Up

Two-Alternative-Forced-Choice Task of Signal Detection Theory with Confidence Rating Model and Bayesian Analysis

Yasuharu Okamoto, 2019

Script and data files are archived in a file SDT_2AFC.zip,and the files for Jupyter Notebook in a file SDT_2AFC_jn.zip. These archived files can be freely downloaded and used.

Python scripts with Stan scripts for Bayesian analyses of data from the standard 2AFC (alternative-forced-choice) task and ones with confidence rating of signal detection theory (SDT) were presented. Klein (2001, p. 1436) recommends using more than two response categories.

On the standard 2AFC task, an observer chooses between the two response categories, e.g. which one is the signal stimulus?. On a 2AFC task with confidence rating, an observer is asked as follows:

In a three rating category task, e.g., "Which one is your judgment?, 'The left stimulus is the signal.', 'Don't' know.', or 'The right stimulus is the signal.'"

In a four rating category task, e.g., "Which one is your judgment?, 'The left stimulus is the signal.', 'Maybe, the left stimulus is the signal.', 'Maybe, the right stimulus is the signal,', or 'The right stimulus is the signal.'"

In the following, models are presented first, then scripts and examples of their uses are shown.

The Basic Model

Sensations of the noise and signal stimuli are denoted by random variables X_N and X_S, which have the following normal distributions:

$$X_N \sim N(\mu_N, \sigma_N^2), \qquad X_S \sim N(\mu_S, \sigma_S^2).$$

To set the origin and unit, it is assumed that

$$\mu_N = 0, \sigma_N = 1.$$

In a 2AFC task, a pair of noise (denoted by N) and signal (denoted by S) stimuli are presented in one of the two conditions, condition <N, S> and condition <S, N>. In conditon <N, S>, N is presented at position 1 and S in position 2. In condition <S, N>, S in position 1 and N in position 2. When positions mean temporal ones, position 1 precedes position 2 in temporal

order. When positions mean spatial ones, position 1 is the left side and position 2 the right side, for example. And so on.

There may be biases in psychophysical judgment. Total effect of the biases is represented by τ, which is measured from position 2.

Hence, in condition <N, S>, sensations have the following distributions

$$X_N \sim N(\tau, 1), \qquad X_S \sim N(\mu_S, \sigma_S^2).$$

In condition <S, N>,

$$X_N \sim N(0, 1), \qquad X_S \sim N(\mu_S + \tau, \sigma_S^2).$$

Psychophysical judgment is assumed to be done based on the difference between the sensations $X_S - X_N$. When the difference is between category boundaries C_{k-1} and C_k, observer chooses category k as his/her judgment. It is assumed that when number of categories is K,

$$-\infty = C_0 < C_1 < \cdots < C_{K-1} < C_K = +\infty.$$

For example, for $K = 3$,

Category 1 means that signal is perceived to be weaker than noise, so wrongly judged to be noise. Category 2 means that I don't know. Category 3 means that signal is perceived to be stronger than noise, so correctly judged to be signal.

In codition <N, S>,

$$X_S - X_N \sim N(\mu_S - \tau, 1 + \sigma_S^2).$$

Denote the probability that rating judgment is category k in codition <N, S> by $P_{NS}(k)$, then we have

$$P_{NS}(k) = P(C_{k-1} < X_S - X_N < C_k) = \Phi\left(\frac{C_k - (\mu_S - \tau)}{\sqrt{1 + \sigma_S^2}}\right) - \Phi\left(\frac{C_{k-1} - (\mu_S - \tau)}{\sqrt{1 + \sigma_S^2}}\right)$$

where $\Phi(z)$ is the cumulative standard normal distribution function.

In codition <S, N>,

$$X_S - X_N \sim N(\mu_S + \tau, 1 + \sigma_S^2)$$

Denote the probability that rating judgment is category k in codition <S, N> by $P_{SN}(k)$, then we have

$$P_{SN}(k) = P(C_{k-1} < X_S - X_N < C_k) = \Phi\left(\frac{C_k - (\mu_S + \tau)}{\sqrt{1 + \sigma_S^2}}\right) - \Phi\left(\frac{C_{k-1} - (\mu_S + \tau)}{\sqrt{1 + \sigma_S^2}}\right)$$

