漁業統計検討会 (清水)

「統計モデリングセミナー」 (2012 年 12 月) 投影資料

全部で 7 回中の 4b 回目

階層ベイズモデルの基礎

WinBUGS を使ってみる

久保拓弥 kubo@ees.hokudai.ac.jp

http://goo.gl/OyB2k

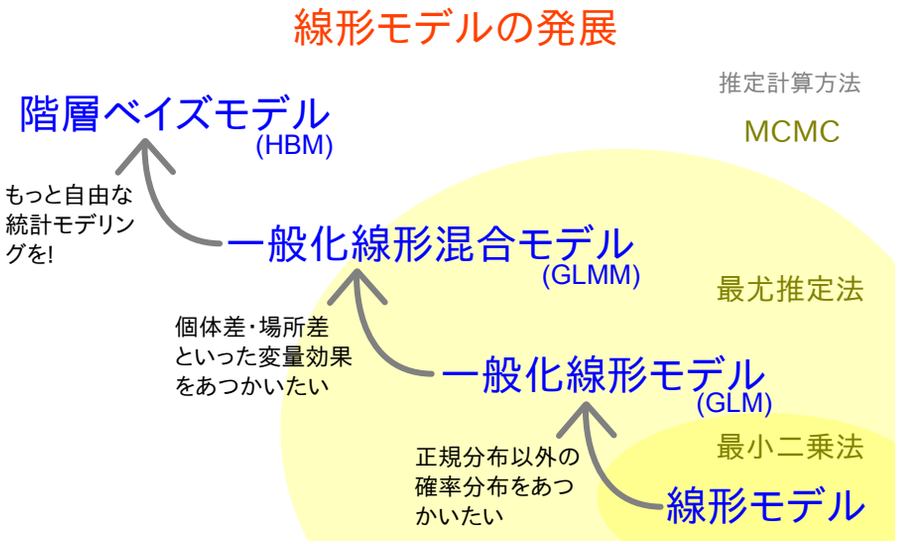
今回のハナシ

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

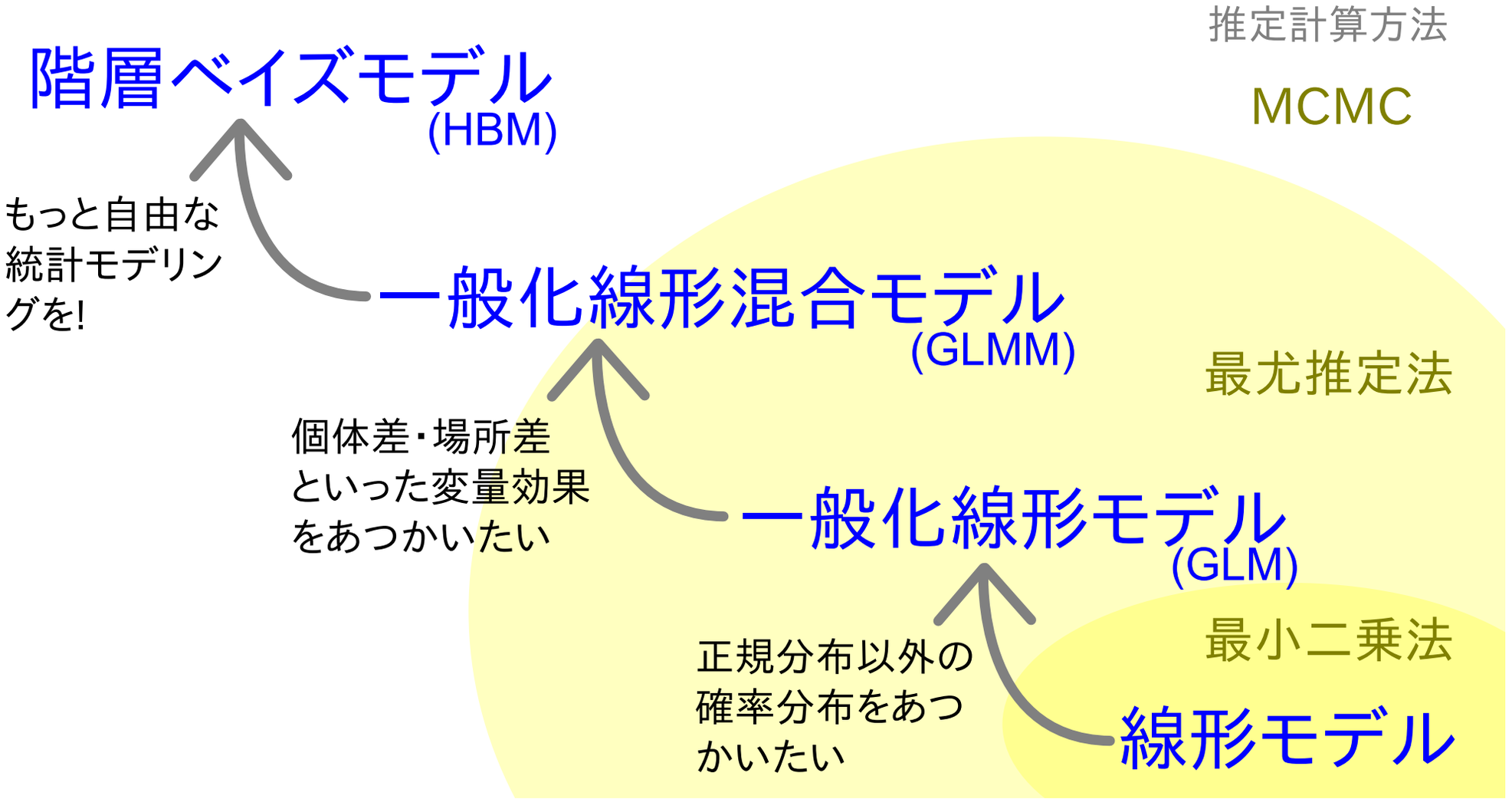
- 統計モデルは尤度であてはまりの良さを調べられる
  - 一番あてはまりが良いところをさがすのが最尤推定法
  - 尤度に比例するパラメーターの確率分布を推定するのが **MCMC**
- 現実的なデータ解析では個体差など **random effects** を考慮する必要あり
  - 単純な GLM ではそういうばらつきを表現できない
  - 階層ベイズモデル!

今回，説明しようとすること

- パラメーターをどうやって推定するか?
  - 最尤推定 $\longrightarrow$ **Markov chain Monte Carlo (MCMC)**
- 一般化線形モデル $\longrightarrow$ 階層ベイズモデル
  - より現実的・実戦的なデータ解析のために



線形モデルの発展


推定計算方法
階層ベイズモデル
(HBM)
MCMC
もっと自由な
統計モデリン
グを!
一般化線形混合モデル
(GLMM)
最尤推定法
個体差・場所差
といった変量効果
をあつかいたい
一般化線形モデル
(GLM)
最小二乗法
正規分布以外の
確率分布をあつ
かいたい
線形モデル

