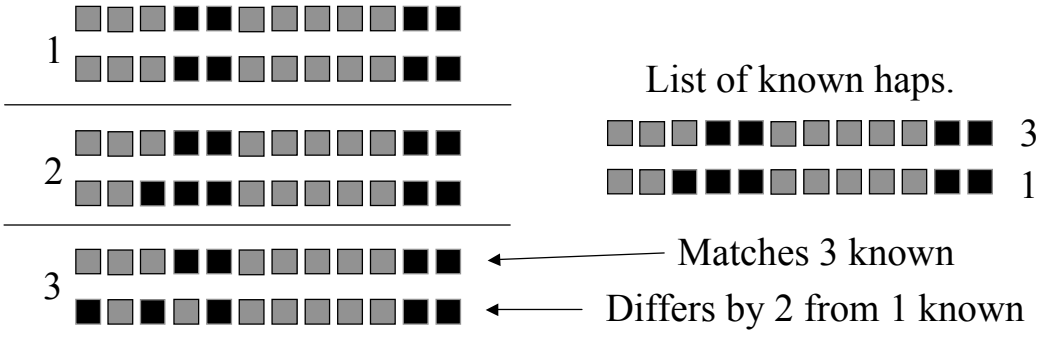
Example



This would have probability proportional to $(1+\alpha)\ \alpha$

:

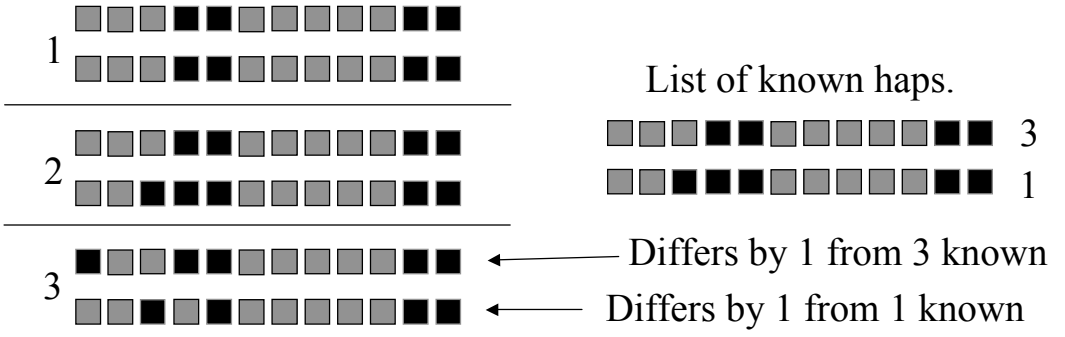
Example



This would have probability proportional to $(3+\alpha)\ \alpha$

:

Example



This would have probability proportional to $\alpha\ \alpha$

:

A Pseudo-Gibbs Sampler

- Want the conditional distribution to capture idea that next haplotype will be same as ***or similar to*** an existing haplotype.
- Do this by approximating the conditional distribution directly (rather than specifying prior).

:

A Pseudo-Gibbs Sampler (cont)

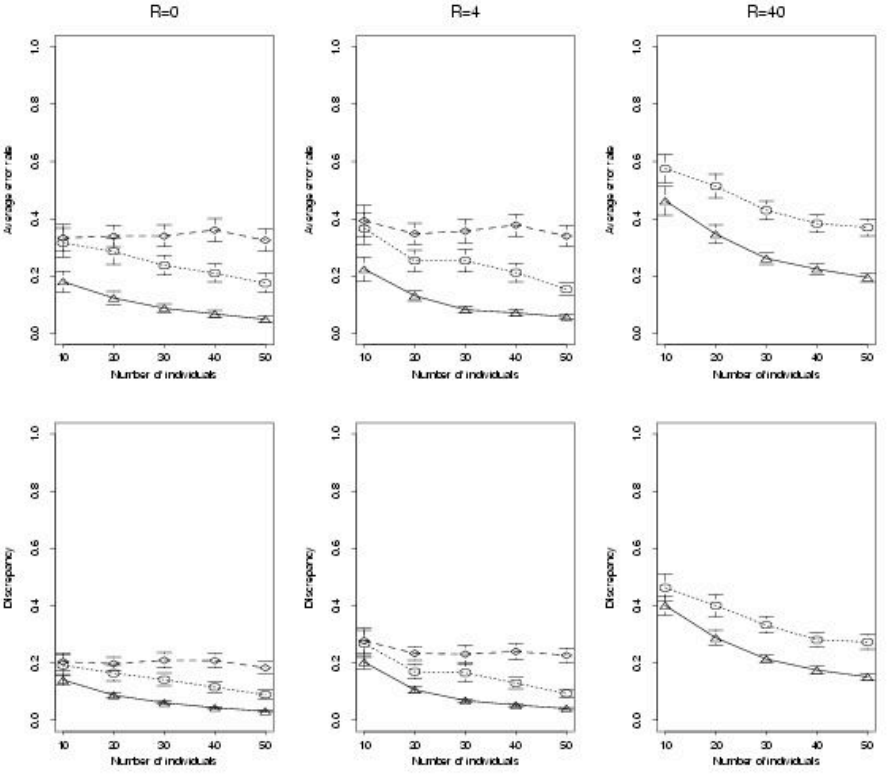
- We assumed that next haplotype differs by M mutations from a randomly-chosen existing haplotype, where M has a geometric distribution.
- Motivated by population genetics theory (the Ewens sampling formula) we choose

$$\mathbf{Pr}(M=0) = n/(n+\theta)$$

- Does not, in general, correspond to a prior.

:

Comparisons on simulated data


R=0
R=4
R=40
Average error rate
Discrepancy
Number of individuals
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50

:

MCMC scheme also estimates missing genotypes

- SeattleSNP data; 48 genes, 48 individuals, average ~80 SNPs per gene.

Error rates for imputing missing data:

PHASE v2.0 (NR)	PHASE v2.0 (R)	Naïve Gibbs	HAP	Straw
3.1%	2.8%	4.2%	6.3%	18.3%

:

Something to think about

- What if the data look like this: n individuals, typed at two loci, all heterozygous at both loci. (ie all 01,01).
- What is the MLE for the haplotype frequencies?
- Conditional on this MLE, what are the diplotype distributions for each individual?
- What is the Bayesian solution?
- What kinds of problems might we expect to come across in MCMC? (especially for large n).

Practical on haplotype inference

All input files are available at

https://github.com/eriqande/sisg_mcmc_course/tree/master/computing_practicals/phase_data

You will need to save the input files example1.txt, example2.txt and example3.txt to your local computer in the same directory as PHASE lives.

Then open an MSDOS window, and change to the directory where PHASE lives.

Example 1

The input file example1.txt contains an input file for the PHASE software. Here is what it looks like, with some comments

```
3 (this is the number of individuals)
5 (number of SNPs)
P 100 200 300 400 500 (positions of SNPs)
SSSSS (types of loci: all 5 are SNPs)
#1   (individual label - this is data for individual 1)
11111   (5 columns, 2 rows, each column is 1 genotype)
11111   (this case is homozygous for 1 allele at all SNPs)
#2
00000
00000
#3
00000  (this one is heterozygous at all SNPs)
11111
```

Before running PHASE on this file, here are some things to think about:

- How many of the individuals have ambiguous haplotypes?
- What would you guess for the haplotypes of the ambiguous individual(s)?
- How confident would you be? Very confident? Less confident? (What might make you more confident?)

- Try running PHASE on this file using

  ```
  PHASE example1.txt example1.out
  ```

  Take a look at the output files example1.out, and example1.out_pairs, and see if PHASE's answers correspond to your intuition.

  You might also like to look at the output file example1.out_freqs which contains estimates of the population haplotype frequencies.

- Try running PHASE again:

  ```
  PHASE example1.txt example1.2.out
  ```

  Do you get the same answers? You should, because by default PHASE uses the same random numbers to simulate the Markov chain.

Changing the Seed

To change the random numbers used, you need to use the -S option to set the ”seed”:

PHASE -S332554 example1.txt example1.3.out

The answers should look similar, but not identical.

Performing multiple runs with different seeds is a helpful way to check that you are running the algorithm long enough to get reliable results.

Example 2

The file example2.txt contains another small input file. This one was created by putting together individuals by randomly pairing haplotypes taken from a pool containing equal numbers of the 8 possible 3-SNP haplotypes.

- What would you expect to happen for this input file?
- Try running PHASE a few times, with different seeds, and compare the results.
- As well as estimating the haplotypes for each individual (eg in the _pairs file) PHASE also estimates population haplotype frequencies: see the _freqs file. Compare these estimates with the description of how the input file was created.
- One way to estimate haplotype frequencies is to first estimate the haplotypes for each individual and then to count up how many times each haplotype occurs in these estimates. But this is the

kind of 2-stage procedure that should be avoided. Can you think of another way?

