# Rによる項目反応理論
# 練習問題略解

加藤健太郎・山田剛史・川端一光

2014 年 7 月 17 日

# 第 2 章

1. ```
   > log(exp(1))
   [1] 1
   > log10(1000)
   [1] 3
   > log10(64)
   [1] 1.80618
   > 3*log10(4)
   [1] 1.80618
   > log2(6)
   [1] 2.584963
   > log2(2)+log2(3)
   [1] 2.584963
   > 1/(1+1.7*1.0*exp(1.0-0.5))
   [1] 0.2629623
   ```

2. ```
   > icc <- function(a, b, theta) {
   +   prob <- 1/(1+exp(-1.7*a*(theta-b)))
   +   prob
   + }
   > icc(a=0.5, b=1.0, theta=1.0)
   [1] 0.5
   > icc(a=0.5, b=1.0, theta=1.5)
   [1] 0.6046791
   > # 複数の項目についてまとめて正答確率を求めることもできる。
   > # 例えば，項目 3 と 4 をまとめると，
   > icc(a=c(1.0, 1.0), b=c(1.0, -0.5), theta=1.5)
   [1] 0.7005671 0.9677045
   > icc(a=1.0, b=-0.5,theta=-0.5)
   [1] 0.5
   ```

3. ```
   > test3 <- read.csv("testdata.csv")
   > test3
       student English Japanese Mathematics
   1  Nakamura      44       59          40
   2   Imamiya      64       73          34
   3  Hasegawa      55       61          44
   4  Uchikawa      41       53          37
   5  Yanagita      51       66          44
   6   Matsuda      59       67          50
   7     Egawa      51       46          39
   8     Honda      48       64          42
   9  Hosokawa      57       55          38
   10   Settsu      30       55          33
   > mean(test3$English) # 平均
   [1] 50
   > median(test3$English) # 中央値
   [1] 51
   > max(test3$English) # 最大値
   [1] 64
   > min(test3$English) # 最小値
   [1] 30
   > # summary() を使うと 3 科目一度に計算できる
   > summary(test3)
        student     English        Japanese     Mathematics
   ```
   ```

```
 Egawa   :1   Min.   :30.0   Min.   :46.0   Min.   :33.00
 Hasegawa:1   1st Qu.:45.0   1st Qu.:55.0   1st Qu.:37.25
 Honda   :1   Median :51.0   Median :60.0   Median :39.50
 Hosokawa:1   Mean   :50.0   Mean   :59.9   Mean   :40.10
 Imamiya :1   3rd Qu.:56.5   3rd Qu.:65.5   3rd Qu.:43.50
 Matsuda :1   Max.   :64.0   Max.   :73.0   Max.   :50.00
 (Other) :4
> var(test3$English) # 不偏分散
[1] 97.11111
> varp <- function(x) {
+   sample.variance <- var(x)*(length(x)-1)/length(x)
+   sample.variance
+ }
> varp(test3$English) # 標本分散
[1] 87.4
> cor(test3$English,test3$Japanese) # 英語と国語の相関
[1] 0.522804
> # 次のスクリプトで，データフレーム test3 から 1 列目の変数を取り除く
> test3b <- test3[,-1]
> test3b
   English Japanese Mathematics
1       44       59          40
2       64       73          34
3       55       61          44
4       41       53          37
5       51       66          44
6       59       67          50
7       51       46          39
8       48       64          42
9       57       55          38
10      30       55          33
> cor(test3b) # 3 科目間の相関
             English  Japanese Mathematics
English     1.0000000 0.5228040   0.3928464
Japanese    0.5228040 1.0000000   0.2962313
Mathematics 0.3928464 0.2962313   1.0000000
```

4. 省略

## 第 3 章

1. (a) 偏差値とは，標準得点の一種である。$z$ 得点を 10 倍し，50 を加えることで計算される。偏差値は平均 50，標準偏差 10 の得点である。

   (b) CTT における項目困難度は，項目に対する正答率として定義される。

   (c) CTT における項目識別力は，項目得点とテスト得点の相関係数として定義されることが一般的である。I-T 相関（item-total correlation），あるいは点双列相関係数として求められる。

2. 本文を参照して下さい。

3. 本文を参照して下さい。

4. あ：項目間の相関，い：項目数，う：内的整合性，え：内部一貫性

## 第 4 章

1. 項目特性曲線は，個々の項目の特徴を表すものであり，横軸に受験者の能力 $\theta$ を，縦軸にその項目に対する正答確率を取ったもの。ある項目の正答確率を受験者の能力パラメタ $\theta$ の関数として表現する。第 1 章あるいは 4.1 節を参照のこと。