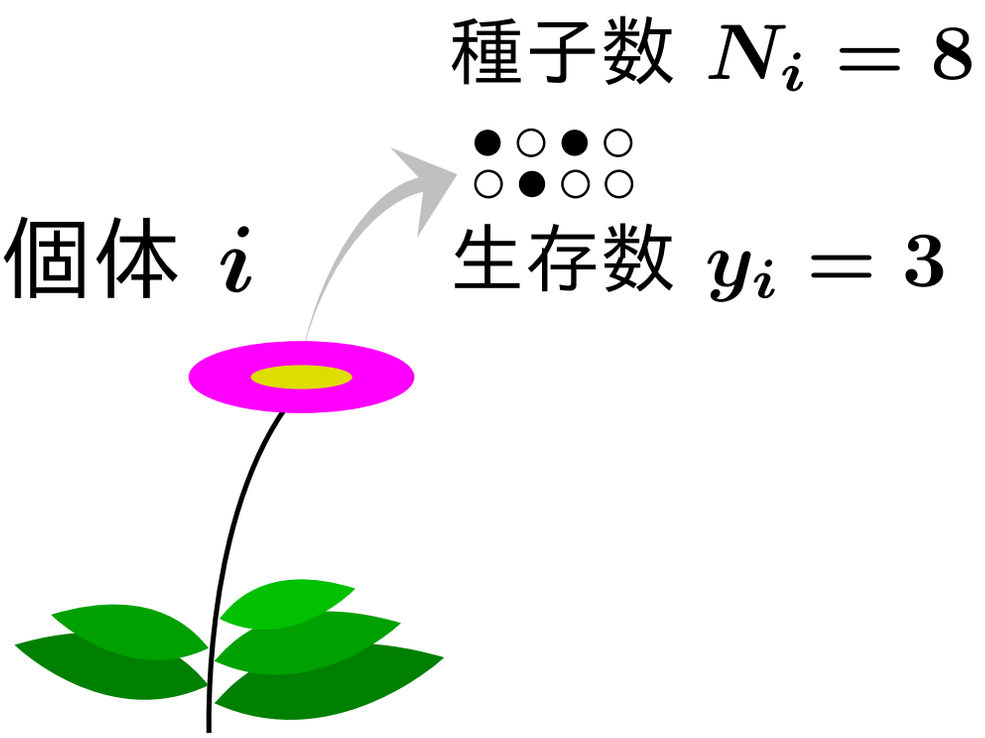
前回までのあらすじ

GLM ではうまくいかない状況

階層ベイズモデルで対処しよう

繁殖生態学の例題: 架空植物の生存確率

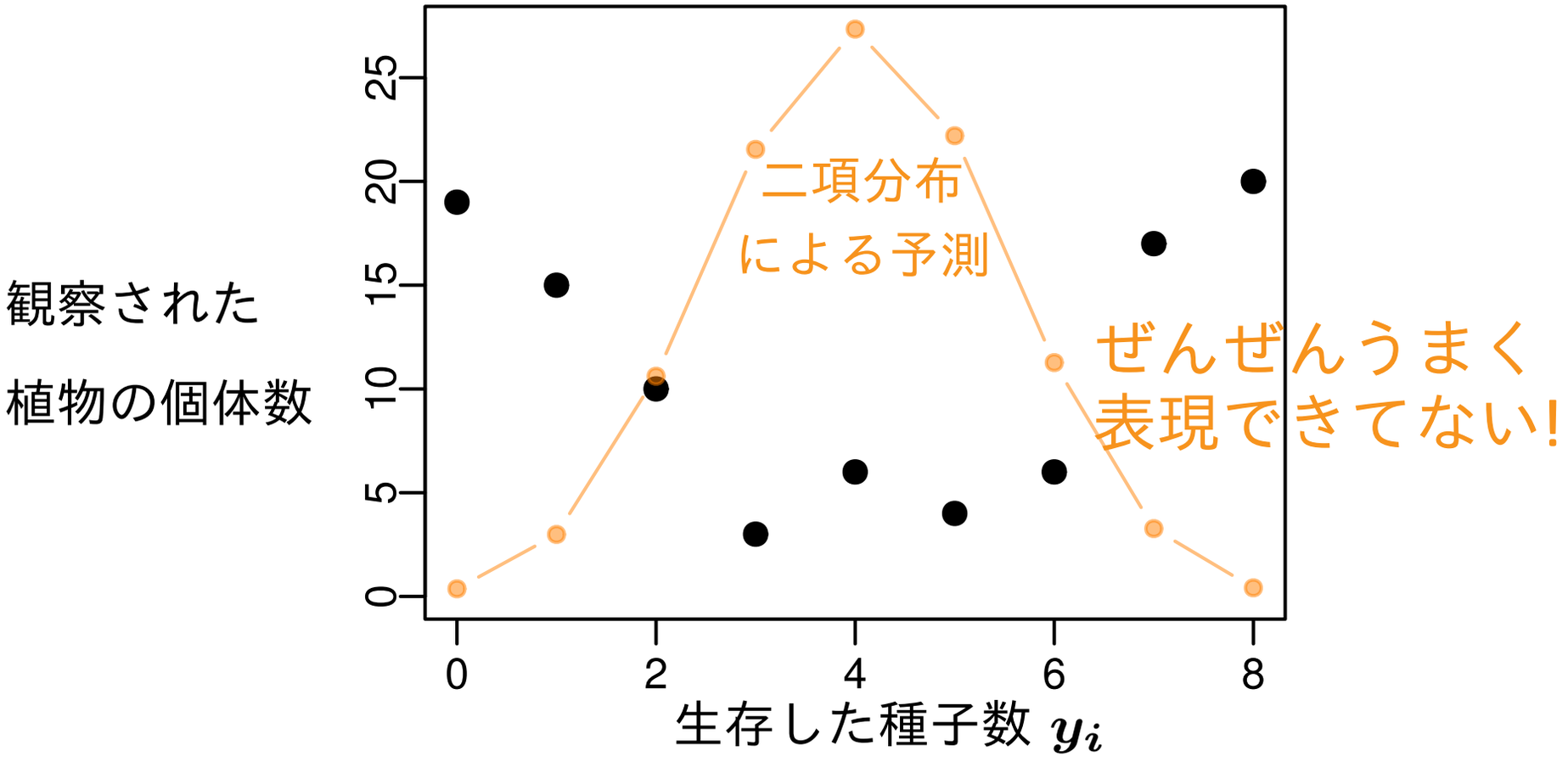
- 架空植物の種子の生存を調べた
- 用語
  - 種子: 種子のもとになる器官 (この植物ではどの個体も **8** 個 持つとする)
  - 生存: 種子が種子になること
  - 生存確率: ある種子が種子になる確率



- 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?

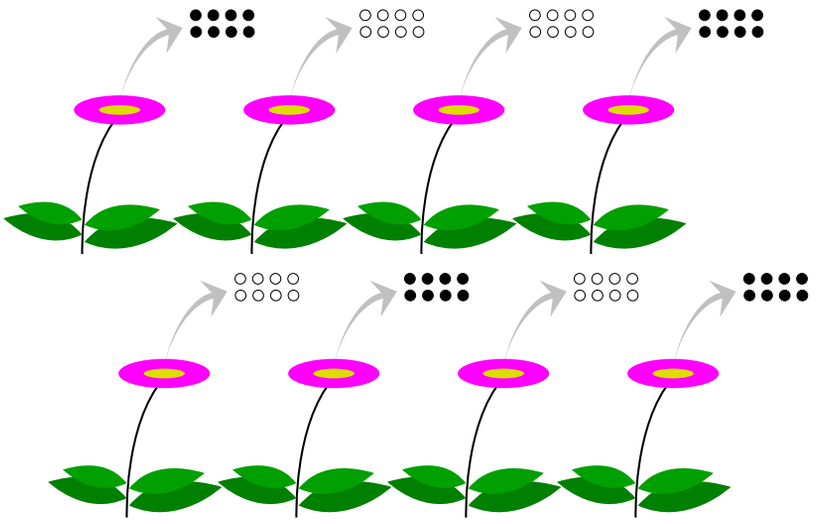
また別の観測データ: 二項分布だめだめ?!

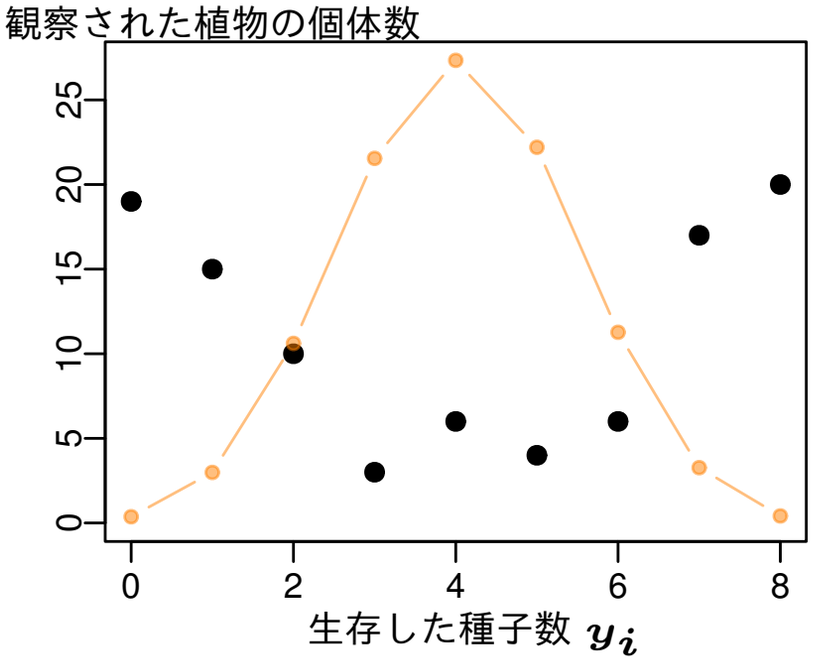
100 個体の植物の合計 800 種子中 **403** 個の生存が見られたので，平均生存確率は 0.50 と推定されたが……



「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: ?

モデリングやりなおし: まず二項分布の再検討

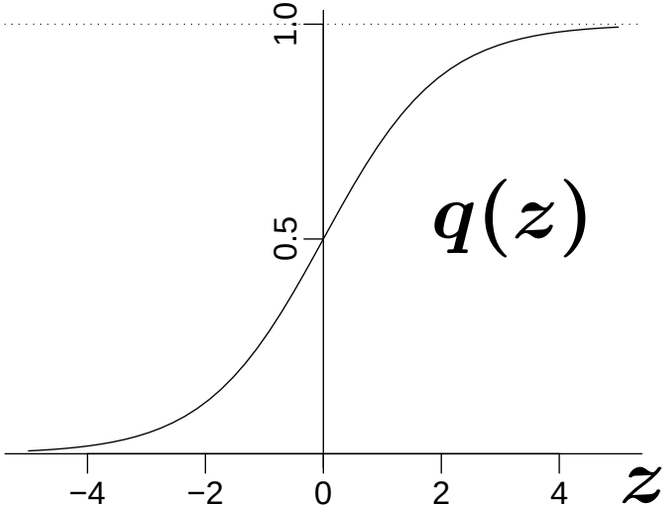
- 生存確率を推定するために 二項分布という確率分布を使う
- 個体 i の N_i 種子中 y_i 個が生存する確率は二項分布

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

- 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

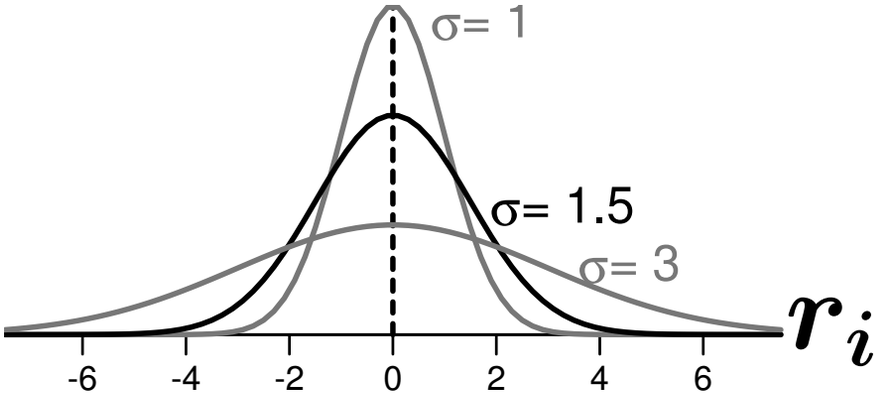


- 線形予測子 $z_i = a + r_i$ とする
  - パラメーター a: 全体の平均
  - パラメーター r_i: 個体 i の個体差 (ずれ)

