統計モデリングの基礎 (2)

GLM の階層ベイズモデル化

久保拓弥 kubo@ees.hokudai.ac.jp

生態学基礎論

2018–01–24

ファイルのダウンロード: http://goo.gl/76c4i

ファイル更新時刻: 2018–01–17 13:09

statistaical models appeared in the class

“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

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

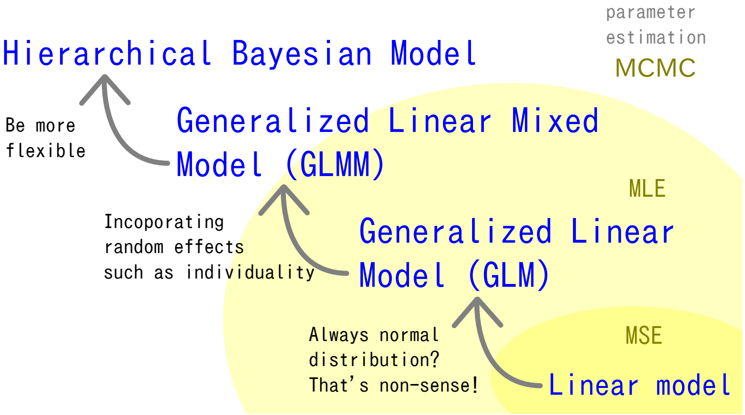
正規分布以外の確率分布をあつかいたい

線形モデル

データの特徴にあわせて線形モデルを改良・発展させる

この講義であつかう統計モデル

The development of linear models



“See the evolution of linear-model family!”

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**
   A simple example for applying binomial distribution
2. **同じ推定を MCMC でやってみる**
   "Sampling" using Monte Carlo Markov Chain method
3. **MCMC のためのソフトウェア**
   How to "sample" from posterior distribution
4. **GLM を階層ベイズモデル化する!**
   How to design hierarchical models
5. **階層ベイズモデルの推定**
   How to use JAGS sampler?
6. **複数のランダム効果をもつ階層ベイズモデル**
   individual effects + block effects

1. MCMC サンプリングのための例題

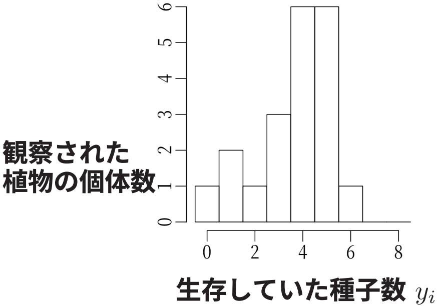
A simple example for applying binomial distribution

二項分布のパラメーターを最尤推定 (さっきと同じ例題)

簡単すぎる例題: 生存確率は全個体で同じ (「個体差」なし)

a simple example

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



これは個体差なしの均質な集団

生存確率 q と二項分布 (binomial distribution) 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**

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

- **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**

maximum likelihood estimation for binomaial distribution
二項分布を使った統計モデルの最尤推定 (MLE)

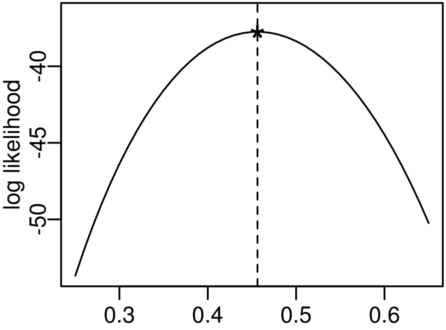
- **対数尤度** $L(q \mid \text{データ})$ **が最大になるパラメーター** q **の値をさがしだすこと**

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

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

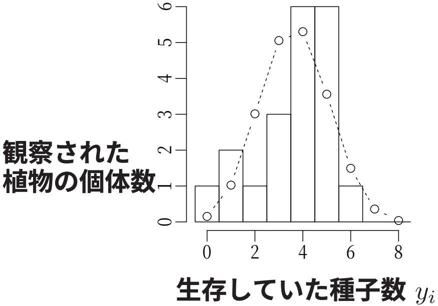
- **生存確率** q **が全個体共通の場合の最尤推定量・最尤推定値は**

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



二項分布で説明できる 8 種子中 y_i 個の生存

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



2. 同じ推定を MCMC でやってみる

“Sampling” using Monte Carlo Markov Chain method

最尤推定と MCMC はちがう!

Maximum likelihood Estimation (MLE) vs. MCMC

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

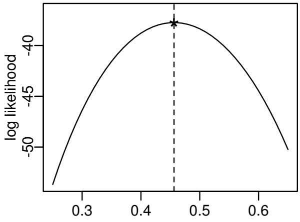
説明のためあえてメトロポリス法を適用してみる

An example for MCMC

MCMC 法を説明するための例題

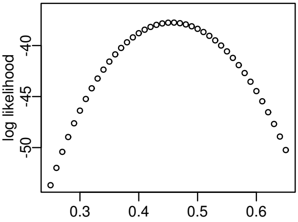
連続的な対数尤度関数

$\log L(q)$



$\rightarrow$

離散化: q がとびとびの値をとる



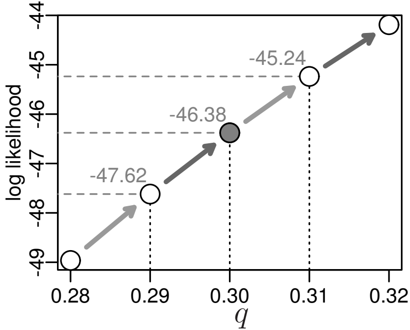
説明を簡単にするため

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**

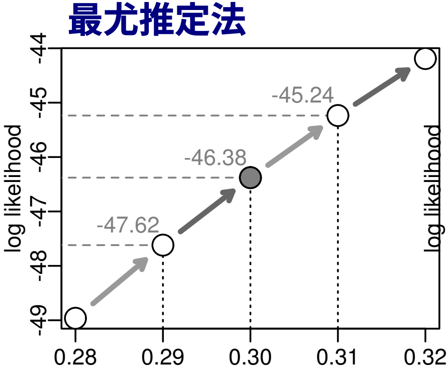
   (新しく選んだ q の値を q_{new} としましょう)

