マルコフ連鎖モンテカルロ法の基礎と実践 (2009 年 2 月 23 日)

第 2–3 時間目

2. ベイジアンモデリングとMCMC

3. R と WinBUGS の使いかた

担当: 久保拓弥 kubo@ees.hokudai.ac.jp

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

もくじ

私 (久保) が担当する第 2, 3 限目では，**Markov Chain Monte Carlo (MCMC) 法**の応用例として，**階層ベイズモデル**のパラメーター推定，つまりパラメーターの事後分布をもとめる方法を紹介します．階層ベイズモデルは最近になって多くの分野でひろく使われるようになった統計モデルの一種です．その特徴はモデリングの柔軟性にあり，たとえば個体差・ブロック差・場所差がある標本 (観測データ) の背後にある規則性などを調べたいときに使われています．

なお，上記 URL (以下，久保サイト; 公開講座のペイジからもリンクされています) には本日のここで使う例題のデータ，プログラム，投影資料が (そしておそらくもうちょっと改訂された配布資料も?) ダウンロードできるようになっています．のちほどアクセスされてみてください．

1. ベイズ統計モデルとは? 非ベイズと比較してみる

ベイズではない統計学は**頻度主義** (frequentism) な統計学とよばれています. で，歴史的には非ベイズとベイズな人たちはキビしく対立してきた経緯があり— いまでも原理的にはあいいれない対立なのですが，とりあえずわれわれは「便利な道具なら何でもいいや」といった態度の統計学ユーザーなので，そのあたりはあまり気にせずハナシを進めていきたいと思います.

「そもそもベイズ統計学って何?」というおおざっぱな質問に対するおおざっぱな回答は以下のとーりです:

- **主観確率** をあつかう，[1] つまり「明日の降水確率は 20%」の解釈は「雨のふる・ふらないは 2:8 ぐらい，といった割合でおこりそうな事象だと考えている」となる
  - これに対して頻度主義な統計学では，**客観確率** をあつかう，つまり「明日の降水確率は 20%」の解釈は「明日という日が無限個存在していたとすると，そのうち 20% では雨が降っていると観測される」となる
- 確率分布の**パラメーターもまた確率分布**として表現される
- この「パラメーターの確率分布」には**事前分布** (prior) と**事後分布** (posterior) がある
- 事前分布は「パラメーターの値はこのへんにありそうだな」をあらわす確率分布，解析者が決める; ただし実際のデータ解析では (**後述するように**) 次のふたつの事前分布をよくつかう
  - 無情報事前分布: 「パラメーターの値はどこにあってもいいんですよ!」ということを表現する事前分布，つまり「べたーっとひらべったい事前分布」— fixed effects 的な効果を表現するパラメーターの事前分布などとして使う
  - 階層モデル的な事前分布: 事前分布のカタチを観測データと超パラメーター (hyper parameter) で決める，超パラメーターは無情報事前分布をもつ— random effects 的な効果を表現するパラメーターの事前分布などとして使う
- 事後分布は「ベイズ統計モデル」と観測データから**推定**される— つまり観測データから事後分布を推定することがベイズ推定 (Bayesian inference) の目的である
- 上でいう「ベイズ統計モデル」は次のふたつの部品からなる
  - 尤度(ゆうど): つまり統計モデルが観測データにどれぐらいあてはまっているか，を確率分布とそのパラメーターであらわす確率 (確率密度) — これまでこの授業でとりあつかってきた尤度と同じ
  - 事前分布 (と超事前分布): 尤度にふくまれるパラメーターをあらわす確率分布 (確率密度関数)

1. 主観確率・客観確率のどちらも公理主義的確率論が要請する条件を満たしています— つまり確率論のさまざまな道具だてを援用できます．つまるところ主観確率・客観確率という区別は,「確率って何なの?」という確率の解釈に関する相違なのです.

ついでに，頻度主義な統計学的ツールとの比較のテイブルでもこしらえてみましょう．

	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**，世界のどこかに「真の値」があるはず	**確率分布**で表現できることにする
事前分布は?	ない (強いていえば無限の幅をもつ一様分布)	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する
観測データ使って何を推定する?	「真の値」に近いはずのパラメーター**最尤推定値**，つまりひとつの値 (点推定値)	パラメーターの**事後分布**，つまり確率分布
推定計算の基本わざ	最尤推定法	Gibbs samling など; 階層ベイズモデルの最尤推定は「経験ベイズ法」と呼ばれる
推定されたパラメーターの信頼区間 α の意味は?	じつは難解	簡単，そのパラメーターが確率 α でとりうる範囲

このような抽象的な対比では何がどう具体的に異なるのか，よくわからないかもしれません．あとから簡単な構造の架空データを使った例題の解析でベイズ統計モデリングに特有な考えかたを応用する方法を具体的に示していきます．

ともあれ，従来の統計学とは異なる考えかたであるにもかかわらず，驚くべきことにと申しますか，いまや統計ユーザーの道具としてベイズ統計モデリングは「非ベイズなわくぐみではうまく表現できなかった現象 (観測データの中にみられるさまざまなパターン) をうまく表現できるように**強化された統計モデリング技法**」として普及しつつあります．

2. 本日の架空データ解析の例題: 打率の推定

ここでは (MCMC 法の応用例のひとつである) 階層ベイズモデルという考えかたを説明するために，(かなり現実感のトボしい) 架空データを解析する例題を考えていきます．

ここでは (アマチュアだかプロだかの) 野球選手の **打率**をあつかいたいと思います．観測データは以下のような想定のものとします:

- 各チームの **3-5 番打者だけ**を選んだ
- 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

打率の定義としては，安打数わる打数，つまり Y_i/N_i なのですが，今日のハナシは

- N_i が小さいときに Y_i/N_i で求めた打率はトンでもない数値になる
- そこでこのような割算推定値はヤメて，階層ベイズモデルを使えば多少はマシに「補正」される……(場合もありますよ)

といったナガレになります.

