統計数理研究所 公開講座 (2009-02-23)

「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

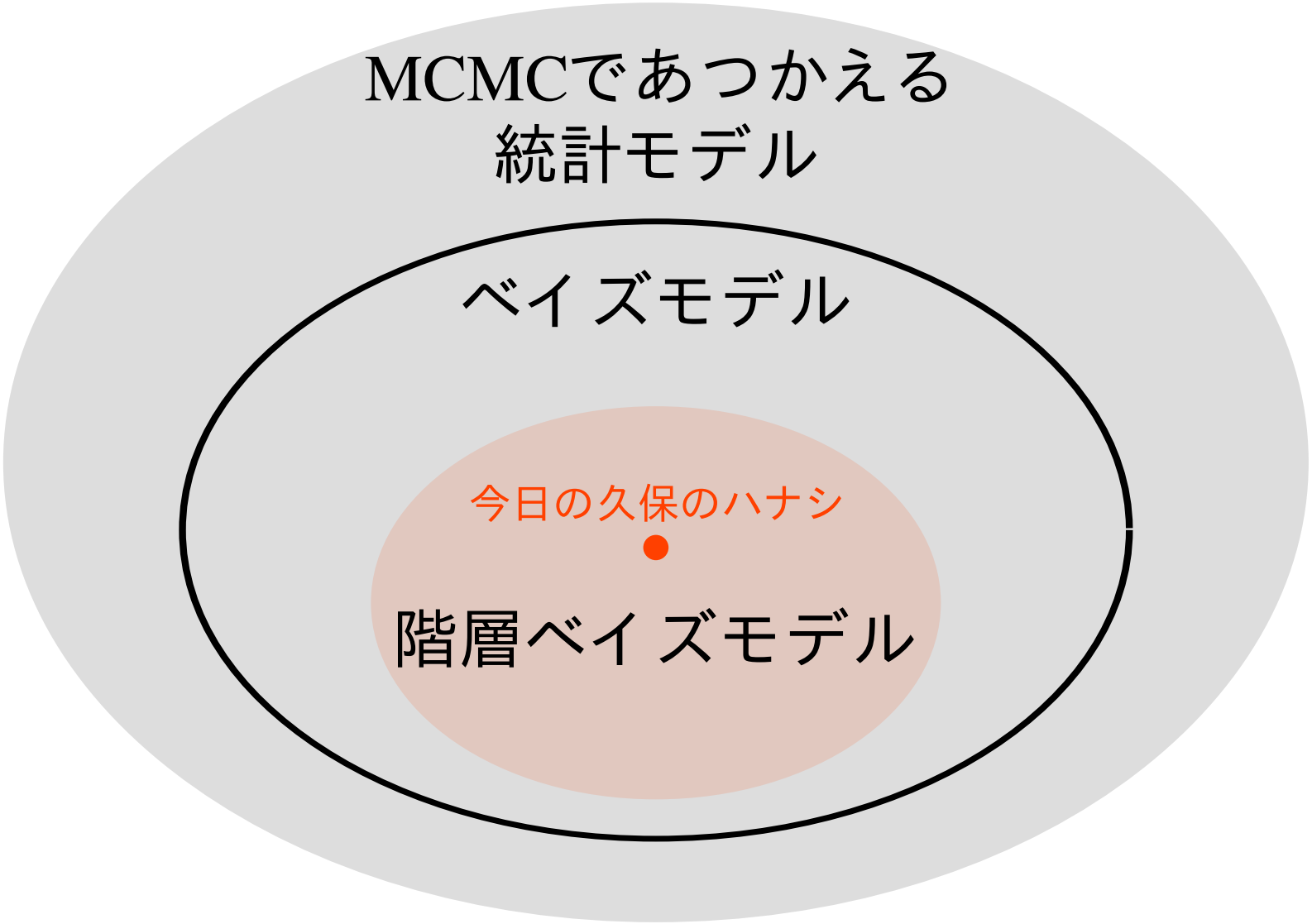
久保拓弥 kubo@ees.hokudai.ac.jp

http://hosho.ees.hokudai.ac.jp/~kubo/ce/IsmBayes2009.html

伊庭さんの MCMC 講座と久保バナシの関連

- **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)
- それって統計学的なデータ解析と何か関係あるの? ベイズ?
- 久保のハナシ:
  1. 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，と言ってみて ……
  2. 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)

とても限られた範囲のハナシです



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

「ベイジアンモデリングと MCMC」の内容

1. いきなり今日の例題にとりかかる

   打席数が不足ぎみな「打率」の推定

2. 今日の例題を階層ベイズモデル化

   MCMC と階層ベイズモデルの関係は?

3. とりあえずのまとめ

   ベイズモデリングの概念の整理

2-1. いきなり今日の例題にとりかかる

今日の例題: (架空の) 打者の打率について

- プロ野球 (架空) の打者 20 名の打数 (打席数) と安打数のデータがある
- ここから各打者の「打率」のようなものを推定したい
- 各チームの 3-5 番打者だけを選んだ
  - みなさんけっこう打率が高い?
- これは架空のハナシなので，われわれななぜかしら全打者の「真の打率」なるものを知っている

架空データ: 打数・安打数データ 20 打者ぶん

- 打者は i という記号であらわされ,
  $i = 1, 2, 3, \cdots, 20$, つまり 20 打者いる
- 打者 i の (打席数) 打数 は N_i, そのうち Y_i 回安打した
- しかしながら, 打者によって打数 N_i が まったくばらばら
- そもそも N_i の範囲が 20–48 と 少ない
- 各打席が独立となるように間隔あけてデータをとった
- ともかくそのデータとやらを見てみますか ……

(d.RData に格納されているデータを表示してみる)

その前に統計ソフトウェア R についてヒトコト

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
- 今日の例題データは R 上であつかう
  - 下記 URL からデータファイル d.RData をダウンロード
  - R を起動して load("d.RData")
  - データオブジェクト (data.frame) である d が読みこまれる

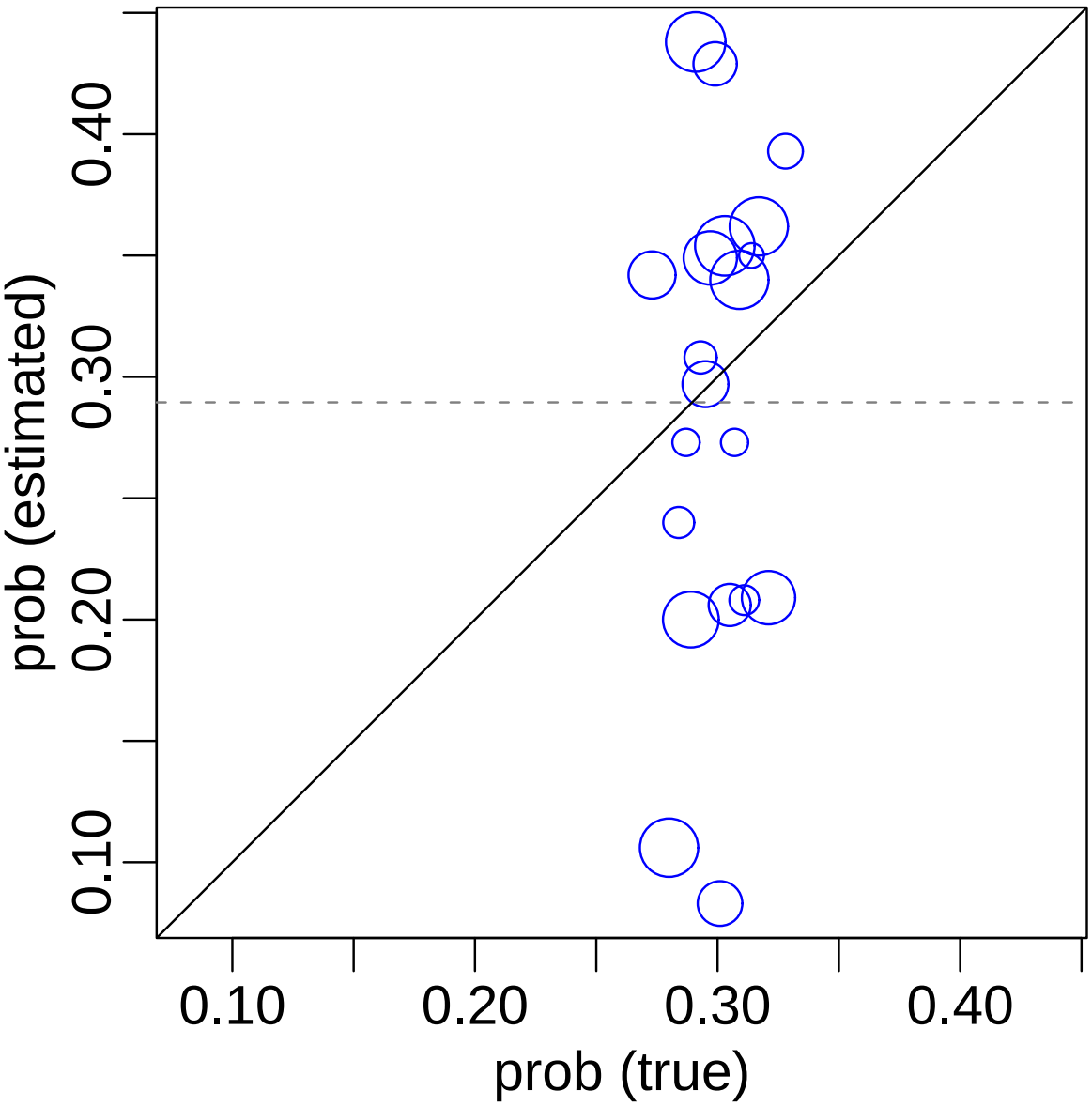
http://hosho.ees.hokudai.ac.jp/~kubo/ce/IsmBayes2009.html

架空データ: 打数・安打数データ 20 打者ぶん

	id	p.true	N	Y	p.naive		id	p.true	N	Y	p.naive
1	1	0.273	38	13	0.342	11	11	0.301	36	3	0.083
2	2	0.280	47	5	0.106	12	12	0.303	48	17	0.354
3	3	0.284	25	6	0.240	13	13	0.305	34	7	0.206
4	4	0.287	22	6	0.273	14	14	0.307	22	6	0.273
5	5	0.289	45	9	0.200	15	15	0.309	47	16	0.340
6	6	0.291	48	21	0.438	16	16	0.311	24	5	0.208
7	7	0.293	26	8	0.308	17	17	0.314	20	7	0.350
8	8	0.295	37	11	0.297	18	18	0.317	47	17	0.362
9	9	0.297	43	15	0.349	19	19	0.321	43	9	0.209
10	10	0.299	35	15	0.429	20	20	0.328	28	11	0.393

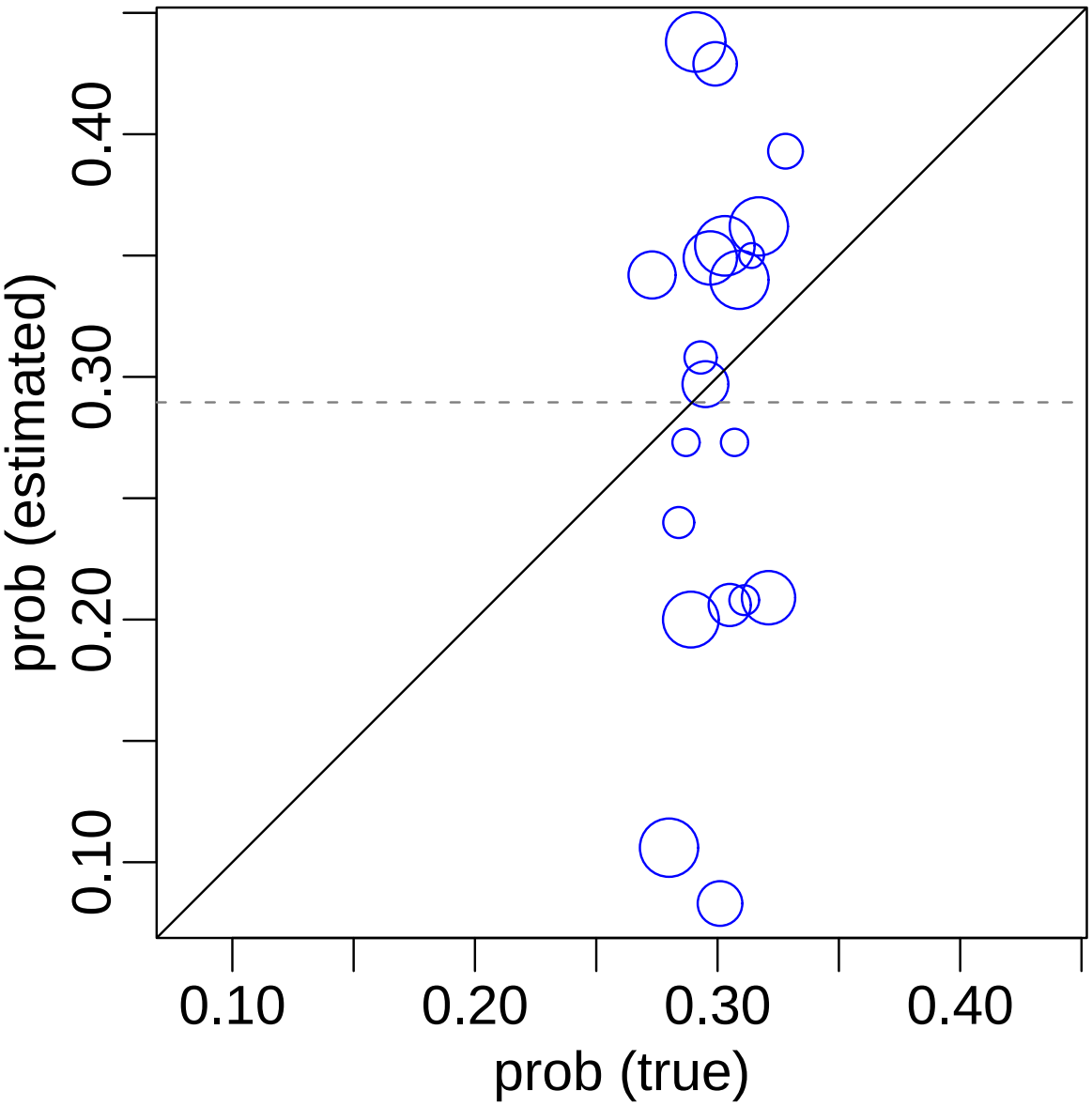
id: 打者番号, p.true: ホントの「打率」, N: 打数 (打席数), Y: 安打数, p.naive わり算 (Y / N) な打率推定値

