

パラメータ推定の理論と実践

BESTGUY

BEhavior Study for Transportation Graduate school, Univ. of Yamanashi

山梨大学
佐々木邦明

最尤推定法

- 点推定量を求める最もポピュラーな方法

$$L_n(\theta | x) = \prod_{i=1}^n f(x_i | \theta)$$

- 右上の式を θ の関数とみなしたものが尤度関数

データ(a, b)が得られたとき, 全体の平均がいくつとするのがよいか
⇔平均がいくつだったら(a, b)が得られやすいか?

- 尤度関数を最大化する θ の値を最尤推定量とするのが最尤推定法

選択モデルの場合, f が選択確率

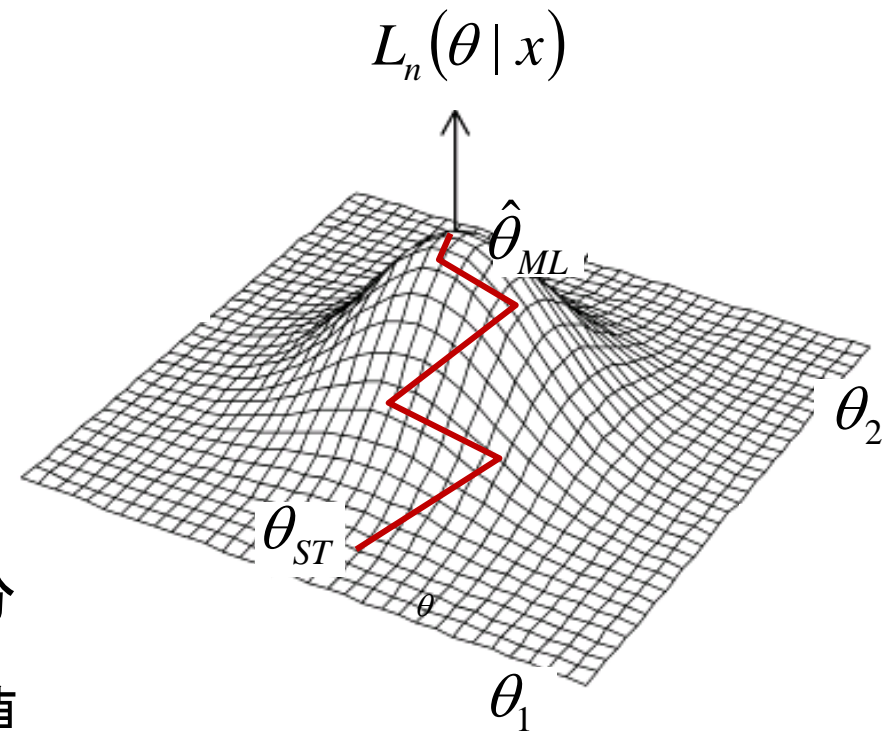


個人の選択確率を全員で掛け合わせる

$$\text{MaxLikelihood} = \max_{\theta \in \Theta} \prod_i P_i(x_i | \theta)$$

最大化アルゴリズムの考え方

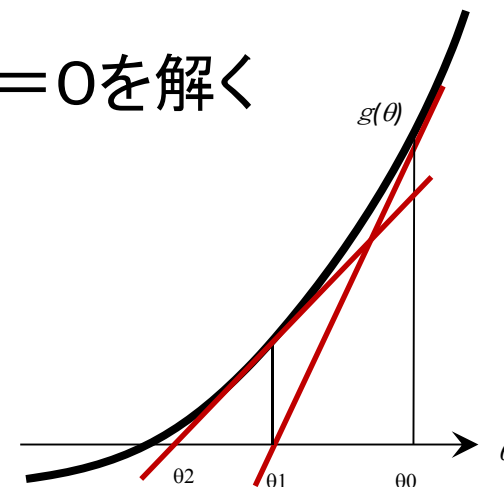
- 対数尤度関数の段階的な最大化
 1. 初期値を与える
 2. 初期値周りで勾配(1次微分)等を用いて次の推定値の方向を決める
 3. 初期値付近のステップサイズを1次微分, 2次微分を用いて適切に決めて次の推定値を決める
 4. 収束基準(尤度関数の一時微分ベクトル)を判定し, 収束していない場合は, 現在の値を初期値として2に進む



