プログラムの使用法

カテゴリ数が４以上の場合

岡本安晴、2018

モデル

カテゴリ項目を用いる心理尺度においては、統計学的信頼性と心理学的信頼性を区別する必要がある。統計学的信頼性は、連続量の項目に対する統計学的モデルを形式的にカテゴリ項目に適用したモデルに基づくと理解されるもので、心理学的モデル（サーストンのカテゴリ判断の法則／モデル）に基づく心理学的信頼性より大きい推定値になる傾向がある（Okamoto, 2016, 2017）。

統計学的信頼性は、以下のように連続量項目のモデルと統計学的形式において同じである。

人iのカテゴリ項目jの回答カテゴリ値をX_{ij}で表し、真値T_{ij}と誤差E_{ij}の和

(1)　$X_{ij} = T_{ij} + E_{ij},$

に分解する。カテゴリ変数X_{ij}の真値T_{ij}を統計学的にX_{ij}の平均値として定義する。すなわち、

(2)　$T_{ij} = E[X_{ij}]$

である。式（１）と式（２）により、項目がカテゴリ変数の場合も連続変数の場合と同じように扱うことができる。しかし、カテゴリ変数の場合は、その値は整数値であり、真値も整数値であると考える方が自然である。真値を平均値（２）で与えるのは、統計学的便法である。ポピュラーな信頼性係数であるクロンバックのアルファ係数は、カテゴリ項目からなる尺度の場合、モデル（１）と（２）に基づく統計学的信頼性係数であると理解される。α係数は、次に説明する心理学的信頼性係数より大きい値をとる傾向がある（Okamoto, 2016, 2017）。

心理学的信頼性は心理学的概念の真値とその測定値との関係の強さを表すものである。心理学的概念の強さとカテゴリ項目のカテゴリ値との関係をサーストンのカテゴリ判断の法則に基づいてモデル構成する。このモデルは、項目反応理論（item response theory, IRT）では Samejima のモデルとして知られている。

人iのカテゴリ値をとる項目jへの回答をY_{ij}とする。この回答は、項目jへの連続変量で表される潜在反応U_{ij}のカテゴリ化によって生成される。潜在変量である反応U_{ij}は、１因子モデルで表される場合を考えると、次式（３）で表される（Okamoto, 2017）。

(3)　$U_{ij} = \mu_j + \lambda_j F_i + E_{ij}$

ここで、F_iは測定対象である心理学的概念の人iの値であり、項目jのU_{ij}がF_iを因子とする因子モデルとして設定されている。誤差項E_{ij}は他の変数と独立であり、正規分布に従うとす

る。この潜在変量U_{ij}に対して項目jのカテゴリ反応Y_{ij}が次式（４）により与えられる。

(4)　　$Y_{ij} = k, \quad if \;\; C_{k-1} \le U_{ij} < C_{k,}$

ここで、C_kはカテゴリ境界である。

人iの尺度値Y_iが次式（５）のカテゴリ値の和で与えられるとする。

(5)　　$Y_i = \sum_{j=1}^{M} Y_{ij}.$

このとき、測定値Y_iと真値F_iとの関係を次の回帰モデルで表す。

(6)　　$Y_i = a_F + b_F F_i + \varepsilon_F$

または、

(7)　　$F_i = a_U + b_U Y_i + \varepsilon_U$

測定値Y_iと真値F_iとの関係の強さは、回帰モデル（６）あるいは（７）の決定係数$R^2 - index$で与えることができる。決定係数の値は式（６）あるいは（７）で同じ値であり、測定値Y_iと真値F_iとの相関係数の２乗で与えられ、R^2（Okamoto, 2016）あるいはρ_{cat}（Okamoto, 2017）で表すと

(8)　　$R^2 = \rho_{cat} = R^2 - index = [Cor(Y_i, F_i)]^2$

となる。

上式で与えられる信頼性係数R^2あるいはρ_{cat}は、測定値Y_iと測定対象である心理学的概念の真値F_iとの関係の強さを表すもので、心理学的信頼性係数といえる。これに対して、統計学的信頼性係数と考えられるα係数は、このρ_{cat}より大きい値となる傾向がある（Okamoto, 2016, 2017）。実証科学としての心理学研究においては、心理学的信頼性係数ρ_{cat}を求めておくべきである。

式（８）の計算式は Okamoto（2017）で与えられているが、やや複雑である。しかし、次式（９）を用いる方法は、次に示すように式（９）が間違っているので使えないことに注意。

(9)　　$Cov(X,Y) = \int_{-\infty}^{+\infty} Cov(X,Y|\theta)p(\theta)d\theta, \qquad (Wrong \;\; Equation)$

