

# 統計モデリング入門 道総研 [08]

## マルコフ連鎖モンテカルロ法

久保拓弥 [kubo@ees.hokudai.ac.jp](mailto:kubo@ees.hokudai.ac.jp), @KuboBook

道総研勉強会 <http://goo.gl/HQbeoh>

2015-11-17

ファイル更新時刻: 2015-11-12 22:11

# この時間に説明したいこと

- ① MCMC サンプルングのための例題  
二項分布のパラメーターを最尤推定 (前の時間の最初に登場!)
- 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  
事後分布からサンプルングしたい

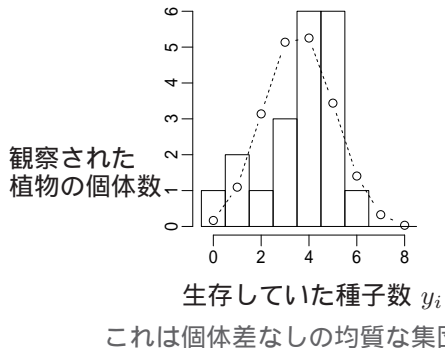
# 1. MCMC サンプリングのための例題

二項分布のパラメーターを最尤推定 (前の時間の最初に登場!)

あえてものすごく簡単な例題

簡単すぎる例題: 生存確率は全個体で同じ (「個体差」なし)

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

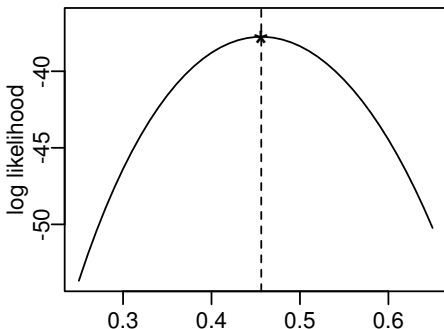
- 対数尤度  $L(q \mid \text{データ})$  が最大になるパラメーター  $q$  の値をさがすこと

- 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

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

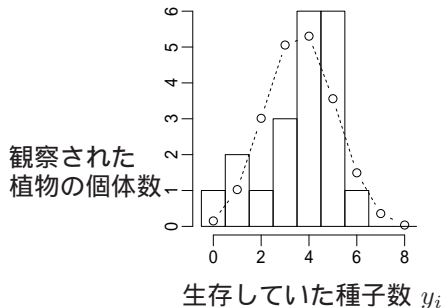
- 生存確率  $q$  が全個体共通の場合の最尤推定量・最尤推定値は

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



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「なぜかしら」ベイズ統計モデルと関連づけ

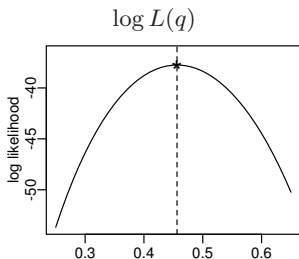
## ここでやること: 尤度と MCMC の関係を考える

- 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」.....??

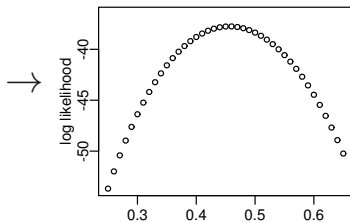
MCMC をもちださなくてもいい簡単すぎる問題  
説明のためあえてメトロポリス法を適用してみる

# 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



離散化:  $q$  がとびとびの値をとる

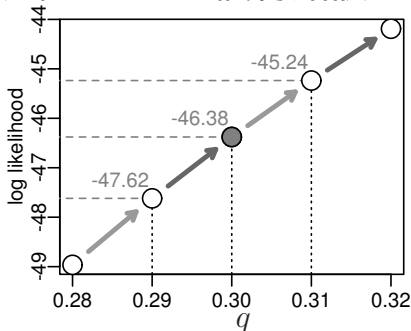


説明を簡単にするため  
生存確率  $q$  の軸を離散化する

(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  $q_{\text{new}}$  としましょう)

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

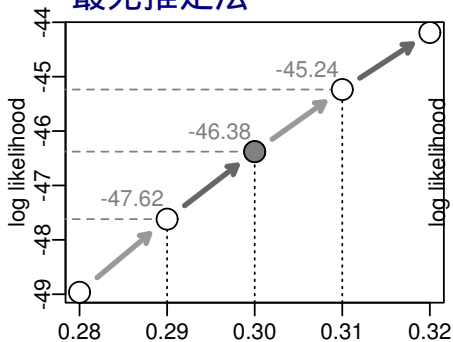
- $L(q_{\text{new}}) \geq L(q)$  (あてはまり改善):  $q \leftarrow q_{\text{new}}$
- $L(q_{\text{new}}) < L(q)$  (あてはまり改悪):
  - 確率  $r = L(q_{\text{new}})/L(q)$  で  $q \leftarrow q_{\text{new}}$
  - 確率  $1 - r$  で  $q$  を変更しない