2. (a) 1PLM

```
curve(1/(1+exp(-(x+0.5))),-4,4)     # b=-0.5
curve(1/(1+exp(-(x-0.5))),add=TRUE) # b=0.5
```

   (b) 2PLM

```
curve(1/(1+exp(-1.7*0.5*(x+0.5))),-4,4)     # a=0.5, b=-0.5
curve(1/(1+exp(-1.7*1.2*(x-0.5))),add=TRUE) # a=1.2, b=0.5
```

   (c) 3PLM

```
curve(0.2+((1-0.2)/(1+exp(-1.7*0.5*(x+0.5)))),-4,4)       # a=0.5, b=-0.5, c=0.2
curve(0.25+((1-0.25)/(1+exp(-1.7*1.2*(x-0.5)))),add=TRUE) # a=1.2, b=0.5,  c=0.25
```

   (d)
```
> # 3PLM a=0.5, b=-0.5, c=0.2
> 0.2+((1-0.2)/(1+exp(-1.7*0.5*(-0.5+0.5)))) # theta = -0.5
[1] 0.6
> 0.2+((1-0.2)/(1+exp(-1.7*0.5*(1.5+0.5)))) # theta = 1.5
[1] 0.8764278
```

3. ラッシュモデルについては，4.6.2 項を参照のこと。

4. テスト特性曲線とは，テストに含まれる項目の ICC を合計したもので，能力 $\theta$ に対応する期待正答数を表す。4.7 節を参照のこと。

## 第 5 章

1. 3PLM

```
icc3PL <- function(a, b, c, theta){
  prob <- c + (1-c)/(1+exp(-1.7*a*(theta-b)))
  prob
}
```

2.
```
> icc3PL <- function(a, b, c, theta){
+   prob <- c + (1-c)/(1+exp(-1.7*a*(theta-b)))
+   prob
+ }
> (Q1 <- 1 - icc3PL(1.27, 1.19, 0.10, 0))
[1] 0.8359682
> (Q2 <- 1 - icc3PL(1.34, 0.59, 0.15, 0))
[1] 0.6741777
> (Q3 <- 1 - icc3PL(1.14, 0.15, 0.15, 0))
[1] 0.4863424
> (P4 <- icc3PL(1.00, -0.59, 0.20, 0))
[1] 0.7853184
```

```
> (P5 <- icc3PL(0.67, -2.00, 0.01, 0))
[1] 0.9079682
> Q1*Q2*Q3*P4*P5
[1] 0.1954441
```

3. 
```
x <- seq(-3,3,0.1)
Q1 <- 1 - icc3PL(1.27, 1.19, 0.10, x)
Q2 <- 1 - icc3PL(1.34, 0.59, 0.15, x)
Q3 <- 1 - icc3PL(1.14, 0.15, 0.15, x)
P4 <- icc3PL(1.00, -0.59, 0.20, x)
P5 <- icc3PL(0.67, -2.00, 0.01, x)
like <- Q1*Q2*Q3*P4*P5
plot(x,like,type="l")
```

4. 
```
l.like <- log(like)
plot(x,l.like,type="l")
```

5. 
```
> x <- seq(-3,3,0.1)
> Q1 <- 1 - icc3PL(1.27, 1.19, 0.10, x)
> Q2 <- 1 - icc3PL(1.34, 0.59, 0.15, x)
> Q3 <- 1 - icc3PL(1.14, 0.15, 0.15, x)
> P4 <- icc3PL(1.00, -0.59, 0.20, x)
> P5 <- icc3PL(0.67, -2.00, 0.01, x)
> (like <- Q1*Q2*Q3*P4*P5)
 [1] 3.455889e-02 3.790111e-02 4.154613e-02 4.552181e-02
 [5] 4.985999e-02 5.459720e-02 5.977524e-02 6.544159e-02
 [9] 7.164932e-02 7.845650e-02 8.592451e-02 9.411509e-02
[13] 1.030856e-01 1.128822e-01 1.235304e-01 1.350231e-01
[17] 1.473061e-01 1.602616e-01 1.736913e-01 1.873014e-01
[21] 2.006917e-01 2.133527e-01 2.246756e-01 2.339766e-01
[25] 2.405403e-01 2.436796e-01 2.428098e-01 2.375303e-01
[29] 2.277007e-01 2.134987e-01 1.954441e-01 1.743781e-01
[33] 1.513919e-01 1.277122e-01 1.045591e-01 8.300417e-02
[37] 6.385660e-02 4.759957e-02 3.438639e-02 2.408906e-02
[41] 1.638044e-02 1.082600e-02 6.965239e-03 4.370362e-03
[45] 2.679640e-03 1.608888e-03 9.479822e-04 5.493242e-04
[49] 3.136973e-04 1.768867e-04 9.866555e-05 5.452930e-05
[53] 2.990318e-05 1.629212e-05 8.828460e-06 4.762594e-06
[57] 2.559733e-06 1.371590e-06 7.331128e-07 3.910487e-07
[61] 2.082413e-07
```