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

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

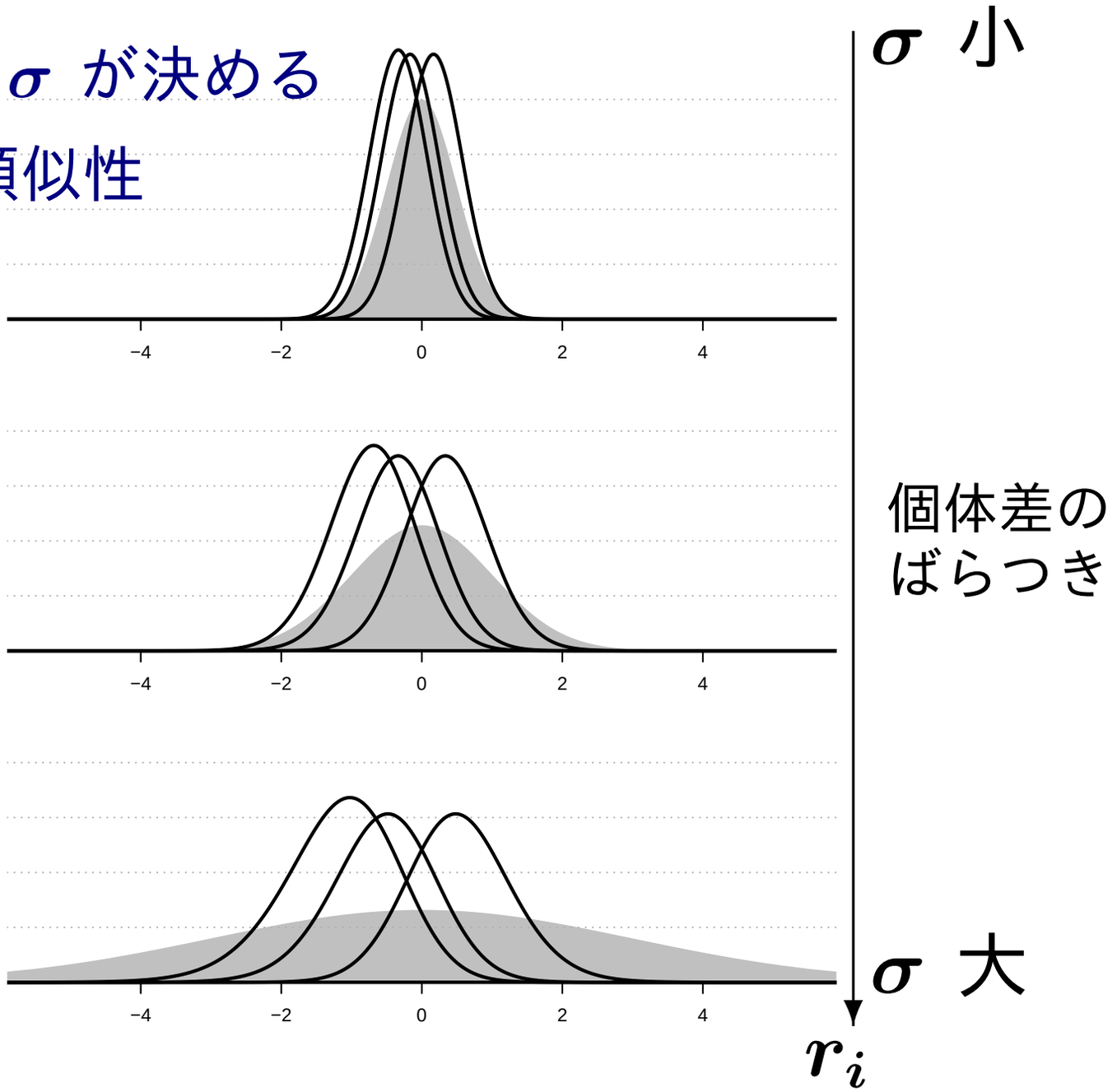


個体差 $\{r_1, r_2, \cdots, r_{100}\}$ がこの確率分布に従うとする

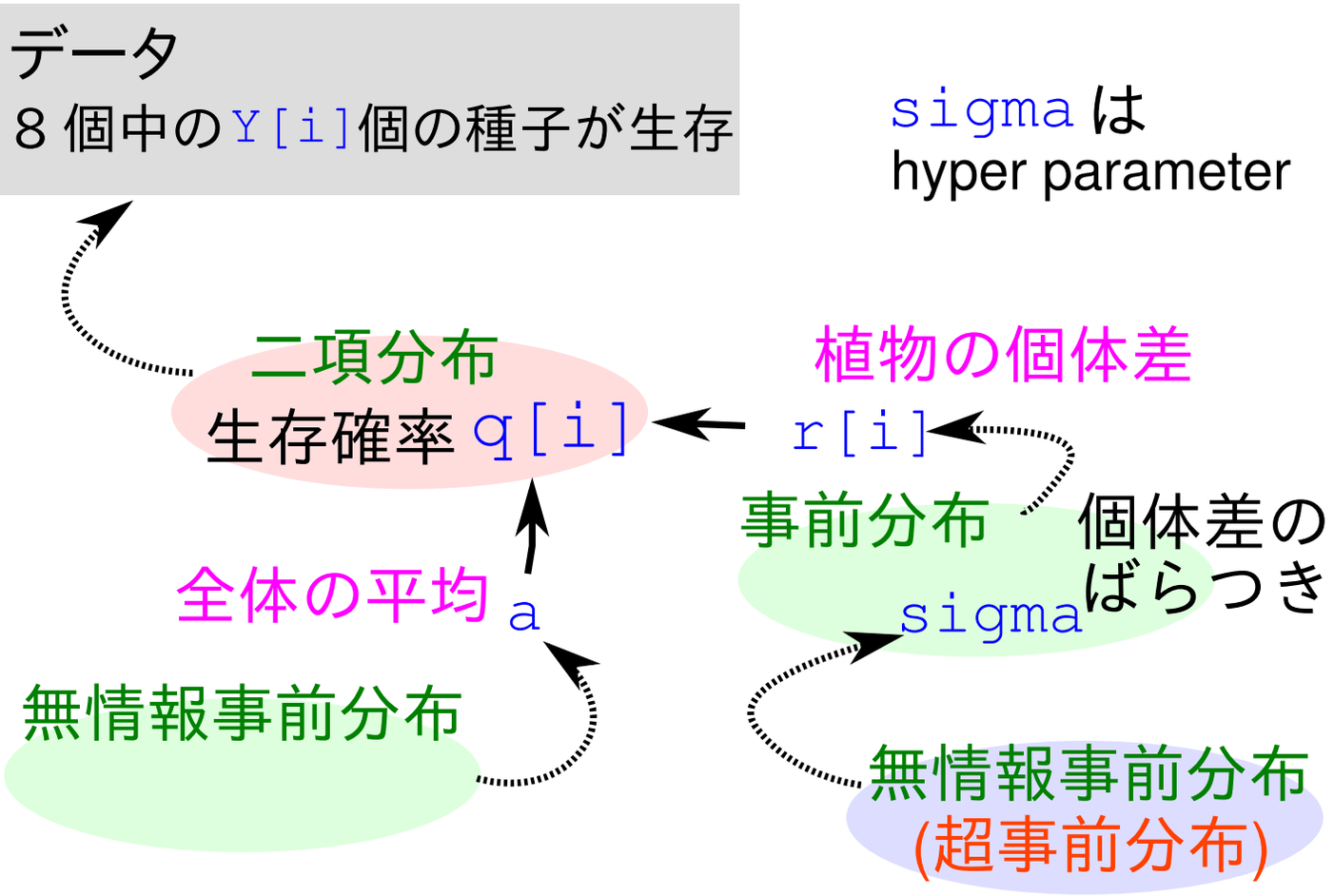
階層的な事前分布と $y_i \in \{2, 3, 5\}$ の個体の r_i

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?

