20080514 高橋正義

Bayesian Computation with R
Chapter 11
Using R to Interface with WinBUGS

役立つ情報

Bayesian Computation with R web

http://bayes.bgsu.edu/bcwr/

正誤表や WinBUGS 用のスクリプトなどがある

11.1 Introduction to WinBUGS

WinBUGS はじめに

WinBUGS は MCMC を使った複雑な統計モデルによるベイズ当てはめのためのソフトウェア

一般的な事後分布にギブズサンプリングと一般的な提案分布(proposal densities)を使ったベイズモデルで解くことのできるプログラム

使い方は簡単。

二項分布(n,p)する観測値 y と p となる事前ベータ分布(α,β)で α,β ともに 0.5 とすると、サンプル数 n=50 で y=7 の時、90%の確率で p の予測することができる。

WinBUGS の使用方法

```
model
{
        y ~ dbin(p,n)
        p ~ dbeta( alpha, beta)
}
```

スクリプトは model から始める

前提となる分布は~で示す

分布の記述方法は R と同じ

モデルに続いてデータと未知のパラメータを記述。冒頭には data。

```
data
list(y = 7, n = 50, aplpha = 0.5, beta = 0.5)
```

最後に MCMC シミュレーションの初期値を与える。冒頭には inits。

```
inits
list(p = 0.1)
```

モデル、データ、初期値を与えたらシミュレーション時にモニタリングするパラメータを Sample Monitor Tool で指定する(例の場合は p)

Update tool を使うと事後分布から任意サイズのシミュレートサンプルを取り出すことができる。

MCMC 計算が正常に終了すれば Plot や(計算が収束したかどうか判断するための)各種統計量を計算できる。

WinBUGS は高次のベイズモデル(階層ベイズモデル?)を計算することができる。R などの外部プログラムと連

携してデータ解析をするツールができている。
ここではRとbugsの場合を示す。

11.2 An R Interface to WinBUGS

WinBIGSのためのRのインターフェイス

RとWinBUGSを使うには、R2WinBUGSとBRugsをインストール。
インストールが無事できればRからwinBUGSを使ってベイズプログラムを走らせることが簡単にできる。以下の4つの入力が必要。
model. WinBUGSでいうmodelと同じ
data. モデルの変数(ベクターか行列)や定数
Parameters. シミュレーションでモニタリングする変数
Initial values. 初期値
例えば、model1.bugにモデルを定義し、データ、変数、初期値をR上でそれぞれdata.,parameters,initsで定義したとするとのコマンドでRのbugsコマンドでベイズシミュレーションを行うことができる。
> model.sim <- bugs (data, inits, parameters, "model.bug")
モデルが実行可能であればWinBUGSがバックグラウンドで動く。シミュレーションが完了すればwinBUGSは終了してR consoleに戻る。bugsの出力はWinBUGSからのアウトプットである。model.simにはモニタリングした変数のシミュレーション結果が出力される。
bugsのオプションを使うことでシミュレーションを操作できる。
bugs(data, inits, parameters.to.save, model1.file = "model1.bug", n.chains = 3, n.iter = 2000, n.burnin = floor(n.iter/2), n.thin = max(1, floor(n.chains*(n.iter - n.burnin)/1000)),bin = (n.iter - n.burnin) / n.thin)
n.chains マルコフ連鎖の数。デフォルトは3
n.iter それぞれの連鎖の反復数
n.burnin バーンインまでの数(不変分布に収束するまでの数)、デフォルトは反復数の半分(n.iter/2)
n.thin 間引き率 標本として収集する割合。デフォルトは2(1つ飛ばしで収集)
bin 結果の収集回数。デフォルトは最後のもののみを保存。

11.3 MCMC Diagnostics Using the boa Package

boaパッケージを使ったMCMCの診断

MCMCを実行してシミュレーションの結果が得られたら、事後分布が得られたかどうかについて何らかの判断を下さなければならない。
その際、下記のような点について考慮する必要がある。
1.いくつの連鎖がシミュレートされたのか？初期値の違いで結果に違いがあったか?
2.バーンインまでの長さ
3.いくつのシミュレーションの経路(simulation draws)が見られたか?
4.シミュレーションの標準誤差？
5.シミュレーションの経路に高い相関関係が見られたか?