6. 
```
> max(like)
[1] 0.2436796
> x[which.max(like)]
[1] -0.5
```

より，$\hat{\theta} = -0.5$ と求められる。

7. $\hat{\theta} = 1.1$（受験者 1），$\hat{\theta} = 0.0$（受験者 2），$\hat{\theta} = 1.0$（受験者 4），$\hat{\theta} = -1.5$（受験者 5）

## 第 6 章

1. 6.1 節および 6.2 節を参考にまとめるとよい。表 6.1 も参照のこと。

2. (a) IIF を描画

```
iif3PL <- function(a, b, c, theta) {
  p <- c + (1-c)/(1+exp(-1.7*a*(theta-b)))
  q <- 1-p
  p.prime <- 1.7*a*q*(p-c)/(1-c)
  iif <- p.prime^2/(p*q)
  iif
}
curve(iif3PL(1.27, 1.19, 0.10, x), -4,4)
curve(iif3PL(1.34, 0.59, 0.15, x), add=TRUE)
curve(iif3PL(1.14, 0.15, 0.15, x), add=TRUE)
curve(iif3PL(1.00,-0.59, 0.20, x), add=TRUE)
curve(iif3PL(0.67,-2.00, 0.01, x), add=TRUE)
```

(b) TIF を描画

```
curve((iif3PL(1.27,  1.19, 0.10, x)
     + iif3PL(1.34,  0.59, 0.15, x)
     + iif3PL(1.14,  0.15, 0.15, x)
     + iif3PL(1.00, -0.59, 0.20, x)
     + iif3PL(0.67, -2.00, 0.01, x)),-4,4)
```

(c) 一番項目困難度の値が低い項目 5（$b = -2.00$）を除いた，項目 1 から項目 4 でテストを構成するとよい。

## 第 7 章

1. 測定の一次元性が成り立っていれば，必然的に局所独立性が成り立つことになる。局所独立性が成り立つ場合，能力パラメタや項目パラメタの推定に必要な尤度関数が，ICC やその補確率の積という単純な形で表されることになり，計算上扱いやすくなる。また，テスト情報関数も，項目情報関数の和として容易に求められる。

2. 【例】項目連鎖（前の項目への解答を利用しなければ解けない項目）や文脈効果（同じ刺激に対して複数の項目に解答する），測定の多次元性（複数の能力を同時に測定している項目に一次元モデルを当てはめた場合），測定対象とは関係ない受験者（あるいは採点者）特性の影響，などの具体例を挙げられるとよい。

   【対処法】局所依存している項目の一方（一部）を分析や採点から除外するのが最も単純な対処法であるが，それらの項目をまとめて採点して多値型項目と見なして多値型 IRT モデルを適用したり，文脈効果が顕著な場合はテストレット IRT モデル，測定対象以外の受験者特性が関与していると考えられる場合は多次元 IRT モデルなど，局所依存性を許容する特別な IRT モデルを適用することも考えられる。測定したい能力に関係のない受験者の要因が反応傾向に影響することを避けるため，テストの実施条件を統制することも重要である。