- By default PHASE estimates recombination rates, and uses conditional distributions based on these rates within a model known as the coalescent. One can also specify, among other things, that PHASE is to use a Dirichlet prior for the haplotype frequencies (use the -ME option), or to assume a coalescent prior with no recombination (-MS). The main reason one might want to use these options instead of the default is that for big problems they can run appreciably quicker. Try running PHASE with these options on this data set, and compare the results (eg the _freqs file) with the results from the default.

Example 3

In addition to estimating haplotypes, PHASE can also be used to estimate recombination rates in a region, and to assess whether the region contains a recombination ”hotspot”. By invoking the -MR1 1 option, PHASE assumes that the recombination rate in the region is constant, with the possible exception of a single recombination hotspot somewhere in the gene. PHASE makes various assumptions about the location, width and intensity of the hotspot by making assumptions about the prior distributions of these quantities. It then outputs a _hotspot file, which contains a sample from the posterior distribution for these quantities.

The input file example3.txt contains data from a gene CD36. Run PHASE using the -MR1 1 option:

./PHASE -MR1 1 example3.txt example3.out

and examine the contents of the file example3.out_hotspot.

The following code can be used to read the _hotspot file into R and plot summaries. (Or you could use Excel if you prefer.) Try performing multiple runs of PHASE and comparing results (eg the posterior distribution of the hotspot intensity) for different seeds.

```
h = read.table("example3.out_hotspot")
hist(h[,4]) # h[,4] contains samples from the posterior
            # distribution of intensities
plot(h[,4])
mean(h[,4]>1) # returns the proportion of samples
              # where the intensity is > 1
              # (note that intensity 1 corresponds to
              # "no hotspot")
hist(h[,3]-h[,2]) # h[,3] and h[,2] are samples from the
                  # limits of the hotspot
plot(h[,3]-h[,2])
hist(h[,1]) # h[,1] is an estimate of the recombination
            # parameter outside the hotspot
sum(h[,4]>1)
```

Bayesian Model Choice

In Bayesian inference model choice is, in principle, no different from other inferential procedures: put priors on the models and compute their posterior probabilities.

Suppose, M_1 and M_2 are two models under consideration (not necessarily nested). Suppose observe data Y. Then, by Bayes Theorem, relative plausibility of M_1 and M_2 are given by

$$\frac{\Pr(M_1|Y)}{\Pr(M_2|Y)} = \frac{\Pr(M_1)}{\Pr(M_2)} \frac{\Pr(Y|M_1)}{\Pr(Y|M_2)}$$

or

$$\text{Posterior odds} = \text{Prior odds} \times \text{Bayes Factor}.$$

Odds:

Note that, assuming M_1 and M_2 are the only possible models, then the posterior odds determines their posterior probabilities, since $\Pr(M_1) + \Pr(M_2) = 1$.

Eg: if posterior odds = 1, then posterior probability of each model is 0.5.

If posterior odds = 10 the posterior probability of model $M_1 \approx 0.91$ ($0.91/0.09 \approx 10$).

Bayes Factor:

The Bayes factor, is a convenient summary of the evidence in the data for M_1 vs M_2.

Note that it does not depend on the prior probabilities assigned to models M_1 and M_2. But *does* depend on priors on parameters in the two models.

$$\text{BF} = \frac{p(Y|M_1)}{p(Y|M_2)} = \frac{\int p(Y|\theta_1, M_1)p(\theta_1|M_1)d\theta_1}{\int p(Y|\theta_2, M_2)p(\theta_2|M_2)d\theta_2}$$

These integrals are sometimes referred to as the “marginal likelihoods”.

Note: BF is somewhat similar to a likelihood ratio, but involves integration rather than maximisation.

Example:

Consider throwing a six-sided die, with probability q_i of throwing an i.

$$M_1 : q_i = 1/6$$

$$M_2 : q_i = \begin{cases} 0 & (i \neq 6) \\ 1 & (i = 6) \end{cases}$$

If we observe a single throw, of 1, BF $= \infty$.

If we observe a single throw, of 6, BF $= 1/6$.

Example:

Consider throwing a six-sided die, with probability q_i of throwing an i.

$$M_1 : q_i = 1/6$$

$$M_2 : q \sim \text{Dir}(q; 1, 1, 1, 1, 1, 1)$$

If we observe a single throw, of 1, BF $= 1$.

If we observe a single throw, of 6, BF $= 1$.

Interpretation of BF:

Note that BF cannot really be interpreted in isolation, without consideration of the prior odds.

For example, in studies where M_1 and M_2 are both considered somewhat equally plausible *a priori*, a BF of 10 constitutes fairly strong evidence for M_1 vs M_2.

In a genome-wide association study, where very few variants are expected to be associated with disease, the prior odds is very small (eg $1/10^4$), so a BF of 10 is worth hardly a mention.... unless it is for a SNP near a gene that is a good candidate for influencing disease susceptibility.

Model Choice: general considerations:

In general, model choice is a difficult problem, best avoided unless really necessary. Unfortunately it is often necessary.

The Bayes Factor for model M_1 vs M_2 depends, roughly, on the amount of prior mass each model has on parameters consistent with observed data. (This naturally penalises more complex models: no need to do this explicitly.)

If you have a natural "null" hypothesis, against which to compare, then the "alternative" should place moderate prior mass on parameter values near to the null.

Eg: in a drug trial, a natural null hypothesis is that there is no difference between two treatments ($\theta = 0$).

Therefore, the alternative hypothesis should include moderate mass on values of θ "close" to 0.

Effects of Prior in *structure* on Bayes Factors:

The *structure* model implements two different priors on the allele frequencies within each subpopulation.

The "correlated allele frequencies" prior assumes that the allele frequencies in different subpopulations are correlated *a priori*.

The other (original) prior assumes that they are independent *a priori*.

The second model places much less prior mass on the allele frequencies being very similar.

If the truth is that there are $K = 2$ subpopulations, but with very similar allele frequencies, which prior will tend to give a larger BF for $K = 2$ vs $K = 1$?

Computing BFs:

Not only are BFs sensitive to prior assumptions, but they are also difficult to compute!

Recall that BF $= P(Y|M_1)/P(Y|M_2)$, where each term in this ratio is an integral:

$$P(Y|M_1) = \int_\theta P(Y|\theta)P(\theta|M_1)d\theta$$

In high-dimensional problems this integral is difficult to compute (although Importance Sampling or numerical integration methods are sometimes used successfully). See also Gelman and Meng (1998) for review of methods.

Computing BFs via MCMC:

Note that the integral is the normalizing constant in Bayes Law:

$$P(\theta|Y) = \frac{P(Y|\theta)P(\theta|M_1)}{\int_\theta P(Y|\theta)P(\theta|M_1)d\theta}$$

One of the most helpful features of MCMC is that it allows one to sample from this distribution *without* evaluating the normalizing constant.

It turns out that it is also possible to compute BFs without evaluating this integral. The idea is to use MCMC to sample not only parameters within a model, but also to move between the two models.

This allows one to compute the posterior probability of the two models, and hence the posterior odds, from which one obtains the BF.

Sampling across models via MCMC:

Approach 1: do standard MCMC on the space I, θ_1, θ_2, where I is an indicator (1 or 2) for whether we are in model 1 or model 2, and the likelihood is

$$\Pr(Y|I, \theta_1, \theta_2) = Pr(Y|M_I, \theta_I).$$

Approach 2: do MCMC on the space (I, θ_I). This involves jumping between spaces of different dimension, for which the standard MH algorithm is not directly applicable, but a modified form ("reversible jump MCMC") exists (Green, 1995).

In both cases the challenge is to jump between the two models. In approach 1 we just have to hope that it happens. In approach 2 one can try to be clever to make it happen, but being clever enough is difficult in high-dimensional problems.

Example: Structure and the number of subpopulations:

In our discussion of *structure* up to this point, we have mostly assumed that the number of subpopulations, K, is known.

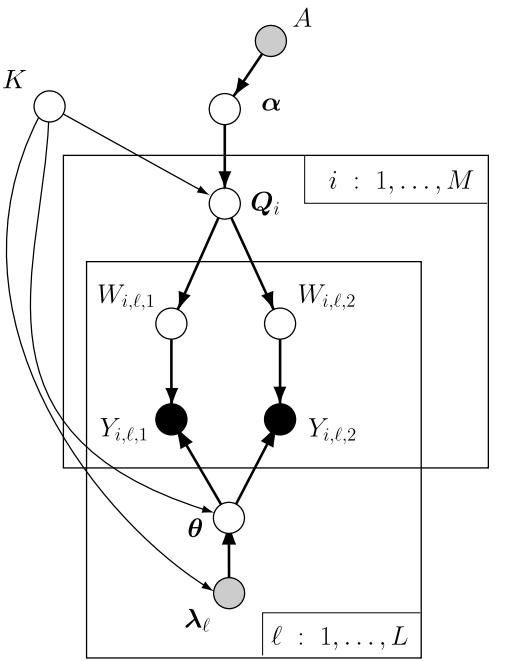
However, sometimes (as in the case of "cryptic" population structure) the number of subpopulations is not known.