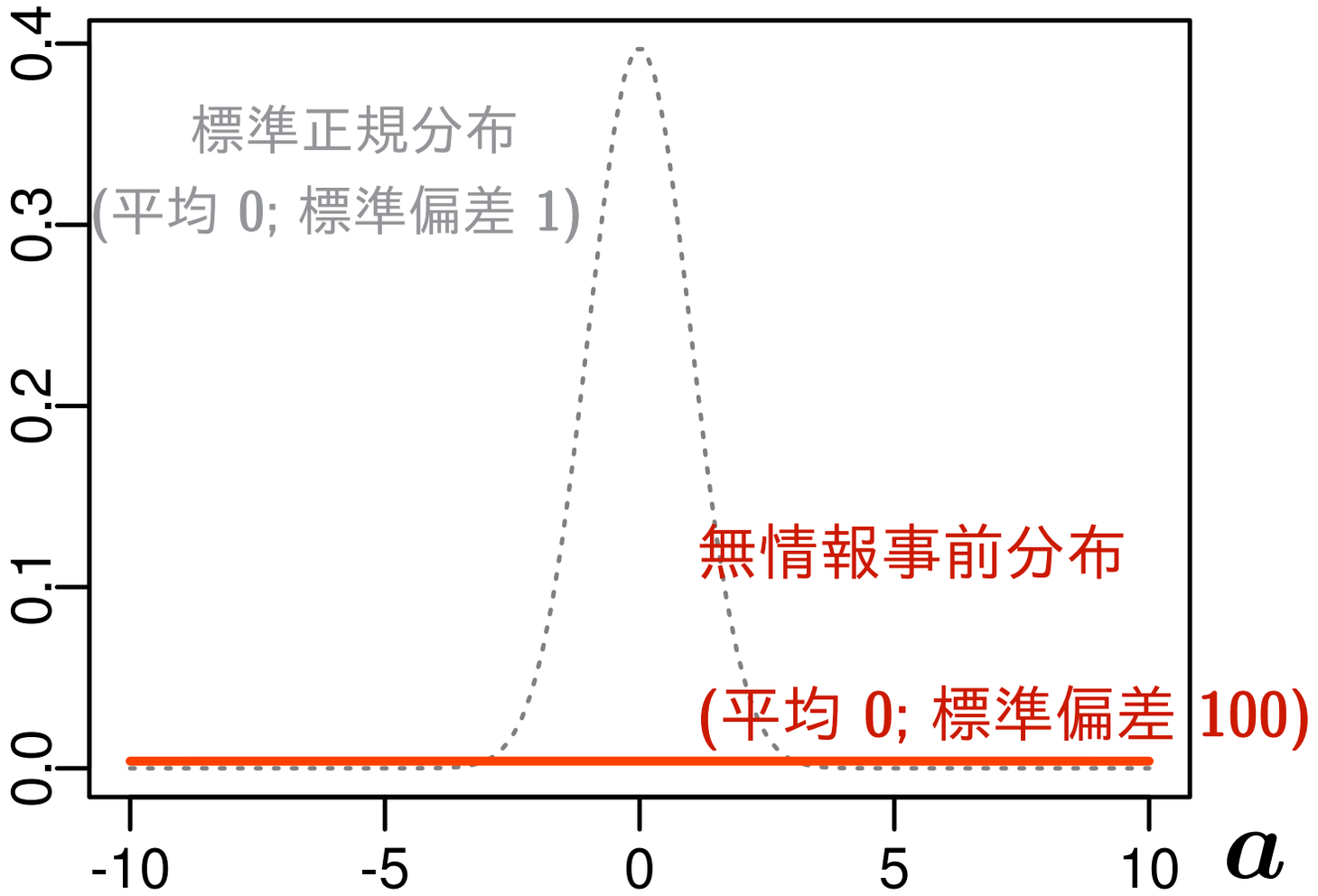


超事前分布 → 事前分布という階層があるから

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

ベイズモデルのパラメーター

を推定するための

汎用 MCMC 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

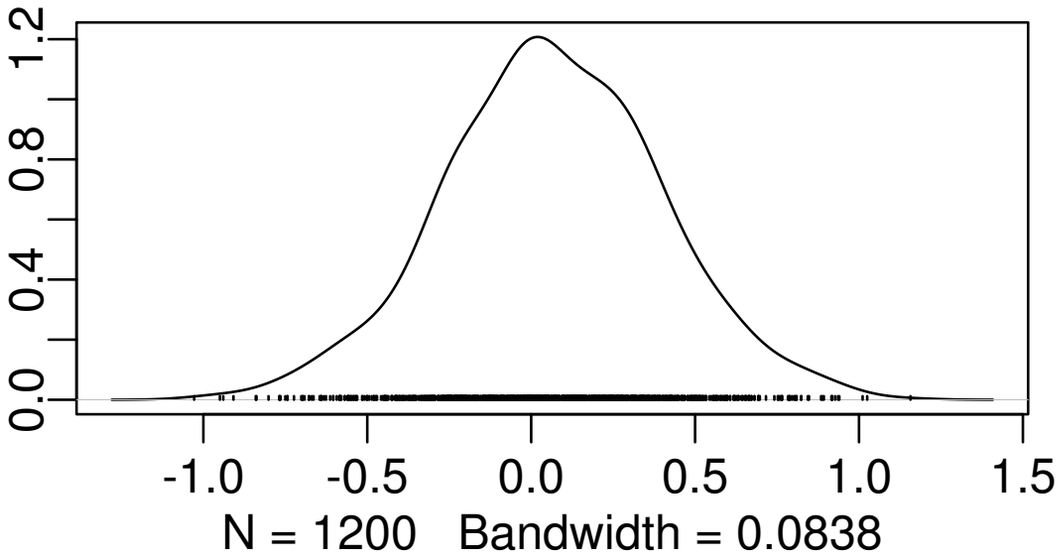
MCMC による事後分布からのサンプリング

- **M**arkov **C**hain **M**onte **C**arlo : 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」てパラメーター推定するためには **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，けっこうめんどう
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: 幅ひろい問題に適用できて，便利
   - 欠点: 「まちがいさがし」 (debug) がめんどう

統計ソフトウェア R

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

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 ($\mathbf{Y} = \{y_i\}$ はデータ)

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

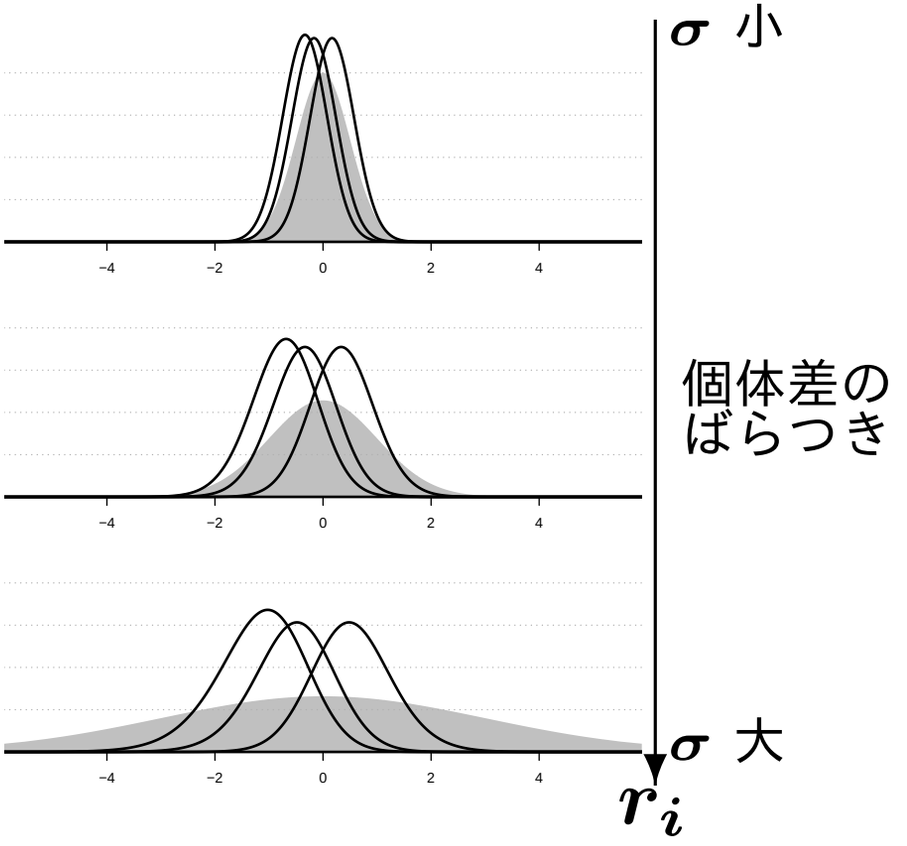
積分で「個体差」 r_i を消して，周辺尤度を定義する

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

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 (教科書の第 7 章を参照)



- 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 だけで何とかなる? 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(glmmML)` の `glmmML()`
  - `library(lme4)` の `lmer()`
  - `library(nlme)` の `nlme()` (正規分布のみ)
  - `library(MCMCglmm)` の `MCMCglmm()`
- しかしながら「めんどうな状況」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・……)
- 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
- 「隠れた」状態をあつかうモデル
  - 例: 「欠側値を補う」処理

いろいろな MCMC の方法:

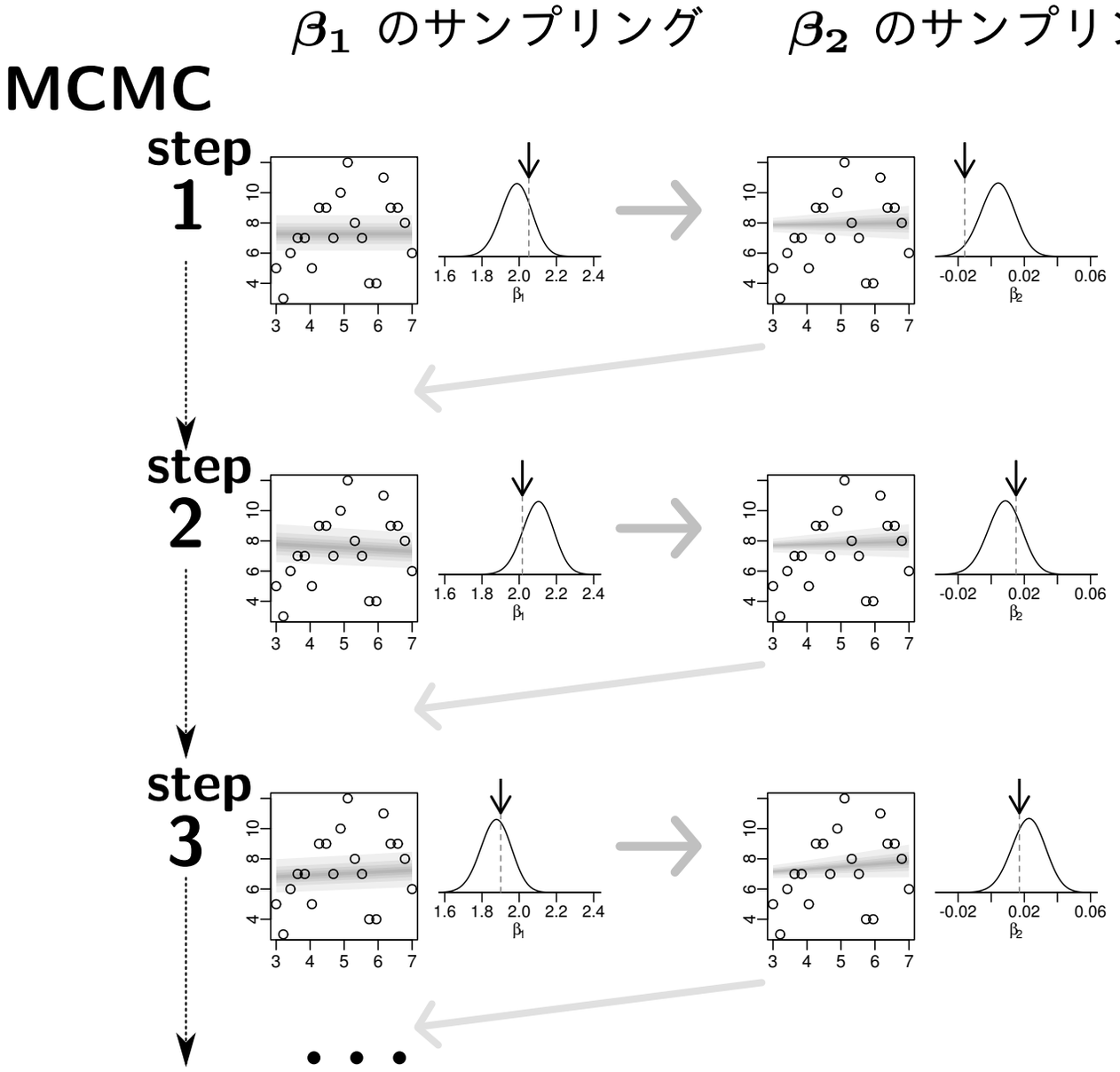
- メトロポリス法: 試行錯誤で値を変化させていく MCMC
  - メトロポリス・ヘイスティングス法: その改良版
- ギブス・サンプリング: 条件つき確率分布を使った MCMC
  - 普通は複数の変数 (パラメーター・状態) のサンプリングのためにもちいる

- ここからあとで登場する MCMC はギブス・サンプリングと考えてください

Gibbs sampling とは何か?