代表的な繰り返し計算法

尤度関数を最大化:尤度関数の一階微分=0を解く

- Newton-Raphson法
 - テイラー展開の1次近似を利用して進める
 - 解の収束が早い(ステップ数が少ない)
- 準Newton法(BFGS法)
 - ヘッセ行列を逐次近似する.



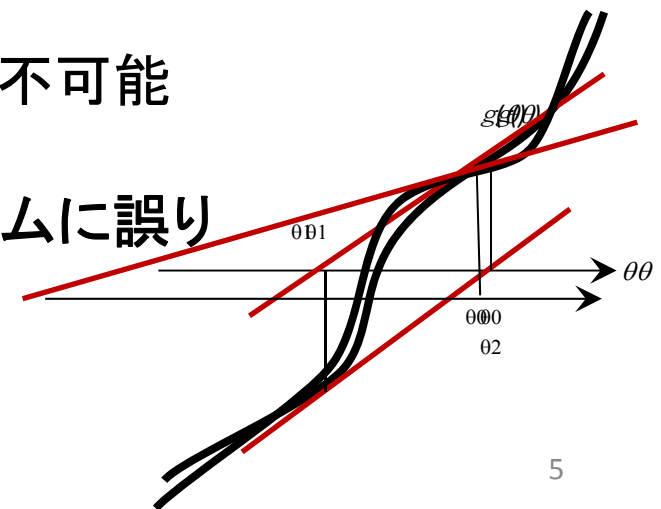
$$\theta_{n+1} = \theta_n - H^{-1} g_1$$

H:尤度関数の二階微分
ヘッセ行列

g: 尤度関数の一階微分

パラメータ推定がうまくいかない

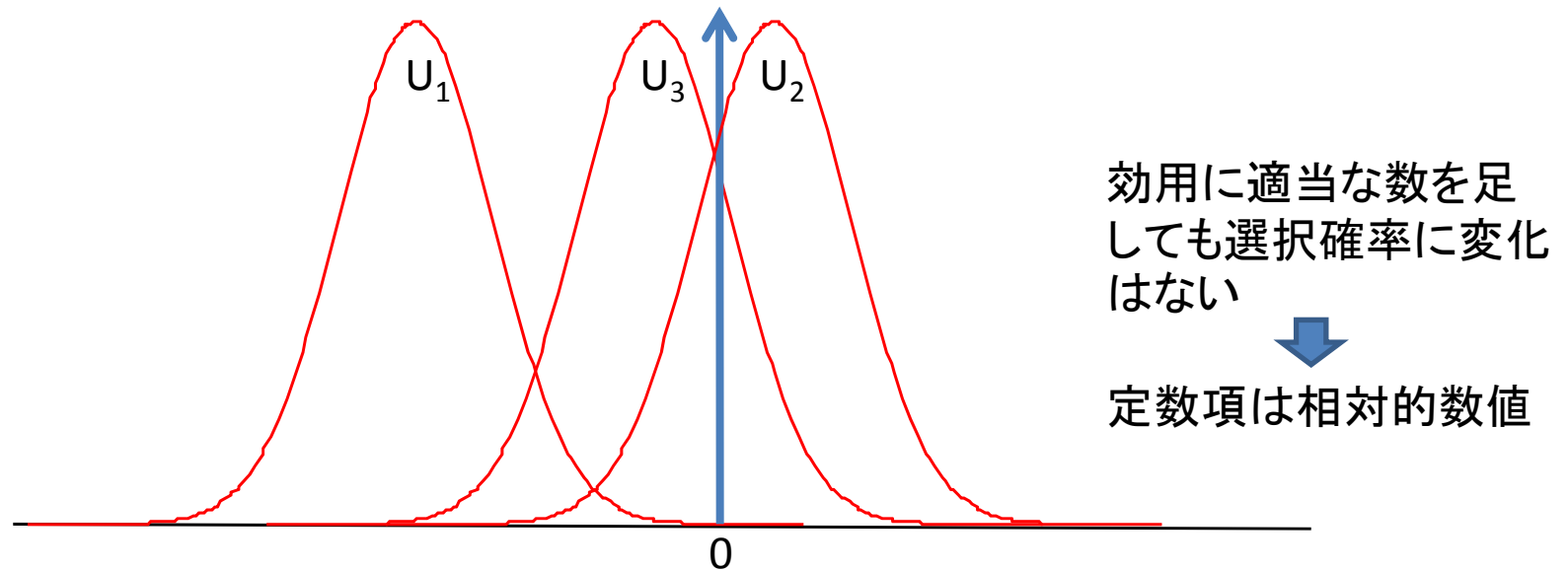
- 収束するとは ϑ_{n+1} と ϑ_n が同じになる
 - g_1 が0になる
 - 収束しない
 - 無限に繰り返す
 - ϑ_2 が計算不能
 - 局所最適解
 - 見かけ上の最大化
- H^{-1} が存在しない(計算できない)
 - 変数が完全相関
 - 変数が効用関数に影響していない
 - 関数の近似状況
 - 2次関数近似
 - 初期値の問題
 - そもそも推定不可能
 - 推定プログラムに誤り



