# 行動モデルの推定法

山梨大学
佐々木邦明

# 最尤推定法

- 点推定量を求める一般的な方法
- 右上の式をθの関数とみなしたものが尤度関数
- 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

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

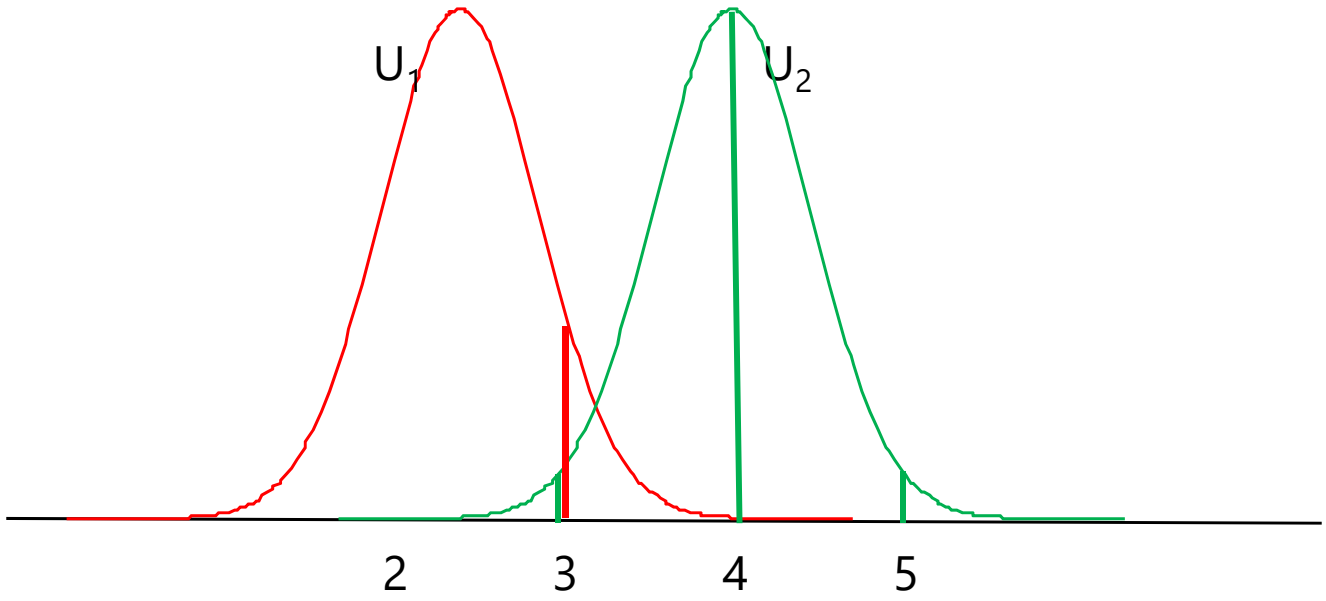
平均値の推定を例にすると

データ(**x：3,5,4**)が得られたとき，
平均をいくつとするのがよいか？

⇒平均がいくつの分布だったら
データ(**x：3,5,4**)が得られやすいか？

# 平均値の例の図

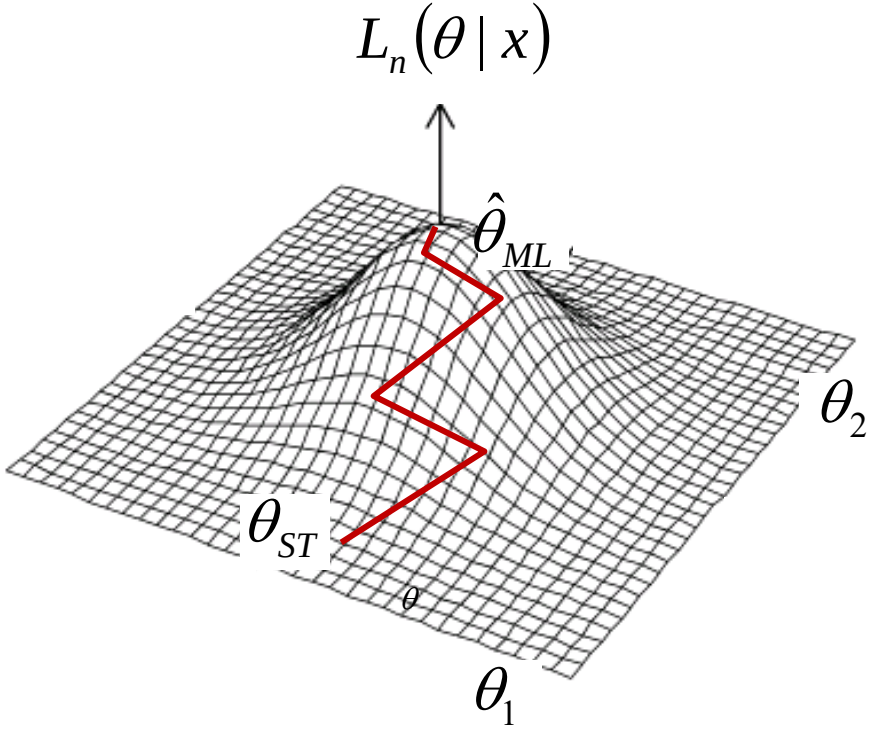
どちらのほうが確率（密度：縦棒）の掛け算が大きいか？



# 最大化アルゴリズムの考え方

**周りが見えない中で，近傍の情報から頂点を目指す**

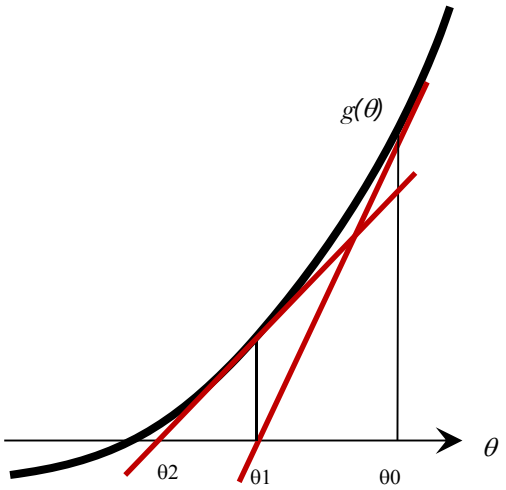
- 対数尤度関数の段階的な最大化
  1. 初期値を与える
  2. 初期値周りで勾配(1次微分)等を用いて次の推定値の方向を決める
  3. 初期値付近で１次微分，２次微分を用いて適切に次の点を決めて推定値を得る