- MCMC アルゴリズムのひとつ
- 複数のパラメーターの MCMC サンプリングに使う
- 例: パラメーター β_1 と β_2 の Gibbs sampling
  1. β_2 に何か適当な値を与える
  2. β_2 の値はそのままにして，その条件のもとでの β_1 の MCMC sampling をする
  3. β_1 の値はそのままにして，その条件のもとでの β_2 の MCMC sampling をする
  4. 2. – 3. をくりかえす
- 教科書の第 9 章の例題で説明

図解: Gibbs sampling (統計モデリング入門の第 9 章)



種子の生存確率の問題の場合: 事後分布

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

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: $p(a)\, p(r_i \mid \sigma)\, p(\sigma)$

種子の生存確率の問題の場合: 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

$$
\begin{aligned}
p(a \mid \cdots) &\propto \prod_{i=1}^{100} p(y_i \mid q(a + r_i))\, p(a) \\
p(\sigma \mid \cdots) &\propto \prod_{i=1}^{100} p(r_i \mid \sigma)\, p(\sigma) \\
p(r_1 \mid \cdots) &\propto p(y_1 \mid q(a + r_1))\, p(r_1 \mid \sigma) \\
p(r_2 \mid \cdots) &\propto p(y_2 \mid q(a + r_2))\, p(r_2 \mid \sigma) \\
&\vdots \\
p(r_{100} \mid \cdots) &\propto p(y_{100} \mid q(a + r_{100}))\, p(r_{100} \mid \sigma)
\end{aligned}
$$

便利な “BUGS” 汎用 Gibbs sampler たち

- 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
  - Stan — 評: 期待の新鋭
- 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) {
        r[i] ~ dnorm(0, tau)
    }
    tau <- 1 / (sigma * sigma)
    sigma ~ dunif(0, 1.0E+4)
}
# あとで説明
```

なんとなく使われ続けている WinBUGS 1.4.3

- 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 $\longrightarrow$ 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
- ヘンな 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

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 があって，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

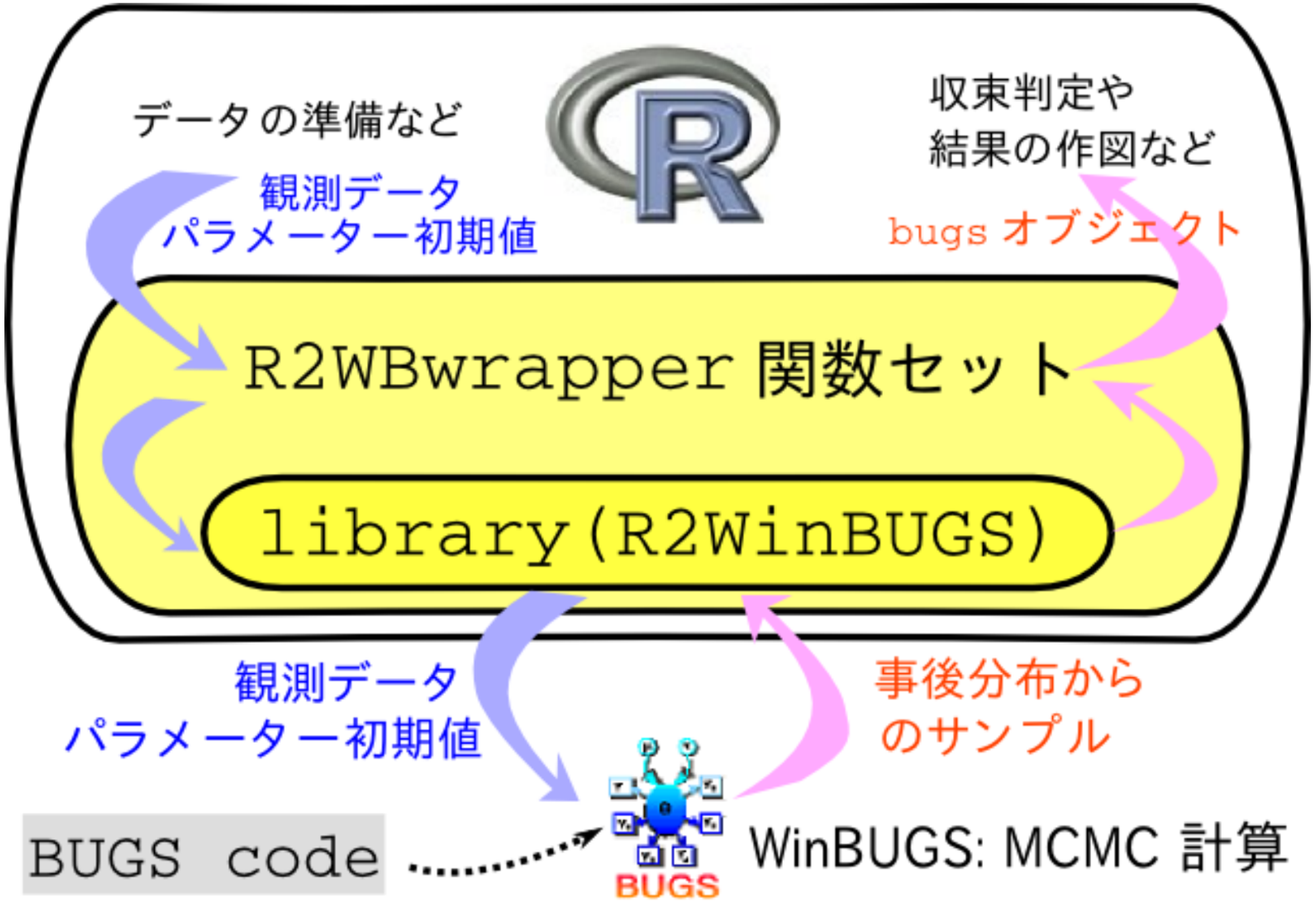
なんで R2WinBUGS をラップして使うの?

- R2WinBUGS 直接利用がめんどうだから
  - 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)



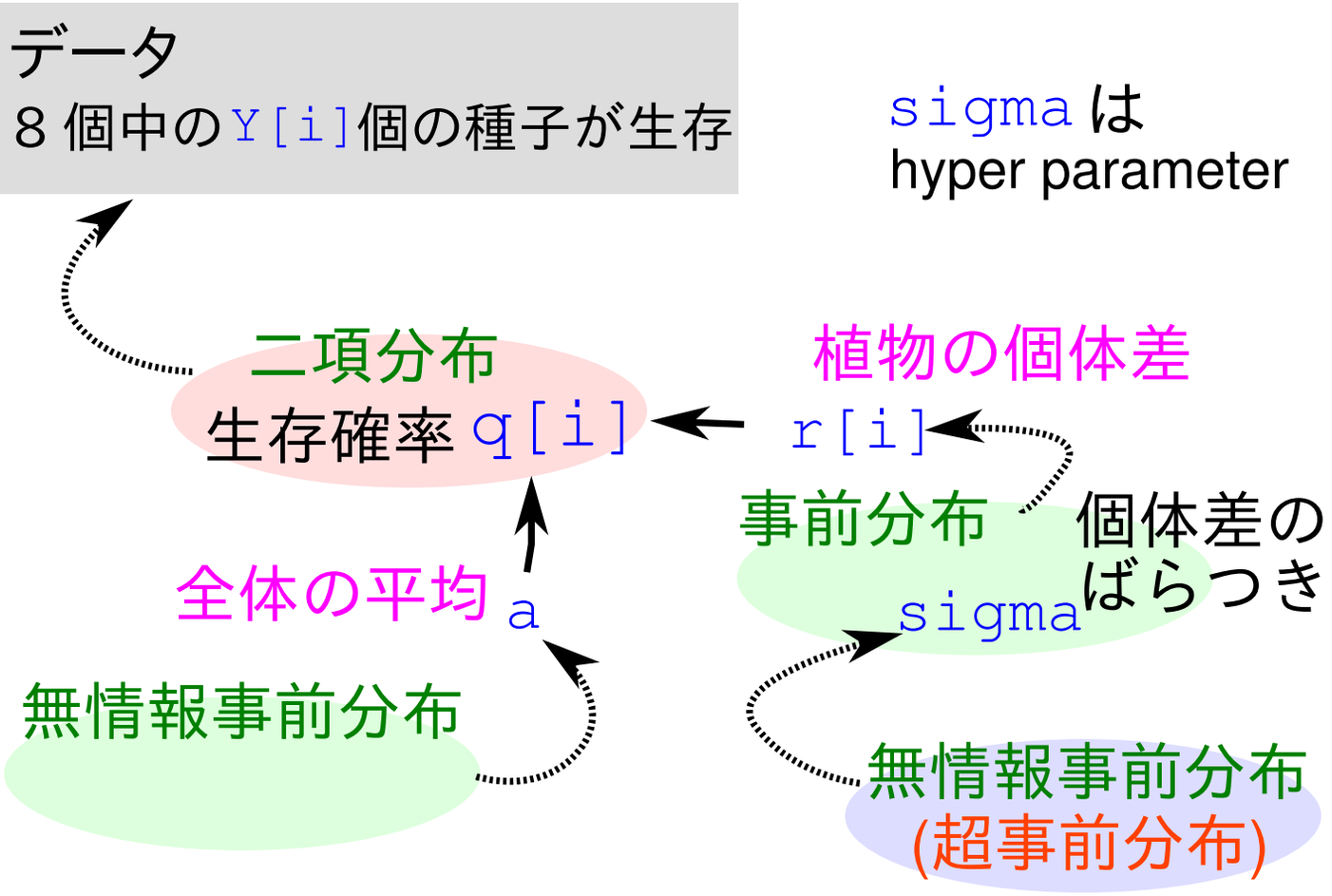
「生存確率の推定」例題を

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「関係」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. ~ sthochastic relationship
  2. <- deterministic relationship

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 = 1300, n.burnin = 100, n.thin = 3
)
```