ここで、$Cov(X,Y|\theta)$は、条件θの下でのXとYの共分散を表し、$p(\theta)$はθの確率密度関数である。

式（９）が成り立たない例

２つの確率変数XとYを次のようにおく。

$X = F, \qquad F \sim N(0,1)$　標準正規分布

$Y = a + bX$

確率変数XとYの平均値を$\bar{X}$と$\bar{Y}$で表すと、

$$Cov(X,Y) = E\big((X-\bar{X})(Y-\bar{Y})\big) = E(XY) - \bar{X}\bar{Y}$$

である。

次式

$$\bar{X} = E(F) = 0,$$

より、

$$\bar{X}\bar{Y} = 0$$

をとなる。

また、

$$E(XY) = E(aX + bX^2) = a\bar{X} + bE(F^2) = bVar(F) = b$$

であるので、次式（10）を得る。

(10) $\quad Cov(X, Y) = b.$

いま、確率変数XとYの条件Fでの確率変数を$X|F$と$Y|F$で表し、それらの平均値を$\bar{X}|F$と$\bar{Y}|F$で表すと、

$$\begin{aligned} Cov(X, Y|F) &= E[(X|F - \bar{X}|F)(Y|F - \bar{Y}|F)]|F \\ &= E\big[(F - F)\big((a + bF) - (a + bF)\big)\big]|F \\ &= 0 \end{aligned}$$

を得る。ここで、$E[\cdot]|F$は、条件Fの下での期待値を表す。

これより、

(11) $$\int_{-\infty}^{+\infty} Cov(X, Y|F)\phi(F)dF = 0$$

である。ここで、$\phi(F)$は標準正規分布$N(0, 1)$の確率密度関数である。

式（１０）と式（11）より、式（9）が成り立たないことがわかる。

モデル（３）および（４）よりカテゴリ反応の確率は次式で与えられる。

$$\begin{aligned} &P\big(Y_{ij} = k \ |\boldsymbol{F}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\psi}, \boldsymbol{C}\big) \\ &= P\big(C_{k-1} \le U_{ij} < C_k \ |\boldsymbol{F}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\psi}, \boldsymbol{C}\big) \\ &= \Phi\left(\frac{C_k - \big(\mu_j + \lambda_j F_i\big)}{\psi_j}\right) - \Phi\left(\frac{C_{k-1} - \big(\mu_j + \lambda_j F_i\big)}{\psi_j}\right) \end{aligned} \quad (12)$$

式（１２）より、パラメータの原点と単位が任意であることがわかる。パラメータ値の推定のために何らかの制約条件が必要である（Okamoto, 2017）。原点と単位を設定するために次の制約条件を設定する（Okamoto, 2017）。

