

統計モデリング入門 BSJ2015 (4)

MCMC と階層ベイズモデル

久保拓弥 kubo@ees.hokudai.ac.jp, @KuboBook

日本行動計量学会 春セミナー <http://goo.gl/vIdtcv>

2015-03-07

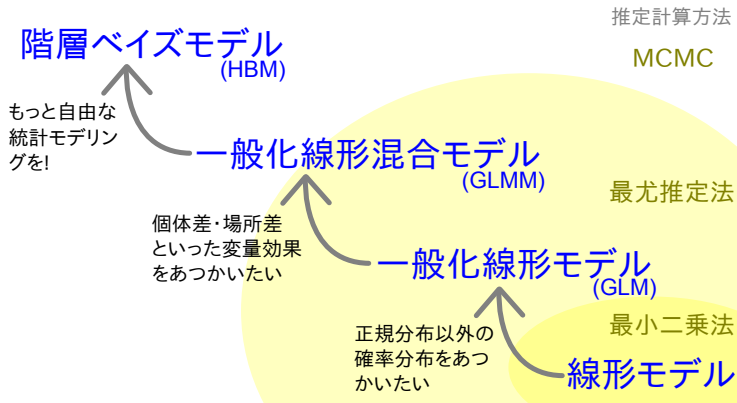
ファイル更新時刻: 2015-03-24 18:05

この時間に説明したいこと

- ① MCMC サンプリングのための例題
二項分布のパラメーターを最尤推定 (同じもの再掲)
- ② 同じような推定を MCMC でやってみる
最尤推定と MCMC はちがう!
- ③ MCMC のためのソフトウェア
“Gibbs sampling” などが簡単にできるような.....
- ④ GLMM と階層ベイズモデル
GLMM のベイズモデル化
- ⑤ 階層ベイズモデルの推定
ソフトウェア JAGS を試してみる
- ⑥ 複数ランダム効果の階層ベイズモデル
個体差 + グループ差, など

“統計モデリング入門” に登場する統計モデル

線形モデルの発展



データの特徴にあわせて線形モデルを改良・発展させる

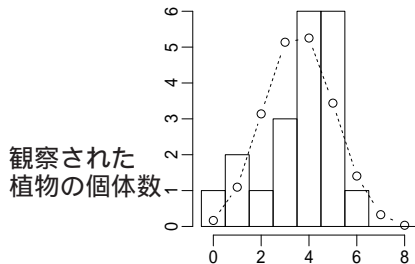
1. MCMC サンプリングのための例題

二項分布のパラメーターを最尤推定 (同じもの再掲)

前の時間の最初のハナシを少しくりかえます

簡単すぎる例題: 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	1	2	1	3	6	6	1	0	0



生存していた種子数 y_i

これは個体差なしの均質な集団

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

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

- 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**

最尤推定 (MLE) とは何か

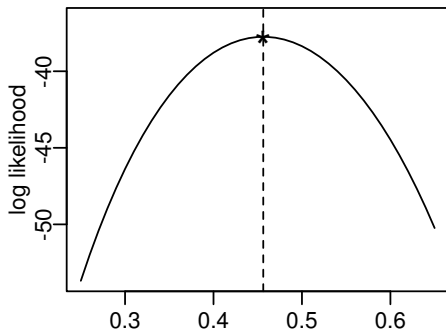
- 対数尤度 $L(q \mid \text{データ})$ が最大になるパラメータ q の値をさがすこと

- 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

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

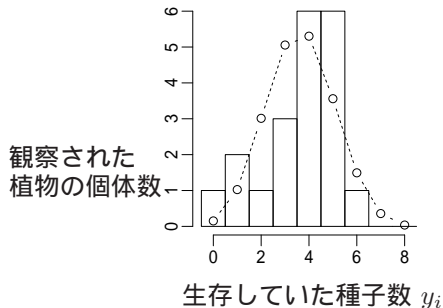
- 生存確率 q が全個体共通の場合の最尤推定量・最尤推定値は

$$\hat{q} = \frac{\text{生存種子数}}{\text{調査種子数}} = \frac{73}{160} = 0.456 \text{ くらい}$$



二項分布で説明できる 8 種子中 y_i 個の生存

$$\hat{q} = 0.46 \text{ なので } \binom{8}{y} 0.46^y 0.54^{8-y}$$



とりあえずここまでで
確率分布を適切に選んで統計モデリング,
統計モデルのパラメーターを
最尤 (さいゆう) 推定 する
..... といったことを説明しました

2. 同じような推定を MCMC でやってみる

最尤推定と MCMC はちがう!

そして “なんとなく” ベイズ統計モデルと関連づけ

ここでやること: 尤度と MCMC の関係を考える

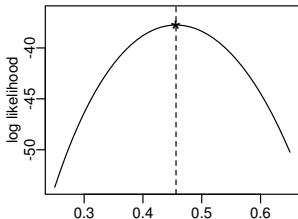
- さきほどの簡単な例題 (生存確率) のデータ解析を
- 最尤推定ではなく
- Markov chain Monte Carlo (MCMC) 法のひとつである**メトロポリス法** (Metropolis method) であつかう
- 得られる結果: 「パラメーターの値の分布」.....??

MCMC をもちださなくてもいい簡単すぎる問題
説明のためあえてメトロポリス法を適用してみる

メトロポリス法を説明するための準備

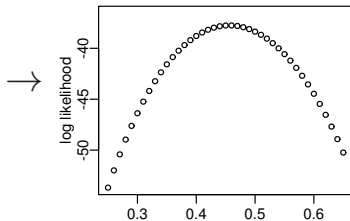
連続的な対数尤度関数

$\log L(q)$



離散化: q がとびとびの値を

とる



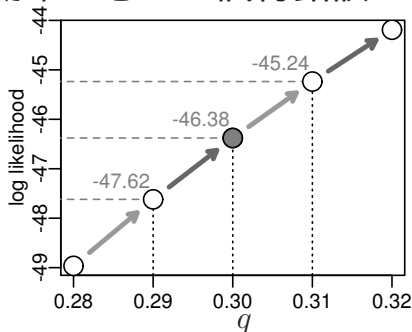
説明を簡単にするため

生存確率 q の軸を離散化する

(実際には離散化する必要などない)

試行錯誤による q の最尤推定値の探索

ちょっと効率の悪い「試行錯誤の最尤推定」



- ① q の値の「行き先」を「両隣」どちらかにランダムに決める
- ② 「行き先」が現在の尤度より高ければ、 q の値をそちらに変更
- ③ 尤度が変化しなくなるまで (1), (2) をくりかえす

メトロポリス法のルール: この例題の場合

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 0.3)

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_{new} としましょう)

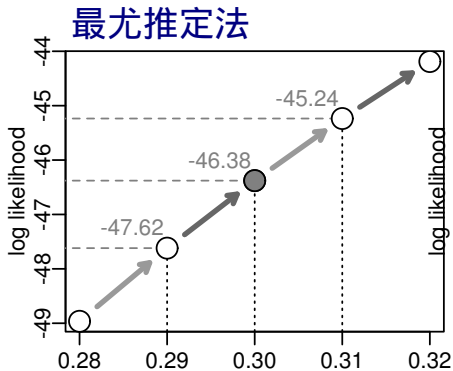
③ q_{new} における尤度 $L(q_{\text{new}})$ ともとの尤度 $L(q)$ を比較