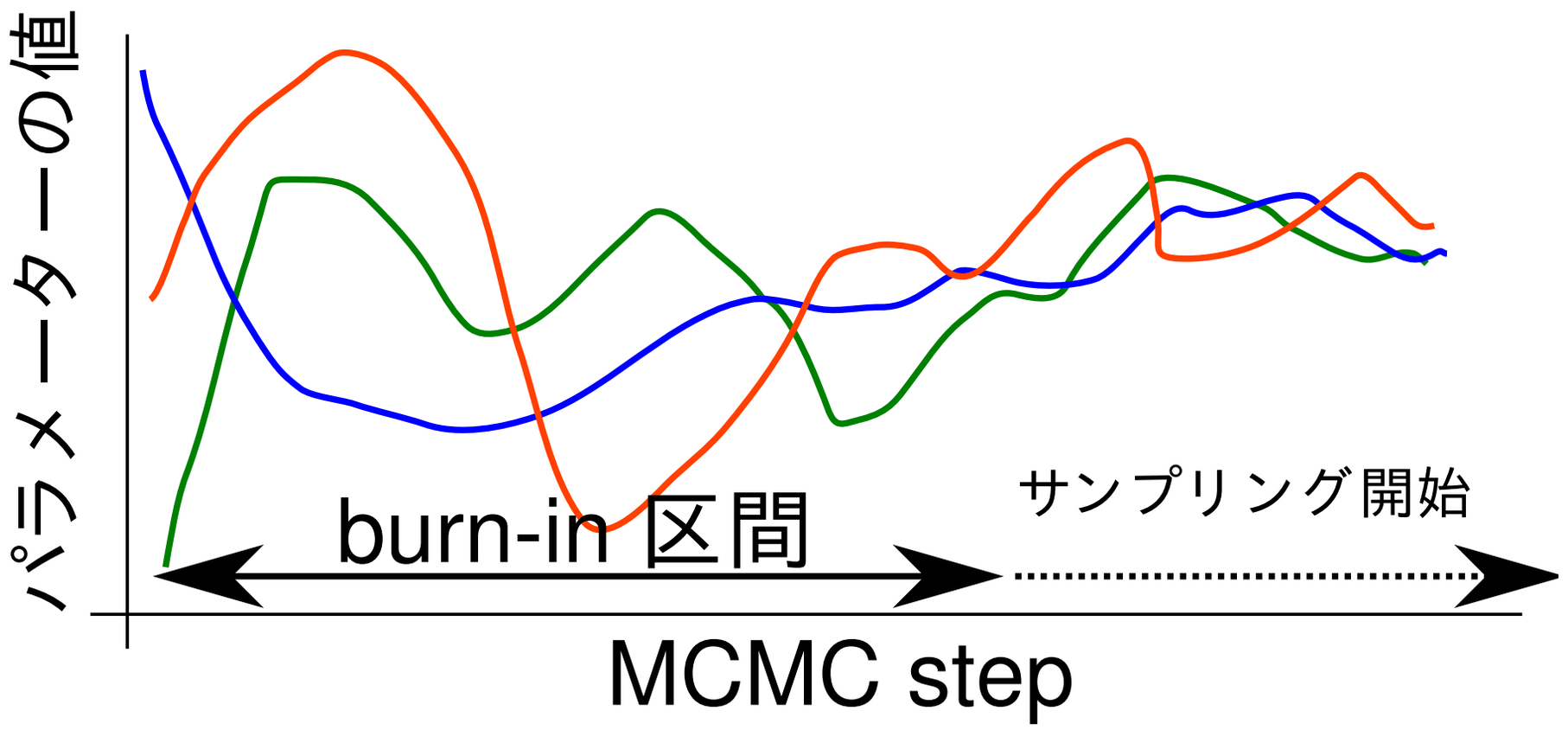
- じつは default では独立に (並列に)

  (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 1300 step (n.iter = 1300)
- 最初の 100 step は捨てる(n.burnin = 100)
- 101 から 1300 step まで 3 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

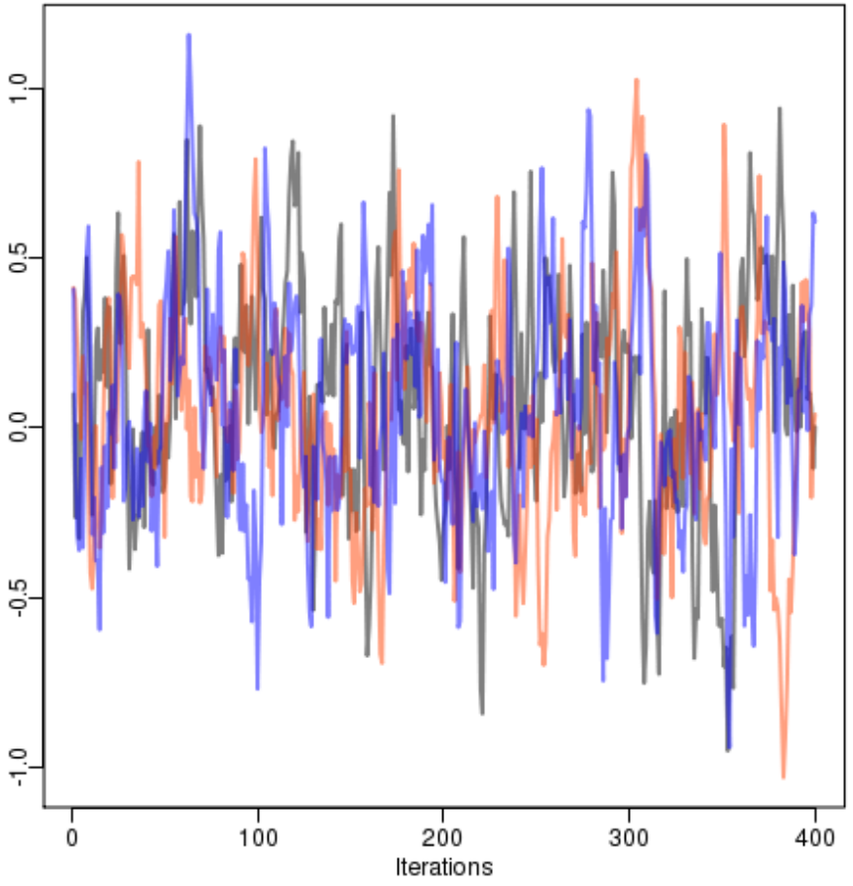


で，実際に動かすには?

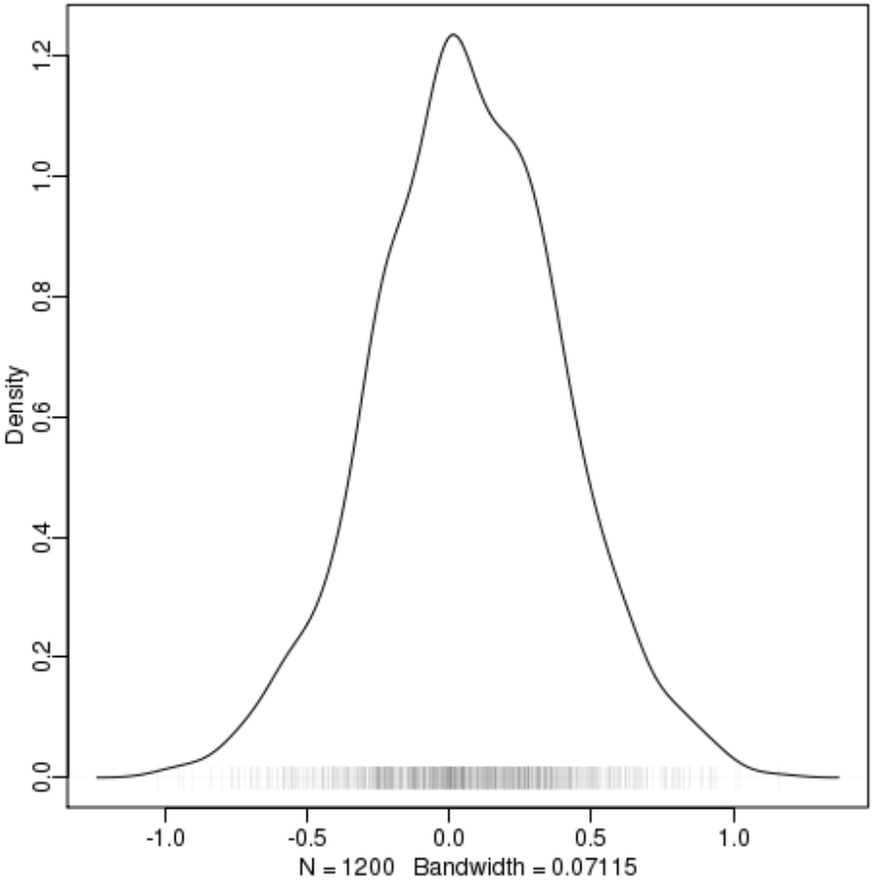
- たとえば，R 上で `source("runbugs.R")` とか
- すると WinBUGS が起動して MCMC sampling をはじめる
- この例題は簡単なのですぐに計算が終了する (WinBUGS 内で図などが表示される)
- 手動で WinBUGS を終了する
- すると WinBUGS が得た結果が R にわたされ，`post.bugs` というオブジェクトにそれが格納される