(13) $\quad C_1 = -1 \quad and \quad C_{K-1} = 1$

式（１３）に対応する Stan スクリプトをリスト 1 に示す。

プログラムの使い方

メインスクリプト Main.py を実行すると、図 1 のようになる入力データファイルと出力

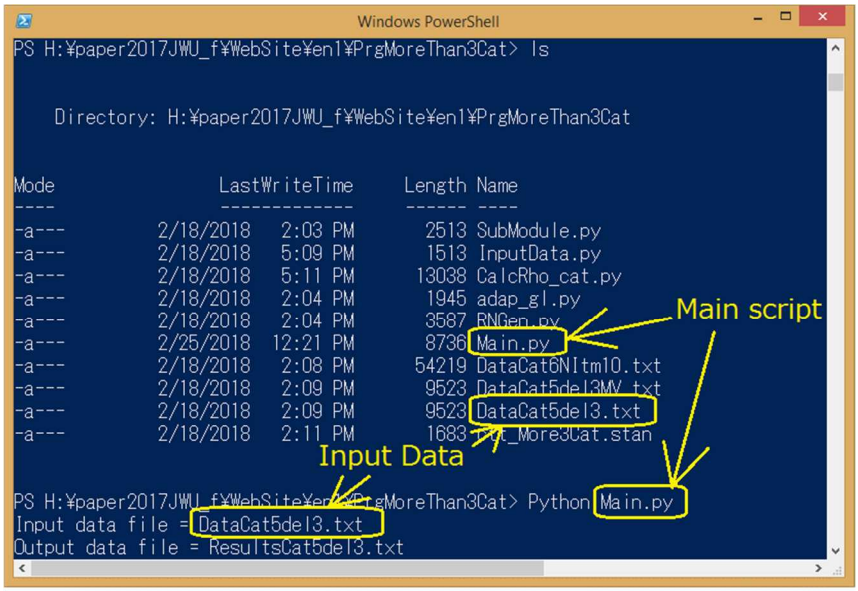

図 1

用ファイル名の設定が求められる。出力用ファイル名は任意のテキストファイル名である。
入力ファイルは、図 2 に示す形式で用意する。

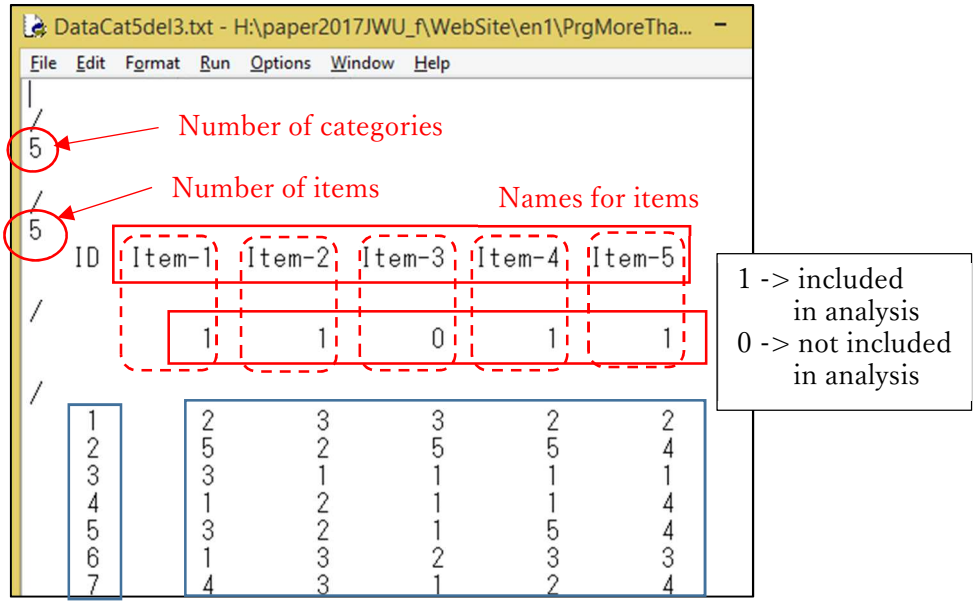

図 2

行の先頭がスラッシュ / で始まる最初の行の次の行にカテゴリ数を書く。
行の先頭がスラッシュ / で始まる 2 番目の行の次の行に項目数を書き、その次の行に変

数名を並べる。変数名は、1番目が人のIDを表す変数名であり、2番目から項目名を1番目のものから順に並べて書く。

行の先頭がスラッシュ / で始まる3番目の行の次にの行に、項目を分析に含めるかどうかの指示を書く。この指示は数値1または0で示し、項目の順番に並べる。指示数値が1である項目は分析に含まれ、0であれば除かれる。

行の先頭がスラッシュ / で始まる4番目の行の次の行から各人の項目に対する反応のデータを1行に1人ずつの形式で書く。各行の1番目の値は人のID用の文字列である。2番目から各項目に対する反応を順番に並べて書く。項目に対する反応は、項目のカテゴリ数がKであれば、1からKまでの整数値である（Okamoto, 2017）。1からKまでの整数値以外の整数値は欠測値として扱われる。

最後のデータの次の行は、先頭をスラッシュ / で始め、データの終わりであることを示す（図3）。

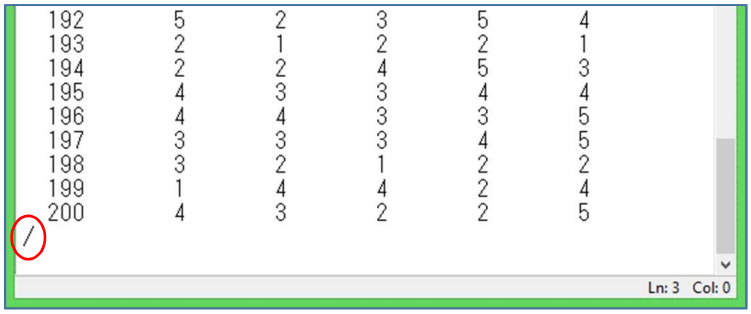

図3

出力ファイル名の入力後、Stanスクリプトによる計算が始まる。Stanスクリプトによるサンプリングが終わると、サンプルの中央値が算出され、それに基づいて信頼性係数ρ_{cat}が算出され、Est_Rho_cat by Medsの値として表示される。図4では、$\hat{\rho}_{cat} \approx 0.704$が表示されている。

