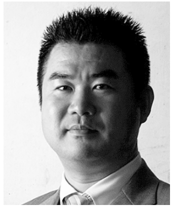
[連載] フリーソフトによるデータ解析・マイニング　第82回

# 空間の統計学(10)：ベイズ空間計量経済学②

慶應義塾大学総合政策学部准教授
古谷　知之（Furutani Tomoyuki）

■兵庫県生まれ。2001年東京大学大学院工学系研究科博士課程修了。博士（工学）。東京大学大学院助手、慶應義塾大学環境情報学部専任講師を経て、07年４月より現職。専門分野：空間統計学、都市交通計画、観光政策。



## 1. はじめに

前回は、マルコフ連鎖モンテカルロ法（MCMC）による線形回帰モデルのベイズ推定法について紹介しました。今回は、いよいよ空間計量経済モデルのベイズ推定について紹介します。具体的には、空間的自己回帰モデル、空間ダービンモデル、空間誤差モデル及びマルチレベルモデルのベイズ推定について紹介します。このうち、空間的自己回帰モデル、空間ダービンモデル、空間誤差モデルについては、前回紹介したJAGSとR2jagsパッケージを使ってベイズ推定します。

演習には、前回までと同様に、首都圏の市区町村別地価データ（住宅地標準地地価の平均価格）並びに夜間人口密度及び第三次産業従業人口密度データを用いて、地価を推定するモデルを例に挙げます。

JAGS及びR2jagsパッケージの利用方法、データのダウンロードについては、筆者のホームページを参照してください。
(http://web.sfc.keio.ac.jp/~maunz/wiki/)

## 2. データの準備

まず、パッケージとデータを読み込みましょう。今回は、spdep、R2jags、bayesmの３つのパッケージを使います。

```
library(spdep)
library(R2jags)
library(bayesm)
lph <- read.table("lph.csv, sep=",", header=T)
summary(lph)
```

次に、説明変数行列と被説明変数ベクトルなどを定義します。

```
# 変数の指定
# 被説明変数
y <- as.vector(lph$LPH)
# 地域数
n <- length(y)
# 地域セグメント数
m <- max(lph$seg)
# 説明変数
x <- as.matrix(cbind( lph$POPD,
lph$EMP3D, lph$seg, lph$ID ))
```

さらに、空間隣接行列、空間重み付け行列及び説明変数と被説明変数に空間重み付け行

列をかけた行列・ベクトルを、それぞれ作成します。

```
coords <-
as.matrix(cbind(lph$Easting,lph$Northing))
nb <- tri2nb(coords)
nb.mat <- nb2mat(nb, style="W")
yL <- nb.mat %*% y
xL <- nb.mat %*% x
```

## 3. 空間的自己回帰（SAR）モデル

SARモデルは、被説明変数の空間ラグを表現するモデルで、空間ラグモデルとも言われます。SARモデルをベイズ推定するために、次式のように定式化します。

$$y_i = \rho W_i y + \beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2 + \varepsilon_i$$
$$\varepsilon_i \sim N(\mu_i, \sigma^2)$$

ここで、サフィックス$i$は地区番号、$y_i$は地価、$x_{1i}$は夜間人口密度、$x_{2i}$は第三次産業従業人口密度、$W_i$は地区$i$に対する空間重み付け行列の要素ベクトル、$\rho$及び$\beta_0$、$\beta_1$、$\beta_2$は未知パラメータ、$\varepsilon_i$は誤差項を意味します。また、$\mu_i$と$\sigma^2$は、それぞれ誤差項の平均と分散を意味します。このモデルをベイズ推定するために、未知パラメータ及び誤差項の平均と分散については、例えば以下のような事前情報を与えます。

$$\beta_0 \sim N(0, 1\times10^{-6})$$
$$\beta_1 \sim N(0, 1\times10^{-6})$$
$$\beta_2 \sim N(0, 1\times10^{-6})$$
$$\rho \sim N(0, 1\times10^{-6})$$
$$\sigma^2 \sim IG(0.001, 0.001)$$

ここで、$IG(a, b)$は逆ガンマ関数を意味します。

分散の逆数を精度$\tau = 1/\sigma^2$として定義することにより、次のようなJAGSコード（slags.txt）を用いて、SARモデルをベイズ推定できます。

```
# spatial lag model
model{
for (i in 1 : n) {
y[i] ~ dnorm(mu[i], tau)
mu[i] <- rho*yL[i,1]+b0+b1*x[i,1]+b2*x[i,2]
}
b0 ~ dnorm(0.0, 1.0E-6)
b1 ~ dnorm(0.0, 1.0E-6)
b2 ~ dnorm(0.0, 1.0E-6)
tau ~ dgamma(0.001, 0.001)
rho ~ dnorm(0.0, 1.0E-6)
sigma <- 1/sqrt(tau)
}
```