  4. 収束基準(一時微分ベクトル)で判定し，収束していない場合は，現在の値を初期値として2に進む



# 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
- 準Newton法（BFGS，L-BFGS法）
  - ヘッセ行列を，パラメータの差分と一回微分の差分を用いて逐次近似する.
  - L-BFGSはヘッセ行列の更新式を展開して，初期値と差分の関数和で表す．複雑な場合にメモリの節約などが可能
- 焼きなまし法（SANN）
  - 適当にパラメータを遷移しつつ，関数が大きくなる遷移を選ぶ.
  - 悪化する遷移もある確率で許容し，**局所解**を避ける



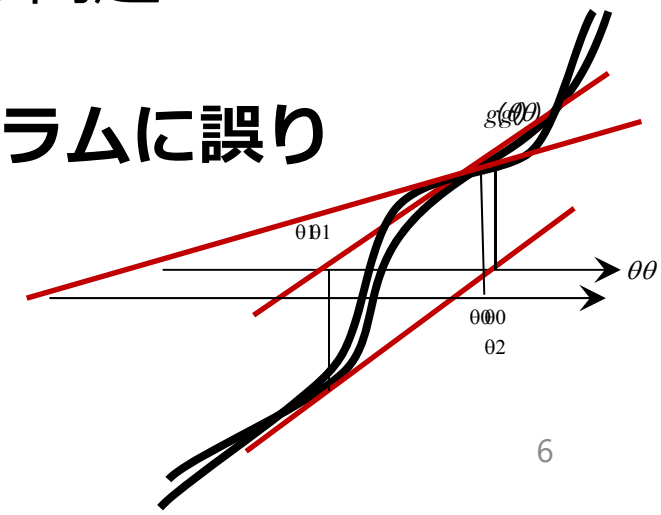

$$\theta_{n+1} = \theta_n - H^{-1} g_1$$

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$\theta_{n+1}$と$\theta_n$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
  - $\theta_2$が計算不能
- 局所最適解
  - 見かけ上の最大化

