

統計数理研究所 公開講座

「マルコフ連鎖モンテカルロ法の基礎と実践」の投影資料

(久保担当部分)

2. ベイジアンモデリングとMCMC

3. R と WinBUGS の使いかた

久保拓弥 kubo@ees.hokudai.ac.jp

<http://goo.gl/Yvz50>

- 今日の投影資料は配布資料版を少し修正したものです

配布版と同じ・ほとんど同じ

ちょっと変更したページ

追加ページ (4 ページ)

- 最新版 PDF ファイルは
<http://goo.gl/Yvz50>
からダウンロードできます

久保担当部分で説明しようとする事

- 直線回帰などといった **統計モデル** あてはめと MCMC はどう関係するんだろう?
- 階層ベイズモデルと MCMC の関係は?
 - そもそも **階層ベイズモデル** とは?
- 階層ベイズモデルは, **R** や **WinBUGS** といった誰でも簡単に入手できるソフトウェアであつかえる

観測データを定量的に説明する統計モデル

- 統計モデルは観測データのパターンをうまく**説明**できるようなモデル
- 基本的部品: **確率分布** (とそのパラメーター)
- データにもとづくパラメーター推定, **あてはまりの良さ**を定量的に評価できる

たとえば「結果 ← 原因」関係を一番単純に表現している**線形モデル**という統計モデルについて.....

「結果 ← 原因」関係を表現する線形モデル

- 結果: 応答変数
- 原因: 説明変数
- 線形予測子 (linear predictor):

(応答変数の平均) = 定数 (切片)

(あるいは応答変数の平均の関数)

+ (係数 1) × (説明変数 1)

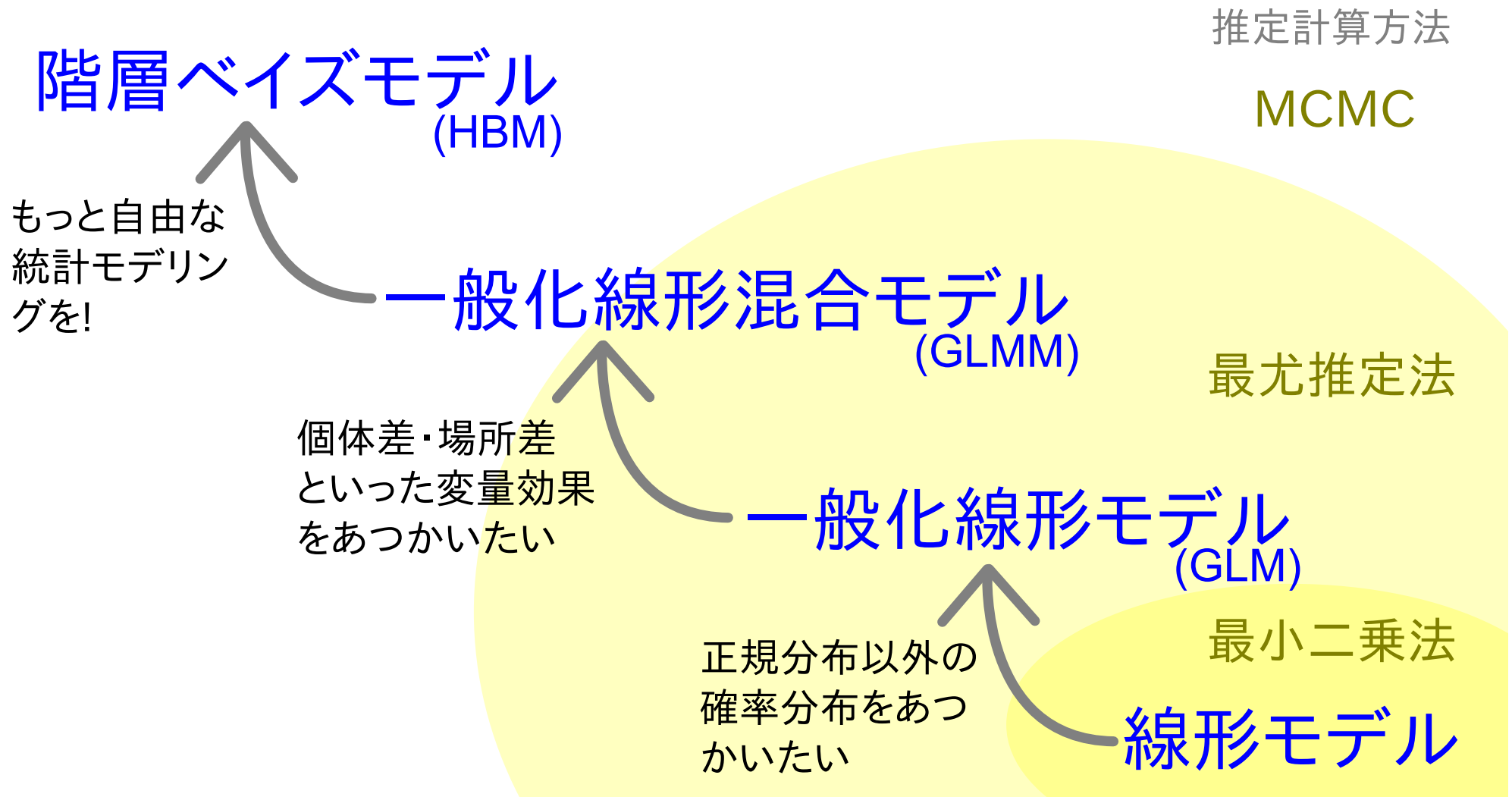
+ (係数 2) × (説明変数 2)

+ (係数 3) × (説明変数 3)

+ ...

(じつは今日のハナシは「説明変数」ないんですが.....)

線形モデルの発展



今日の話: ベイズモデルを WinBUGS で

2. ベイジアンモデリングと MCMC

階層ベイズモデルとは何か?

MCMC とどういう関係にあるのか?

3. R と WinBUGS の使いかた

階層ベイズモデルを推定するソフトウェアは?

WinBUGS はどうやって使うか?

2. ベイジアンモデリング と MCMC

このあとの統計モデル化の説明の手順

1. 簡単な例題: GLM でうまくいく場合

- 統計モデルの部品: 二項分布モデル (GLM)
- 統計モデルの推定方法: 最尤推定法

2. MCMC とベイズモデリング

- 最尤推定を MCMC におきかえてみる
- MCMC で得られた結果をベイズ的に解釈

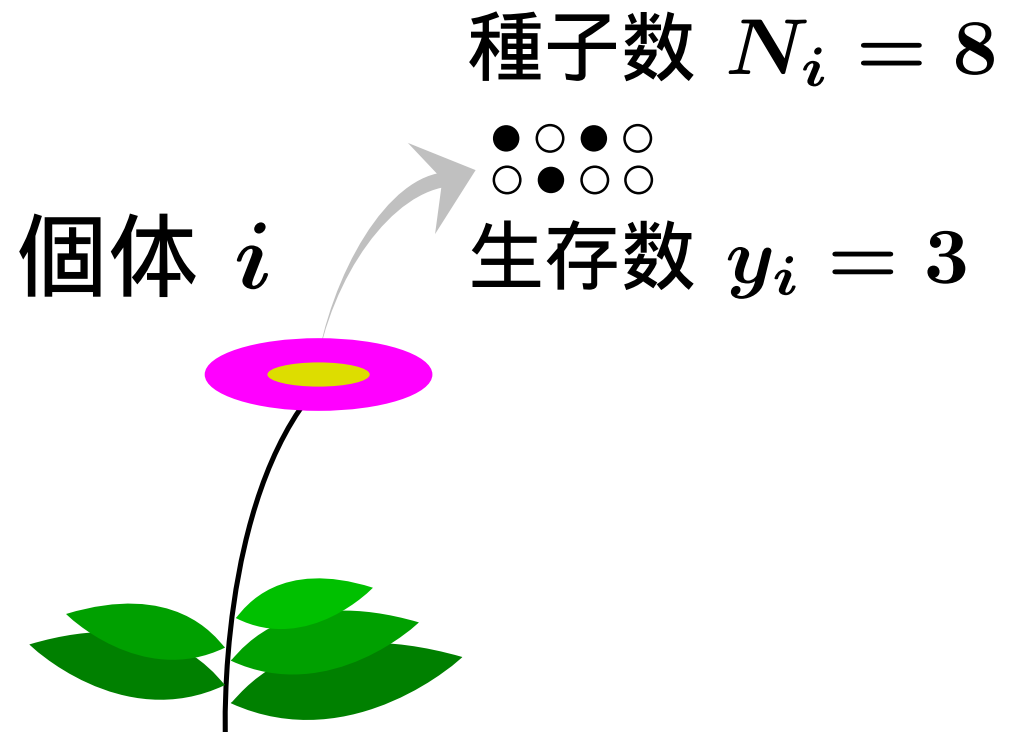
3. ちょっと難しい例題: GLM でうまくいかない場合

- 「個体全体の平均」と「個体差」をどうあつかう?
- 階層ベイズモデル!

本日の例題: 植物の種子と二項分布

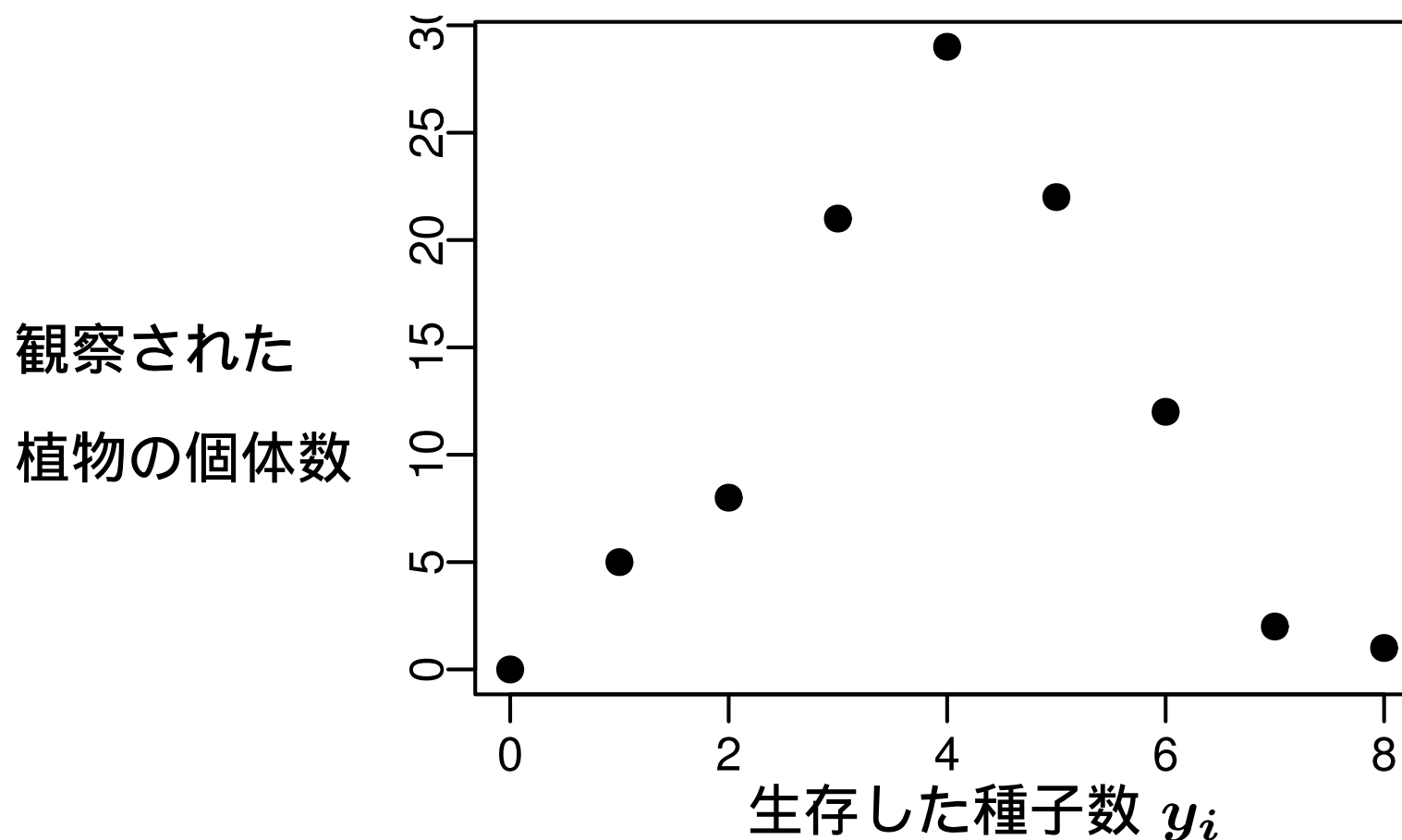
繁殖生態学の例題: 架空植物の生存確率

- 架空植物の種子の生存を調べた
- 種子: 生きていれば発芽する
 - どの個体でも **8 個** の種子を調べた
- 生存確率: ある種子が生きている確率
- データ: 植物 100 個体, 合計 800 種子の生存の有無を調べた
- 問: この植物の生存確率はどのように統計モデル化できるか?



簡単な例題: 生存確率は全個体で同じ (「個体差」なし)

個体ごとの生存数	0	1	2	3	4	5	6	7	8
観察された個体数	0	5	8	21	29	22	12	2	1



生存確率 q と二項分布の関係

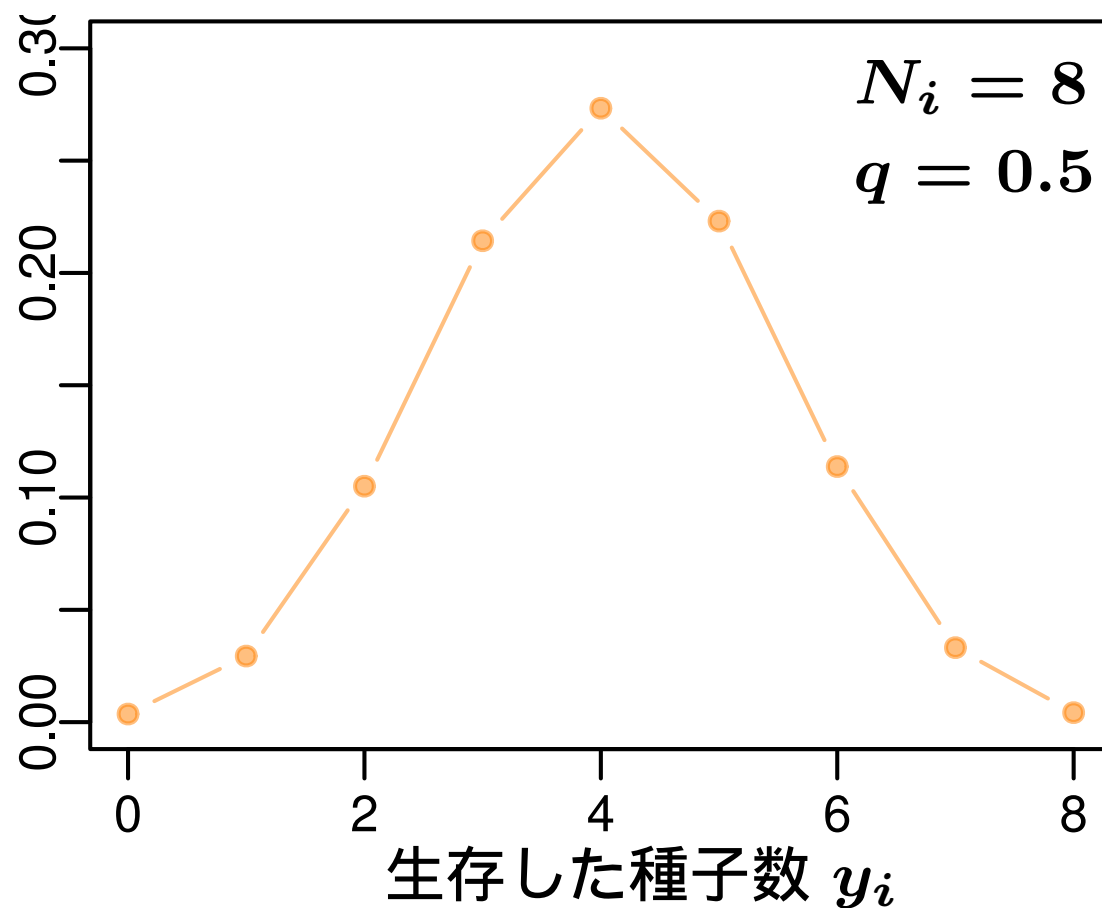
- 生存確率を推定するために **二項分布** という確率分布を使う
- 個体 i の N_i 種子中 y_i 個が生存する確率は二項分布

$$f(y_i | q) = \binom{N_i}{y_i} q^{y_i} (1 - q)^{N_i - y_i},$$

- ここで仮定していること
 - **個体差はない**
 - つまり **すべての個体で同じ生存確率 q**

二項分布で「 N_i 個中の y_i 個」型データをあつかう

$$f(y_i | q) = \binom{N_i}{y_i} q^{y_i} (1 - q)^{N_i - y_i},$$



尤度: 100 個体ぶんのデータが観察される確率

- 観察データ $\{y_i\}$ が与えられたもので, パラメータ q は値が自由にとりうると考える
- この 100 個体ぶんの確率はパラメータ q の関数として定義される

$$L(q | \text{全 } y_i) = \prod_{i=1}^{100} f(y_i | q)$$

個体ごとの生存数	0	1	2	3	4	5	6	7	8
観察された個体数	0	5	8	21	29	22	12	2	1

対数尤度方程式と最尤推定

- この尤度 $L(q \mid \text{データ})$ を最大化するパラメータの推定量 \hat{q} を計算したい
- 尤度を対数尤度になおすと

$$\begin{aligned} \log L(q \mid \text{データ}) &= \sum_{i=1}^{100} \log \binom{N_i}{y_i} \\ &+ \sum_{i=1}^{100} \{y_i \log(q) + (N_i - y_i) \log(1 - q)\} \end{aligned}$$

- この対数尤度を最大化するように未知パラメーター q の値を決めてやるのが**最尤推定**

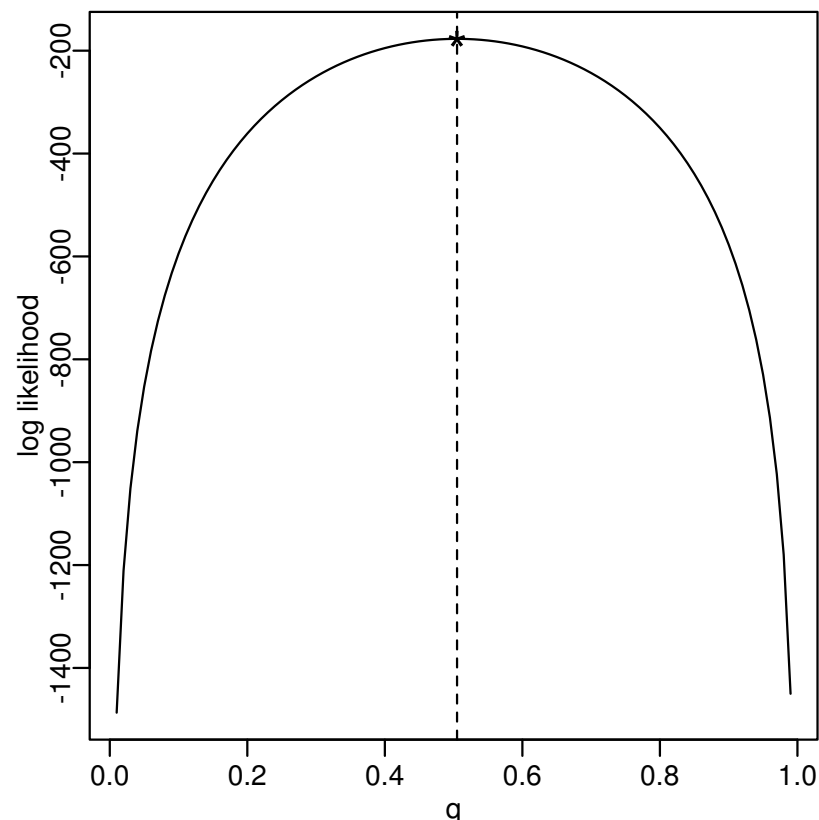
最尤推定とは何か

- 対数尤度 $L(q \mid \text{データ})$ が最大になるパラメーター q の値をさがしだすこと
- 対数尤度 $\log L(q \mid \text{データ})$ を q で偏微分して 0 となる \hat{q} が対数尤度最大