ここから先の説明では，私が統計ソフトウェア R を使っているような想定でハナシをすすめていきます (R については久保サイトにつけているリンクなどを参照してください).

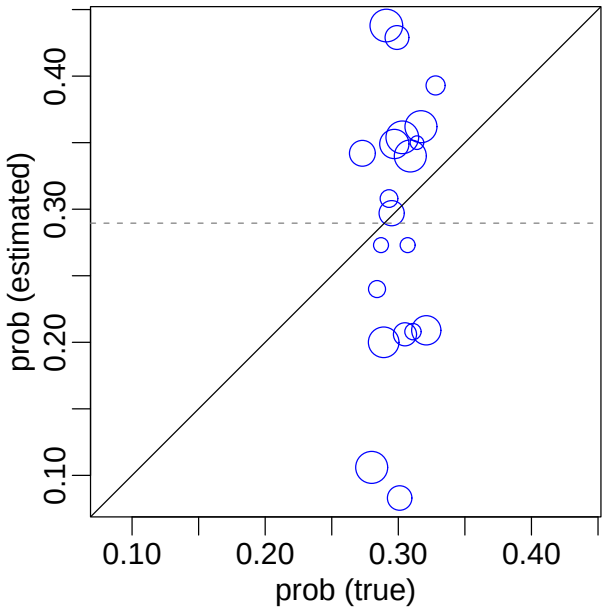
```
> load("d.RData") # R 内でデータを読みこむ
> print(d)      # 読みこんだデータの表示
   id p.true  N  Y p.naive
1   1  0.273 38 13   0.342
2   2  0.280 47  5   0.106
3   3  0.284 25  6   0.240
4   4  0.287 22  6   0.273
5   5  0.289 45  9   0.200
6   6  0.291 48 21   0.438
7   7  0.293 26  8   0.308
8   8  0.295 37 11   0.297
9   9  0.297 43 15   0.349
10 10  0.299 35 15   0.429
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
```

上の各列について説明すると:

- id 列: 打者番号 ($i = 1, 2, \cdots, 20$)
- p.true 列: 「ホントの打率」
- N 列: 打数
- Y 列: 安打数
- p.naive 列: 単純に安打数 / 打数で得られた打率

「ホントの打率」とはいかにもウソくさい数値ですが，これはこの架空データを二項乱数で作るときに使った確率です．今回は，単純な安打数 / 打数のアヤうさと，階層ベイズモデルでそれをどこまで「補正」できるか，に興味があるので，あまり気にせずにこの p.true と推定結果を比較していきましょう．

たとえば p.true 列と p.naive を比較するとこのようになります．円の大きさは N_i をあらわしています．



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「各チームの 3-5 番打者なのに一割も打てないヤツ」から「四割以上」まで混在していることになります．

もちろんこのように推定されてしまう原因は，ひとえに打数 N_i が少なすぎることにあります．それではいま手もちのデータはまったくの無価値で，このめちゃくちゃな推定値はどうにもならないものなのでしょうか? 20 名もの打者からデータをとったのに，それが何の役にも立たない……?

基本的にはデータの無いところから情報は推定できません．しかしデータの

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しれません．このあと徐々に構築していく階層ベイズモデルでは,「データの背後にある構造」として，次のように仮定してみます:

> **これらの打者 20 名の打率はそんなに異なるものではない**: 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 より低い打者もいないだろうし，いっぽうで長期的に打率 0.4 以上を維持できる打者もいないだろう．

このような構造があるものとして想定します.「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?」と疑問にもたれるのは当然ですが，それはこのハナシのあとのほうで観測データと階層ベイズモデルが**自動的に**決めてくれます.

まずは，階層ベイズモデルを構築するのに必要な部品，**確率分布，ロジスティック関数，事前分布**をひとつずつ導入していきましょう.

3. 打率 q と二項分布の関係

ここから先は打率を推定するために **二項分布**という確率分布を使います.「二項分布とは何か?」を説明するために，次のような簡単化した推定計算を考えてみましょう．ここではいったん**「打者差はない」**と仮定してみます.

すべての打者で打率 q が共通しているという仮定のもとでは，打者 i の N_i 打数の中で安打数 Y_i 本となる確率は二項分布

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

で表現できます．20 打者の観察値 $\{Y_i\} = \{y_1, y_2, \cdots, y_{20}\}$ が観察される確率は上の $f(\text{データ} \mid q)$ を 20 打者ぶん掛けあわせたものになります．このときに，逆に観察データ $\{Y_i\}$ 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と，この 20 打者ぶんの確率はパラメータ q の関数となります．これは尤度とよばれ，形式的には

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

と定義されます．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 $\hat{q}$ を計算してみましょう．尤度を対数尤度になおすと

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

となります．この対数尤度を最大化するように未知パラメーター q の値を決めてやることを，最尤（さいゆう）推定とよびます．対数尤度が最大になる状態とは，観測データに対してこの二項分布がもっとも良くあてはまっている状態です．