図示: 真の打率と割算推定値の打率



- マルの大きさが打数の多さをあらわす (打数は 20–48)
- ホントの打率は 0.273–0.328 の範囲であり，どの打者も「おおよそ三割打者」
- 推定された打率は 0.083–0.438 と広い範囲にちらばり，すなわち「3-5 番打者なのに一割も打てないヤツ」から「四割以上」まで混在

打数が少ないので割算推定値な打率はめちゃくちゃ



- もちろん原因は「打数 N_i が少なすぎ」(蛇足: 現実の世界には「規定打席」なるものがある)
- それではいま手もちのデータはまったくの無価値?
- 20 名もの打者からデータをとったのだから……なんとかならない?
- 単純な割算推定はやめて, もう少し統計モデルを念頭に考えれば, 少しはマシにならないだろうか?

20 打者ぶんのデータの背後に「構造」?

```
   id p.true  N  Y p.naive          id p.true  N  Y p.naive
1   1  0.273 38 13   0.342      11 11  0.301 36  3   0.083
2   2  0.280 47  5   0.106      12 12  0.303 48 17   0.354
3   3  0.284 25  6   0.240      13 13  0.305 34  7   0.206
                    ...(略)...
```

- 基本的にはデータの無いところから情報は推定できない
- しかし 20 名もの打者からデータをとったのだから…… なんとかならない?
- データの背後にある構造に着目すると，少しはマシな打率の推定ができるかも?

「打者ごとの打率」はちがうだろうけど …… 「似てる」?

- 今回の統計モデリングで仮定する「データの背後にある構造」

  打者 **20** 名の打率はそんなに異ならない: そもそも各チームの **3, 4, 5** 番打者を集めたわけだから，ホントの打率が **0.2** より低い打者とかいそうにないし，逆に長期的に **0.4** 以上を維持できる打者もいないだろう

- 「それじゃ，打者どうしはおたがいどれぐらい似てると考えればいいの?」

  → 「うまい」統計モデリングで何とかならない?

このあとの統計モデル化の説明の手順

1. 統計モデルの部品: 二項分布
2. 統計モデルの推定方法: 最尤推定法
3. 「打者全体の平均」と「打者差」をどうあつかう?
4. 階層ベイズモデル! **(ここで MCMC がよーやく登場)**

打率 q と二項分布の関係

- 打率を推定するために 二項分布という確率分布を使う
- 二項分布とは何か? 簡単化した推定計算で説明
  - ここではいったん「打者差はない」 と仮定
  - すべての打者で打率 q が共通している
- 打者 i の N_i 打数中 Y_i 安打となる確率は二項分布

$$f(\{Y_i, N_i\} \mid q) = \binom{N_i}{Y_i} q^{Y_i}(1-q)^{N_i - Y_i},$$

(以下では f(データ $\mid q$) と略記)

尤度 (ゆうど): 20 打者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 20 打者ぶんの確率はパラメータ q の関数として定義される尤度

$$L(q|\text{データ}) = \prod_{i=1}^{20} f(\text{データ} \mid q)$$

	id	p.true	N	Y	p.naive		id	p.true	N	Y	p.naive
1	1	0.273	38	13	0.342	11	11	0.301	36	3	0.083
2	2	0.280	47	5	0.106	12	12	0.303	48	17	0.354
3	3	0.284	25	6	0.240	13	13	0.305	34	7	0.206

...(略)...

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

$$\log L(q \mid \text{データ}) = \sum_{i=1}^{20} \log \binom{N_i}{Y_i}$$

$$+ \sum_{i=1}^{20} \{Y_i \log(q) + (N_i - Y_i) \log(1-q)\}$$

- 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

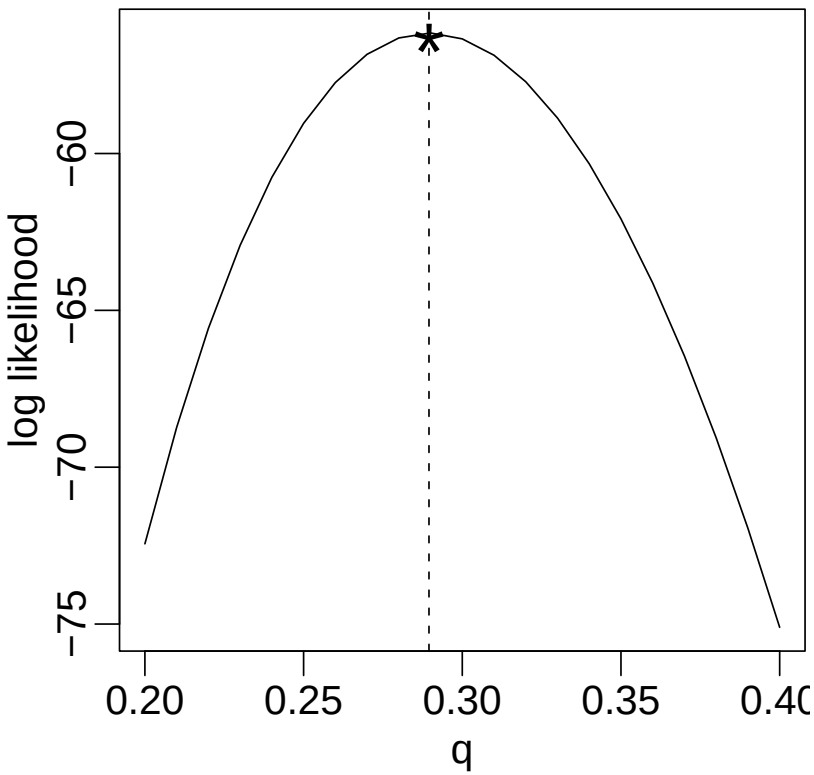
最尤推定とは何か

- 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 $L(q \mid \text{データ})$ を q で偏微分して 0 となる $\hat{q}$ が対数尤度最大

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

- 打率 q が全打者共通の場合の最尤推定量・最尤推定値は

$$\hat{q} = \frac{\text{安打数合計}}{\text{打数合計}} = \frac{207}{715} = 0.290$$



それでは各打者ごとに打率 q_i が異なるなら?

- 最尤推定量は単純な割算 $(q_i = Y_i/N_i)$ になってしまう……
- これでは `p.naive` から何も進歩してない!

	id	p.true	N	Y	p.naive
1	1	0.273	38	13	0.342
2	2	0.280	47	5	0.106
3	3	0.284	25	6	0.240
4	4	0.287	22	6	0.273
5	5	0.289	45	9	0.200
6	6	0.291	48	21	0.438
7	7	0.293	26	8	0.308
8	8	0.295	37	11	0.297
9	9	0.297	43	15	0.349
10	10	0.299	35	15	0.429

	id	p.true	N	Y	p.naive
11	11	0.301	36	3	0.083
12	12	0.303	48	17	0.354
13	13	0.305	34	7	0.206
14	14	0.307	22	6	0.273
15	15	0.309	47	16	0.340
16	16	0.311	24	5	0.208
17	17	0.314	20	7	0.350
18	18	0.317	47	17	0.362
19	19	0.321	43	9	0.209
20	20	0.328	28	11	0.393

ここまでのまとめ: 二項分布と最尤推定

- 統計モデルの基本部品である確率分布を導入した
- 「打数 N_i 中 Y_i 安打」というデータを表現できる二項分布
- 二項分布をかけあわせて尤度方程式を作った
- 対数尤度を最大にする打率 q をきめるのが最尤推定
- 全打者の打率は同じと仮定 $\longrightarrow \hat{q} = \mathbf{0.290}$
- 打者ごとに異なる $\hat{q}_i$ …… たんなる割算値だった
- もうちょっと統計モデリングに工夫が必要だろう
- このタイプの統計モデルの世界をちょっとながめてみよう…… $\longrightarrow$ 次

尤度 (ゆうど) をあつかう統計モデル

パラメーターを確率分布として表現する Bayes 統計学

階層 **Bayes** モデル の MCMC 計算による推定など

最尤推定法 であつかう統計モデル

パラメーターを点推定する，random effects もあつかえる

階層ベイズモデルである一般化線形混合モデル (GLMM) など

一般化線形モデル (GLM)

指数関数族の確率分布 + 線形モデル, fixed effects のみ

最小二乗法 であつかう統計モデル

等分散正規分布 + 線形モデル

直線回帰，いわゆる「分散分析」など

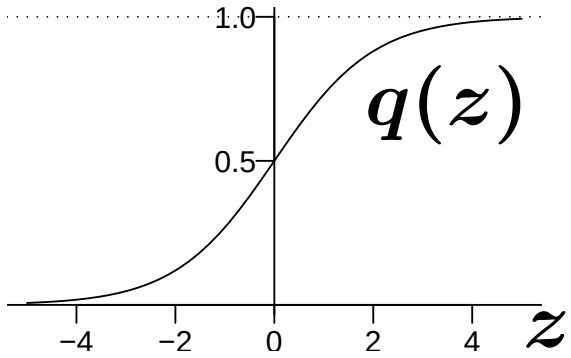
2-2. 今日の例題を階層ベイズモデル化

「打者ごとに異なる打率」モデルの準備

- 打率 q は打者によって異なるとする
- 各打者の打率 q を「全打者の平均」＋「打者差」と分割したい
- しかし $q = q_{\text{平均}} + q_{\text{打者差}}$ と定式化するのは，あれこれ面倒・不都合
  - たとえば $0 < q < 1$ となるよう気づかう必要があったりするから

ロジスティック関数で表現する打率

- そこで安打する確率 $q(z)$ 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 $z_i = a + b_i$ とする
  - パラメーター a: 全体の平均
  - パラメーター b_i: 打者 i の打者差 (ずれ)

これらのパラメーターがどのような値でも打率は $0 < q(z_i) < 1$ となる

$a + b_i$ で打率 $q(a + b_i)$ が決まる

- ある打者 i の z_i を $a + b_i$ とすると,

$$q(a + b_i) = \frac{1}{1 + \exp(-(a + b_i))}$$

尤度

$$L(a, \{b_i\} \mid \text{データ}) = \prod_{i=1}^{20} f(\text{データ} \mid q(a + b_i))$$

観測データから a, b_i を推定したい

個々の打者差 b_i を最尤推定するのはまずい

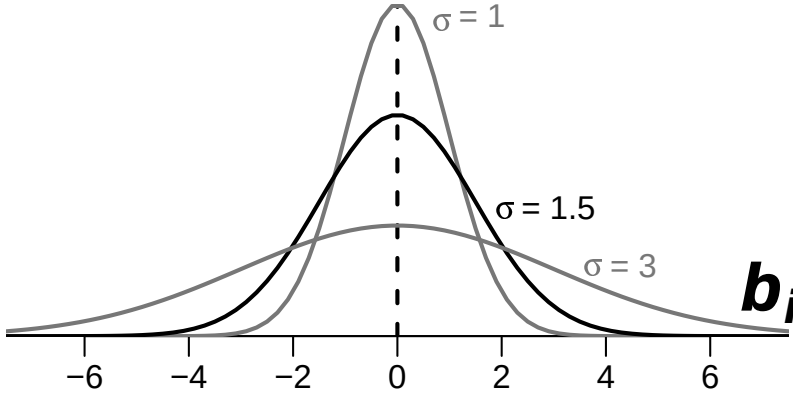
- 20 名の打率を推定するためにパラメーター 21 個(a と $\{b_1, b_2, \cdots, b_{20}\}$) を推定すると……
- 打者ごとに安打数 / 打数で推定するのと同じ!
- さきほどのアイデアを統計モデル化できないか?

打者 20 名の打率はそんなに異ならない: そもそも各チームの 3, 4, 5 番打者を集めたわけだから，ホントの打率が 0.2 より低い打者とかいそうにないし，逆に長期的に 0.4 以上を維持できる打者もいないだろう

打者差 b_i の確率分布を仮定したらどうだろう?

