（一社）日本官能評価学会ワークショップ

ベイズ統計学と官能評価

第２回： Stanと３点識別試験
（ロジスティック回帰モデルの試み）

（日本女子大学）
岡本 安晴

ベイズ統計学の基本的考え方

$$P(\theta, D) = P(\theta)P(D|\theta)$$

同時確率　　事前分布　　データ生成モデル
尤度

事後分布

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

Bayes' rule

$\propto P(\theta)P(D|\theta)$

$P(D)$　定数

$\approx P(D|\theta)$

$P(\theta) \approx$ 定数のとき

事後分布 $P(\theta|D)$ を求める

確率分布は数学的解析が難しい場合でも
乱数を用いて推定することができる

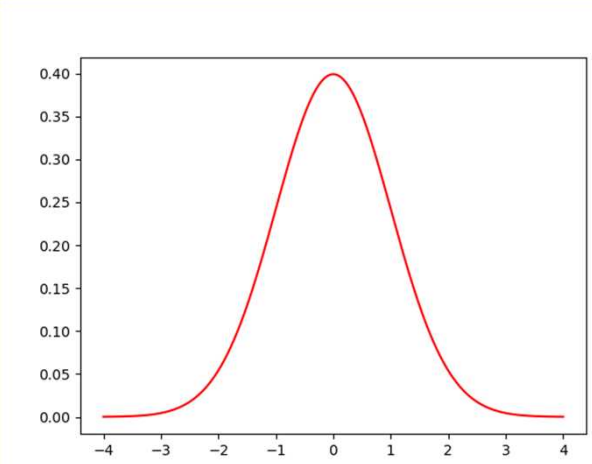
事後分布用の乱数生成法：

Markov chain Monte Carlo (MCMC) サンプリング

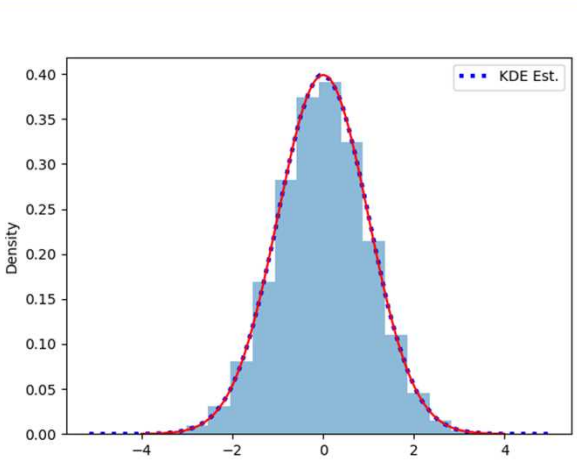
（疑似）乱数

```
(base) PS V:¥xxxxx> cat random_1.py
x = 19
i = 0
while True:
    x = (7 * x) % 31
    print(x, end = ' ')
    i += 1
    if i > 20:
        break
(base) PS V:¥xxxxx> python random_1.py
9 1 7 18 2 14 5 4 28 10 8 25 20 16 19 9 1 7 18 2
14
(base) PS V:¥xxxxx>
```

確率分布の乱数による推定



標準正規分布




```
import scipy.stats as ss; import matplotlib.pyplot as plt
import seaborn as sb; import numpy as np
xcoord = np.linspace(-4, 4, 1000); ycoord = ss.norm.pdf(xcoord)
plt.plot(xcoord, ycoord, c = 'r')
samples = ss.norm(0.0, 1.0).rvs(size = 1_000_000)
plt.hist(samples, bins = 20, density = True, alpha = 0.5)
sb.kdeplot(samples, color = 'b', linestyle = ':', lw = 3, label = 'KDE Est.')
plt.show()
```

2項分布のベイズ分析

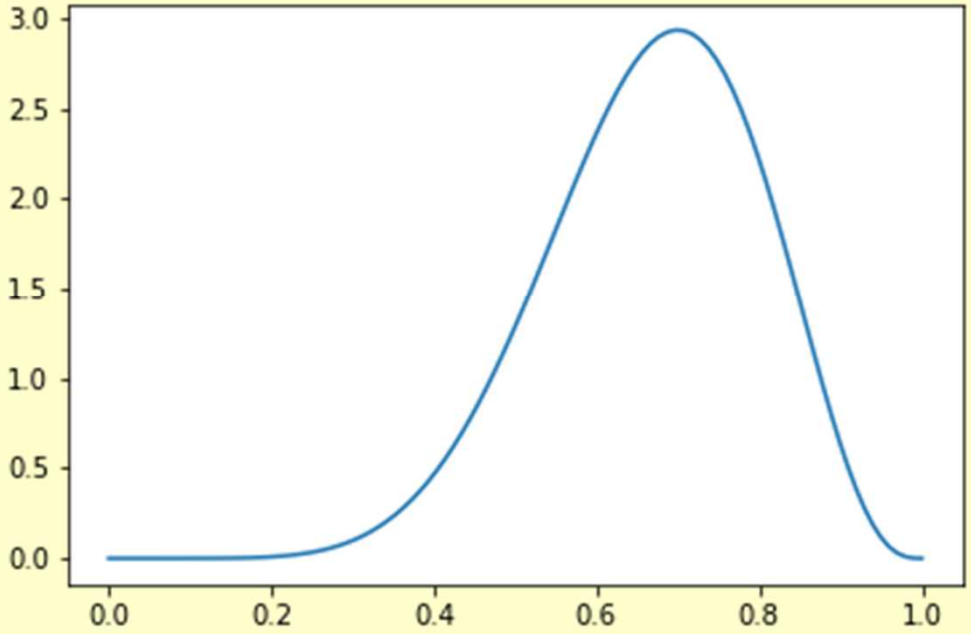
成功確率がθ、失敗確率が$1-\theta$である N回の独立な試行において、
成功試行数がk試行、失敗試行数が$N-k$試行

2項分布　　$P(k|\theta) = {}_N C_k \theta^k (1-\theta)^{N-k}$

$P(\theta) = 1$のとき、

$$P(\theta|k) = \frac{P(\theta,k)}{P(k)} = \frac{P(\theta)P(k|\theta)}{P(k)} = Beta(\alpha+k, \beta+N-k)$$

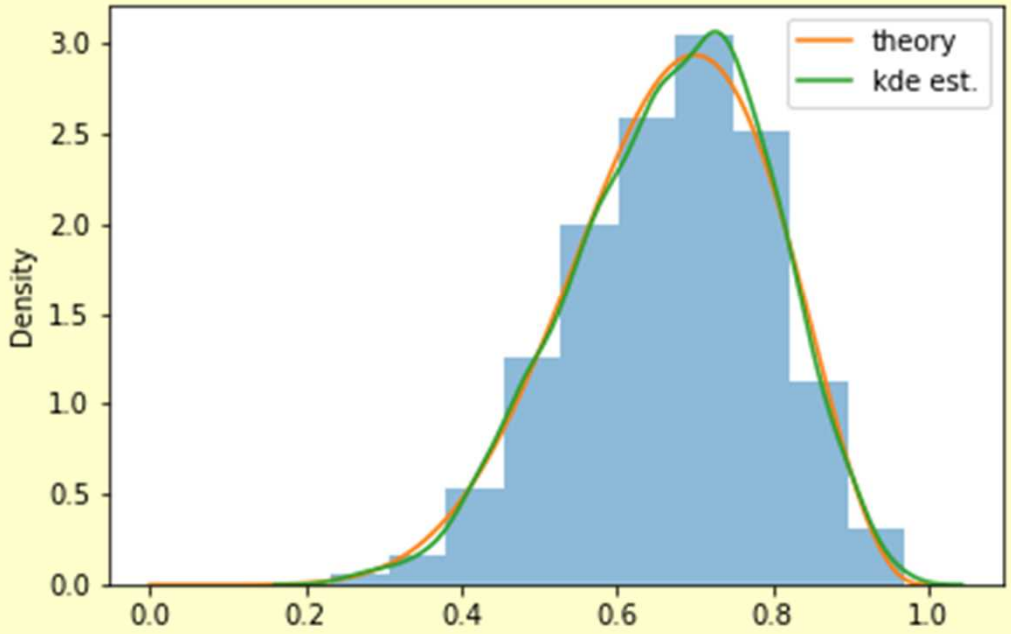
```
import scipy.stats as ss
import numpy as np
import matplotlib.pyplot as plt

thetas = np.linspace(0, 1, 100)
#   k = 7,  N = 10,  N-k = 3
ps = ss.beta.pdf(thetas, 1+7, 1+3)
plt.plot(thetas, ps)
plt.savefig('BinExact.png')
plt.show()
```