そもそも推定不能

- 最大値において唯一解が求まらない可能性がある(最大値となるパラメータベクトルが無数にある) → Identification Problem

例



シミュレーションによる推定

シミュレーションによる尤度計算

- 手順

1. 誤差項の密度関数から選択肢の数の次元の(準)乱数を発生させる
2. この乱数を誤差の値として, 各代替案の効用値を計算(積分)する
3. 代替案*i*の効用値とその他の代替案の効用との値を比較し, それらの大小関係を1-0の変数*G*で記述する.
4. 1~3のステップを繰り返す. その反復回数を*R*とする.
5. シミュレーションされた確率は $P_i = \frac{1}{R} \sum_{r=1}^R G^r$ となり, この値は不偏推定量である.

効用を確定値にする

確定的に選択を決定

比率を確率に置き換える

これを尤度として最大になるようにパラメータをアップデートする

準乱数の例

- 代表的例としてHalton数列がある. その計算方法は素数 p に対して

$$s_{t+1} = \{s_t, s_t + 1/p^t, s_t + 2/p^t \cdots, s_t + (p-1)/p^t\}$$

例えば $p=3$ ならば, 初期値0として $1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9 \cdots$

- 多次元化
 - 数列の異なる素数 p を決めて, それぞれに応じて数列を作り多次元化する.
- 正規分布化
 - 数列を制約付きの乱数発生と同様の変換をして正規分布化

連続型シミュレーション

1. 誤差項の密度関数から選択肢の数の次元の乱数を発生させる
2. この乱数を誤差の値として、各代替案の効用値を計算する
3. この効用をロジット変換を行い、連続的な0-1変数に変換する。この変換後の値を s とする
4. 1~3のステップを繰り返す。その反復回数を R とする。
5. シミュレーションされた確率は $P_i = \frac{1}{R} \sum_{r=1}^R S^r$ となり、この値は不偏推定量である。

シミュレーションベースのパラメータ推定法

- シミュレーション尤度最大化 (MSL)
 - シミュレーションによって計算された確率を尤度として, 最大化を行う.
- 特性
 - サンプル数と乱数発生回数に依存する.
 - 乱数発生回数が十分大きいと一貫性や漸近的有効性を持ち解析積分と一緒の特性を持つ.
 - 乱数発生回数がサンプル数に対して小さく固定されると一貫性もない.