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

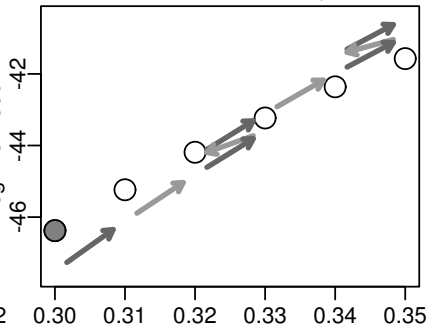
④ 手順 2. にもどる

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

メトロポリス法のルールで q を動かす



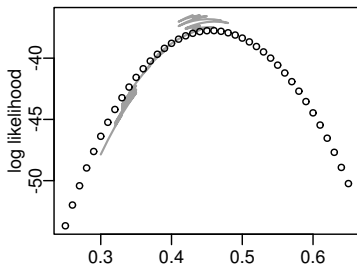
メトロポリス法 (MCMC)



メトロポリス法だと
「単調な山のぼり」にはならない

対数尤度関数の「山」でうろろうろする q の値

メトロポリス法 (そして一般の MCMC) は
最適化ではない

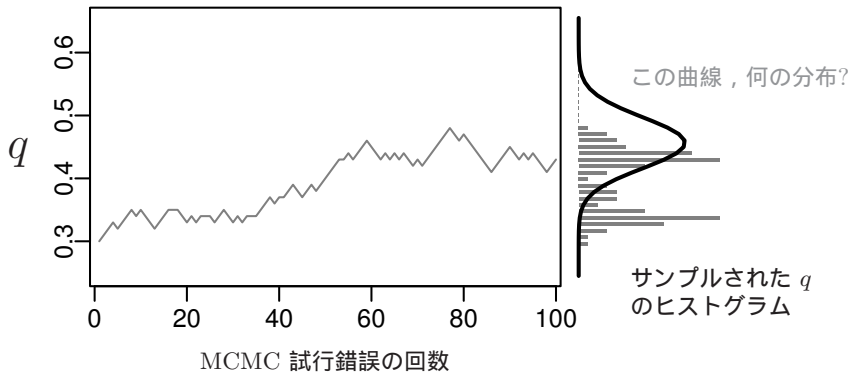


ときどきはでに落っこちる

何のためにこんなことをやるのか?

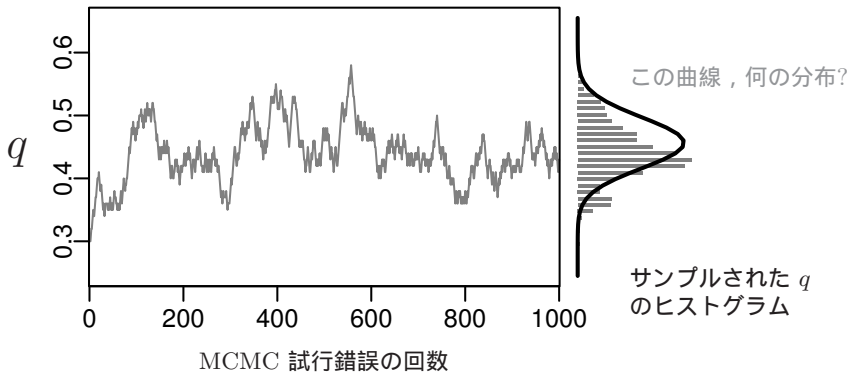
q の変化していく様子を記録してみよう

ステップごとに q の値をサンプリング



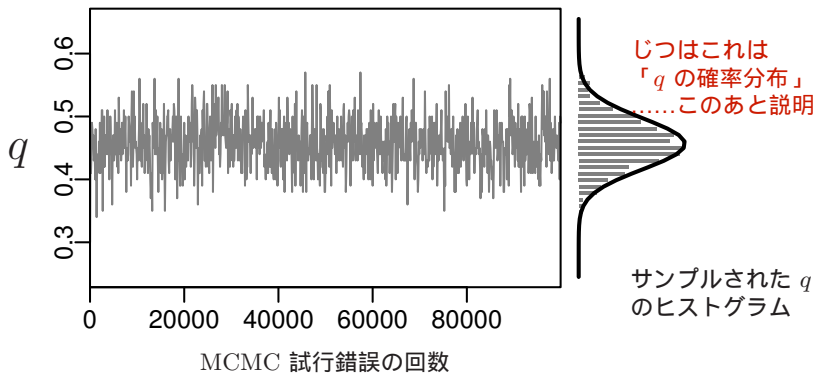
もっと試行錯誤してみたほうがいいのか?

もっと長くサンプリングしてみる



まだまだ.....?

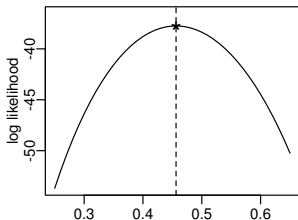
もっともっと長くサンプリングしてみる



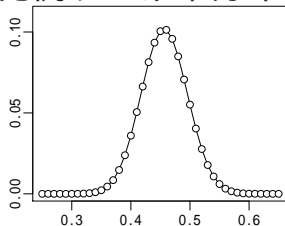
なんだか、ある「山」のかたちにとまとまったぞ?

MCMC は何をサンプリングしている?

対数尤度 $\log L(q)$

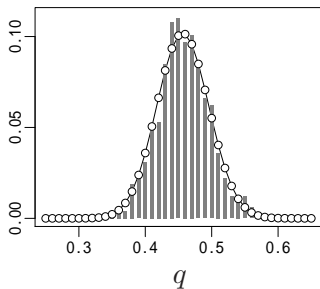


尤度 $L(q)$ に
比例する確率分布



尤度に比例する確率分布からのランダムサンプル

マルコフ連鎖の定常分布は $p(q) = \frac{L(q)}{\sum_q L(q)}$ となる

MCMC の結果として得られた q の経験分布

- データと統計モデル (二項分布) を決めて, MCMC サンプルングすると, $p(q)$ からのランダムサンプルが得られる
- このランダムサンプルをもとに, q の平均や 95% 区間などがわかる — **便利じゃないか!**

MCMC という推定方法から
「パラメーター q の確率分布」
というちょっと奇妙な考えかたが
でてきた

「ふつう」の統計学では
「パラメーターの確率分布」といった
考えかたはしない, しかし

ベイズ統計学なら
「パラメーターの確率分布」はぜんぜん
自然な考えかただ

ベイズモデル: 尤度・事後分布・事前分布.....