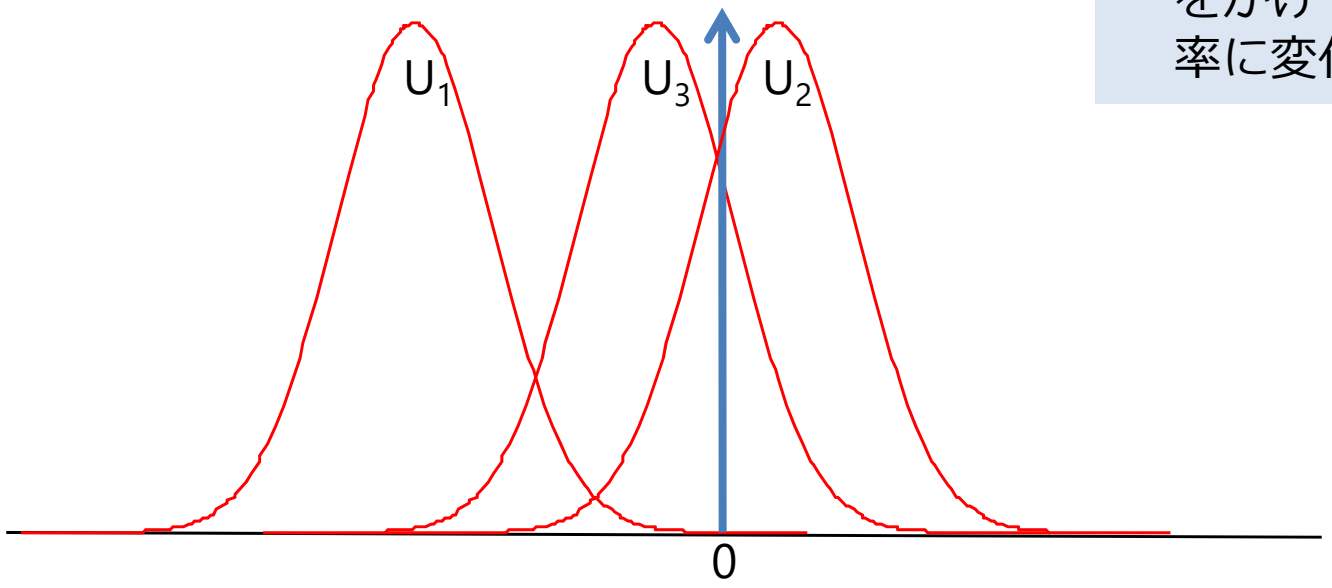
- $H^{-1}$が計算できない
  - 変数が完全相関
  - 変数が効用関数に影響しない式形
- 関数の近似状況
  - 2次関数近似
  - 初期値の問題
- **推定プログラムに誤り**



# Identification Problem

- 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）
  - 意思決定者間では異なるが，選択肢間では異ならない変数
  - 選択肢間では異なるが，意思決定者間で異ならない変数
  - 異なるモデル間でのパラメータの直接比較は無意味

例




- 効用に適当な数を足しても選択確率に変化はない
- 効用に適当な倍率をかけても選択確率に変化がない

# 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

- 準乱数の例としてHalton数列がある.
  - 計算方法は素数pに対して

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

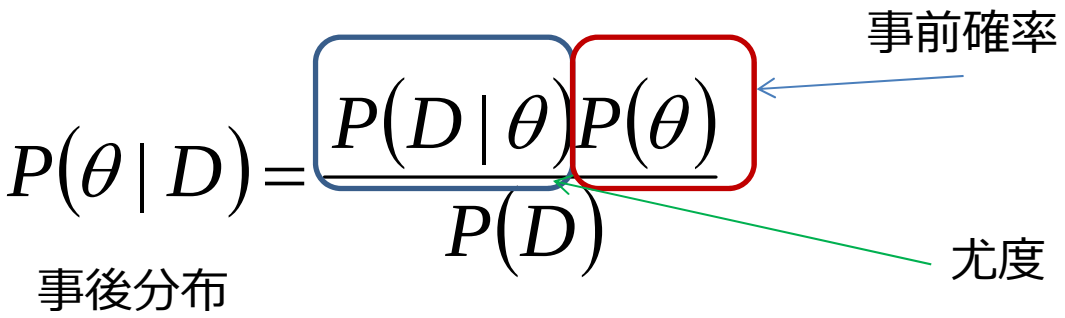
例えばp=3ならば，初期値0として 1 /3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9・・・