boa(Bayesian Outout Analysis)パッケージはCODA(Convergence Diagnosis and Output Analysis Software)パッ

ケージをベースにしている。このパッケージは MCMC 出力を診断に有用な多種の機能を提供するものである。
・多種の記述統計（summary statistics） 例えば平均、SD、分位、最高確率密度区間？（highest probability density intervals）、バッチ平均ベースの修正出力に基づくシミュレーション標準誤差など
・異なるパラメータサンプルでシミュレートした自己相関（autocorrelation）と相互相関（cross-correlation）の比較
・収束の診断 例えば、Geweke,Gelman, Rubin の方法、Raftery と Lewis の方法など
・図の描画 例えば、ラグ収束、密度の推定、移動平均など
bugs 関数で WinBUGS で MCMC サンプリングを行なった後に boa パッケージを使うのが便利。boa function はメニューオプションである boa.muenu().を使うと利用できる。メニューの Import Data > Data Matrix Object を選びシミュレートしたパラメータのベクターか行列をパッケージに読み込む。メニューの Analysis と Plot に MCMC 診断ツールがある。

11.4 A Change-Point Model

変化点モデル

Carlin et al (1992)によるイギリスの石炭採掘災害に関する分析から始める。1851 年から 1962 年までの災害数が記録されている。t 年の災害数を y_t とする(t は実年から 1850 を引いた値）。データを見ると 19 世紀末の数年は事故率が減少しているように見える。そこで $t<\tau$ の時、y_t は log 平均が $\log\mu_t=\beta_0$ のポアソン分布で、後年（$t\geqq\tau$）は $\log\mu_t=\beta_0+\beta_1$ となると仮定した。

式

非負の場合 σ()は 1、その他は 0 と定義。未知の変数は回帰パラメータである β_0, β_1 で change-point 変数は τ である。β_0, β_1 は曖昧な一様分布を事前分布とし、τ には区間（1.N）、ただし N は年、で一様分布を事前分布とした。

WinBUGS を使う第一歩は、BUGS 言語で短いスクリプトを記述することである。

ここで観測年は D[year]、対応する平均は mu[year]と記述する。（未知の）パラメータは b[1],b[2]と、change-point パラメータ τ は changeyear とした。式の文法は R とほとんど同じ。

```
D[year] ~ dpois(mu[year])
```

D[year]は平均 mu[year]のポアソン分布をとる。同様に

```
b[j] ~ dnorm( 0.0,1.0E-6)
```

βj は平均 0 の正規事前分布で精度（precision、分散の逆数）は 0.000001 である。WinBUGS ではすべてのパラメータで厳密な分布を指定しなければならないので、この正規事前分布は一様な事前分布とほぼ同じ意味である。また、

```
changeyear ~ dunif(1,N)
```

は、τ が区間(1,N)において一様な事前分布を持つことを意味する。オペレーター<-は変数への代入を意味する。例えば

```
log(mu[year]) <- b[1] + step(year - changeyear) * b[2]
```

は右側の線形式を左側の変数 log(mu[ear])に代入することを意味する。

WinBUGS の step 機能は既に定義した σ()と同じ機能である。モデル全体をファイル名 coalmining.bug として保存したものは下記である。

```
model
{
```

```
  for( year in 1 : N ) {
      D[year] ~ dpois(mu[year])
      log(mu[year]) <- b[1] + step(year - changeyear) * b[2]    }
  for (j in 1:2) {b[j] ~ dnorm( 0.0,1.0E-6)}
  changeyear ~ dunif(1,N)
}
```

注:保存する場所は R のワークディレクトリ
次にデータについて、年 N と観測値 D を定義する。

```
N=112
D=c(4,5,4,1,0,4,3,4,0,6,
3,3,4,0,2,6,3,3,5,4,5,3,1,4,4,1,5,5,3,4,2,5,2,2,3,4,2,1,3,2,
1,1,1,1,1,3,0,0,1,0,1,1,0,0,3,1,0,3,2,2,
0,1,1,1,0,1,0,1,0,0,0,2,1,0,0,0,1,1,0,2,
2,3,1,1,2,1,1,1,1,2,4,2,0,0,0,1,4,0,0,0,
1,0,0,0,0,0,1,0,0,1,0,0)
data=list("N","D")
```

続いてパラメータを定義する。

```
parameters <- c("changeyear","b")
```

でモニターしたいパラメータτとβを指定する。
最後に

```
inits = function() {list(b=c(0,0),changeyear=50)}
```