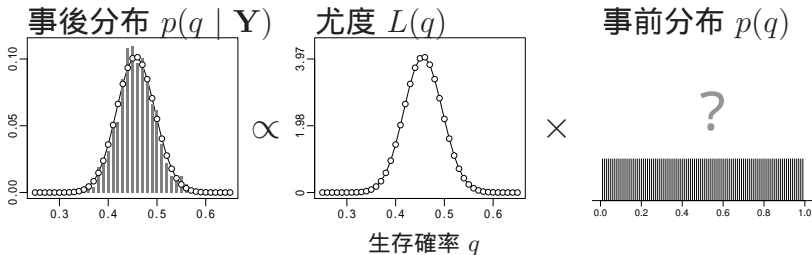
- ベイズの公式
$$p(q | \mathbf{Y}) = \frac{p(\mathbf{Y} | q) \times p(q)}{p(\mathbf{Y})}$$
- $p(q | \mathbf{Y})$ は何かデータ (\mathbf{Y}) のもとで何かパラメーター (q) が得られる確率 (事後分布)
- $p(q)$ はあるパラメーター q が得られる確率 (事前分布)
- $p(\mathbf{Y} | q)$ パラメーターを決めたときにデータが得られる確率 (尤度に比例)
- $p(\mathbf{Y})$ はデータ \mathbf{Y} が得られる確率 (単なる規格化定数)

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

$$\propto \text{尤度} \times \text{事前分布}$$

ベイズ統計にむりやりこじつけてみると?

q の事前分布は一様分布, と考えるとつじつまがあう?



事前分布ってのがよくわからない.....

以上の説明は、
「MCMC によって得られる結果」
は
「ベイズ統計でいうパラメーターの事後分布」
と考えると解釈しやすいかも
といったことを
ばくぜんかつなんとなく対応づける
ひとつのこころみでありました……

厳密な正当化とかそういったものではありません

3. MCMC のためのソフトウェア

“Gibbs sampling” などが簡単にできるような.....

事後分布から効率よくサンプリングしたい

統計ソフトウェア R

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



簡単な GLMM なら R だけで推定可能

今回の例題の事後分布 ($\mathbf{Y} = \{y_i\}$ はデータ)

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

積分で「個体差」 r_i を消して，周辺尤度を定義する

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

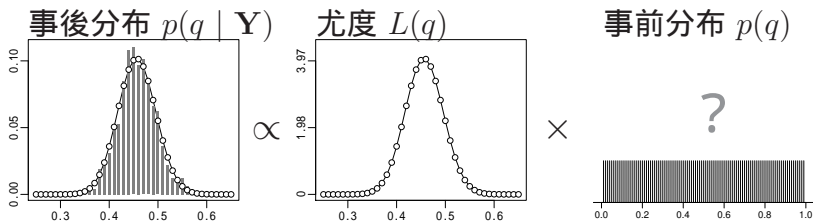
これを最大化する \hat{a} と \hat{s} を推定すればよい

— 経験ベイズ法 (empirical Bayesian method)

しかし, 「R だけ」では限界があるかも

- R にはいろいろな GLMM の最尤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)
- しかし もうちょっと複雑な GLMM, たとえば個体差 + 地域差をいれた統計モデルの最尤推定はかなり難しい (ヘンな結果が得られたりする)
- 積分がたくさん入っている尤度関数の評価がしんどい

そこで MCMC による事後分布からのサンプリング!



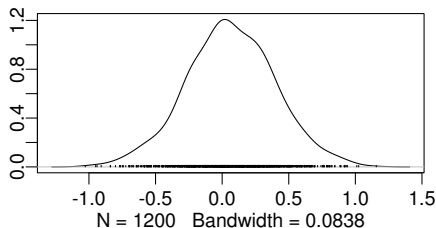
生存確率 q

アルゴリズムにしたがって
乱数を発生させていくだけで OK

再確認: 「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 “BUGS” で “Gibbs sampler” なソフトウェア

- 利点: 幅ひろい問題に適用できて, 便利
- 欠点というほどでもないけど, 多少の勉強が必要
- えーっと “Gibbs sampler” って何?

さまざまな MCMC アルゴリズム

いろいろな MCMC

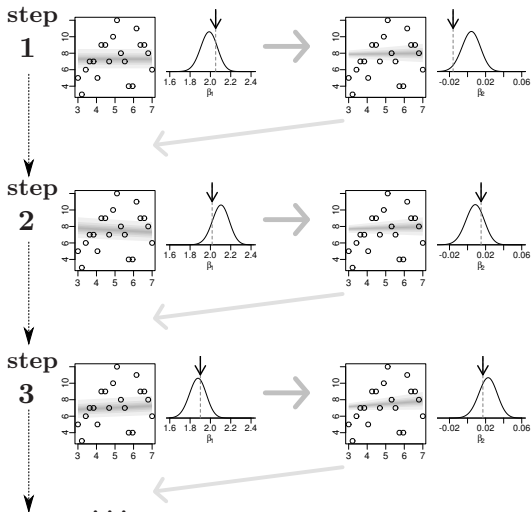
- **メトロポリス法**: 試行錯誤で値を変化させていく MCMC
 - メトロポリス・ヘイスティングス法: その改良版
- **ギブス・サンプリング**: 条件つき確率分布を使った MCMC
 - 複数の変数 (パラメーター・状態) を効率よくサンプリング

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 (条件つき事後分布)
 - 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 (条件つき事後分布)
 - 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)

MCMC β_1 のサンプリング β_2 のサンプリング



便利な “BUGS” 汎用 Gibbs sampler たち

- BUGS 言語 (+ っぽいもの) でベイズモデルを記述できるソフトウェア
 - WinBUGS — **ありがとう**, さようなら?
 - OpenBUGS — 予算が足りなくて停滞?
 - **JAGS** — お手軽で良い, どんな OS でも動く
 - Stan — たぶん “次” はこれ
— 今日は紹介しませんが
- リンク集: <http://hosho.ees.hokudai.ac.jp/~kubo/ce/BayesianMcmc.html>

えーと.....BUGS 言語って何?

このベイズモデルを BUGS 言語で記述したい

データ $Y[i]$
種子数8個のうちの生存数

二項分布

$\text{dbin}(q, 8)$

生存確率 q

無情報事前分布

BUGS 言語コード

```
for (i in 1:N.sample) {
  Y[i] ~ dbin(q, 8)
}
q ~ dunif(0.0, 1.0)
```

矢印は手順ではなく，依存関係をあらわしている

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.

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 → 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
- 歴史を変えたソフトウェアだけど, 開発も停止していることだし, まあ, もう “ごくろうさま” ということで.....