上で定義されている対数尤度の q に関する微分がゼロになる q を計算すると，$\hat{q} = \sum_{i=1}^{20} Y_i / \sum_{i=1}^{20} N_i$ つまり「(全打者の安打数) / (全打者の打数)」という割算値が最尤推定量になっています．最尤推定値は

```
> (q <- sum(d$Y) / sum(d$N)) # 割り算で得た打率の最尤推定値
[1] 0.28951
```

つまり，全打者の打率が等しいと仮定した場合の打率の最尤推定値は $\hat{q} = 0.290$ ぐらい，となります．

4. 統計モデルに打者差パラメーターをいれてみる

実際のところ打率 q は打者によって異なっています．そこで安打する確率 q が打者によって異なるよう統計モデルを拡張する準備として，安打する確率 $q(z)$ をロジスティック (logistic) 関数 [2] $q(z) = 1/\{1 + \exp(-z)\}$ で表現することにします．

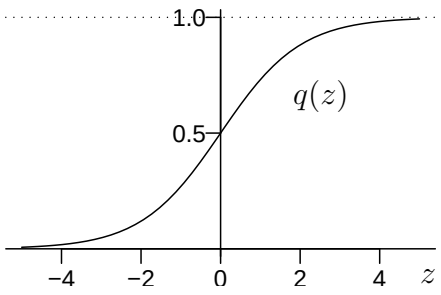
2. なお logistic 関数の逆関数が logit 関数です．

ある打者 i の z を $a + b_i$ とすると，

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

といったぐあいに**全打者の平均** a と**打者差** b_i に分割します．[3] 二項分布モデルの尤度方程式は

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



3. このような定式化によって，a, b_i どちらも $[-\infty, \infty]$ の範囲の値をとれるようになる，という利点が生じます．

となります．

打者差なしモデルのときのように，この尤度を最大化するようなパラメータを推定すればよいのでしょうか? 打者差をあらわすパラメータ $\{b_1, b_2, \cdots, b_{20}\}$ は 20 個あり，これに加えて安打する確率のうち全打者共通部分 a を加えると 21 個のパラメータを推定することになります．多少工夫すれば推定すべきパラメータ数を 20 個にできます．いずれにせよ 20 打者の挙動を説明するために 20 個以上のパラメータの推定値を確定しています……これでは，あの「安打数 / 打数」という割算推定と同じ結果になってしまいます．

5. いろいろな事前分布

打者差はモデルにいれなければならない，しかし打者差パラメータ $\{b_i\}$ を 20 個も最尤推定するのはいかにもおかしい，という状況を改善するために**階層ベイズモデル**をじりじりと作っていきます．

さて，それではこの例題におけるベイズモデルはどう定式化されるのでしょうか? 端的に言うと全 20 打者の打者差 b_i をいちいち最尤推定しない，ということになります．つまり b_i の推定結果として何か特定の値で表現するのではなく，としないで確率分布としてあらわすことにしよう，という方針転換です．これによってモデルの柔軟性が高まります．

しかしながら，打者差 b_i の確率分布は好き勝手に決めてよし，と許可してしまうとかなり無秩序な推定結果になります．そこで，各 b_i の確率分布は観察データ $\{Y_i\}$ と「観察された 20 打者の打率には，どこか似ている部分がある」というルールによって制約してしまいたい，つまり「観察データをうまく説明できる範囲で，**打者たちはできるだけ似ている** (b_i がゼロに近い) となるように b_i を決めようね」と，なかなか都合のよいことをもくろんでいるわけです．このように $\{b_i\}$ を制約する役目を与えられた確率分布をベイズ統計学では事前分布とよびます．これに対して，観察データと事前分布で決まる b_i の確率分布は事後分布です．

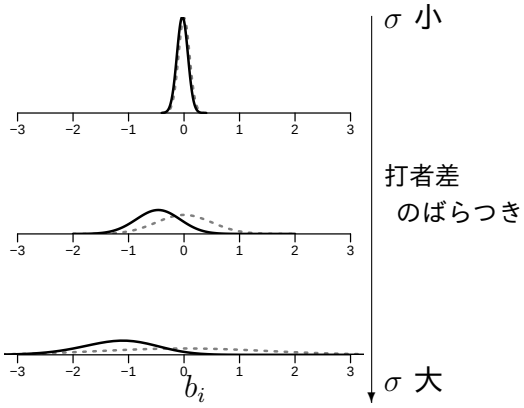
この打者差 b_i の事前分布は，ここでは簡単のため平均ゼロで標準偏差 σ の正規分布

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

で表現できることにしましょう．ここで登場した τ は分散の逆数，つまり $1/\sigma^2$ です．これについてはあとで説明します．

観察された打者全体に共通するパラメータ σ は，この打者たちはおたがいどれぐらい似ているかをあらわしていて，たとえば，

- 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

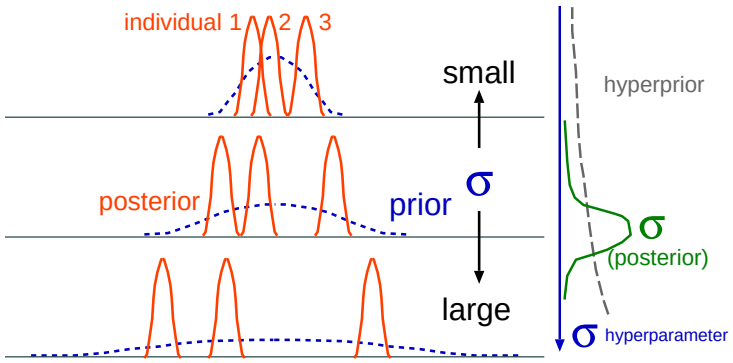


といった状況が表現できるようになりました．
ある打者 i の b_i の事後分布が事前分布 $g_b(b_i \mid \sigma)$
に依存している様子を示します．ここで灰色の
破線は $\{b_i\}$ の事前分布，黒い実線はある打者 i の b_i の事後分布です．全体のばらつき σ が小さいと b_i はゼロに近く，σ が大きいときには事前分布による制約 [4] が弱くなるのでゼロからずれています．