StanのMCMCサンプリングによる乱数生成

```
import scipy.stats as ss; import numpy as np; import pystan
import matplotlib.pyplot as plt; import seaborn as sb

thetas = np.linspace(0, 1, 100)
#   k = 7,  N = 10,  N-k = 3
ps = ss.beta.pdf(thetas, 1+7, 1+3)

Code_Bin = """
   data { int N; int k; }
   parameters { real<lower = 0.0, upper = 1.0> theta; }
   model {
      theta ~ uniform(0.0, 1.0);  k ~ binomial(N, theta);
   }
"""
sm = pystan.StanModel(model_code = Code_Bin)
fit = sm.sampling(data = {'N':10, 'k':7}, n_jobs = 1)
plt.hist(fit['theta'], density = True, alpha = 0.5)
plt.plot(thetas, ps, label = 'theory')
sb.kdeplot(fit['theta'], label = 'kde est.')
plt.legend(); plt.savefig('FigSampleExact.png'); plt.show()
```

MCMCサンプリングとBeta分布



Python用のStan（PyStan）は、Anaconda上での利用が薦められる。
本講習会でのPyStanは、AnacondaにPython 3.9の仮想環境を構築して利用した。

Stanの準備

StanはAnacondaの仮想環境でインストールするのが簡単である。

Anacondaのインストールと使い方の説明ウェブサイト

http://y-okamoto-psy1949.la.coocan.jp/Python/tips/UsingPython/#BM_InstAna

AnacondaでのPyStan（Python用のStan）のインストールの説明ウェブサイト

http://y-okamoto-psy1949.la.coocan.jp/bayesian/pystan/

Stanの説明

参考資料
岡本安晴「いまさら聞けないPythonでデータ分析」丸善出版、第10章
http://y-okamoto-psy1949.la.coocan.jp/booksetc/pyda/

Stanスクリプトの構成

https://mc-stan.org/docs/2_28/reference-manual/index.html

関数宣言部：functions { … }
データ宣言部：data { … }
変換データ部：transformed data { … }
パラメータ宣言部： parameters { … }
変換パラメータ部： transformed parameters { … }
モデル宣言部： model { … }

各宣言部・変換部内のコードの記述は、
C／C++の簡易版である。配列の添え字は、C/C++では
０から始まるが、Stanでは１からである。
各宣言部・変換部は、必要ない場合は省いてよい。

データの型

整数 int　　実数 real　　など

ベクトル　　vector　row_vector　など

simplex[3]　s;　　s[i] >= 0.0
s[1] + s[2] + s[3] = 1

行列　matrix

配列　　real[2] a[3];

a[1][1]	a[1][2]
a[2][1]	a[2][2]
a[3][1]	a[3][2]

値（変数）~　確率分布

左辺の値が右辺の確率分布に従う

for (k in 1:N) {　文　}

while (条件) {　文　}

```
if (条件)
    文
```

```
if (条件)
    文
else
    文
```

```
If (条件)
  文
else if (条件)
  文
```

事後分布に従う乱数の生成

Stan

$$P(\theta|D) \propto P(\theta)P(D|\theta)$$

$P(\theta)$と$P(D|\theta)$を記述 ⟶ $P(\theta|D)$に従う乱数のサンプリング

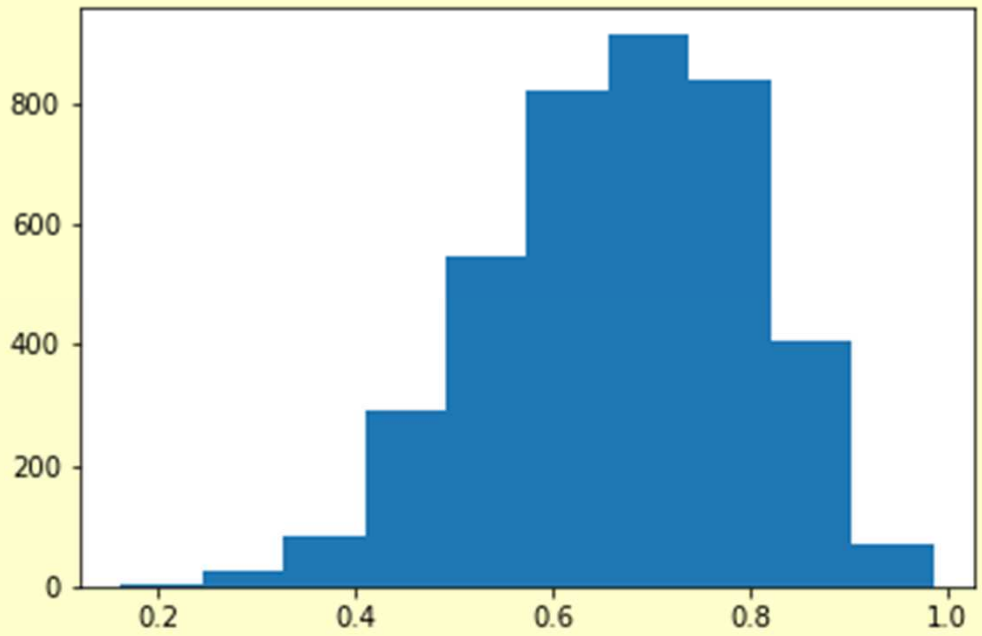
```
data {
   int N;
   int k;
}
parameters {
   real<lower = 0.0, upper = 1.0> theta;
}
model {
   theta ~ beta(1.0, 1.0);
   k ~ binomial(N, theta);
}
```

D

$P(\theta)$

$P(D|\theta)$

```
import pystan; import numpy as np
import matplotlib.pyplot as plt
Stan_Code = """
   data {
      int N;
      int k;
   }
   parameters {
      real<lower = 0.0, upper = 1.0> theta;
   }
   model {
      theta ~ beta(1.0, 1.0);
      k ~ binomial(N, theta);
   }
"""
sm = pystan.StanModel(model_code = Stan_Code)
fit = sm.sampling(data = {'N':10, 'k':7}, n_jobs = 1)
print(fit)
plt.hist(fit['theta'])
plt.savefig('FigStanSample.png'); plt.show()
```


800
600
400
200
0
0.2
0.4
0.6
0.8
1.0

Stanスクリプトのコンパイル Pythonスクリプト

sm = pystan.StanModel(model_code = スクリプト文字列)

sm = pystan.StanModel(file = Stanスクリプトファイル名)

MCMCサンプリング実行 Pythonスクリプト

fit = sm.sampling(data = データ, n_jobs = 1)

Windowsでは、n_jobs=1は必須

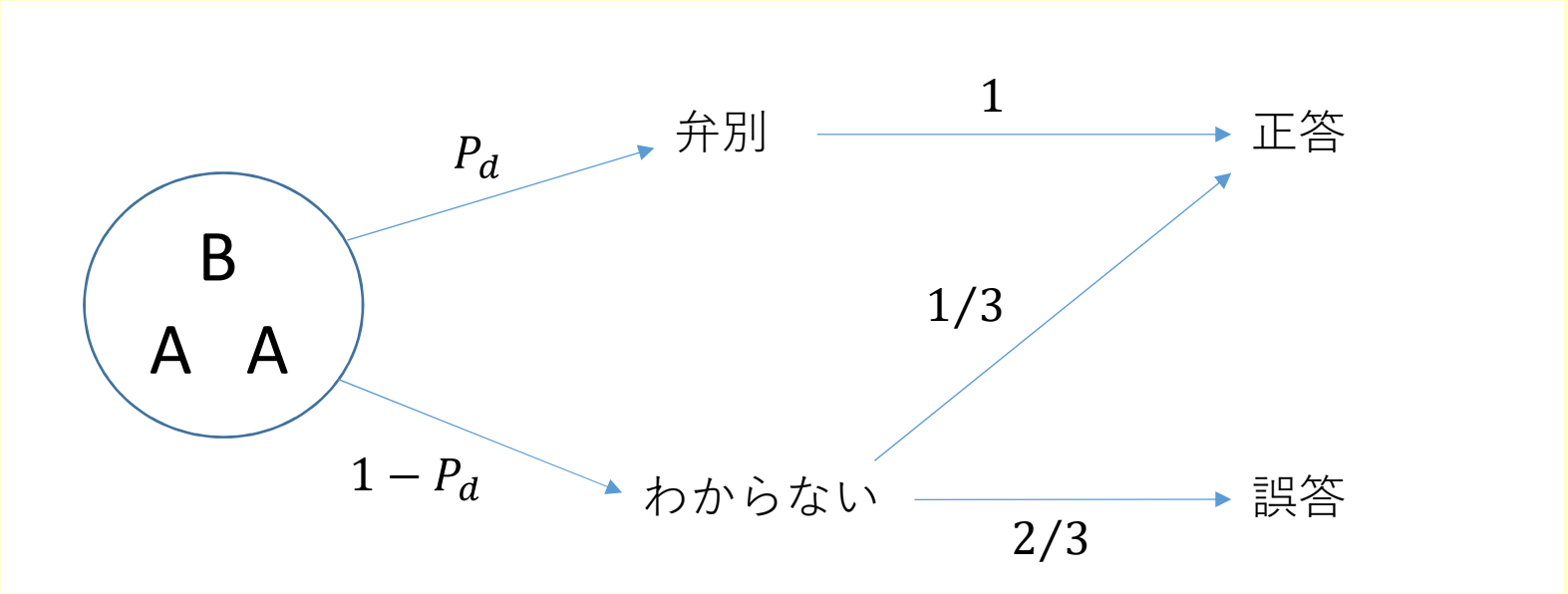
PyStanのクラスStanModelとメソッド（関数）samplingの説明

https://pystan2.readthedocs.io/en/latest/api.html

class **pystan.StanModel**(*file=None, charset='utf-8', model_name='anon_model', model_code=None, stanc_ret=None, include_paths=None, boost_lib=None, eigen_lib=None, verbose=False, obfuscate_model_name=True, extra_compile_args=None, allow_undefined=False, include_dirs=None, includes=None*) [source]