## 第 8 章

1. 
```
## ------------------------------------------------------------
## 能力パラメタの最尤推定
## ------------------------------------------------------------
## 反応パタン（4 項目× 5 名分）
```

```
u.all <- matrix(c(0,0,0,1,
                  0,1,0,1,
                  1,1,1,0,
                  1,0,1,0,
                  1,0,1,1),ncol=4,byrow=TRUE)

## 項目パラメタ
a <- c(0.7880079,2.0406337,0.6121101,1.0210670) # 識別力
b <- c(1.45240645,0.33258311,0.07252222,-1.13507144) # 困難度

## 関数の定義
icc2PL <- function(a,b,theta){
  1/(1+exp(-1.7*a*(theta-b)))
}

LL <- function(u,a,b,theta){ # 対数尤度関数
  sum(u*log(icc2PL(a,b,theta)) + (1-u)*log(1-icc2PL(a,b,theta)))
}

dLL <- function(u,a,b,theta){ # 一次導関数
  1.7*sum(a*(u-icc2PL(a,b,theta)))
}

ddLL <- function(a,b,theta){ # 二次導関数
  -1.7^2*sum(a^2*icc2PL(a,b,theta) * (1-icc2PL(a,b,theta)))
}

## 初期設定
ni <- nrow(u.all) # 受験者数 (I)
e <- 0.0001 # 収束基準
theta.mle <- numeric(ni) # 能力パラメタ推定値格納用

## 最適化計算
for(i in 1:ni){ # 受験者ごとにループ
  u <- u.all[i,] # i 番目の受験者の反応ベクトル
  t0 <- log(sum(u)/(length(u)-sum(u))) # 初期値
  conv <- 0 # 収束判定の結果（収束したら 1 とする）
  while(conv==0){ # 収束していない間は繰り返す
    t1 <- t0-dLL(u,a,b,t0)/ddLL(a,b,t0) # 更新
    if(abs(t1-t0)<e) conv <- 1 # 収束を判定
    t0 <- t1 # 更新した解を保存
  }
  theta.mle[i] <- t1 # 最適解を保存
}

theta.mle # 推定値
## ------------------------------------------------------------------


## ------------------------------------------------------------------
## 能力パラメタの MAP 推定
## ------------------------------------------------------------------
## 反応パタン（4 項目× 5 名分）
u.all <- matrix(c(0,0,0,1,
                  0,1,0,1,
```

```
                  1,1,1,0,
                  1,0,1,0,
                  1,0,1,1),ncol=4,byrow=TRUE)

## 項目パラメタ
a <- c(0.7880079,2.0406337,0.6121101,1.0210670) # 識別力
b <- c(1.45240645,0.33258311,0.07252222,-1.13507144) # 困難度

## 関数の定義
icc2PL <- function(a,b,theta){
  1/(1+exp(-1.7*a*(theta-b)))
}

LL <- function(u,a,b,theta){ # 対数尤度関数
  sum(u*log(icc2PL(a,b,theta)) + (1-u)*log(1-icc2PL(a,b,theta)))
}

dLL <- function(u,a,b,theta){ # 一次導関数
  1.7*sum(a*(u-icc2PL(a,b,theta)))
}

ddLL <- function(a,b,theta){ # 二次導関数
  -1.7^2*sum(a^2*icc2PL(a,b,theta) * (1-icc2PL(a,b,theta)))
}

LG <- function(u,a,b,theta,mu,sigma){ # 対数事後分布
  LL(u,a,b,theta) - ((theta-mu)/sigma)^2/2
}

dLG <- function(u,a,b,theta,mu,sigma){ # 一次導関数
  dLL(u,a,b,theta) - (theta-mu)/sigma^2
}

ddLG <- function(a,b,theta,sigma){ # 二次導関数
  ddLL(a,b,theta) - 1/sigma^2
}

## 初期設定
mu <- 0 # 事前分布の平均
sigma <- 1 # 事前分布の標準偏差
ni <- nrow(u.all) # 受験者数 (I)
e <- 0.0001 # 収束基準
theta.map <- numeric(ni) # 能力パラメタ推定値格納用

## 最適化計算
for(i in 1:ni){ # 受験者ごとにループ
  u <- u.all[i,] # i 番目の受験者の反応ベクトル
  t0 <- log(sum(u)/(length(u)-sum(u))) # 初期値
  conv <- 0 # 収束判定の結果（収束したら 1 とする）
  while(conv==0){ # 収束していない間は繰り返す
    t1 <- t0-dLG(u,a,b,t0,mu,sigma)/ddLG(a,b,t0,sigma) # 更新
    if(abs(t1-t0)<e) conv <- 1 # 収束を判定
    t0 <- t1 # 更新した解を保存
  }
  theta.map[i] <- t1 # 最適解を保存
```

```
}

theta.map # 推定値
## ------------------------------------------------------------------


## ------------------------------------------------------------------
## 能力パラメタの EAP 推定
## ------------------------------------------------------------------
## 反応パタン（4 項目× 5 名分）
u.all <- matrix(c(0,0,0,1,
                  0,1,0,1,
                  1,1,1,0,
                  1,0,1,0,
                  1,0,1,1),ncol=4,byrow=TRUE)

## 項目パラメタ
a <- c(0.7880079,2.0406337,0.6121101,1.0210670) # 識別力
b <- c(1.45240645,0.33258311,0.07252222,-1.13507144) # 困難度

## 関数の定義
icc2PL <- function(a,b,theta){ # 2PLM の ICC
  1/(1+exp(-1.7*a*(theta-b)))
}

L <- function(u,a,b,theta){ # 尤度関数
  prod(icc2PL(a,b,theta)^u*(1-icc2PL(a,b,theta))^(1-u))
}

## 初期設定
mu <- 0 # 事前分布の平均
sigma <- 1 # 事前分布の標準偏差
ni <- nrow(u.all) # 受験者数 (I)
Xm <- seq(-4,4,length.out=15) # 求積点
Wm <- dnorm(Xm,mu,sigma) # 重み
Wm <- Wm/sum(Wm) # 総和=1 となるように調整
theta.eap <- numeric(ni) # 能力パラメタ推定値格納用

## 離散近似
for(i in 1:ni){ # 受験者ごとにループ
  u <- u.all[i,] # i 番目の受験者の反応ベクトル
  Lm <- numeric(length(Xm))
  for(m in 1:length(Xm)) Lm[m] <- L(u,a,b,Xm[m])
  Gm <- Lm*Wm/sum(Lm*Wm)
  theta.eap[i] <- sum(Xm*Gm) # 推定値を保存
}

theta.eap # 推定値
## ------------------------------------------------------------------
```