4. ゼロ付近に集まれというしばり，「個性」なんて主張するな，という束縛ですね．

観察された 20 打者のばらつきをあらわすパラメータ σ をどう決めるか，が重要な問題になります．しかし，ここではこの問題を先おくりすることにして，ただ単にパラメータ σ もまた何か確率分布 $h(\sigma)$ にしたがう確率変数だ，ということにしてしまいます．これは事前分布のパラメータの事前分布なので**超事前分布** (hyper prior) とよばれています．

ここまでの概念を**やや不正確**でおおざっぱな図でまとめると，このようになります．



打者差 b_i はこのような，いわば**階層的な事前分布**をもつ，といえます．このような「幅」をデータにあわせて選べる事前分布の使用がベイズモデリングで重要になります．

いっぽうで,「平均的な打率 (の logit 変換値)」をあらわす a の事前分布である $g_a(a)$ と σ の超事前分布である $h(\sigma)$ はどのように設定すればいいのでしょうか? これらのパラメーターに関してはどういう値をとってもらってもかまわないので，その事前分布は**無情報事前分布** (non-informative prior) と設定します．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」，[5]

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

なんかを使うことにしましょう．

5. logit 変換された世界の中で「すごくひらべったい」といった意味です

つぎに「打者たちのばらつき」こと σ に関しても情報がないので，σ もやはり無情報事前分布になりそうです．では σ はどんな確率分布にすればよいのでしょうか? ベイズの世界には **共役な事前分布** (conjugate prior) なる考えか

たがあります．これは「パラメーター α の事前分布を共役な事前分布である X 分布にしておくと，α の事後分布も X 分布になる」ということがわかっている確率分布です．正規分布の分散 (σ^2) パラメーターの共役な事前分布は逆ガンマ分布です．これは分散の逆数 $\tau\ (=1/\sigma^2)$ がガンマ分布にしたがう，というものです．そこで τ の事前分布を「すごくひらべったいガンマ分布」と指定します．

ベイズ統計なヒトたちの中には**主観的な事前分布**を使うのが重要だ，主張されるかたがたもいます．この例題でいえば「これまでの経験･観測から，各チームの 3, 4, 5 番打者は平均 0.29 ぐらいで SD は 0.1 ぐらいだから，これを打者 i の打率 q_i の事前分布にしよう」といったご意見になりましょうか．

もちろんそのようなモデリングも可能ですが，私自身はこの方式は可能なかぎり回避しています．なぜかというと上で述べた「階層的な事前分布」と「無情報事前分布」を組みあわせて使えば，多くの問題には対処できるからです．

事前分布についてまとめると，事前分布には三種類ばかりあり，その使いわけは

- 階層的な事前分布: 打者差とか地域差といった，ある範囲でまとまっているけれど個別の値をとると考えられるパラメーターの事前分布として使う (全体の類似度を調節する超パラメーターが必要)
- 無情報事前分布: 「どんな値でもよい」パラメーターの事前分布として使う
- 主観的な事前分布: あまり使いたくないけど ……使わざるをえない場合もある (測定誤差の事前分布など)

といったところで，これらを組みあわせて階層ベイズモデルを構築していくことになります． [6]

6. 今回は主観的な事前分布は使いませんが．

6. 階層ベイズモデルと事後分布

さて，いよいよベイズの公式をもちだして推定すべき事後分布を定義してみましょう．ベイズの公式とはこういった関係をあらわすもので，

$$p(P \mid D) = \frac{p(D \mid P) \times p(P)}{p(D)}$$

ここで $p(P \mid D)$ 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 (つまりこれが事後分布); $p(D \mid P)$ はその逆でパラメーターを決めたときにデータが得られる確率となるのでこれは尤度 [7] ; $p(P)$ はあるパラメー

7. ただし，尤度関数の定義なんかでは形式的に $L(P \mid D)$ と書くのがふつうです．

ター P が得られる確率なので事前分布の定義; そして分母の $p(D)$ は「てもとにあるデータ D が得られる確率」というナゾの確率になります．書きなおしてみると，

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

となり，この関係に例題の統計モデルをあてはめると，観察データのもとで

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

となります．この中で右辺の分母が計算困難な値なのですが，ともかくパラメーターに依存してない**ナゾの定数**だということはわかります．分母が定数なので，

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

すなわち，事後分布の確率密度は尤度 (観察データのもとでの) と事前分布・超事前分布の確率密度の積に比例しています．

打者差パラメータ b_i などの事後分布を得るためにその事前分布 $g_b(b_i \mid \tau)$ が必要で，さらにこの事前分布を決めるために超事前分布 $h(\tau)$ が必要になる，といった階層構造があるので，[8] このような統計モデルは**階層ベイズモデル**と呼ばれています．

8. ここでいう階層構造なるものが，事前分布は超事前分布に制約されていて，といった nested な関係に限定されていることに注意してください．

なんだかややこしそうになってきましたが，観測データからこのモデルを特徴づけるパラメーターは推定できるのでしょうか? 階層ベイズモデルのパラメータを推定する方法としては**経験ベイズ法**と **Markov Chain Monte Carlo (MCMC) 法**がよく使われています．ここでは MCMC 法をつかって事後分布を推定します．