平均ゼロで標準偏差 σ の正規分布

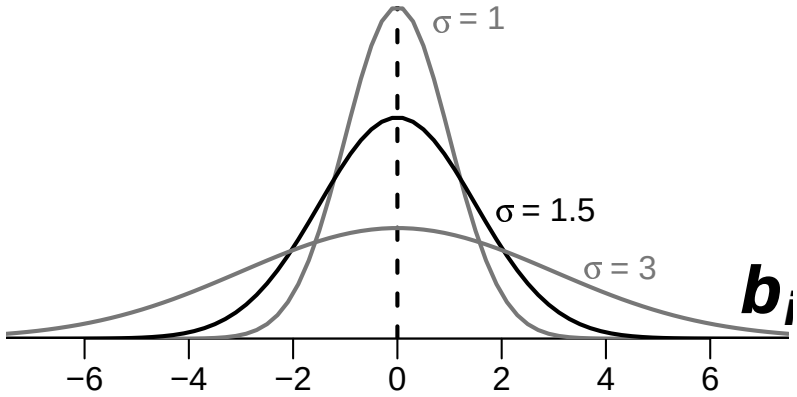
$$g_b(b_i \mid \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-b_i^2}{2\sigma^2},$$



打者差 $\{b_1, b_2, \cdots, b_{20}\}$ がこの確率分布に従う?

ベイズモデル化: 事前分布・事後分布の導入

打者差 $\{b_1, b_2, \cdots, b_{20}\}$ がこの確率分布に従う?

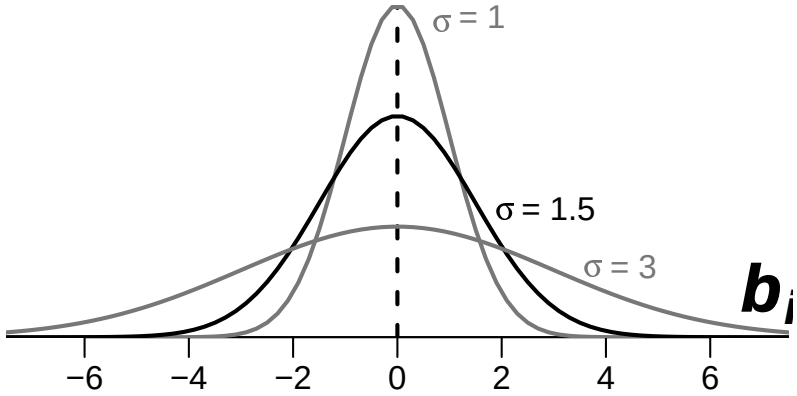


- ベイズ統計モデリングでは推定したいパラメーターすべて に 事前分布 (prior distribution) を設定する
- 推定された個々の打者差 b_i も確率分布である事後分布 (posterior distribution) として表現される

(モデル全体の定式化はまたあとで)

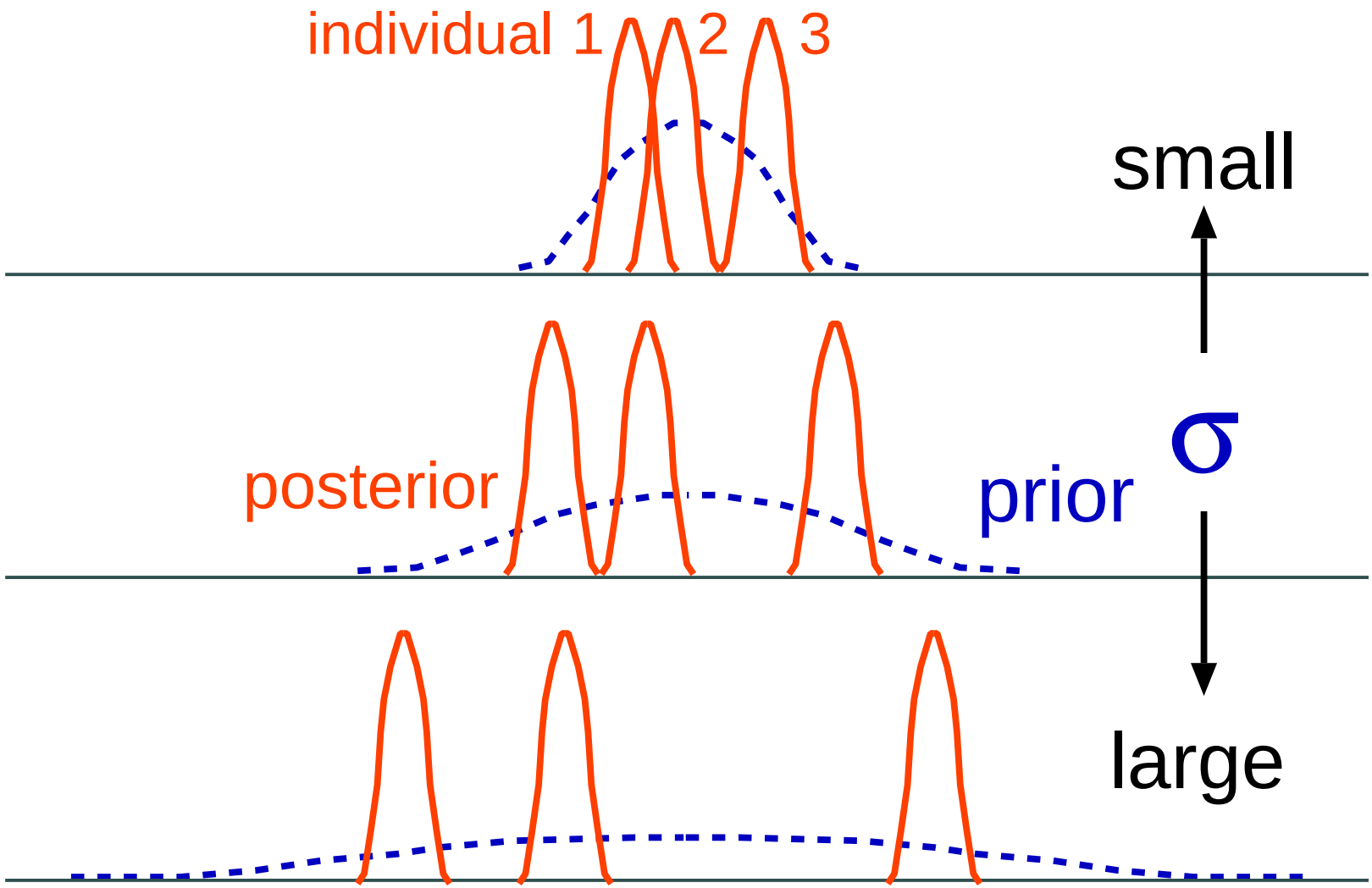
打者差 b_i の事前分布のカタチは何をあらわす?

平均ゼロで標準偏差 σ の正規分布

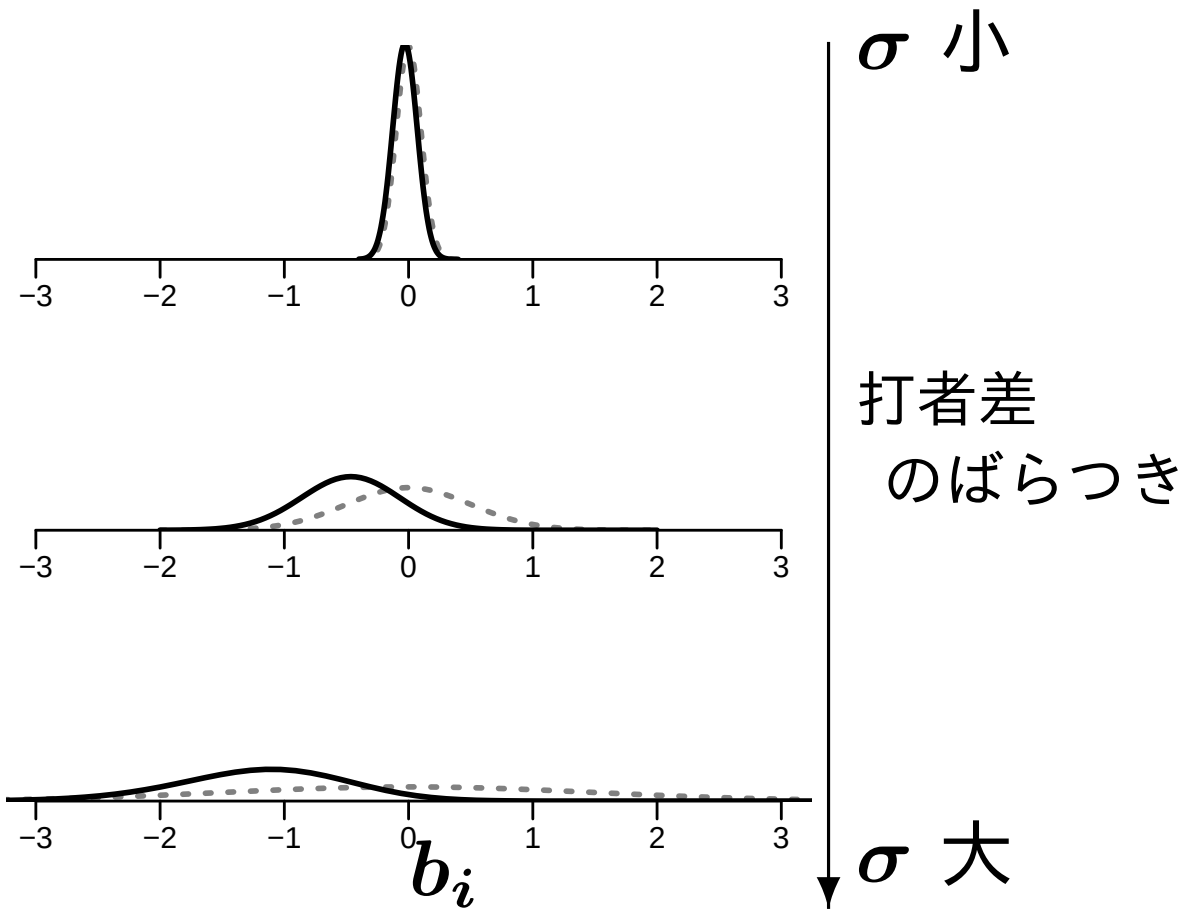


- σ がとても小さければ打者差 b_i はどれもゼロちかくになりますから「どの打者もおたがい似ている」
- σ がとても大きければ，b_i は各打者の安打数 Y_i にあわせるような値をとる
- 「中間的な」σ がよさそう? (この問題はまたあとで)

パラメーター σ が決める打者間の類似性



「リアル」に作図するとこうなります



しかしまあ，いいかげんな図示のほうがわかりやすい

ような気がするので，そちらをつかいます

「b_i のばらつき」 σ の事前分布は?

- σ も推定したいパラメーターなので事前分布を設定する必要がある
- しかしながら，この一個のパラメーターに関して「事前分布はこう設定すれば便利」といったハナシは無い
- そこで共役な事前分布を使う
  - 事後分布が事前分布と同じ確率分布になるような事前分布
- 正規分布の分散 σ^2 の共役事前分布は逆ガンマ分布
  - つまり $\tau = 1/\sigma^2$ がガンマ分布にしたがう