パラメータ(β_1,β_2)の初期値を(0,0)、τの初期値を 50 と指定する。

```
bugs で WinBUGS を動かす。
```

注)boa パッケージ(なぜか arm も)のインストールを忘れずに！

```
coalmining.sim <- bugs (data, inits, parameters, "coalmining.bug", n.chains=3, n.iter=1000)
```

bugs の結果はシミュレーションオブジェクト coalmining.sim に渡される。
シミュレーションの軌跡に関する基本的な情報はシミュレーションオブジェクトである coalmining.sim の print 機能で出力できる。下記出力を見ると連鎖数は 3、反復数はそれぞれ 1000 で最初の 500 でバーンインとしている。
統計の要約を見ると全部で 1500 反復が保存されている。

```
> print(coalmining.sim)
Inference for Bugs model at "coalmining.bug", fit using WinBUGS,
 3 chains, each with 1000 iterations (first 500 discarded)
 n.sims = 1500 iterations saved
           mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
```

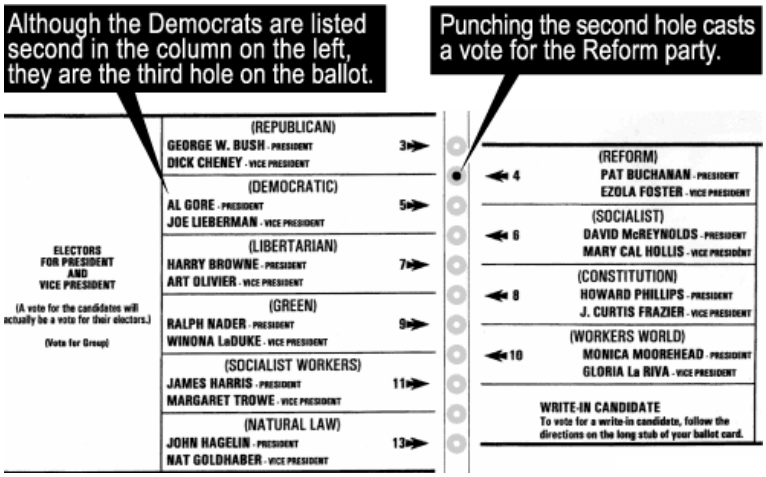
changeyear 39.5 2.1 36.1 37.8 39.8 40.7 43.6   1 1500
b[1]   1.1 0.1   0.9   1.1   1.1   1.2   1.3   1   350
b[2]   -1.3 0.2 -1.6 -1.4 -1.3 -1.2 -1.0   1 1300
deviance   337.5 2.6 334.2 335.6 336.8 338.6 344.0   1   820

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 3.5 and DIC = 341.0
DIC is an estimate of expected predictive error (lower deviance is better).

attach.bugs コマンドを使うとシミュレートした結果を R から使うことができる。
attach.bugs(coalmining.sim)
さらに plot(density())で結果を図示できる。今回の場合、1850 年からおよそ 37 年と 40 年たったところで変換点が見られる。
plot(density(changeyear))
β_1とβ_2の密度は下記で表示できる。
par(mfrow=c(2,1))
plot(density(b[,1]),xlab="beta1")
plot(density(b[,2]),xlab="beta2")

11.5 A Robust Regression Model
　頑健（ロバスト）な回帰モデル
次は頑健（ロバスト）な単回帰モデルの例
フロリダ州の大統領選挙での 1996 年と 2000 年の投票数の関係について
67 郡毎の 2000 年はパット・ブキャナン候補の投票、1996 年のロス・ペロー候補の投票数である。明らかな外れ値 1 つをのぞいて線形関係が見られる。外れ値はブキャナン候補に対するパームビーチ郡の非常に高い投票数によるもので、チョウ型投票用紙（butterfly ballot　下図参照）によるものである。



y_i　i 郡におけるブキャナン候補の投票数の平方根

xi　i 郡におけるペロー候補の投票数の平方根

初期の分析から通常の誤差をみると不適切だが下記回帰モデルを使う。

$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

$\varepsilon_1, \cdots, \varepsilon_n$ は平均 0 のスケールパラメータ σ の t 分布のランダムサンプルで、自由度 ν は 4