2. 省略

3. 
```
## ------------------------------------------------------------------
## EM アルゴリズムによる項目パラメタの周辺最尤推定
## ------------------------------------------------------------------
## 反応パタン行列読み込み
```

```
u <- read.csv("data_EM.csv",header=FALSE)

## 関数の定義
icc2PL <- function(a,b,theta){ # 2PLM の ICC
  1/(1+exp(-1.7*a*(theta-b)))
}

L <- function(u,a,b,theta){ # 尤度関数
  prod(icc2PL(a,b,theta)^u * (1-icc2PL(a,b,theta))^(1-u))
}

ELL <- function(r,N,a,b,theta){ # 期待対数完全データ尤度関数
  sum(r*log(icc2PL(a,b,theta)) + (N-r)*log(1-icc2PL(a,b,theta)))
}

dtj <- function(r,N,a,b,theta){ # 勾配ベクトル
  da <-  1.7*sum((theta-b)*(r-N*icc2PL(a,b,theta)))
  db <- -1.7*a*sum(r-N*icc2PL(a,b,theta))
  c(da,db)
}

Itj <- function(N,a,b,theta){ # 情報行列
  daa <- sum(N*icc2PL(a,b,theta)*(1-icc2PL(a,b,theta))*(theta-b)^2)
  dab <- sum(-N*icc2PL(a,b,theta)*(1-icc2PL(a,b,theta))*a*(theta-b))
  dbb <- sum(N*icc2PL(a,b,theta)*(1-icc2PL(a,b,theta))*a^2)
  (1.7^2)*matrix(c(daa,dab,dab,dbb),nrow=2)
}

## 初期設定
ni <- nrow(u) # 受験者数 (I)
nj <- ncol(u) # 項目数 (J)

e <- 0.0001 # 収束基準
convEM <- 0 # 収束判定の結果（収束したら 1 とする）
mll <- numeric() # 周辺対数尤度の履歴格納用

## 項目パラメタの初期値（行＝項目，列 1 ＝識別力，列 2 ＝困難度）
r <- as.vector(cor(rowSums(u),u)) # 点双列相関係数
p <- colMeans(u) # 正答率
t0 <- matrix(nrow=nj,ncol=2)
t0[,1] <- 1.7*r/sqrt(1-r^2)
t0[,2] <- qnorm(p,0,1,lower.tail=FALSE)/r

Lim <- matrix(nrow=ni,ncol=nm) # 行＝受験者，列＝求積点
for(i in 1:ni){ # 各受験者× Xm ごとの尤度
  for(m in 1:nm) Lim[i,m] <- L(u[i,],t0[,1],t0[,2],Xm[m])
}
cat(0,":",(mll <- sum(log(Lim%*%Wm))),"\n") # 周辺対数尤度＠初期値

iEM <- 1
while(convEM==0){ # EM サイクル
## ------------------------------------------------------------------
## E ステップ
  Lim <- matrix(nrow=ni,ncol=nm) # 行＝受験者，列＝求積点
  for(i in 1:ni){ # 各受験者× Xm ごとの尤度
```

```
    for(m in 1:nm) Lim[i,m] <- L(u[i,],t0[,1],t0[,2],Xm[m])
  }

  Gim <- matrix(nrow=ni,ncol=nm) # 行＝受験者，列＝求積点
  for(i in 1:ni){ # 事後分布の重み
    Gim[i,] <- Lim[i,]*Wm/sum(Lim[i,]*Wm)
  }

  Nm <- colSums(Gim) # 期待度数（求積点ごと）
  rjm <- t(u)%*%Gim # 期待正答数（行＝項目，列＝求積点）

## ------------------------------------------------------------------
## M ステップ
  t0m <- t1m <- t0 # 初期値をコピー

  for(j in 1:nj){
    conv <- 0 # 収束判定の結果（収束したら 1 とする）
    while(conv==0){ # 収束していない間は繰り返す
      t1m[j,] <- t0m[j,]+
        solve(Itj(Nm,t0m[j,1],t0m[j,2],Xm))%*%
          dtj(rjm[j,],Nm,t0m[j,1],t0m[j,2],Xm) # 更新
      if(all(abs(t1m[j,]-t0m[j,])<e)) conv <- 1 # 収束を判定
      t0m[j,] <- t1m[j,] # 更新した解を保存
    }
  } #j

  Lim <- matrix(nrow=ni,ncol=nm) # 行＝受験者，列＝求積点
  for(i in 1:ni){ # 各受験者× Xm ごとの尤度
    for(m in 1:nm) Lim[i,m] <- L(u[i,],t1m[,1],t1m[,2],Xm[m])
  }
  mllm <- sum(log(Lim%*%Wm)) # 周辺対数尤度＠更新値
  cat(iEM,":",mllm,"\n")
  mll <- c(mll,mllm)

  if(all(abs(t1m-t0)<e)) convEM <- 1 # 収束を判定
  t0 <- t1m # 更新した解を保存
  iEM <- iEM + 1
} # EM サイクル

t1m # 最適解
## ------------------------------------------------------------------
```