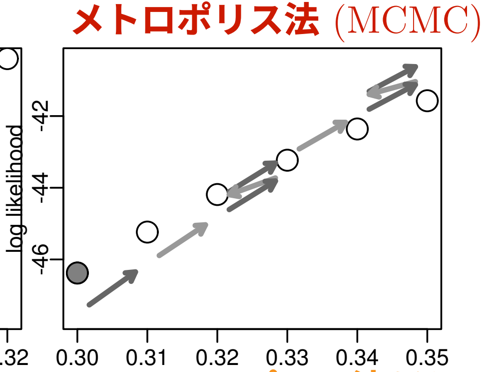
3. **q_{new} における尤度 $L(q_{\text{new}})$ ともとの尤度 $L(q)$ を比較**
   - $L(q_{\text{new}}) \geq L(q)$ **(あてはまり改善)**: $q \leftarrow q_{\text{new}}$
   - $L(q_{\text{new}}) < L(q)$ **(あてはまり改悪)**:
     - 確率 $r = L(q_{\text{new}})/L(q)$ で $q \leftarrow q_{\text{new}}$
     - 確率 $1 - r$ で q を変更しない

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



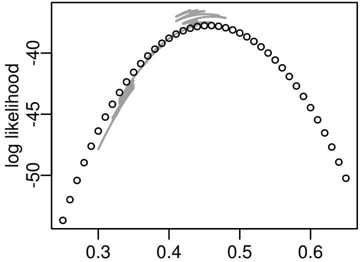


メトロポリス法だと
「単調な山のぼり」にはならない

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

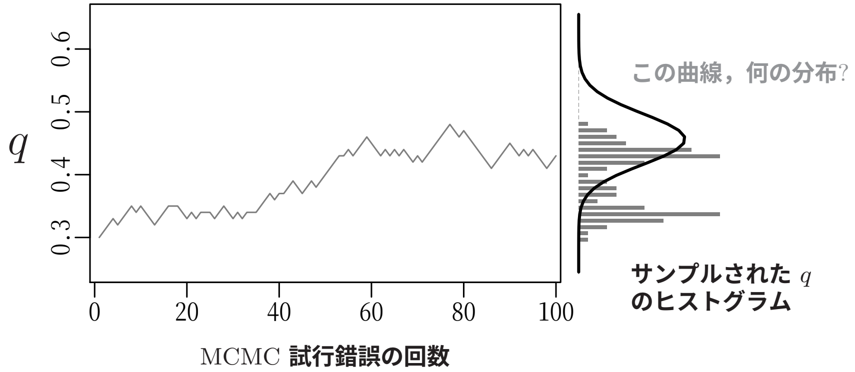


ときどきはでに落っこちる

何のためにこんなことをやるのか?

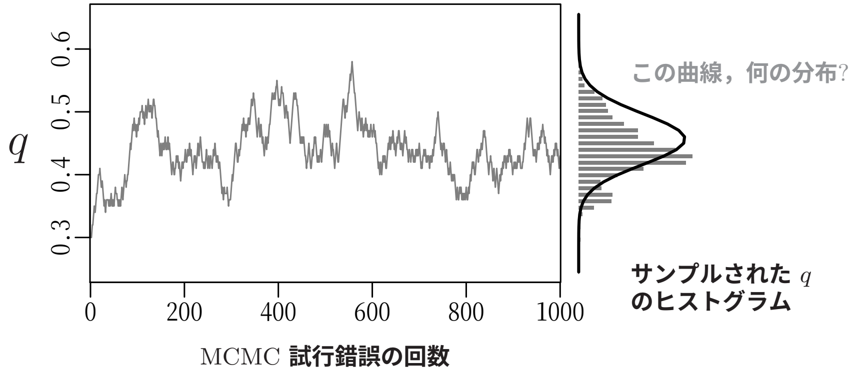
q の変化していく様子を記録してみよう

ステップごとに q の値をサンプリング



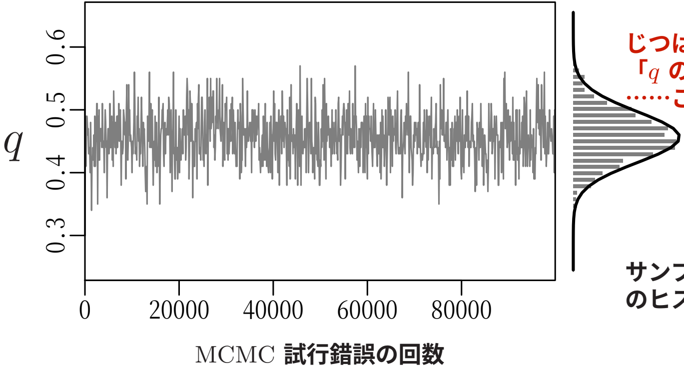
もっと試行錯誤してみたほうがいいのかな?

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



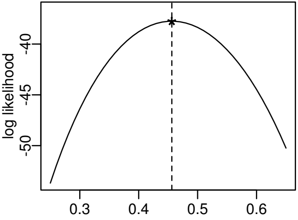
じつはこれは「q の確率分布」……このあと説明

サンプルされた q のヒストグラム

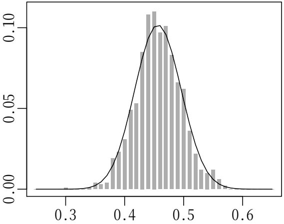
なんだか，ある「山」のかたちにまとまったぞ?