$\tau = 1/\sigma^2$ の事前分布を無情報事前分布

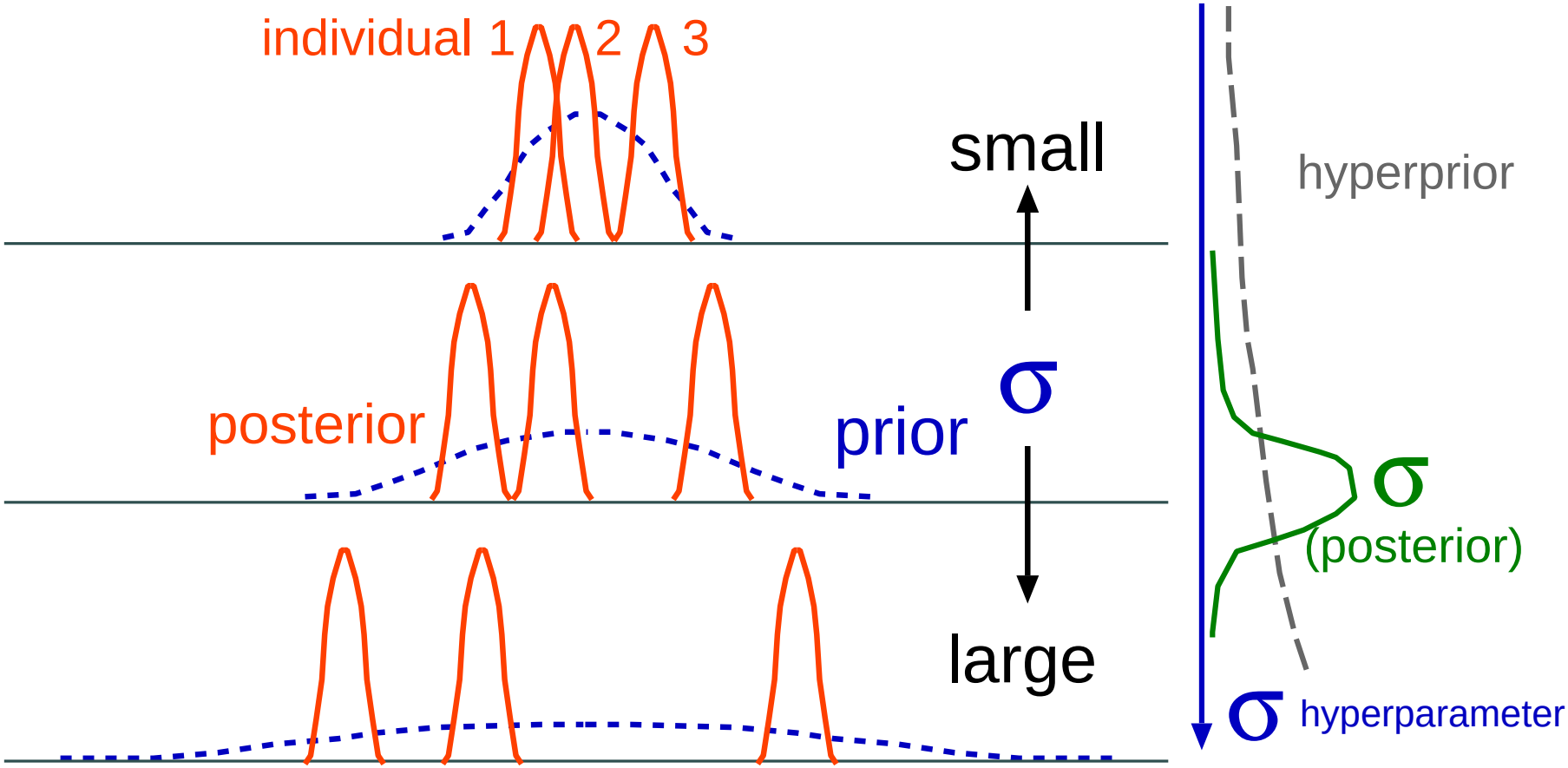
- σ はどのような値をとってもかまわない
- そこで τ の事前分布は 無情報事前分布(non-informative prior) とする
- たとえば「ひらべったいガンマ分布」

$$p(\tau) = \tau^{\alpha-1} \frac{e^{-\tau\beta}}{\Gamma(\alpha)\beta^{-\alpha}}, \qquad \alpha = \beta = 10^{-2}$$

(実演: R でちょっと作図してみる)

(モデル全体の定式化はまたあとで)

打者差 b_i とそのばらつき σ の事前分布・事後分布



「ちょうどいいぐあい」の打者差のばらつきになる

あたりを σ の事後分布となるようにしたい⇔ MCMC

打者の平均 a も事前分布は無情報事前分布

- 打者の平均 a: 平均的な打率 (の logit 変換値)
- a はどのような値をとってもかまわない
- 事前分布は 無情報事前分布(non-informative prior)
- たとえば平均ゼロで標準偏差 10 の「すごくひらべったい正規分布」

$$g_a(a) = \frac{1}{\sqrt{2\pi 10^2}} \exp \frac{-a^2}{2 \times 10^2},$$

(実演: R でちょっと作図してみる)

階層ベイズモデル全体の定式化 (1)

- ベイズの公式　　$p(P \mid D) = \dfrac{p(D \mid P) \times p(P)}{p(D)}$
- $p(P \mid D)$ は何かデータ (D) のもとで何かパラメーター (P) が得られる確率 $\longrightarrow$ 事後分布
- $p(P)$ はあるパラメーター P が得られる確率 $\longrightarrow$ 事前分布
- $p(D)$ は「てもとにあるデータ D が得られる確率」
- $p(D \mid P)$ パラメーターを決めたときにデータが得られる確率 $\longrightarrow$ 尤度

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

階層ベイズモデル全体の定式化 (2)

$$p(a, \{b_i\}, \tau \mid \text{データ}) = \frac{\prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))\, g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)}{\iint \cdots \int (\text{分子 } \uparrow \quad \text{そのまま})\, db_i\, d\tau\, da}$$

分母は何か定数になるので (伊庭さんの話にでてきた Z)

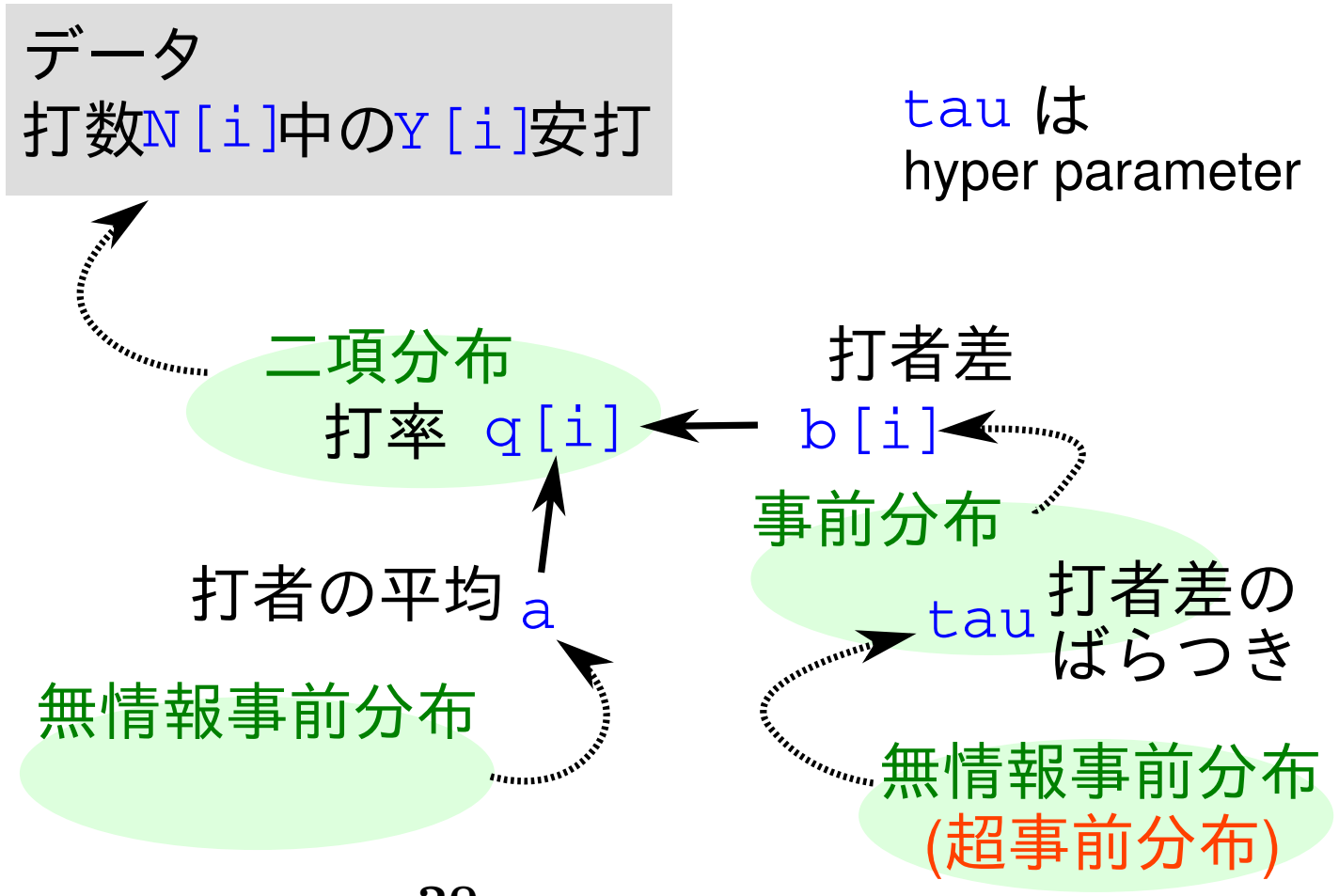
$$p(a, \{b_i\}, \tau \mid \text{データ}) \propto \prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))\, g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)$$

事後分布: $p(a, \{b_i\}, \tau \mid \text{データ})$

尤度: $\prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))$

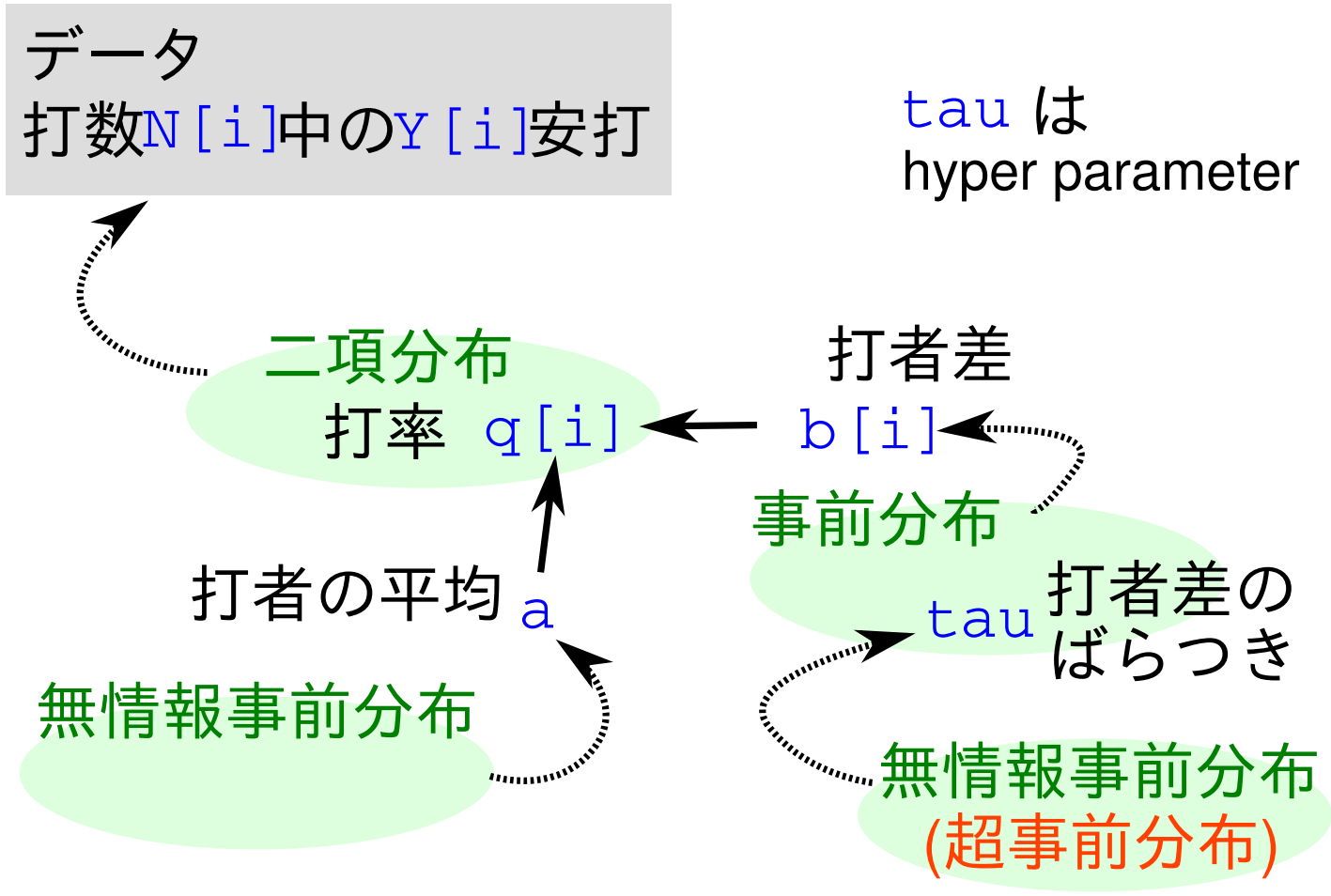
事前分布たち: $g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)$

なぜ「階層」ベイズモデルといわれるのか?



$$p(a, \{b_i\}, \tau \mid \text{データ}) \propto \prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))\, g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)$$

なぜ「階層」ベイズモデルといわれるのか?



超事前分布 ⟶ 事前分布という階層があるから

どうやって事後分布を推定するの?