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



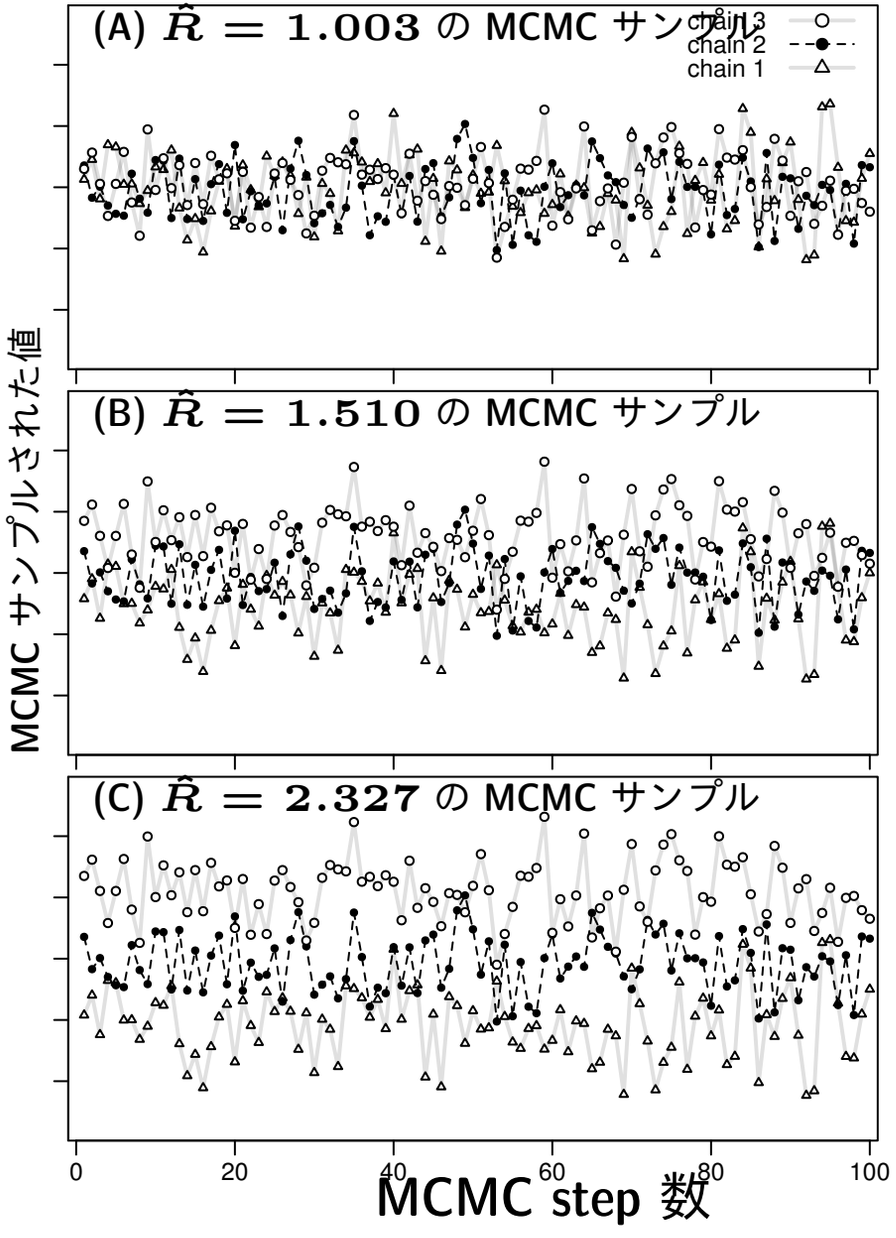
a の事後確率密度の推定



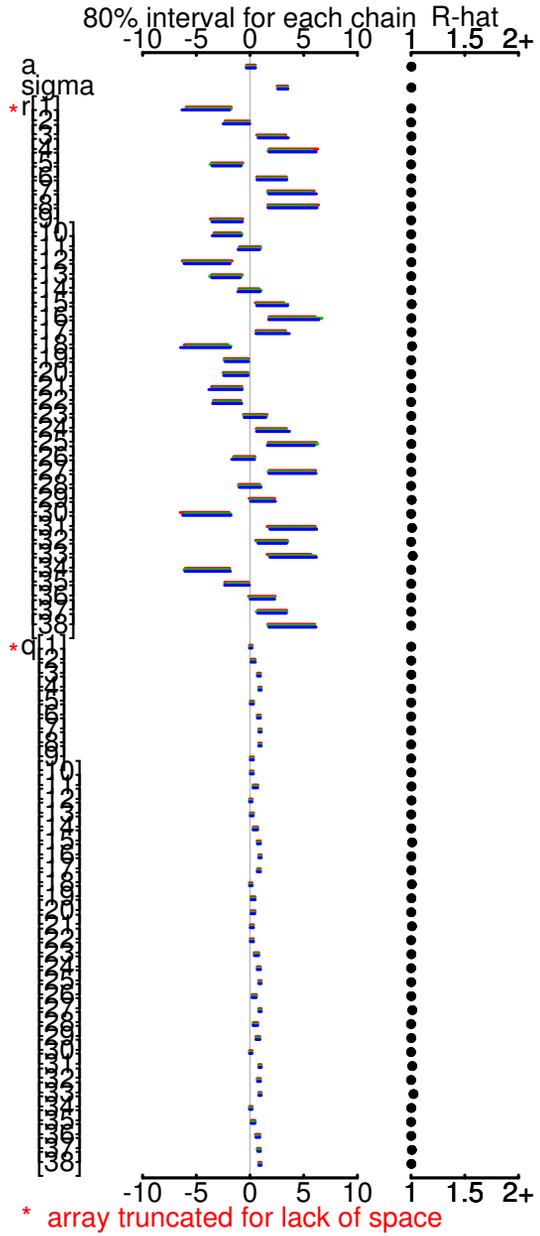
bugs オブジェクトの post.bugs を調べる (1)

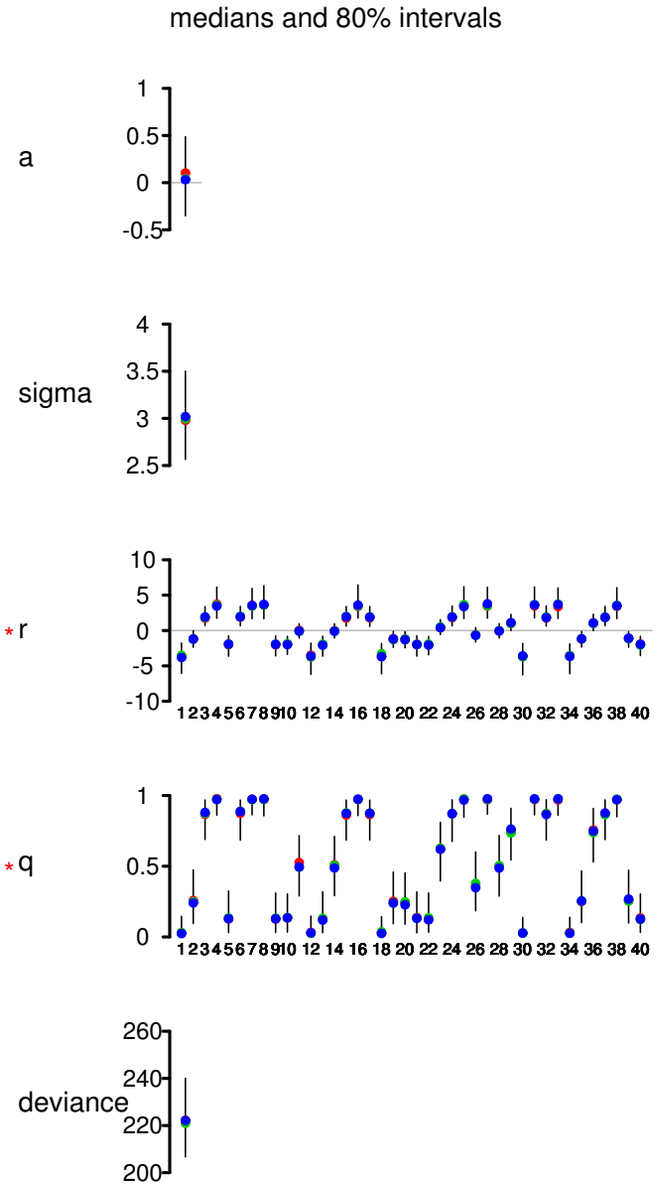
- plot(post.bugs)　　→ 次のページ, 実演表示
- R-hat は Gelman-Rubin の収束判定用の指数
  - $\hat{R} = \sqrt{\dfrac{\hat{\mathrm{var}}^{+}(\psi|y)}{W}}$
  - $\hat{\mathrm{var}}^{+}(\psi|y) = \dfrac{n-1}{n}W + \dfrac{1}{n}B$
  - Gelman et al. 2004. Bayesian Data Analysis. Chapman & Hall/CRC

収束判定の指数のひとつ: $\hat{R}$



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





bugs オブジェクトの post.bugs を調べる (2)