いろいろな OS で使える JAGS3.4.0

- 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
 - R がインストールされていることが必要
- Linux, Windows, Mac OS X バイナリ版もある
- ざりざりと開発進行中
- R からの使う: `library(rjags)`

R から JAGS にこんなかんじで仕事を命じる (1 / 3)

```
library(rjags)
library(R2WinBUGS) # to use write.model()

model.bugs <- function()
{
  for (i in 1:N.data) {
    Y[i] ~ dbin(q, 8) # 二項分布にしたがう
  }
  q ~ dunif(0.0, 1.0) # q の事前分布は一様分布
}

file.model <- "model.bug.txt"
write.model(model.bugs, file.model) # ファイル出力

# 次につづく.....
```

R から JAGS にこんなかんじで仕事を命じる (2 / 3)

```
load("data.RData")
list.data <- list(Y = data, N.data = length(data))
inits <- list(q = 0.5)
n.burnin <- 1000
n.chain <- 3
n.thin <- 1
n.iter <- n.thin * 1000

model <- jags.model(
  file = file.model, data = list.data,
  inits = inits, n.chain = n.chain
)

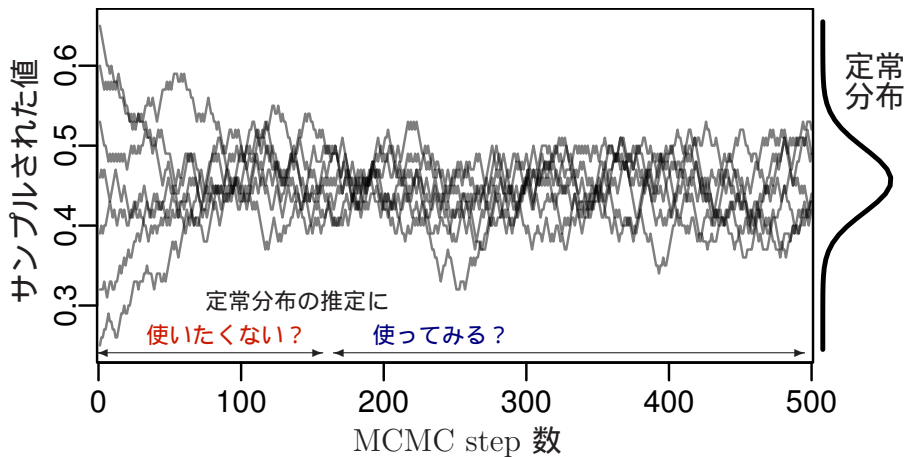
# まだ次につづく.....
```

R から JAGS にこんなかんじで仕事を命じる (3 / 3)

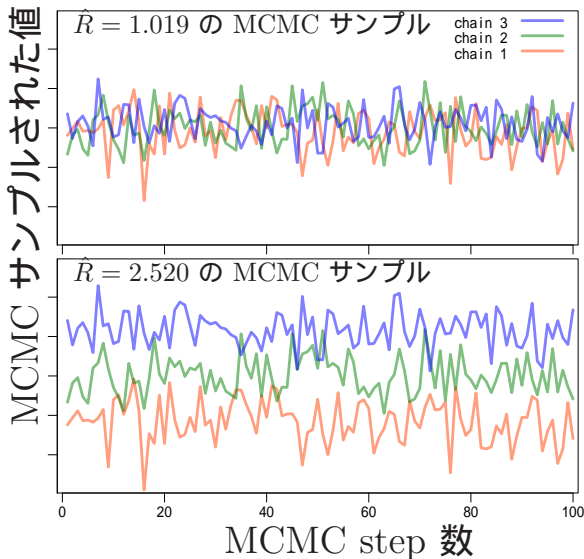
```
# burn-in
update(model, n.burnin) # burn in

# サンプルング結果を post.mcmc.list に格納
post.mcmc.list <- coda.samples(
  model = model,
  variable.names = names(inits),
  n.iter = n.iter,
  thin = n.thin
)
# おわり
```

burn in って何? → 「使いたくない」長さの指定



試行間で差がないかを「診断」する

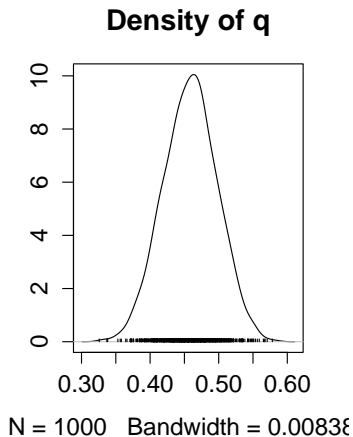
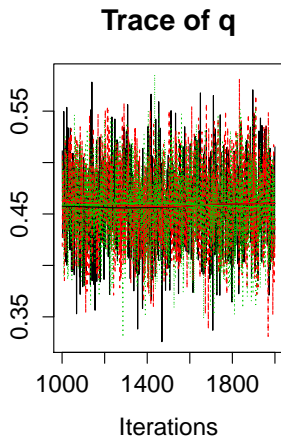
まあ、
いいかな.....何やら
問題あり!

収束診断の \hat{R} 指数

- `gelman.diag(post.mcmc.list)` → 実演表示
- R-hat は Gelman-Rubin の収束判定用の指数
 - $\hat{R} = \sqrt{\frac{\hat{\text{var}}^+(\psi|y)}{W}}$
 - $\hat{\text{var}}^+(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$
 - W : サンプル列内の variance の平均
 - B : サンプル列間の variance
 - Gelman et al. 2004. Bayesian Data Analysis. Chapman & Hall/CRC

Gibbs sampling → 事後分布の推定

- `plot(post.mcmc.list)`



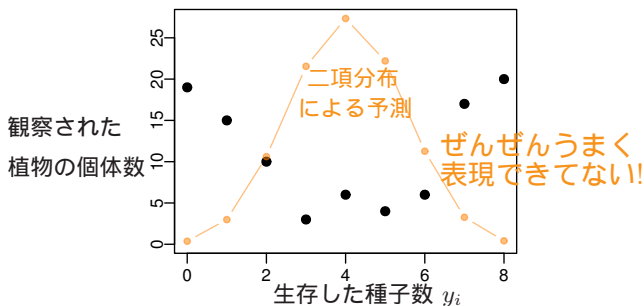
4. GLMM と階層ベイズモデル

GLMM のベイズモデル化

階層ベイズモデルとなる

二項分布では説明できない観測データ!

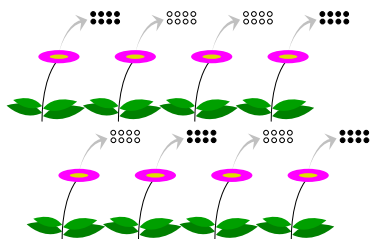
100 個体の植物の合計 800 種子中 **403 個** の生存が見られたので, 平均生存確率は 0.50 と推定されたが.....