sampling(*data=None, pars=None, chains=4, iter=2000, warmup=None, thin=1, seed=None, init='random', sample_file=None, diagnostic_file=None, verbose=False, algorithm=None, control=None, n_jobs=-1, **kwargs*) [source]

3 点識別試験



正答確率 $P_c = P_d \times 1 + (1 - P_d) \times \frac{1}{3}$

ノンパラメトリックモデル

d'： パラメトリックモデル

N試行（人）中、正答 k 試行（人）である確率

$$P(P_d, k) = P(P_d)P\left(k\middle|P_d\right)$$

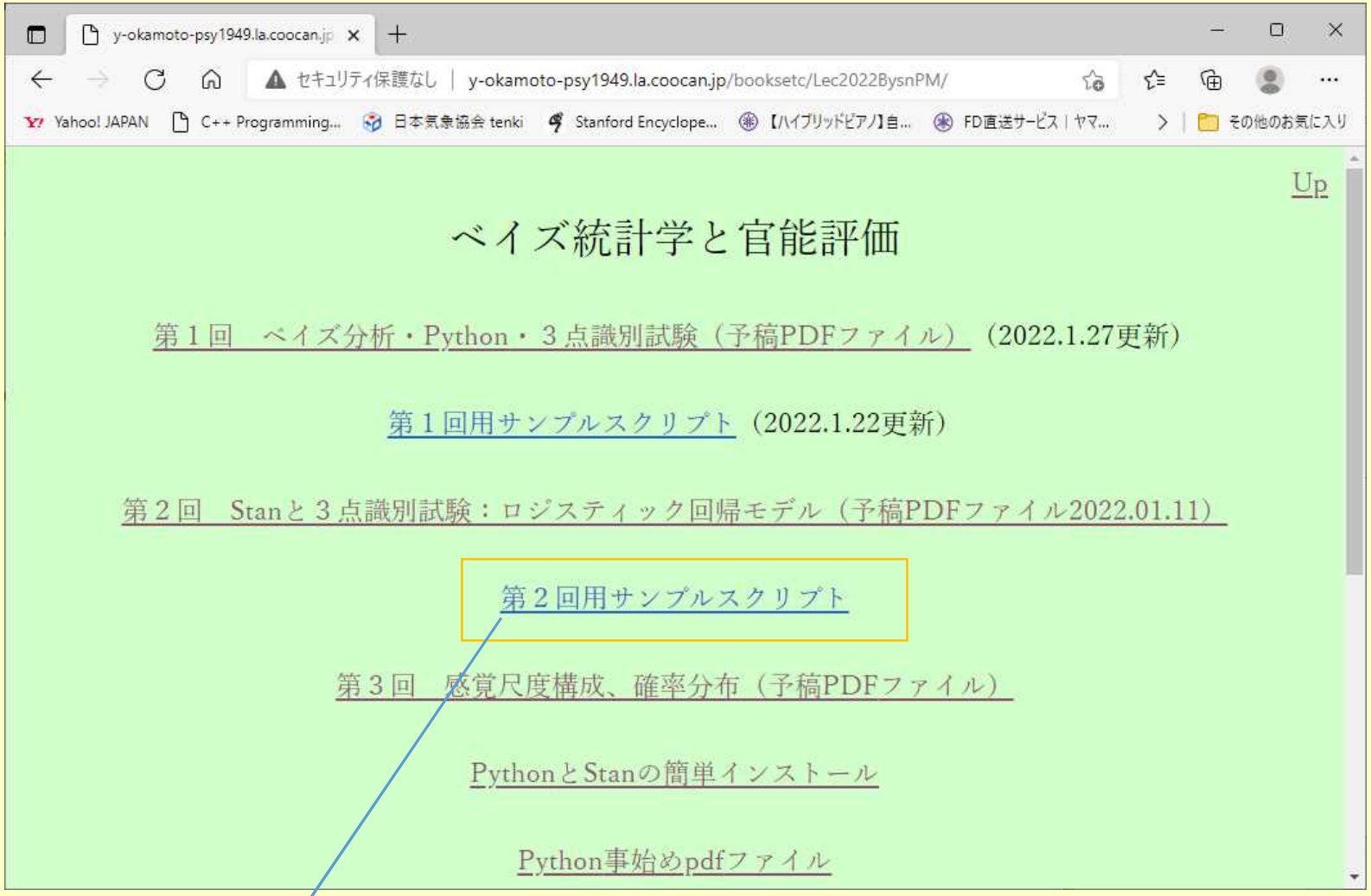
$$P(P_d)=1, \quad 0 \leq P_d \leq 1$$

$$P_c = P_d \times 1 + (1 - P_d) \times \frac{1}{3}$$

$$P(k|N) = {}_N C_k {P_c}^k (1 - P_c)^{N-k} = P\left(k\middle|P_d\right)$$

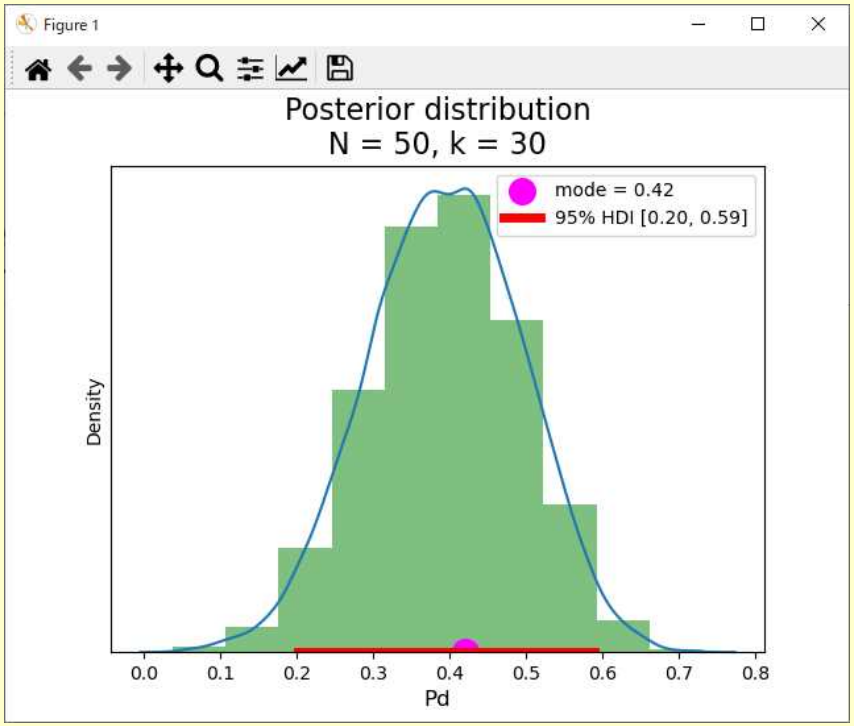
```
data {    int N;    int k;    }
parameters {    real<lower = 0.0, upper = 1.0> Pd;    }
transformed parameters {    real Pc;   Pc = Pd + (1.0 - Pd) * (1/3.0);    }
model {    Pd ~ uniform(0.0, 1.0);    k ~ binomial(N, Pc);    }
```

```
N = int(input('N = '))
k = int(input('k = '))
Data = {'N': N, 'k': k}
sm = pystan.StanModel(file = 'tri_test.stan')
fit = sm.sampling(data = Data, iter = 20_000, n_jobs = 1)
print(fit)
sb.kdeplot(fit['Pd'])
plt.hist(fit['Pd'], density = True, color = 'g', alpha = 0.5)
MAPEst = CalcMAPEst(fit['Pd'])[0]
plt.scatter([MAPEst], [0], marker = 'o', s = 200,
          c = 'magenta', label = f'mode = {MAPEst:.2f}')
L_bndry, R_bndry = calcHDI(fit['Pd'], 0.05)
plt.plot([L_bndry, R_bndry], [0, 0], c = 'r', lw = 5,
        label = '95% HDI [{0:.2f}, {1:.2f}]'.format(L_bndry, R_bndry))
plt.yticks([])
plt.xlabel('Pd', fontsize = 12)
plt.title(f'Posterior distribution¥nN = {N}, k = {k}', fontsize = 16)
plt.legend()
plt.savefig('FigPd.png')
plt.show()
```



samples_2.zip

(py39) PS V:¥XXXXX¥samples_2> python TriTest.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
N = 50
k = 30
INFO:pystan:COMPILING THE C++ CODE FOR MODEL
anon_model_1a0cebab3a4d4923fab357ad23b08ab6 NOW.