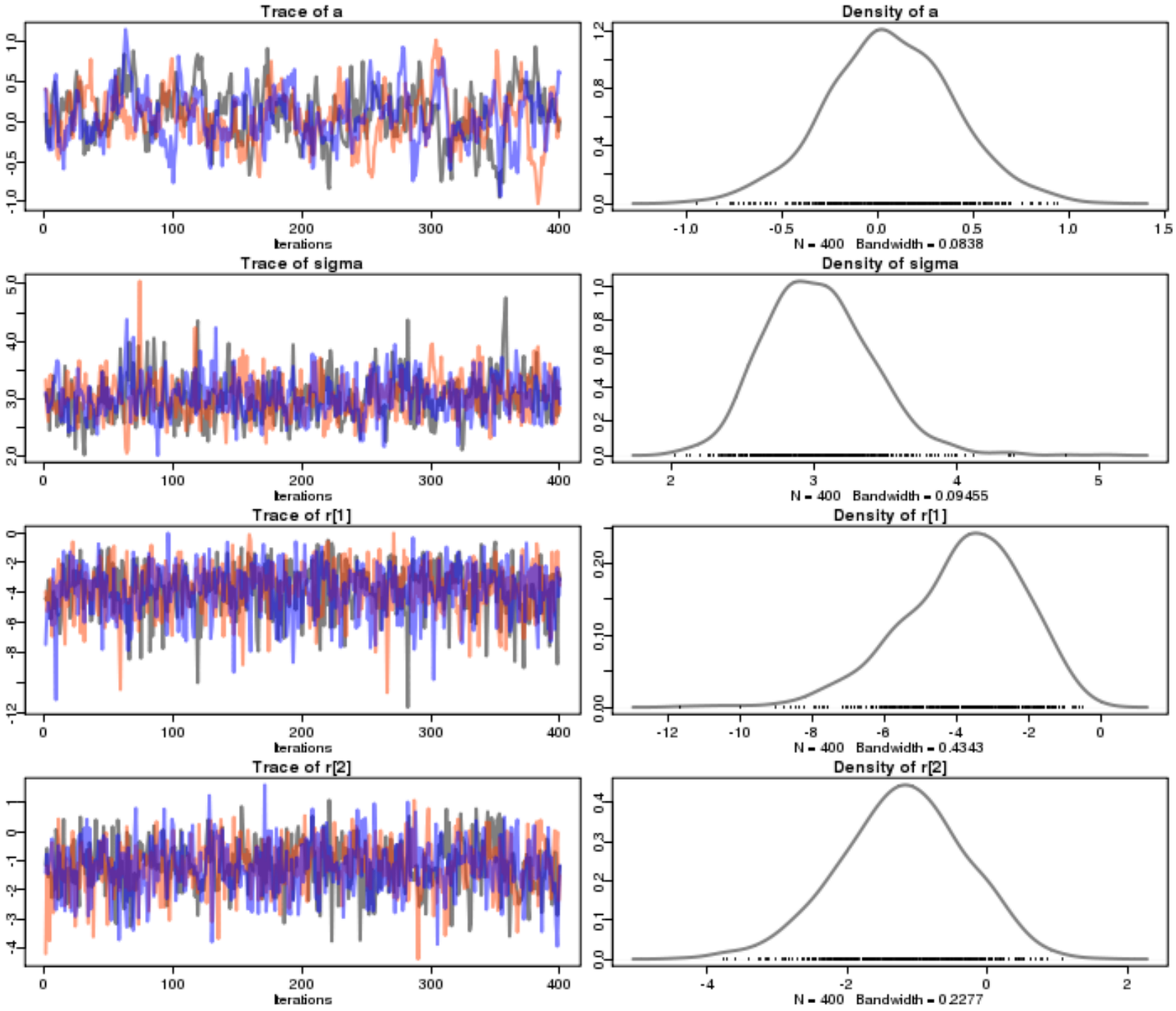
- `print(post.bugs, digits.summary = 3)`
- 事後分布の 95% 信頼区間などが表示される

	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)`

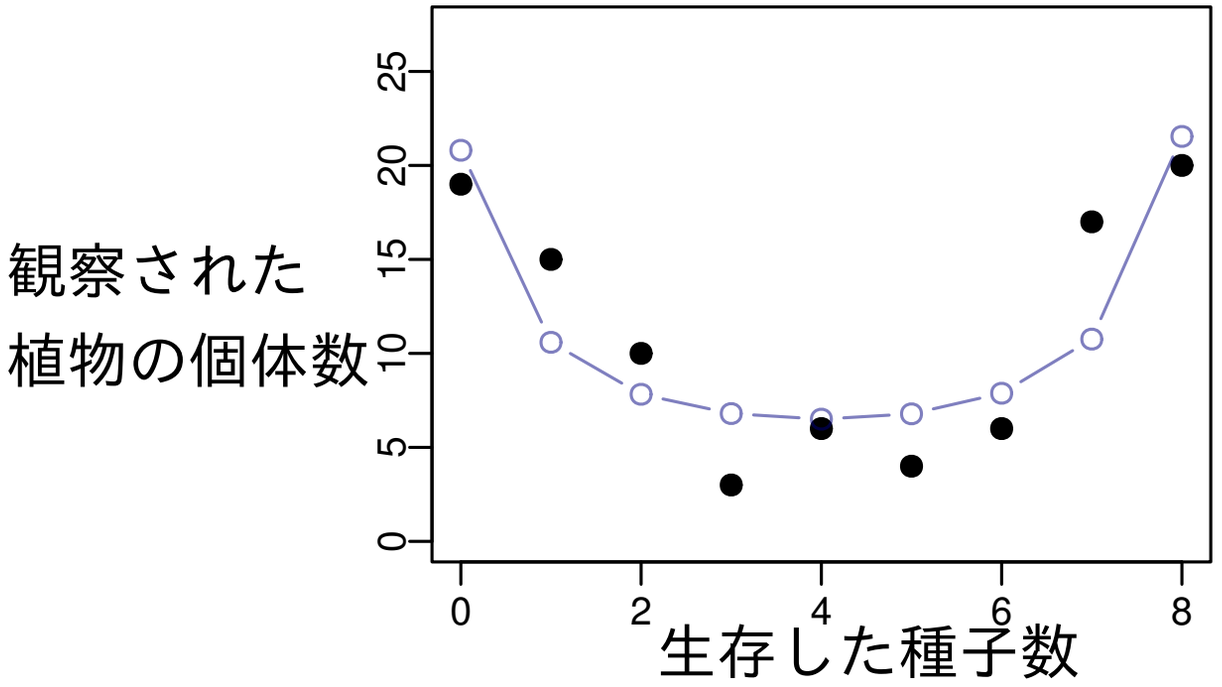
→ 次のページ, 実演表示


Trace of a
Density of a
N = 400 Bandwidth = 0.0838
Trace of sigma
Density of sigma
N = 400 Bandwidth = 0.09455
Trace of r[1]
Density of r[1]
N = 400 Bandwidth = 0.4343
Trace of r[2]
Density of r[2]
N = 400 Bandwidth = 0.2277
Iterations

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



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**

今日，説明したこと

1. **GLM は階層ベイズモデル化する**
2. **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

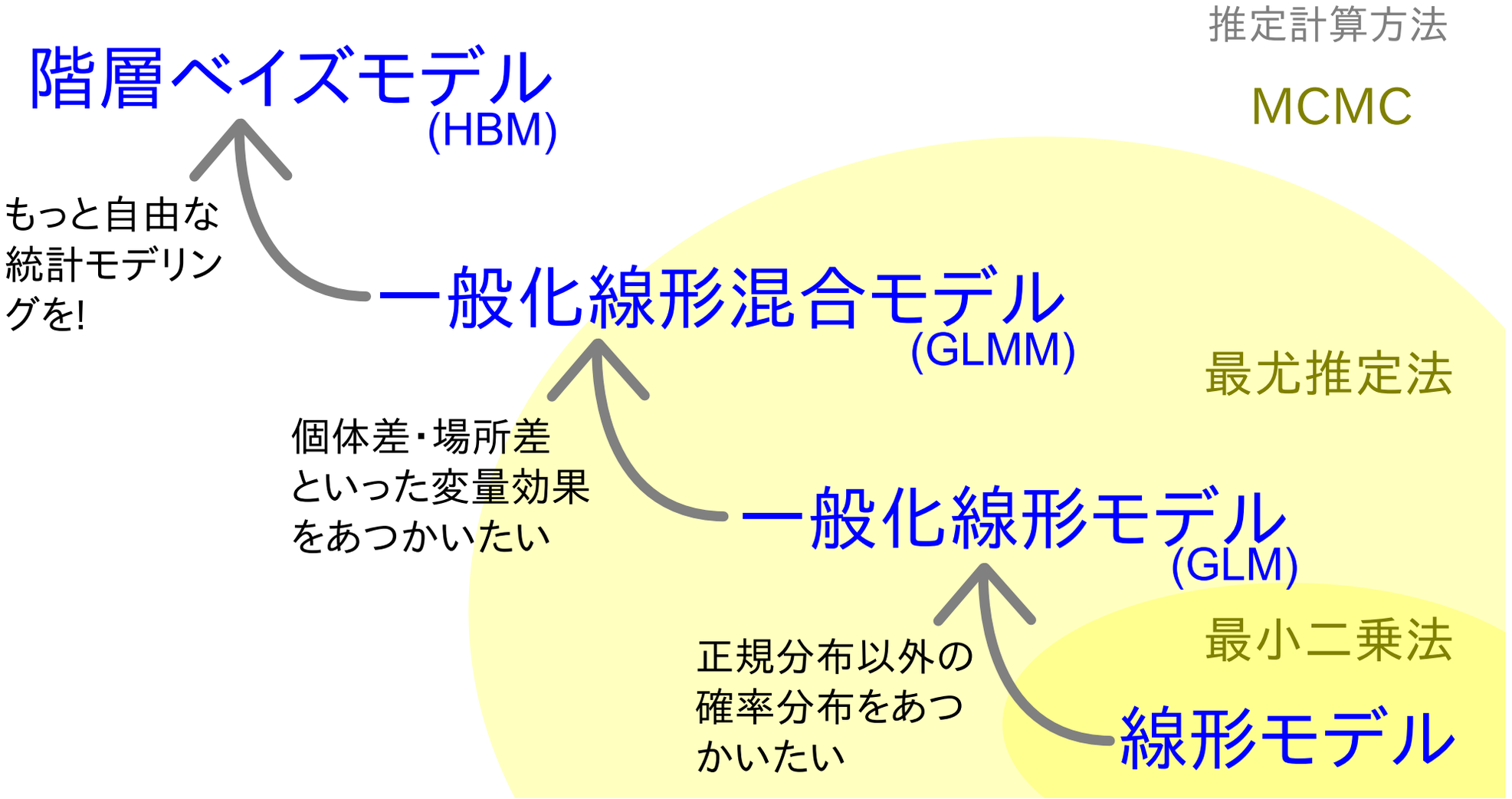
3. **WinBUGS を R で使う**

   R2WBwrapper 関数セットを経由して

4. **WinBUGS でパラメーターの事後分布推定**

   そして結果を **R** 内で解析・作図・変換する

線形モデルの発展


推定計算方法
階層ベイズモデル
(HBM)
MCMC
もっと自由な
統計モデリン
グを!
一般化線形混合モデル
(GLMM)
最尤推定法
個体差・場所差
といった変量効果
をあつかいたい
一般化線形モデル
(GLM)
最小二乗法
正規分布以外の
確率分布をあつ
かいたい
線形モデル