7. 階層ベイズモデルの MCMC 計算

階層ベイズモデルによって，パラメーターの事後分布はこのように定義されました．

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

しかしながら，コンピューターを使ってもこの事後分布 $p(a, \{b_i\}, \tau \mid \text{データ})$ を直接に導出することは困難そうです．

そこで事後分布を直接に計算するのではなく，(一限目に伊庭さんが説明された) MCMC 計算を適用します．この MCMC 計算は事後分布からのランダムサンプルセットを得る方法です．あとから示すように，そのような「事後分布からの無作為抽出乱数」がたくさんあれば，このモデルで使っているパラメーター a や $\{b_1, b_2, \cdots, b_{20}\}$ や τ の概要がわかります．

ここでは，Gibbs sampler ソフトウェアを使って事後分布を推定します．(すでに説明があったように) Gibbs sampling とは MCMC 計算の一部であり，効率よく事後分布からのランダムサンプルを可能にする方法です．

R 用の使いやすい Gibbs sampler がないので，WinBUGS というソフトウェアを使うことにします．ただしこれを「R の部品」であるかのように使うことにします．まず，この打率の階層ベイズモデルを BUGS 言語で coding します (コード中，打率 q_i が p[i] になっています……わかりにくくてすみません)．

```
model
{
    for (i in 1:N) { # 全打者 i = 1, 2, ..., 20 について
        Y[i] ~ dbin(p[i], N[i]) # 二項分布
        logit(p[i]) <- a + b[i] # 打率
    }
    Tau.noninformative <- 1.0E-2 # 分散の逆数
    a ~ dnorm(0, Tau.noninformative) # 無情報事前分布
    for (i in 1:N) {
        b[i] ~ dnorm(0, tau) # 打者差
    }
    P.gamma <- 1.0E-2 # 無情報ガンマ事前分布のパラメーター
    tau ~ dgamma(P.gamma, P.gamma) # 無情報事前分布
    sigma <- sqrt(1 / tau) # 分散逆数→標準偏差 (output 用)
}
```

今回の例題は比較的簡単なので，BUGS コードも単純なものになりました．もう少し解説してみると，

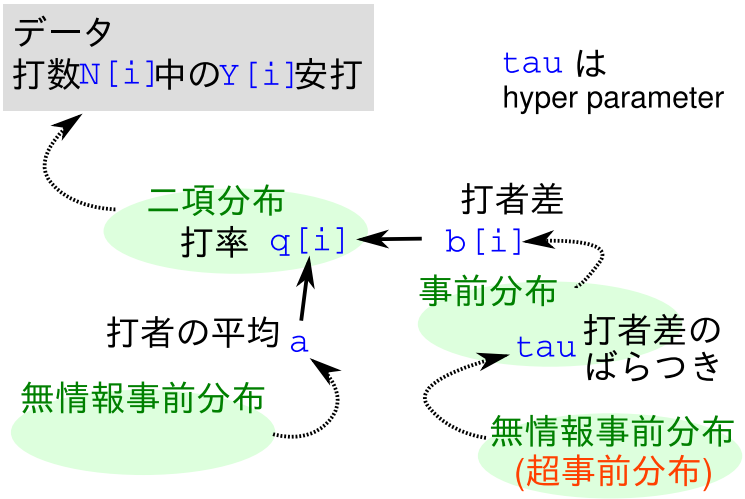
```
Y[i] ~ dbin(p[i], N[i]) # 二項分布
logit(p[i]) <- a + b[i] # 打率
```

確率 p[i] が a と b[i] が決まっているときに，といった意味になります．

またパラメーター a と b[i] については

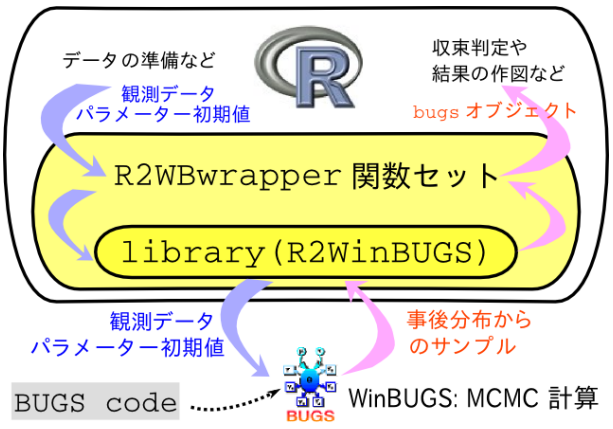
```
a ~ dnorm(0, Tau.noninformative) # 無情報事前分布
for (i in 1:N) {
    b[i] ~ dnorm(0, tau) # 打者差
}
```

ここでは右辺で定義されている正規分布 (dnorm(...)) から a と b[i] の新しい値をランダムサンプリングしなさい，ただし上で定義している「安打数データ (Y[i]) へのあてはまりがあまり悪くならないように」といったような制約がつけられています．



さて，この階層ベイズモデル定義ファイルだけでは WinBUGS は動いてくれません．観測データやパラメーターの初期値，いったい何回ぐらいサンプリングするのか，といった指示も必要です．これを WinBUGS 上でいちいち指示するのはたいへんなので，R の中でそういったことを準備して「WinBUGS を R の下うけ」として働かせるのがうまいやりかただろうと思います．それを可能にするのが R の R2WinBUGS package です．

ただし R2WinBUGS もあまりデキがよろしくないので，私は R2WinBUGS をもっと快適につかう関数セットというのを自分で作ってしまいました．それを R2WBwrapper.R というファイルにまとめているので，[9]