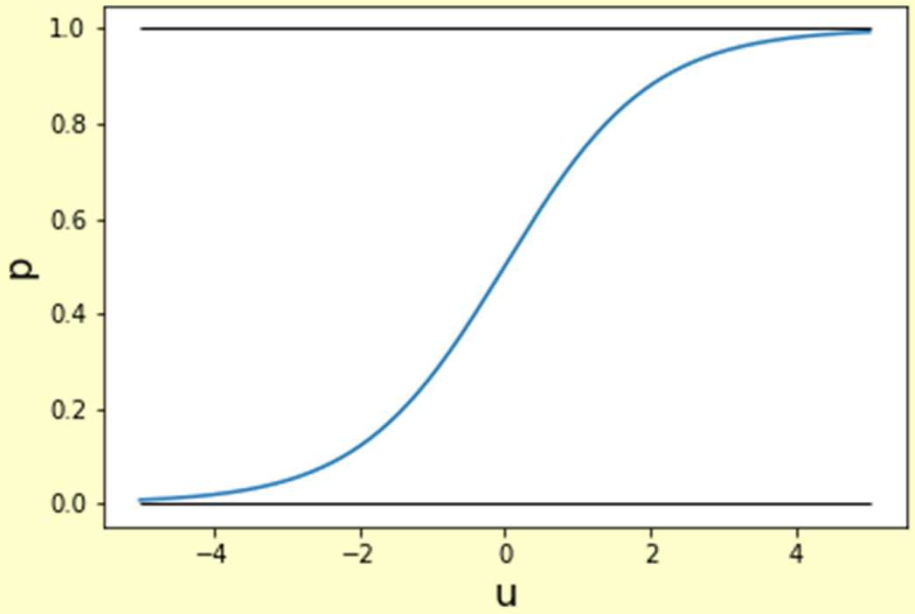


ロジスティック回帰モデル

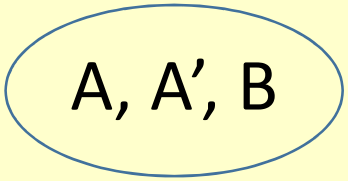
$$\mathrm{u} = \mathrm{logit}(p) = \log\frac{p}{1-p}$$ ロジット変換

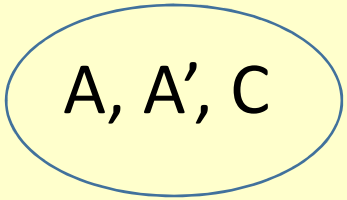
$$p = logit^{-1}(u) = \frac{1}{1+exp(-u)}$$ 逆ロジット変換

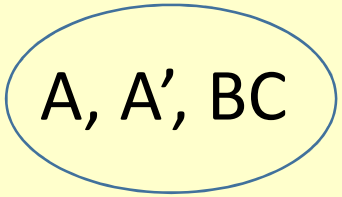
$$0 < p < 1, \qquad -\infty < u < +\infty$$



```
import numpy as np
import matplotlib.pyplot as plt

def inv_logit(u):
    return 1.0 / (1.0 + np.exp(-u))

x_coord = np.linspace(-5, 5, 1001)
y_coord = inv_logit(x_coord)

plt.plot(x_coord, y_coord)
plt.hlines(0, -5, 5, colors = 'k', lw = 1)
plt.hlines(1, -5, 5, colors = 'k', lw = 1)
plt.xlabel('u', fontsize = 16)
plt.ylabel('p', fontsize = 16)
plt.savefig('DrawLogit.png')
plt.show()
```







刺激B、C、BCの弁別確率PdB、PdC、PdBCを、刺激Bの主効果fB、刺激Cの主効果fC、交互採用fBCを設定して次式のように表す。

$$\mathrm{PdB} = 1/\left(1 + \exp(-\mathrm{fB})\right)$$

$$\mathrm{PdC} = 1/\left(1 + \exp(-\mathrm{fC})\right)$$

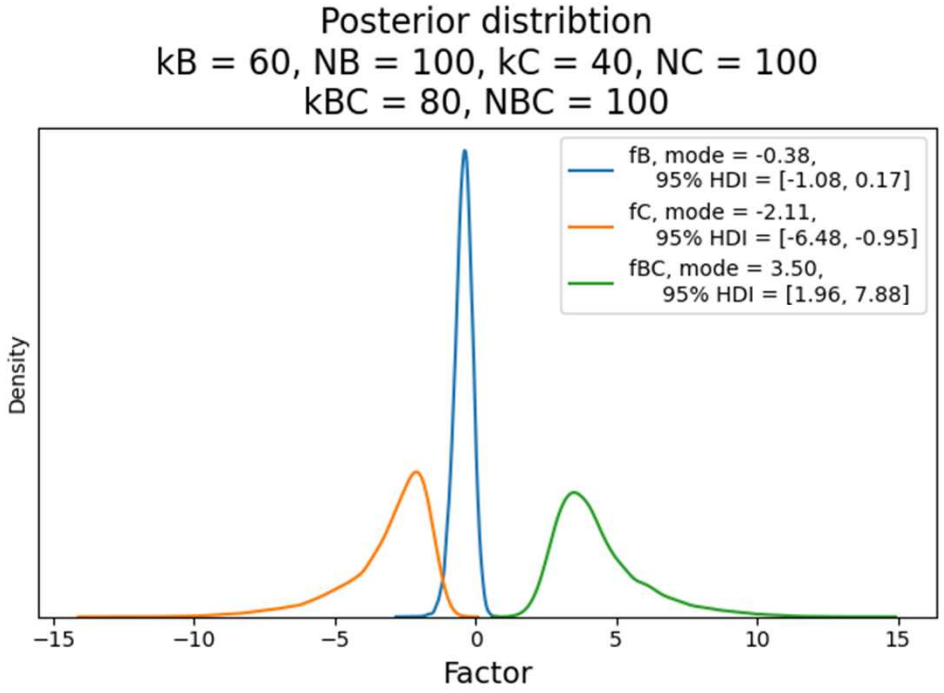
$$\mathrm{PdBC} = 1/\left(1 + \exp\left(-(\mathrm{fB} + \mathrm{fC} + \mathrm{fBC})\right)\right)$$