Differing numbers of subpopulations correspond to different models, (*i.e.*, M_1 might be $K = 2$ and M_2 might be $K = 3$).

Ideally, we would like to compare models on the basis of their BF, $P(\boldsymbol{Y}|K = 2)/P(\boldsymbol{Y}|K = 3)$.

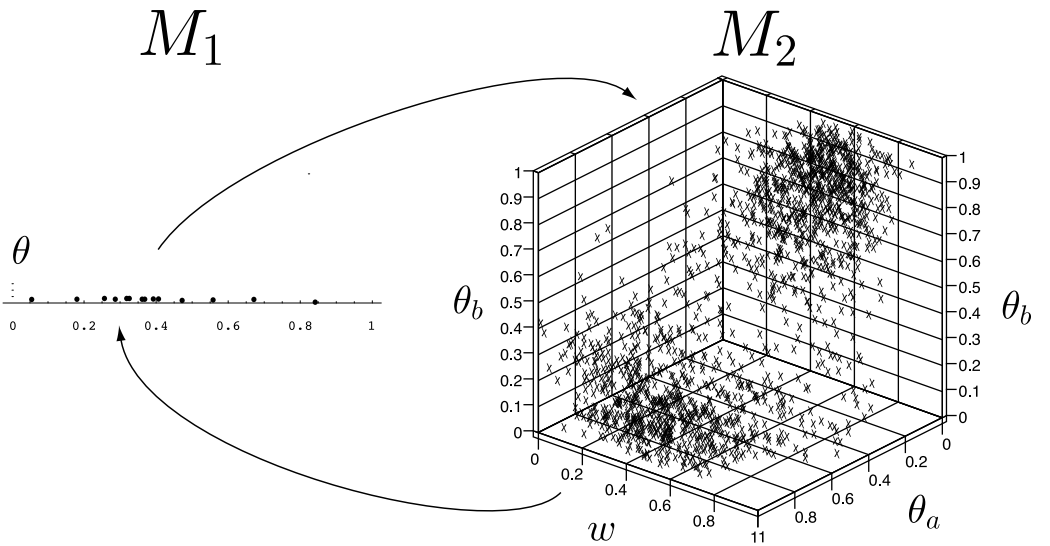
Computing BFs by sampling over K?:

If K could be included as another variable in the DAG representing the *structure* model, then we could propose Metropolis-Hastings updates to K, and estimate $P(K = i|\boldsymbol{Y})$ as the proportion of time the chain spends in states with $K = i$.



Traversing state spaces of differing dimensions—reversible jump MCMC:

- This is non-trivial because it involves jumping between spaces of different dimensions.
- Requires the “reversible jump” algorithm of Green (1995).



Unfortunately, in a model like that of *structure*, with numerous multiallelic loci, going from $K = 2$ to $K = 3$ increases the dimensionality dramatically, and RJMCMC is very difficult to apply successfully in such a situation.

The problem is that it is very hard to propose reasonable values of the allele frequencies for the "new" subpopulations that get added when K is increased.

How *structure* tries to solve the problem of choosing K:

Instead of computing the BF, *structure* introduced a different approach to model choice.

The approach is motivated by an attempt to "estimate" the normalizing constants that appear in the BF. As such the output of structure includes quantities that are labelled as estimates of $\log \Pr(Y|K)$.

However, the authors note that these estimates are unlikely to be accurate.

Selecting K by choosing K that maximises these estimates should be viewed as an *ad hoc* and computationally convenient approach to model choice, that performed well in simulations.

How *structure* tries to solve the problem of choosing K:

For notational convenience let's denote by Ω the latent variables $\boldsymbol{W}$, $\boldsymbol{Q}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, and denote all the observed data by Y.

1. First notice that from the law of conditional probability:

$$P(\Omega, Y|K) = P(Y|K, \Omega)P(\Omega|K)$$

   it follows that:

$$P(\Omega|K) = \frac{P(\Omega, Y|K)}{P(Y|K, \Omega)}$$

2. Then because probability distributions integrate to one:

$$1 = \int_{\Omega} P(\Omega|K)d\Omega = \int_{\Omega} \frac{P(\Omega, Y|K)}{P(Y|K, \Omega)}d\Omega$$

3. Then, write the numerator of the right side above as a product of densities, and rearrange:

$$1 = \int_{\Omega} \frac{P(\Omega|Y,K)P(Y|K)}{P(Y|K,\Omega)} d\Omega$$

implies

$$\frac{1}{P(Y|K)} = \int_{\Omega} \frac{P(\Omega|Y,K)}{P(Y|K,\Omega)} d\Omega = \int_{\Omega} \left(\frac{1}{P(Y|K,\Omega)} \right) P(\Omega|Y,K)$$

4. The above can be recognized as an expectation

$$\int_{\Omega} \left(\frac{1}{P(Y|K,\Omega)} \right) P(\Omega|Y,K) = \mathbb{E}\left[\frac{1}{P(Y|K,\Omega)} \middle| Y,K \right]$$

5. And so approximated by Monte Carlo:

$$\approx \frac{1}{n} \sum_{i=1}^{n} \frac{1}{P(Y|K,\Omega^{(i)})}$$

where $\Omega^{(i)}$ is simulated from its conditional distribution given Y and K. This is precisely the sort of sample that doing MCMC in *structure* at a fixed value of K gives you.

6. Unfortunately, this Monte Carlo estimator, known as the "harmonic mean" estimator, is notoriously unstable and may have infinite variance.
7. Instead, consider the random variable $D = -2 \log P(Y|\Omega, K)$ (the "deviance") which is a function of the random variable Ω. Assume that D (conditional on the data) is normally distributed (an "admittedly dubious" assumption).
8. Going back to our harmonic mean expectation, we can write it in terms of D:

$$\frac{1}{P(Y|K)} = \mathbb{E}\left[\frac{1}{P(Y|K,\Omega)}\middle| Y, K\right] = \mathbb{E}[e^{D/2}]$$

9. If D really is normally distributed then the right hand side of the above equation is just the moment generating function for a normal random variable (recall that $\mathbb{E}[e^{tX}] = e^{\mu t + \sigma^2 t^2/2}$).

10. So:

$$\frac{1}{P(Y|K)} = e^{\mu/2+\sigma^2/8} \Rightarrow -\log P(Y|K) = \mu/2 + \sigma^2/8$$

where $\mu = \mathbb{E}[D|Y]$ and $\sigma^2 = \text{Var}[D|Y]$.

11. Finally, it is easy to estimate μ and σ by Monte Carlo from the values of $D^{(i)}$ obtained when the chain samples from the values of $\Omega^{(i)}$.

Deviance Information Criterion (DIC):

Spiegelhalter et. al. (2002) have proposed an alternative method for selecting models based on MCMC output, called the "Deviance Information Criterion" (DIC).

In common with the *structure* approach, DIC also involves penalising the mean deviance, but with a different penalty.

Unpublished simulations from Carlos Bustamante suggest that DIC may perform better at selecting K than the current approach implemented in *structure*, and this approach may be implemented in future versions.

Thermodynamic Integration:

A 2016 paper in *Genetics*

Genetics: Early Online, published on June 17, 2016 as 10.1534/genetics.115.180992
GENETICS | **INVESTIGATION**

Estimating the number of subpopulations (K) in structured populations

Robert Verity[*,1] **and Richard A. Nichols**[†]
[*]MRC centre for outbreak analysis and modelling, Imperial College London, London, W2 1PG, UK, [†]Queen Mary University of London, London, E1 4NS, UK

ABSTRACT A key quantity in the analysis of structured populations is the parameter K, which describes the number of subpopulations that make up the total population. Inference of K ideally proceeds via the *model evidence*, which is

These authors show some oft-observed behavior of *structure*'s estimator of the $\log P(\text{Data}|K)$—the values keep climbing as K gets larger:

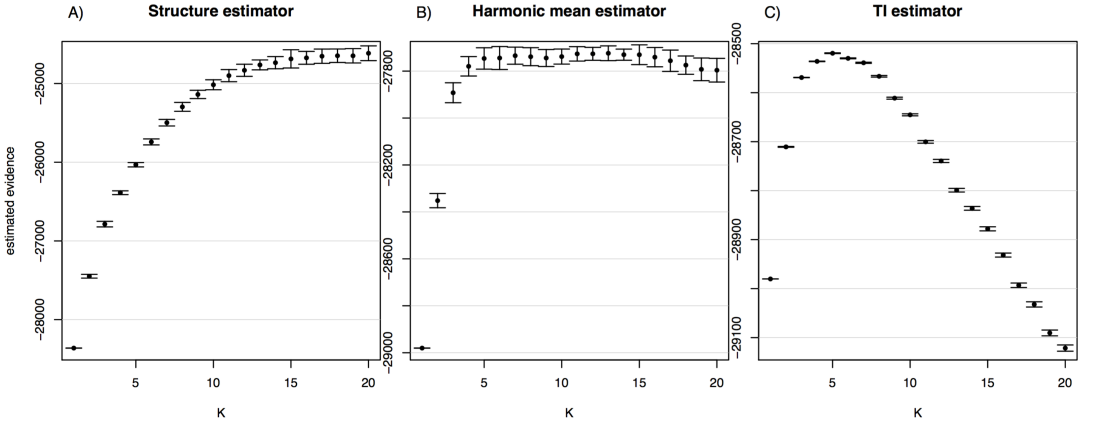


Figure 2 Estimates of the model evidence for $K = 1 : 20$ obtained using A) the STRUCTURE estimator L_K, B) the harmonic mean estimator $\hat{h}_K$, C) the TI estimator $\hat{T}_K$. For A) and B), solid points give the mean over 21 replicates and error bars give 95% confidence intervals calculated from the variance over replicates. For C) the TI estimation procedure results in a single point estimate of the evidence and an estimate of the 95% confidence interval without the need to average over replicates.

- And their thermodynamic integration approach seems to give a more stable, reasonable result (confirmed in simulations, as well).

ECA has prepared a short(-ish) R-notebook describing the thermodynamic inegration method. It is available at:

http://eriqande.github.io/sisg_mcmc_course/thermodynamic-integration.nb.html

Summary:

- Avoid drawing strong conclusions based solely on choice of K indicated by a *structure* analysis.
- Whenever you come across a Bayes Factor, ask yourself whether the priors used are sensible, and how they might influence the result.

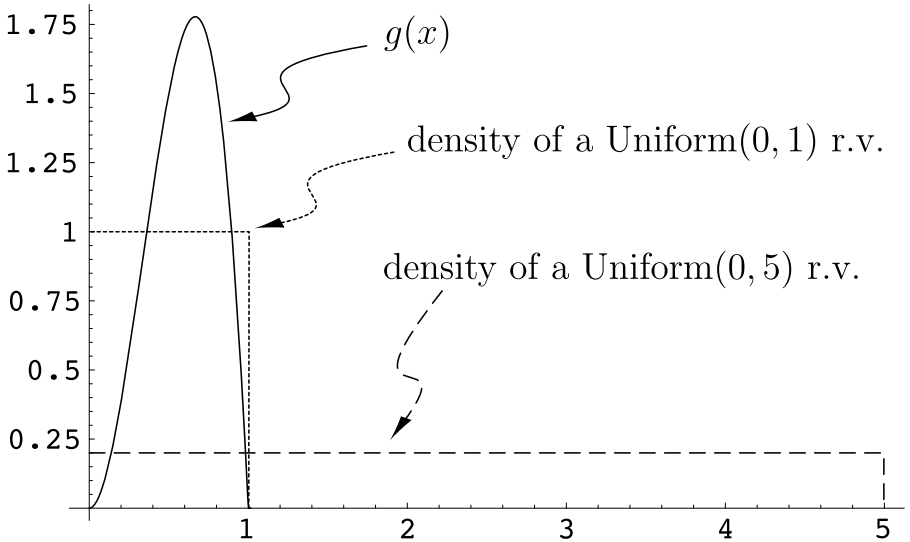
The Bottom Drawer—A Variety of Topics

The goals of this lecture are:

- Introduce *importance sampling*—a Monte Carlo variance reduction technique
- Explain how MCMC can be used to compute likelihood ratios
- Demonstrate MCMC in simple pedigrees and discuss problems of reducibility
- Discuss poor mixing more generally and describe Metropolis-coupled MCMC

The opposite of importance sampling—"minimal relevance" sampling:

Let's say you wanted to estimate the area under the curve $g(x)$ below by Monte Carlo:



Note that we are back to "vanilla" Monte Carlo (not MCMC) for the next five slides or so...

- We could simulate random variables $U \sim \text{Uniform}(0, 1)$

$$\int_0^1 g(x)dx = \int_0^1 g(x)1dx = \mathbb{E}[g(U)] \approx \frac{1}{n}\sum_{i=1}^{n} g(u^{(i)})$$

And that would work pretty well.

- We could also simulate random variables $W \sim \text{Uniform}(0, 5)$

$$\int_0^1 g(x)dx = 5\int_0^1 g(x)\frac{1}{5}dx = 5\mathbb{E}[g(W)] \approx \frac{5}{n}\sum_{i=1}^{n} g(w^{(i)})$$

- Each method is correct, but the second is clearly inefficient. Any values of $w^{(i)}$ greater than 1 don't provide us with any information about the area under $g(x)$!!

- The second method has larger Monte Carlo variance.

Importance sampling:

- Importance sampling is choosing a good distribution from which to simulate one's random variables for Monte Carlo.
- It involves multiplying the integrand by 1 (usually dressed up in a "tricky fashion") to yield an expectation of a quantity that varies less than the original integrand over the region of integration.
- For example, say we wish to approximate $\int_{x \in \mathcal{A}} g(x)dx$ and we can "dream up" a distribution $h(x)$.

$$\int_{x \in \mathcal{A}} g(x)dx = \int_{x \in \mathcal{A}} g(x)\frac{h(x)}{h(x)}dx = \int_{x \in \mathcal{A}} \frac{g(x)}{h(x)}h(x)dx$$

which is

$$\mathbb{E}_h \left[\frac{g(X)}{h(X)} \right]$$

where $\mathbb{E}_h$ denotes the expectation with respect to the density h, so long as $h(x) > 0$ for all values x for which $g(x) \neq 0$.

So we could approximate $\int_{x \in \mathcal{A}} g(x)dx$ by:

$$G_n^h = \frac{1}{n} \sum_{i=1}^{n} \frac{g(x^{(i)})}{h(x^{(i)})}$$

where each $x^{(i)}$ is simulated from the density h.

- Using the Cauchy-Schwarz inequality, it is not too difficult to show that the Monte Carlo variance of G_n^h is minimized when $h(x) \propto |g(x)|$.
- Even more straightforward, if we consider only the case where $g(x) > 0$, it is immediately apparent that the Monte Carlo variance is minimized (is zero, in fact) when $h(x) \propto g(x)$.
- Of course, if you could simulate independent r.v.'s from a distribution exactly proportional to $g(x)$, that implies you know the value of the integral anyway. So this is a little circular—but it does emphasize that importance sampling works best when $h(x)$ is close to proportional to $g(x)$.

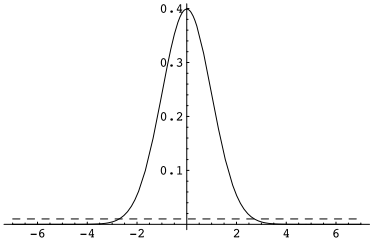
Choosing a good importance sampling distribution:

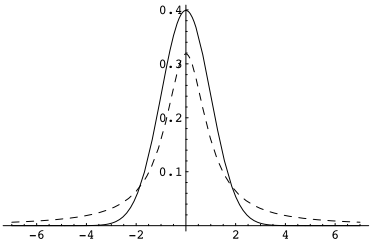
A good importance sampling function $h(x)$ should have the following properties:

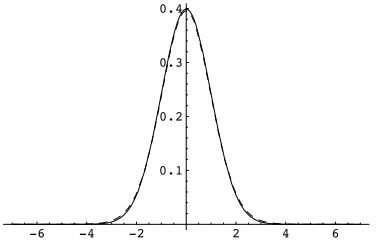
1. $h(x) > 0$ whenever $g(x) \neq 0$
2. $h(x)$ should be close to being proportional to $|g(x)|$
3. it should be easy to simulate values from $h(x)$
4. it should be easy to compute the density $h(x)$ for any value x that you might realize.

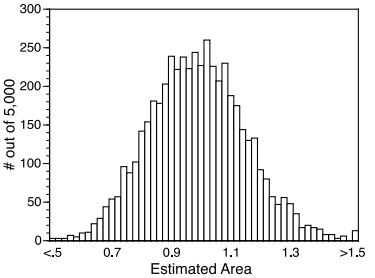
Fulfilling this wish-list in high dimensional space (where Monte Carlo techniques are most useful) is quite often a tall task.