Because the independent variables in function Φ are given in ratios of parameters, the unit must be given to identify parameter values. The popular method is to set $\sigma_S = 1$, i.e. equal-variance model $\sigma_N = \sigma_S = 1$. In this case,

$$1 + \sigma_S^2 = 2.$$

This assumption may be replaced more generally by

$$\sigma_N^2 + \sigma_S^2 = 2.$$

From either of these assumptions, we have

In codition <N, S>, we have

$$P_{NS}(k) = P(C_{k-1} < X_S - X_N < C_k) = \Phi\left(\frac{C_k - (\mu_S - \tau)}{\sqrt{2}}\right) - \Phi\left(\frac{C_{k-1} - (\mu_S - \tau)}{\sqrt{2}}\right).$$

In codition <S, N>, we have

$$P_{SN}(k) = P(C_{k-1} < X_S - X_N < C_k) = \Phi\left(\frac{C_k - (\mu_S + \tau)}{\sqrt{2}}\right) - \Phi\left(\frac{C_{k-1} - (\mu_S + \tau)}{\sqrt{2}}\right).$$

Parameters C_k and μ_S appear as difference, so an assumption to set a origin is necessary. To set an origin, the following constraints are set on category boundaries.

For even number of rating categories $K = 2H, H$ is a positive integer, set

$$C_H = 0$$
$$C_k = -C_{2H-k} \quad 1 \le k < H.$$

For odd number of rating categories $K = 2H + 1, H$ is a positive integer, set

$$C_k = -C_{2H+1-k} \quad 1 \le k \le H.$$

Stan scripts and Python scripts were developed for three cases of number of category boundaries, $K = 2$ i.e. the standard task, K is odd, and K is even(> 2).

In case of $K = 2$:

Stan script in this case is shown in Listing A.1.

Listing A.1　Stan script for K=2 (SDT2AFC2Cat.stan).

```
data {
    int n_NS[2];
    int n_SN[2];
}
parameters {
    real mu_s;
    real tau;
}
transformed parameters {
    simplex[2] p_NS;
    simplex[2] p_SN;
    p_NS[1] = Phi(-(mu_s - tau)/sqrt(2.0));
    p_NS[2] = 1 - p_NS[1];
    p_SN[1] = Phi(-(mu_s + tau)/sqrt(2.0));
    p_SN[2] = 1 - p_SN[1];
}
model {
    mu_s ~ normal(0.0, 100.0);
    tau ~ normal(0.0, 100.0);
    n_NS ~ multinomial(p_NS);
```

```
    n_SN ~ multinomial(p_SN);
}
```

Python script which uses the Stan script above is shown in Listing A.2.

Listing A.2　Python script which uses the Stan script above (SDT2AFC2Cat.py).

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

flnm = input('Input data file (*.csv) = ')
with open(flnm, 'r') as f:
    data = [v for v in csv.reader(f)]

fout_nm = input('Output text file (*.txt) = ')
fout = open(fout_nm, 'w')
fout.write('Input Data File = {}¥n'.format(flnm))

Freq_NS = []
Freq_NS.append(int(data[1][1]))
Freq_NS.append(int(data[1][2]))
Freq_SN = []
Freq_SN.append(int(data[2][1]))
Freq_SN.append(int(data[2][2]))
print(Freq_NS)
print(Freq_SN)
fout.write('<N, S>: {}'.format(Freq_NS))
fout.write('¥n<S, N>: {}'.format(Freq_SN))
TN_NS = sum(Freq_NS)
Rating_NS = np.array(Freq_NS)

TN_SN = sum(Freq_SN)
Rating_SN = np.flip(np.array(Freq_SN))

Data = {'n_NS': Rating_NS, 'n_SN': Rating_SN}

sm = pystan.StanModel(file = 'SDT2AFC2Cat.stan')

fit = sm.sampling(data = Data, n_jobs = 1)

print(fit)
fout.write('¥n{}¥n'.format(fit))

mu = fit['mu_s']
tau = fit['tau']
p_NS = fit['p_NS']
p_SN = fit['p_SN']

mu_L05 = np.percentile(mu, 5)
mu_med = np.percentile(mu, 50)
mu_U95 = np.percentile(mu, 95)
```

```
fout.write("¥nd': ¥nMed. = {0:<.3f},  90%CI = [{1:<.3f}, {2:<.3f}".
           format(mu_med, mu_L05, mu_U95))

tau_L05 = np.percentile(tau, 5)
tau_med = np.percentile(tau, 50)
tau_U95 = np.percentile(tau, 95)
fout.write('¥ntau: ¥nMed. = {0:<.3f},  90%CI ~ [{1:<.3f}, {2:<.3f}]'.
           format(tau_med, tau_L05, tau_U95))

p1 = np.median(p_NS, axis = 0)
p2 = np.median(p_SN, axis = 0)
p2 = [p2[1], p2[0]]