$$\frac{\partial \log L(q \mid \text{データ})}{\partial q} = 0$$

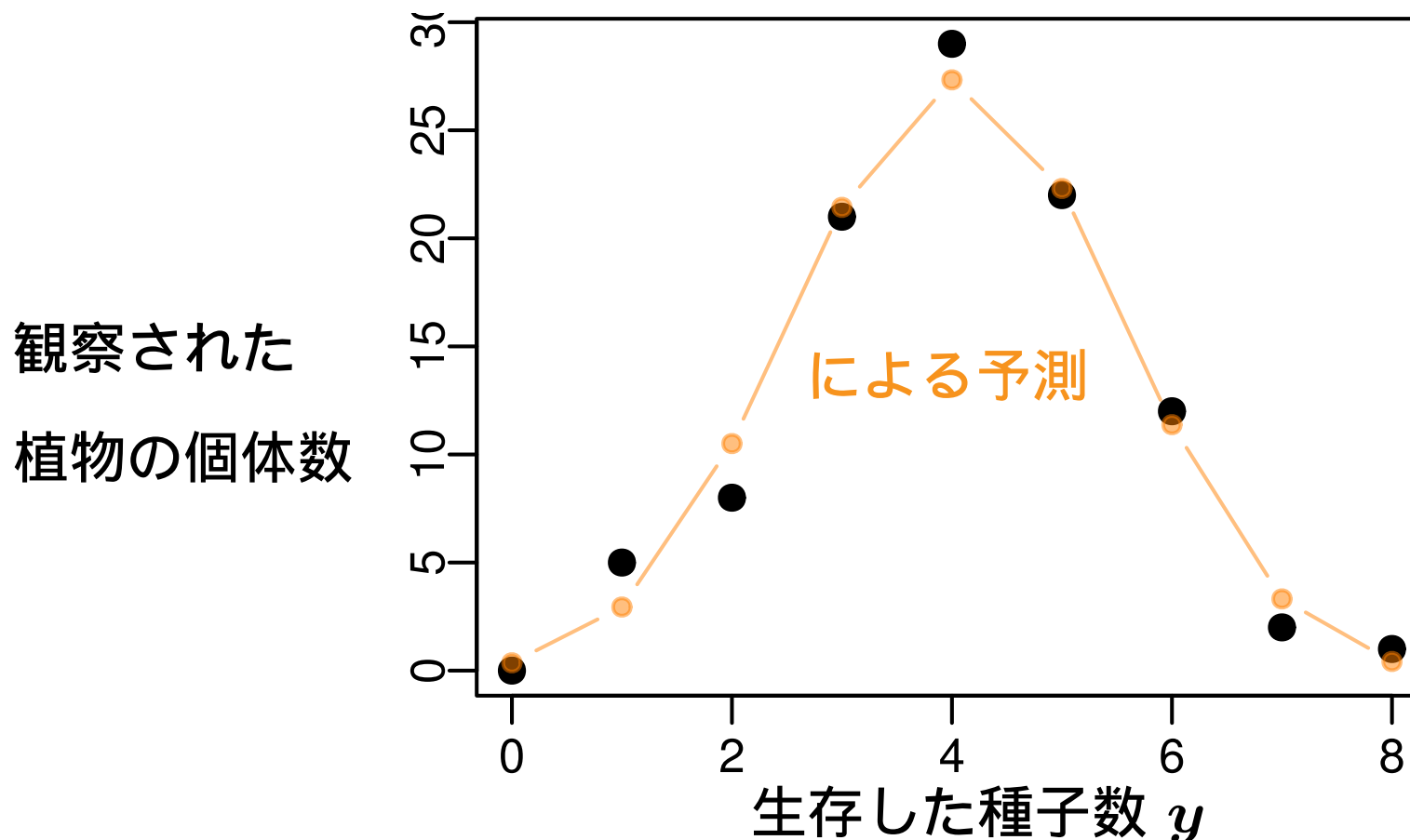
- 生存確率 q が全個体共通の場合の最尤推定量・最尤推定値は

$$= \frac{404}{800} = 0.505$$

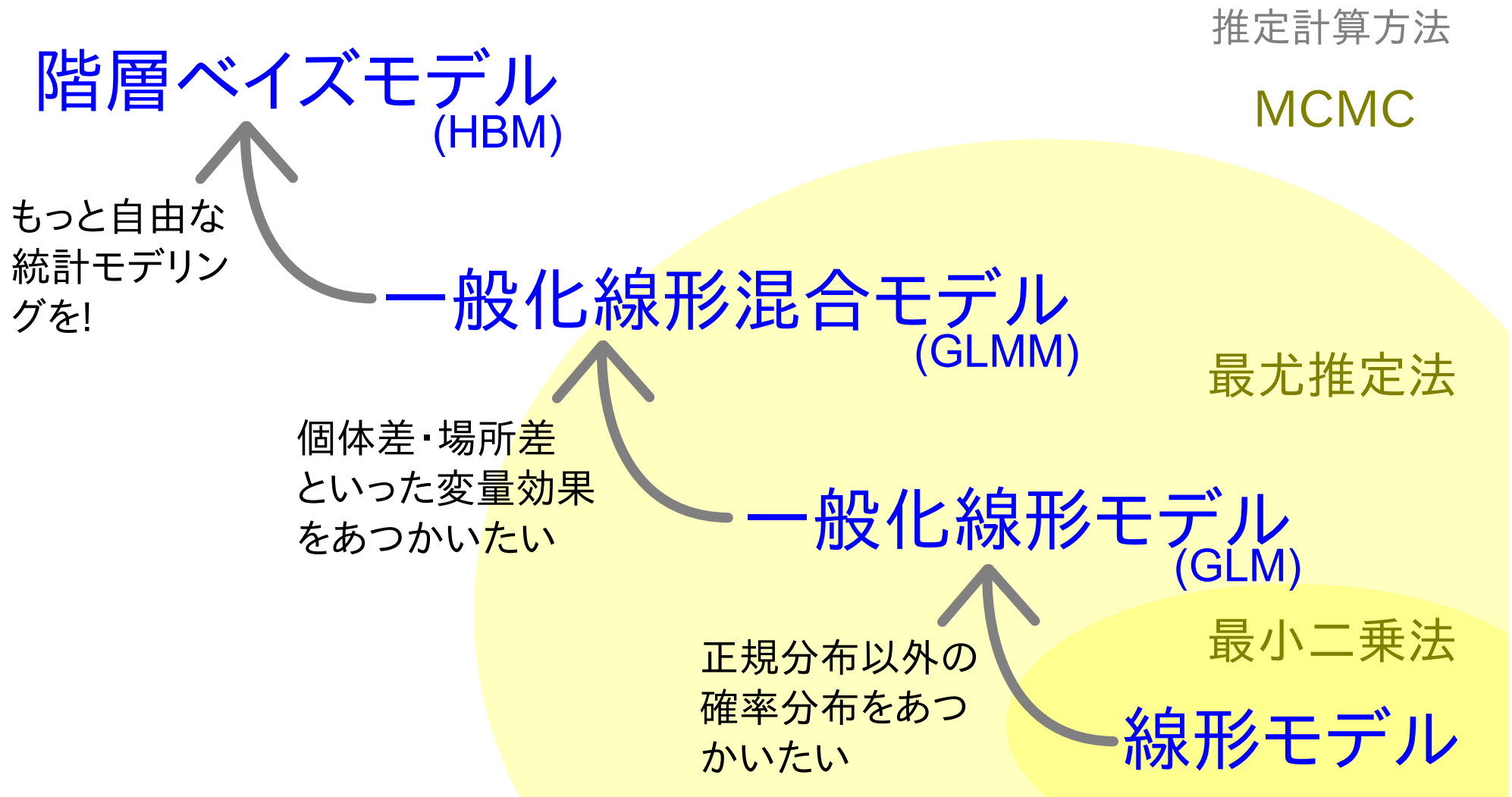


二項分布で説明された 8 種子中 y_i 個の生存

$$\hat{q} = 0.505 \text{ なので } \binom{8}{y} 0.505^y 0.495^{8-y}$$

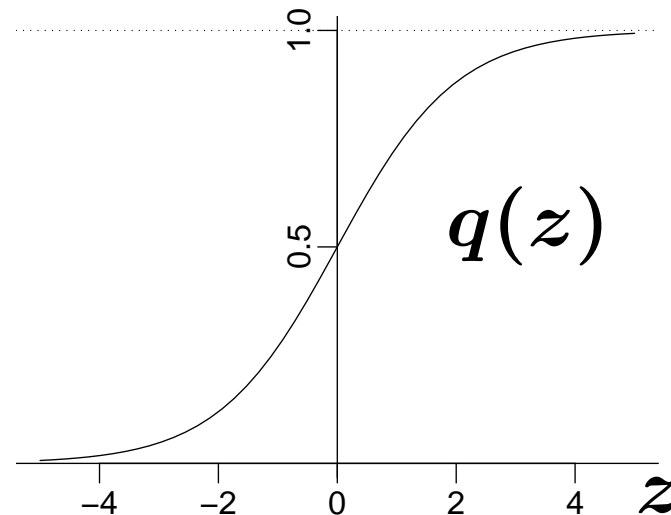


線形モデルの発展



ロジスティック関数で表現する生存確率

- そこで生存する確率 $q_i = q(z_i)$ をロジスティック (logistic) 関数 $q(z) = 1 / \{1 + \exp(-z)\}$ で表現



- 線形予測子 $z_i = a + r_i$ とする
 - パラメーター a : 全体の平均
 - パラメーター r_i : 個体 i の個体差 (ずれ)

ちょっと整理: logistic と logit

- logistic 関数

$$q = \frac{1}{1 + \exp(-(a + bx))} = \text{logistic}(a + bx)$$

- logit 変換

$$\text{logit}(q) = \log \frac{q}{1 - q} = a + bx$$

logit は logistic の逆関数 , logistic は logit の逆関数

R の glm() によるロジスティック回帰

```
> glm(cbind(y, 8 - y) ~ 1, family = binomial, data = d1)
```

```
... (一部略) ...
```

```
Coefficients:
```

```
(Intercept)
```

```
0.02
```

```
Degrees of Freedom: 99 Total (i.e. Null); 99 Residual
```

```
Null Deviance: 110
```

```
Residual Deviance: 110 AIC: 356
```

```
> 1 / (1 + exp(-0.02))
```

```
[1] 0.505
```

ロジスティック回帰の `glm()` 指定

- `family`: `binomial`, 二項分布
- `link` 関数: `"logit"`
- モデル式 (線形予測子 z): たとえば $y \sim x$ と指定

- **線形予測子** $z = a + bx$

a, b は推定すべきパラメーター

- **事象の生起確率** を q とすると $\text{logit}(q) = z$

$$\text{つまり } q = \frac{1}{\exp(-z) + 1} = \frac{1}{1 + \exp(-(a + bx))}$$

- **応答変数** は確率 q でサイズ N の二項分布に従う:

$$y \sim \text{Binom}(q, N)$$

MCMC で生存確率 q を推定する:
パラメータ q の確率分布?

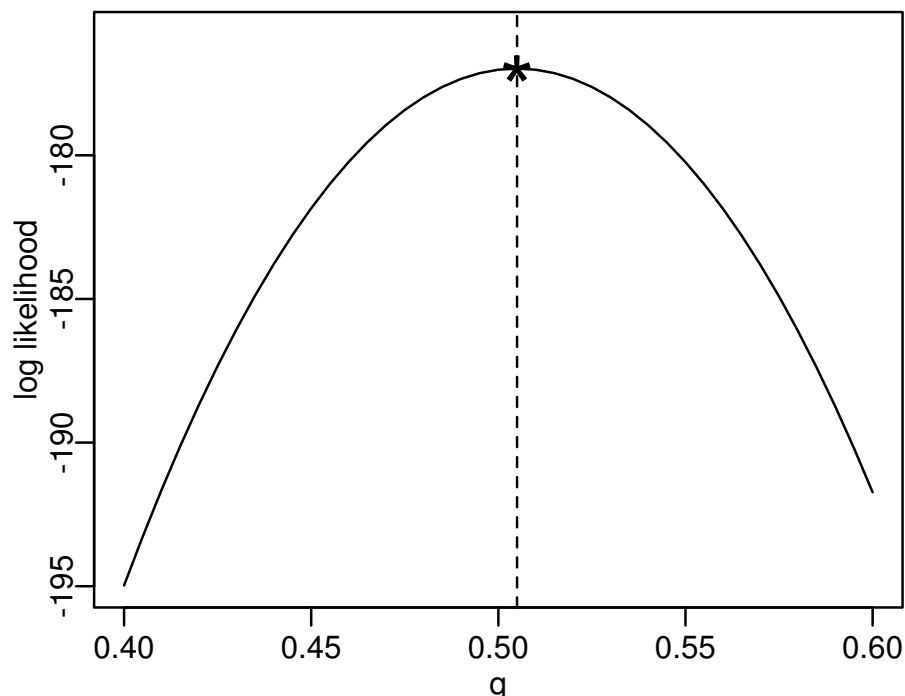
ここでやること: 尤度と MCMC の関係を考える

- さきほどの簡単な例題 (生存確率) のデータ解析を
- 最尤推定ではなく
- 試行錯誤な MCMC 法である **メトロポリス** 法であつかう
- 得られる結果: パラメーターの分布?

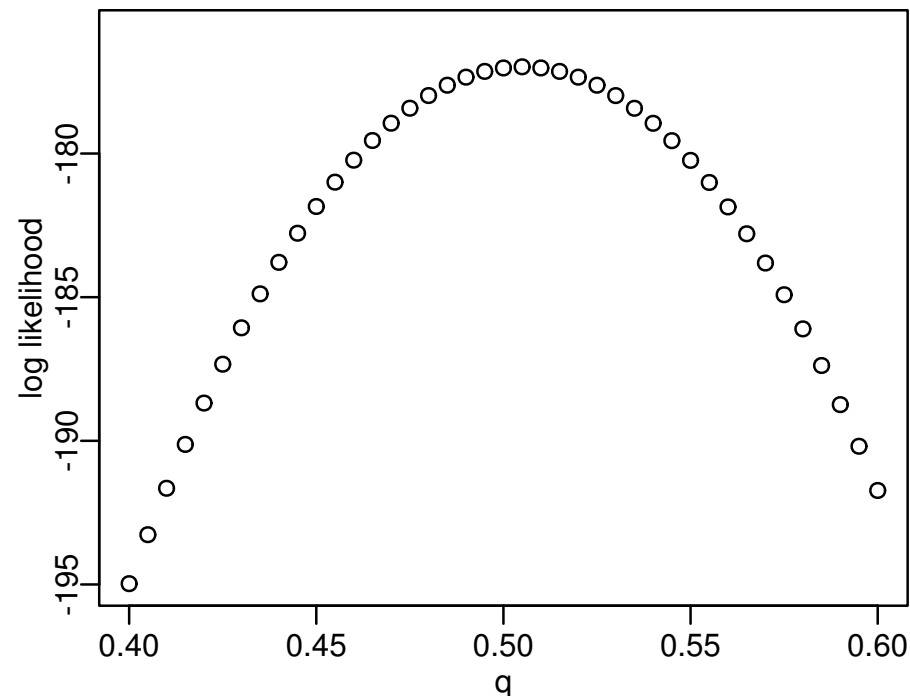
あえて MCMC をもちださなくてもいい問題に関して
メトロポリス法を適用してみて、
その挙動だの得られる結果だのをながめてみる

数値的に試行錯誤するパラメーター推定

連続的な対数尤度関数 $\log L(q)$



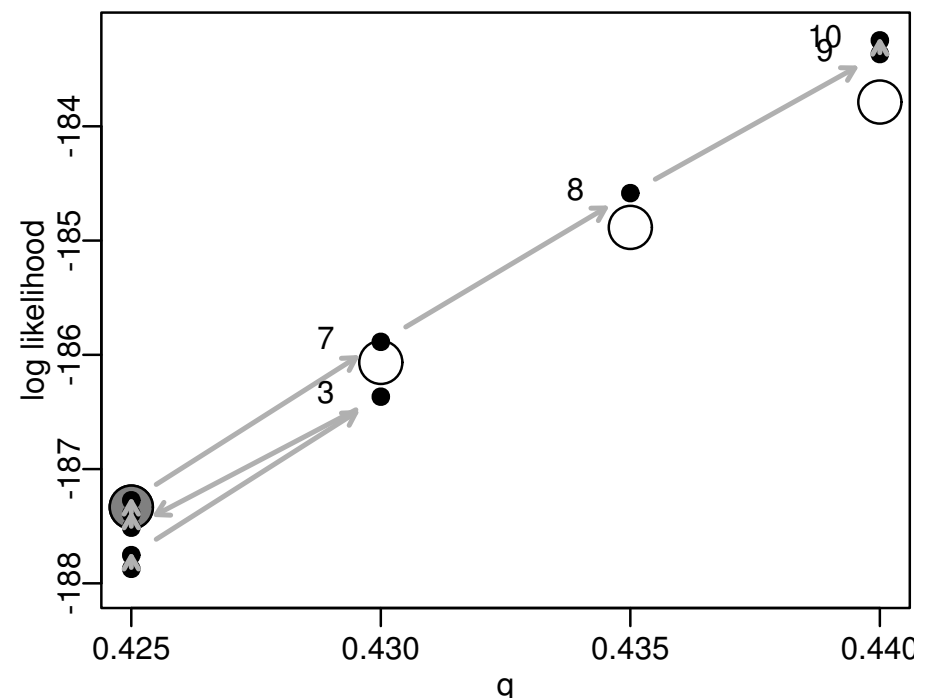
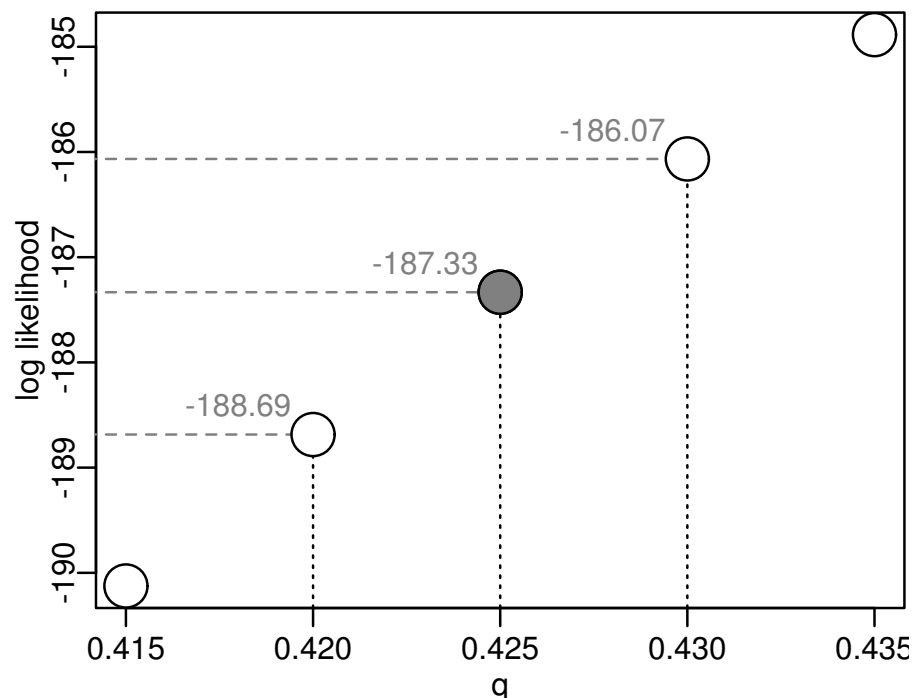
離散化: q がとびとびの値をとる



(簡単のため, 生存確率 q の軸を離散化する)

メトロポリス法で q を変化させていく

メトロポリス法は MCMC アルゴリズムのひとつ (cf. 伊庭さんの解説)



(q の初期値を 0.425 , ランダムウォークで移動先を選ぶ)

(補足) この例題のメトロポリス法

1. パラメーター q の初期値を選ぶ

(ここでは q の初期値が 0.425)

2. q を増やすか減らすかをランダムに決める

(新しく選んだ q の値を $q_{\text{新}}$ としましょう)

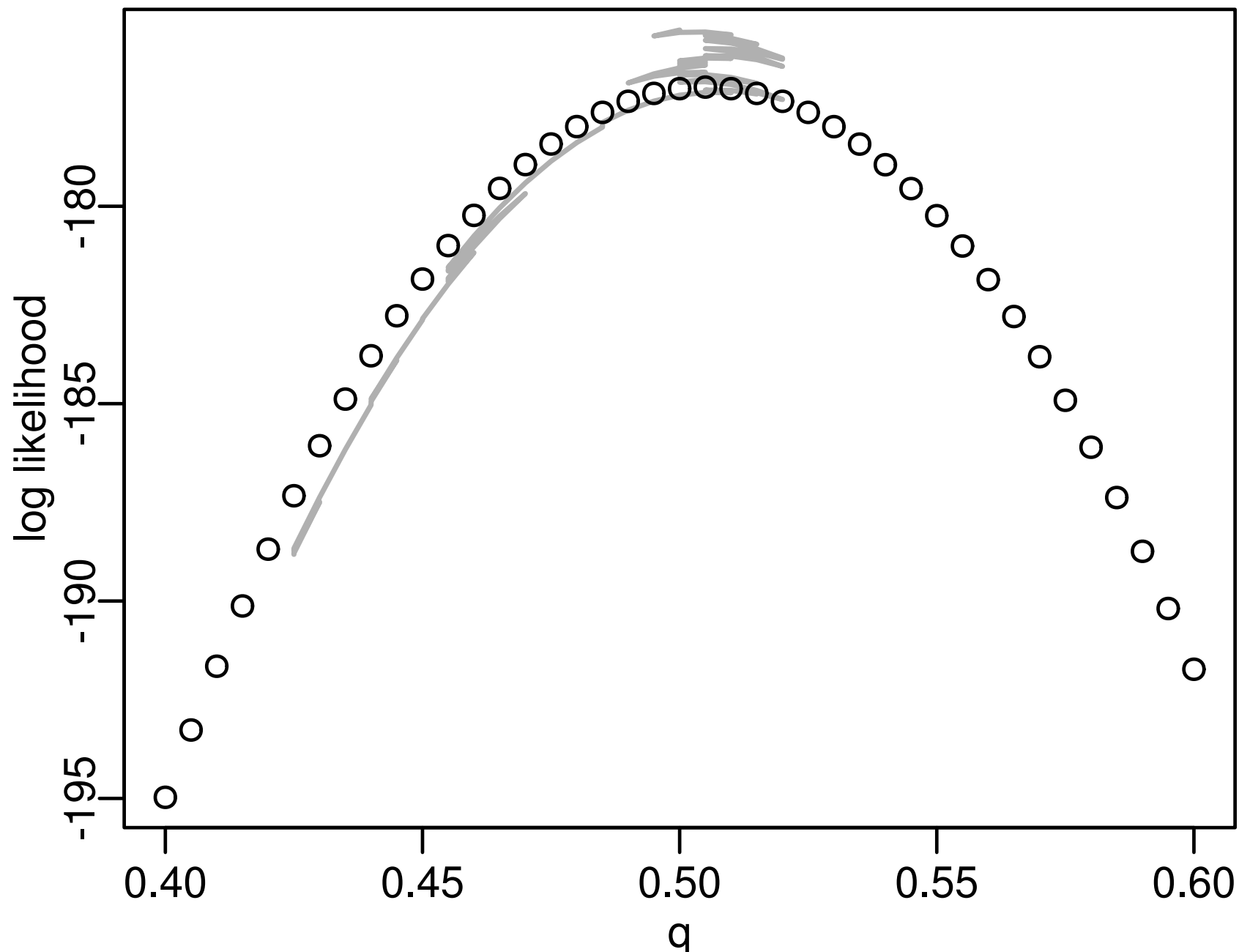
3. $q_{\text{新}}$ における尤度 $L(q_{\text{新}})$ ともとの尤度 $L(q)$ を比較

- $L(q_{\text{新}}) \geq L(q)$ (あてはまり改善): $q \leftarrow q_{\text{新}}$
- $L(q_{\text{新}}) < L(q)$ (あてはまり改悪):
 - 確率 $r = L(q_{\text{新}})/L(q)$ で $q \leftarrow q_{\text{新}}$
 - 確率 $1 - r$ で q を変更しない

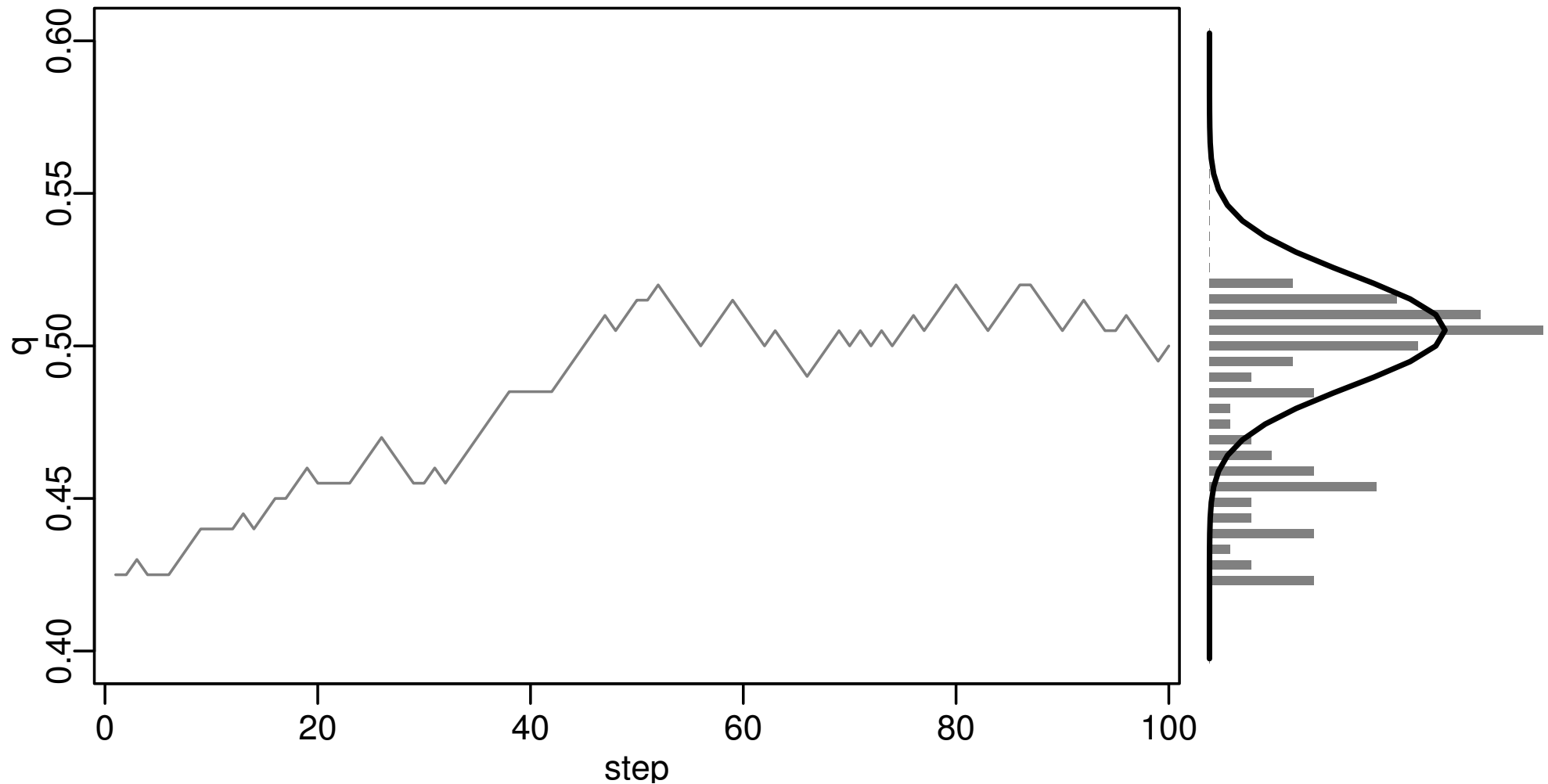
4. 手順 2. にもどる

($q = 0.01$ や $q = 0.99$ でどうなるんだ, といった問題は省略)