ベイズ推定

$$P(\theta | D) = \frac{P(D | \theta)P(\theta)}{P(D)}$$

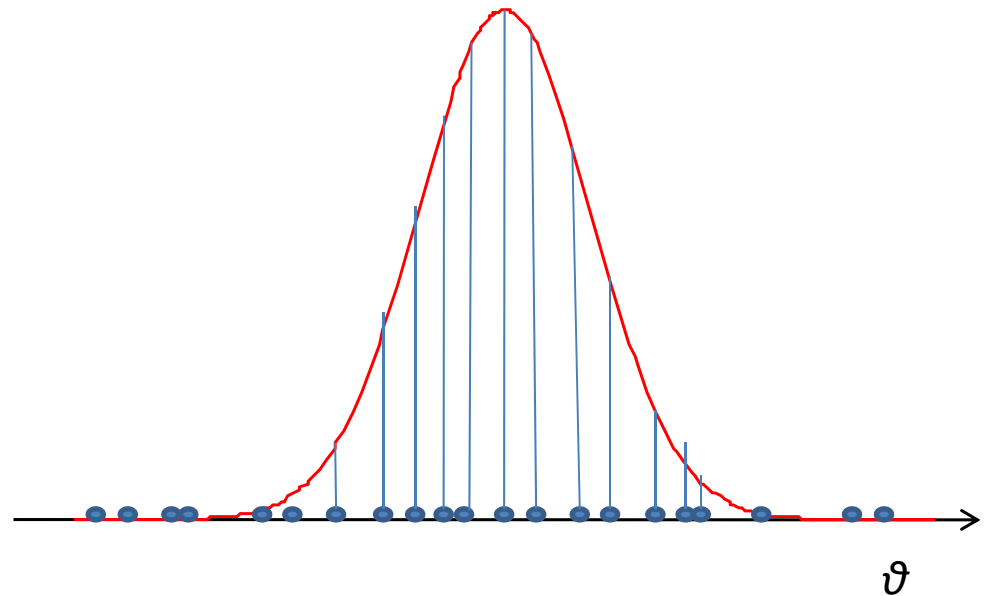
事後分布

事前確率

尤度

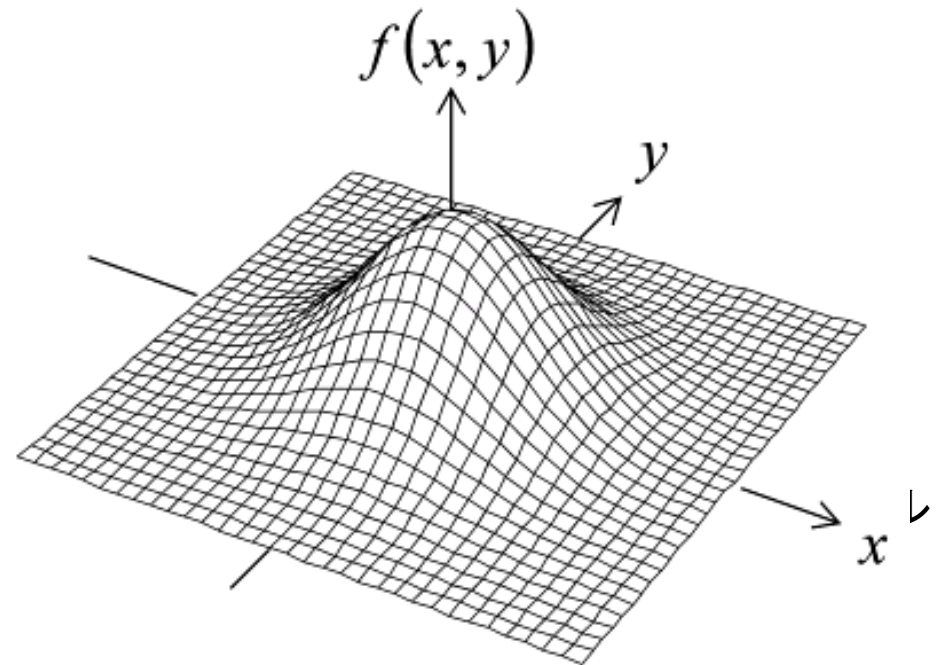
乱数による最大値計算

- ある変数に対して乱数を発生させ、その時の関数の値を調べる
 - 単純な乱数だと効率が悪い
 - 尤度関数の値の大きさを、次のパラメータにあたりをつける
 - 尤度分布に基づく乱数を発生できれば、その頻度がパラメータの分布になる