9. 例によって講義ページからダウンロードできます．R2WinBUGS package だけでなく coda もインストールする必要があります．

R2WBwrapper.R を読みこんだうえで上の BUGS コードで定義した階層ベイズモデル[10] を WinBUGS に下うけ計算させる R コードはこのようになります.

```
# 関数定義とデータの読みこみ
source("R2WBwrapper.R")
load("d.RData") # データの読みこみ

clear.data.param()
# 観測データを設定する
set.data("N.sample", nrow(d)) # 標本数 20
set.data("N", d$N) # 打数
set.data("Y", d$Y) # 安打数

# パラメーターの設定
set.param("a", 0) # 平均打率
set.param("b", rnorm(N, 0, 0.1)) # 打者差
set.param("tau", 1, save = FALSE) # 分散の逆数
set.param("sigma", NA) # sqrt(tau の逆数)
set.param("q", NA) # 打者ごとの打率 (出力用)

# WinBUGS の実行
post.bugs <- call.bugs(n.iter = 2000, n.burnin = 1000, n.thin = 5)
```

10. これは model.bug.txt というテキストファイルとして保存されているものとします

最後の WinBUGS の実行 (そして結果を post.bugs に格納) するところでは計算 step 数を指定しています:

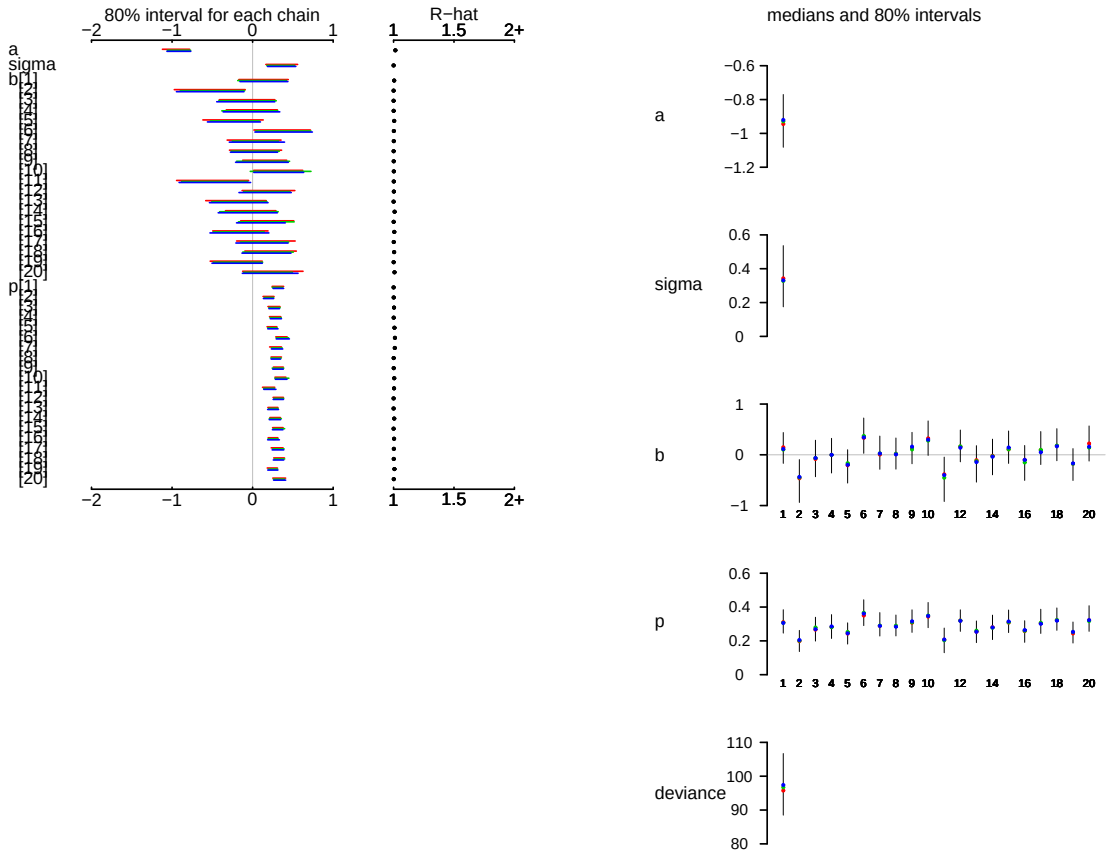
- n.iter = 1000: 全体で 2000 MCMC step 計算せよ
- n.burnin = 100: ただし最初の 1000 step は焼き捨て (burn-in) せよ
- n.thin = 5: 1001-2000 step の 1000 step に関しては，5 step とばし，つまり合計 200 sample を取れ
- (上では明示的に指定してませんが引数 n.chains の default の指定値として) このようなサンプリングを独立に 3 回試行せよ

あとは R を起動して source("runbugs.R") とすれば R から WinBUGS が呼びだされて下うけ計算をやってくれます.

8. 推定結果の事後分布を吟味する

WinBUGS の Gibbs sampling による MCMC 法で得られた推定計算の結果は post.bugs という bugs クラスのデータオブジェクトに格納されています．結果の概要は plot(post.bugs) で見ることができます.