対数尤度関数上での生存確率 q の変化

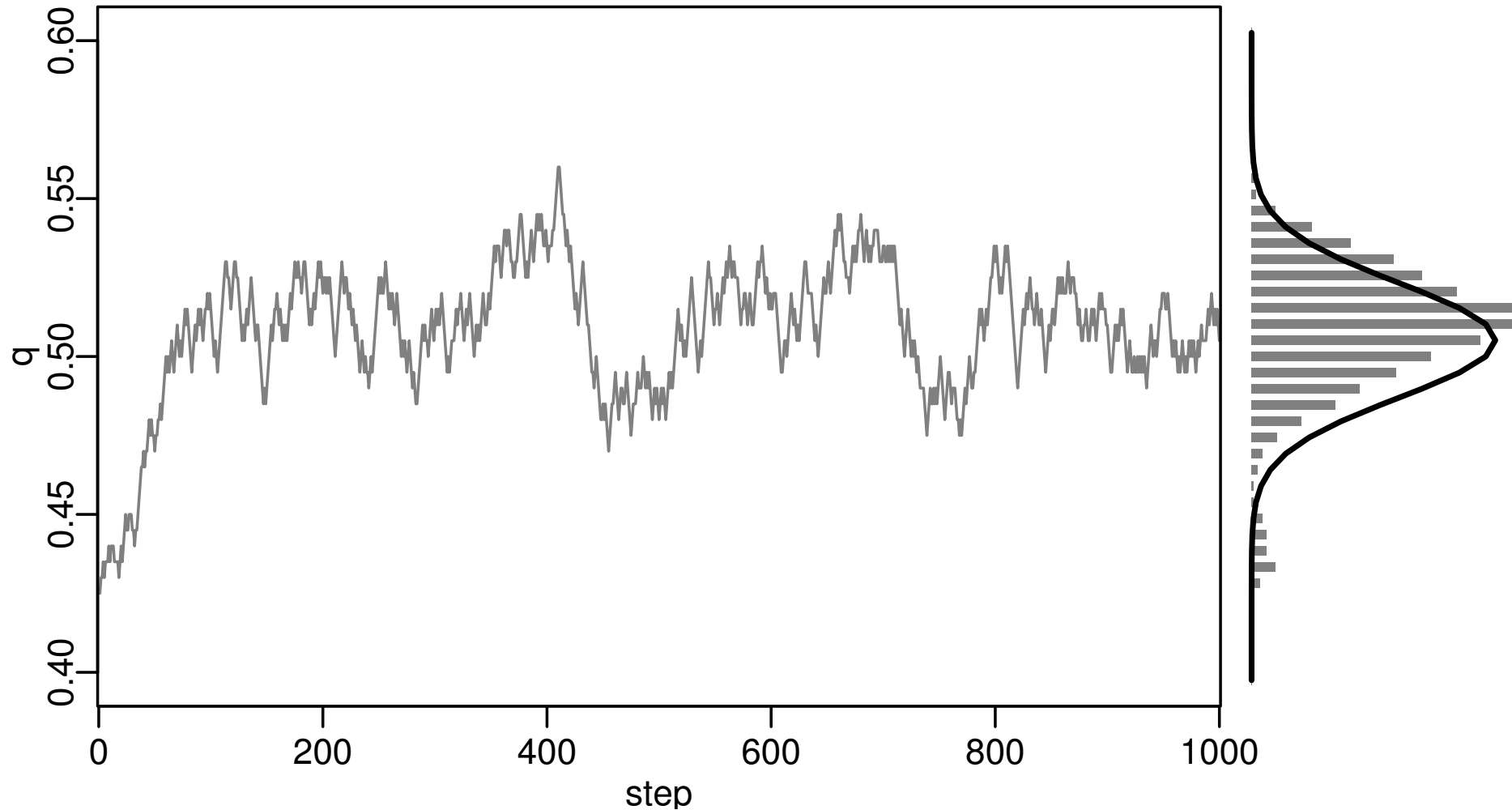


MCMC ステップにそった q の変化



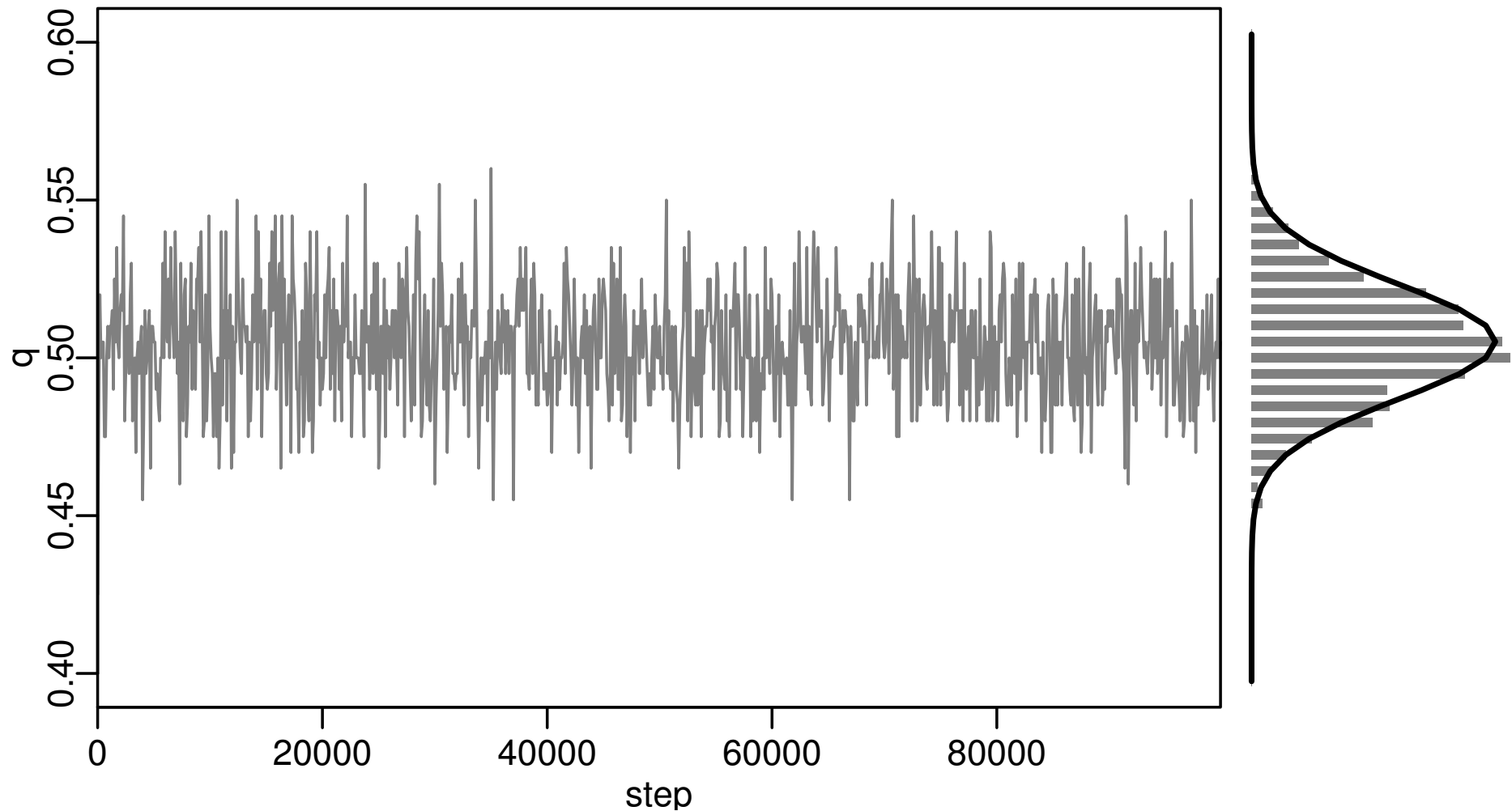
右側は q のヒストグラム

もっと長くサンプリングしてみる



まだまだ.....?

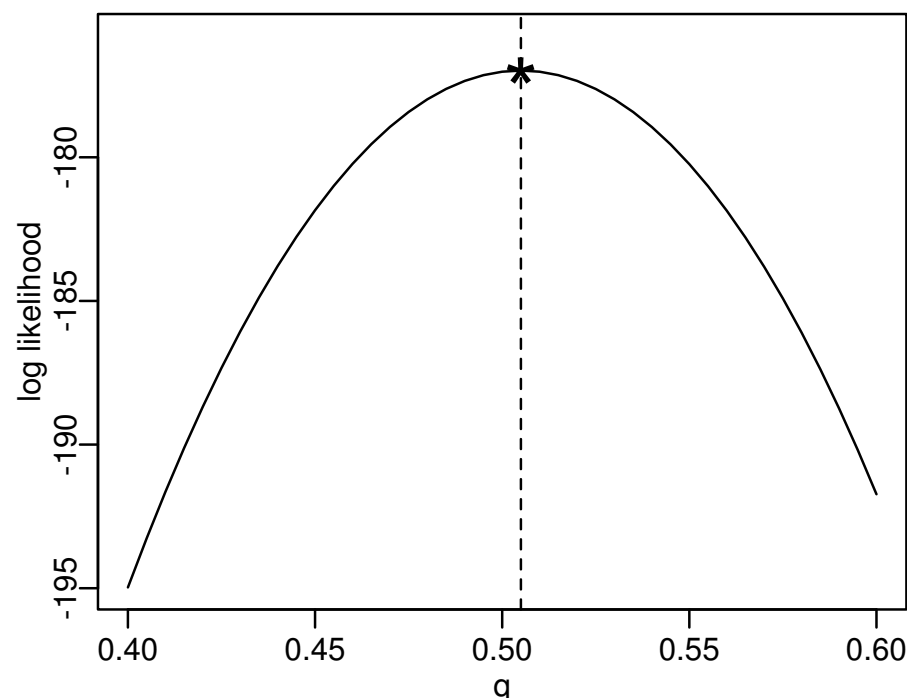
もっともっと長くサンプリングしてみる



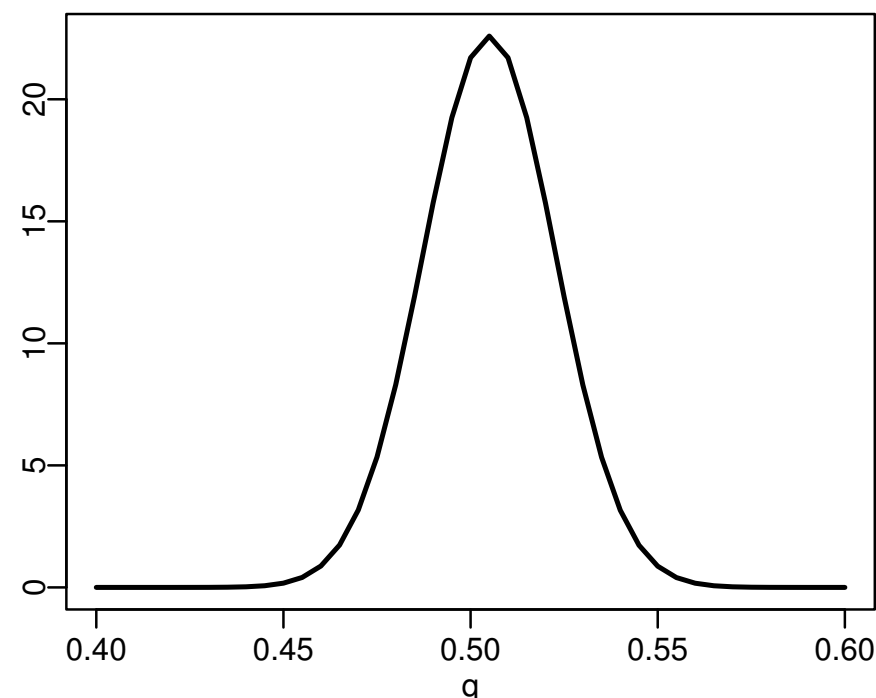
ターゲットとなる「パラメーターの分布」に近づいてきた

MCMCは何をサンプリングしている？

既出の対数尤度 $\log L(q)$



尤度 $L(q)$ に比例する確率分布

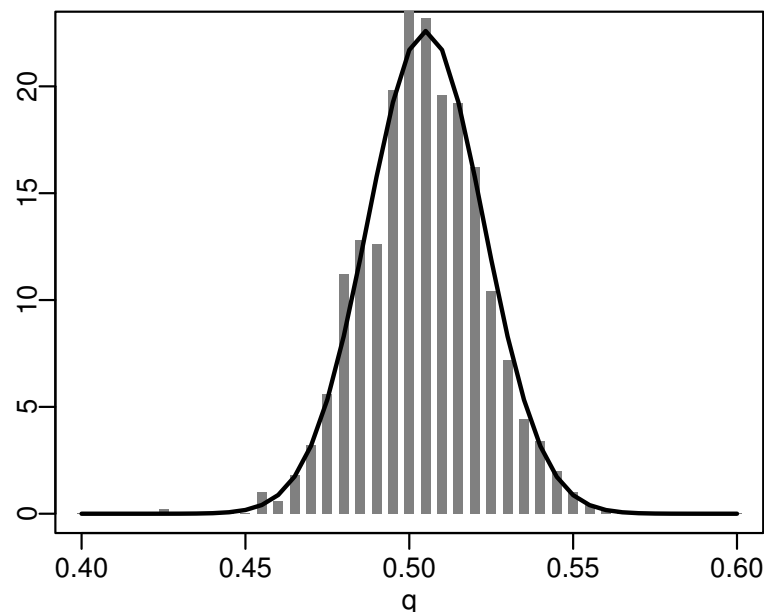


尤度に比例する確率分布からのランダムサンプル

(「パラメーターの分布」と仮称)

「マルコフ連鎖の収束定理」のおかげ (cf. 伊庭さんの説明)

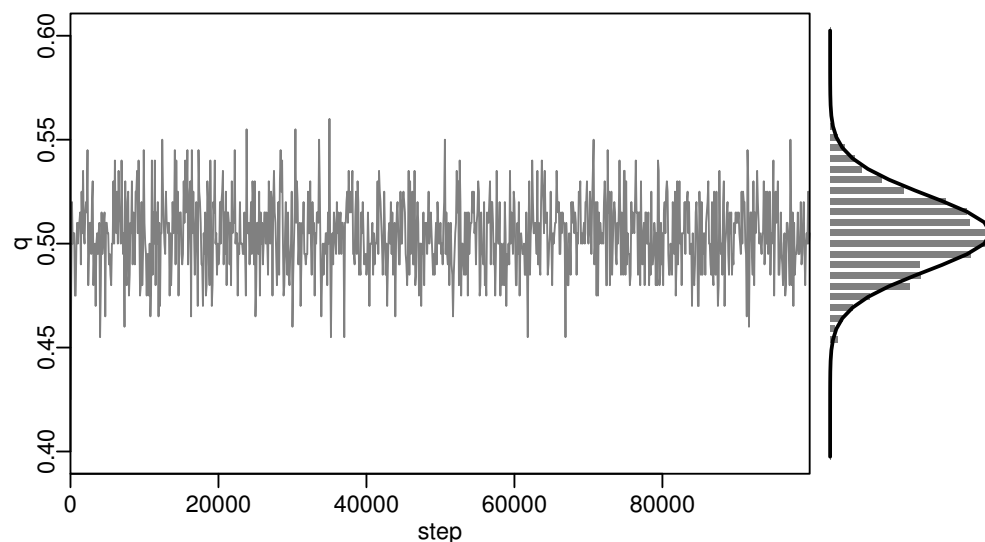
MCMC の結果として得られた「 q の分布」



- データからえられる推定結果としては有用: 分布の平均や区間推定など
- 「パラメーターの分布」 ベイズ統計でいうところの事後分布

いったん整理: 尤度と MCMC の関係

- 統計モデルを作ると, あるデータのもとでの**尤度**が定義される
- この尤度に対して MCMC すると「尤度に比例する**パラメーターの分布**」からのランダムサンプルがえられる
- ベイズとの関連: これは**事後分布**からのサンプリングである



いったん整理: いろいろな MCMC の方法

- **メトロポリス法**: 試行錯誤で値を変化させていく MCMC
 - メトロポリス・ヘイスティングス法: その改良版
 - **ギブス・サンプラー**: 条件つき確率分布を使った MCMC
 - 普通は複数の変数 (パラメーター・状態) のサンプリングのためにもちいる (あとでこの例題にそった簡単な説明)
-
- メトロポリス法で説明したけれどギブス・サンプラーでも同じことが言える
 - ここからあとで登場する MCMC はギブス・サンプラーと考えてください

ベイズモデル: 尤度・事後分布・事前分布.....

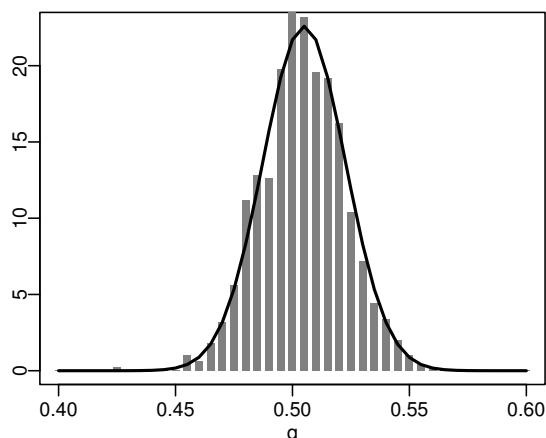
- ベイズの公式
$$p(q | Y) = \frac{p(Y | q) \times p(q)}{p(Y)}$$
- $p(q | Y)$ は何かデータ (Y) のもとで何かパラメーター (q) が得られる確率 (事後分布)
- $p(q)$ はあるパラメーター q が得られる確率 (事前分布)
- $p(Y | q)$ パラメーターを決めたときにデータが得られる確率 (尤度に比例)
- $p(Y)$ は「てもとにあるデータ Y が得られる確率」(単なる規格化定数)

$$\begin{aligned} \text{(事後分布)} & \propto \frac{\text{尤度} \times \text{事前分布}}{\text{(データが得られる確率)}} \\ & \propto \text{尤度} \times \text{事前分布} \end{aligned}$$

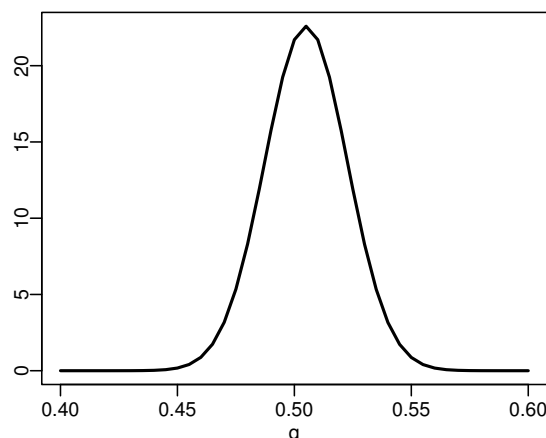
現在の例題で仮定している事前分布

q の事前分布は一様分布，と考えるとつじつまがあう？

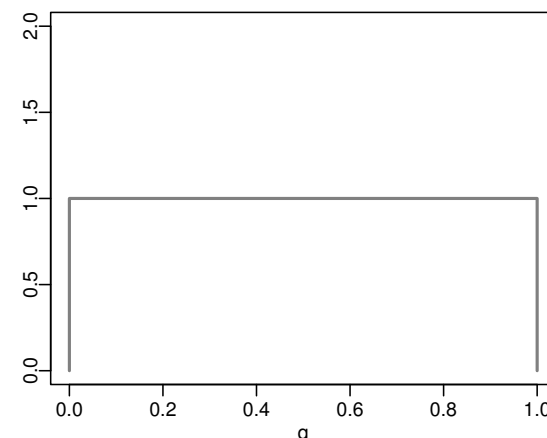
q の事後分布
(posterior)



q の尤度
(likelihood)



q の事前分布
(prior)



\propto

\times

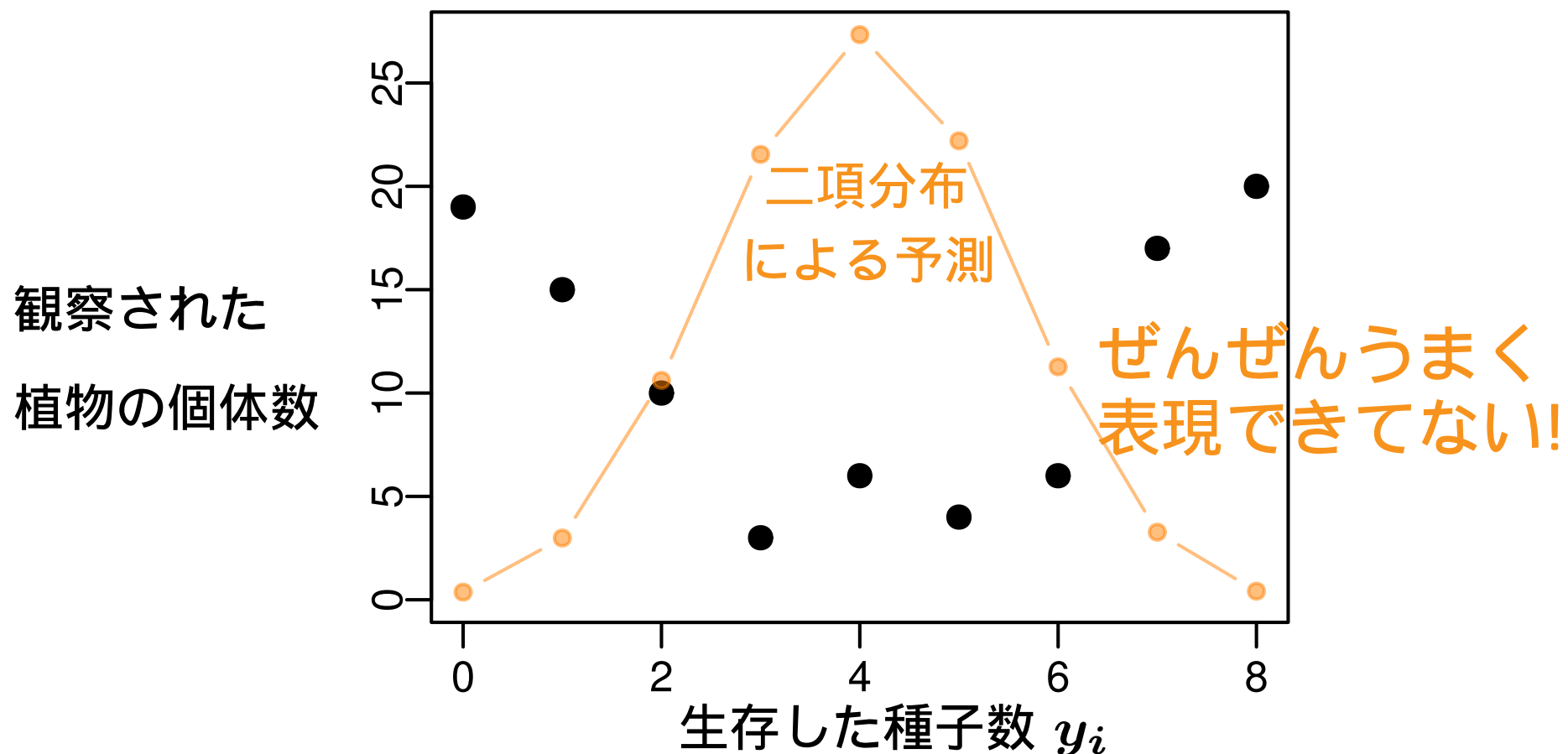
このように「 q はどんな値でもいいんですよ」という気分を表現するための事前分布が**無情報事前分布** (non-informative prior)

ちょっと難しい例題:

階層ベイズモデルが必要になる状況

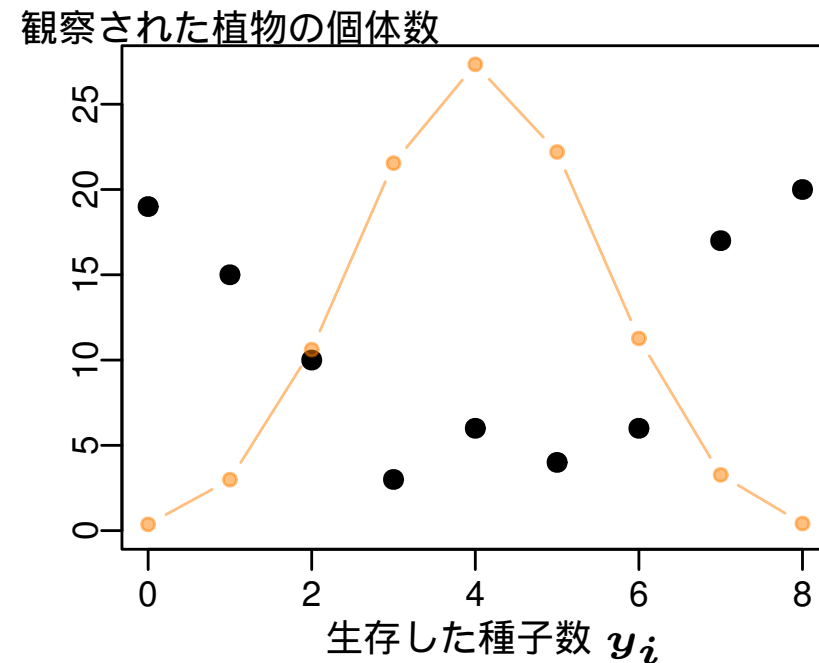
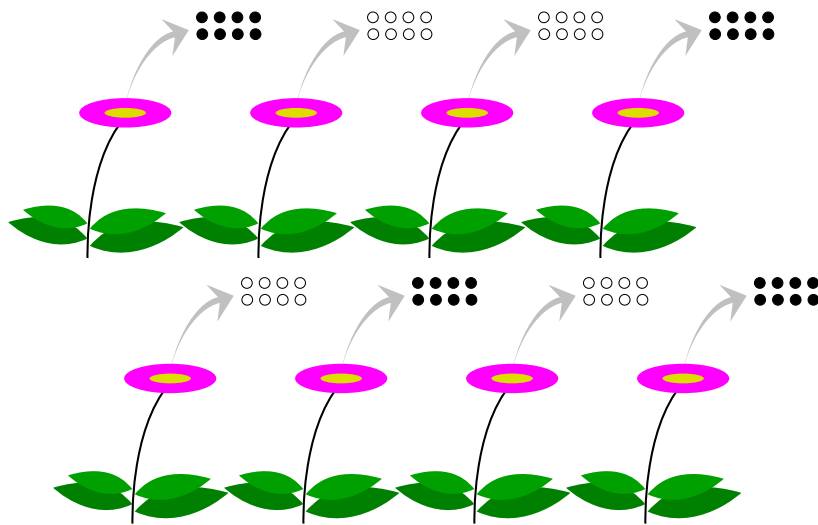
また別の観測データ: 二項分布だめだめ?!