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



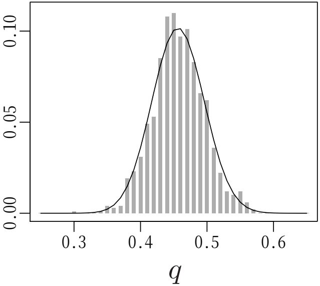
尤度に比例する確率分布からのランダムサンプル

最尤推定はパラメーターの値の点推定

MCMC は "パラメーターの事後分布" (←推定したいこと)

は **こういう分布ですよ** と推定している

MCMC の結果として得られた q の経験分布

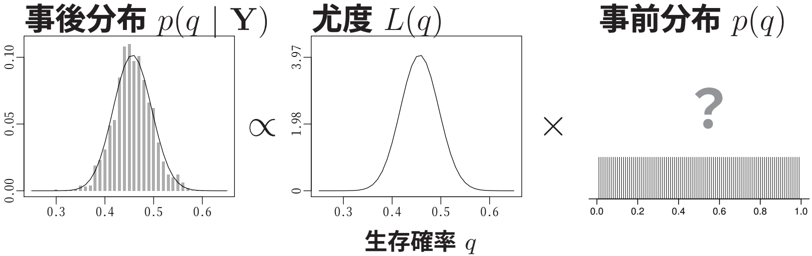


- データと統計モデル (二項分布) を決めて，MCMC サンプリングすると，$p(q)$ からのランダムサンプルが得られる
- このランダムサンプルをもとに，q の平均や 95% 区間などがわかる— **便利じゃないか!**

ベイズ統計モデルの推定

統計モデルとデータにもとづいて事後分布の推定

- パラメーター数の少ないベイズモデルであれば，尤度の数値計算やメトロポリス法で可能
- パラメーター数の多い複雑な統計モデルであれば，**あとで説明する** サンプリングソフトウェアを使用する



3. MCMC のためのソフトウェア

How to “sample” from posterior distribution

“Gibbs sampling” などが簡単にできるような……

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

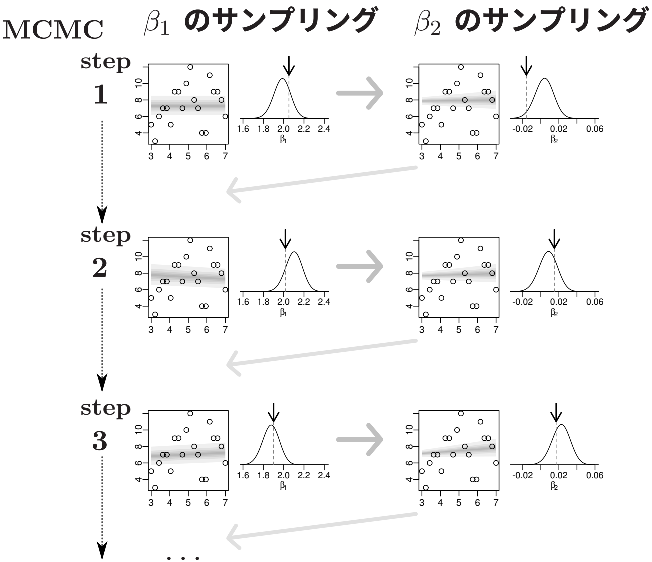
いろいろな MCMC

- **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)

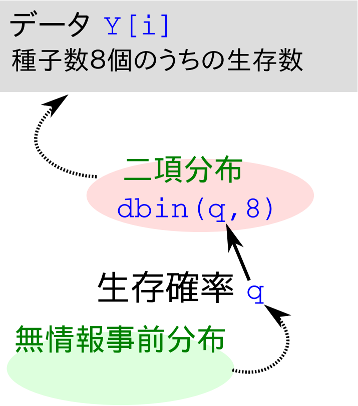


便利な “BUGS” 汎用 Gibbs sampler たち

- 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 — 歴史を変えて……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 — いま一番の注目
      — 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



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.

いろいろな OS で使える JAGS4.2.0

- R core team **のひとり** Martyn Plummer **さんが開発**
  - Just Another Gibbs Sampler
- C++ **で実装されている**
  - 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**
- **開発進行中**
- R **からの使う**: `library(rjags)`

JAGS を R の “したうけ” として使う

BUGS言語

モデルの構造

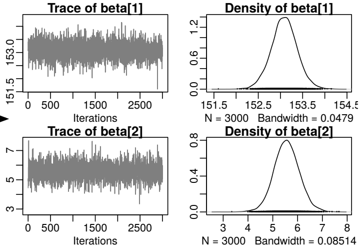
データと
パラメーター
の初期値

サンプリング
の詳細

Input

JAGS

事後分布からの
ランダムサンプル



Output

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……
```

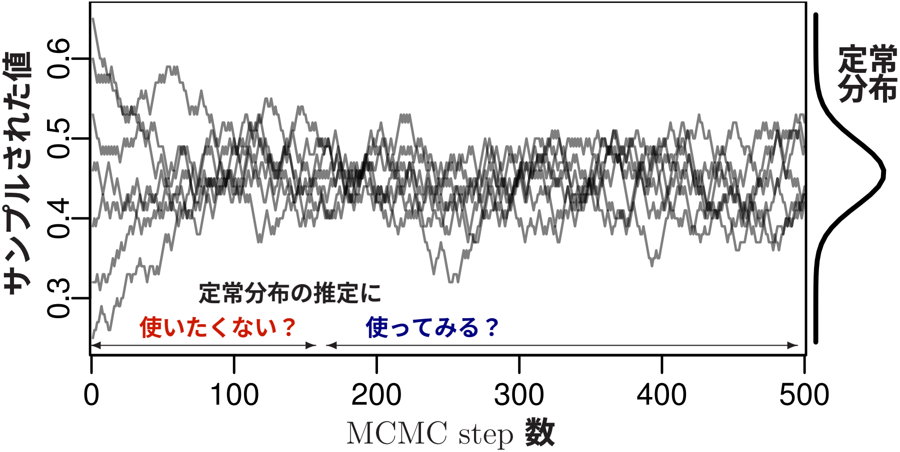
R から JAGS にこんなかんじで仕事を命じる (2 / 3)