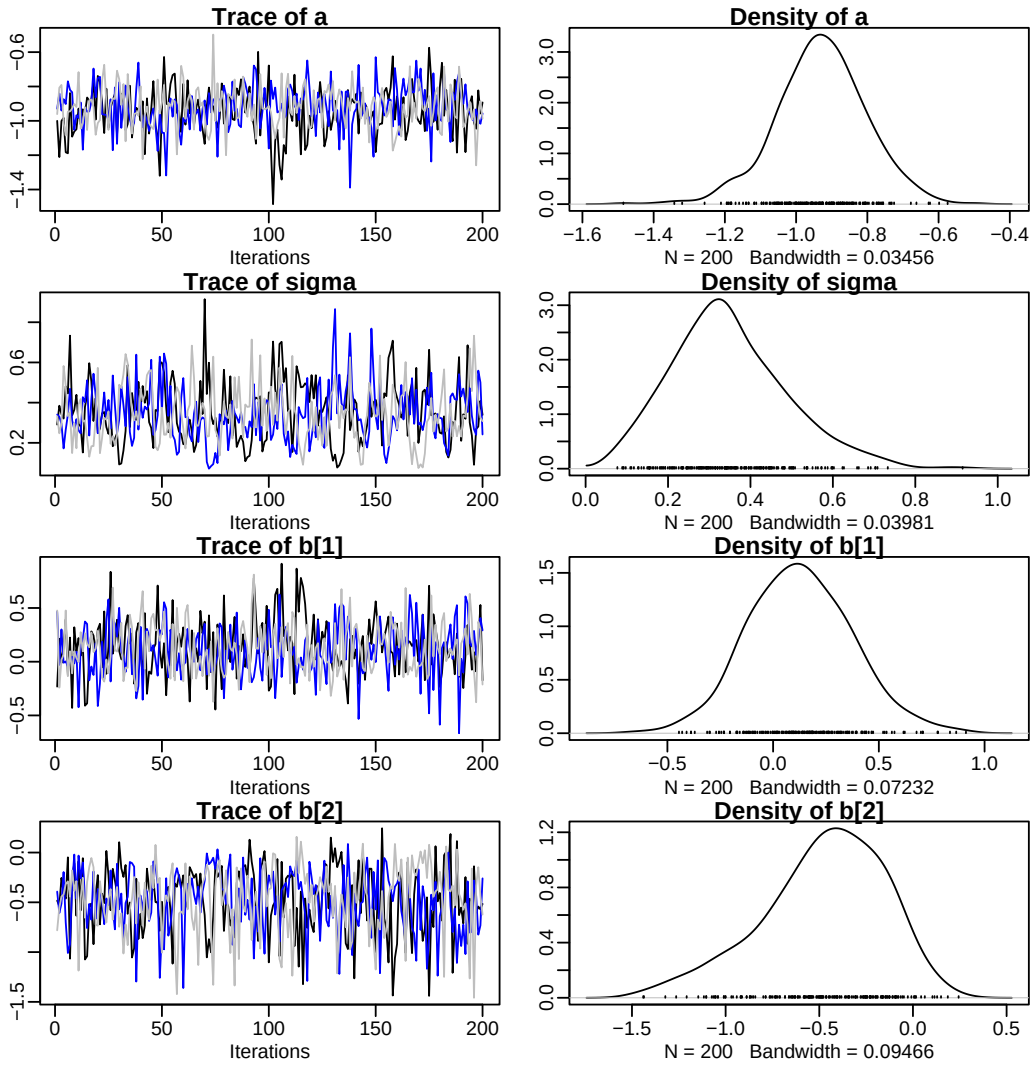
上の図についてだいたいのところを説明してみますと……



- 図の左側: MCMC 計算の収束ぐあいを示している．パラメーターごとに Gelman-Rubin の $\hat{R}$ 値が示されていて，これが 1 に近いほど収束がよい，つまり MCMC 計算がうまくいってることを示している
- 図の右側: パラメーターの事後分布の平均値 (点) と 80% 信頼区間 (縦のバー)
- 左右どちらの図でもパラメーターが多すぎて全部を示せない場合は途中でうちきっている (* マークがついている)

またひとつひとつのパラメーターが MCMC 法によってサンプリングされている様子や事後分布のカタチなんかも R に作図させることができます．

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



上の図の左側は MCMC step 数とともに変化しているパラメーターごとのサンプリングの様子，右側は事後分布の近似的な確率密度分布を示しています．

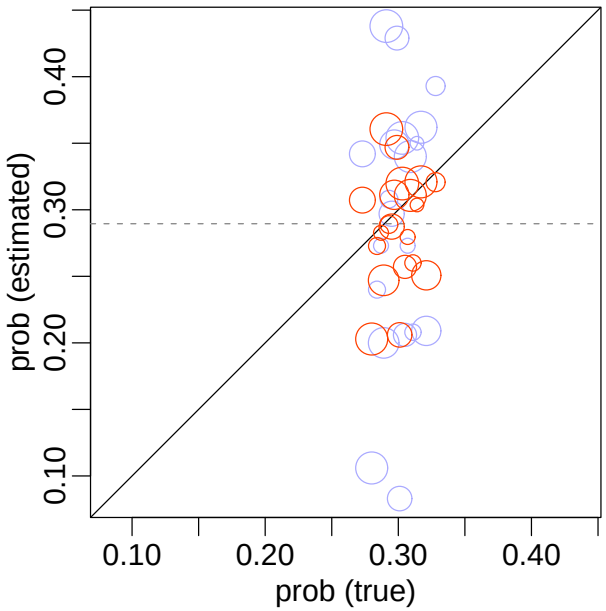
パラメーターの事後分布に関する table はこのように出力できます．

```
> print(post.bugs, digits.summary = 3)
... (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
... (略) ...
b[20]     0.199  0.281 -0.283  0.013  0.160  0.393   0.747 1.007   300
tau      16.044 23.486  2.275  5.385  9.000 16.004 101.498 1.000   600
```