100 個体の植物の合計 800 種子中 **403 個** の生存が見られたので, 平均生存確率は 0.50 と推定されたが.....



「個体差」 → 過分散 (overdispersion)

極端な過分散の例



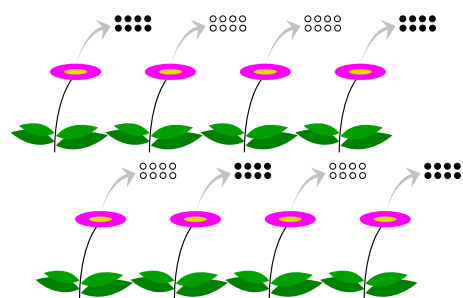
- 種子全体の平均生存確率は 0.5 ぐらいかもしれないが.....
- 植物個体ごとに種子の生存確率が異なる: 「個体差」
- 「個体差」があると overdispersion が生じる
- 「個体差」の原因: ?

あのー …… 「個体差」とは?

- 生物学的には明確な定義はない
- しかしデータ解析においては人間が主観的に「これは個体差由来の効果であり、観察されたパターンに影響している」と定義、そして以下の二種類を区別する:
 1. fixed effects 的な効果
 2. random effects 的な効果
 - これって何なの?

「個体差」の fixed だの random だの って何?

- 「個体ごとに異なる何かに由来する効果」を fixed/random effects にわけて統計モデリングする:
 1. fixed effects 的な効果: **観測者がわざわざ設定・測定した**要因 (実験処理, 植物のサイズなど), logit 変換された世界において生存確率の「効果の大きさ」を変える
 - この例題では fixed effects 的な要因なし
 2. random effects 的な効果: fixed effects 的ではない要因 (観測対象個体に関連する, **人間が設定・測定していないすべて**)
 - logit 変換された世界において生存確率の「効果の大きさ」を変えずにばらつきだけを変えたと考える

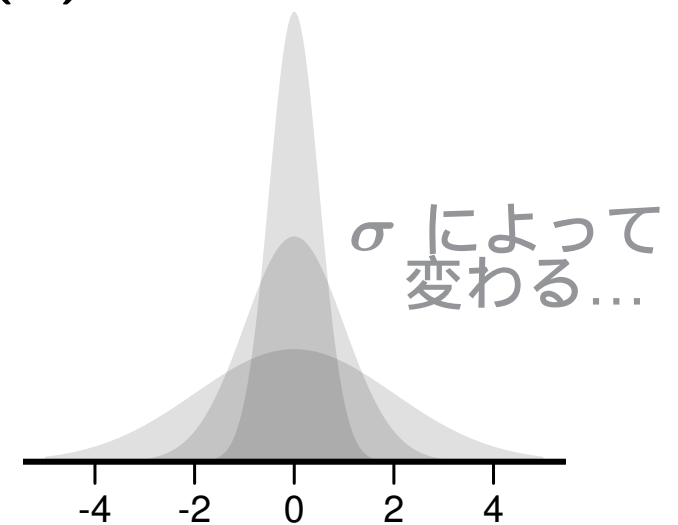
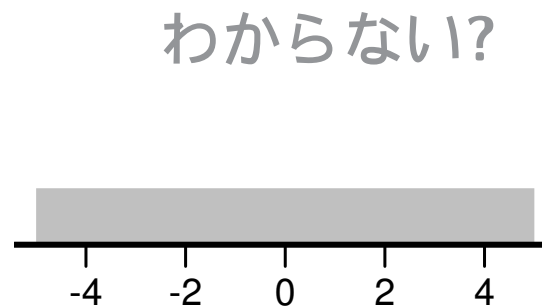
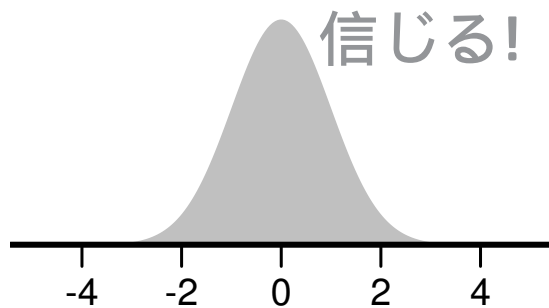


今回の例題では random effects な
「個体差」だけをあつかう (説明変数なし)

やっぱりよくわからない?

パラメーターごとに設定してやる**事前分布の種類**(あとで説明するベイズ統計モデリング)にもとづいて考えたほうが,わかりやすいかも?

(A) 主観的な事前分布 (B) 無情報事前分布 (C) 階層的な事前分布

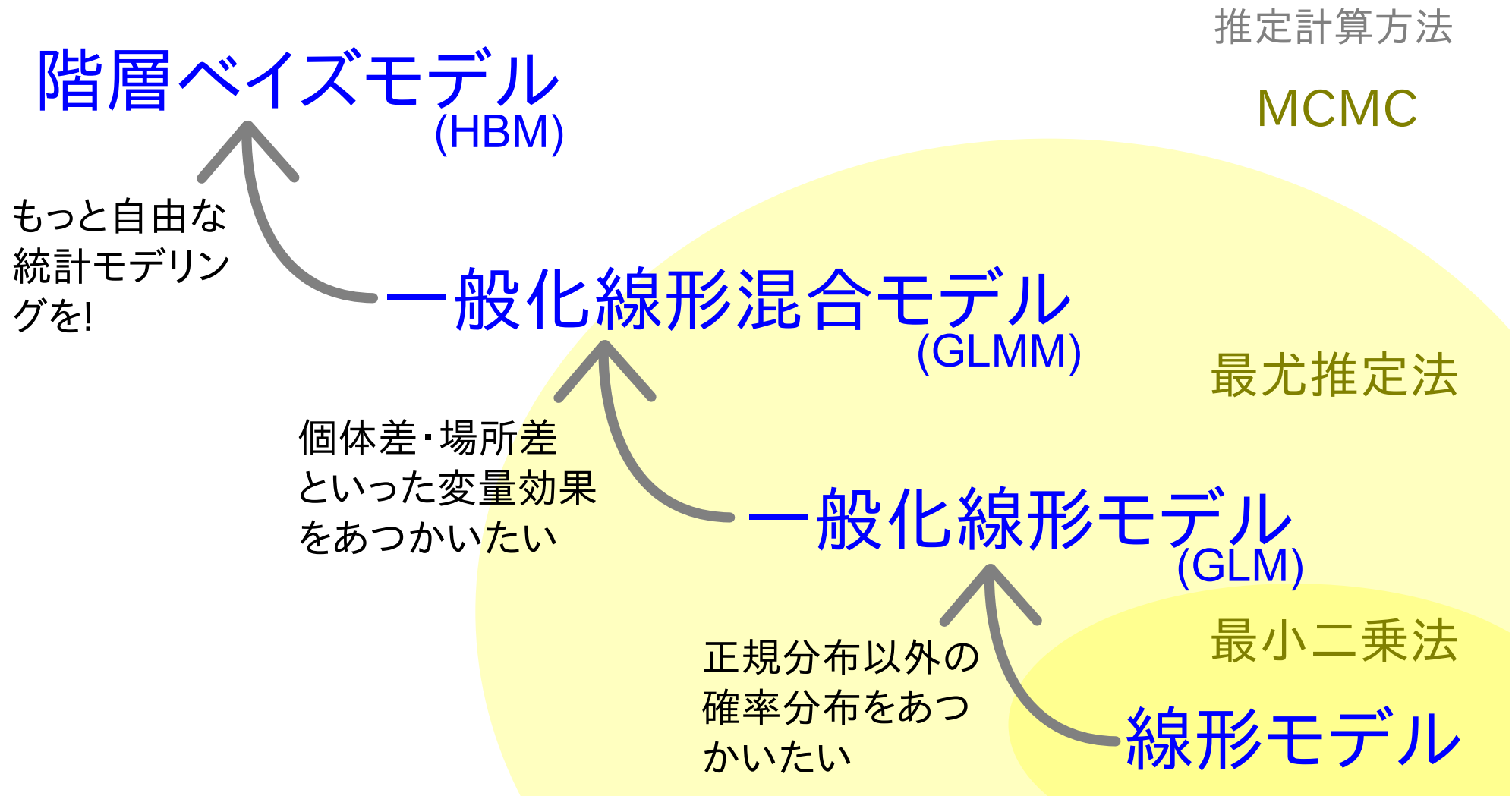


このパラメーターの事前分布は

どう設定するのが妥当だろうか, と検討する

ちょっとまた線形モデルのたぐいについて整理してみましよう.....

線形モデルの発展



モデリングやりなおし: まず二項分布の再検討

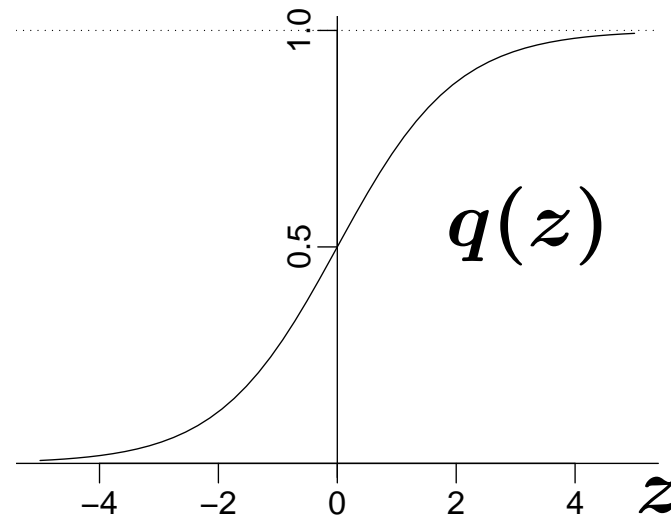
- 生存確率を推定するために **二項分布** という確率分布を使う
- 個体 i の N_i 種子中 y_i 個が生存する確率は二項分布

$$p(y_i | q_i) = \binom{N_i}{y_i} q_i^{y_i} (1 - q_i)^{N_i - y_i},$$

- ここで仮定していること
 - **個体差がある**
 - 個体ごとに異なる生存確率 q_i

ロジスティック関数で表現する生存確率

- そこで生存する確率 $q_i = q(z_i)$ をロジスティック (logistic) 関数 $q(z) = 1 / \{1 + \exp(-z)\}$ で表現



- 線形予測子 $z_i = a + r_i$ とする
 - パラメーター a : 全体の平均
 - パラメーター r_i : 個体 i の個体差 (ずれ)

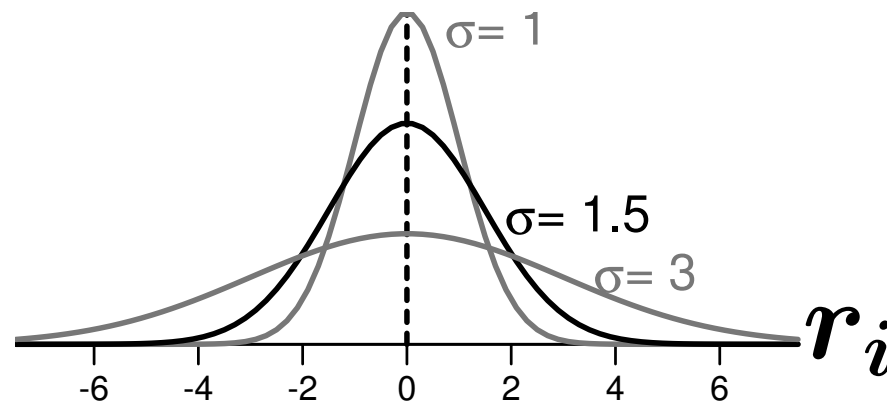
個々の個体差 r_i を最尤推定するのはまずい

- 100 個体の生存確率を推定するためにパラメーター 101 個(a と $\{r_1, r_2, \dots, r_{100}\}$) を推定すると.....
- 個体ごとに生存数 / 種子数を計算していることと同じ!
(「データのよみあげ」と同じ)
- こう仮定すると問題がうまくあつかえないだろうか?
 - 個体間の生存確率はばらつくけど, そんなにすごく異ならない?
 - 観測データを使って, 「個体差」にみられるパターンを抽出したい (統計モデル化)

階層ベイズモデル化: r_i の事前分布の設計

平均ゼロで標準偏差 σ の正規分布

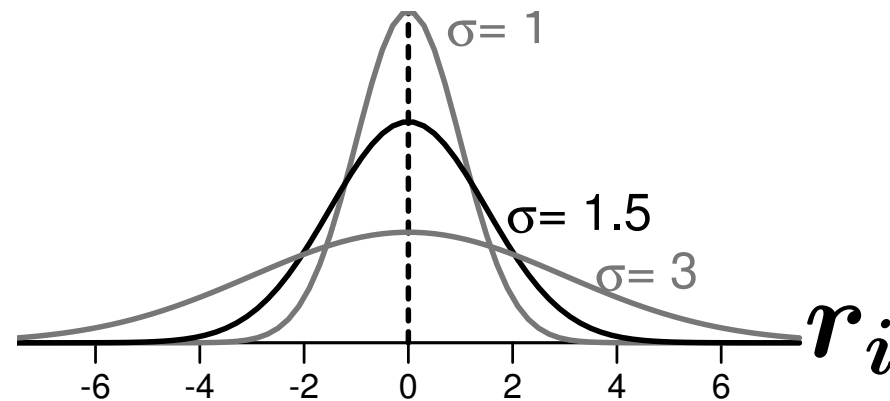
$$p(r_i | \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-r_i^2}{2\sigma^2}\right)$$



個体差 $\{r_1, r_2, \dots, r_{100}\}$ がこの確率分布に従うとする

r_i の事前分布は無情報事前分布ではない

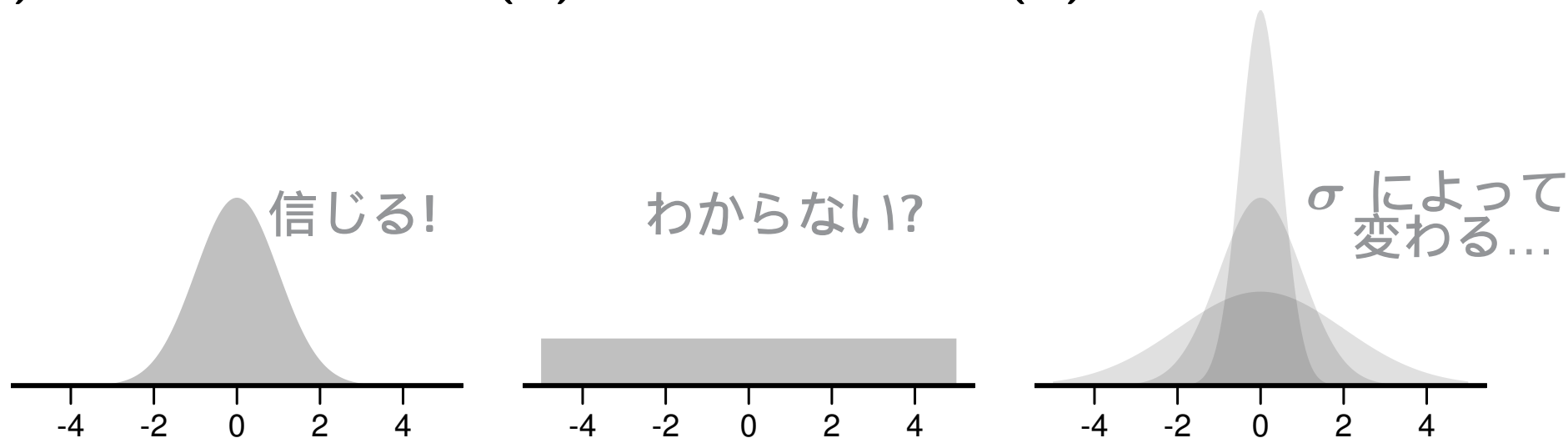
データにあわせて σ が変化する階層的な事前分布



- σ がとても小さければ個体差 r_i はどれもゼロちかくなる → 「どの個体もおたがい似ている」
- σ がとても大きければ, r_i は各個体の生存数 y_i にあわせるような値をとる

個体差 r_i の事前分布は階層的な事前分布

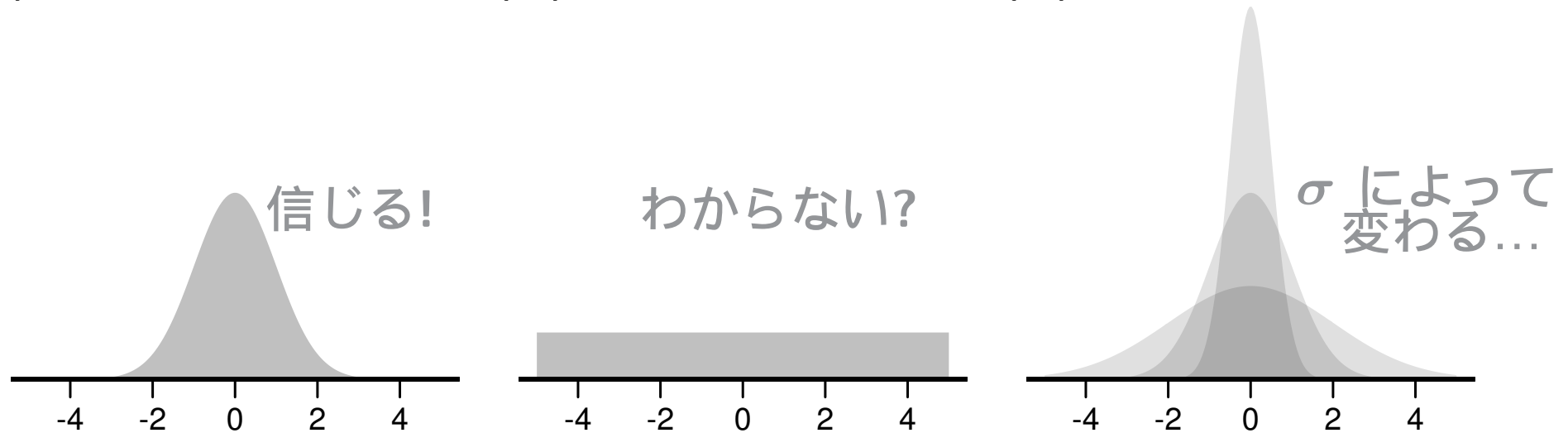
(A) 主観的な事前分布 (B) 無情報事前分布 (C) 階層的な事前分布



- (A) 主観的な事前分布: 「自分の信じるところによれば, r_i たちはこんな分布になる」を表現している.
- (B) 無情報事前分布: 「 r_i たちがどんな値になるのかまったくわかりません」を表現しようとしている (しかし -5 から 5 ぐらい, という主観も表現している).
- (C) **階層的な事前分布**: r_i の事前分布のパラメーター σ がいろいろな値をとる, そして σ についての超事前分布を設定する.