以下のRコードを使って、モデルを推定してみましょう。

```
# データ
data <- list("n", "y", "x", "yL")
# MCMC初期値（事前情報）
in1 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
rho=0, tau=1)
in2 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
rho=0.5, tau=1)
in3 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
rho=1, tau=1)
inits <- list(in1,in2,in3)
# パラメータ
parameters <- c("b0", "b1", "b2",
"rho", "tau", "sigma")
# モデルファイル
model.file <- system.file(
package="R2jags", "model",
"slag1.txt")
```

```
# MCMC
slag.jags <- jags(data=data,
inits=inits, parameters,
n.iter=10000,  n.burnin=1000,
n.chains=3, model.file=model.file)
print(slag.jags, digits=3)
plot(slag.jags)
traceplot(slag.jags)
slag.fit <- update(slag.jags)
print(slag.fit, digits=3)
```

　モデル推定結果は、以下のようになります（図１、２）。

slag.jagsの結果

```
> slag.jags <- jags(data=data, inits=inits, parameters, n.iter=10000,
+ n.burnin=1000, n.chains=3, model.file=model.file)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 3570

  |**************************************************| 100%
> print(slag.jags, digits=3)
Inference for Bugs model at "/Library/Frameworks/R.framework/Resources/library/
R2jags/model/slag1.txt", fit using jags,
 3 chains, each with 10000 iterations (first 1000 discarded), n.thin = 9
 n.sims = 3000 iterations saved
             mean    sd     2.5%      25%      50%      75%    97.5%  Rhat n.eff
b0          1.419 0.518    0.386    1.070    1.416    1.758    2.463 1.001  3000
b1          0.372 0.179    0.022    0.257    0.372    0.486    0.716 1.002  1300
b2          1.380 0.124    1.141    1.298    1.380    1.463    1.624 1.001  3000
deviance 2480.637 3.370 2476.413 2478.265 2479.989 2482.286 2488.091 1.003  3000
rho         0.588 0.067    0.453    0.544    0.590    0.632    0.719 1.001  3000
sigma       7.446 0.280    6.925    7.253    7.436    7.624    8.029 1.001  3000
tau         0.018 0.001    0.016    0.017    0.018    0.019    0.021 1.001  3000

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 = 5.7 and DIC = 2486.3
DIC is an estimate of expected predictive error (lower deviance is better).
```

図１　SARモデルの推定結果（ベイズ推定）

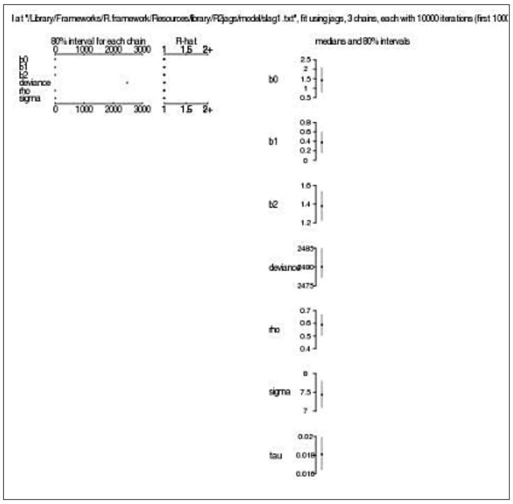
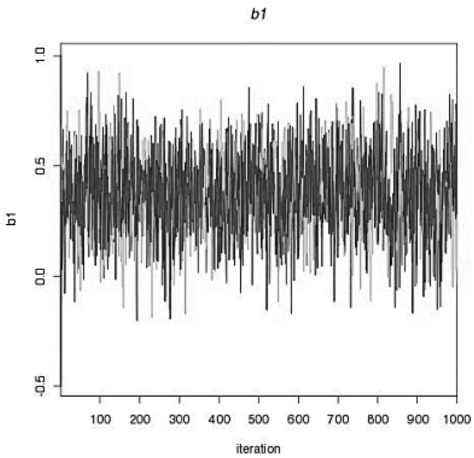