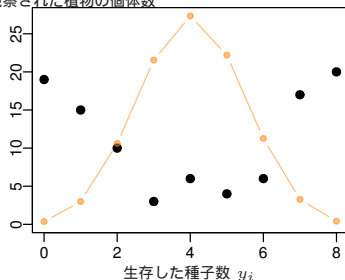
さっきの例題と同じようなデータなのに?
 (「統計モデリング入門」第 10 章の最初の例題)

個体差 → 過分散 (overdispersion)

極端な過分散の例



観察された植物の個体数



- 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」の原因は観測できない・観測されていない

モデリングやりなおし: 個体差を考慮する

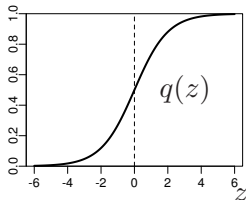
- 生存確率を推定するために **二項分布** という確率分布を使う
- 個体 i の N_i 種子中 y_i 個が生存する確率は二項分布

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

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

GLM わざ: ロジスティック関数で表現する生存確率

- 生存確率 $q_i = q(z_i)$ をロジスティック関数 $q(z) = 1/\{1 + \exp(-z)\}$ で表現



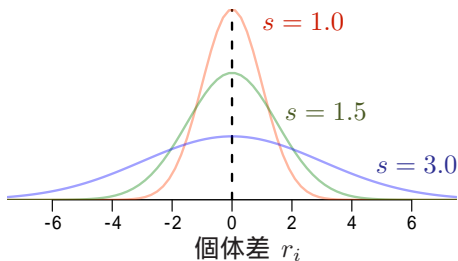
- 線形予測子 $z_i = a + r_i$ とする
 - パラメーター a : 全体の平均
 - パラメーター r_i : 個体 i の個体差 (ずれ)

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)

そこで、次のように考えてみる

$\{r_i\}$ のばらつきは正規分布だと考えてみる

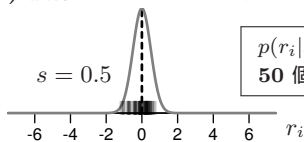


$$p(r_i | s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

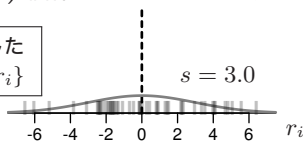
この確率密度 $p(r_i | s)$ は r_i の「出現しやすさ」をあらわしていると解釈すればよいでしょう。 r_i がゼロにちかい個体はわりと「ありがち」で、 r_i の絶対値が大きな個体は相対的に「あまりいない」。

ひとつの例示: 個体差 r_i の分布と過分散の関係

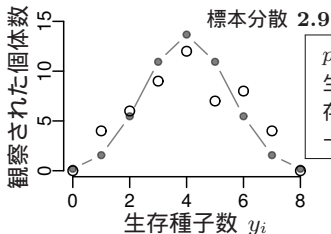
(A) 個体差のばらつきが小さい場合 (B) 個体差のばらつきが大きい場合



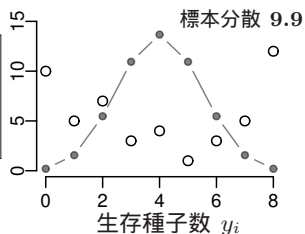
$p(r_i|s)$ が生成した
50 個体分の $\{r_i\}$



確率 $q_i = \frac{1}{1 + \exp(-r_i)}$
の二項乱数を発生させる

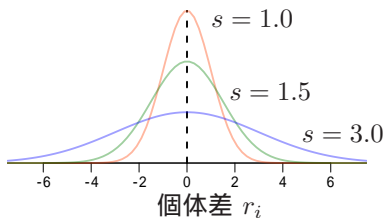


$p(y_i|q_i)$ が
生成した生
存種子数の
一例



これは r_i の事前分布の指定，ということ

前回の授業で $\{r_i\}$ は正規分布にしたがうと仮定したが
ベイズ統計モデリングでは「100 個の r_i たちに
共通する事前分布として正規分布を指定した」
ということになる



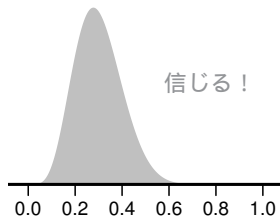
$$p(r_i | s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

ベイズ統計モデルでよく使われる三種類の事前分布

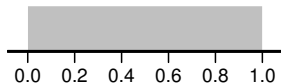
たいていのベイズ統計モデルでは、ひとつのモデルの中で複数の種類の事前分布を混ぜて使用する。

(A) 主観的な事前分布

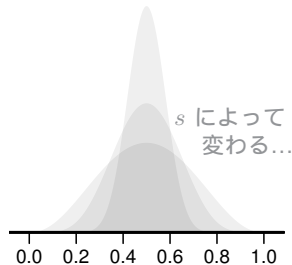
(できれば使いたくない!)



(B) 無情報事前分布



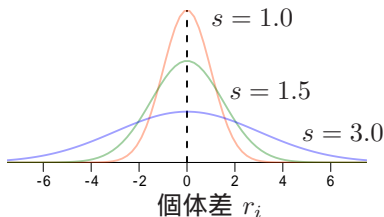
(C) 階層事前分布



r_i の事前分布として階層事前分布を指定する

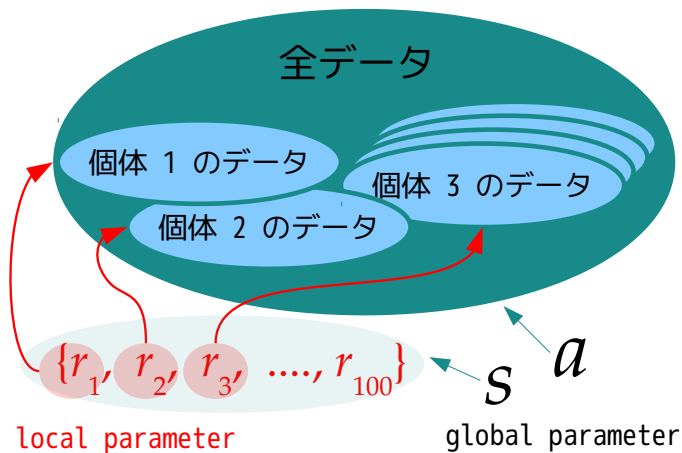
階層事前分布の利点

「データにあわせて」事前分布が変形!



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

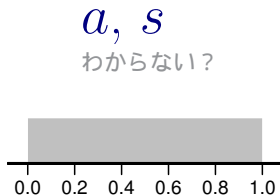
統計モデルの大域的・局所的なパラメーター



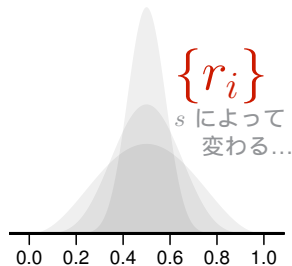
データのどの部分を説明しているのか?

パラメーターごとに適切な事前分布を選ぶ

(B) 無情報事前分布



(C) 階層事前分布



パラメーターの
種類

説明する範囲

事前分布

全体に共通する平均・ばらつき

大域的

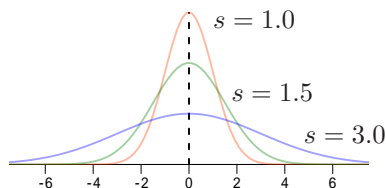
無情報事前分布

個体・グループごとのずれ

局所的

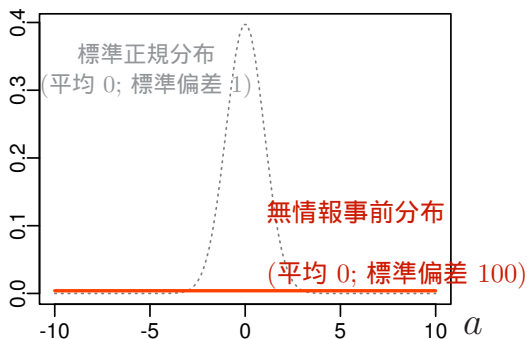
階層事前分布

個体差 $\{r_i\}$ のばらつき s の無情報事前分布



- s はどのような値をとってもかまわない
- そこで s 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, ここでは $0 < s < 10^4$ 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