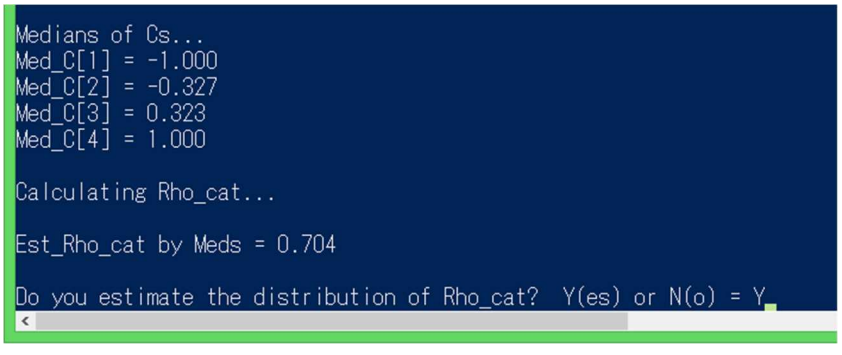

図4

信頼性係数の表示後、信頼性係数の事後分布を求めるかどうかが聞かれる。文字N

を押して、Enter キーを押すと、プログラムは終了する。文字 Y を押して、Enter キーを押すと、信頼性係数 ρ_cat のサンプリングが始まる。サンプル数は 100 個で、これを 4 グループに分けて、それぞれがプロセスを呼び出して並行処理される（図 5）。

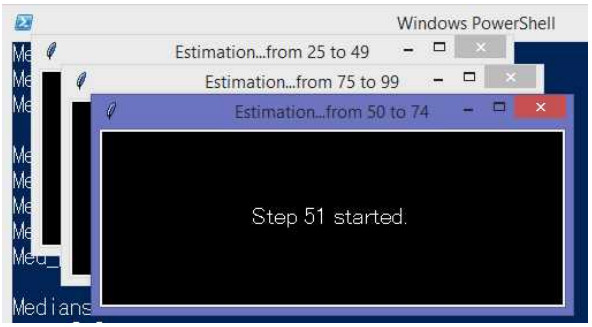

図 5

サンプリング後、サンプルされた信頼性係数 ρ_cat のヒストグラムが表示される（図 6）。

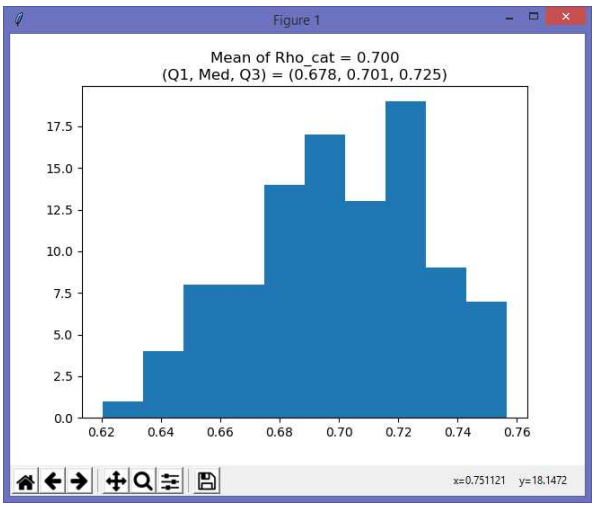

図 6

図 6 のウィンドウを閉じると、プログラムは終了する。プログラムの終了後、適当なテキストエディタで出力ファイルを開くと、以下のようになっている。

```
Input data file = DataCat5del3.txt
```

入力データファイル名

```
NPerson = 200
NItems = 4
NCat = 5
1/1:  2 3 2 2
2/2:  5 2 5 4
3/3:  3 1 1 1
4/4:  1 2 1 4
5/5:  3 2 5 4
6/6:  1 3 3 3
```

人数、項目数、カテゴリ数

入力データ
通し番号/ID：　項目 1　・・・　項目 4

```
7/7:  4 3 2 4
8/8:  3 4 3 1
9/9:  4 2 1 3
10/10:  3 5 4 5
11/11:  1 1 2 1
```

```
195/195:  4 3 4 4
196/196:  4 4 3 5
197/197:  3 3 4 5
198/198:  3 2 2 2
199/199:  1 4 2 4
200/200:  4 3 2 5

Reconstructed data for the Stan script...
1/800  IDpsn/1  IDitm/1  Res/2
2/800  IDpsn/1  IDitm/2  Res/3
3/800  IDpsn/1  IDitm/3  Res/2
4/800  IDpsn/1  IDitm/4  Res/2
5/800  IDpsn/2  IDitm/1  Res/5
6/800  IDpsn/2  IDitm/2  Res/2
7/800  IDpsn/2  IDitm/3  Res/5
8/800  IDpsn/2  IDitm/4  Res/4
9/800  IDpsn/3  IDitm/1  Res/3
10/800  IDpsn/3  IDitm/2  Res/1
11/800  IDpsn/3  IDitm/3  Res/1
12/800  IDpsn/3  IDitm/4  Res/1
.
.
.
```

```
795/800  IDpsn/199  IDitm/3  Res/2
796/800  IDpsn/199  IDitm/4  Res/4
797/800  IDpsn/200  IDitm/1  Res/4
798/800  IDpsn/200  IDitm/2  Res/3
799/800  IDpsn/200  IDitm/3  Res/2
800/800  IDpsn/200  IDitm/4  Res/5

Coeff_alpha = 0.712
```