4. 省略

## 第 9 章

1. 省略
2. 省略

## 第 10 章

1. カテゴリ確率曲線（$a = 1.5$，$b = (-1.5, 0, 1)$ の場合）

(a) 本文と同様のやり方

```
x <- seq(-3,3,.01)
bunshi <- rbind(0,x,x,x)
bunshi <- 1.5*(bunshi-c(0,-1.5,0,1))
for(k in 2:4) bunshi[k,] <- bunshi[k-1,]+bunshi[k,]
bunshi <- exp(bunshi)
bunbo <- colSums(bunshi)
cpc.gpcm <- sweep(bunshi,2,bunbo,"/")

plot(x,type="n",xlim=c(-3,3),ylim=c(0,1),xlab="θ",ylab="反応確率")
for(k in 1:4) points(x,cpc.gpcm[k,],type="l",lty=k)
```

(b) 別のやり方

```
x <- seq(-3,3,.01)
aj <- 1.5 # 識別力
bjk <- c(-1.5,0,1) # 遷移点
z <- -aj*outer(bjk,x,FUN="-")
nk <- nrow(z)
for(k in 2:nk) z[k,] <- z[k,]+z[k-1,]
ez <- exp(rbind(0,z))
pj <- sweep(ez,2,colSums(ez),FUN="/") # カテゴリ確率曲線
nk <- nrow(pj)

plot(x,pj[1,],type="n",ylim=c(0,1),xlab=expression(theta),ylab="反応確率")
for(k in 1:nk) points(x,pj[k,],type="l",lty=k)
```

2. IIF（$a = 1.5$，$b = (-1.5, 0, 1)$ の場合；1(a) で求めた cpc.gpcm を利用）

```
term1 <- 1.5^2 * (0:3)^2%*%cpc.gpcm # 第１項
term2 <- 1.5^2 * ((0:3)%*%cpc.gpcm)^2 # 第２項
iif.gpcm <- term1 - term2 # IIF
plot(x,iif.gpcm,type="l",xlab="θ",ylab="情報量")
```