MCMC法による推定

- Gibbs Sampling
 - 一つを除いて残りのパラメータを固布を考える
 - あるパラメータの事後分布を求め、パラメータをサンプリングする
 - 同様に事後分布から順々にサンプル
- Metropolis-Hastings Sampling
 - あるパラメータに対する尤度を計算
 - パラメータをある方向(ランダムに)
 - 基本的には
 - 尤度が大きくなるようならその方向に
 - 尤度が小さくなるようなら、逆方向に
 - 実際は一様乱数との大小で採択を決める
- 前の状態に基づいて(Markov Chain)新しいパラメータをランダムにサンプリング(Monte Carlo)する

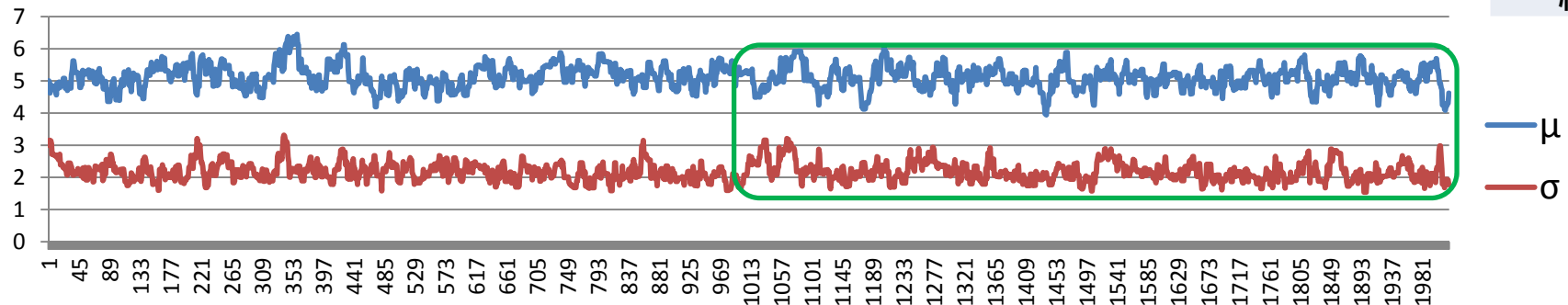


平均値も計算可能

- 例

- 平日一日の平均トリップ数が右の表のように与えられる
- 分散の事前分布を逆ガンマ, 平均値の事前分布を正規分布とする

メトロポリス法のMCMCで平均値と分散の分布を計算してみる



- この時 μ と σ の平均値は単純に平均で計算可能で5.01と2.17 (※1000まで廃棄)