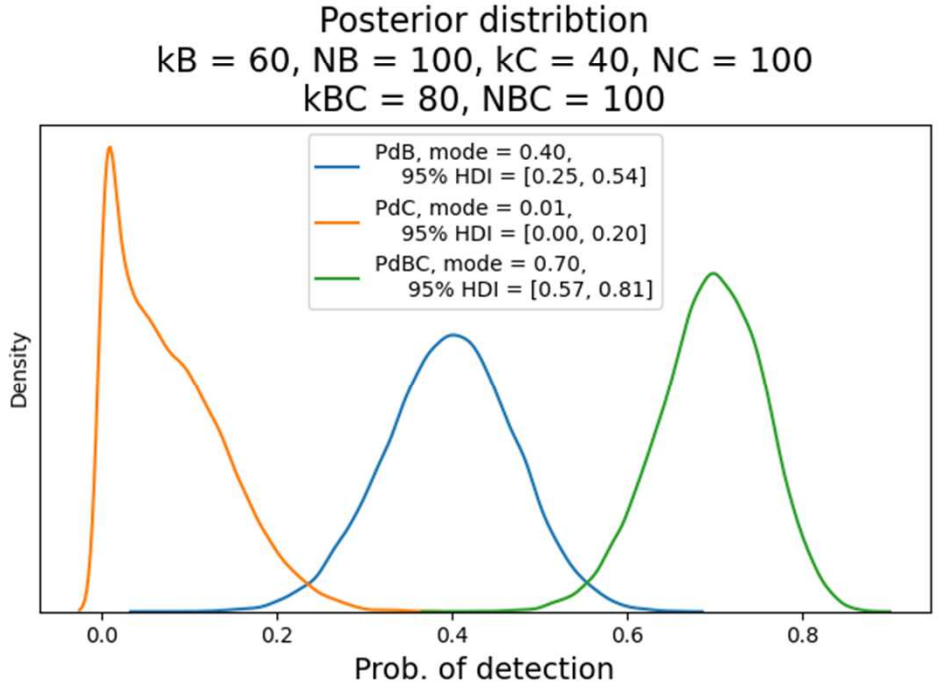
Stanスクリプト

```
data {
  int NB;   int kB;   int NC;   int kC;   int NBC;   int kBC;
}
parameters {  real fB;   real fC;   real fBC;  }
transformed parameters {
  real<lower = 0.0, upper = 1.0> PdB;   real<lower = 0.0, upper = 1.0> PdC;
  real<lower = 0.0, upper = 1.0> PdBC;  real<lower = 0.0, upper = 1.0> PcB;
  real<lower = 0.0, upper = 1.0> PcC;   real<lower = 0.0, upper = 1.0> PcBC;

  PdB = 1.0 / (1.0 + exp(-fB));   PdC = 1.0 / (1.0 + exp(-fC));
  PdBC = 1.0 / (1.0 + exp(-(fB + fC + fBC)));

  PcB = PdB + (1.0 - PdB) * (1.0 / 3.0);   PcC = PdC + (1.0 - PdC) * (1.0 / 3.0);
  PcBC = PdBC + (1.0 - PdBC) * (1.0 / 3.0);
}
model {
  fB ~ normal(0.0, 5.0);   fC ~ normal(0.0, 5.0);   fBC ~ normal(0.0, 5.0);
  kB ~ binomial(NB, PcB);  kC ~ binomial(NC, PcC);
  kBC ~ binomial(NBC, PcBC);
}
```

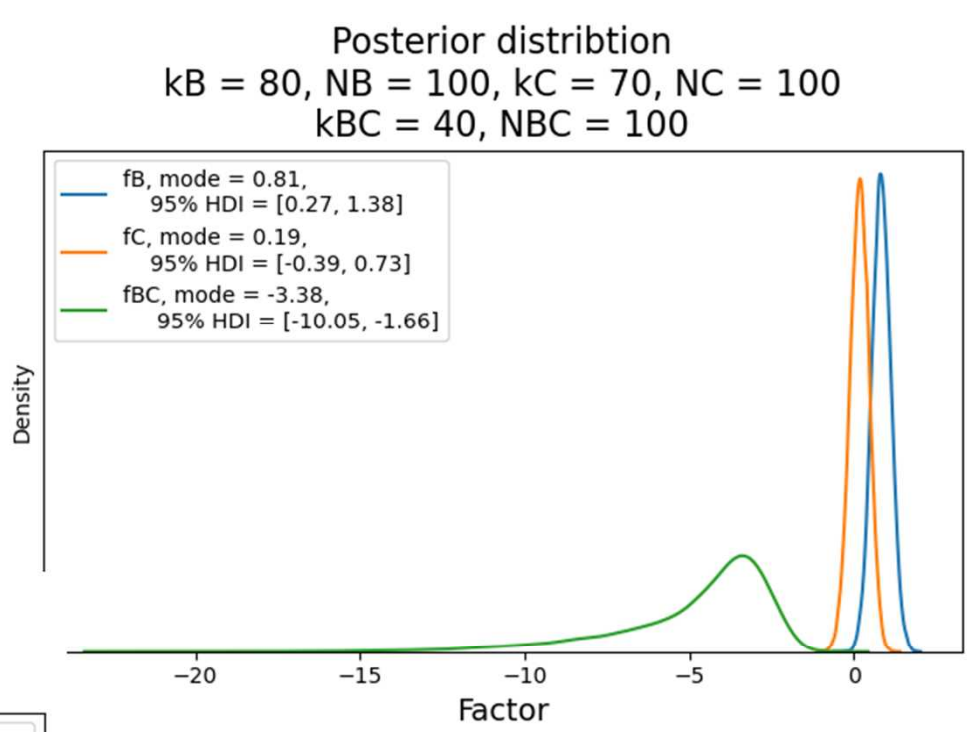
```
kB = int(input('kB = '))
NB = int(input('NB = '))
kC = int(input('kC = '))
NC = int(input('NC = '))
kBC = int(input('kBC = '))
NBC = int(input('NBC = '))


sm = pystan.StanModel(file = 'tri_test_logistic.stan')

Data = {'kB': kB, 'NB': NB, 'kC': kC, 'NC': NC, 'kBC': kBC, 'NBC': NBC}
fit = sm.sampling(data = Data, iter = 10_000, n_jobs = 1)
print(fit)
```

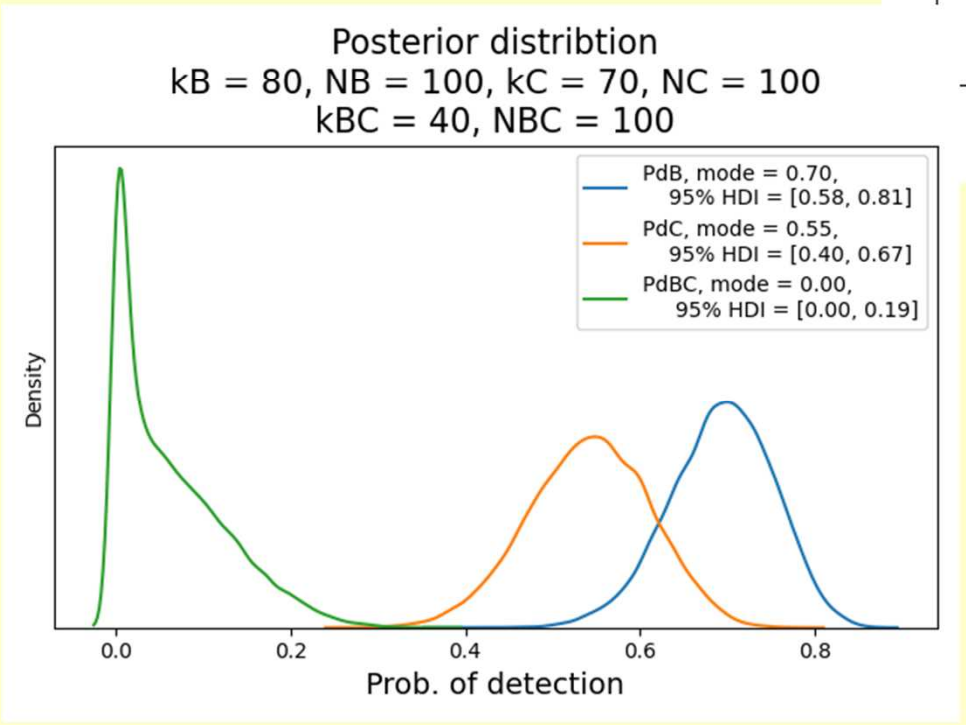
```
(py39) PS V:¥XXXXX¥prgs¥samples_2> python TriTestLogistic.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
kB = 60
NB = 100
kC = 40
NC = 100
kBC = 80
NBC = 100
INFO:pystan:COMPILING THE C++ CODE FOR MODEL
anon_model_4312910542bc7d69288e7e381e1cbb27 NOW.
```


Posterior distribtion
kB = 60, NB = 100, kC = 40, NC = 100
kBC = 80, NBC = 100
fB, mode = -0.38,
95% HDI = [-1.08, 0.17]
fC, mode = -2.11,
95% HDI = [-6.48, -0.95]
fBC, mode = 3.50,
95% HDI = [1.96, 7.88]
Density
−15
−10
−5
0
5
10
15
Factor


Posterior distribtion
kB = 60, NB = 100, kC = 40, NC = 100
kBC = 80, NBC = 100
PdB, mode = 0.40,
95% HDI = [0.25, 0.54]
PdC, mode = 0.01,
95% HDI = [0.00, 0.20]
PdBC, mode = 0.70,
95% HDI = [0.57, 0.81]
Density
0.0
0.2
0.4
0.6
0.8
Prob. of detection

```
(py39) PS V:¥XXXXX¥samples_2> python TriTestLogistic.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
kB = 80
NB = 100
kC = 70
NC = 100
kBC = 40
NBC = 100
INFO:pystan:COMPILING THE C++ CODE FOR MODEL
anon_model_4312910542bc7d69288e7e381e1cbb27 NOW.
```


Posterior distribtion
kB = 80, NB = 100, kC = 70, NC = 100
kBC = 40, NBC = 100
fB, mode = 0.81,
95% HDI = [0.27, 1.38]
fC, mode = 0.19,
95% HDI = [-0.39, 0.73]
fBC, mode = -3.38,
95% HDI = [-10.05, -1.66]
Density
−20
−15
−10
−5
0
Factor