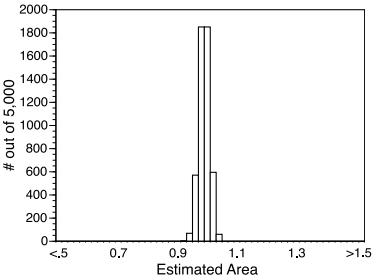
Simple example: Estimating the area under a Normal(0, 1) curve with $n = 1,000$:

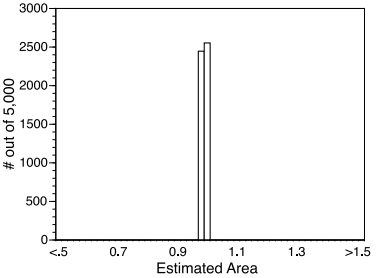












$h(x) \equiv$ Uniform $h(x) \equiv t_1$ Dist. $h(x) \equiv t_{30}$ Dist.

Example of importance sampling in genetics:

- Griffiths and Tavare (1994) developed a “recursion” method for simulating genealogies under the coalescent model conditional on genetic data at the tips of the branches (but not *exactly* from their distribution given the data).
- This method simulates independent genealogies that can be used in Monte Carlo.
- It wasn’t until somewhat later that Joe Felsenstein and his colleagues were able to dissect the method and show that it is an example of importance sampling.
- Viewing the method as an importance sampling task, Matt Stephens and Peter Donnelly (2000) were able to improve upon the method considerably.

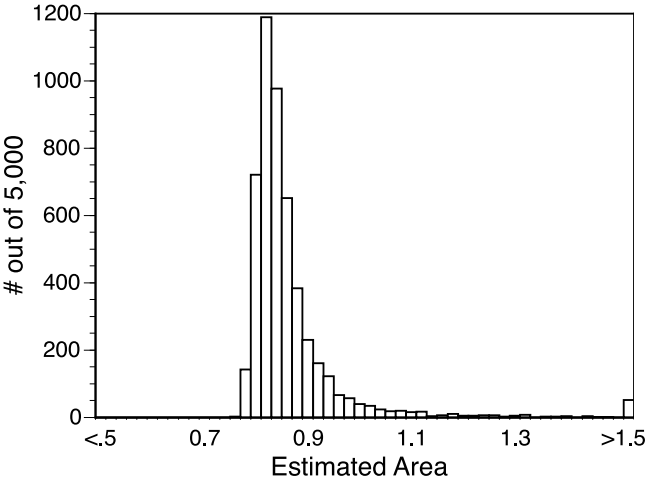
A common pitfall of importance sampling:

- Whatever is going on in the "tails" of the distribution is, unfortunately, very important...
- If $h(x)$ is thinner-tailed than $g(x)$ there could be problems: if for a rarely-realized value $x^{(i)}$, $h(x^{(i)})$ is very small, but $g(x^{(i)})$ is not terribly small, then $g(x^{(i)})/h(x^{(i)})$ could be much larger than typical ratios of $g(x)/h(x)$, leading to an unstable estimator.
- Consequently, assessing the convergence of importance sampling estimators can, at times, be quite difficult.

An example of a problematic thin-tailed importance sampling distribution:

Approximate the area under a thick-tailed Cauchy (t_1) distribution using a thin-tailed $\text{Normal}(0, 1)$ density as the importance sampling distribution, and $n = 1,000$:

The distribution of Monte Carlo estimates appears to the right. The correct area is 1.0. Notice the number of estimates that are greater than 1.5.



MCMC in a frequentist framework—not just for Bayesians anymore:

Geyer and Thompson (1992) use an importance sampling method to compute likelihood ratios using MCMC.

- The basic inference problem is to compute the relative likelihood for different values of θ given data $\boldsymbol{Y}$, but the likelihood $P_\theta(\boldsymbol{Y})$ is an intractable sum over latent variables $\boldsymbol{X}$:

$$P_\theta(\boldsymbol{Y}) = \sum_{\boldsymbol{X}} P_\theta(\boldsymbol{Y}, \boldsymbol{X})$$

- We wish to be able to compute $P_\theta(\boldsymbol{Y})/P_{\theta_0}(\boldsymbol{Y})$—the ratio between the likelihood for some arbitrary θ and the likelihood for a fixed θ_0.
- Geyer and Thompson found a tricky way of multiplying the above equation by 1 to get a useful result:

$$P_\theta(\boldsymbol{Y}) = \sum_{\boldsymbol{X}} P_\theta(\boldsymbol{Y},\boldsymbol{X})\frac{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X})}{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X})}$$

$$= \sum_{\boldsymbol{X}} P_\theta(\boldsymbol{Y},\boldsymbol{X})\frac{P_{\theta_0}(\boldsymbol{X}|\boldsymbol{Y})P_{\theta_0}(\boldsymbol{Y})}{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X})}$$

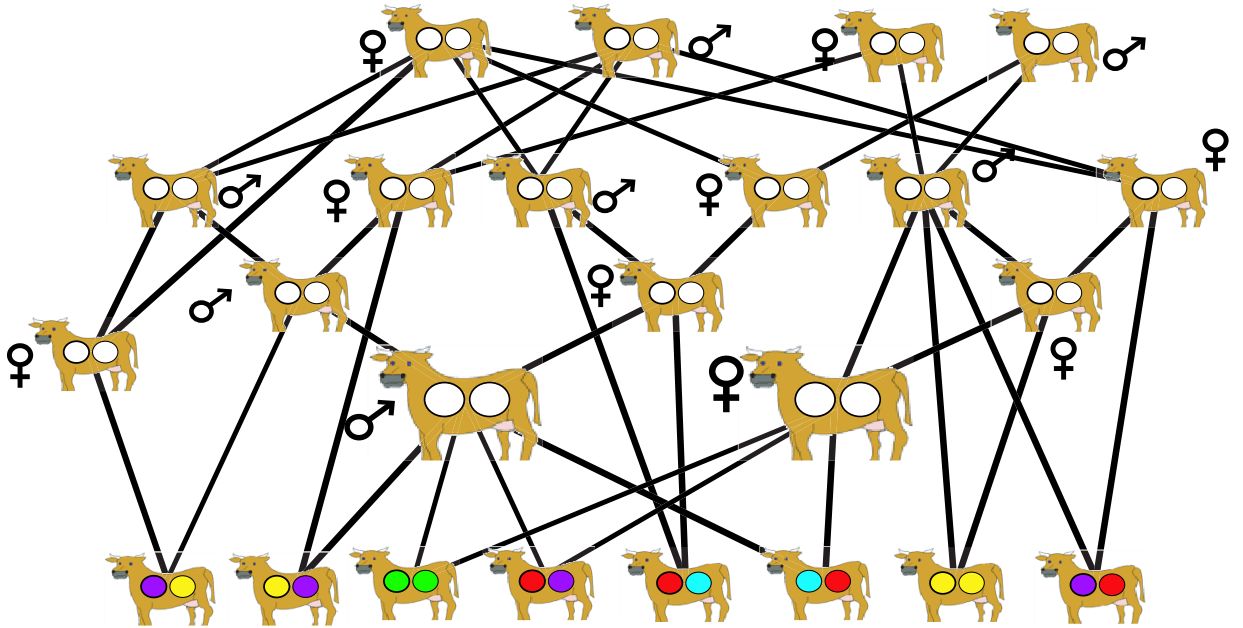
So,

$$\frac{P_\theta(\boldsymbol{Y})}{P_{\theta_0}(\boldsymbol{Y})} = \sum_{\boldsymbol{X}} \frac{P_\theta(\boldsymbol{Y},\boldsymbol{X})}{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X})} P_{\theta_0}(\boldsymbol{X}|\boldsymbol{Y}) = \mathbb{E}_{\theta_0}\left[\frac{P_\theta(\boldsymbol{Y},\boldsymbol{X})}{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X})}\middle| Y\right]$$

$$\frac{P_\theta(\boldsymbol{Y})}{P_{\theta_0}(\boldsymbol{Y})} \approx \frac{1}{n}\sum_{i=1}^{n} \frac{P_\theta(\boldsymbol{Y},\boldsymbol{X}^{(i)})}{P_{\theta_0}(\boldsymbol{Y},\boldsymbol{X}^{(i)})} \quad \text{with} \quad \boldsymbol{X}^{(i)} \sim P_{\theta_0}(\boldsymbol{X}|\boldsymbol{Y})$$

and $P_{\theta_0}(\boldsymbol{X}|\boldsymbol{Y})$ is a distribution known only up to scale, so it may be simulated from via MCMC.

Switching gears—a simple example of MCMC on a pedigree with "data at the bottom":



The colored balls represent observed allelic types at a locus.

Asking questions about the ancestors:

Simple questions:

- What is the probability that only n copies of the red allele occurred in the founders?
- What is the probability that ancestor Z carried exactly one copy of the purple allele?