ロジット・プロビットモデルの MCMC推定プログラム(兵藤2009)

```
• library(MCMCpack)
• ###データファイルの読み込み
• Dat<-read.csv("H:/2014/data.csv",header=TRUE)
• hh<-nrow(Dat) ##データ数:Data の行数を数える
• rtime <- Dat[, 6]/100; btime <- Dat[, 9]/100; ctime <- Dat[,12]/100
• rcost <- Dat[, 7]/100; bcost <- Dat[,10]/100; ccost <- Dat[,13]/100
• raged <- matrix(0,nrow=hh,ncol=1); baged <- 1*(Dat[,3]>=6); caged <- raged
• rcar <- matrix(0,nrow=hh,ncol=1); bcar <- rcar; ccar <- 1*(Dat[,4]>=2)
• ##選択結果
• ch <- matrix(0,nrow=hh,ncol=3)
• colnames(ch) <- c("1", "2", "3")
• for (i in 1:hh){
•   if (Dat[i, 5]==0) ch[i,1] <- -999
•   if (Dat[i, 2]==1) ch[i,1] <- 1
•   if (Dat[i, 8]==0) ch[i,2] <- -999
•   if (Dat[i, 2]==2) ch[i,2] <- 1
•   if (Dat[i,11]==0) ch[i,3] <- -999
•   if (Dat[i, 2]==3) ch[i,3] <- 1
• }
• post <- MCMCmnl(ch ~
•   choicevar(rtime, "time", "1") +
•   choicevar(btime, "time", "2") +
•   choicevar(ctime, "time", "3") +
•   choicevar(rcost, "cost", "1") +
•   choicevar(bcost, "cost", "2") +
•   choicevar(ccost, "cost", "3") +
•   choicevar(raged, "aged", "1") +
•   choicevar(baged, "aged", "2") +
•   choicevar(caged, "aged", "3") +
•   choicevar(rcar, "car", "1") +
•   choicevar(bcar, "car", "2") +
•   choicevar(ccar, "car", "3"),
•   baseline="3", burnin=1000,
•   mcmc.method="RWM",
•   b0=0, B0=0, seed=2348,
•   verbose=1000, mcmc=10000, B=0.001)
• plot(post)
• summary(post)
```