Posterior distribtion
kB = 80, NB = 100, kC = 70, NC = 100
kBC = 40, NBC = 100
PdB, mode = 0.70,
95% HDI = [0.58, 0.81]
PdC, mode = 0.55,
95% HDI = [0.40, 0.67]
PdBC, mode = 0.00,
95% HDI = [0.00, 0.19]
Density
0.0
0.2
0.4
0.6
0.8
Prob. of detection

識別確率の年齢による変化

ロジット変換は次式で表され、

$$\mathrm{u} = \mathrm{logit}(p) = \log\frac{p}{1-p}$$

これより、次式を得る。

$$p = \frac{1}{1+exp(-u)} = logit^{-1}(u)$$

uを年齢をAの1次式で表せば、

$$\mathrm{u} = \mathrm{a} + \mathrm{bA}$$

である。
このとき、識別確率Pdは次式で与えられる。

$$\mathrm{Pd} = logit^{-1}(a + bA)$$

正答確率Pcは

$$\mathrm{Pc} = \mathrm{Pd} + (1 - \mathrm{Pd}) \times \frac{1}{3}$$

である。

uを年齢Aの1次式で表した次式、

$$\mathrm{u} = \mathrm{a} + \mathrm{bA}$$

を設定すればよい。

　上の式の場合、独立変数は１個A_iであるが、２個以上の場合も同様である。

　例えば、実験の実施時間、朝、昼、晩、を区別するときは、

$$T_i = \text{朝、昼、晩}$$

と置いてカテゴリ変量を含む場合の回帰モデルを設定すればよい。このとき、

$$\text{データ}i = (R_i, A_i, T_i)$$

である。

Stanスクリプト

```
data {
   int N;
   int<lower = 0, upper = 1> Rs[N];
   real As[N];
}
parameters {
   real a;    real b;
}
transformed parameters {
   real <lower = 0.0, upper = 1.0> Pd[N];
   real<lower = 0.0, upper = 1.0> Pc[N];  //  Probability of correct response
   for (i in 1:N) {
      Pd[i] = inv_logit(a + b * As[i]);  //  Logistic regression model
      Pc[i] = Pd[i] + (1.0 - Pd[i]) * (1.0/3.0);
   }
}
model {
   a ~ normal(0.0, 10.0);  b ~ normal(0.0, 10.0);    //   weakly informative priot
   Rs ~ bernoulli(Pc);
}
```

仮想データ：　(正答(1)／誤答(0), 年齢)

```
Data = (
  (1,33),(0,20),(0,42),(1,20),(0,44),(1,23),(0,54),(1,24),(0,59),(0,49),
  (0,42),(1,35),(0,49),(0,54),(0,59),(1,20),(0,33),(0,41),(1,59),(1,34),
  (1,50),(1,52),(1,24),(1,22),(1,20),(1,50),(1,53),(1,32),(1,57),(0,48),
  (1,47),(1,35),(1,21),(1,56),(1,58),(0,36),(1,49),(1,23),(1,24),(1,36),
  (1,38),(1,48),(0,59),(1,23),(1,46),(1,32),(1,36),(1,58),(0,47),(1,31),
  (0,59),(0,50),(1,33),(0,55),(1,28),(1,53),(1,36),(1,45),(1,52),(0,49),
  (0,53),(1,36),(1,58),(1,33),(0,40),(0,43),(0,50),(1,30),(1,58),(1,32),
  (1,29),(1,56),(0,34),(0,54),(1,40),(0,46),(1,43),(1,20),(1,27),(1,23),
  (0,56),(1,28),(1,24),(1,44),(0,50),(1,38),(0,41),(0,30),(1,48),(0,37),
  (1,41),(0,38),(0,56),(0,38),(0,20),(1,32),(0,26),(0,35),(1,41),(0,52),
  (1,37),(1,52),(1,26),(0,35),(0,50),(1,29),(1,36),(0,34),(1,32),(1,34),
  (0,58),(1,28),(1,26),(1,32),(1,50),(1,21),(0,25),(0,33),(1,20),(0,59),
  (0,47),(1,42),(1,25),(1,59),(0,47),(0,35),(1,57),(1,24),(1,50),(0,53),
  (1,41),(0,50),(0,59),(1,46),(1,27),(1,40),(0,50),(1,47),(1,53),(1,48),
  (1,37),(1,42),(0,24),(0,44),(1,34),(0,38),(1,43),(1,32),(1,50),(0,39),
  (0,29),(0,51),(0,44),(1,23),(0,21),(1,54),(1,25),(0,49),(0,42),(0,27),
  (0,58),(1,34),(0,59),(1,40),(1,40),(1,35),(1,54),(0,59),(1,37),(0,40),
  (0,53),(1,45),(1,34),(1,42),(1,20),(1,51),(1,34),(1,21),(1,37),(1,43),
  (1,34),(1,29),(0,59),(0,47),(1,25),(1,24),(1,33),(1,27),(1,32),(1,31),
  (1,54),(1,48),(0,34),(0,59),(1,33),(0,49),(1,42),(1,31),(0,51),(0,54)
  )
```

N個のデータ (R_1, A_1)、・・・、(R_N, A_N) は、
1人の判断者から１個のデータを得る場合と、
各判断者から複数のデータを得る場合が考えられる。

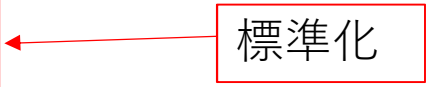
1人の判断者から１個のデータを得る場合：

N人の判断者を集め、各判断者に判断を求める

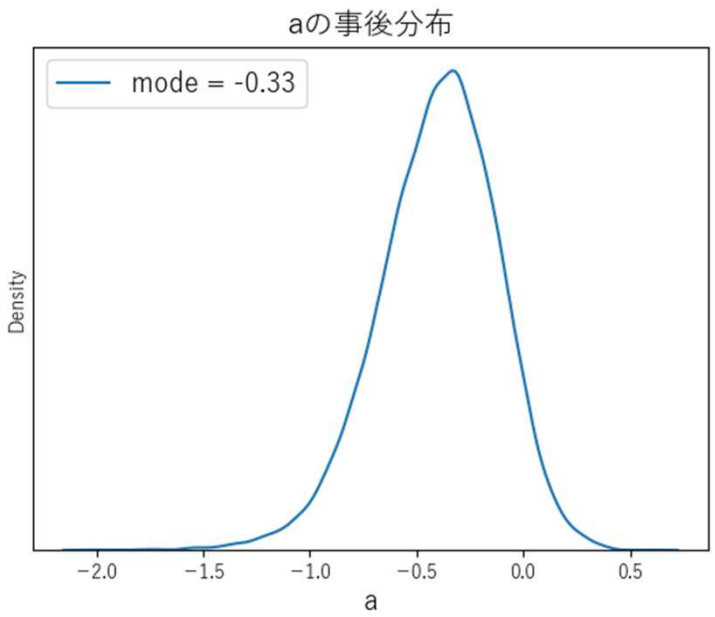
1人の判断者から１個以上のデータを得る場合：

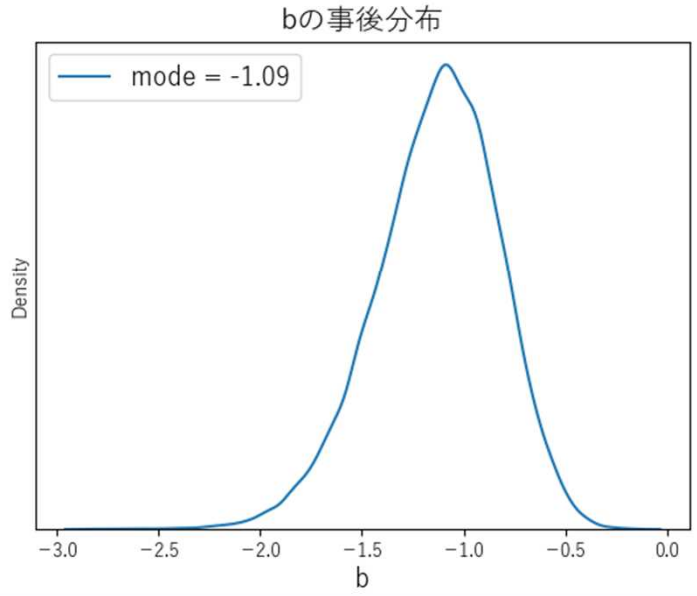
M人の判断者に対して、各判断者に１回以上の判断を求める。
各判断は、独立な試行とする。