```
load("mcmc.RData") # (data.RData ではなく mcmc.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……
```

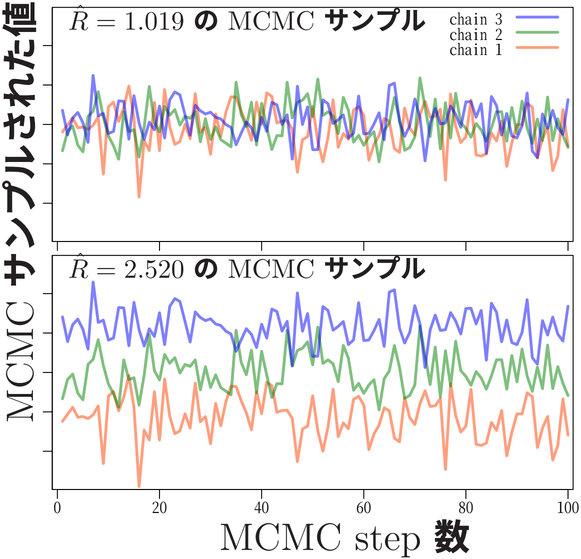
R から JAGS にこんなかんじで仕事を命じる (3 / 3)

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

# サンプリング結果を 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……

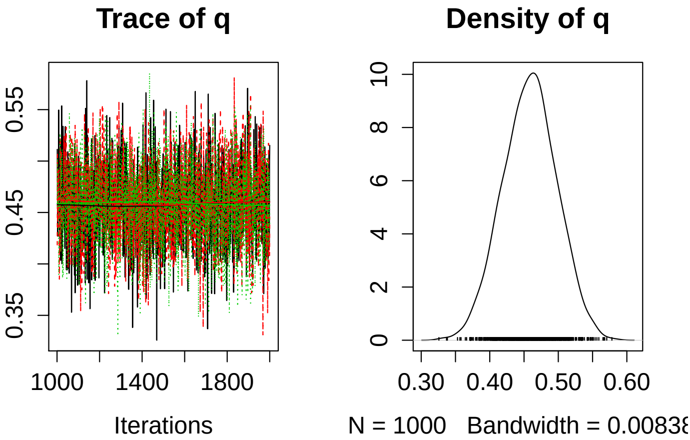
何やら問題あり!

収束診断の $\hat{R}$ 指数

- `gelman.diag(post.mcmc.list)` $\rightarrow$ 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 : 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)`



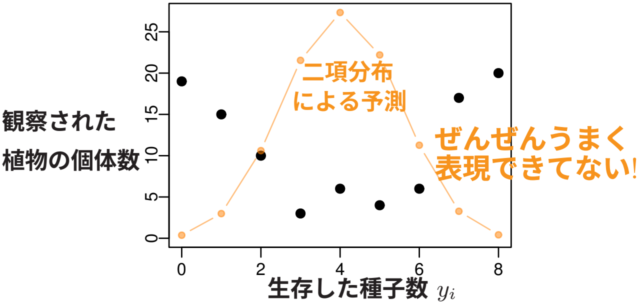
4. GLM を階層ベイズモデル化する!

How to design hierarchical models

個体差・地域差も考慮した統計モデル作り

二項分布では説明できない観測データ!

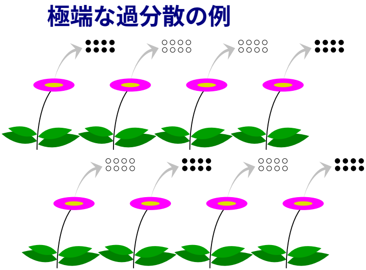
100 **個体の植物の合計** 800 **種子中** **403 個**の生存が見られたので，平均生存確率は 0.50 と推定されたが……

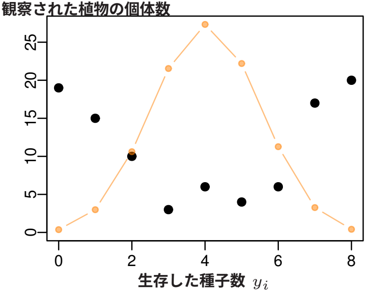


さっきの例題と同じようなデータなのに?

(「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)





- 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

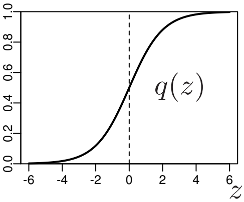
- 生存確率を推定するために **二項分布**という確率分布を使う
- 個体 i の N_i 種子中 y_i 個が生存する確率は二項分布

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

- ここで仮定していること
  - **個体差がある**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)**

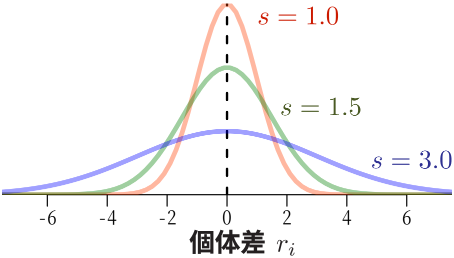
個々の個体差 r_i を最尤推定するのはまずい

パラメーター数 $>$ サンプルサイズ

- 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 $\{r_1, r_2, \cdots, r_{100}\})$ 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

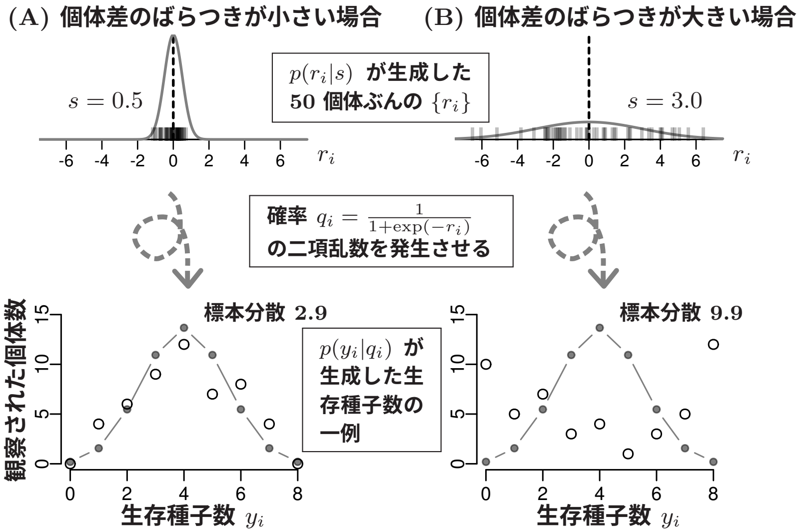