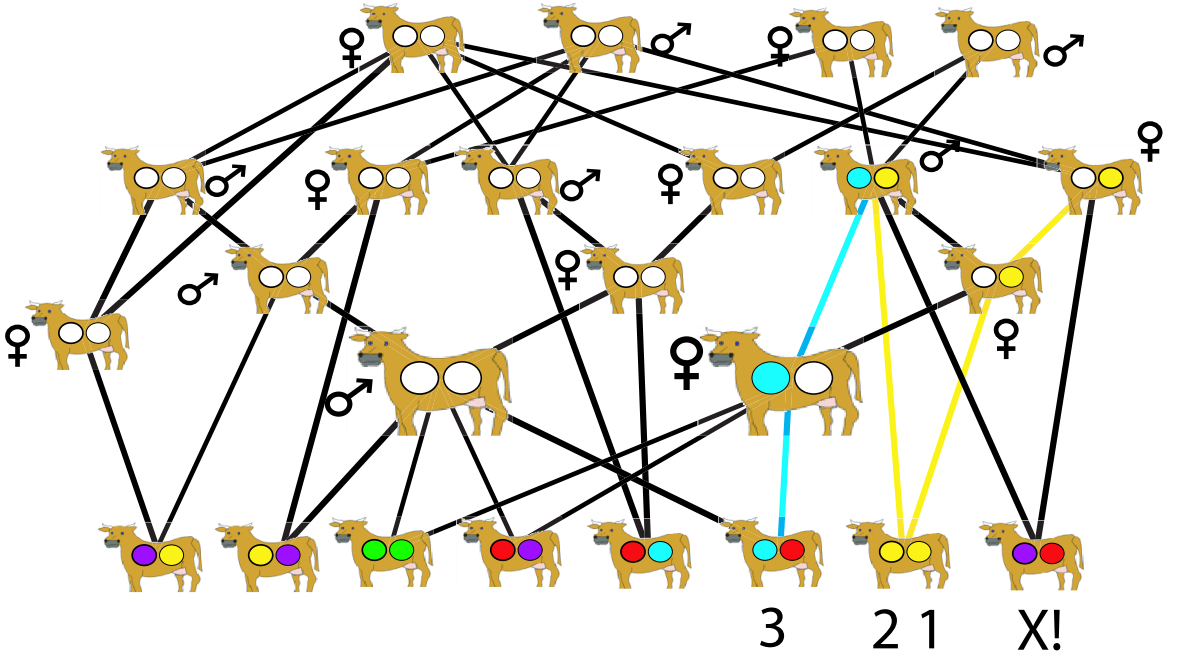
More complex questions:

- What is the probability of the observed data at two loci, given that the recombination fraction between those loci is θ?

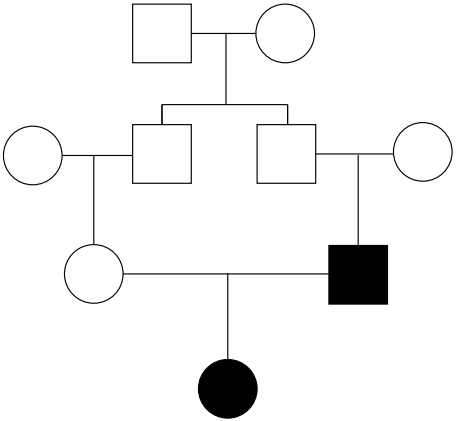
All such questions can be written as a sum over the unobserved (latent) genotypes of the ancestors, and hence they can be expressed as expectations and approximated by Monte Carlo.

Simulating ancestral genotypes naively will fail:

You can't just "flip-a-coin" your way up the pedigree the way that Wright and McPhee (1925) could to estimate inbreeding:

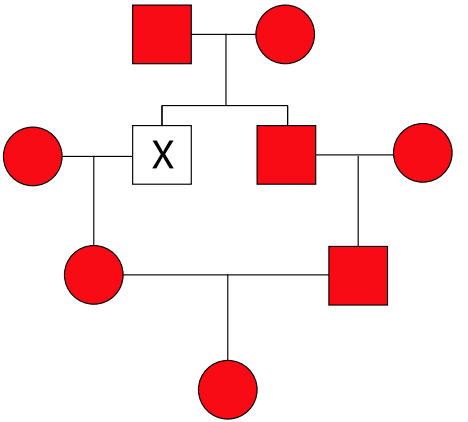


Another graph-like notation—pedigrees:



Males are squares. Females are circles. Black nodes mean they have observed data (*i.e.*,they are part of $\boldsymbol{Y}$).

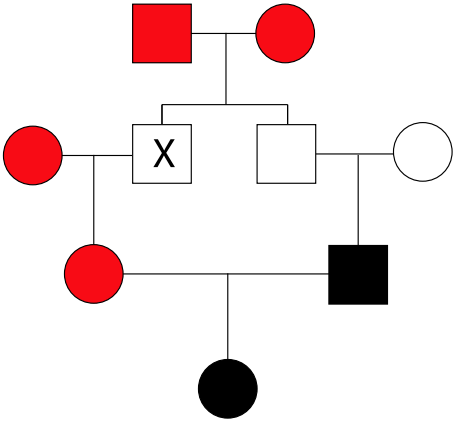
The full conditional distribution for an individual's genotype at a locus. . . :



(Red nodes represent the elements that are conditioned on to determine the full conditional for the node labeled "X".)

. . . depends only on its neighbors. . .

These are the people in your neighborhood:



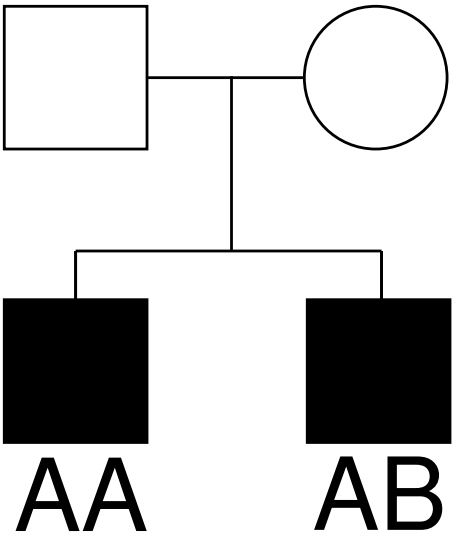
Parents, children, and spouses, (the red ones) are neighbors of node "X".

So assuming you can find a valid starting configuration, you can buzz around and do Gibbs updates to each individual's genotypes, and thus simulate the latent ancestor genotypes ($\boldsymbol{X}$) conditional on the observed data $\boldsymbol{Y}$.

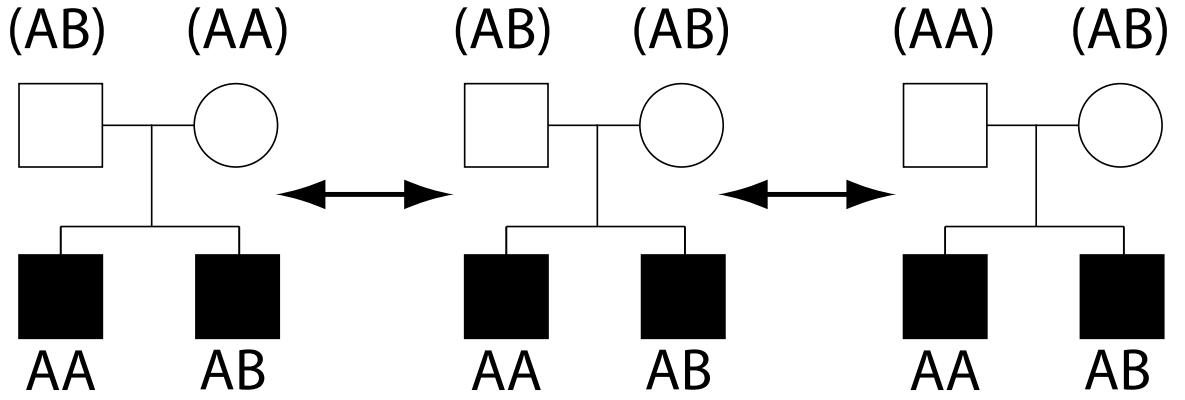
Irreducibility of single-site Gibbs sampler with two alleles:

Demonstrating this principle on an even simpler pedigree,

If the data look like:

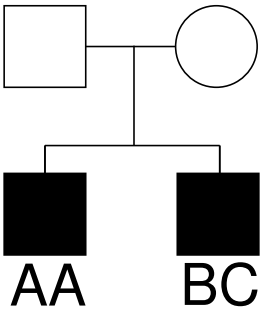


Then there are only three possible configurations of genotypes in the unshaded individuals, and these states are reachable from one another in a finite number of steps of single-site Gibbs updating. Hence this sampler is irreducible here.