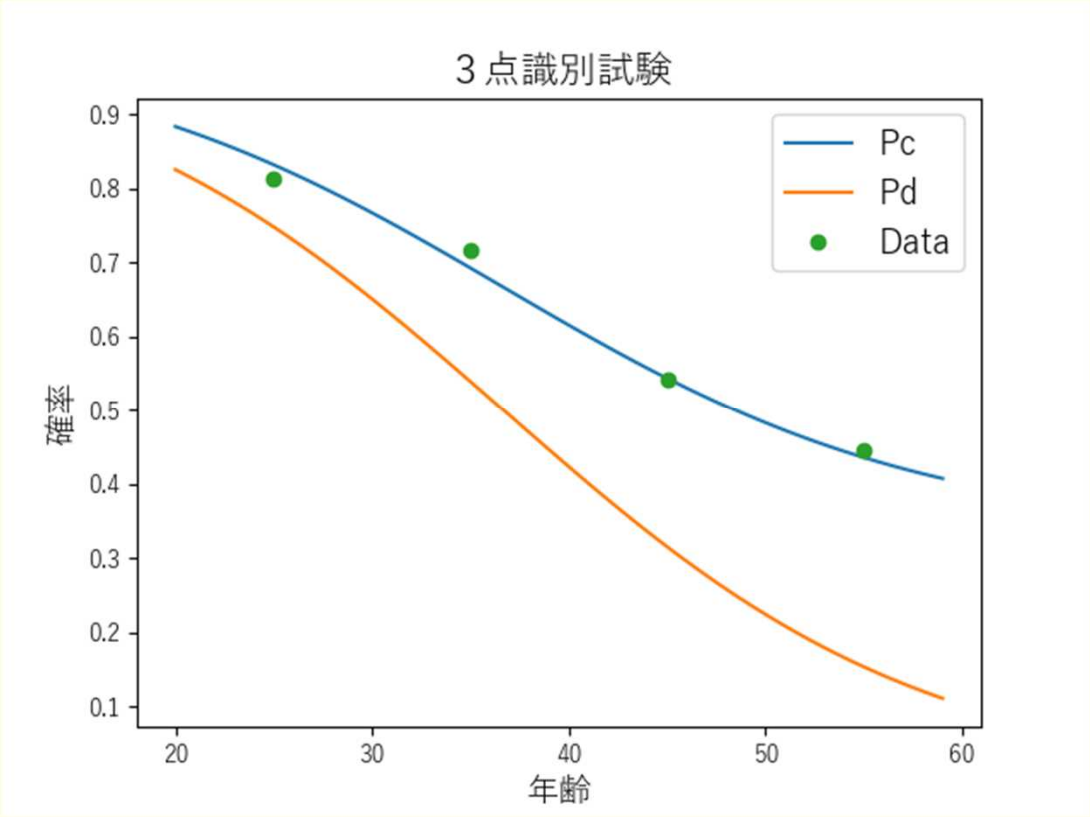
```
N = len(Data)
print(N)

Rs = np.array(Data).T[0]
As = np.array(Data).T[1]
Raw_As = As.copy()
As_mean = np.mean(As)
As_sd = np.std(As)
As = (As - As_mean) / As_sd

#      Stanスクリプトのコンパイル
sm = pystan.StanModel(file = 'reg_age.stan')

#       MCMCサンプリング
fit = sm.sampling(data = {'N':N, 'Rs': Rs, 'As': As},
                  pars = ['a', 'b'], iter = 10_000, n_jobs = 1)
print(fit)
```



(py39) PS V:¥XXXXX¥samples_2> python TriLogRegAge.py







パラメータの事後分布のモードをその値として、識別確率Pdと
正答確率Pcのグラフを描いた。
Dataは、年齢を20～29才、30～39才、40～49才、50～59才の
グループに分け、各グループ内の正答率を表す。

単回帰モデルで、データをCSV形式のファイルで用意する場合。

http://y-okamoto-psy1949.la.coocan.jp/Python/misc/TriangularTestPd/#BM_Logistic

d’の回帰モデル

http://y-okamoto-psy1949.la.coocan.jp/Python/misc/TriTestDRgrssn/

練習問題

1．サンプルサイズの比較

(a)　N＝10、k＝5

(b)　N=50、k＝25

(c)　N=100、k＝50

2．正答比率がランダム判断より低い場合

N＝50、k ＝10

3．全判断正答の場合

N＝50、k ＝50

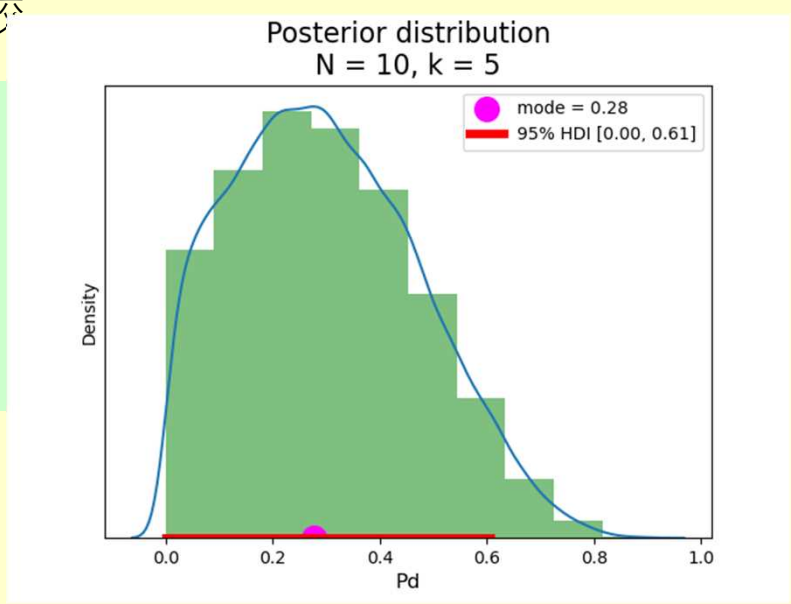
プログラムTriTest.pyを含むファイルsample_2.zipのダウンロード

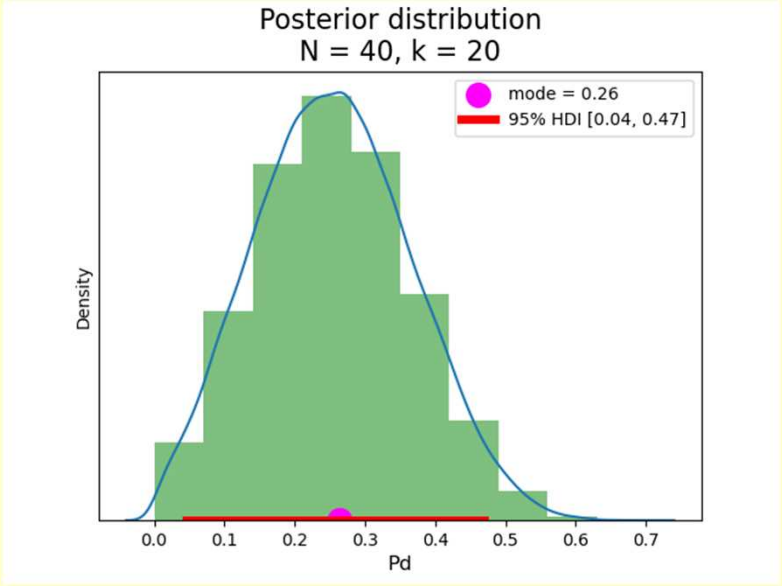
http://y-okamoto-psy1949.la.coocan.jp/booksetc/Lec2022BysnPM/