suppose $\{r_i\}$ follow the Gausssian distribution
$\{r_i\}$ のばらつきは正規分布だと考えてみる



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

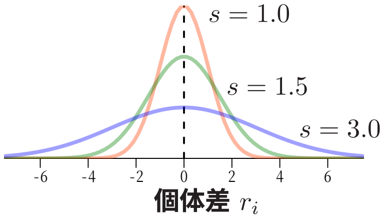
この確率密度 $p(r_i \mid s)$ は r_i の「出現しやすさ」をあらわしていると解釈すればよいでしょう．r_i がゼロにちかい個体はわりと「ありがち」で，r_i の絶対値が大きな個体は相対的に「あまりいない」．

ひとつの例示: 個体差 r_i の分布と過分散の関係



これは r_i の事前分布の指定，ということ

前回の講義で $\{r_i\}$ 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



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

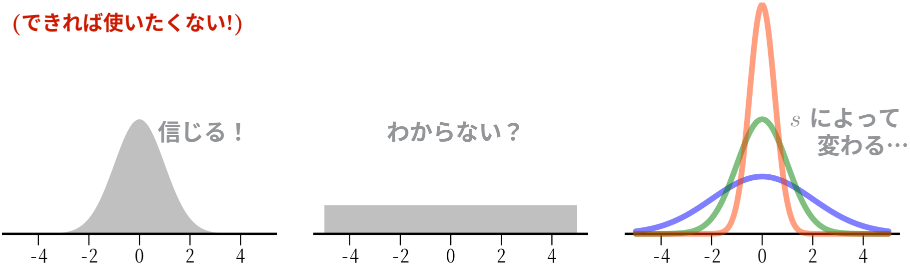
ベイズ統計モデルでよく使われる三種類の事前分布

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**

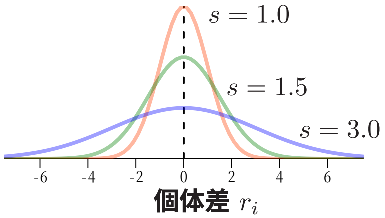
(C) **階層事前分布**



r_i の事前分布として階層事前分布を指定する

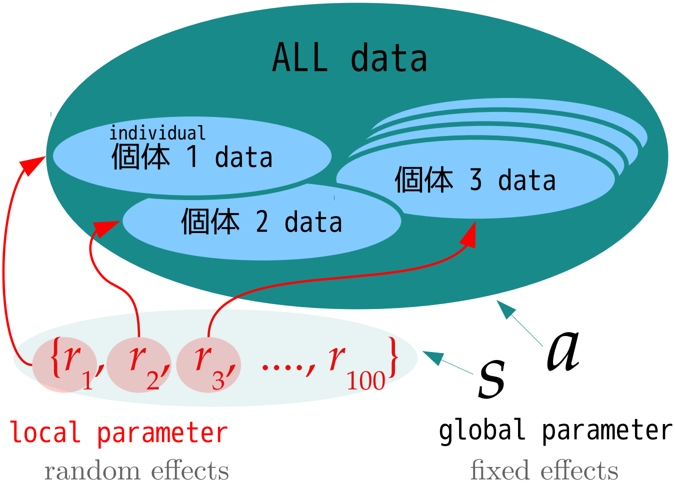
階層事前分布の利点

「データにあわせて」事前分布が変形!



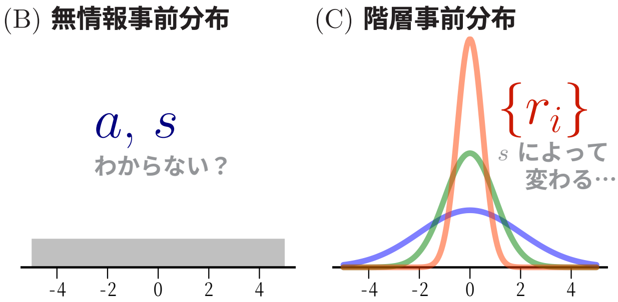
$$p(r_i \mid s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

統計モデルの大域的・局所的なパラメーター



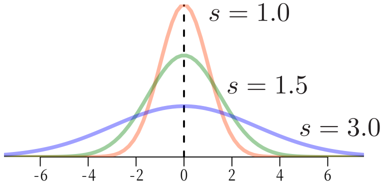
データのどの部分を説明しているのか?

パラメーターごとに適切な事前分布を選ぶ



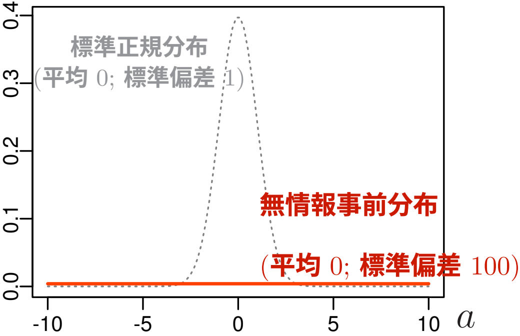
パラメーターの種類	説明する範囲	事前分布
全体に共通する平均・ばらつき	global 大域的	無情報事前分布
個体・グループごとのずれ	local 局所的	階層事前分布

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

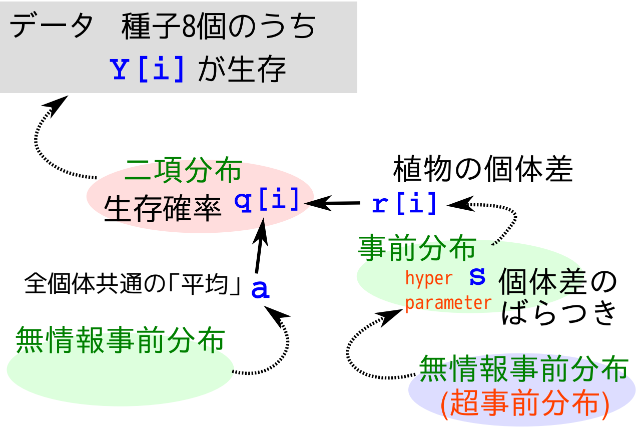
全個体の「切片」 a の無情報事前分布



「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



矢印は手順ではなく，依存関係をあらわしている

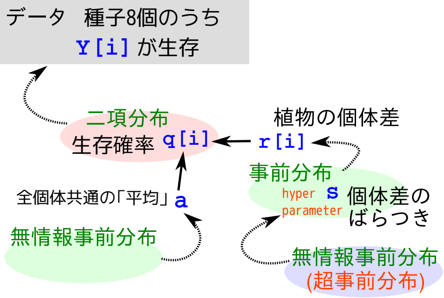
5. 階層ベイズモデルの推定

How to use JAGS sampler?

R の “したうけ” として JAGS を使う

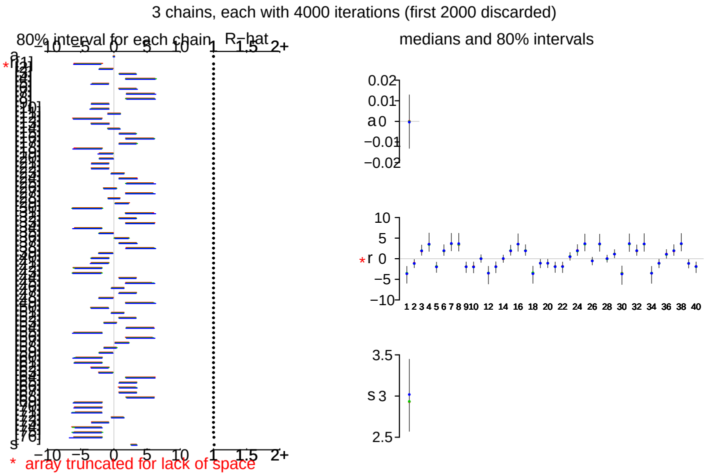
階層ベイズモデルを BUGS コードで記述する