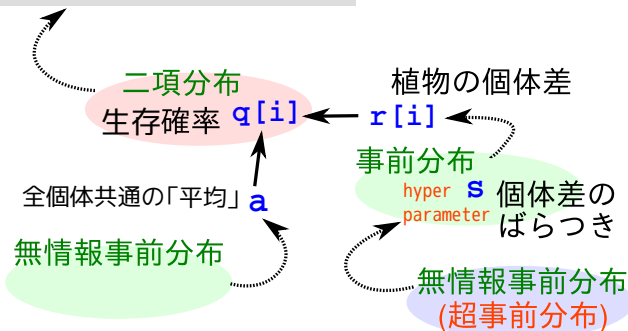


「生存確率の (logit) 平均 a は何でもよい」と表現している

階層ベイズモデル: 事前分布の階層性

超事前分布 → 事前分布という階層があるから

データ 種子8個のうち
 $y[i]$ が生存



矢印は手順ではなく、依存関係をあらわしている

5. 階層ベイズモデルの推定

ソフトウェア JAGS を試してみる

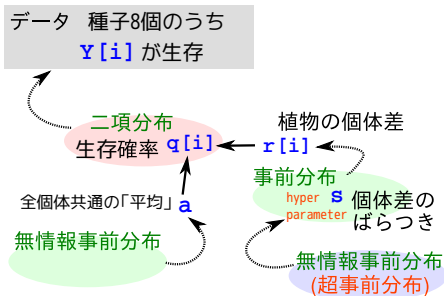
R の “したうけ” として JAGS を使う

階層ベイズモデルを BUGS コードで記述する

```

model
{
  for (i in 1:N.data) {
    Y[i] ~ dbin(q[i], 8)
    logit(q[i]) <- a + r[i]
  }
  a ~ dnorm(0, 1.0E-4)
  for (i in 1:N.data) {
    r[i] ~ dnorm(0, tau)
  }
  tau <- 1 / (s * s)
  s ~ dunif(0, 1.0E+4)
}

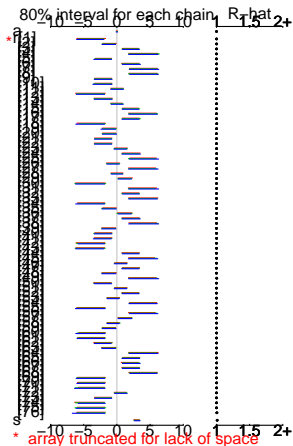
```



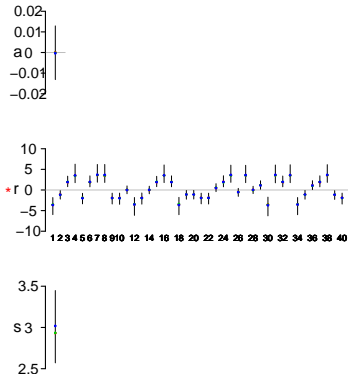
JAGS で得られた事後分布サンプルの要約

```
> source("mcmc.list2bugs.R") # なんとなく便利なので...
> post.bugs <- mcmc.list2bugs(post.mcmc.list) # bugs クラスに変換
```

3 chains, each with 4000 iterations (first 2000 discarded)



medians and 80% intervals



bugs オブジェクトの `post.bugs` を調べる

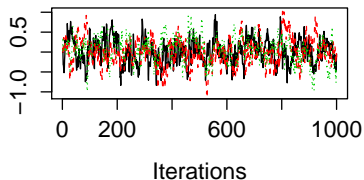
- `print(post.bugs, digits.summary = 3)`
- 事後分布の 95% 信頼区間などが表示される

```
3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
n.sims = 3000 iterations saved
```

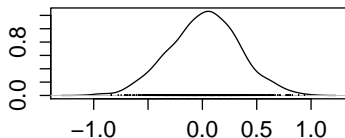
	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
a	0.020	0.321	-0.618	-0.190	0.028	0.236	0.651	1.007	380
s	3.015	0.359	2.406	2.757	2.990	3.235	3.749	1.002	1200
r[1]	-3.778	1.713	-7.619	-4.763	-3.524	-2.568	-1.062	1.001	3000
r[2]	-1.147	0.885	-2.997	-1.700	-1.118	-0.531	0.464	1.001	3000
r[3]	2.014	1.074	0.203	1.282	1.923	2.648	4.410	1.001	3000
r[4]	3.765	1.722	0.998	2.533	3.558	4.840	7.592	1.001	3000
r[5]	-2.108	1.111	-4.480	-2.775	-2.047	-1.342	-0.164	1.001	2300
...	(中略)								
r[99]	2.054	1.103	0.184	1.270	1.996	2.716	4.414	1.001	3000
r[100]	-3.828	1.766	-7.993	-4.829	-3.544	-2.588	-1.082	1.002	1100