パラメーターごとに事前分布を選ぶ (1)

(A) 主観的な事前分布 (B) 無情報事前分布 (C) 階層的な事前分布

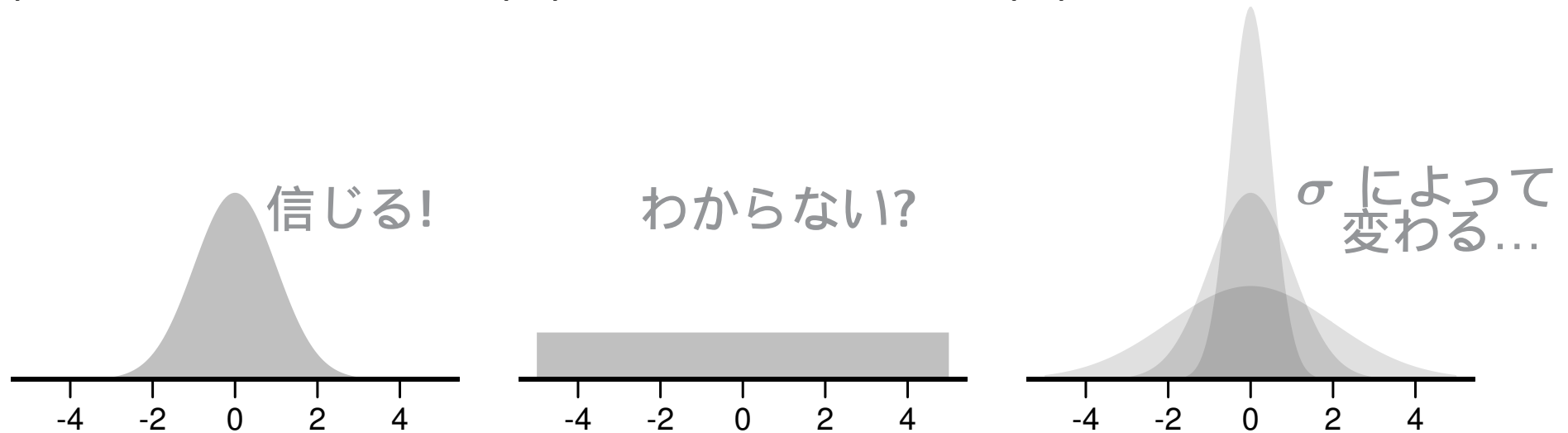


(何か植物たちの統計モデリングをやっているとすると)

- fixed effects 的な効果:
 - 無情報事前分布を設定する
 - (実験処理の効果, 植物個体の大きさなど属性の効果などを) 全個体に共通する効果をひとつのパラメーターで表現しているから

パラメーターごとに事前分布を選ぶ (2)

(A) 主観的な事前分布 (B) 無情報事前分布 (C) 階層的な事前分布

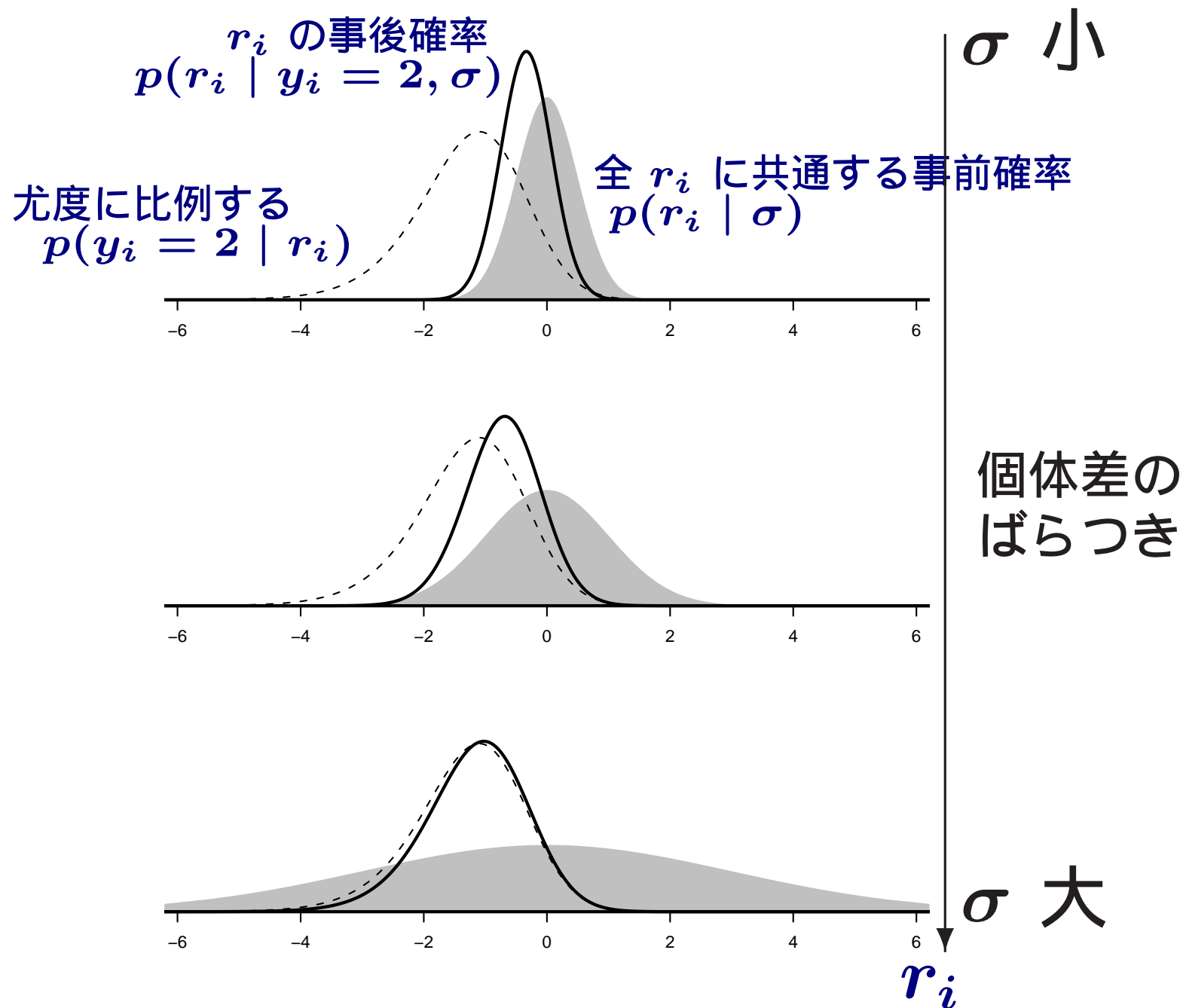


(何か植物たちの統計モデリングをやっているとすると)

- random effects 的な効果:

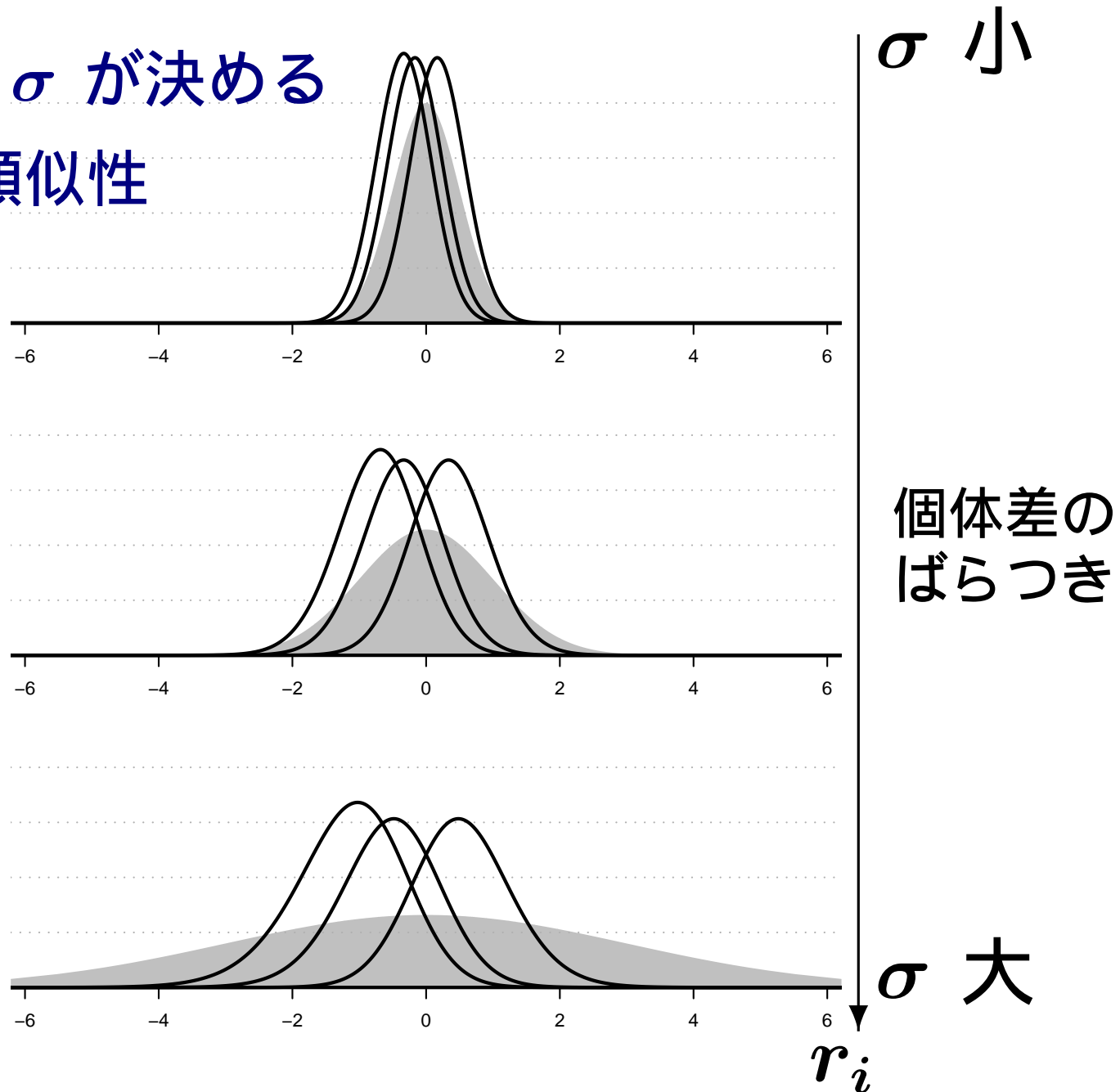
- 階層的な事前分布を設定する
- 個体ごとに異なるばらつき r_i が何かあって, しかし集団全体で $\{r_i\}$ が何かの (意味ありげな) 確率分布にしたがっていると考えるのが妥当そうだから

階層的な事前分布と $y_i = 2$ の個体の r_i

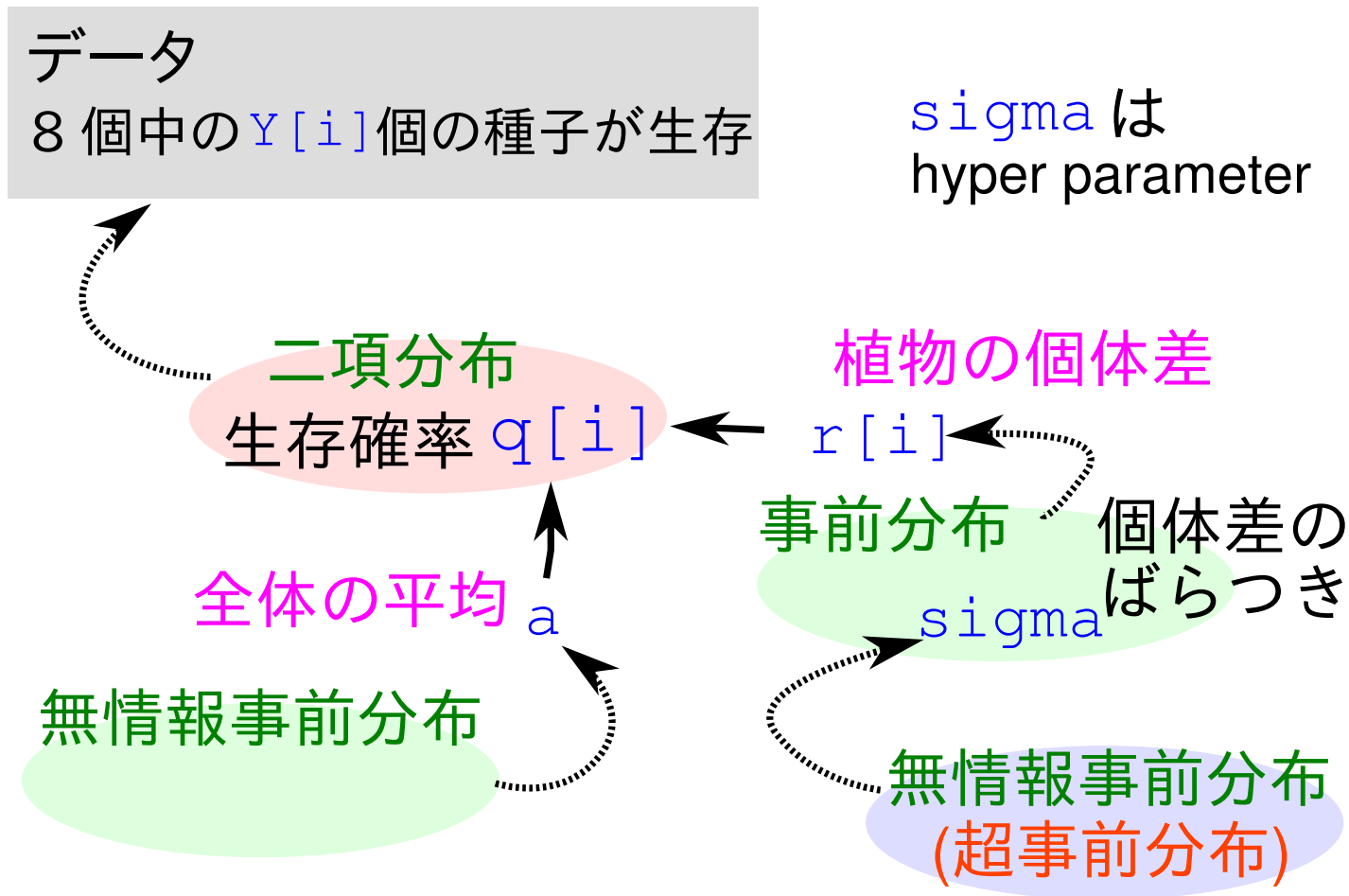


階層的な事前分布と $y_i \in \{2, 3, 5\}$ の個体の r_i

パラメーター σ が決める
個体間の類似性

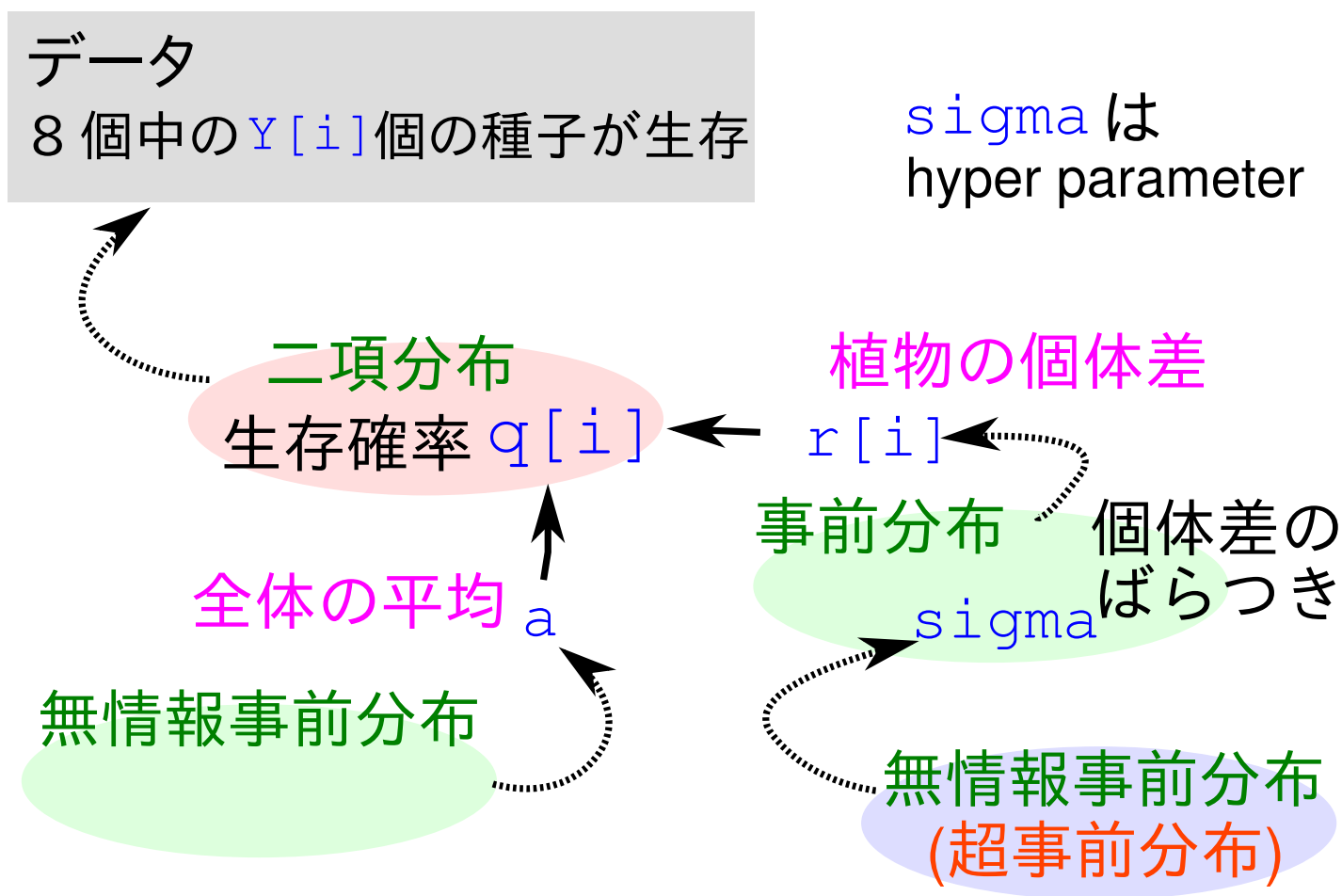


なぜ「階層」ベイズモデルと呼ばれるのか？



超事前分布 → 事前分布という階層があるから

全パラメーターを一斉に推定する

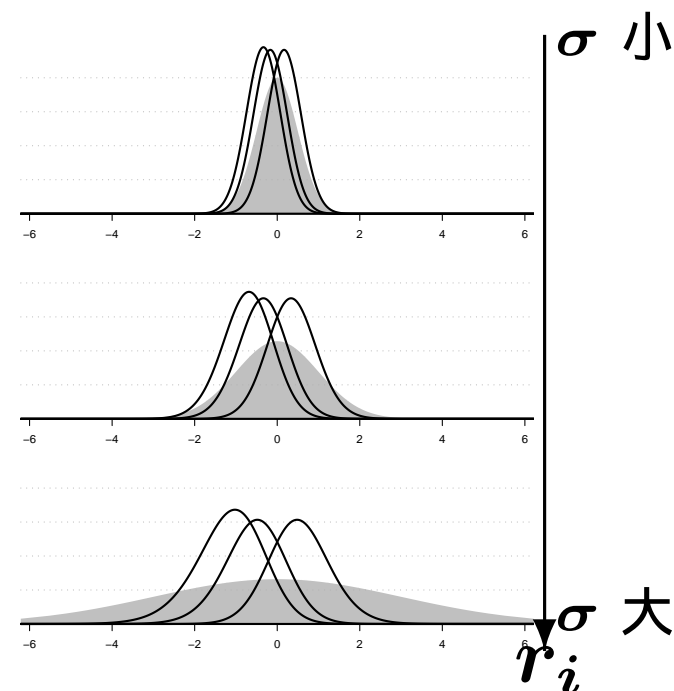


矢印は手順ではなく，依存関係をあらわしている

階層ベイズモデルではないベイズモデルって何でしょう？

個体差 r_i の事前分布の設定を例に検討してみる

- 事前分布を主観的に決める
「自分は $\sigma = 0.1$ と信じるので、それを使う」
- 以前のデータを使う？
「これまでの経験から $\sigma = 0.1$ 」
- 無情報事前分布ばかりにする
「よくわからないので σ をすごく大きくする」



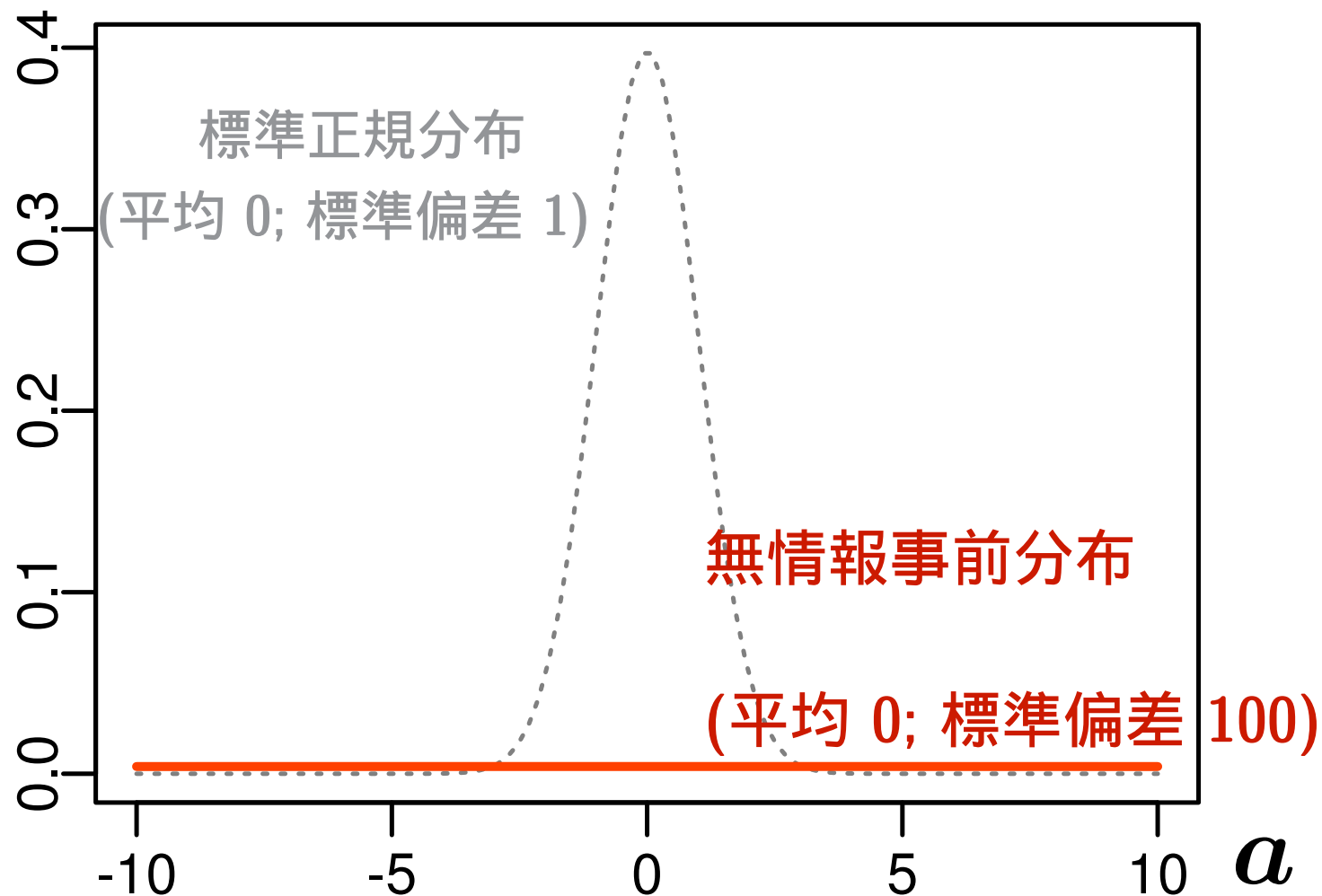
(これらに対して)

観測データにもとづいて σ を決めようとする
のが階層ベイズモデル

個体差 $\{r_i\}$ のばらつき σ の無情報事前分布

- σ はどのような値をとってもかまわない
- そこで σ の事前分布は **無情報事前分布**(non-informative prior) とする
- たとえば一様分布
 - とりあえず, ここでは $0 < \sigma < 10^4$ の一様分布としてみる

全個体の「切片」 a の無情報事前分布



「生存確率の (logit) 平均 a は何でもよい」と表現している

階層ベイズモデル全体の定式化

$$p(a, \{r_i\}, \sigma \mid \text{データ}) = \frac{\prod_{i=1}^{100} p(y_i \mid q(a + r_i)) p(a) p(r_i \mid \sigma) p(\sigma)}{\iint \cdots \int (\text{分子} \uparrow \text{そのまま}) dr_i d\sigma da}$$

分母は何か**定数**になるので

$$p(a, \{r_i\}, \sigma \mid \text{データ}) \propto \prod_{i=1}^{100} p(y_i \mid q(a + r_i)) p(a) p(r_i \mid \sigma) p(\sigma)$$

$$p(a, \{r_i\}, \sigma \mid \text{データ})$$

$$\prod_{i=1}^{100} p(y_i \mid q(a + r_i))$$

事前分布たち: $p(a) p(r_i \mid \sigma) p(\sigma)$

どうやって事後分布を推定するの?

$$p(a, \{r_i\}, \sigma \mid \text{データ}) \propto \prod_{i=1}^{100} p(y_i \mid q(a + r_i)) p(a) p(r_i \mid \sigma) p(\sigma)$$

- 観測データと事前分布を組みあわせれば **事後分布** $p(a, \{r_i\}, \sigma \mid \text{データ})$ を知ることができるはず
- しかし右辺をみてもよくわからない
- Markov chain Monte Carlo (MCMC) を使えば「よくわからない確率分布」から事後分布が得られる!
→ ということで, **WinBUGS** のハナシに.....