図２　$\beta_1$の推定結果




slag.fitの結果

```
> slag.fit <- update(slag.jags)
  |**************************************************| 100%
> print(slag.fit, digits=3)
Inference for Bugs model at "/Library/Frameworks/R.framework/Resources/library/
R2jags/model/slag1.txt", fit using jags,
 3 chains, each with 1000 iterations (first 10000 discarded)
 n.sims = 3000 iterations saved
             mean    sd     2.5%      25%      50%      75%    97.5%
b0          1.443 0.529    0.401    1.079    1.446    1.810    2.452
b1          0.374 0.176    0.025    0.255    0.375    0.493    0.713
b2          1.381 0.124    1.131    1.297    1.383    1.464    1.624
deviance 2480.664 3.105 2476.513 2478.408 2480.000 2482.306 2488.413
rho         0.586 0.067    0.461    0.541    0.587    0.631    0.725
sigma       7.450 0.272    6.947    7.258    7.436    7.639    7.994
tau         0.018 0.001    0.016    0.017    0.018    0.019    0.021
          Rhat n.eff
b0       1.001  3000
b1       1.006  1100
b2       1.003   980
deviance 1.004   580
rho      1.008   760
sigma    1.001  3000
tau      1.001  3000

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 = 4.8 and DIC = 2485.5
DIC is an estimate of expected predictive error (lower deviance is better).
```

## 4. 空間ダービンモデル

空間ダービンモデルは、説明変数と被説明変数の両方に空間ラグを取り入れたモデルです。空間ダービンモデルをベイズ推定するために、以下のようにモデルを定式化し、事前情報を与えることにします。

$$y_i = \rho W_i y + \beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2 + \rho W_i \beta_0 + \rho W_i x_1 \beta_1 + \rho W_i x_2 \beta_2 + \varepsilon_i$$

$$\varepsilon_i \sim N(\mu_i, \sigma^2)$$

$$\beta_0 \sim N(0, 1\times 10^{-6})$$

$$\beta_1 \sim N(0, 1\times 10^{-6})$$

$$\beta_2 \sim N(0, 1\times 10^{-6})$$

$$\rho \sim N(0, 1\times 10^{-6})$$

$$\sigma^2 \sim IG(0.001, 0.001)$$

空間ダービンモデルをベイズ推定するためのJAGSコードは次のようになります（sdm1.txt）。プログラムの中では、定数項となる$\beta_0$及び$\rho W_i \beta_0$を$\beta_0$にまとめ、さらに空間重み付け項$W_i x_1$と$W_i x_2$に対する未知パラメータを$g_1$及び$g_2$としています。

```
# spatial durbin model
model{
for (i in 1 : n) {
y[i] ~ dnorm(mu[i], tau)
mu[i] <- rho*yL[i,1]+b0+b1*x[i,1]+
b2*x[i,2]+g1*xL[i,1]+g2*xL[i,2]
}
b0 ~ dnorm(0.0, 1.0E-6)
b1 ~ dnorm(0.0, 1.0E-6)
b2 ~ dnorm(0.0, 1.0E-6)
g1 ~ dnorm(0.0, 1.0E-6)
g2 ~ dnorm(0.0, 1.0E-6)
tau ~ dgamma(0.001, 0.001)
rho ~ dnorm(0.0, 1.0E-6)
sigma <- 1/sqrt(tau)
}
```

以下のRコードを使って、モデルを推定してみましょう。

```
# JAGS変数設定
# データ
data <- list("n", "y", "x", "yL", "xL")
# MCMC初期値（事前情報）
in1 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
g1=model.lm$coefficients[2],
g2=model.lm$coefficients[3],
rho=0, tau=1)
in2 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
g1=model.lm$coefficients[2],
g2=model.lm$coefficients[3],
rho=0.5, tau=1)
in3 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
g1=model.lm$coefficients[2],
g2=model.lm$coefficients[3],
rho=1, tau=1)
inits <- list(in1,in2,in3)
# パラメータ
parameters <- c("b0", "b1", "b2",
"g1", "g2", "rho", "tau", "sigma")
# モデルファイル
model.file <- system.file(
package="R2jags", "model", "sdm1.txt")
# MCMC
sdm.jags <- jags(data=data,
inits=inits, parameters, n.iter=10000,
n.burnin=1000, n.chains=3,
model.file=model.file)
print(sdm.jags, digits=3)
plot(sdm.jags)
traceplot(sdm.jags)
sdm.fit <- update(sdm.jags)
print(sdm.fit, digits=3)
```