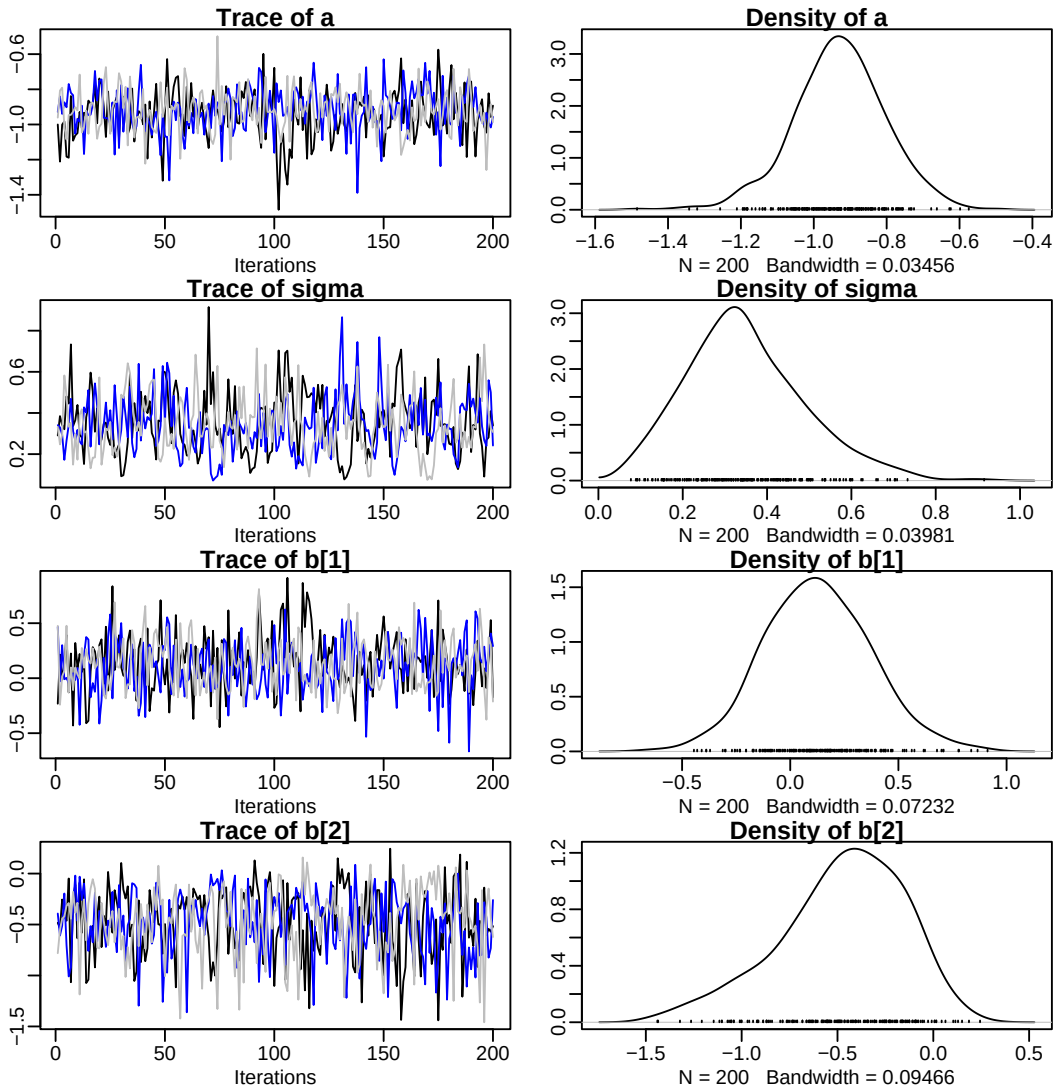
事後分布

$$p(a, \{b_i\}, \tau \mid \text{データ}) \propto \prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))\, g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)$$

- 観測データと事前分布を組みあわせれば事後分布 $p(a, \{b_i\}, \tau \mid \text{データ})$ 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
- **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!

この階層ベイズモデルの MCMC 計算の方法は?

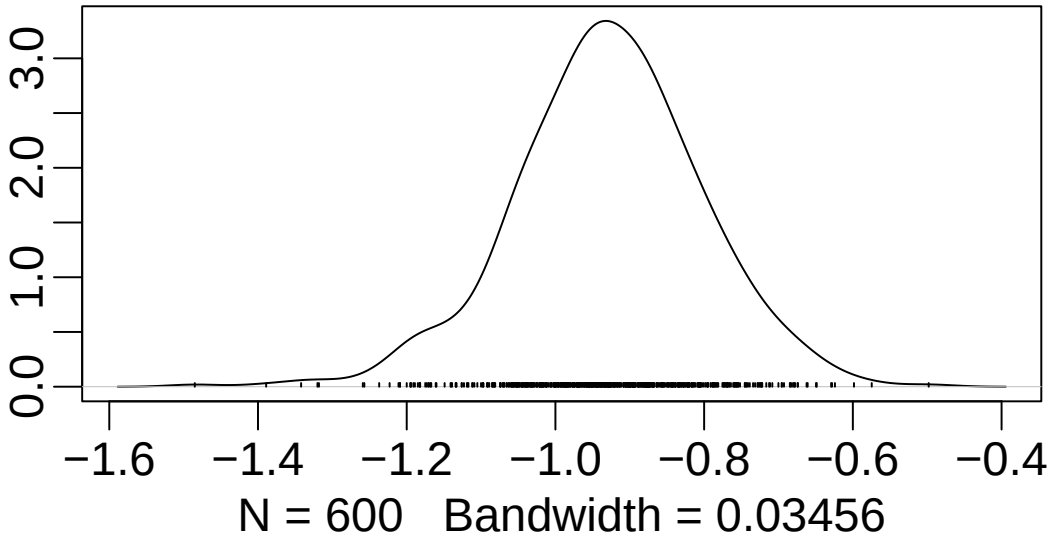
- 午後の「3. R と WinBUGS の使いかた」でくわしく説明します
- とりあえず WinBUGS というソフトウェアによって MCMC によって事後分布からのサンプルが得られた，としましょう