パラメーターの条件つき分布から Gibbs sampling

サンプリングの対象とするパラメーター以外は値を固定する

$$p(a \mid \cdots) \propto \prod_{i=1}^{100} p(y_i \mid q(a + r_i)) p(a)$$

$$p(\sigma \mid \cdots) \propto \prod_{i=1}^{100} p(r_i \mid \sigma) p(\sigma)$$

$$p(r_1 \mid \cdots) \propto p(y_1 \mid q(a + r_1)) p(r_1 \mid \sigma)$$

$$p(r_2 \mid \cdots) \propto p(y_2 \mid q(a + r_2)) p(r_2 \mid \sigma)$$

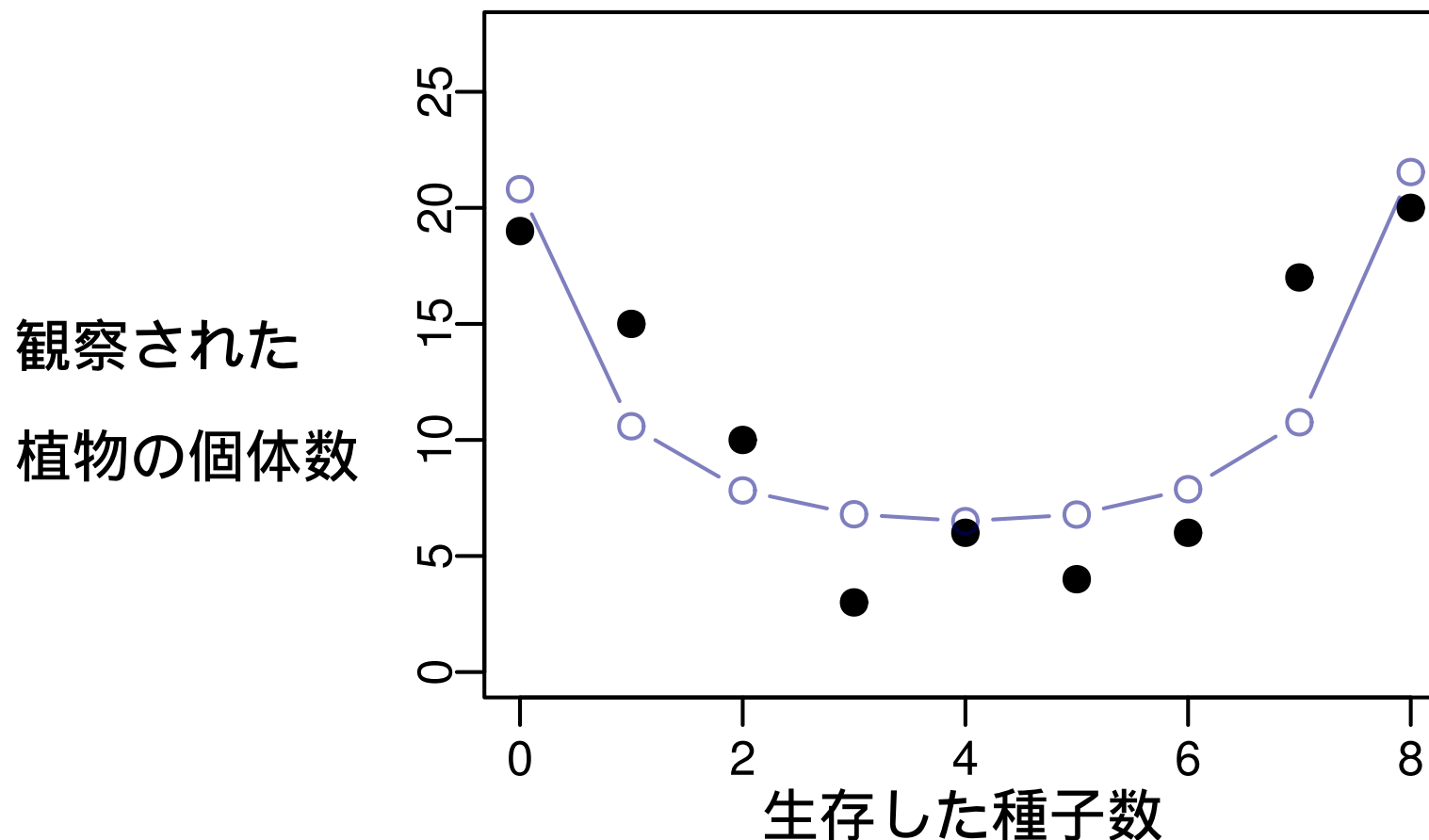
⋮

$$p(r_{100} \mid \cdots) \propto p(y_{100} \mid q(a + r_{100})) p(r_{100} \mid \sigma)$$

(Gibbs sampling するソフトウェアについては午後に解説!)

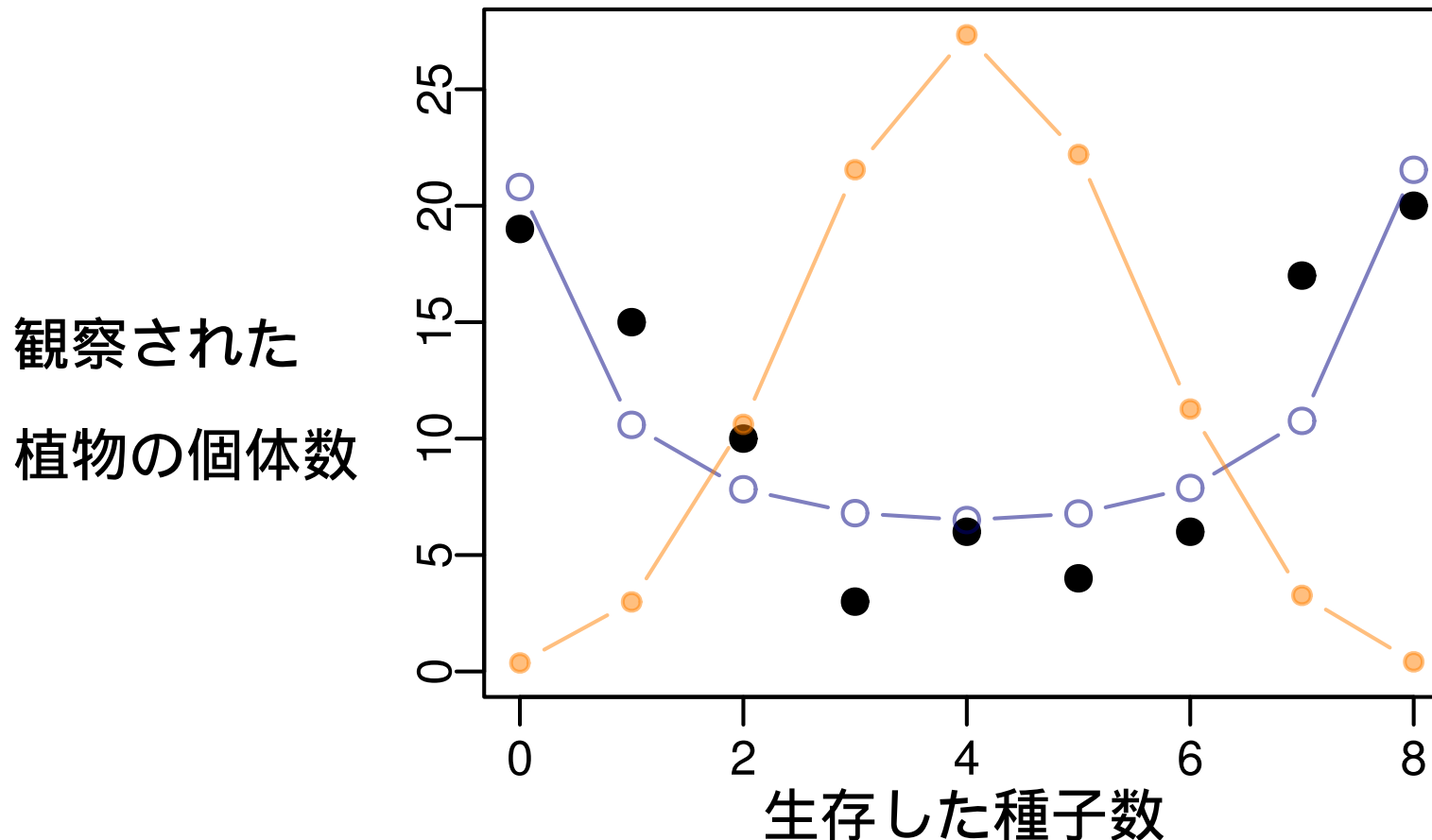
推定された事後分布に基づく予測

(推定計算方法は午後に解説!)



「個体差」を考慮することで、
少しはマシな統計モデルが作れた

ここで使った解決策: さまざまな二項分布をませる

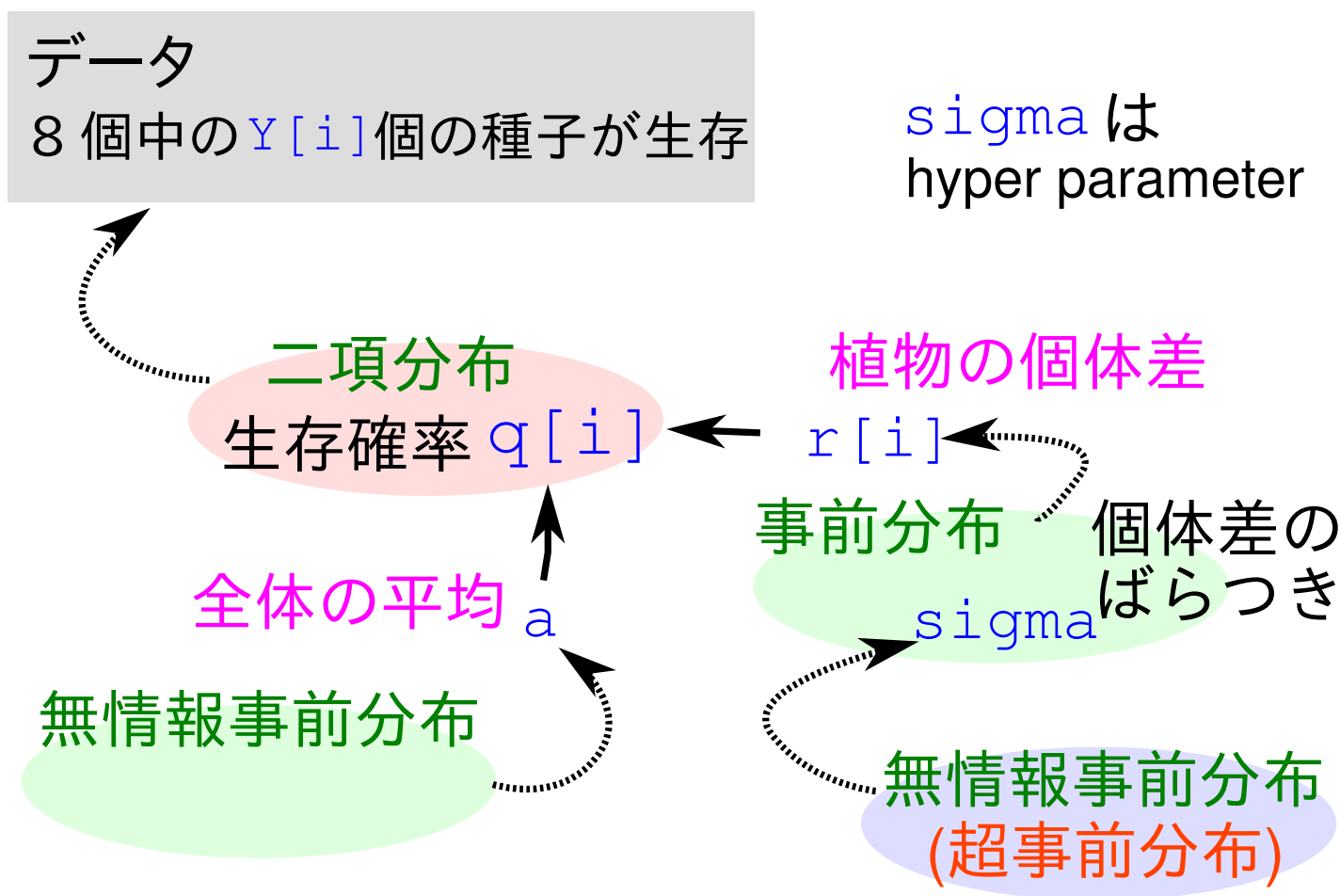


複雑な確率分布を新しく導入するのではなく
二項分布と正規分布をませることで現象を表現した

ここまでの用語の整理

- 階層ベイズモデル

$$(\text{事後分布}) \propto (\text{尤度}) \times (\text{事前分布}) \times (\text{超事前分布})$$

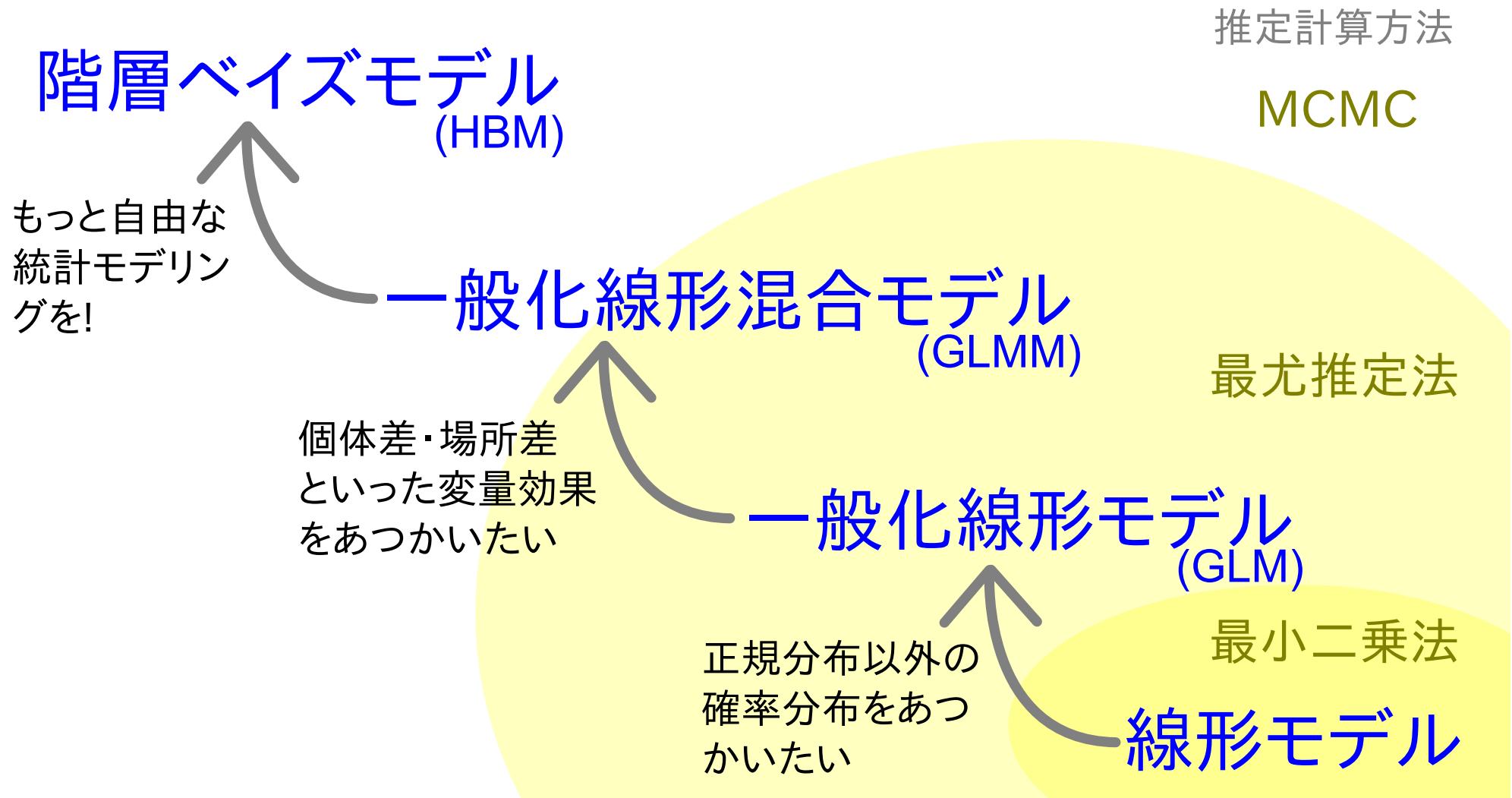


階層ベイズモデルのご利益とは?

階層ベイズモデルでないとうまく表現できない現象がある

- 複数の random effects (個体差・ブロック差・縦断的データ・.....)
- 「隠れた」状態をあつかうモデル
 - 例: 「欠側値を補う」処理
- **空間構造**ある問題も MCMC 計算で
 - 例: 「隣は似てるよ」効果 – Gaussian Random Field

線形モデルの発展



3. R と WinBUGS の使いかた

「R と WinBUGS の使いかた」の内容

1. MCMC をどんなソフトウェアで動かす?

「できあい」の Gibbs sampler あれこれ

2. WinBUGS を R で使う

R2WBwrapper 関数セットを經由して

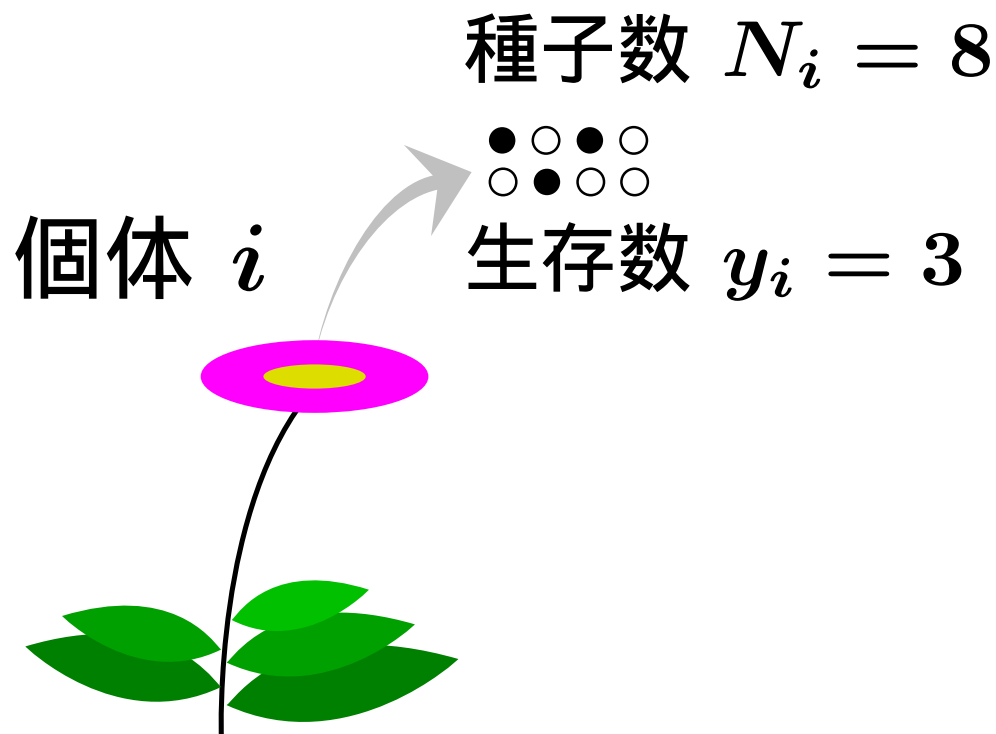
3. 「生存確率の推定」例題を WinBUGS で推定

実際に使ったコードを説明しつつ

繁殖生態学の例題: 架空植物の生存確率

- 架空植物の種子の生存を調べた
- 用語

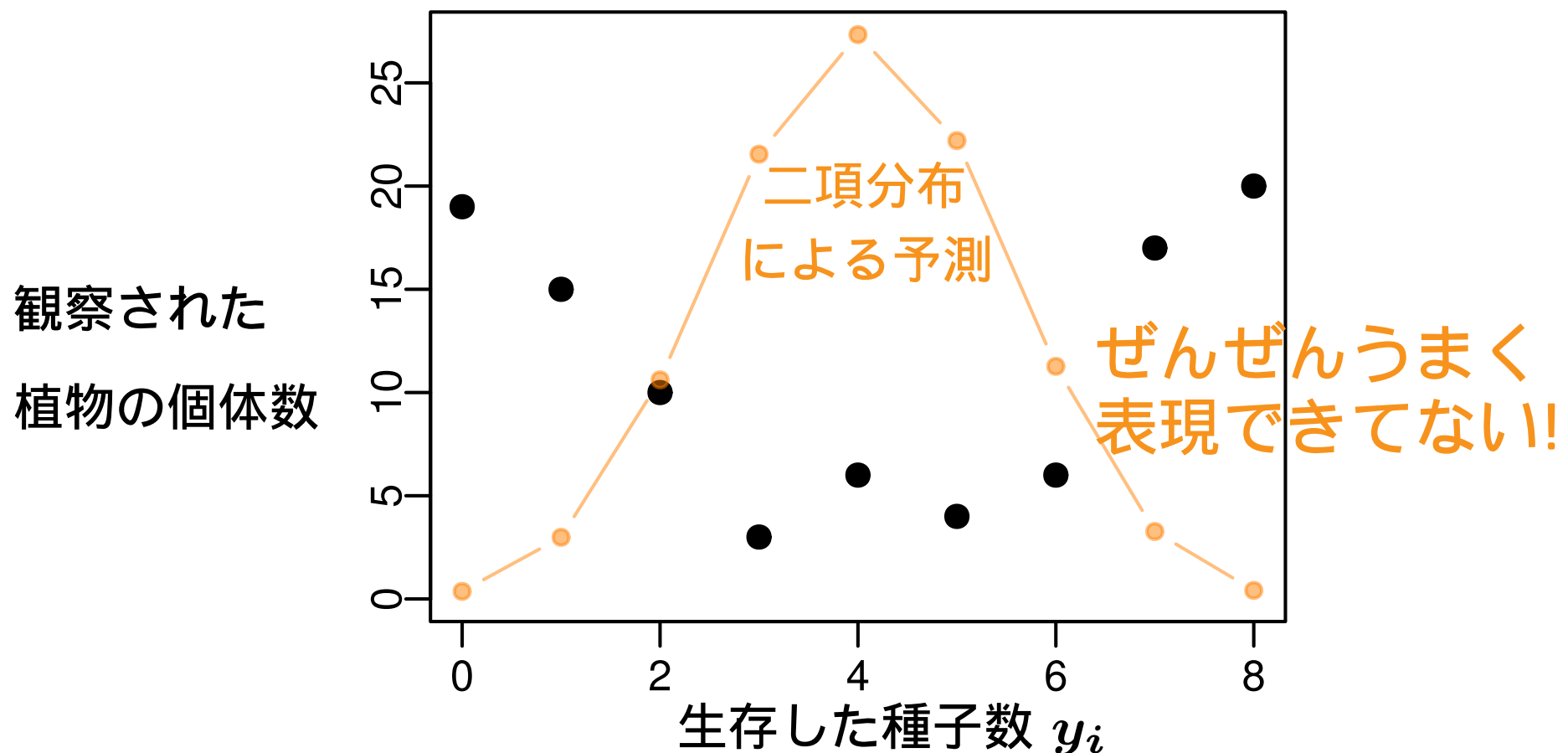
- 種子: 種子のもとになる器官 (この植物ではどの個体も **8 個** 持つとする)
- 生存: 種子が種子になること
- 生存確率: ある種子が種子になる確率



- データ: 植物 100 個体, 合計 800 種子の生存の有無を調べた
- 問: この植物の生存確率はどのように統計モデル化できるか?

また別の観測データ: 二項分布だめだめ?!

100 個体の植物の合計 800 種子中 **403 個** の生存が見られたので, 平均生存確率は 0.50 と推定されたが.....