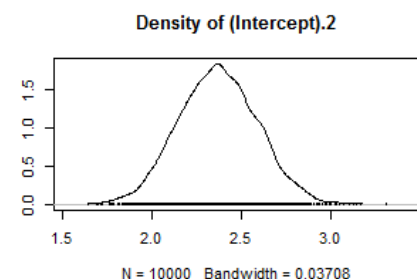
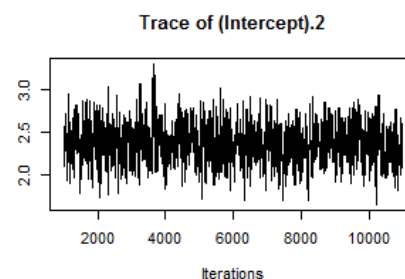
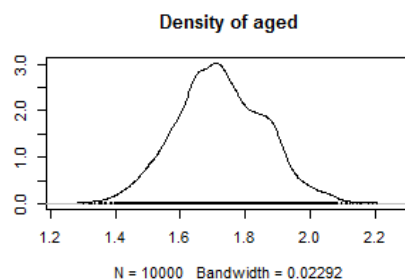
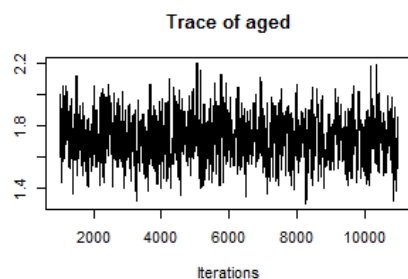
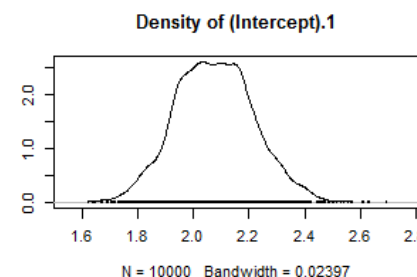
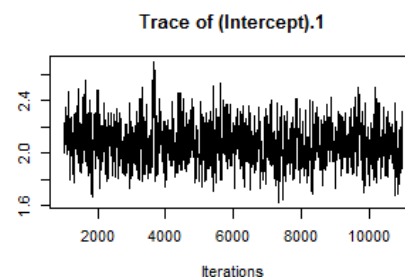
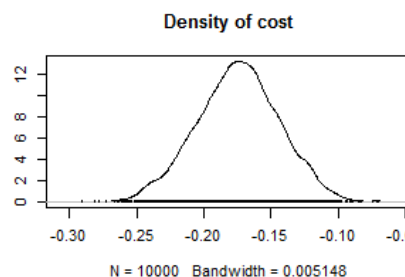
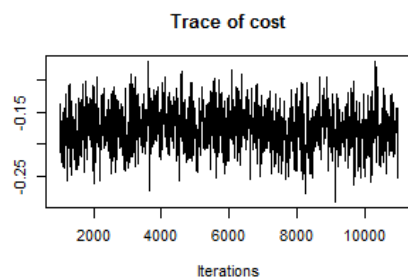
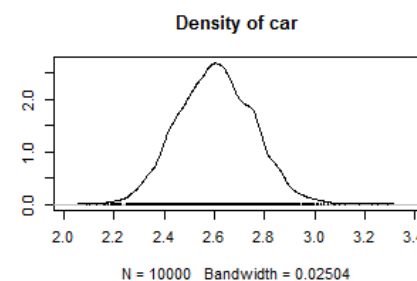
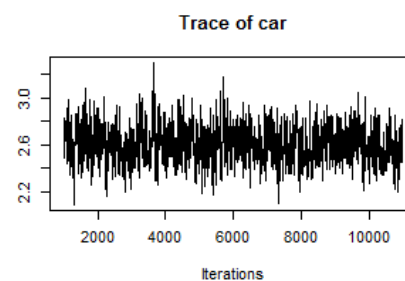
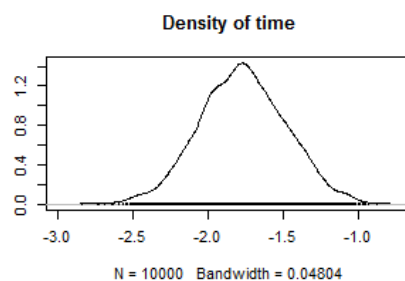
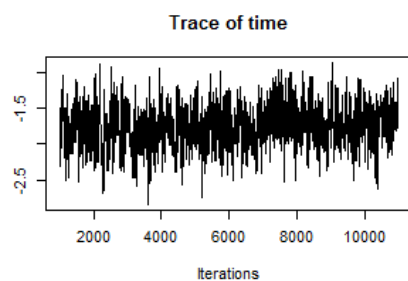
ロジット

```
library(bayesm)
###データファイルの読み込み

Dat<-read.csv("h:/2014/data.csv",header=TRUE)
hh<-nrow(Dat) ##データ数:Data の行数を数える
alt <- 3
rtime <- Dat[, 6]/100; btime <- Dat[, 9]/100; ctime <- Dat[,12]/100
rcost <- Dat[, 7]/100; bcost <- Dat[,10]/100; ccost <- Dat[,13]/100
raged <- matrix(0,nrow=hh,ncol=1); baged <- 1*(Dat[,3]>=6); caged <- raged
rcar <- matrix(0,nrow=hh,ncol=1); bcar <- rcar; ccar <- 1*(Dat[,4]>=2)
cres <- Dat[,2]
na <- 4
Xa <- cbind(rtime,btime,ctime,rcost,bcost,ccost,raged,baged,caged,rcar,bcar,ccar)
nd <- 0
X <- createX(alt, na=na, nd=nd, Xa=Xa, Xd=NULL, DIFF=TRUE, base=3)
dat1 <- list(p=alt, y=cres, X=X)
mcmc1 <- list(R=5000,k=1)
res1 <- rmnpGibbs(Data=dat1, Mcmc=mcmc1)
plot(res1$betadraw)
plot(res1$sigmadraw)
```

プロビット

推定例 (MHアルゴリズムでロジット)



ベイズ推定のメリットデメリット

- メリット
 - 解析的にはパラメータが求まらない複雑なモデルもパラメータ分布が求まる
 - 例: パネルデータの個人モデルを考慮した階層モデル
 - パラメータの分布がわかる
 - 多峰性の分布になったならばモデル構造を考え直す
- デメリット
 - 時間がかかる
 - MNLの推定 MCMC:15秒 最尤推定:7秒
 - パラメータの分布が収束しない場合もある

参考文献等

- 入門ベイズ統計学 松原望
- ベイズモデリングによるマーケティング分析 照井伸彦
- Rによる離散選択モデルの推定方法メモ 兵藤哲朗