## ④ 手順 2. にもどる

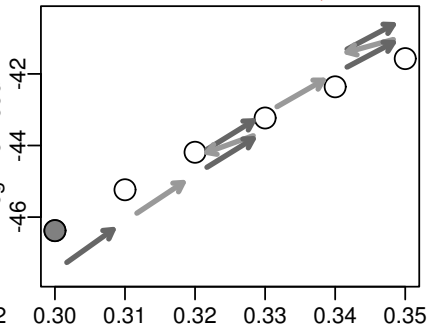
( $q = 0.01$  や  $q = 0.99$  でどうなるんだ, といった問題は省略)

# メトロポリス法のルールで $q$ を動かす

最尤推定法



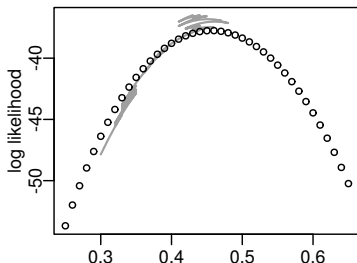
メトロポリス法 (MCMC)



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

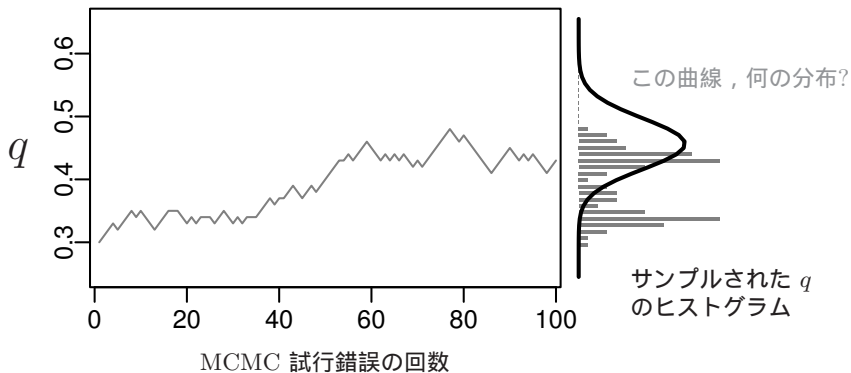


ときどきはでに落っこちる

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

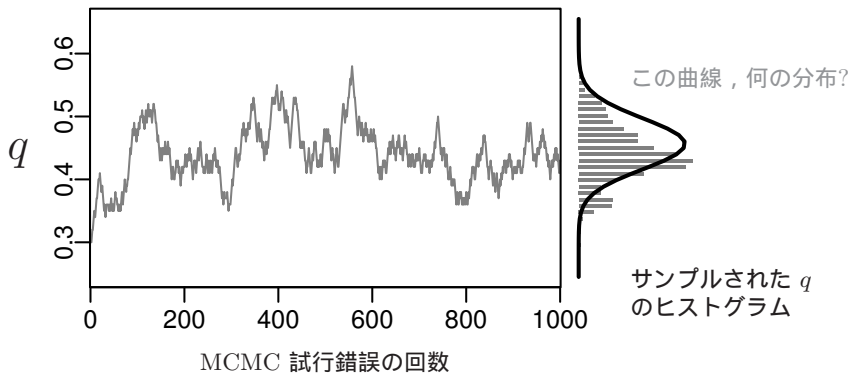
# ステップごとに $q$ の値をサンプリング



もっと試行錯誤してみたほうがいいのか?

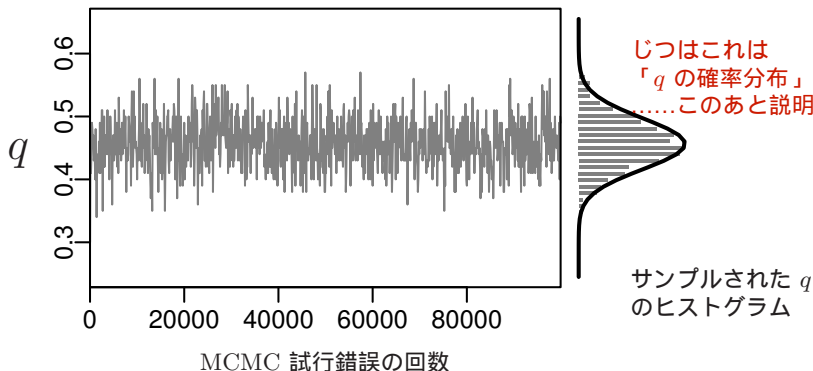


# もっと長くサンプリングしてみる



まだまだ.....?