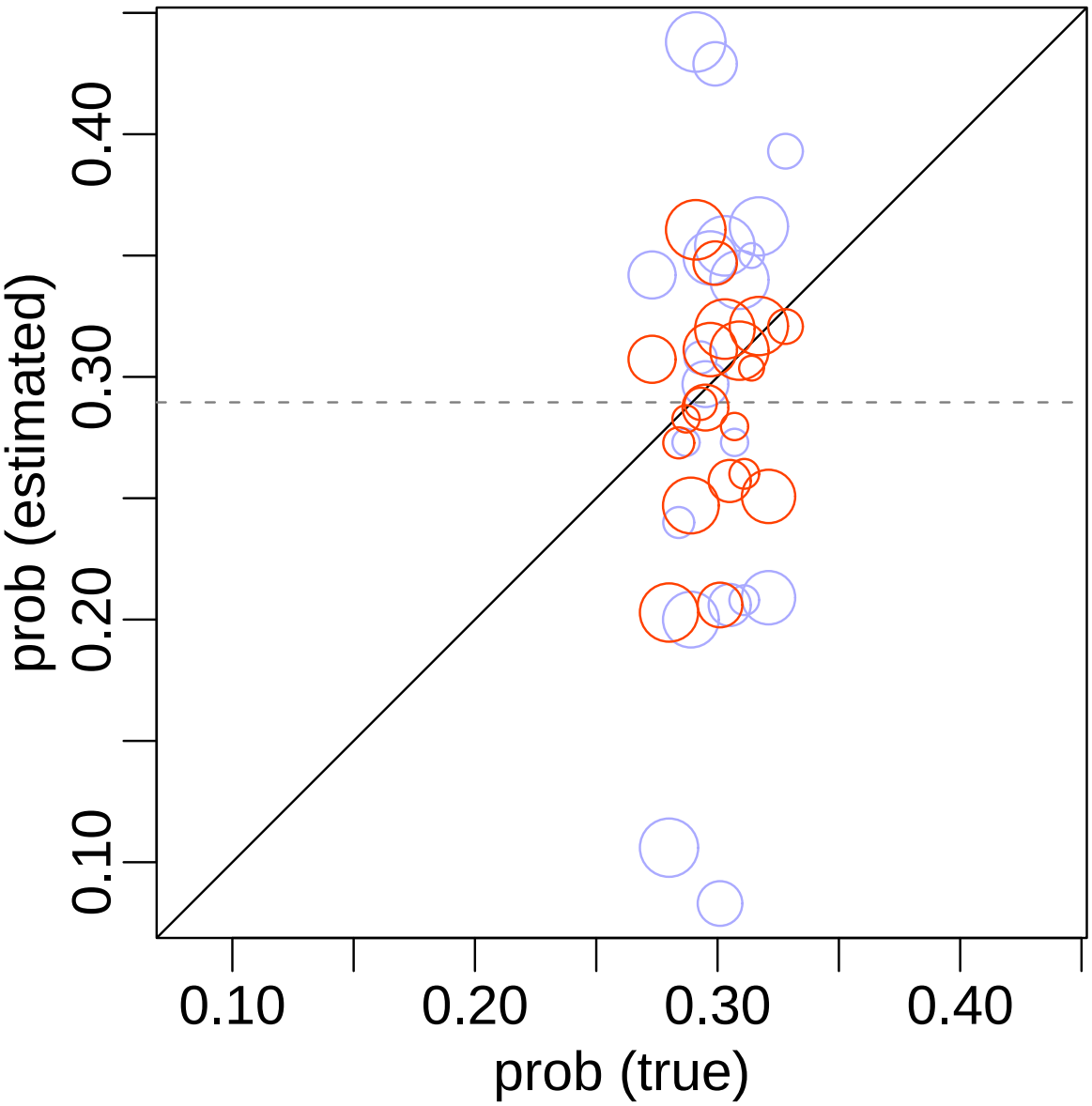
「事後分布からのサンプル」って何の役にたつの?

```
> 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

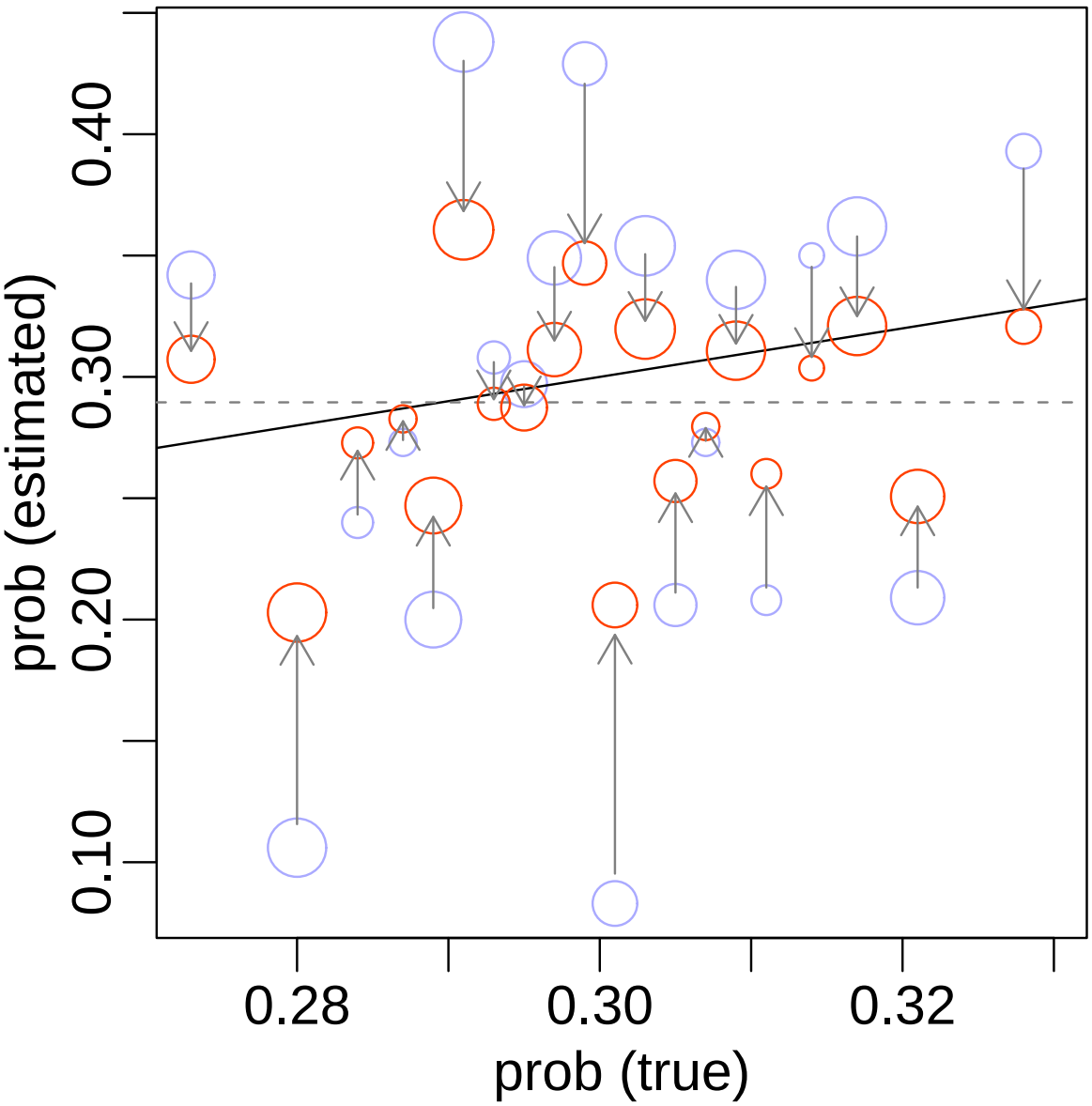


各打者の打率の中央値を図示してみる (赤)



- 多少はマシになった?
- とはいえ,「打者 **20** 名の打率はそんなに異ならない」という仮定 (事前分布) が反映されて「一割打者」「四割打者」はなくなった
- 図を拡大して「修正されぐあい」を見てみよう

集団平均に近づく方向に修正された



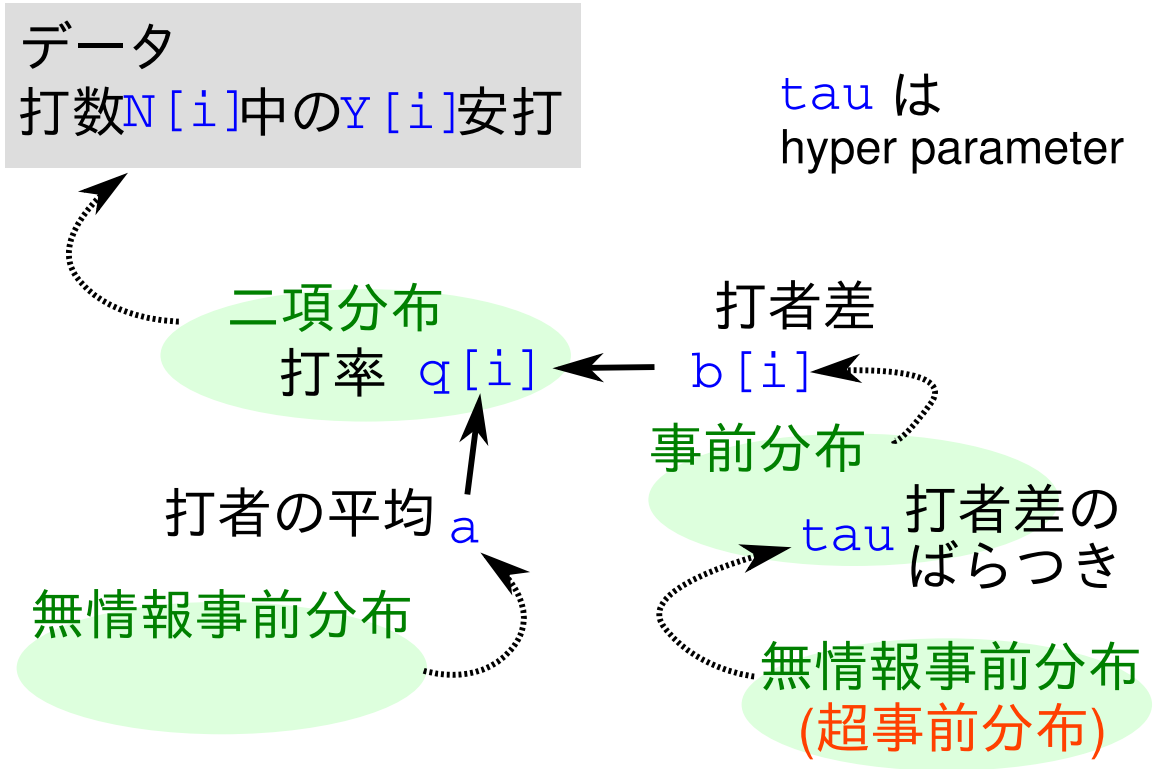
- 「それっぽく」補正されてる打者もいる
- しかし，ホントの打率から大きくずれてる場合には修正が難しい
- とりあえずの結論: 打数が少ないときに個々の打者で安打数/打数と推定するより，階層ベイズモデルで「全体を見つつ」推定したほうがマシ

2-3. とりあえずのまとめ

今日でてきた用語の整理

- 階層ベイズモデル

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

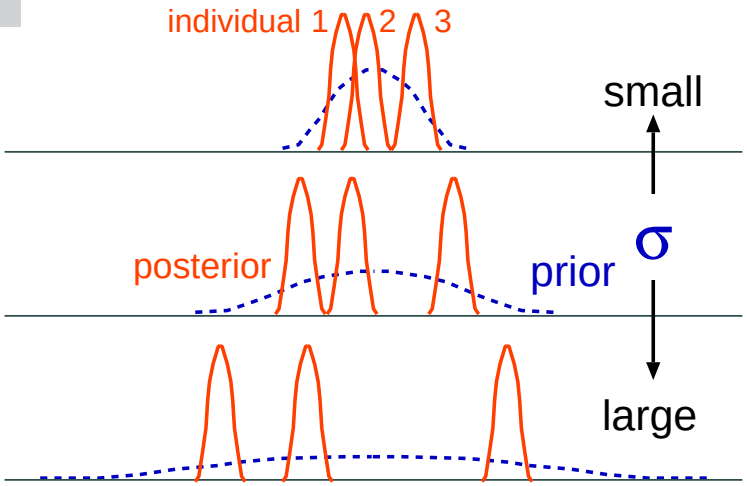


- 事後分布の推定計算方法: **Markov Chain Monte Carlo (MCMC) 法**

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?

打者差 b_i の事前分布の設定を例に

- 主観的な事前分布を使う (1)
  「自分は $\sigma = 0.1$ と信じるので，それを使う」
- 主観的な事前分布を使う (2)
  「今までのデータ解析で $\sigma = 0.1$ ぐらいとなっていたので，それを使う」
- 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」



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

ベイズな統計学とそうでない統計学の比較

	頻度主義な道具	ベイズな道具
確率とは?	客観確率	主観確率
パラメーターとは?	ある特定の値	確率分布
事前分布は?	ない	推定すべきパラメーターにあわせて様々な事前分布を設定
よく使う推定計算方法は?	最尤推定法	**MCMC 法**
パラメーターの何を推定?	最尤推定値	事後分布

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・……)
- 多重 **nest** した **random effects** の導入
- 「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**

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: **BUGS code** を書いて **WinBUGS** で事後分布からのサンプルを得る
- 事後分布を解釈する

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 関数セットを経由して

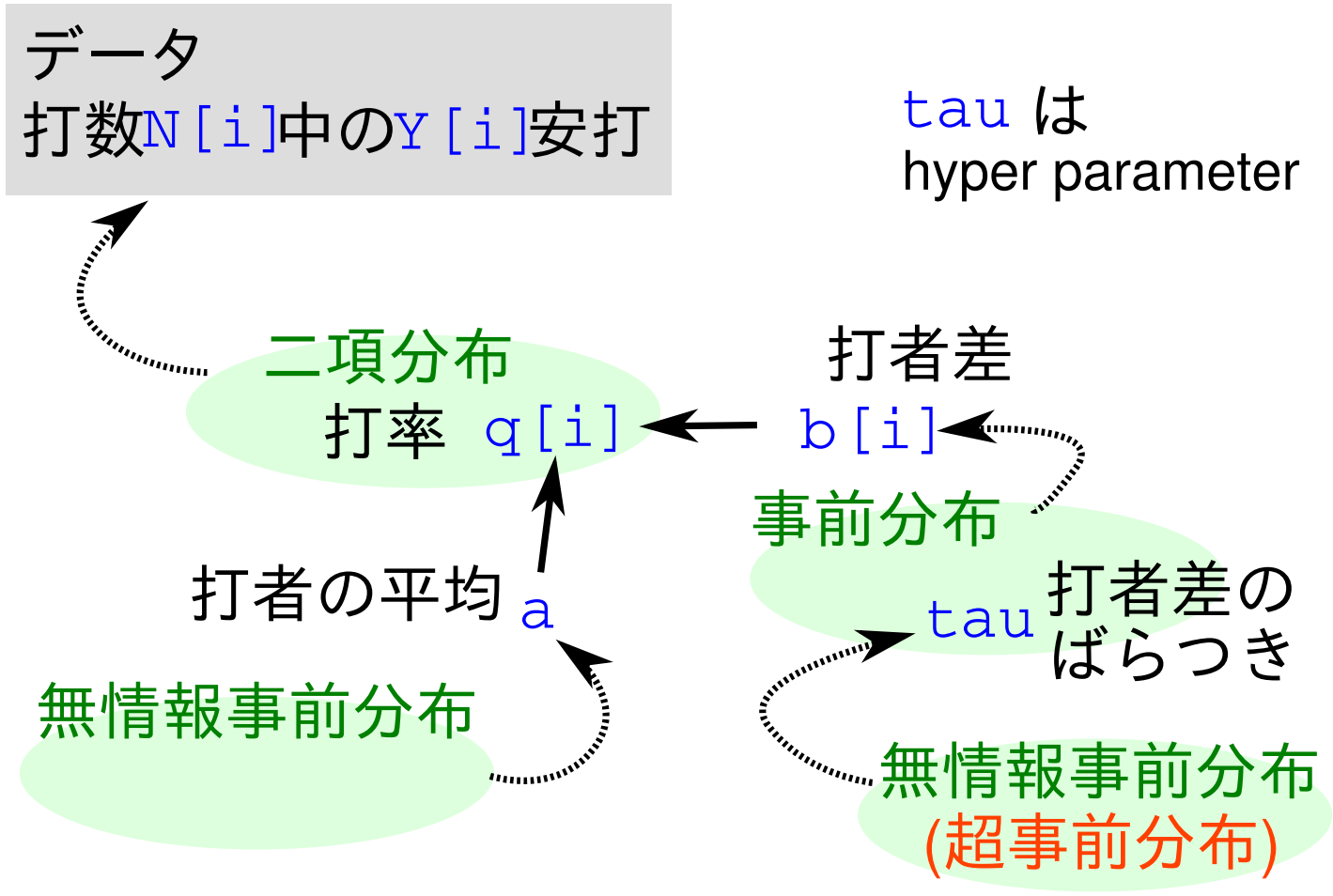
3. **「打率の推定」例題を WinBUGS で推定**

   実際に使ったコードを説明しつつ

3-1. MCMC をどんな

ソフトウェアで動かす?

なんで MCMC 計算って必要でしたっけ?

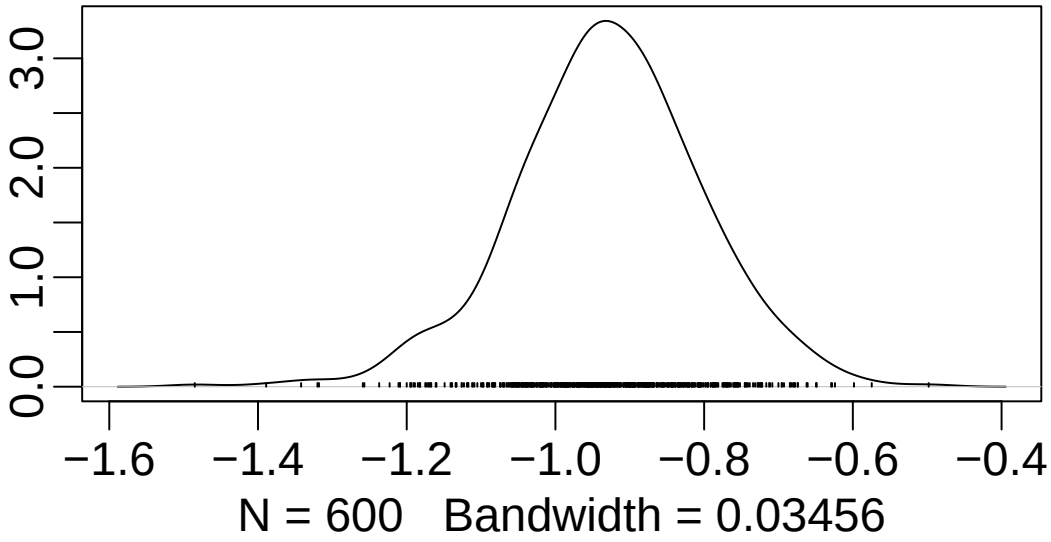


パラメーター a, b[i], tau の事後分布を知るために必要

「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: 「原因 $\longrightarrow$ 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……

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
    Tau.noninformative <- 1.0E-2
    P.gamma <- 1.0E-2
    for (i in 1:N.sample) {
        Y[i] ~ dbin(p[i], N[i])
        logit(p[i]) <- a + b[i]
    }
    a ~ dnorm(0, Tau.noninformative)
    for (i in 1:N.sample) {
        b[i] ~ dnorm(0, tau)
    }
    tau ~ dgamma(P.gamma, P.gamma)
}
# あとで説明
```

“BUGS” な 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`

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)

GPL な WinBUGS めざして: OpenBUGS 3.0.3

- Thomas Andrew さん他が開発している
- 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
- 2007 年 9 月以降新しいニュースなし
- どうなっているのかよくわからない

R な (?) Gibbs sampler: JAGS 1.0.3

- 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
- Linux, Windows, Mac OS X バイナリ版もある
- ぢりぢりと開発進行中
- R からの使う: `library(rjags)`

3-2. 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
- `library(R2WinBUGS)` をラップする R2WBwrapper 関数 (久保作) を使う

なんで WinBUGS を R 経由で使うの?