3. 【メリット】特に低能力域を中心に，IIF の向上が期待できる。その結果，二値型項目でテストを組む場合と比較して，同程度の TIF を得るのに少ない項目数で済む。カテゴリ確率曲線により，項目の特徴をより細かく評価検討できる。

   【デメリット】項目パラメタの推定にあたり，二値型モデルよりも多くの受験者が必要である。受験者の能力分布においても，より注意を払う必要がある。作問・採点にかかる時間やコストが大きくなる可能性がある。

## 第 11 章

1. $A = 1$ という状況で，$K$ が 0 より大きい場合には，変換する尺度の平均の方が，基準尺度の平均よりも低い。$K = 0$ という状況で，$A$ が 1 より小さい場合には，変換する尺度の方が，同一の能力幅に対して，より細かい目盛りを割り当てている。

2. 困難度が低いパラメタが少ないので，低能力者の能力パラメタの推定誤差が大きくなる可能性がある。誤差の大きい能力パラメタを利用して求めた等化係数にはバイアスが含まれる可能性が高い。

3. 【方法 1】固定母数による等化法を適用する。

   【方法 2】同一受験者について，$T$ の等化済み 25 項目と，$F$ の新規 25 項目でそれぞれ能力パラメタを求め，5,000 人の共通受験者を利用して等化係数を求める。

   【方法 3】尺度 $F$ の全 50 項目に対する 5,000 人の項目反応を利用して，等化前パラメタを推定する。次に，$T$ の部分の等化前パラメタと等化後パラメタを利用して，共通項目法で等化係数を求める。

4. 省略

## 第 12 章

1. 【項目 1】正答率の高い項目であるが，能力の低い受験者を非常によく識別できる。

   【項目 2】正答率，識別力ともに低い項目である。中程度の能力域（正答数＝ 15 点付近）で正答率が落ち込み，代わりに誤答である選択肢 1 や 4 が多く選ばれる傾向がある。このような項目に IRT を当てはめることは難しい。

   【項目 3】正答率，識別力ともに低い項目である。誤答である選択肢 1 の選択率が能力が高くなるにつれて上昇しており,「引っかけ」的な誤答である可能性がある。

   【項目 4】識別力の高い項目である。誤答肢にもおかしな動きをするものはなく，全体として良好な性質の項目であると言える。

2. 
```
## u は二値反応パタン行列
## y の各列には各項目に対応する除外正答数を入れる
y <- u
for(j in 1:ncol(u)) y[,j] <- rowSums(u[,-j],na.rm=TRUE)
diag(cor(u,y,use="pairwise.complete.obs"))
```

## 第 13 章

1. 
```
## ICL
ip.icl <- est(resp=u,model="2PL",engine="icl",
              a.prior=FALSE,b.prior=FALSE,c.prior=FALSE,
              bilog.defaults=FALSE,run.name="vocab_2PL")
## ltm
ip.ltm <- est(resp=u,model="2PL",engine="ltm",
              a.prior=FALSE,b.prior=FALSE,c.prior=FALSE,
              bilog.defaults=FALSE,run.name="vocab_2PL")
```

   【検証プラン】シミュレーションによる検証が有効である。項目パラメタの真値を決め，反応パタン行列をシミュレーションによって生成し，それぞれのプログラムで項目パラメタを推定する。これをくりかえし，項目パラメタの真値と推定値のズレの大きさを評価する。

2. 省略

## 第 14 章

```
param <- read.csv("equatetest.csv",header=TRUE) # データ読み込み
cond <- c(5,25,45,65,85,100) # 各条件の項目数を設定
storage <- list(NA) # 結果格納用

j <- 1
for(i in cond){
  pm <- list(cbind(param[1:i,1:2],0),cbind(param[1:i,3:4],0))

  comx <- data.frame("T"=1:i,"F"=1:i)
  rescat <- list(rep(2,i),rep(2,i))

  pmT <- as.poly.mod(n=i,model="drm",items=1:i)
  pmF <- as.poly.mod(n=i,model="drm",items=1:i)

  res <- as.irt.pars(x=pm,common=comx,cat=rescat,poly.mod=list(pmT,pmF))

  out <- plink(x=res,rescale="MS",base.grp=1)
  storage[[j]] <- link.con(out)
  j <- j + 1
}

storage # 分析結果を表示
```