```
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
```

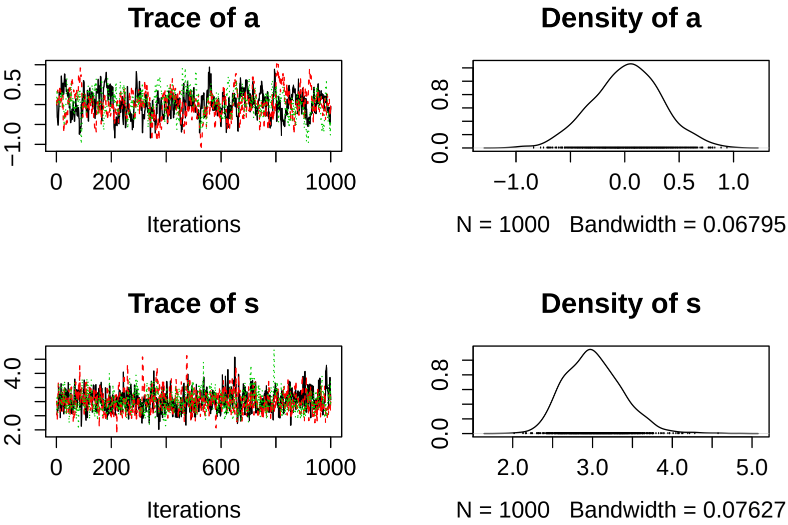


bugs オブジェクトの post.bugs を調べる

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

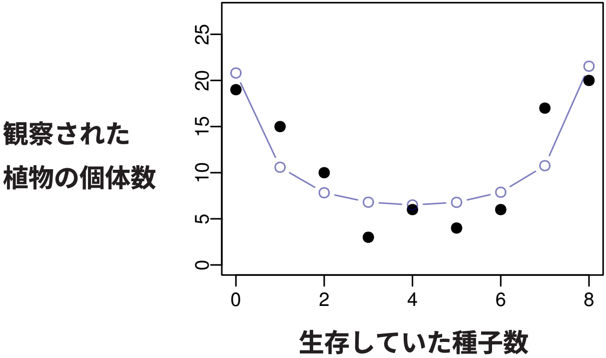
```
3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
n.sims = 3000 iterations saved
         mean    sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
a       0.020 0.321 -0.618 -0.190  0.028  0.236  0.651 1.007   380
s       3.015 0.359  2.406  2.757  2.990  3.235  3.749 1.002  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