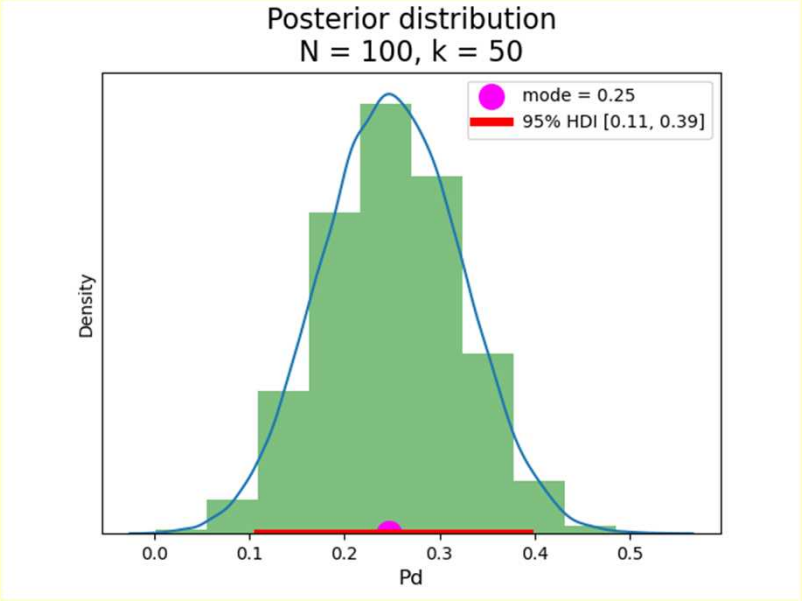
第２回用サンプルスクリプト（sample_2.zip）

1．サンプルサイズの比較

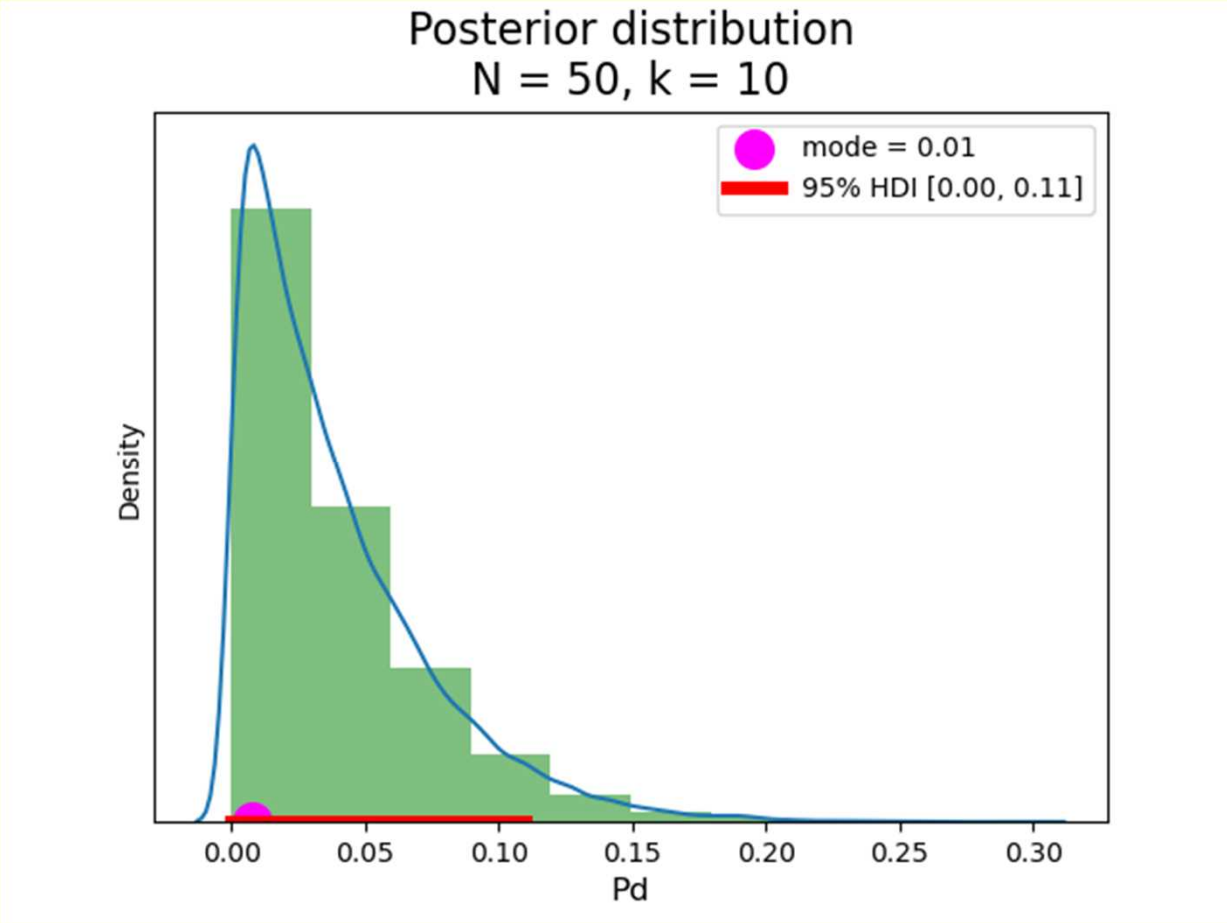
```
(py39) PS V:¥XXXX¥samples_2> python .¥TriTest.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
N = 10
k = 5
INFO:pystan:COMPILING THE C++ CODE FOR MODEL
anon_model_1a0cebab3a4d4923fab357ad23b08ab6 NOW.
```



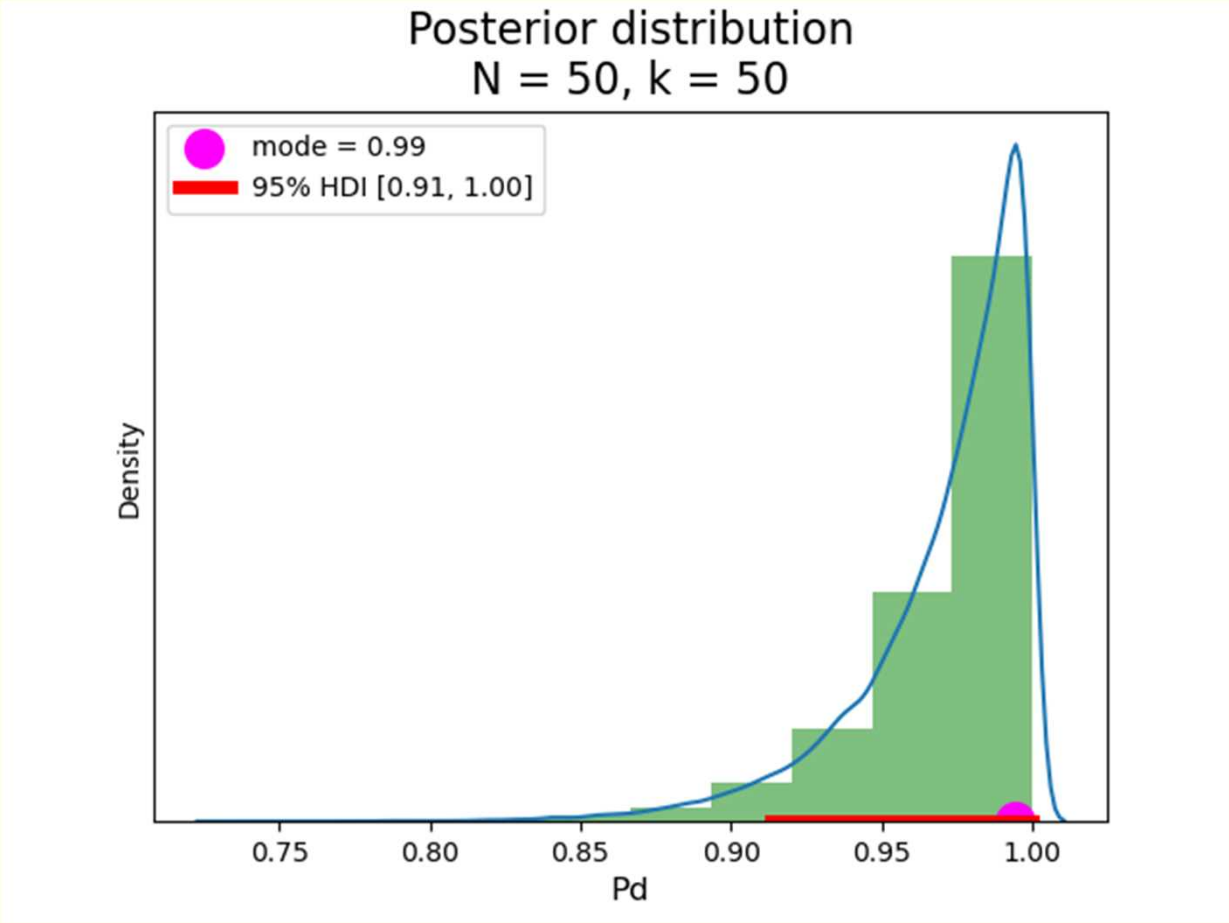




２．正答比率がランダム判断より低い場合



3．全判断正答の場合


Posterior distribution
N = 50, k = 50
mode = 0.99
95% HDI [0.91, 1.00]
Density
0.75
0.80
0.85
0.90
0.95
1.00
Pd

まとめ

1．ベイズ分析のためのソフトはPtyhon用としてPyStanがあり、Anaconda上で簡単にインストールできる。

2．３点識別試験データは、ロジスティック回帰分析を用いると統計学的に統一した分析ができる。

サンプルスクリプトファイル

http://y-okamoto-psy1949.la.coocan.jp/booksetc/Lec2022BysnPM/

次回予告

アンケートの回答

感覚強度の数値化

順序カテゴリ評価法

差尺度構成法

確率分布のいろいろ