(次のページにつづく)

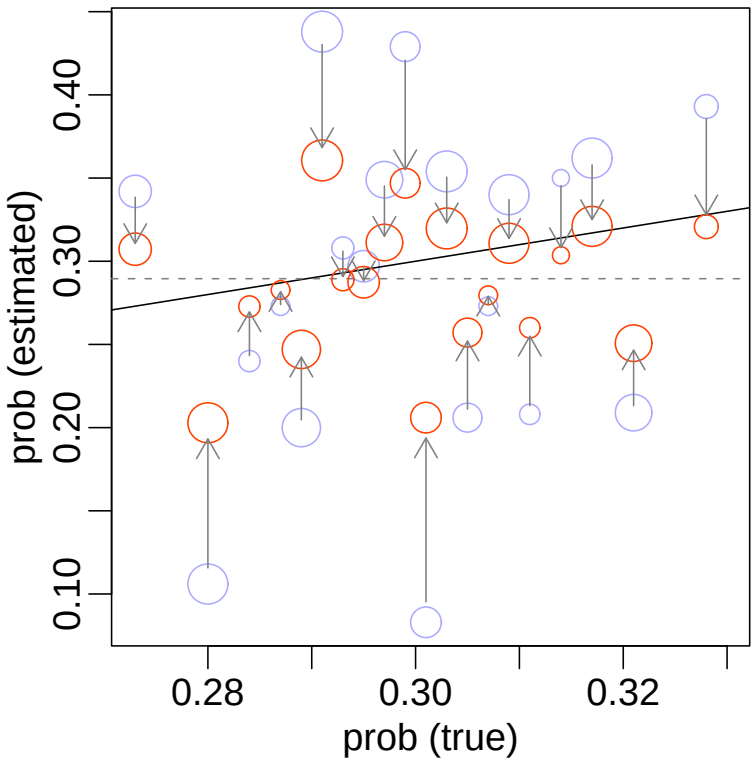
```
          mean     sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
p[1]     0.312  0.054  0.213  0.275  0.307  0.346  0.432 1.000   600
p[2]     0.201  0.049  0.108  0.167  0.203  0.238  0.300 1.001   600
p[3]     0.272  0.056  0.164  0.237  0.273  0.306  0.392 1.004   450
p[4]     0.284  0.056  0.181  0.247  0.283  0.319  0.397 1.001   600
p[5]     0.247  0.048  0.152  0.215  0.247  0.277  0.346 1.007   240
p[6]     0.365  0.060  0.261  0.323  0.361  0.404  0.491 1.011   160
p[7]     0.293  0.055  0.188  0.257  0.289  0.326  0.417 1.010   300
p[8]     0.289  0.050  0.198  0.253  0.287  0.323  0.390 1.001   600
p[9]     0.315  0.053  0.221  0.279  0.311  0.349  0.432 1.001   600
p[10]    0.351  0.059  0.249  0.310  0.347  0.384  0.486 1.005   600
p[11]    0.206  0.056  0.099  0.168  0.206  0.246  0.308 1.004   480
p[12]    0.320  0.050  0.235  0.283  0.320  0.352  0.423 0.999   600
p[13]    0.255  0.050  0.150  0.221  0.257  0.290  0.346 1.000   600
p[14]    0.279  0.057  0.165  0.243  0.280  0.314  0.393 1.005   600
p[15]    0.314  0.051  0.228  0.278  0.311  0.346  0.418 1.000   600
p[16]    0.259  0.052  0.165  0.224  0.260  0.292  0.358 1.004   590
p[17]    0.309  0.058  0.205  0.270  0.304  0.341  0.440 1.002   500
p[18]    0.325  0.053  0.234  0.289  0.321  0.357  0.440 1.002   600
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
... (略) ...
```

ここから値をとりだせば，この階層ベイズモデルの予測が観測データにあてはまってるかどうか，といったことも図示できます．



白丸がベイズ推定 (事後分布の中央値) の結果であり，灰色の円盤が安打数 / 打数の割算推定値です．ホントの打率と推定値が一致するとナナメの直線上にのります．

これではわかりにくいので，横軸を拡大，割算推定値 → ベイズ推定値というように矢印をつけてみましょう．



図に示されているように，割算推定値の極端な「打率」はベイズ推定によって，多少はまともな方向に「補正」されていることがわかります．

割算推定値は打率を打者ごとに独立に推定していたのに対して，ベイズ推定では「それぞれ打率は何か共通の事前分布にしたがう」といった仮定をもうけ，それがあたっていた場合にはこのように改善された推定結果が得られます．

「データの背後にある構造」をくみこんだ統計モデル，今回の例題でいうと「各チームの 3, 4, 5 番あたりの打者の打率はある程度『似ている』だろう」といった仮定を表現する事前分布を使った階層ベイズモデルを使うと推定結果が改善されることがよくあります．私たちが観察・比較したい現象は何らかの意味で似ているからです．

今回の例題であつかった階層ベイズモデルは比較的簡単なものでした．実際のデータ解析であつかう階層ベイズはより複雑なものであり，事後分布の推定には Gibbs sampling (MCMC 計算) してくれるソフトウェアが必要不可欠になります．

今回の例題はたしかに単純すぎるものですが,「階層ベイズモデルとは何か?」「Gibbs sampling ソフトウェアの挙動はどうなっているのか?」を理解するた

めにはちょうど良いものだと思います．まずはここから始めてみましょう!