MCMC 用のソフトウェア

階層ベイズモデリング, その手順のまとめ

- 観測データを説明できそうな確率分布を選ぶ
- その確率分布の平均・分散などのモデリング
- パラメーターの事前分布を設定する
 - 階層的な事前分布 — 個体差・場所差など
 - 無情報事前分布 — いわゆる「処理の効果」など
- モデリングできたら, 事後分布を推定する
 - 例: MCMC 計算によって事後分布からのサンプルを得る
- 事後分布を解釈する

MCMC による事後分布からのサンプリング

- **Markov Chain Monte Carlo** : 単純な乱数を **うまく** つかって「あつかいづらい」確率分布から**ランダムサンプル**を得る方法 (アルゴリズム)
- ある種のデータを解析するためには**階層ベイズモデル**が必要
- そういったベイズモデルを観測データに「あてはめ」てパラメータ推定するためには **MCMC** が役にたつ, ということにしたい (MCMC 利用法のひとつ)

「事後分布からのサンプル」って何の役にたつの？

```
> post.mcmc[, "a"] # 事後分布からのサンプルを表示
```

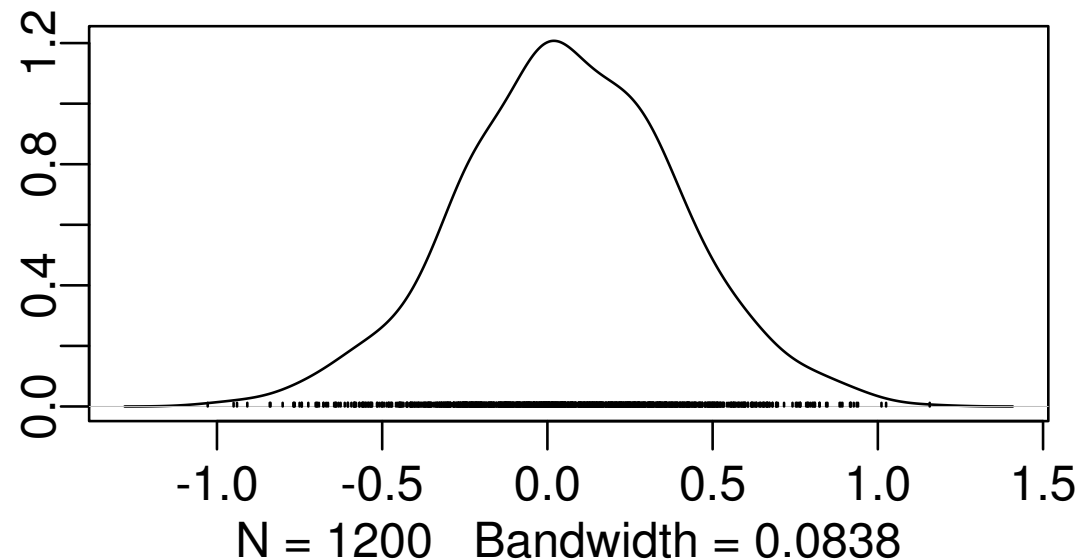
```
[1] -0.7592 -0.7689 -0.9008 -1.0160 -0.8439 -1.0380 -0.8561 -0.9837
```

```
[9] -0.8043 -0.8956 -0.9243 -0.9861 -0.7943 -0.8194 -0.9006 -0.9513
```

```
[17] -0.7565 -1.1120 -1.0430 -1.1730 -0.6926 -0.8742 -0.8228 -1.0440
```

```
... (以下略) ...
```

- これらのサンプルの平均値・中央値・95% 区間を調べることで「もと」の事後分布の概要がわかる



どのようなソフトウェアで MCMC 計算するか?

1. 自作プログラム

- 利点: 問題にあわせて自由に設計できる
- 欠点: 階層ベイズモデル用の MCMC プログラミング, けっこうめんどろ

2. R のベイズな package

- 利点: 空間ベイズ統計など便利な専用 package がある
- 欠点: 汎用性, とぼしい

3. 「できあい」の Gibbs sampler ソフトウェア

- 利点: 「原因 → 結果」型の階層ベイズモデルは得意
- 欠点: それ以外の問題に応用するには.....

統計ソフトウェア R

<http://www.r-project.org/>

- いろいろな OS で使える **free software**
- 使いたい機能が充実している
- **作図**機能も強力
- S 言語による **プログラミング**可能
- よい教科書が出版されつつある
 - 多すぎて紹介できません
 - **ネット**上のあちこちにも資料など



R だけで何とかなる: 経験ベイズ法 (1)

今回の例題の事後分布

$$p(a, \{r_i\}, \sigma \mid \text{データ}) \propto \prod_{i=1}^{100} p(y_i \mid q(a + r_i)) p(a) p(r_i \mid \sigma) p(\sigma)$$

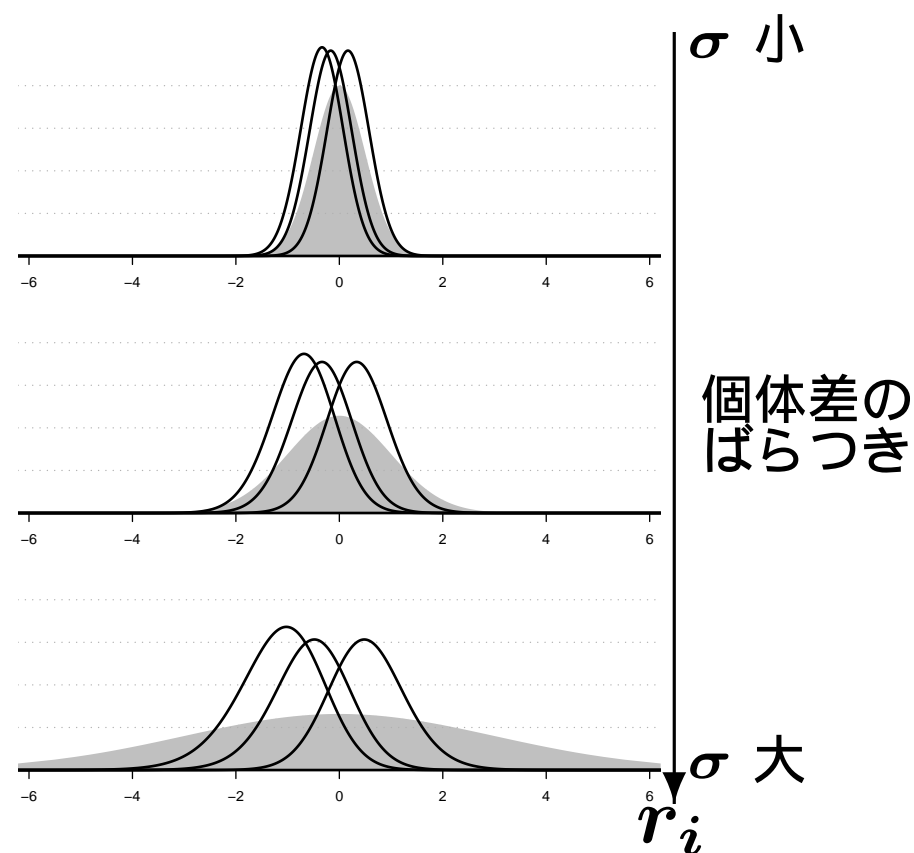
積分で「個体差」 r_i を消して, 周辺尤度を定義する

$$(a, \sigma \text{の尤度}) = \prod_{i=1}^{100} \int_{-\infty}^{\infty} p(y_i \mid q(a + r_i)) p(r_i \mid \sigma) dr_i$$

これを最大化する \hat{a} と $\hat{\sigma}$ を推定すればよい

R だけで何とかなる: 経験ベイズ法 (2)

- 周辺尤度最大化って何をやっていることになるのだろうか?
 - 最も良さそうな「『個体差』の幅」を探している
 - 同時に全個体共通の a も探している
- これは一般化線形混合モデル (GLMM) の最尤推定と同じ
- R での対処例: `library(glmmML)` の `glmmML()` 関数



`glmmML(cbind(y, N - y) ~ 1, family = binomial, などなど指定)`

R だけで何とかできる? ちょっと無理かも.....

GLMM は階層ベイズモデルの一部

- R にはいろいろな GLMM 推定関数が準備されている
 - `library(glmML)` の `glmML()`
 - `library(lme4)` の `lmer()`
 - `library(nlme)` の `nlme()` (正規分布のみ)
 - `library(MCMCglmm)` の `MCMCglmm()`
- しかしながら「めんどろな状況」では..... ちょっと無理があるかも (推定計算がうまくいかない)

「めんどうな状況」では MCMC が有用である

この例題は簡単すぎるので経験ベイズ法で対処できる
しかし、現実的なデータ解析のための
複雑な階層ベイズモデルは MCMC で推定するほかない

- 複数の random effects (個体差・ブロック差・縦断的データ・.....)
- 「隠れた」状態をあつかうモデル
 - 例: 「欠側値を補う」処理
- **空間構造**ある問題も MCMC 計算で
 - 例: 「隣は似てるよ」効果 – Gaussian Random Field

そこで “BUGS” な汎用 Gibbs sampler たち

(じつは Gibbs sampling 以外の手法も使ってるようなのだが.....)

- BUGS でベイズモデルを記述できるソフトウェア (と久保の蛇足な論評):
 - **WinBUGS** — 評: 「とりあえず, これしかない」って現状?
 - **OpenBUGS** — 評: ココロザシは高いんでしょうけど, どうなってんの?
 - **JAGS** — 評: じりじりと発展中, がんばってください
- リンク集:
<http://hosho.ees.hokudai.ac.jp/~kubo/ce/BayesianMcmc.html>

BUGS 言語: ベイズモデルを記述する言語

- Spiegelhalter et al. 1995. BUGS: Bayesian Using Gibbs Sampling version 0.50.

```
model { # BUGS コードで定義された階層ベイズモデルの例
  for (i in 1:N.sample) {
    Y[i] ~ dbin(q[i], N[i])
    logit(q[i]) <- a + r[i]
  }
  a ~ dnorm(0, 1.0E-4)
  for (i in 1:N.sample) {
    b[i] ~ dnorm(0, tau)
  }
  tau <- 1 / (sigma * sigma)
  sigma ~ dunif(0, 1.0E+4)
}
# あとで説明
```

君臨しつづける WinBUGS 1.4.3 (あとで詳しく説明)

- おそらく世界でもっともよく使われている Gibbs sampler
- **BUGS 言語の実装**
- 2004-09-13 に最新版 (ここで開発停止 → OpenBUGS)
- ソースなど非公開, 無料, ユーザー登録**不要**
- Windows バイナリーとして配布されている
 - Linux 上では WINE 上で動作
 - MacOS X 上でも Darwine など駆使すると動くらしい
- へんな GUI (Linux ユーザーの偏見)
- **R ユーザーにとっては R2WinBUGS が快適 (後述)**

WinBUGS は Gibbs sampling しているのか?

よくある質問: WinBUGS は Gibbs sampling してるの?

- 事前分布・尤度の組みあわせによって、サンプリング方法を自動的に変更している
 - 共役事前分布がない場合は、さまざまな数値的な方法を使う
- ユーザーはそのあたりをまったく指定する必要なし (指定できない)

くわしくは WinBUGS のマニュアル読みましょう

<http://www.google.com/search?q=winbugs+user+manual>

DJ Spiegelhalter, A Thomas, NG Best, D Lunn. 2003. WinBUGS version 1.4 user manual. MRC Biostatistics Unit, Cambridge.

Continuous target distribution

Method

Conjugate

Direct sampling using standard algorithms

Log-concave

Derivative-free adaptive rejection sampling (Gilks, 1992)

Restricted range

Slice sampling (Neal, 1997)

Unrestricted range

Current point Metropolis

Discrete target distribution

Method

Finite upper bound

Inversion

Shifted Poisson

Direct sampling using standard algorithm

GPL な WinBUGS めざして: OpenBUGS 3.1.2

- WinBUGS の後継プロジェクト
- ソースは公開しているが
 - Component Pascal で実装
 - ソースを読んだりするには
BlackBox Component Builder が必要
- Windows バイナリ配布 , Linux でもなんとか使えた
- 長いあいだ停滞していたが.....
- 最近になって少しずつ改訂版がリリース

R な Gibbs sampler: JAGS 2.2.0

- R core team のひとり Martyn Plummer さんが開発
 - Just Another Gibbs Sampler
- C++ で実装されている , 誰でもコンパイルできる
 - R がインストールされていることが必要
 - 拡張 plugin を簡単に書ける設計になっている
- じっくりと開発進行中
- R から使う: `library(rjags)`

WinBUGS を R で使う

今回説明する WinBUGS の使いかた (概要)

- WinBUGS を R から使う
 - R から WinBUGS をよびだし「このベイズモデルのパラメーターの事後分布をこういうふうに MCMC 計算してね」と指示する
 - WinBUGS が得た事後分布からのサンプルセットを R がうけとる
- R の中では `library(R2WinBUGS)` package を使う
R2WBwrapper 関数 (久保作) を使う

なんで WinBUGS を R 経由で使うの?

- WinBUGS のユーザーインターフェイスを使うのがめんど
うだから
- どうせ解析に使うデータは R で準備するから
- どうせ得られた出力は R で解析・作図するから
- R には R2WinBUGS という (機能拡張用) package があつて, R から WinBUGS を使うしくみが準備されてるから
 - R 上で `install.packages("R2WinBUGS")` でインストールできる

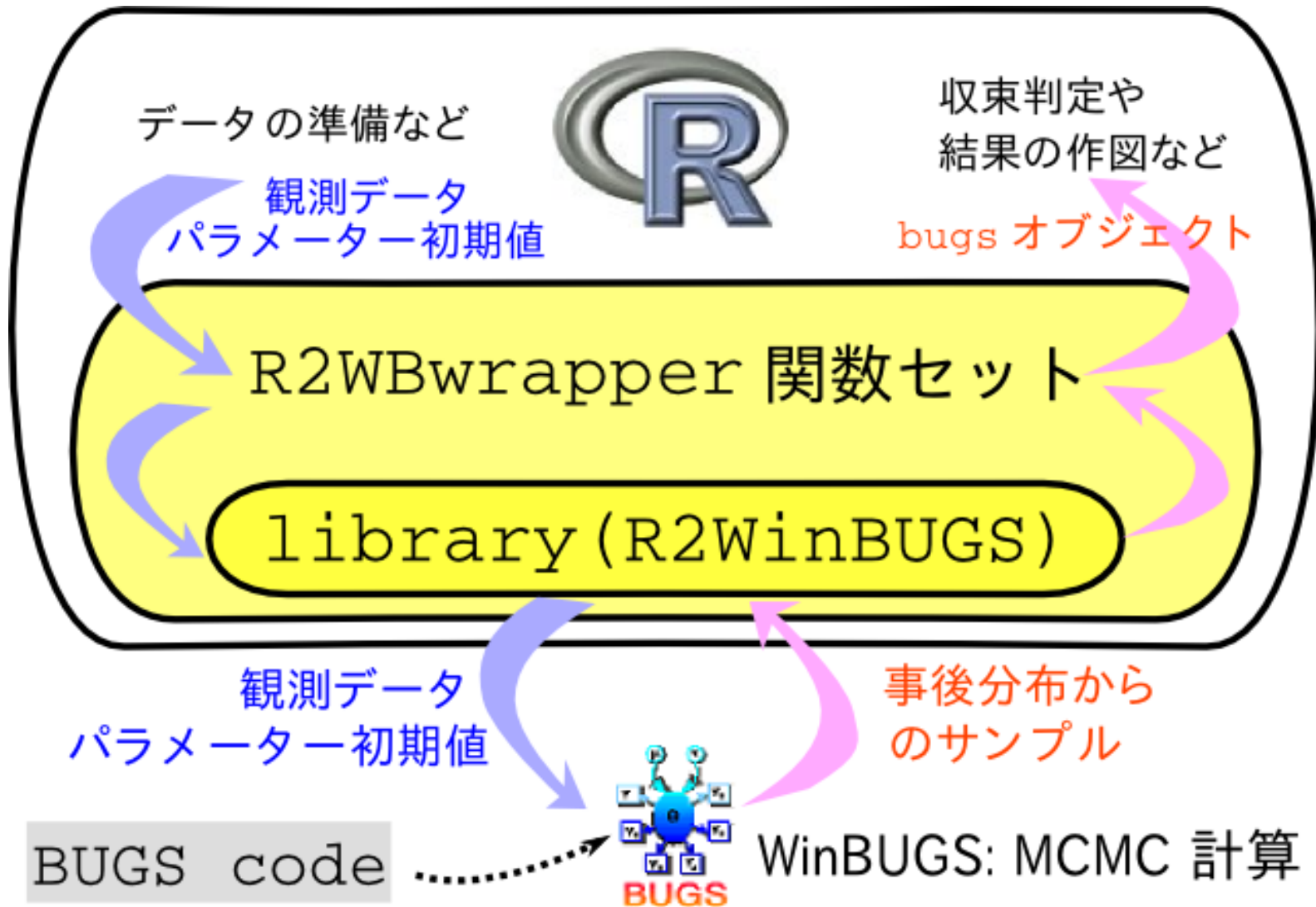
なんで R2WinBUGS をラップして使うの?

- R2WinBUGS の GUI を使うのがめんどうだから
 - モデルをちょっと変更したらあちこち書きなおさないといけない
 - R2WBwrapper を使うとそのあたりがかなりマシになる
- Linux と Windows で「呼びだし」方法がびみょーに異なるため
 - R2WBwrapper を使うと自動的に OS にあわせた WinBUGS よびだしをする

R2WBwrapper 経由で WinBUGS を使う (1)

1. BUGS 言語でかかれた model ファイルを準備する
2. R2WBwrapper 関数を使う R コードを書く
3. R 上で 2. を実行
4. 出力された結果が bugs オブジェクトで返される
5. これを `plot()` したり `summary()` したり.....オブジェクトに変換して, いろいろ事後分布の図なんかを描いてみたり.....

R2WBwrapper 経由で WinBUGS を使う (2)



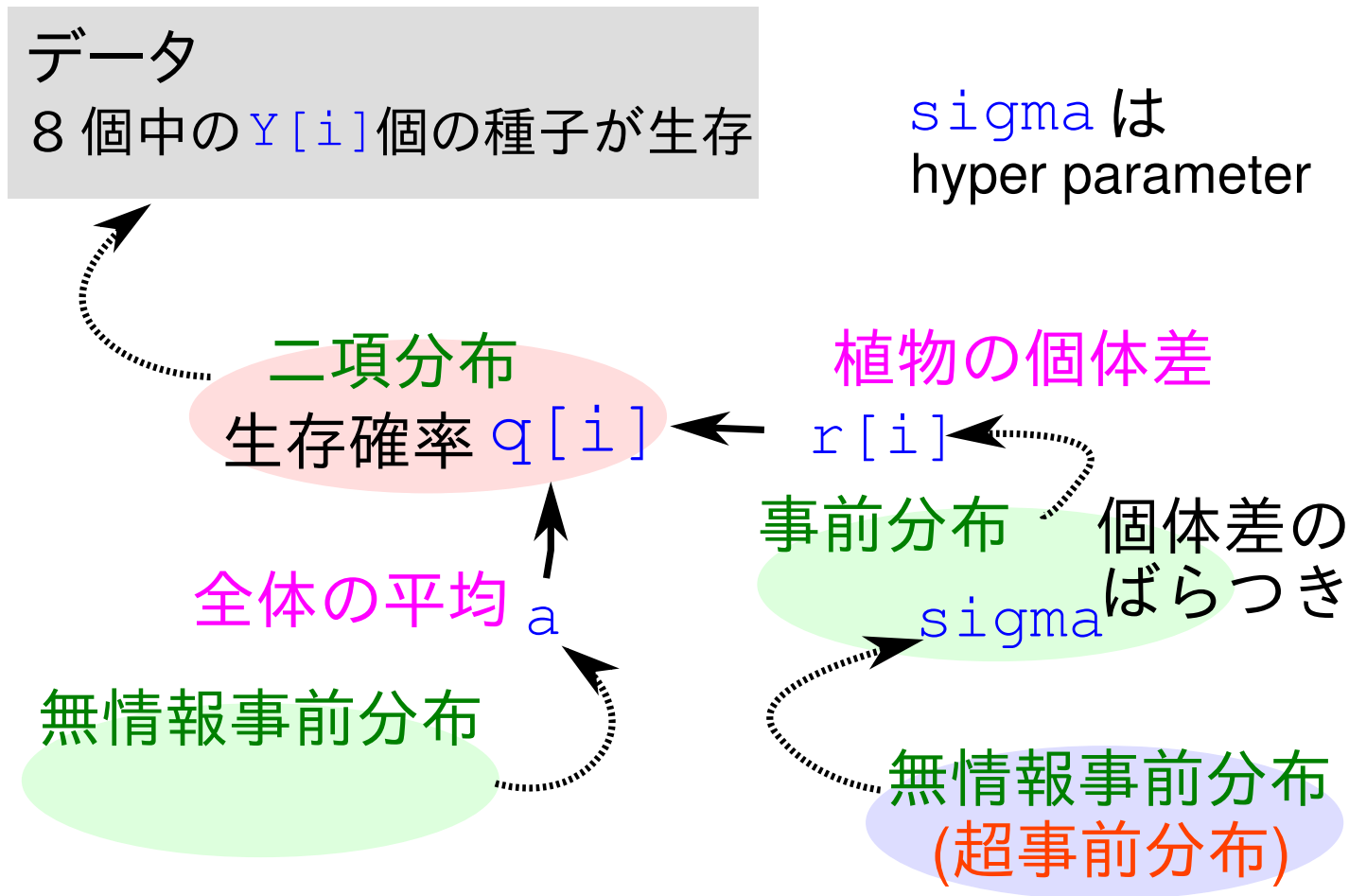
「生存確率の推定」例題を

WinBUGS で推定

「生存確率の推定」例題を WinBUGS に推定させる手順

1. 生存確率の階層ベイズモデルの構築する
2. それを BUGS 言語でかく (`model.bug.txt`)
3. R2WBwrapper 関数を使って R コードを書く (`runbugs.R`)
4. R 上で `runbugs.R` を実行 (`source(runbugs.R)` など)
5. 出力された結果が `bugs` オブジェクトで返される

生存確率の階層ベイズモデルってどんなでしたっけ？



$$p(a, \{r_i\}, \sigma \mid \text{データ}) \propto \prod_{i=1}^{100} p(\text{データ} \mid q(a + r_i)) p(a) p(r_i \mid \sigma) p(\sigma)$$

事前分布の設定方法

- 階層的な (hierarchical) 事前分布にする
 - random effects 的な個体差・場所差
- 無情報 (non-informative) 事前分布にする
 - 切片や説明変数の係数など fixed effects 的なパラメーター
- 主観的な (subjective) 事前分布にする
 - あまりおすすりめできない
 - (反復測定していないときの) 測定時のエラーとか

生存確率の階層ベイズモデルを BUGS 言語で

ファイル `model.bug.txt` の内容 (一部簡略化)

```
model{
  for (i in 1:N.sample) {
    Y[i] ~ dbin(q[i], 8) # 観測値との対応
    logit(q[i]) <- a + r[i] # 生存確率 q[i]
  }
  a ~ dnorm(0, 1.0E-4) # 「切片」
  for (i in 1:N.sample) {
    r[i] ~ dnorm(0, tau) # 個体差
  }
  tau <- 1 / (sigma * sigma) # tau = 1 / variance
  sigma ~ dunif(0, 1.0E+4) # 個体差のばらつき
}
```

BUGS 言語について, いくつか

- BUGS 言語は普通の意味でのプログラミング言語ではない
 - 「式」を列挙しているだけ, と考える
 - 「式」の並び順を変えても計算結果は (ほぼ) 変わらない
- 各パラメーターは二種類の **node** それぞれで一度ずつ定義できる (二度以上は定義できない)
 1. ~ stochastic node
 2. <- deterministic node

R2WBwrapper な R コード runbugs.R (前半部)

観測データの設定

```
d <- read.csv("data.csv") # 観測データよみこみ
```

```
clear.data.param() # いろいろ初期化
```

```
set.data("N.sample", nrow(d)) # データ数
```

```
set.data("Y", d$Y) # 生存種子数
```

R2WBwrapper な R コード runbugs.R (後半部)

パラメーターの初期値の設定など

```
set.param("a", 0)          # 「切片」
set.param("sigma", 1)     # 個体差のばらつき
set.param("r", rep(0, N.sample)) # 個体差
set.param("q", NA)       # 生存確率

post.bugs <- call.bugs(    # WinBUGS よびだし
  file = "model.bug.txt",
  n.iter = 1300, n.burnin = 100, n.thin = 3
)
```