通し番号　ID　項目番号　反応

信頼性係数 α

```
fit...
Inference for Stan model: anon_model_7366729ff08dfa9ee4125fb5cab9973f.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

            mean se_mean     sd   2.5%    25%    50%    75%  97.5%  n_eff   Rhat
Lambda[0]    0.7  2.2e-3   0.09   0.52   0.64    0.7   0.76   0.88   1691    1.0
Lambda[1]   0.61  1.8e-3   0.09   0.44   0.55   0.61   0.67   0.79   2456    1.0
Lambda[2]   0.67  1.8e-3   0.09    0.5   0.61   0.66   0.73   0.85   2363    1.0
Lambda[3]   0.62  1.8e-3   0.09   0.45   0.56   0.62   0.68   0.81   2566    1.0
mu[0]       0.05  1.7e-3   0.08   -0.1 1.4e-3   0.05    0.1   0.21   2149    1.0
mu[1]      -0.05  1.6e-3   0.07  -0.19   -0.1  -0.05 5.0e-3    0.1   2225    1.0
mu[2]      -0.01  1.7e-3   0.08  -0.16  -0.06-8.4e-3   0.04   0.14   2119    1.0
mu[3]    -9.7e-3  1.5e-3   0.08  -0.16  -0.06-7.9e-3   0.04   0.14   2526    1.0
psi[0]      0.74  1.5e-3   0.07    0.6   0.69   0.74   0.79   0.89   2277    1.0
psi[1]      0.76  1.1e-3   0.06   0.64   0.72   0.76    0.8   0.89   3227    1.0
psi[2]      0.76  1.4e-3   0.07   0.63   0.72   0.76    0.8    0.9   2303    1.0
psi[3]      0.81  1.1e-3   0.07   0.69   0.77   0.81   0.86   0.95   4000    1.0
C[0]        -1.0     0.0    0.0   -1.0   -1.0   -1.0   -1.0   -1.0   4000    nan
C[1]       -0.33  6.0e-4   0.04   -0.4  -0.35  -0.33   -0.3  -0.25   4000    1.0
C[2]        0.32  6.1e-4   0.04   0.24    0.3   0.32   0.35    0.4   4000    1.0
C[3]         1.0 2.0e-175.7e-17    1.0    1.0    1.0    1.0    1.0      8    nan
lp__        -1189    0.53  15.46  -1220  -1199  -1188  -1178  -1160    861    1.0
```

```
Samples were drawn using NUTS at Tue Feb 27 19:57:16 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).

Medians of Lambdas...
Med_Lambda[1] = 0.699
Med_Lambda[2] = 0.606
Med_Lambda[3] = 0.665
Med_Lambda[4] = 0.622

Medians of mus...
Med_mu[1] = 0.050
Med_mu[2] = -0.048
Med_mu[3] = -0.008
Med_mu[4] = -0.008

Medians of psis...
Med_psi[1] = 0.740
Med_psi[2] = 0.761
Med_psi[3] = 0.759
Med_psi[4] = 0.811

Medians of Cs...
Med_C[1] = -1.000
Med_C[2] = -0.327
Med_C[3] = 0.323
Med_C[4] = 1.000

Est_Rho_cat by Meds = 0.704

Mean of Rho_cat = 0.700
Median of Rho_cat = 0.701
[Q1, Q3] = [0.678, 0.725]
```

ρ_{cat}: 各パラメータの中央値から算出

ρ_{cat}の事後分布から算出
Q1（第１四分位数)、Q3（第３四分位数）

スクリプトのリスト

ウェブサイトからダウンロードした圧縮ファイル PrgMoreThan3Cat.zip には、図７のファイルが含まれている。このうち、Stan スクリプト ctt_More3Cat.stan をリスト１に、メインの Python スクリプト Main.stan をリスト２に示す。メイン Python スクリプトから Stan スクリプトが呼ばれてパラメータのサンプリングが行われ、このサンプル値に基づいて、心理学的信頼性係数ρ_{cat}が、モジュール SubModule.py で宣言されているクラス型 CalcRhoPreRhoYYP によって求められる。