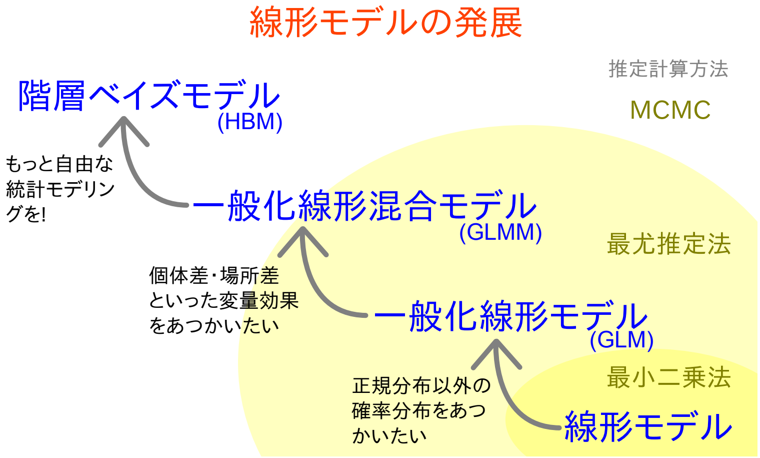
得られた事後分布サンプルを組みあわせて予測

- `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**
- **……このあとごちゃごちゃと計算する必要あるけど，省略……**



statistaical models appeared in the class

“統計モデリング入門” に登場する統計モデル



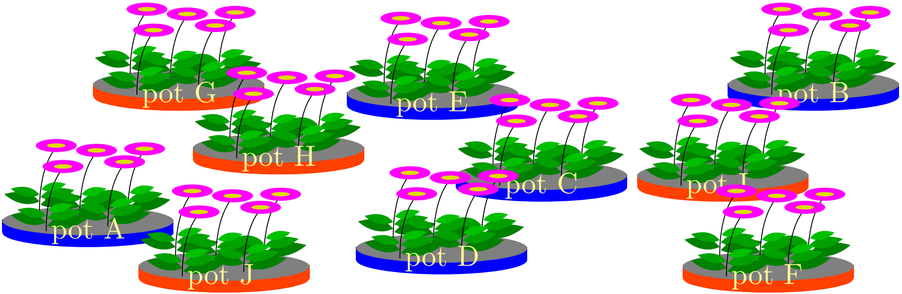
データの特徴にあわせて線形モデルを改良・発展させる

6. 複数のランダム効果をもつ階層ベイズモデル

individual effects + block effects

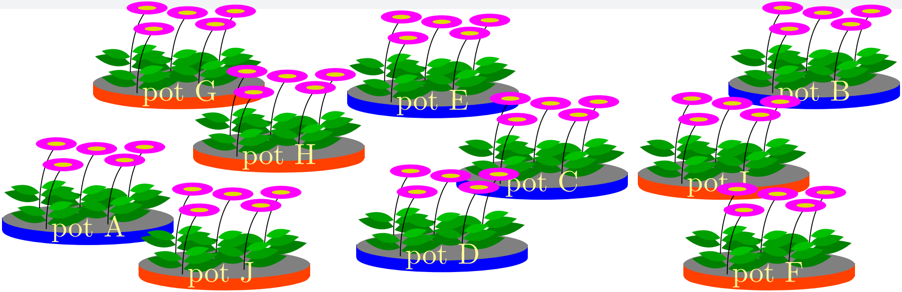
そして “てぬき” モデリングの危なさについて

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 = **C**) 5 鉢 (合計 50 個体)
  - 肥料をやる処理 (f_j = **T**) 5 鉢 (合計 50 個体)

このへんてこな例題，言い換えると……?



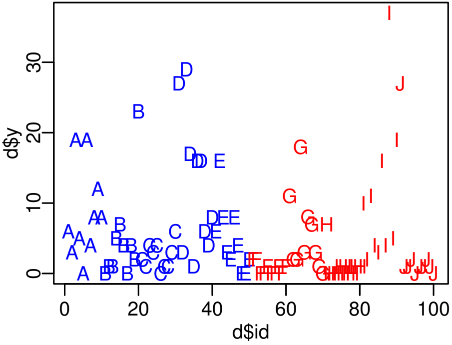
- ○○**教育法**をやったら**児童ごとの算数の得点** y_i が増えるかどうかを調べたい
- **小学校 10 校**，各校に 10 人の児童に算数のテスト (合計 100 児童)
  - **コントロール** ($f_j = \mathrm{C}$) 5 校 (合計 50 児童)
  - ○○**教育法実施** ($f_j = \mathrm{T}$) 5 校 (合計 50 児童)