## 第 15 章

1. 決定変数：$x_j$, $j = 1, \ldots, J$

   目的関数：$\sum_{j=1}^{J} \alpha_j x_j \longrightarrow$ 最大化

   制約条件：$\sum_{j=1}^{J} t_j x_j \geq 50$, $\sum_{j=1}^{J} t_j x_j \leq 60$, $\sum_{j=1}^{J} \beta_j x_j \geq 0$

2. 以下の制約を元の仕様に追加して，最適化を実行する。

   ```
   ## 追加制約 1：18,38,62 以外の項目は強制投入
   x0 <- x # 最適化された決定変数の値を x0 にコピー
   x0[c(18,38,62)] <- 0 # 差し替える項目の値を 0 に変更
   A <- rbind(A,x0) # A に行として追加
   lb <- c(lb,sum(x0))
   ub <- c(ub,sum(x0))
   type <- c(type,5)
   Alab.row <- c(Alab.row,"Keep")

   ## 追加制約 2：18,38,62 は強制排除
   A <- rbind(A,0) # A に全要素 0 の行を追加
   A[nrow(A),c(18,38,62)] <- 1 # 追加行の当該項目箇所を 1 に変更
   lb <- c(lb,0)
   ub <- c(ub,0)
   type <- c(type,5)
   Alab.row <- c(Alab.row,"Drop")

   ## 制約数を更新（必須）
   ni <- nrow(A)
   ```

```
## glpkAPI で計算実行
library(glpkAPI)
prob <- initProbGLPK() # オブジェクト生成・初期化
setObjDirGLPK(prob,GLP_MIN) # 最適化の方向を指定

addRowsGLPK(prob,ni) # A の行数を指定
addColsGLPK(prob,nj) # A の列数を指定
setRowsNamesGLPK(prob,1:ni,Alab.row) # 行ラベル
setColsNamesGLPK(prob,1:nj,Alab.col) # 列ラベル

idx.a <- which(A!=0) # A の非ゼロ要素のフラグ
na <- length(idx.a) # 非ゼロ要素の数
ia <- rep(1:ni,nj)[idx.a] # 非ゼロ要素の行番号
ja <- rep(1:nj,each=ni)[idx.a] # 非ゼロ要素の列番号
aij <- as.vector(A)[idx.a] # 非ゼロ要素の値ベクトル
loadMatrixGLPK(prob,na,ia,ja,aij) # 値指定（A）
setRowsBndsGLPK(prob,1:ni,lb,ub,type) # 値指定（b）

setColsKindGLPK(prob,1:nj,rep(GLP_BV,nj)) # 決定変数の型
setObjCoefsGLPK(prob,1:nj,obj.coef) # 決定変数の係数

solveSimplexGLPK(prob)
printSolGLPK(prob,"ATAexample_LP2.log")
solveMIPGLPK(prob)
printMIPGLPK(prob,"ATAexample_IP2.log")
x <- mipColsValGLPK(prob) # 結果を変数に格納
delProbGLPK(prob) # オブジェクト消去
```

【実行結果：APAexample_IP2.log の中身】

```
Problem:
Rows:       13
Columns:    90 (90 integer, 90 binary)
Non-zeros:  666
Status:     INTEGER OPTIMAL
Objective:  32.35273 (MINimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 N_total                   30            30             =
     2 N_geometry                15             8
     3 N_algebra                  8             8
     4 N_statistics               7             5
     5 Enemies                    1                           1
     6 Item_set                   0             0             =
     7 TTIF1                3.05925          2.57
     8 TTIF2                6.40036             6
     9 TTIF3                8.43235          8.43
    10 TTIF4                6.26248             6
    11 TTIF5                3.25201          2.57
    12 Keep                      27            27             =
    13 Drop                       0             0             =

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
```

```
     1 x1           *              0              0              1
     2 x2           *              0              0              1
     3 x3           *              0              0              1
     4 x4           *              1              0              1
     5 x5           *              1              0              1
[以下省略]
```

【実行結果：選ばれた項目の一覧】

```
> which(x==1) # 選ばれた項目一覧
 [1]  4  5  6  8 11 12 16 17 19 20 21 25 27 28 29 31 32 36 43 46
[21] 48 50 60 64 71 73 83 85 86 89
> item[x==1,] # 選ばれた項目の情報
   item area        a         b
4    x4    1 1.215321 -0.153958
5    x5    1 0.604725 -1.672740
6    x6    1 1.323379 -0.475509
[中略]
89  x89    3 1.213201  0.451392
```