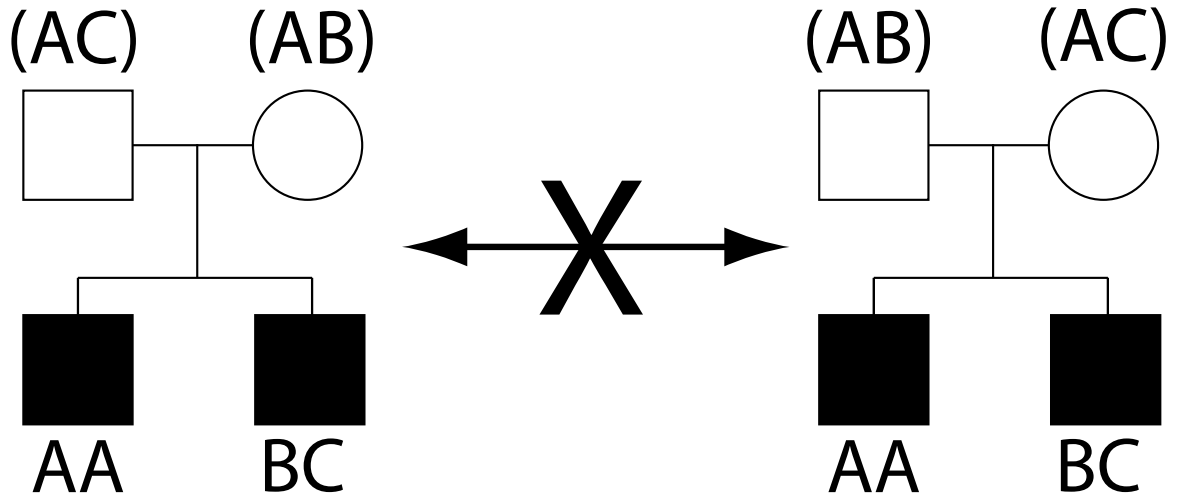


With multiple alleles, however, the single-site Gibbs sampler is reducible:

For example, with data like. . .



. . . There are only two possible states for the genotypes of the unshaded individuals and they do not communicate:



So, the single-site Gibbs sampler is not irreducible for most pedigree problems of consequence. It leads to a reducible Markov chain.

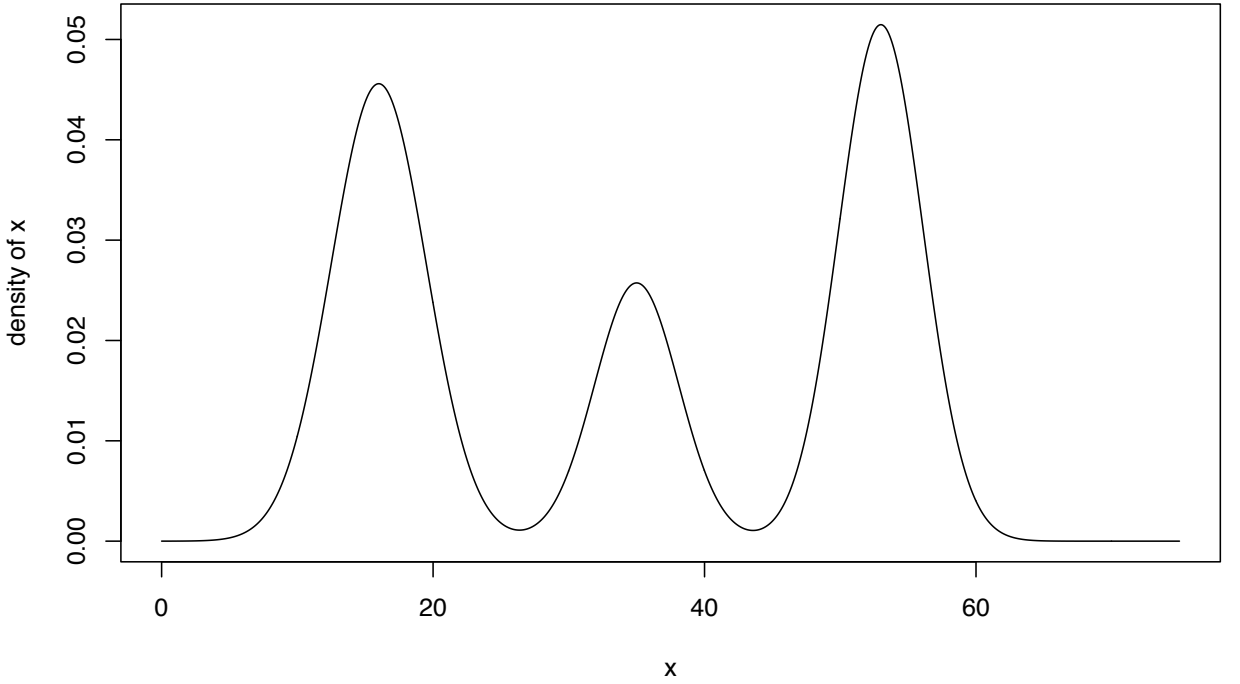
Irreducibility ain't the only thing:

While irreducibility is essential for MCMC, it is often possible to have an irreducible sampler, but have *poor mixing*.

i.e., you may be able to show theoretically that your sampler is irreducible, but *practically* it may not reach all areas of high probability.

Poor mixing means that the chain does not adequately sample all regions of the space.

Consider the following one-dimensional example—a mixture of three normal densities:



Computer Demo

Designing samplers to mix well:

This is often the hardest challenge in MCMC, especially in high dimensional space with bumpy or multimodal likelihood surfaces.

Obvious improvements could come from:

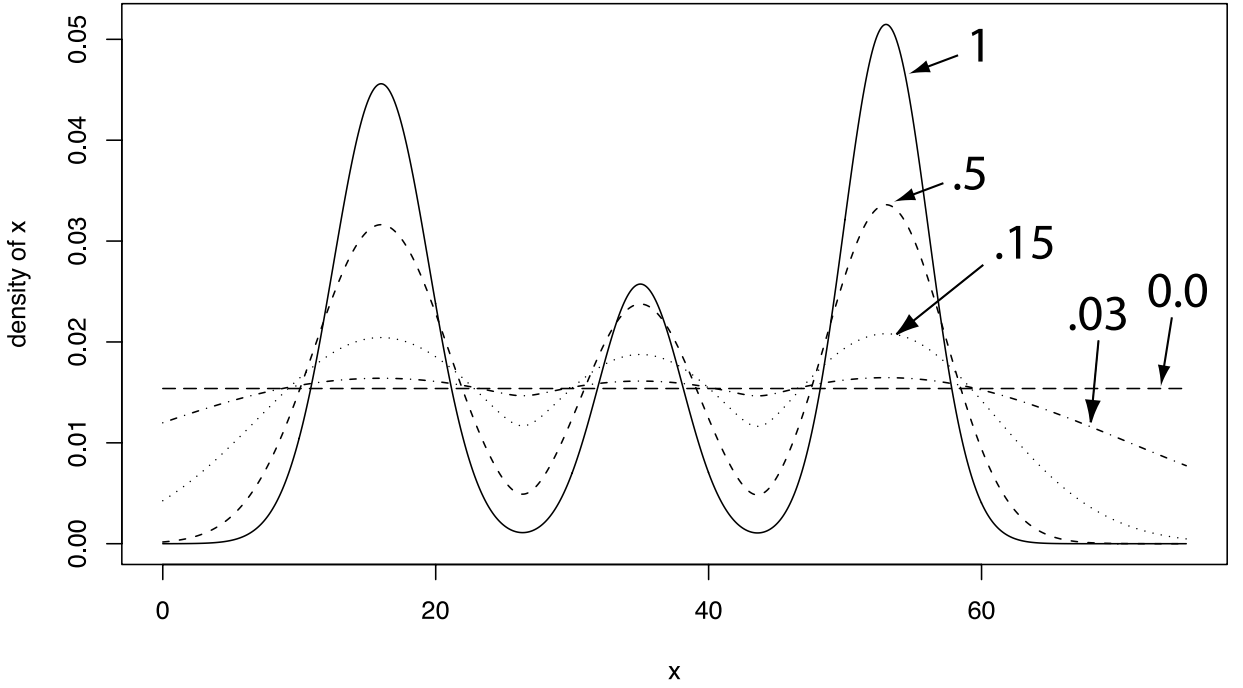
- Tweaking the proposal distribution (but note that if the variance of the proposal distibution is very high, the acceptance rate may $\downarrow$)
- Updating several variables jointly. In Gibbs sampling, in particular, this can be useful. (However, the computations necessary may require considerable extra time).

One general method for improving mixing is now called *Metropolis-coupled* MCMC (Geyer 1991).

This is used extensively in the Bayesian phylogenetic analysis package MR BAYES.