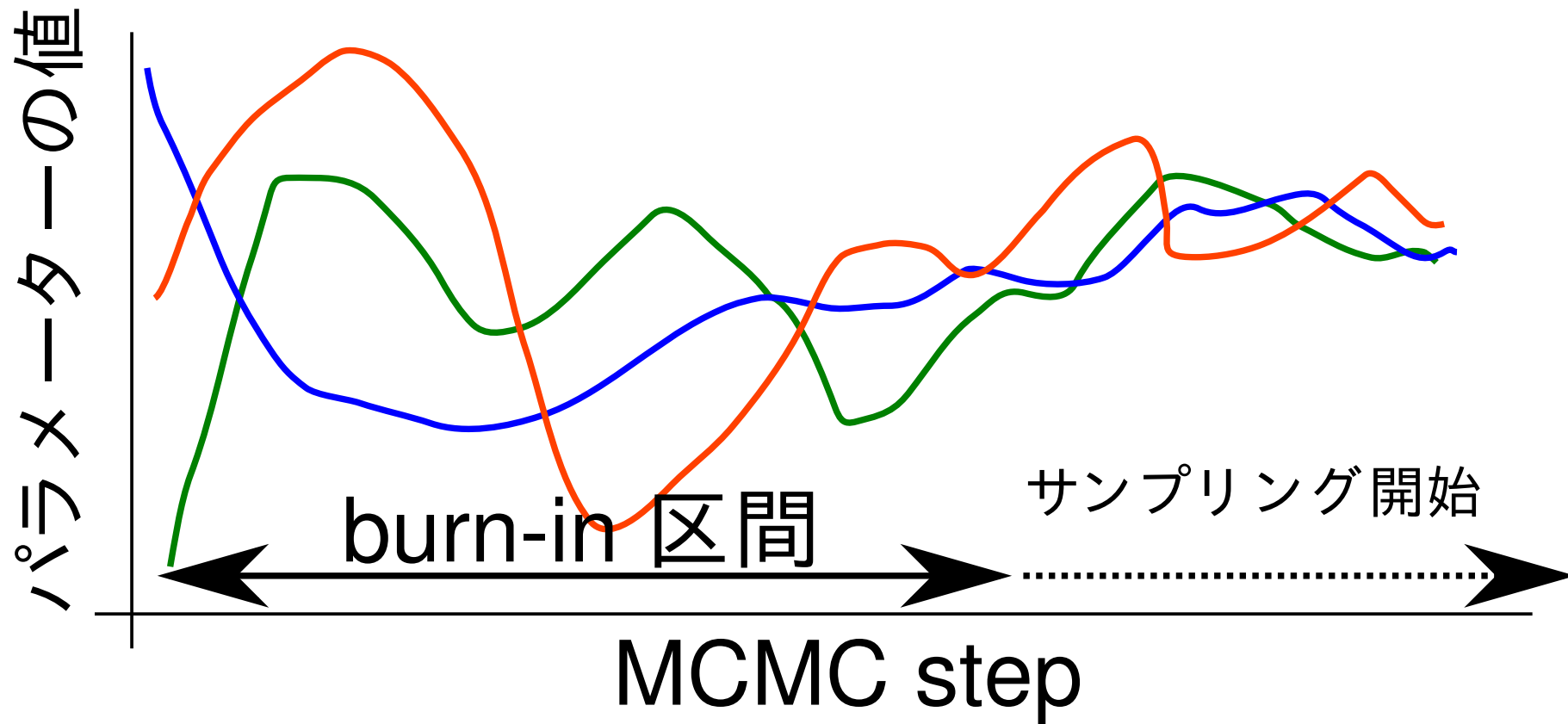

WinBUGS に指示した事後分布のサンプリング

```
post.bugs <- call.bugs(      # WinBUGS よびだし
  file = "model.bug.txt",
  n.iter = 2000, n.burnin = 1000, n.thin = 5
)
```

- じつは default では独立に (並列に)
(n.chains = 3) MCMC sampling せよと指定されている (収束性をチェックするため)
- ひとつの chain の長さは 2000 step (n.iter = 2000)
- 最初の 1000 step は捨てる (n.burnin = 1000)
- 1001 から 2000 step まで 5 step おきに値を記録する (n.thin = 5)

このあたりの設定はデータ・統計モデルによって変わる

“burn-in”: MCMC の最初のほうを捨てる

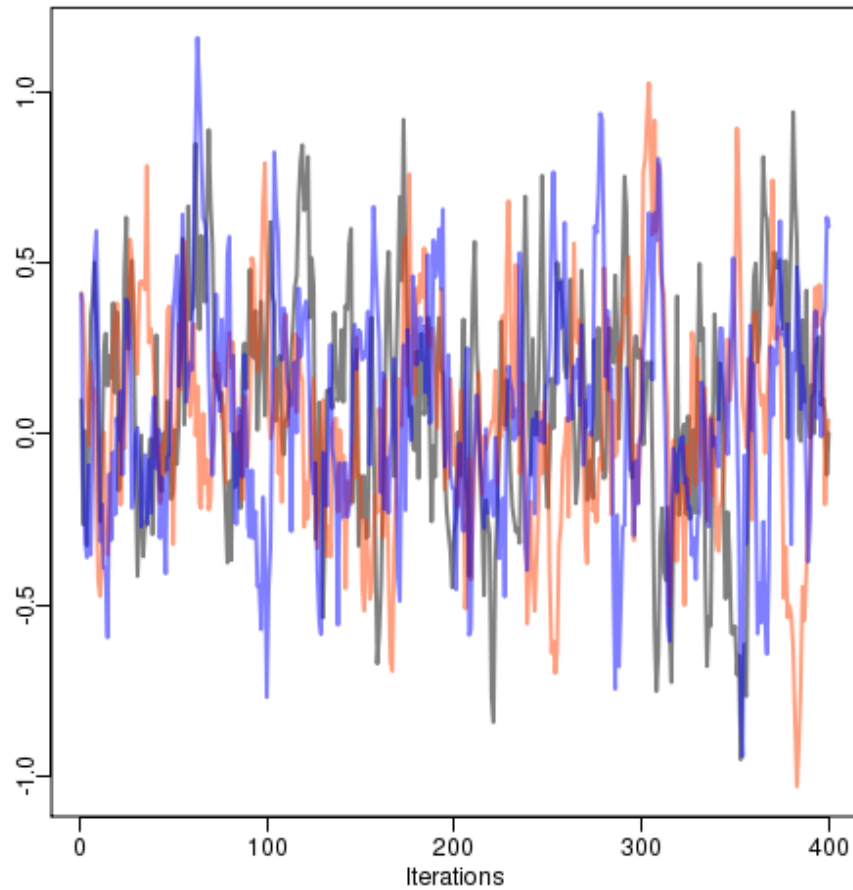


で、実際に動かすには?

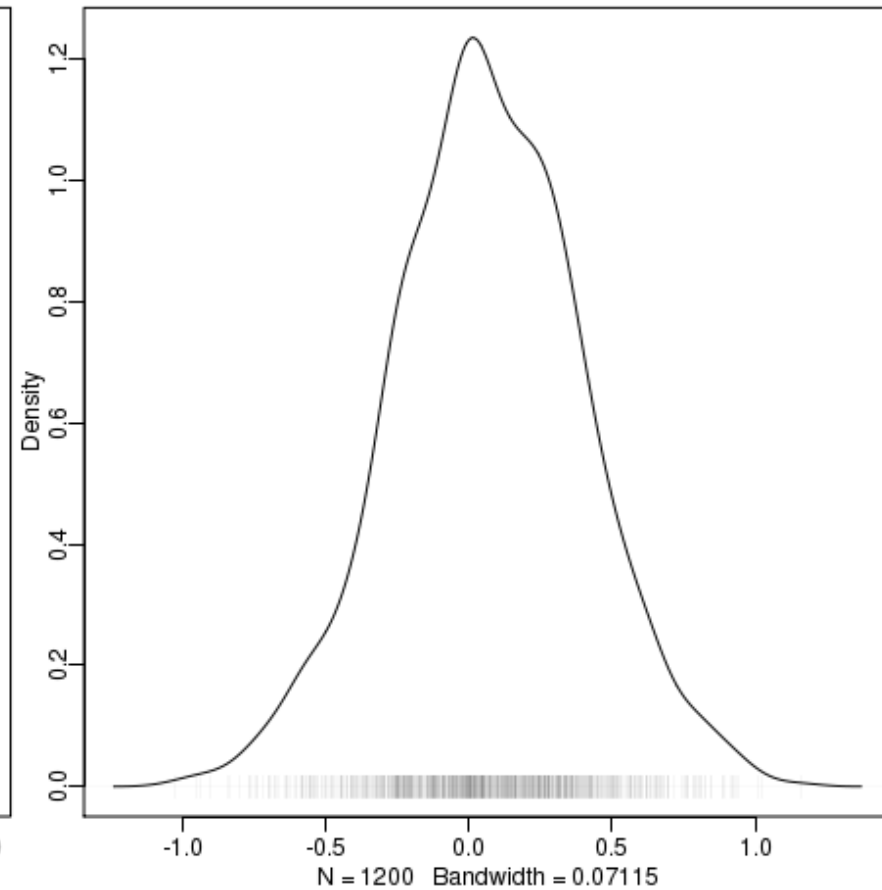
- たとえば, R 上で `source("runbugs.R")` とか
- すると WinBUGS が起動して MCMC sampling をはじめる
- この例題は簡単なのですぐに計算が終了する (WinBUGS 内で図などが表示される)
- 手動で WinBUGS を終了する
- すると WinBUGS が得た結果が R にわたされ, `post.bugs` というオブジェクトにそれが格納される

事後分布のサンプルを R で調べる

a のサンプリングの様子



a の事後確率密度の推定



bugs オブジェクトの `post.bugs` を調べる (1)

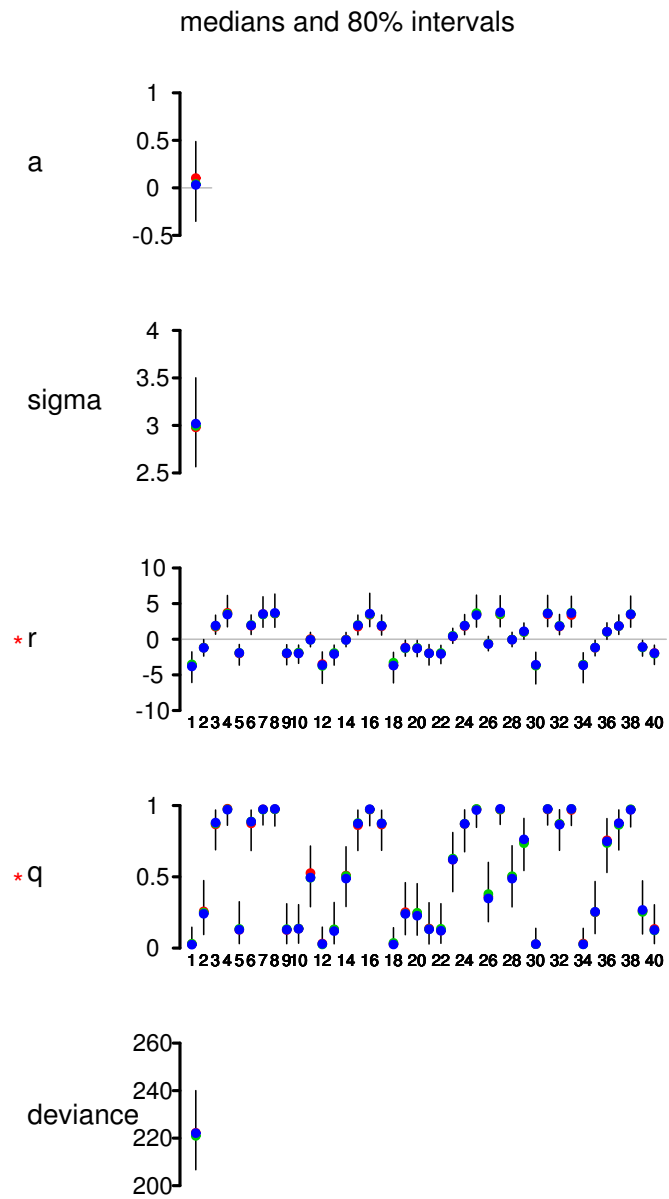
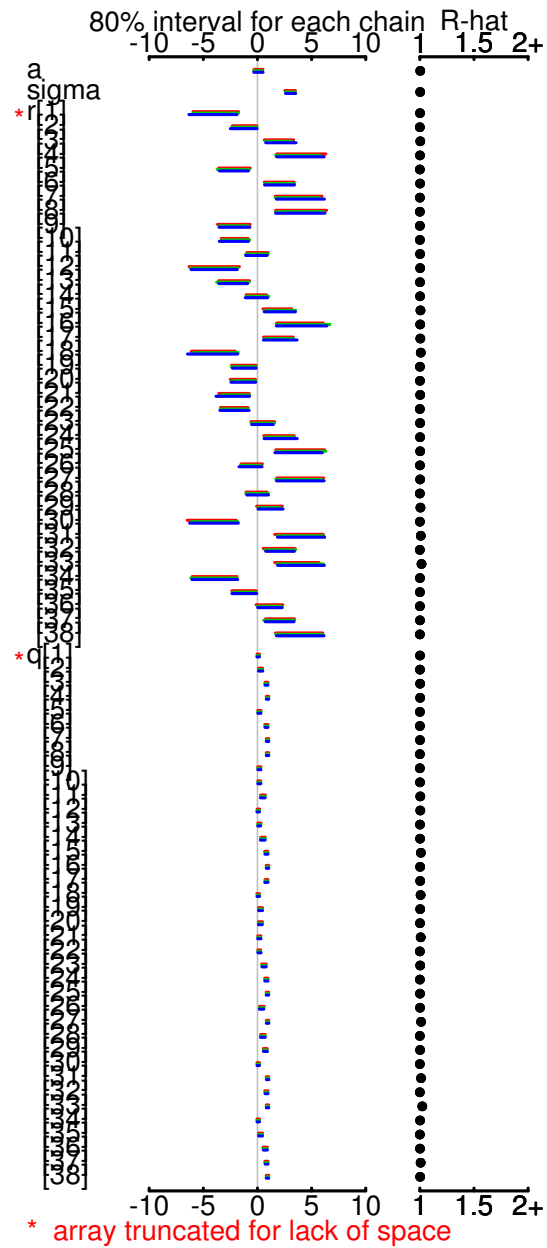
- `plot(post.bugs)` → 次のページ, 実演表示
- `R-hat` は Gelman-Rubin の収束判定用の指数

- $$\hat{R} = \sqrt{\frac{\hat{\text{var}}^+(\psi|y)}{W}}$$

- $$\hat{\text{var}}^+(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

- Gelman et al. 2004. Bayesian Data Analysis. Chapman & Hall/CRC

/kubo/public_html/stat/2010/ism/winbugs/model.bug.txt", fit using WinBUGS, 3 chains, each with 1300 iterator



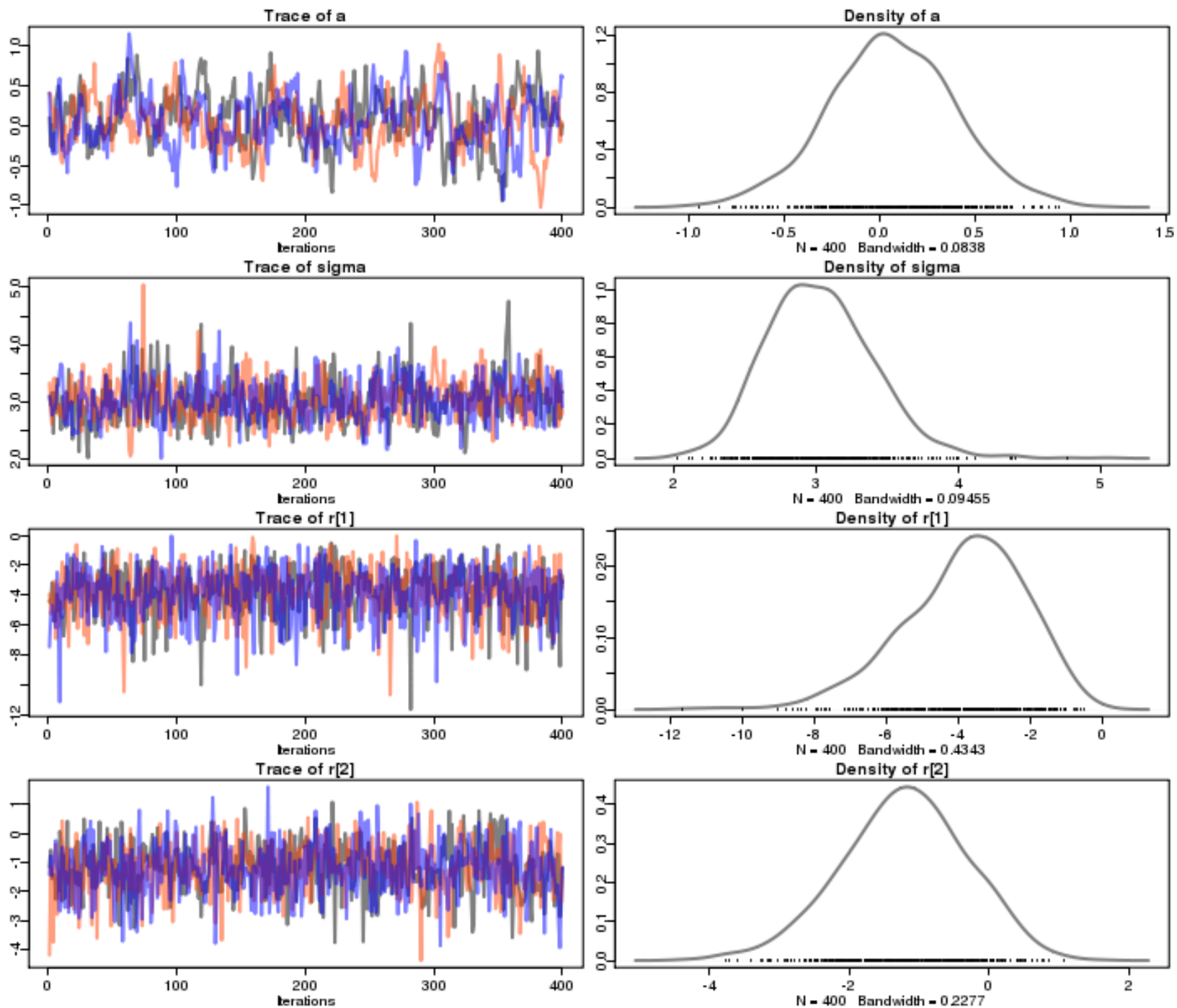
bugs オブジェクトの `post.bugs` を調べる (2)

- `print(post.bugs, digits.summary = 3)`
- 事後分布の 95% 信頼区間などが表示される

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
a	0.031	0.357	-0.718	-0.187	0.041	0.268	0.682	1.034	72
sigma	3.060	0.376	2.365	2.807	3.029	3.288	3.830	1.002	1200
r[1]	-3.890	1.903	-8.238	-4.918	-3.514	-2.546	-1.174	1.001	1200
r[2]	-1.190	0.905	-3.137	-1.763	-1.159	-0.559	0.438	1.007	290
r[3]	2.062	1.128	0.185	1.296	1.931	2.730	4.611	1.002	1200
r[4]	3.985	1.860	1.058	2.635	3.745	5.105	8.520	1.021	130
r[5]	-2.049	1.077	-4.458	-2.679	-1.971	-1.276	-0.255	1.008	270
r[6]	1.995	1.061	0.137	1.266	1.922	2.629	4.300	1.002	900
r[7]	3.886	1.765	1.144	2.664	3.583	4.894	8.223	1.008	320
r[8]	3.862	1.763	1.142	2.590	3.591	4.814	7.993	1.011	330
r[9]	-2.093	1.136	-4.532	-2.788	-1.978	-1.313	-0.130	1.003	540
r[10]	-1.993	1.082	-4.358	-2.631	-1.905	-1.250	-0.158	1.000	1200
r[11]	-0.049	0.786	-1.654	-0.555	-0.032	0.466	1.462	1.006	320
r[12]	-3.849	1.788	-8.204	-4.874	-3.547	-2.598	-1.144	1.001	1200
r[13]	-2.005	1.115	-4.593	-2.640	-1.908	-1.254	-0.069	1.001	1200

mcmc.list クラスに変換して作図

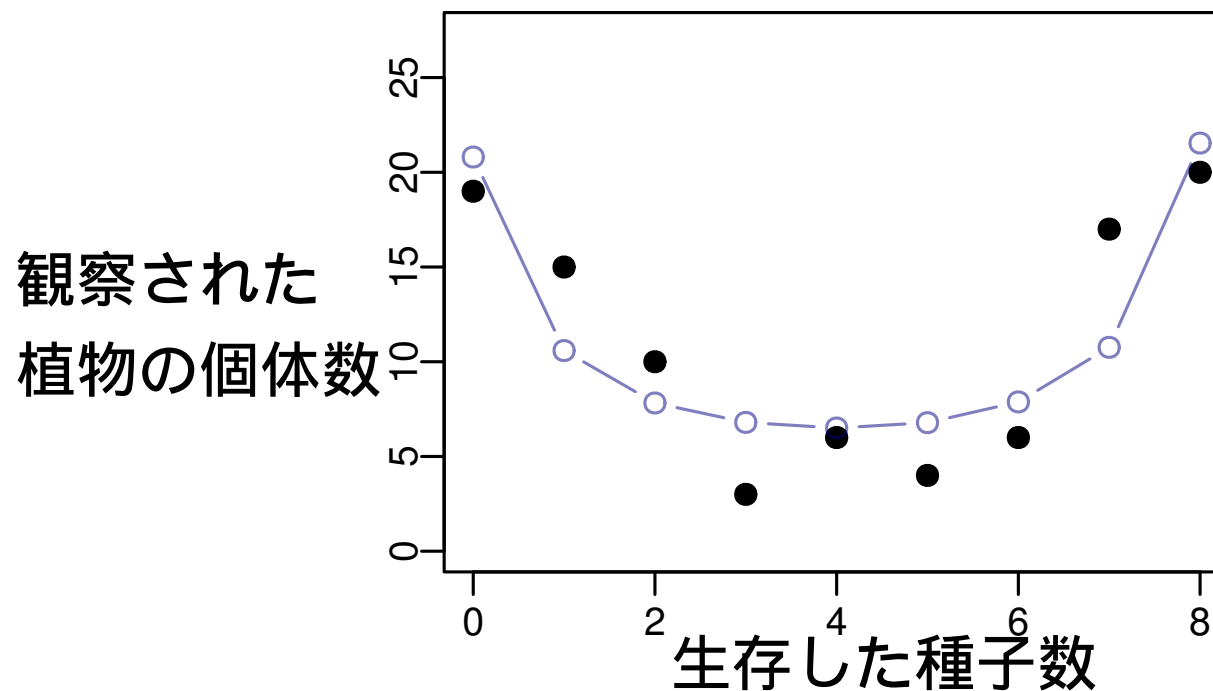
- `post.list <- to.list(post.bugs)`
- `plot(post.list[,1:4,], smooth = F)`
→ 次のページ, 実演表示



mcmc クラスに変換して作図

- `post.mcmc <- to.mcmc(post.bugs)`
- これは `matrix` と同じようにあつかえるので、作図に便利

例: 推定された事後分布に基づく予測



「R と WinBUGS の使いかた」のまとめ

1. MCMC をどんなソフトウェアで動かす?

まあ, WinBUGS + R が無難ではないでしょうか

2. WinBUGS を R で使う

R2WBwrapper 関数セットを經由して

3. 「結実確率の推定」例題を WinBUGS で推定

準備するファイル: model.bug.txt, runbugs.R

結果を R 内で解析・作図・変換する

公開講座に参加していただき、ありがとうございました

<http://goo.gl/Yvz50>

- 本日の例題のデータなどは上記 URL のページからダウンロードできます
 - この公開講座のページからリンクされています
- 投影資料 (修正を反映させていったもの) もダウンロードできます