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
```

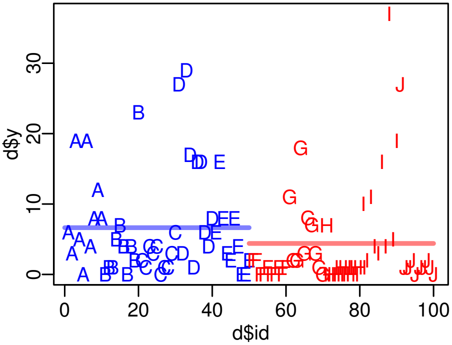
- `id` 列: 個体番号 $\{1, 2, 3, \cdots, 100\}$
- `pot` 列: 植木鉢名 $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \cdots, \mathrm{J}\}$
- `f` 列: 処理: コントロール `C`, 肥料 `T`
- `y` 列: 種子数 (応答変数)

データはとにかく図示する!!



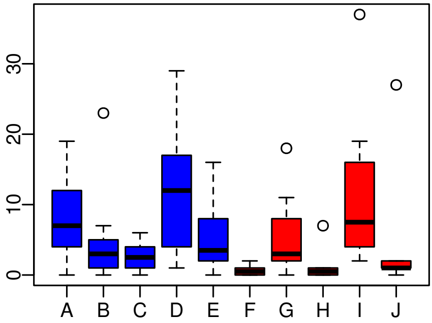
- `plot(d$id, d$y, pch = as.character(d$pot), ...)`
- **コントロール・処理 でそんなに差がない**?

処理ごとの平均も図に追加してみる



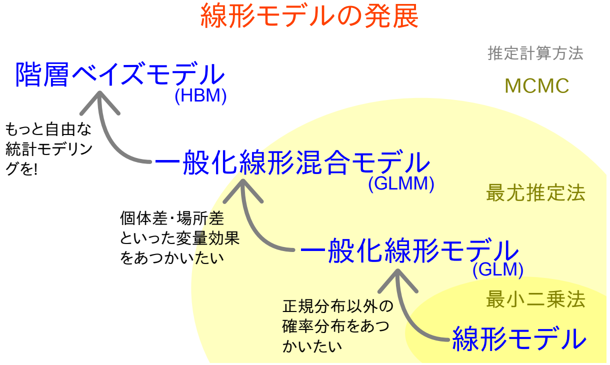
- むしろ **処理** のほうが平均種子数が低い?
- (注) この架空データは **肥料の効果はゼロ** と設定して生成した

個体差だけでなく植木鉢差もありそう?



- `plot(d$pot, d$y, 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



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(glmmML)
> summary(glmmML(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?

統計モデルが不適切 → まちがった「ゆーい差」!

- “個体差はないことにする”, “植木鉢差はないことにする”
- こういう手ぬきな統計モデルを使うと “まちがった「ゆーい差」” が出力
- 図をよくみる → “ないこと” にしない

しかし，複数の ランダム効果 をもつ

統計モデルのパラメーター推定は 難しい!

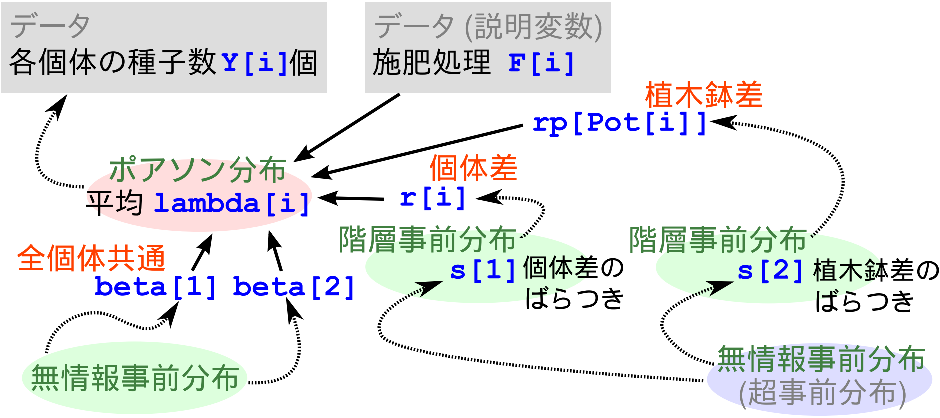
“R だけ” では無理! Use JAGS!

- 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(glmmML) の glmmML()
  - 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

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 $\log(\lambda_i) = a + bf_i +$ (個体差) + (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
```

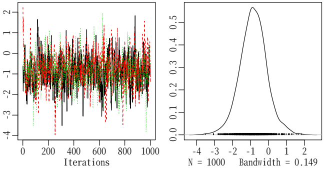
ここでの BUGS coding のポイント

- **因子型の説明変数** $f_i \in \{\mathtt{C}, \mathtt{T}\}$ **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?

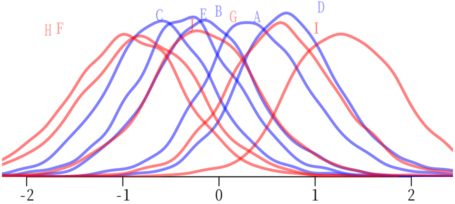


	mean	sd	2.5%	25%	50%	75%	97.5%	Rha
a	1.501	0.529	0.482	1.157	1.493	1.852	2.565	1.0
b	-1.016	0.706	-2.436	-1.476	-0.993	-0.565	0.395	1.0
sigma[1]	1.020	0.114	0.822	0.939	1.014	1.089	1.265	1.0

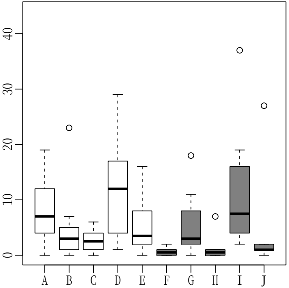
...(略)...

この架空データを生成した種子数シミュレイションでは，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)



ブロック差
rp[j]



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

この講義の流れ: 例題を考えながら理解する

1. **統計モデル・確率分布・最尤推定**
2. **ポアソン分布の一般化線形モデル** (GLM)
3. **二項分布の** GLM
4. MCMC **と階層ベイズモデル**

単純化した例題にそって統計モデルを説明