Metropolis-coupled MCMC:

MCMCMC takes advantage of the fact that by raising a probability density to a power β between zero and one, the relative heights of peaks, and the relative depths of valleys, may be made less extreme:



Recipe for Metropolis-coupled MCMC:

- Let X denote all the different variables being sampled in the MCMC
- Run one chain, labelled 1, as you normally would with limiting distribution $f(X)$ being the distribution you wish to sample from. Note: $\beta_1 = 1$.
- Run $n-1$ other chains (chains $2, \ldots, n$) in the same way, except run them so that the limiting distributions are "flattened out" versions of f. *i.e.*, the i^{th} chain ($i = 2, \ldots, n$) uses a Hastings ratio that looks like:

$$\frac{q(X|X')}{q(X'|X)} \frac{[f(X')]^{\beta_i}}{[f(X)]^{\beta_i}}$$

  where each β_i is between 0 and 1. These chains are often called the "hot" chains, because they move around more easily—much like a heated molecule.
- In addition to the normal MCMC updates, also perform "chain-swapping" updates:

1. Choose a pair of chains i and j ($1 \leq i < j \leq n$) at random and propose swapping their current states ($X_{[i]}$ and $X_{[j]}$, respectively).
2. Accept or reject the proposed swap using the Hastings ratio:

$$\alpha = \min \left\{ 1, \frac{[f(X_{[i]})]^{\beta_j}}{[f(X_{[i]})]^{\beta_i}} \frac{[f(X_{[j]})]^{\beta_i}}{[f(X_{[j]})]^{\beta_j}} \right\}$$

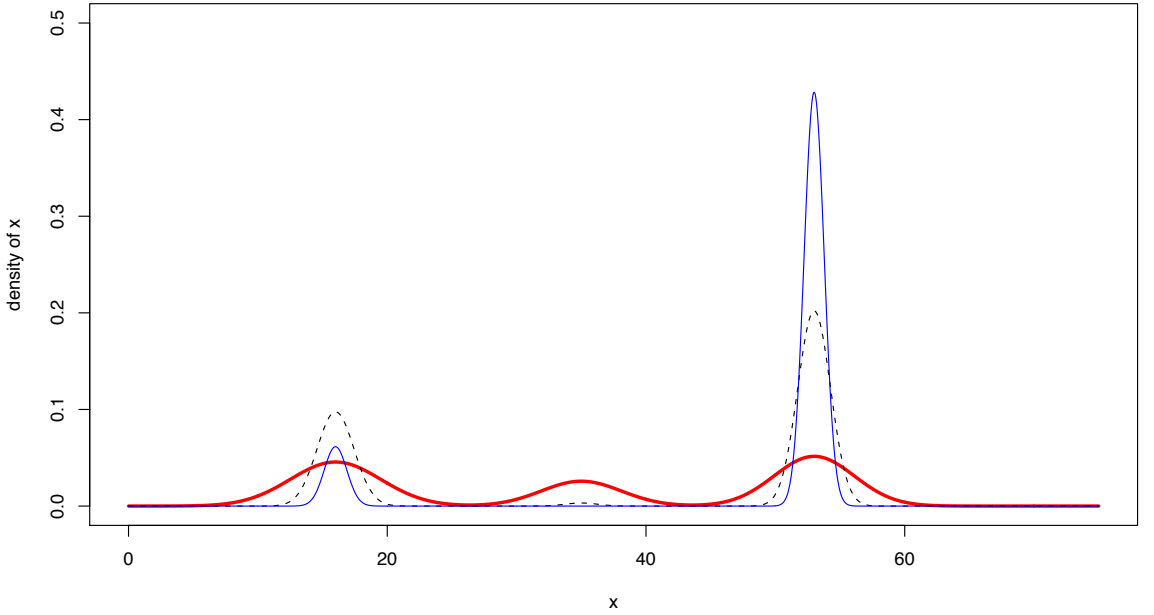
- The Monte Carlo sample itself is *only drawn from chain* 1!
- Derivation of the Hastings ratio is easy: given i and j and the current states $X_{[i]}$ and $X_{[j]}$, the proposal distribution is symmetrical, so those terms cancel out. The bits that remain are just the ratio of limiting distributions of the proposed *vs.* the original states.
- Computationally, you don't have to swap the states (which could mean copying a lot of memory around). Instead you just swap β's.

Computer Demo: `mc3demo`

Simulated Annealing:

• Simulated annealing is an optimization technique that relies on "tricks" like those employed in Metropolis-coupled MCMC

• Consider what happens if $\beta > 1$. (Thick red line is $\beta = 1$. Dotted black line is $\beta = 6$. Thin solid blue line is $\beta = 16$.)



Recipe for simple simulated annealing:

Imagine $f(x)$ is an objective function you wish to maximize. That is, you want to find the value of x that maximizes $f(x)$. To do so by simulated annealing you...

- Imagine that $f(x)$ is an unnormalized probability density or mass function
- Design an M-H sampler to sample x's from target distribution $f(x)$
- Run a single chain using this M-H sampler while raising $f(x)$ in the M-H ratio to the power of β:

$$\frac{q(X|X')}{q(X'|X)} \frac{[f(X')]^{\beta}}{[f(X)]^{\beta}}$$

- Start with β small (less than 1, typically), then progressively make β larger.

- When β is quite large and the values of x are changing little each step, or you've run out of computer time, call the current value of x the optimum.
- This is not guaranteed to be a global maximum.
- In this simple case, there are not multiple chains being run.
- In simulated annealing, you are not approximating an expectation, but you are using the machinery of MCMC to tackle an optimization problem.

References

Anderson, E. C. and Thompson, E. A. (2002). A model-based method for identifying species hybrids using multilocus genetic data, *Genetics*, **160**, 1217–1229.

Brooks, S. and Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics.*

Clark A.G. (1990). Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, **7**, 111–122.

Falush, D., Stephens, M. and Pritchard, J. K. (2003) Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies, *Genetics*, **164**, 1567-1587.

Feller, W. (1957). *An Introduction to Probability Theory and Its Applications*, 2nd Edition. New York, John Wiley & Sons.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741.

Gelman, A. and Meng, X.-L. (1998). Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, **13**, 163–185.

Gelman, A. and Rubin, D. (1992). Inference from iterative simulation using multiple sequences, *Statistical Science* **7**, 457–511.

Gelman, A., Carlin, J.B., Stern, H.B. and Rubin D.B. (2004). *Bayesian Data Analysis*, (2nd Ed). New York: Chapman & Hall.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In J. M. Bernardo,

A. F. M. Smith, A. P. Dawid and J. O. Berger (eds.), *Bayesian Statistics 4*, Oxford University Press.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface,* Interface Foundation, Fairfax Station, 156–163.

Geyer, C. J. and Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data (with discussion). *Journal of the Royal Statistical Society, Series B*, **54**, 657–699.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, **41**, 337–48.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika*, **82**, 711–732.

Griffiths, R. C. and Tavare, S. (1994). Simulating probability distribu-

tions in the coalescent. *Theoretical Population Biology*, **46**, 131–159.

Guo, S. W. and Thompson, E. A. (1992). Performing the exact test of Hardy-Weinberg proportion for multiple alleles. *Biometrics*, **48**, 361–372.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109.

Heidelberger, P. and Welch, P.D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research* **31**, 1109–1144.

Kass, R.E. and Raftery, A.E. (1995). Bayes factors. *Journal of the American Statistical Association*, **90**, 773–795.

Levene, H. (1949). On a matching problem arising in genetics. *The Annals of Mathematical Statistics*, **20**, 91–94.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and

Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics* **21**, 1087–1092.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, **44**, 335–341.

Pritchard, J. K., Stephens, M. and Donnelly, P. (2000). Inference of population structure using multilocus genotype data, *Genetics*, **155**, 945–959.

Raftery, A. E. and Lewis, S.M. (1992). How Many Iterations in the Gibbs Sampler? In J. M. Bernardo, A. F. M. Smith, A. P. Dawid and J. O. Berger (eds.), *Bayesian Statistics 4*, Oxford University Press.

Ripley, B. D. (1987). *Stochastic Simulation*, New York: Wiley & Sons.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2), 461-464.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. and van der Linde, A.

(2002). Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society*, Series B, **64**,4,583–640.

Stephens, M. and Donnelly, P. (2000). Inference in molecular population genetics (with Discussion), *Journal of the Royal Statistical Society, Series B*, **62**, 605–655.

Wilson, G A and Rannala, B (2003). Bayesian inference of recent migration rates using multilocus genotypes, *Genetics*, **163**, 1177–1191.

Wright, S. and McPhee, H. C. (1925) An approximate method of calculating coefficients of inbreeding and relationship from livestock pedigrees. *Journal of Agricultural Research*, **31**, 377–383

Yu, B. and Mykland, P. A. (1998). Looking at Markov samplers through cusum path plots: a simple diagnostic idea. *Statistics and Computing*, **8** 275–286.