- 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
  - 数列を制約つきの乱数発生と同様の変換をして正規分布化

# 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致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致性もない.

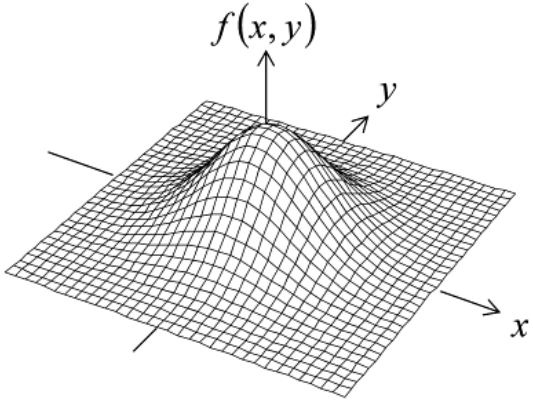
# ベイズ推定



# 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定して条件付き分布を考える
  - あるパラメータの事後分布を求め，その分布から次の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リングする

- Metropolis-Hastings Sampling
  - あるパラメータをある方向（ランダムに）に変え
  - パラメータに対する尤度を計算する
    - 基本的には
      - 尤度が大きくなるようならその方向を採択
      - 尤度が小さくなるようなら、逆方向にパラメータを変え



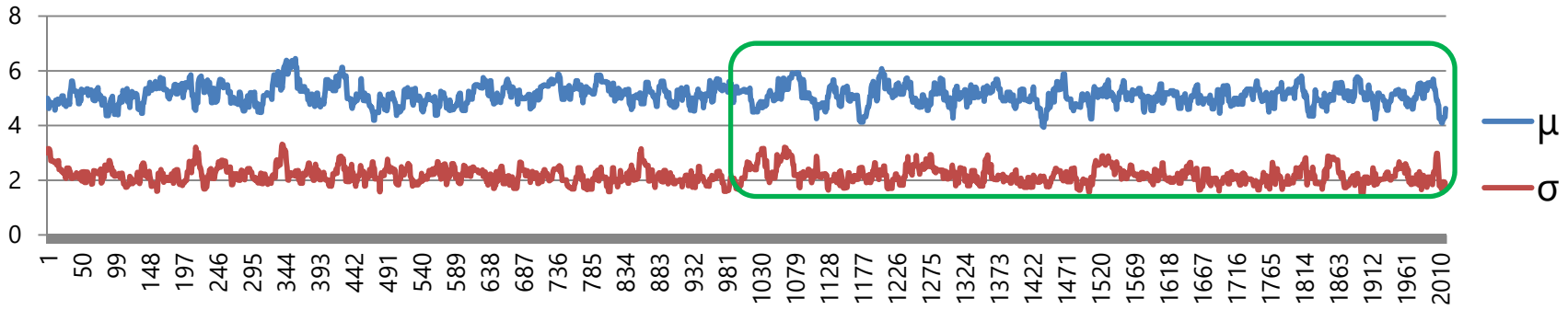

- 前の状態に基づいて（Markov Chain）新しいパラメータをランダムにサンプリング（Monte Carlo）する

# 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

| |
|---|
| 6.0 |
| 8.1 |
| 7.2 |
| 3.9 |
| 4.5 |
| 2.5 |
| 6.6 |
| 4.1 |
| 2.0 |
| 5.5 |




- 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2009参照）

```
 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)
```