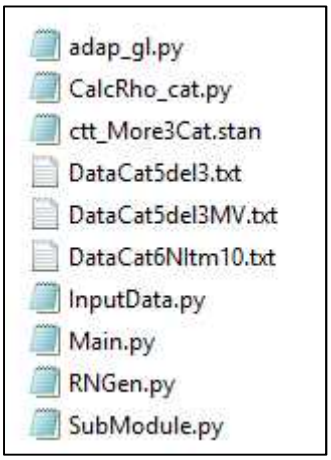

図７

リスト 1. カテゴリ数が４以上の場合の Stan スクリプト script ctt_More3Cat.stan

```
//      Yasuharu Okamoto, 2017.07

data {
    int <lower = 4> K;                          //      Number of categories > 3
    int Npsn;                                   //      Number of persons
    int Nitm;                                   //      Number of items
    int Ntot;                                   //      Number of data,  Ntot <= Nprn * Nitm
    int<lower = 1, upper = Npsn> IDpsn[Ntot];   //      Person ID
    int<lower = 1, upper = Nitm> IDitm[Ntot];   //      Item ID
    int<lower = 1, upper = K> Res[Ntot];        //      Response-> An integer value between 1
and K
}

parameters {
        real<lower = 0.0> Lambda[Nitm];
        real mu[Nitm];
        simplex[K - 2] w_c;
        real<lower = 0.0> psi[Nitm];
        real F[Npsn];
}

transformed parameters {
    vector[K] p[Ntot];
    real C[K - 1];

    C[1] = -1.0;
    for (k in 2:(K - 1)){
        C[k] = C[k - 1] + w_c[k - 1] * 2.0;
    }

    for (i in 1:Ntot){
        p[i][K] = Phi(((mu[IDitm[i]] + (Lambda[IDitm[i]] * F[IDpsn[i]])) - C[K - 1]) /
                    psi[IDitm[i]]);
        p[i][1] =1.0 - Phi(((mu[IDitm[i]] + (Lambda[IDitm[i]] * F[IDpsn[i]])) - C[1]) /
                    psi[IDitm[i]]);

        for (k in 2:(K - 1)){
            p[i][k] = Phi(((mu[IDitm[i]] + (Lambda[IDitm[i]] * F[IDpsn[i]])) - C[k - 1]) /
                    psi[IDitm[i]])
                    - Phi(((mu[IDitm[i]] + (Lambda[IDitm[i]] * F[IDpsn[i]])) - C[k]) /
                    psi[IDitm[i]]);
        }
    }
}

model {
    for (i in 1:Npsn)
        F[i] ~ normal(0.0, 1.0);
    for (i in 1:Ntot)
        Res[i] ~ categorical(p[i]);
}
```