このモデルは下記の混合正規分布で表せる

$y_i \sim N(\beta_0 + \beta_1 x_i, (\tau\lambda_i)^{-1/2})$

$\lambda_i \sim gamma(2,2)$

β_0, β_1 を一様な事前分布、精度 τ は標準無情報事前分布（1/τ）

モデルは robust.bug

```
model {
for (i in 1:N) {y[i] ~ dnorm(mu[i],p[i])
                    p[i] <- tau*lam[i]
                    lam[i] ~ dgamma(2,2)
                    mu[i] <- b[1]+b[2]*x[i]}
for (j in 1:2) {b[j] ~ dnorm(0,0.001)}
tau ~ dgamma(0.001,0.001)
}
```

データは LearnBayes パッケージにあるデータセット election

データにはペアの観測値数 N，投票数 y、共変量 x が含まれる。データを呼び出すには

```
data(election)
attach(election)
y=sqrt(buchanan)
x=sqrt(perot)
N=length(y)
```

初期値の指定は

```
inits = function() {list(b=c(0,0),tau=1)}
```

モニタリングするパラメータの指定は

```
parameters <- c("tau","lam","b")
```

bugs を呼び出すコマンドは

```
robust.sim <- bugs (data, inits, parameters, "robust.bug")
```

実際には

```
robust.sim <- bugs (data, inits, parameters, "robust.bug", n.chains=3, n.iter=1000)
```

ペロー候補の平均に対するブキャナン候補の平均について見てみたい。

R では x の平方根 xo を適切な大きさのデータマトリクスに入れる。ベクトル b とこのマトリクスを掛け合わせ、事後分布を得る。この事後分布 meanresponse の 5,50,95 パーセンタイルを計算して図にしたのが Fig.11.4 であ

る。外れ値に影響を受けない回帰モデルが得られたことがわかる。

```
attach.bugs(robust.sim)
xo<-seq(18,196,2)
x0<-cbind(1,xo)
meanresponse<-b%*%t(x0)
meanp<-apply(meanresponse,2,quantile,c(.05,.5,.95)
lines(xo,meanp[2,])
lines(xo,meanp[1,],lty<-2)
lines(xo,meanp[3,],lty<-2)
```

11.6 Estimating Career Trajectories

キャリア経路の推測

一般にプロ運動選手の運動能力はそのキャリア中盤までは上昇し、その後、そのキャリアを終えるまで徐々に低下していく傾向にある。野球選手の場合、j 年目のシーズンの成績を打席数"at-bat"から三振数"stikeout"を引いた打数？(number of balls)を n_j、本塁打数を y_j で表すこととする。本塁打率 y_j/n_j が選手の年齢 x_j の関数となっているかが知りたい。図 11.5 は名強打者の Mickey Mantle(1950～60 年代にヤンキースで活躍したメジャー史上最高といわれるスイッチヒッター)の例である。

y_i は二項分布(n_j,p_j)で当てはめる。ただし p_j は $_j$ 年目の本塁打率である。

二次の対数式(logistic quadratic model)

$$\mathrm{Log}(p_j / (1- p_j) = \beta_0+\beta_1 x_j +\beta x^2_j$$

で推定する。glm 関数で計算した結果が Fig.11.5

選手の能力のピークがあるのか、あるとすればそれが何歳頃なのか?ということに興味があるだろう。

上記の二次式では $\beta_2<0$ の時、ピークは

$$Age_{\,\mathrm{PEAK}} = -\beta_1/2\beta_2$$

でり、ロジットスケールでのピークの確率は

$$PEAK = \beta_0 -\beta_1^2 / 4\beta_2$$

である。しかし、選手の現役期間が 15-20 年程度であることから正確なピークとそのときの年齢を割り出すことが難しい。しかし多くの野球選手で似たようなキャリア経路が見られることから、似たような能力を持つ多くの野球選手の経歴をまとめることでより正確なキャリア経路を推定することができるだろう。

k 人の似たような能力を持つ選手がいたとして、1 から T までの j シーズンにおける選手 i の本塁打数を y_{ij}、打数を n_{ij}、年齢を x_{ij} とする。選手すべてを合わせた確率 p_{ij} はロジスティックモデル

$$\mathrm{Log}(p_{ij} / (1- p_{ij}) = \beta_{i0}+\beta_{i1} x_{ij} +\beta_2 x^2_{ij},\ j =1, ..., T_i.$$