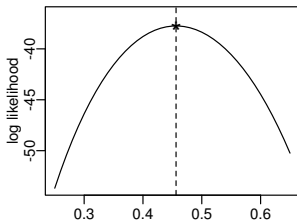
## もっともっと長くサンプリングしてみる



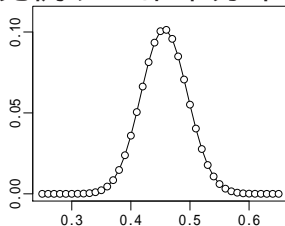
なんだか、ある「山」のかたちにとまとまったぞ?

# MCMC は何をサンプリングしている?

対数尤度  $\log L(q)$

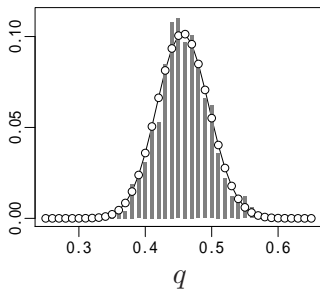


尤度  $L(q)$  に  
比例する確率分布



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)

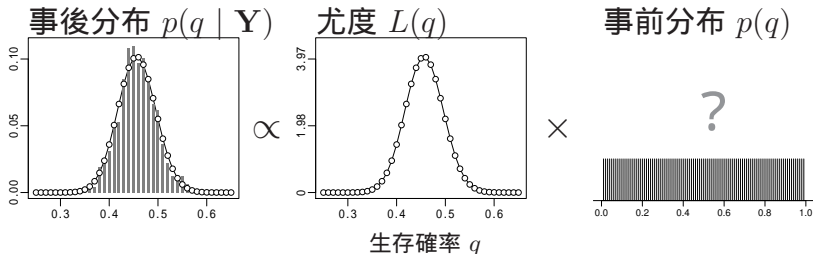
$$\text{(事後分布)} \propto \frac{\text{尤度} \times \text{事前分布}}{\text{(データが得られる確率)}}$$

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



# 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

事後分布からサンプリングしたい

Gibbs sampling

# 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
- じっくりと開発進行中
- 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
)

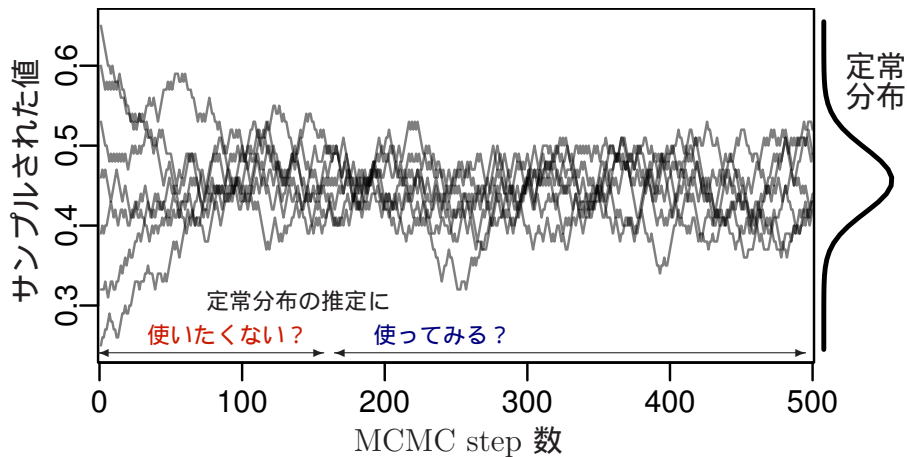
# まだ次につづく.....
```

## R から JAGS にこんなかんじで仕事を命じる (3 / 3)

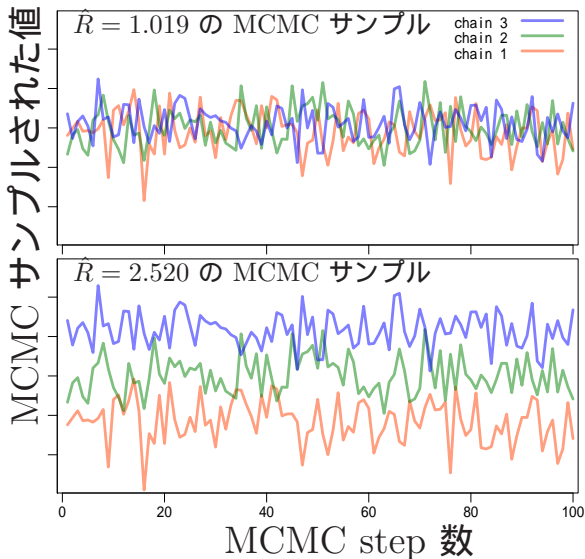
```
# burn-in
update(model, n.burnin) # burn in

# 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)` → 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)`