すると、次のようなモデル推定結果が得られます。

sdm.jagsの結果

```
> print(sdm.jags, digits=3)
Inference for Bugs model at "/Library/Frameworks/R.framework/Resources/library/
R2jags/model/sdm1.txt", fit using jags,
 3 chains, each with 10000 iterations (first 1000 discarded), n.thin = 9
 n.sims = 3000 iterations saved
            mean     sd    2.5%     25%     50%     75%   97.5%
b0         1.017  0.790  -0.148   0.645   1.034   1.423   2.206
b1        -0.531  0.282  -1.075  -0.707  -0.528  -0.351   0.000
b2         1.412  0.132   1.154   1.322   1.409   1.498   1.680
deviance 2463.786 17.650 2457.827 2460.480 2462.574 2465.236 2472.273
g1         1.587  0.365   0.915   1.328   1.581   1.835   2.288
g2         0.232  0.340  -0.433  -0.006   0.243   0.469   0.867
rho        0.387  0.123   0.161   0.307   0.387   0.468   0.617
sigma      7.279  0.428   6.749   7.091   7.261   7.442   7.835
tau        0.019  0.001   0.016   0.018   0.019   0.020   0.022
          Rhat n.eff
b0       1.019  3000
b1       1.002  1100
b2       1.001  3000
deviance 1.050  3000
g1       1.002  1300
g2       1.002  1900
rho      1.001  3000
sigma    1.008  2200
tau      1.008  2200

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 = 155.8 and DIC = 2619.6
DIC is an estimate of expected predictive error (lower deviance is better).
```

## 5. 空間誤差モデル

空間誤差モデルは、誤差項に空間ラグを取り入れたモデルです。空間誤差モデルをベイズ推定するために、以下のようにモデルを定式化し、事前情報を与えることにします。

$$y_i = X_i \beta + u_i$$
$$u_i = \lambda W_i u + \varepsilon_i$$
$$\varepsilon_i \sim N(\mu_i, \sigma^2)$$
$$\beta_0 \sim N(0, 1 \times 10^{-6})$$
$$\beta_1 \sim N(0, 1 \times 10^{-6})$$
$$\beta_2 \sim N(0, 1 \times 10^{-6})$$
$$\lambda \sim U(0, 1)$$
$$\sigma^2 \sim IG(0.001, 0.001)$$

空間誤差モデルをベイズ推定するために、次のようなJAGSコードを用います（sem1.txt）。

```
# spatial error model
model{
for (i in 1 : n)
{
y[i] ˜ dnorm(mu[i], tau)
mu[i] <- lambda*yL[i,1]+
b0*(1-lambda)+b1*x[i,1]+b2*x[i,2]
-b1*lambda*xL[i,1]-b2*lambda*xL[i,2]
}
b0 ˜ dnorm(0.0, 1.0E-6)
b1 ˜ dnorm(0.0, 1.0E-6)
b2 ˜ dnorm(0.0, 1.0E-6)
tau ˜ dgamma(0.001, 0.001)
lambda ˜ dunif(0,1)
sigma <- 1/sqrt(tau)
}
```

このモデルは、次のRコードを用いて推定できます。

```
# JAGS変数設定
# データ
data <- list("n", "y", "x", "yL", "xL")
# MCMC初期値（事前情報）
in1 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
lambda=0, tau=1)
in2 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
lambda=0.5, tau=1)
in3 <- list(
b0=model.lm$coefficients[1],
b1=model.lm$coefficients[2],
b2=model.lm$coefficients[3],
lambda=1, tau=1)
inits <- list(in1,in2,in3)
# パラメータ
parameters <- c("b0", "b1", "b2",
"lambda", "tau", "sigma")
# モデルファイル
model.file <- system.file(
package="R2jags", "model",
"sem1.txt")
# MCMC
sem.jags <- jags(data=data,
inits=inits, parameters,
n.iter=10000,  n.burnin=1000, n.chains=3,
model.file=model.file)
```

```
print(sem.jags, digits=3)
plot(sem.jags)
traceplot(sem.jags)
sem.fit <- update(sem.jags)
```