各パラメーターの事後分布サンプルを R で調べる

Trace of a

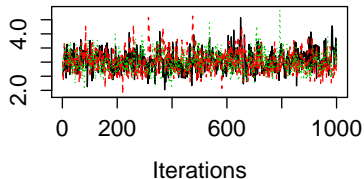


Density of a

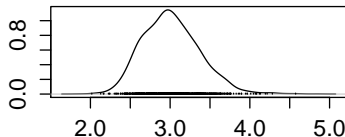


N = 1000 Bandwidth = 0.06795

Trace of s



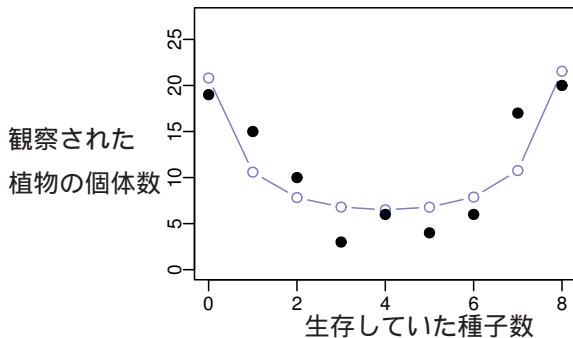
Density of s



N = 1000 Bandwidth = 0.07627

得られた事後分布サンプルを組みあわせて予測

- `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

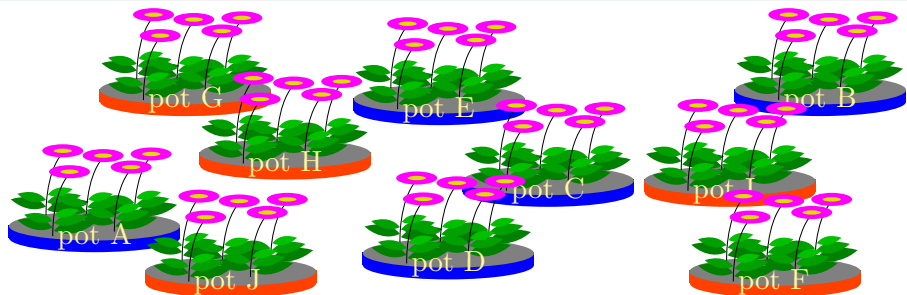


6. 複数ランダム効果の階層ベイズモデル

個体差 + グループ差, など

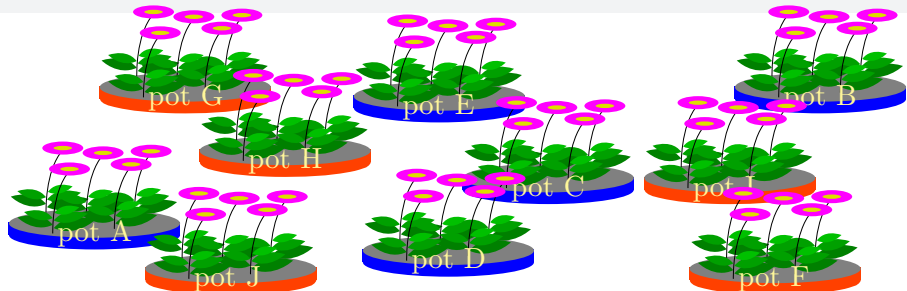
そして “てぬき” モデリングの危なさについて

架空植物の例題: またまた種子数データ



- 肥料をやったら個体ごとの種子数 y_i が増えるかどうかを調べたい
- 植木鉢 10 個, 各鉢に 10 個体の架空植物 (合計 100 個体)
 - コントロール ($f_j = \mathbf{C}$) 5 鉢 (合計 50 個体)
 - 肥料をやる処理 ($f_j = \mathbf{T}$) 5 鉢 (合計 50 個体)

このへんてこな例題, 言い換えると.....?



- 教育法をやったら児童ごとの算数の得点 y_i が増えるかどうかを調べたい
- 小学校 10 校, 各校に 10 人の児童に算数のテスト (合計 100 児童)
 - コントロール ($f_j = \mathbf{C}$) 5 校 (合計 50 児童)
 - 教育法実施 ($f_j = \mathbf{T}$) 5 校 (合計 50 児童)

データはこのように格納されている

```
> d <- read.csv("d1.csv")
```

```
> head(d)
```

	id	pot	f	y
1	1	A	C	6
2	2	A	C	3
3	3	A	C	19
4	4	A	C	5
5	5	A	C	0
6	6	A	C	19

- id 列: 個体番号

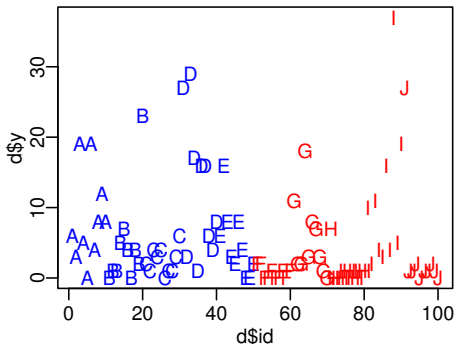
{1, 2, 3, ..., 100}

- pot 列: 植木鉢名 {A, B, C, ..., J}

- f 列: 処理: コントロール C, 肥料 T

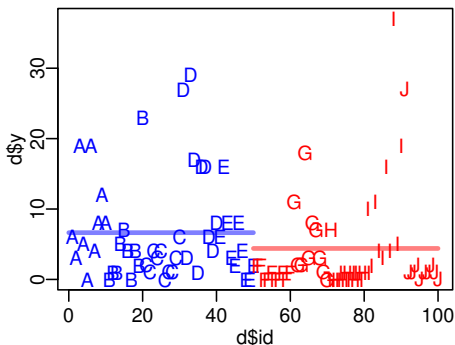
- y 列: 種子数 (応答変数)

データはとにかく図示する!!



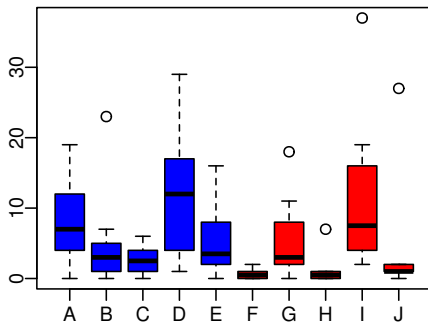
- `plot(did, dy, pch = as.character(d$pot), ...)`
- **コントロール**・**処理** でそんなに差がない?

処理ごとの平均も図に追加してみる



- むしろ **処理** のほうが平均種子数が低い?
- (注) この架空データは **肥料の効果はゼロ** と設定して生成した

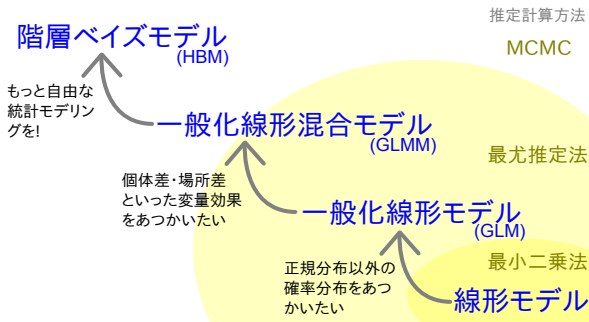
個体差だけでなく植木鉢差もありそう?



- `plot(dpot, dy, col = rep(c("blue", "red"), each = 5))`
- 植木鉢由来の random effects みたいなものは**ブロック差**と呼ばれる

(一般化な) 線形モデルのわくぐみで, とり あえず考えてみる

線形モデルの発展



GLM: 個体差もブロック差も無視

```
> summary(glm(y ~ f, data = d, family = poisson))
```

```
...(略)...
```

```
Coefficients:
```

```
Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept) 1.8931 0.0549 34.49 < 2e-16
```

```
fT -0.4115 0.0869 -4.73 2.2e-06
```

```
...(略)...
```

- 肥料をやる処理 (f) をすると, 平均種子数が下がる?
- AIC でモデル選択しても同じような結果に

GLMM: 個体差だけ考慮, ブロック差は無視

```
> library(glmML)
> summary(glmML(y ~ f, data = d, family = poisson,
+ cluster = id))
...(略)...
```

	coef	se(coef)	z	Pr(> z)
(Intercept)	1.351	0.192	7.05	1.8e-12
fT	-0.737	0.280	-2.63	8.4e-03

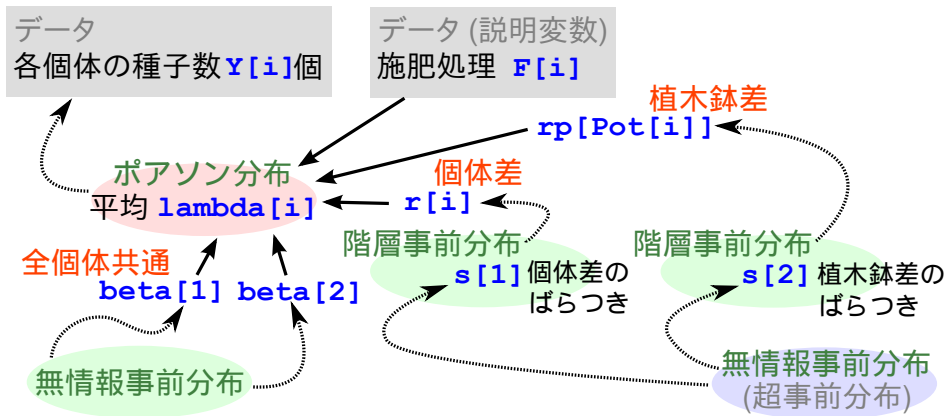
...(略)...