- WinBUGS の「ステキ」なユーザーインターフェイス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

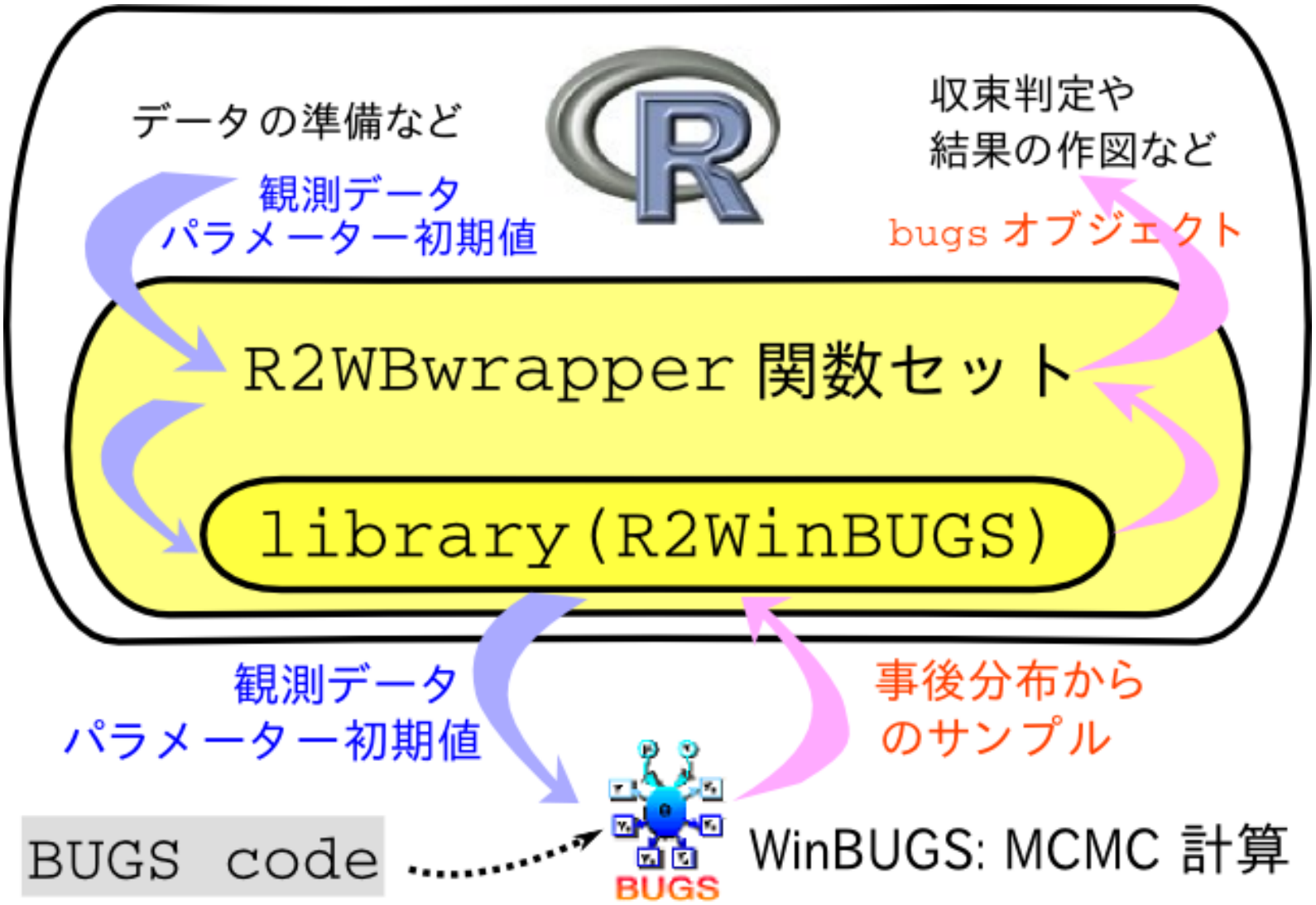
なんで R2WinBUGS をラップして使うの?

- R2WinBUGS の「ステキ」なインターフェイス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……
6. あるいは `mcmc` / `mcmc.list` 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)



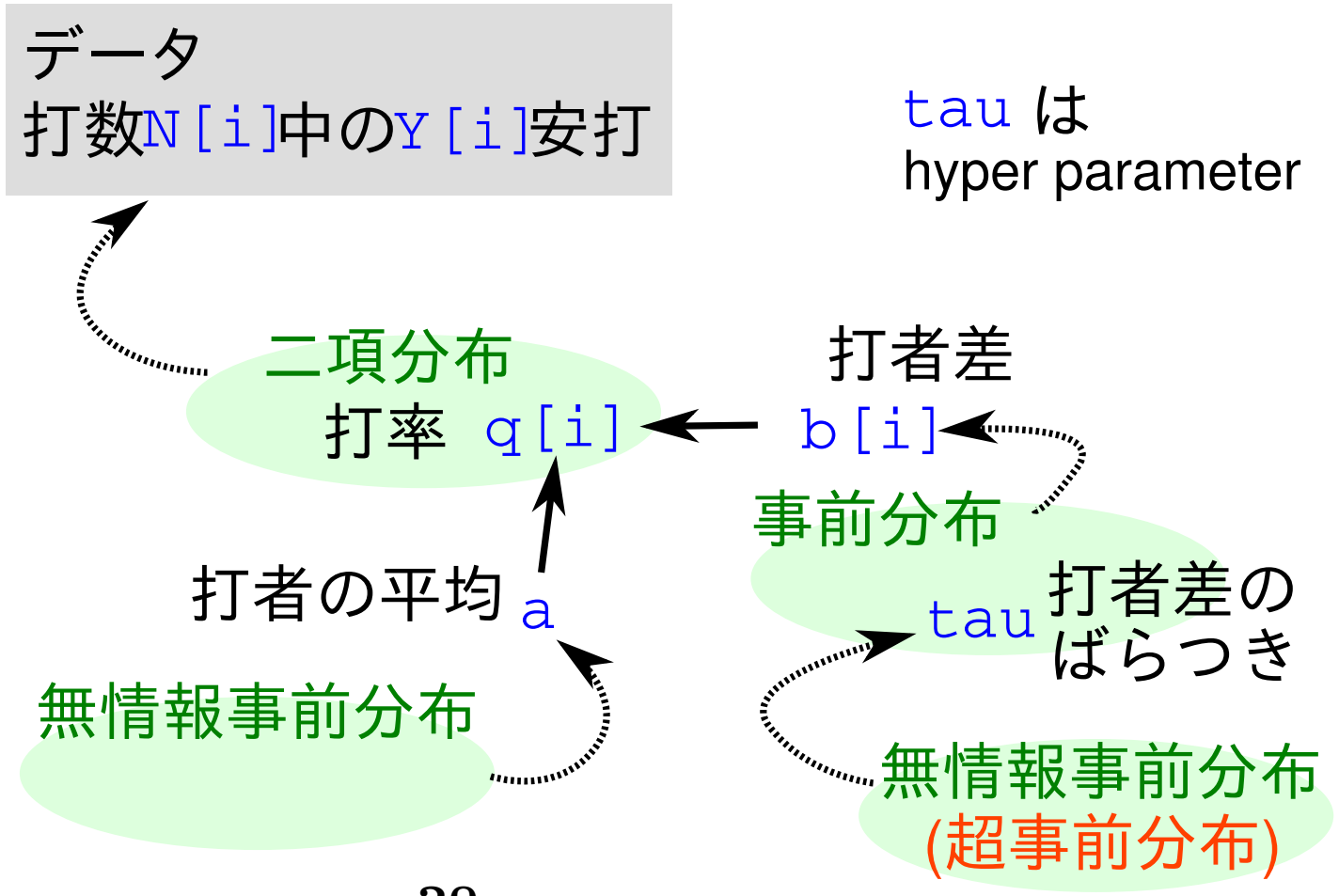
3-3. 「打率の推定」例題を

WinBUGS で推定

「打率の推定」例題を WinBUGS に推定させる手順

1. 「打率」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

「打率」の階層ベイズモデルってどんなでしたっけ?



$$p(a, \{b_i\}, \tau \mid \text{データ}) \propto \prod_{i=1}^{20} f(\text{データ} \mid q(a+b_i))\, g_a(a)\, g_b(b_i \mid \tau)\, h(\tau)$$

「打率」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(p[i], N[i])   # 観測値との対応
        logit(p[i]) <- a + b[i]   # 打率 p[i]
    }
    a ~ dnorm(0, 1.0E-2)          # 打者の平均
    for (i in 1:N.sample) {
        b[i] ~ dnorm(0, tau)      # 打者の個体差
    }
    tau ~ dgamma(1.0E-2, 1.0E-2) # 打者差のばらつき
    sigma <- sqrt(1 / tau)        # tau から SD に変換
}
```

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. ~ sthochastic node
  2. <- deterministic node

R2WBwrapper な R コード runbugs.R (前半部)

観測データの設定

```
source("R2WBwrapper.R") # R2WBwrapper よみこみ
load("d.RData")         # 観測データよみこみ

clear.data.param() # いろいろ初期化（まじない）
set.data("N.sample", nrow(d)) # データ数
set.data("N", d$N)            # 打数
set.data("Y", d$Y)            # 安打数
```

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

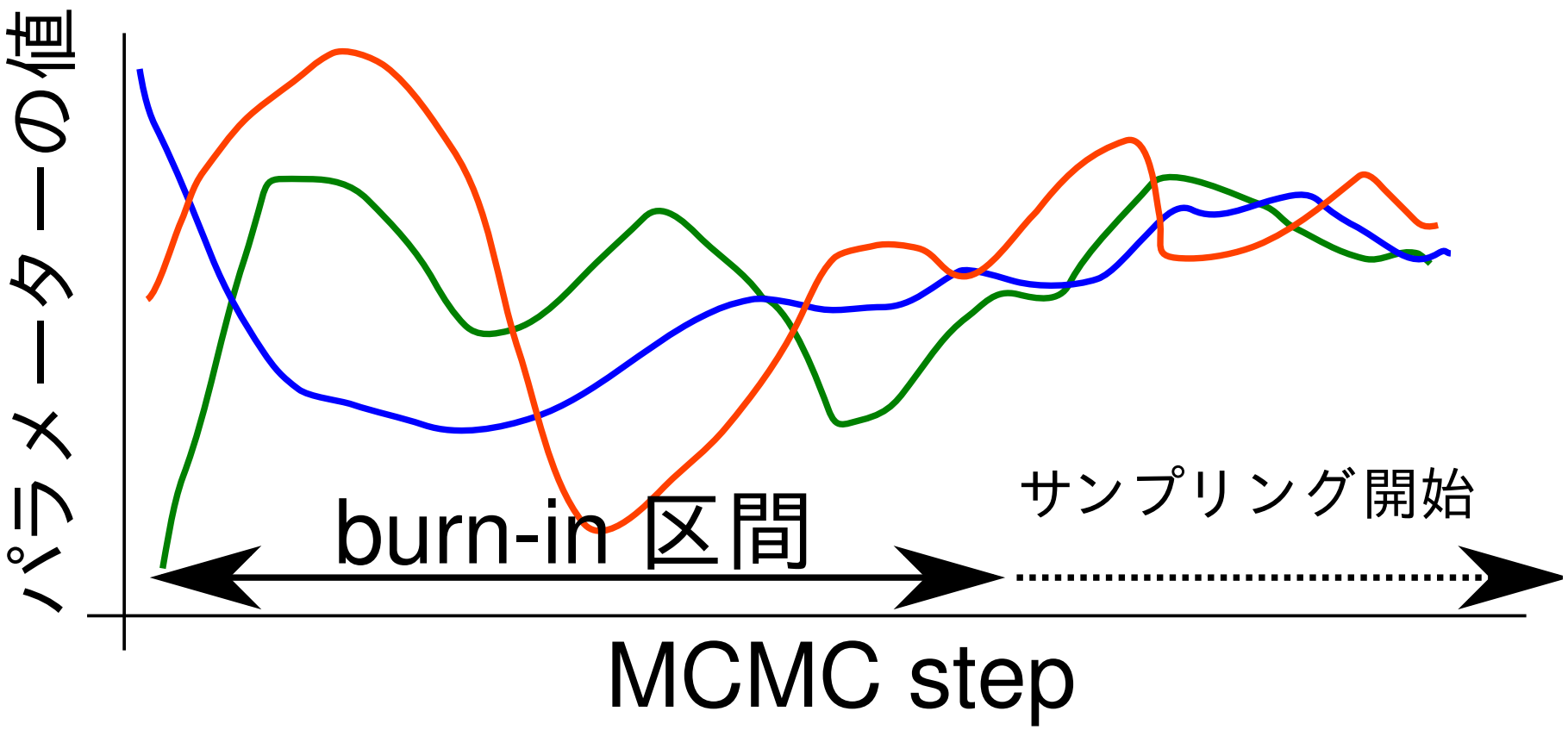
```
set.param("a", 0)       # 打者の平均
set.param("sigma", NA) # 打者の個体差のばらつき
set.param("b", rep(0, N.sample))  # 打者の個体差
set.param("tau", 1, save = FALSE) # ばらつきの逆数
set.param("p", NA)      # 打率

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

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 3 回(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)

“burn-in”: MCMC の最初のほうを捨てる



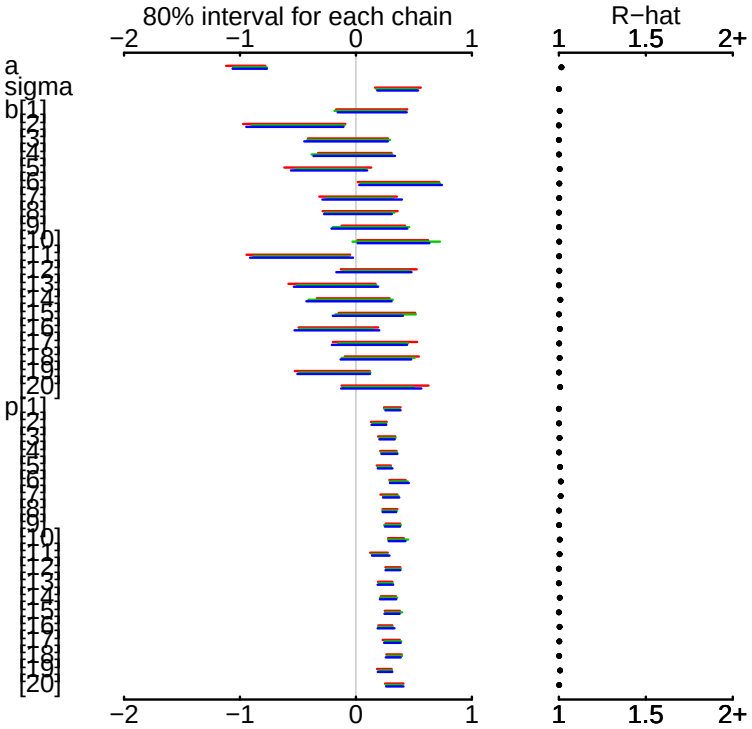
で，実際に動かすには?