モデル推定結果は、以下のようになります。

sem.jagsの結果

```
> print(sem.jags, digits=3)
Inference for Bugs model at "/Library/Frameworks/R.framework/Resources/library/
R2jags/model/sem1.txt", fit using jags,
 3 chains, each with 10000 iterations (first 1000 discarded), n.thin = 9
 n.sims = 3000 iterations saved
             mean      sd      2.5%      25%      50%      75%    97.5%
b0        -12.336 421.919 -1070.984  -82.314    7.483   66.992  941.479
b1         -0.621   0.278    -1.157   -0.811   -0.614   -0.431   -0.076
b2          1.386   0.127     1.134    1.303    1.389    1.469    1.634
deviance 2487.105   2.857  2483.646 2484.996 2486.424 2488.434 2494.427
lambda      0.991   0.013     0.952    0.990    0.997    0.999    1.000
sigma       7.514   0.271     6.986    7.328    7.506    7.690    8.070
tau         0.018   0.001     0.015    0.017    0.018    0.019    0.020
          Rhat n.eff
b0       1.001  3000
b1       1.002  1500
b2       1.001  3000
deviance 1.001  2800
lambda   1.001  3000
sigma    1.001  2900
tau      1.001  2900

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 = 4.1 and DIC = 2491.2
DIC is an estimate of expected predictive error (lower deviance is better).
```

## 6. マルチレベルモデル

マルチレベルは、地域や個人ごとにモデルパラメータを推定する際などに用いられます。ここでは、次式のようにモデルを定式化します。

$$y_i = X_i \beta_i + u_i$$

$$\varepsilon_i \sim N(\mu_i, \sigma^2)$$

$$\beta_i \sim N(0, 1 \times 10^{-6})$$

$$\sigma^2 \sim IG(0.001, 0.001)$$

マルチレベルモデルをベイズ推定するのに、ここでは、baysmパッケージを用いることにします。Rコードは以下のようになります。

```
library(bayesm)
# MCMCの設定
R=10000
keep=1
# 事前分布を設定
reg=levels(factor(lph$JCODE))
nreg=length(reg)
nvar=3
# 説明変数2つ+定数項
# 変数を設定
regdata=NULL
for (j in 1:nreg) {
y=lph$LPH[lph$JCODE==reg[j]]
iota=c(rep(1,length(y)))
X=cbind(iota,
lph$POPD[lph$JCODE==reg[j]],
lph$EMP3D[lph$JCODE==reg[j]])
regdata[[j]]=list(y=y,X=X)}
Z=matrix(c(rep(1,nreg)),ncol=1)
Data1=list(regdata=regdata,Z=Z)
Mcmc1=list(R=R,keep=1)
set.seed(66)
# モデル推定
out1=rhierLinearModel(
Data=Data1,Mcmc=Mcmc1)
# 推定結果の表示
summary(out1$Deltadraw,
burnin=1000)
summary(out1$Vbetadraw,
burnin=1000)
summary(t(out1$betadraw[1,,]),
burnin=1000)
```

モデル推定結果は、以下のようになります。

out1の結果

```
> # 推定結果の表示
> summary(out1$Deltadraw, burnin=1000)
Summary of Posterior Marginal Distributions
Moments
  mean std dev num se rel eff sam size
1  1.2    0.16  0.010      36      243
2  2.2    0.19  0.015      59      150
3  1.7    0.53  0.045      66      134

Quantiles
  2.5%   5% 50% 95% 97.5%
1 0.86 0.92 1.2 1.5   1.5
2 1.88 1.94 2.2 2.6   2.6
3 0.60 0.79 1.7 2.5   2.6
   based on 9000 valid draws (burn-in=1000)
> summary(out1$Vbetadraw, burnin=1000)
Posterior Means of Std Deviations and Correlation Matrix
  Std Dev     1      2     3
1    0.79 1.000  0.015  0.21
2    0.73 0.015  1.000 -0.22
3    1.65 0.213 -0.225  1.00

Upper Triangle of Var-Cov Matrix
Summary of Posterior Marginal Distributions
Moments
       mean std dev num se rel eff sam size
1,1  0.6409    0.20 0.0122      33      273
1,2  0.0023    0.14 0.0092      42      209
2,2  0.5545    0.20 0.0149      53      170
1,3  0.2849    0.45 0.0370      62      145
2,3 -0.3771    0.49 0.0446      73      122
3,3  2.8755    1.37 0.1221      71      125
   based on 9000 valid draws (burn-in=1000)
> summary(t(out1$betadraw[1,,]), burnin=1000)
       V1               V2               V3
 Min.   :-1.975   Min.   :-0.4285   Min.   :-7.429
 1st Qu.: 1.294   1st Qu.: 2.0641   1st Qu.: 1.200
 Median : 1.861   Median : 2.5675   Median : 2.349
 Mean   : 1.872   Mean   : 2.5795   Mean   : 2.329
 3rd Qu.: 2.446   3rd Qu.: 3.0942   3rd Qu.: 3.481
 Max.   : 5.336   Max.   : 6.0575   Max.   :11.845
```