- やっぱり同じ?
- むしろ肥料処理の悪影響が強い?

個体差 + ブロック差を考える階層ベイズモデル

- ここでは \log リンク関数を使う
- 平均の対数 $\log(\lambda_i) = a + bf_i + (\text{個体差}) + (\text{ブロック差})$
- 事前分布の設定
 - 切片 a と f_i の係数 b は無情報事前分布 (すごく平らな正規分布)
 - 個体差とブロック差は階層的な事前分布 (それぞれ標準偏差 σ_1, σ_2 の正規分布, 平均はゼロ)
 - 標準偏差 σ_* は無情報事前分布 ($[0, 10^4]$ の一様分布)

植木鉢問題の階層ベイズモデルの図示



個体差 + ブロック差を考える階層ベイズモデル

- ここでは \log リンク関数を使う
- 平均の対数 $\log(\lambda_i) = a + bf_i + (\text{個体差}) + (\text{ブロック差})$
- 事前分布の設定
 - 切片 a と f_i の係数 b は無情報事前分布 (すごく平らな正規分布)
 - 個体差とブロック差は階層的な事前分布 (それぞれ標準偏差 σ_1, σ_2 の正規分布, 平均はゼロ)
 - 標準偏差 σ_* は無情報事前分布 ($[0, 10^4]$ の一様分布)

個体差 + ブロック差のあるポアソン回帰の BUGS code (1)

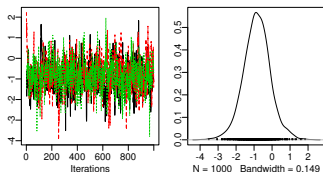
```
model
{
  for (i in 1:N.sample) {
    Y[i] ~ dpois(lambda[i])
    log(lambda[i]) <- a + b * F[i] + r[i] + rp[Pot[i]]
  }
# 次のページの事前分布の定義につづく
```

ここでの BUGS coding のポイント

- 因子型の説明変数 $f_i \in \{C, T\}$ は, それぞれ $F[i]$ を 0, 1 と置きかえる
- $Pot[i]$ は 1, 2, ..., 10 と数字になおした植木鉢名をいれておいて, 植木鉢の効果 $rp[...]$ を参照させる

個体差 + ブロック差のあるポアソン回帰の BUGS code (2)

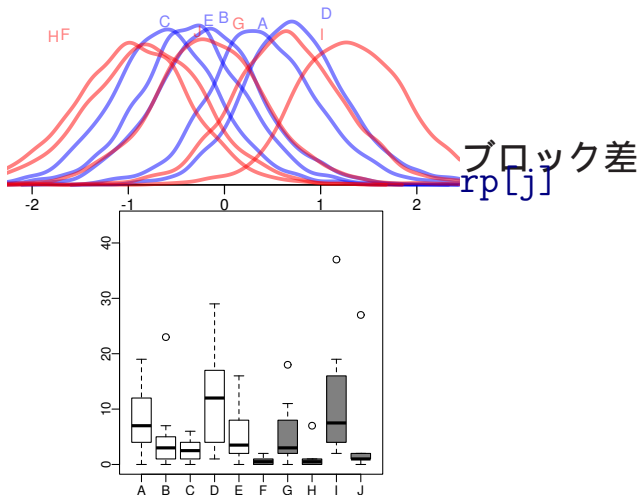
```
# 前のページからのつづき
a ~ dnorm(0, 1.0E-4) # 切片
b ~ dnorm(0, 1.0E-4) # 肥料の効果
for (i in 1:N.sample) {
  r[i] ~ dnorm(0, tau[1]) # 個体差
}
for (j in 1:N.pot) {
  rp[j] ~ dnorm(0, tau[2]) # 植木鉢の差 (ブロック差)
}
for (k in 1:N.tau) {
  tau[k] <- 1.0 / (sigma[k] * sigma[k]) # 個体・植木鉢のばらつき
  sigma[k] ~ dunif(0, 1.0E+4)
}
}
```

肥料の効果 (パラメーター b) はなさそう?

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat
a	1.501	0.529	0.482	1.157	1.493	1.852	2.565	1.00
b	-1.016	0.706	-2.436	-1.476	-0.993	-0.565	0.395	1.00
sigma[1]	1.020	0.114	0.822	0.939	1.014	1.089	1.265	1.00
...(略)...								

この架空データを生成した種子数シミュレーションでは、肥料の効果はまったく無いと設定していた

推定された植木鉢の差 (ブロック差)

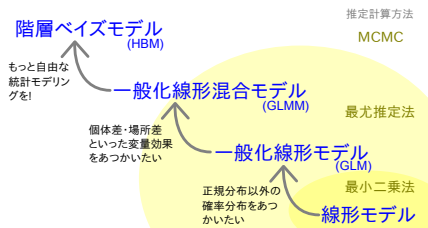


統計モデリングの手ぬきは危険!

- **random effects** つまり 個体差・ブロック差が大きい
- **random effects** の影響が大きいときには, **fixed effects** の大きさが見えにくくなる— ニセの「効果」が見えることもあれば, 見えるはずの傾向が隠されることも
 - 個体差・ブロック差の階層ベイズモデルが必要!
- もしブロック差を人為的に小さくできないなら, ブロック数をもっと増やして, より正確な**植木鉢の効果のばらつき**を正確に推定するしかない

階層ベイズモデルと GLMM の関係は?

線形モデルの発展



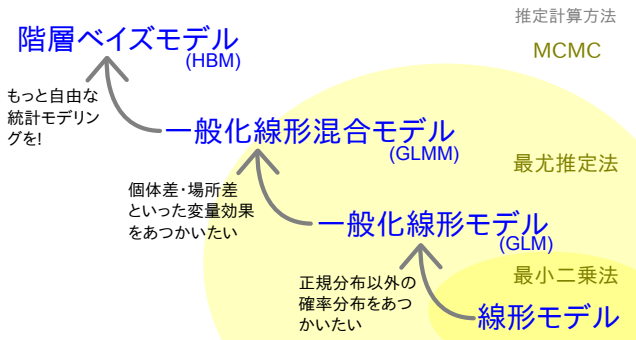
一般化線形混合モデル
(Generalized Linear Mixed Model) は階層ベイズモデルのひとつ

- GLMM では個体差・植木鉢差といった local parameter は積分して消去してしまう
- 階層ベイズモデルでは, 何もかも事後分布として推定してしまう

ここでこの **統計モデリング入門** は終了

- 一般化線形モデル → 階層ベイズモデル
- 最尤推定 → Markov chain Monte Carlo (MCMC)

線形モデルの発展



みなさん, おつかれさまでした!