を満たす。$\beta_i=(\beta_{i0}, \beta_{i1}, \beta_{i2})$ は選手 i の回帰共分散ベクトル

交換可能性を表すために、$\beta_1,\cdots,\beta_k$ は同じ平均 μ_β で分散共分散行列が V の多変量正規事前分布からランダムにサンプルする。

$$\beta_i |\mu_\beta, \mathrm{R} \sim N3(\mu_\beta, V),\ i = 1,.., k.$$

第 2 段階の事前分布として曖昧な事前分布にハイパーパラメーターを仮定する。

$$\mu_\beta \sim \mathrm{C},\ \mathrm{V} \sim \text{inverse Wishart}(\mathrm{S}^{-1}, \nu),$$

式は行列 S で自由度 ν の逆ウィッシャート分布(Inverse Wishart)分布

WinBUGS では分散共分散行列がこの逆ウィッシャート分布で与えられる。

$$P = \sim \text{inverse Wishart}(S, \nu).$$

sluggerdata　10 人の名ホームランバッターのデータ

carrertraj.setup という R の関数を使う。

```
data(sluggerdata)
s=careertraj.setup(sluggerdata)
N=s$N; T=s$T; y=s$y; n=s$n; x=s$x
```

ここで N は選手数、ベクトル T は選手毎の現役年数で、10 列 23 行の行列 *y* は列が選手 *i* の本塁打数を表す。同様に、行列 n は打席数データである。

career.bug に WinBUGS 用のモデルを記述した。beta は選手 *i* の回帰ベクトルをあらわし、

```
beta[i , 1:3] ~ dmnorm(mu.beta[], R[ , ])
```

は多変量正規事前分布を定義していて、

```
    y[i,j] ~ dbin(p[i,j],n[i,j])
    logit(p[i,j])<-beta[i,1]+beta[i,2]*x[i,j]+beta[i,3]*x[i, j]*x[i, j]
```

はロジスティックモデルを記述している。

```
mu.beta[1:3] ~ dmnorm(mean[1:3],prec[1:3 ,1:3 ])
R[1:3 , 1:3] ~ dwish(Omega[1:3 ,1:3 ], 3)
```

は 2 段目の事前分布。mu.beta は平均ベクトルで、平均 mean、予測行列 prec の多変量正規事前分布から推定する。予測行列 R はスケール行列 Omega、自由度 3 のウィッシャート分布を用いる。

モデル全体は下記。

```
model
{
for( i in 1 : N ) {
  beta[i , 1:3] ~ dmnorm(mu.beta[], R[ , ])
  for( j in 1 : T[i] ) {
    y[i,j] ~ dbin(p[i,j],n[i,j])
    logit(p[i,j])<-beta[i,1]+beta[i,2]*x[i,j]+beta[i,3]*x[i, j]*x[i, j]
    }
}
mu.beta[1:3] ~ dmnorm(mean[1:3],prec[1:3 ,1:3 ])
R[1:3 , 1:3] ~ dwish(Omega[1:3 ,1:3 ], 3)
}
```

ハイパーパラメータは下記のように定義

```
mean = c(0, 0, 0)
```

```
Omega=diag(c(.1,.1,.1))
prec=diag(c(1.0E-6,1.0E-6,1.0E-6))
```

初期値は下記

```
beta0=matrix(c(-7.69,.350,-.0058),nrow=10,ncol=3,byrow=TRUE)
mu.beta0=c(-7.69,.350,-.0058)
R0=diag(c(.1,.1,.1))
```

データ、モニタリングするパラメータは以下のように指定。

```
data=list("N","T","y","n","x","mean","Omega","prec")
inits = function() {list(beta=beta0,mu.beta=mu.beta0,R=R0)}
parameters <- c("beta")
career.sim <- bugs (data, inits, parameters, "career.bug", n.chains=1, n.iter=10000, n.thin=1)
```

career.sim に結果を出力。

各選手のピークの年齢を出したい。以下のループコマンドを使って算出。

```
peak.age=matrix(0,5000,10)
for (i in 1:10)
        peak.age[,i]=-career.sim$sims.list$beta[,i,2]/2/career.sim$sims.list$beta[,i,3]
```

これを boa パッケージを使って計算。

11.7 Further Reading

原書を参照して下さい。