リスト 2. Python メインスクリプト Main.py.

```
import pystan
from pystan import StanModel
```

```
import pickle
from multiprocessing.pool import Pool
import numpy as np
import matplotlib.pyplot as plt
from InputData import *
from SubModule import *
from CalcRho_cat import *
from RNGen import *


if __name__ == '__main__':

    #
    #       Prepare the input data file
    #

    fn_in = input("Input data file = ")
    f_in = open(fn_in, "r")

    #
    #       Prepare the output file
    #
    fn_out = input("Output data file = ")
    f_out = open(fn_out, "w")
    f_out.write("Input data file = " + fn_in + "¥n")


    NPerson, NItems, NCat, ID, X = ReadData(f_in, f_out)
    f_out.write('¥n¥nNPerson = {}'. format(NPerson))
    f_out.write('¥nNItems = {}'.format(NItems))
    f_out.write('¥nNCat = {}'.format(NCat))

    for i in range(len(ID)):
        f_out.write('¥n{0}/{1}: '.format((i + 1), ID[i]))
        for v in X[i]:
            f_out.write(' {}'.format(v))

    if (NCat < 4):
        print('¥nNumber of Categories is not larger than three !')
        raise Exception('This programm is for items with more than 3 categries.')


    K = NCat
    Ntot = 0
    IDpsn = []
    IDitm = []
    Res = []
    for i in range(NPerson):
        for j in range(NItems):
            if (X[i][j] >= 1) and (X[i][j] <= K):
                IDpsn.append(i + 1)
                IDitm.append(j + 1)
                Res.append(X[i][j])
                Ntot += 1

    f_out.write('¥n¥nReconstructed data for the Stan script... ')
    for t in range(Ntot):
        f_out.write('¥n{0}/{1}  IDpsn/{2}  IDitm/{3}  Res/{4}'.format(
            (t + 1), Ntot, IDpsn[t], IDitm[t], Res[t]))
```

```
coeff_alpha = CalcAlpha(X, NCat, NPerson, NItems)
if coeff_alpha < 0.0:
    print('¥nA missing value was found in the dataset.')
    print('The coefficient alpha was not calculated.')
    f_out.write('¥n¥nA missing value was found in the dataset.¥n')
    f_out.write('The coefficient alpha was not calculated.¥n')
else:
    print('¥nCoeff_alpha = {0:.3f}¥n'.format(coeff_alpha))
    f_out.write('¥n¥nCoeff_alpha = {0:.3f}¥n'.format(coeff_alpha))

Data = {'K': NCat, 'Npsn': NPerson, 'Nitm': NItems, 'Ntot': Ntot,
        'IDpsn': IDpsn, 'IDitm': IDitm, 'Res': Res}

iLmbd = []
imu = []
ipsi = []
for i in range(NItems):
    iLmbd.append(1.0)
    imu.append(0.0)
    ipsi.append(1.0)

def f_init():
    return dict(Lambda = iLmbd, mu = imu, psi = ipsi)


fit = pystan.stan(file = 'ctt_More3Cat.stan', data = Data,
                  init =f_init, seed = 999,
                  pars = ['Lambda', 'mu', 'psi', 'C'], n_jobs = 1)


"""
sm = StanModel(file = 'ctt_More3Cat.stan')
with open('model.pkl', 'wb') as f:
    pickle.dump(sm, f)

fit = sm.sampling(data = Data, seed = 999,
                  pars = ['Lambda', 'mu', 'psi', 'C'], n_jobs = 1)
with open('fit.pkl', 'wb') as g:
    pickle.dump(fit, g)
"""



"""
sm = pickle.load(open('model.pkl', 'rb'))
fit = pickle.load(open('fit.pkl', 'rb'))
"""



print(fit)

f_out.write('¥n¥nfit...¥n')
f_out.write('{}'.format(fit))

Llmbd = fit['Lambda']
Lmu = fit['mu']
Lpsi = fit['psi']
LC = fit['C']
RawNSamples = len(Llmbd)
```

```
rn = RNp2to191()

LSamplesID = []
for i in range(RawNSamples):
    LSamplesID.append([])
    LSamplesID[i].append(i)
    LSamplesID[i].append(rn.uni())

LSamplesID.sort(key = lambda v: v[1])

MCMCSmpl_lmbd = []
for i in range(NItems):
    MCMCSmpl_lmbd.append([])
for v in Llmbd:
    for i in range(NItems):
        MCMCSmpl_lmbd[i].append(v[i])

MCMCSmpl_mu = []
MCMCSmpl_psi = []
MCMCSmpl_C = []
for i in range(NItems):
    MCMCSmpl_mu.append([])
    MCMCSmpl_psi.append([])

for i in range(NCat - 1):
    MCMCSmpl_C.append([])

for v in Lmu:
    for i in range(NItems):
        MCMCSmpl_mu[i].append(v[i])
for v in Lpsi:
    for i in range(NItems):
        MCMCSmpl_psi[i].append(v[i])
for v in LC:
    for i in range(NCat - 1):
        MCMCSmpl_C[i].append(v[i])

for i in range(NItems):
    MCMCSmpl_lmbd[i].sort()
    MCMCSmpl_mu[i].sort()
    MCMCSmpl_psi[i].sort()
for i in range(NCat - 1):
    MCMCSmpl_C[i].sort()

Med_Lmbd = []
for i in range(NItems):
    Med_Lmbd.append(MCMCSmpl_lmbd[i][int(len(MCMCSmpl_lmbd[i]) / 2.0)])

Med_mu = []
for i in range(NItems):