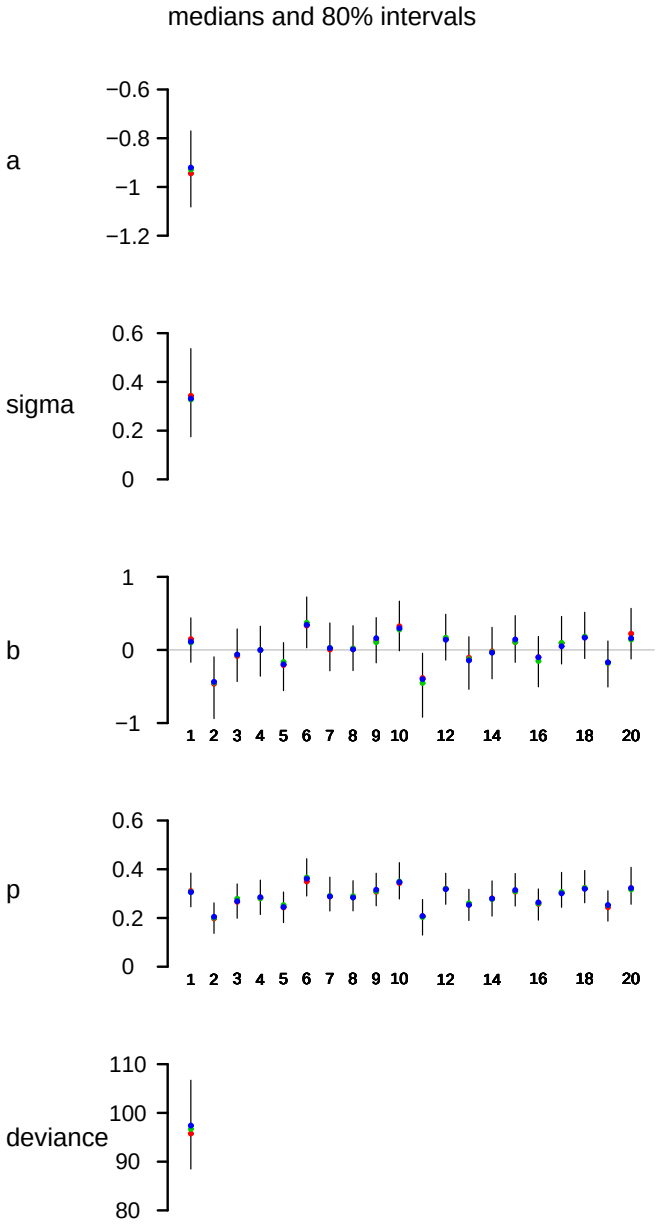
- たとえば，R 上で `source("runbugs.R")` とか
- すると WinBUGS が起動して MCMC sampling をはじめる
- この例題は簡単なのですぐに計算が終了する (WinBUGS 内で図などが表示される)
- 手動で WinBUGS を終了する
- すると WinBUGS が得た結果が R にわたされ，`post.bugs` というオブジェクトにそれが格納される

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
  - $\hat{R} = \sqrt{\dfrac{\hat{\mathrm{var}}^{+}(\psi|y)}{W}}$
  - $\hat{\mathrm{var}}^{+}(\psi|y) = \dfrac{n-1}{n}W + \dfrac{1}{n}B$
  - W : chain 内の variance
  - B : chain 間の variance
  - Gelman et al. 2004. Bayesian Data Analysis. Chapman & Hall/CRC

Bugs model at "/home/kubo/public_html/stat/2008/ism/fignew/model.bug.txt", fit using WinBUGS, 3 chains, each with 2000 iterations (first 1000 discarded)





bugs オブジェクトの post.bugs を調べる (2)

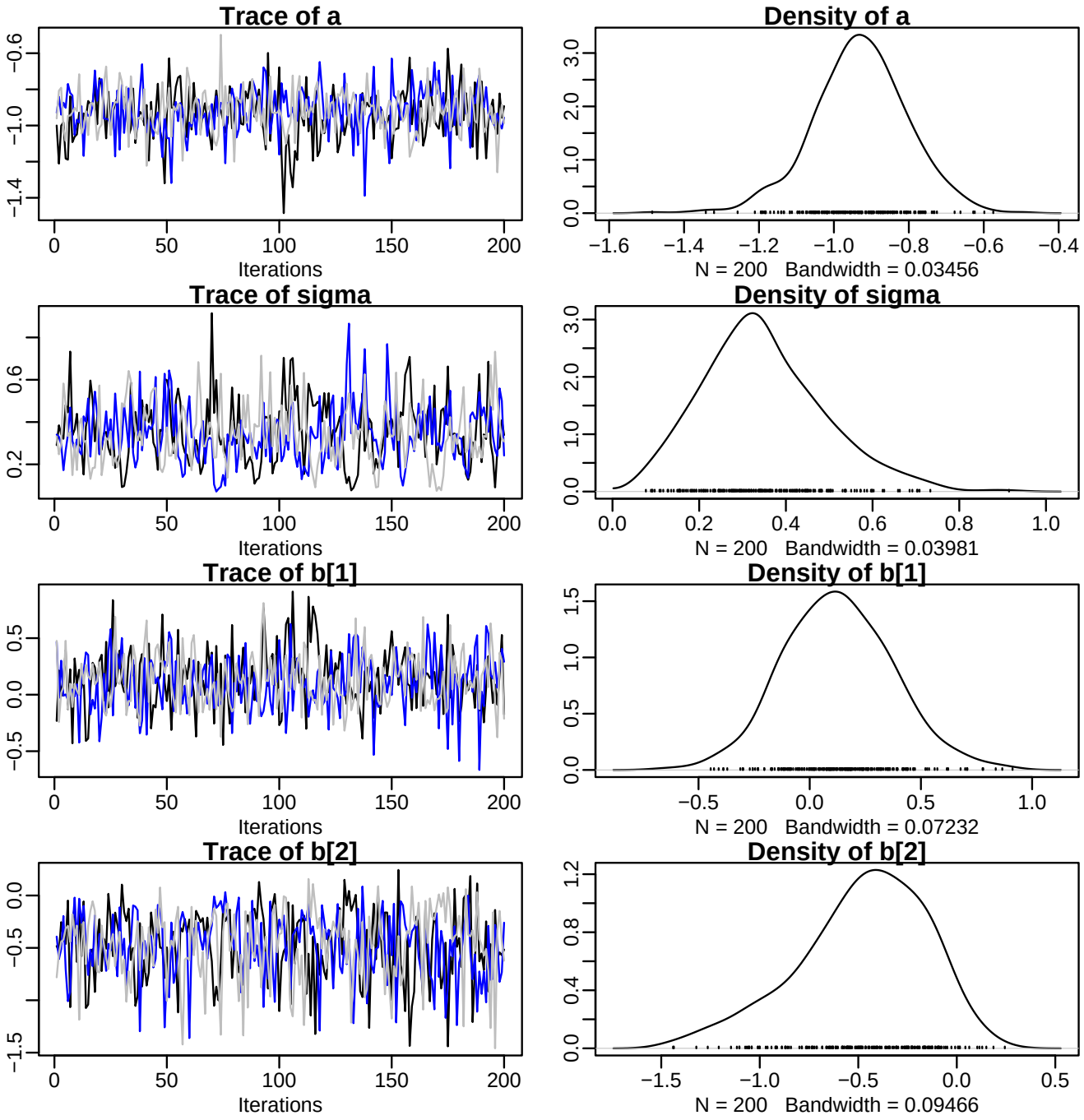
- `print(post.bugs, digits.summary = 3)`
- 事後分布の 95% 信頼区間などが表示される

```
... (略) ....
           mean    sd   2.5%    25%    50%     75%   97.5%  Rhat n.eff
a        -0.930 0.127 -1.194 -1.005 -0.930  -0.848  -0.682 1.015   170
sigma     0.348 0.140  0.099  0.250  0.333   0.431   0.663 1.000   600
b[1]      0.127 0.245 -0.342 -0.044  0.123   0.292   0.625 1.005   390
b[2]     -0.480 0.326 -1.237 -0.674 -0.443  -0.244   0.044 1.000   600
... (略) ....
b[19]    -0.186 0.263 -0.706 -0.353 -0.178  -0.018   0.339 1.001   600
b[20]     0.199 0.281 -0.283  0.013  0.160   0.393   0.747 1.007   300
p[1]      0.312 0.054  0.213  0.275  0.307   0.346   0.432 1.000   600
p[2]      0.201 0.049  0.108  0.167  0.203   0.238   0.300 1.001   600
... (略) ....
p[19]     0.250 0.049  0.161  0.217  0.251   0.280   0.351 1.007   290
p[20]     0.328 0.061  0.228  0.283  0.321   0.368   0.451 1.002   600
deviance 97.100 6.865 85.000 92.560 96.595 101.100 112.102 1.002   510
... (略) ....
```

mcmc.list クラスに変換して作図

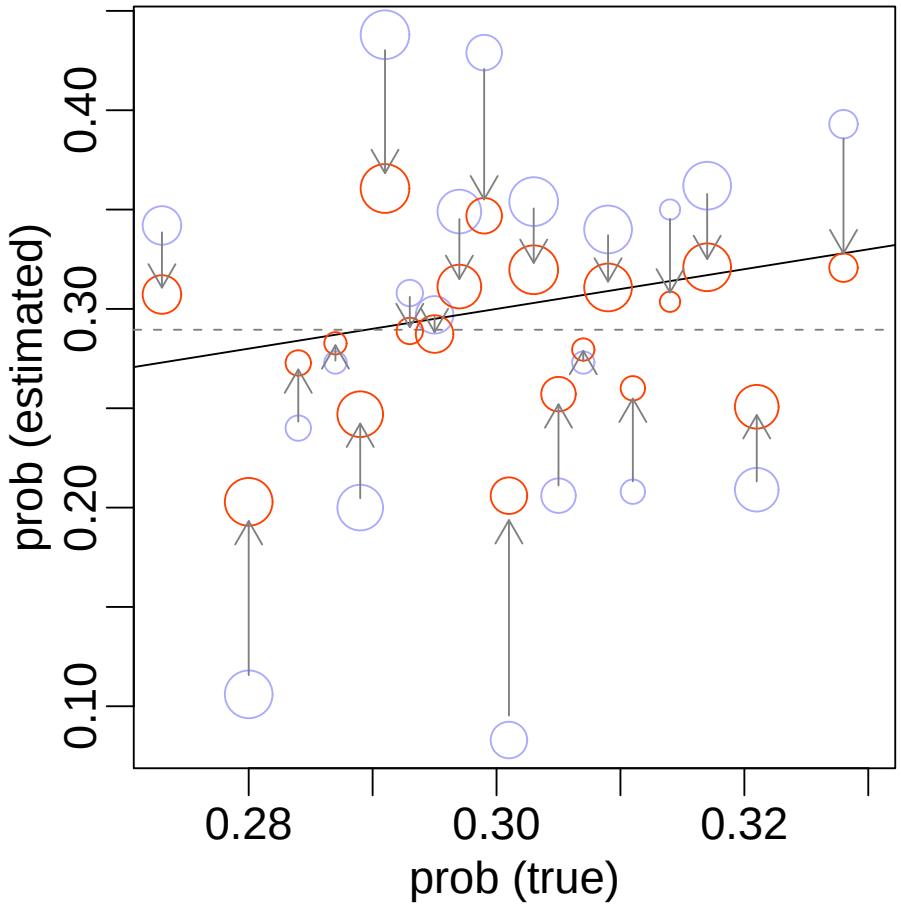
- `post.list <- to.list(post.bugs)`
- `plot(post.list[,1:4,], smooth = F)`

→ 次のページ, 実演表示


Trace of a
Density of a
Iterations
N = 200 Bandwidth = 0.03456
Trace of sigma
Density of sigma
Iterations
N = 200 Bandwidth = 0.03981
Trace of b[1]
Density of b[1]
Iterations
N = 200 Bandwidth = 0.07232
Trace of b[2]
Density of b[2]
Iterations
N = 200 Bandwidth = 0.09466

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



「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 関数セットを経由して

3. 「打率の推定」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://hosho.ees.hokudai.ac.jp/~kubo/ce/IsmBayes2009.html

- 本日の例題のデータなどは上記 URL のペイジからダウンロードできます
  - この公開講座のペイジからリンクされています
- 本日の投影資料・修正した配布資料も近日中にダウンロードできるようにします
  - 早めに対処したいです
- このあとは，伊庭さんによるベイズ & MCMC のより発展した話題の講座があります