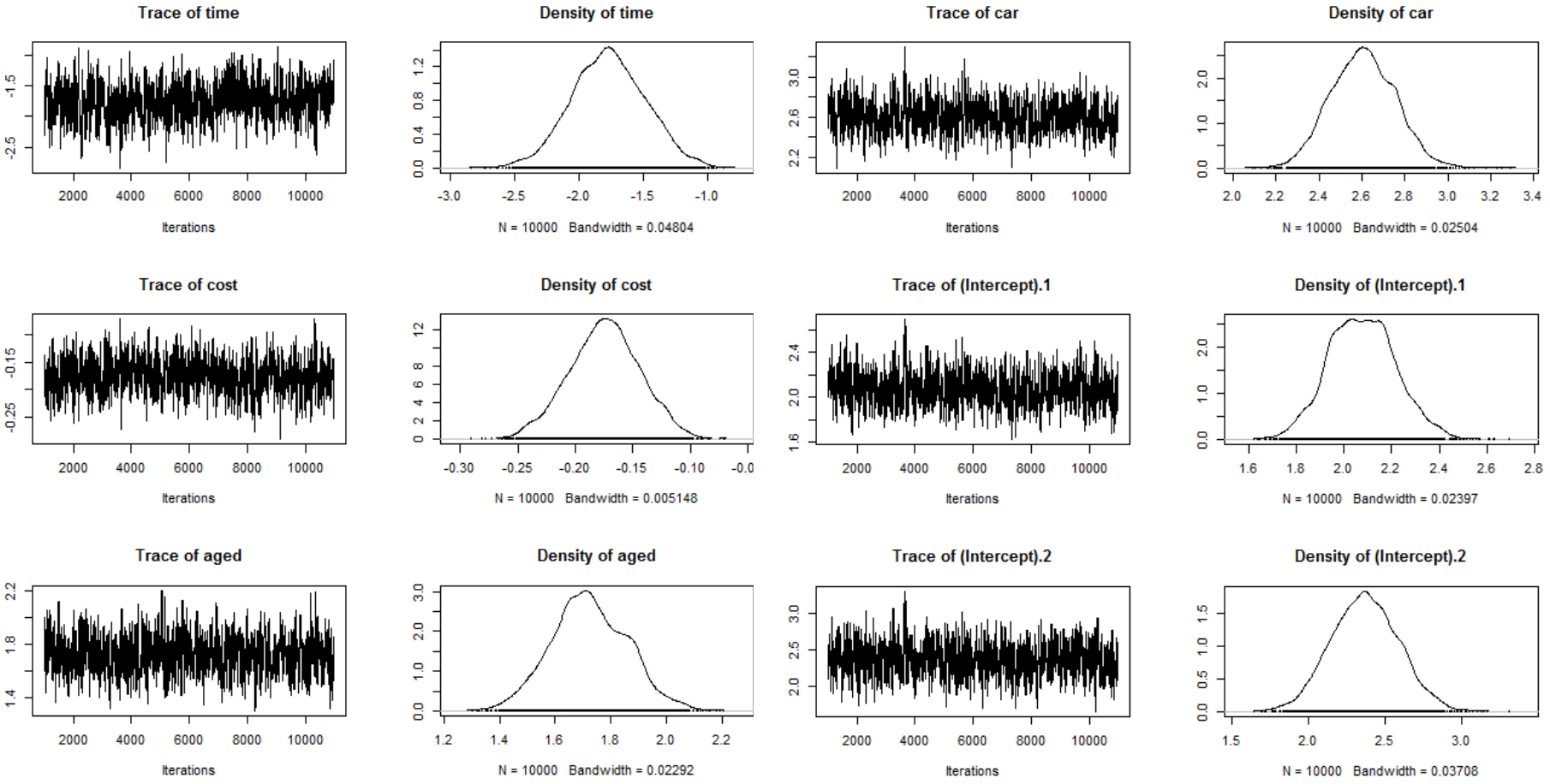
ロジット

```
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・ロジット）




兵藤　2009を用いた

# MCMC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やんわりわかる
    - 多峰性の分布になったなら，モデル構造を考え直す
- デメリット
  - 時間がかかる
    - MNLの推定例　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

# 政策シミュレーション

# 政策シミュレーション

- 効用関数を推定し，そのパラメータを用いて，変数の変化による選択の変化を見る

1. 選択データより効用関数のパラメータを推定する
2. その時の個人の選択確率を記録しておく
3. 政策に応じて，推定に用いた変数を変更する（例：料金を下げる）
4. パラメータをそのままに，変数を変えたデータセットで選択確率を求める
    1. 個人の確率から平均的な選択確率を求める
    2. 誤差項に従う乱数を発生し，最大確率の選択肢を記録 → これを繰り返して確率分布を得る

~~変数を変化させてパラメータを再推定する~~

# 参考文献


- 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