    Med_mu.append(MCMCSmpl_mu[i][int(len(MCMCSmpl_mu[i]) / 2.0)])
Med_psi = []
for i in range(NItems):
    Med_psi.append(MCMCSmpl_psi[i][int(len(MCMCSmpl_psi[i]) / 2.0)])
Med_C = []
for i in range(NCat - 1):
    Med_C.append(MCMCSmpl_C[i][int(len(MCMCSmpl_C[i]) / 2.0)])
```

```
    print('¥nMedian of Lambdas...')
    for i in range(NItems):
        print("Med_Lambda[{0}] = {1:.3f}".format((i + 1), Med_Lmbd[i]))
    print('¥nMedians of mus...')
    for i in range(NItems):
        print('Med_mu[{0}] = {1:.3f}'.format((i + 1), Med_mu[i]))
    print('¥nMedians of psis...')
    for i in range(NItems):
        print('Med_psi[{0}] = {1:.3f}'.format((i + 1), Med_psi[i]))
    print('¥nMedians of Cs...')
    for i in range(NCat - 1):
        print('Med_C[{0}] = {1:.3f}'.format((i + 1), Med_C[i]))


    f_out.write('¥n¥nMedians of Lambdas...¥n')
    for i in range(NItems):
        f_out.write("Med_Lambda[{0}] = {1:.3f}¥n".format((i + 1), Med_Lmbd[i]))
    f_out.write('¥nMedians of mus...¥n')
    for i in range(NItems):
        f_out.write('Med_mu[{0}] = {1:.3f}¥n'.format((i + 1), Med_mu[i]))
    f_out.write('¥nMedians of psis...¥n')
    for i in range(NItems):
        f_out.write('Med_psi[{0}] = {1:.3f}¥n'.format((i + 1), Med_psi[i]))
    f_out.write('¥nMedians of Cs...¥n')
    for i in range(NCat - 1):
        f_out.write('Med_C[{0}] = {1:.3f}¥n'.format((i + 1), Med_C[i]))



    print('¥nCalculating Rho_cat...')
    calc_rhos = CalcRhoPreRhoYYP( NCat, NItems, Med_mu, Med_Lmbd, Med_psi, Med_C )
    print('¥nEst_Rho_cat by Meds = {0:.3f}'.format(calc_rhos.GetRhoPre()))
    f_out.write('¥nEst_Rho_cat by Meds = {0:.3f}¥n'.format(calc_rhos.GetRhoPre()))

    print('¥nDo you estimate the distribution of Rho_cat?  Y(es) or N(o) = ', end = '')
    ck_est = input()
    if (ck_est[0] == 'Y') or (ck_est[0] == 'y'):
        print('Estimation started.')
        StartPos = 0
        StopPos = 9
        param0 = {'NCat': NCat, 'NItems': NItems, 'StartPos': StartPos, 'StopPos': StopPos,
                  'LSamplesID': LSamplesID, 'Llmbd': Llmbd, 'Lmu': Lmu, 'Lpsi': Lpsi, 'LC':
LC}
        n_proc = 4;
        params = []
        NEstRhos = None
        if (RawNSamples > 100):
            NEstRhos = 100
        else:
            NEstRhos = RawNSamples

        for iproc in range(n_proc):
            param_t = param0.copy()
            param_t['StartPos'] = int(iproc * NEstRhos / n_proc)
            param_t['StopPos'] = int(((iproc + 1) * NEstRhos / n_proc) - 1)
            params.append(param_t)

        pool = Pool()
        Results = pool.map(EstDistriRho_cat, params)
```

```
    SamplesRho_cat = []
    for u in Results:
        for v in u:
            SamplesRho_cat.append(v)

    sum = 0.0
    for v in SamplesRho_cat:
        sum += v
    meanRho_cat = sum / len(SamplesRho_cat)
    SamplesRho_cat.sort()

    medRho_cat = SamplesRho_cat[int(len(SamplesRho_cat) / 2)]
    Q1Rho_cat = SamplesRho_cat[int(len(SamplesRho_cat) / 4)]
    Q3Rho_cat = SamplesRho_cat[int(len(SamplesRho_cat) * 3 / 4)]

    print('¥nMean of Rho_cat = {0:.3f}'.format(meanRho_cat))
    print('Median of Rho_cat = {0:.3f}'.format(medRho_cat))
    print('[Q1, Q3] = [{0:.3f}, {1:.3f}]'.format(Q1Rho_cat, Q3Rho_cat))

    f_out.write('¥n¥nMean of Rho_cat = {0:.3f}¥n'.format(meanRho_cat))
    f_out.write('Median of Rho_cat = {0:.3f}¥n'.format(medRho_cat))
    f_out.write('[Q1, Q3] = [{0:.3f}, {1:.3f}]¥n'.format(Q1Rho_cat, Q3Rho_cat))

    title_str = 'Mean of Rho_cat = {0:.3f}¥n' + ¥
                    '(Q1, Med, Q3) = ({1:.3f}, {2:.3f}, {3:.3f})'
    title_str = title_str.format(meanRho_cat, Q1Rho_cat, medRho_cat, Q3Rho_cat)
    plt.title(title_str)
    plt.hist(SamplesRho_cat, bins = 10)
    plt.show()


f_out.close()
```

引用文献

Okamoto, Y. (2016). Reliability coefficients for ordinal categorical items: Actual relation of observed and true scores. *Japan Women's University Journal: Faculty of Integrated Arts and Social Sciences*, 2016, 27, 113-122.

Okamoto, Y. (2017). Bayesian estimation of ordinal categorical items' reliability coefficients: Relationship between true values and observed values. *Japan Women's University Journal: Faculty of Integrated Arts and Social Sciences*, 2017, ??, ??????.