plt.hist(mu)
plt.xlabel("d'($¥mu_S$)", fontsize = 14)
plt.title("Posterior Distribution of d'($¥mu_S$)" + ¥
          '¥nMed. = {0:<.3f},  90%CI = [{1:M<.3f}, {2:<.3f}]'.
          format(mu_med, mu_L05, mu_U95), fontsize = 16)
plt.show()

plt.hist(tau)
plt.xlabel(r'$¥tau$', fontsize = 18)
plt.title(r'Posterior Distribution of $¥tau$' + ¥
          '¥nMed. = {0:<.3f},  90%CI = [{1:<.3f}, {2:<.3f}]'.
          format(tau_med, tau_L05, tau_U95), fontsize = 16)
plt.show()

xcat = [1, 2]
xlabels = ['Cat-1', 'Cat-2']
y1 = np.array(Freq_NS) / TN_NS
y2 = np.array(Freq_SN) / TN_SN

plt.plot(xcat, y1, 'b:', linewidth = 10, label = 'Data/<N, S>')
plt.plot(xcat, p1, 'g-', linewidth = 3, label = 'Model/<N, S>')
plt.plot(xcat, y2, 'r:', linewidth = 10, label = 'Data/<S, N>')
plt.plot(xcat, p2, 'm-', linewidth = 3,label = 'Model/<S, N>')
plt.xticks(xcat, xlabels, fontsize = 14)
plt.xlim(0.8, 2.2)
plt.xlabel('Rating Category', fontsize = 14)
plt.ylabel('Probability/Proportion', fontsize = 14)
plt.title('Rating in Conditions <N, S> and <S, N>', fontsize = 18)
plt.legend(loc = 'upper center', fontsize = 10)
plt.show()

fout.close()
print('¥n' + fout_nm + ' was saved.¥n')
```

Files in Listing A1 and A.2 and a sample data file are archived into a zip file Rating2Cat.zip, which can be down loaded.

Run the above script SDT2AFC2Cat.py, names of the input data file and an output text file are required to be set (Figure A.1).

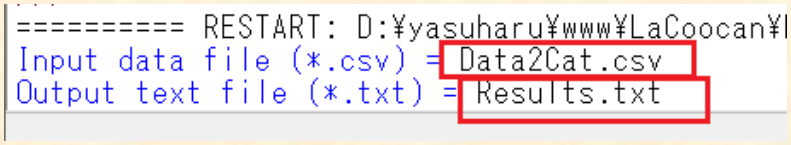

Figure A.1

Name of an output text file is arbitrary. Format of the input data file should be CSV file(Figure .A.2).

	A	B	C	D
1	Cond	Cat_1	Cat_2	
2	N_S	30	70	
3	S_N	84	16	
4				

図 A.2　データファイル Data2Cat.csv

In the first row, names for columns are set. The first column shows conditions, <N, S> and <S, N>, the second column shows frequencies of rating judgment of category 1, and the third column shows frequencies of rating judgment of category 2.

In the second row, data from condition <N, S> are set, and in the third row, data from condition <S, N>.

When the file names are set, the program starts calculation. After sampling by Stan ends, posterior distribution of parameter d' is presented (Figure A.3).

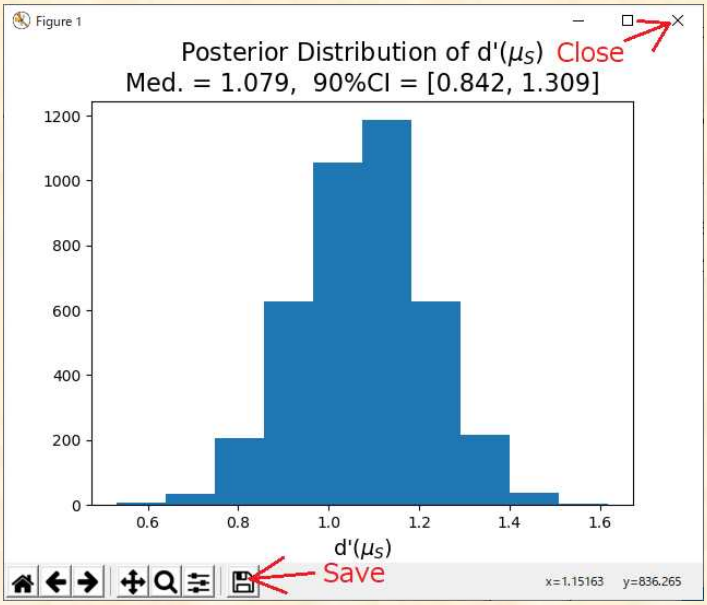

図 A.3

Closing the form in Figure A.3, posterior distribution of τ is shown (Figure A.4).

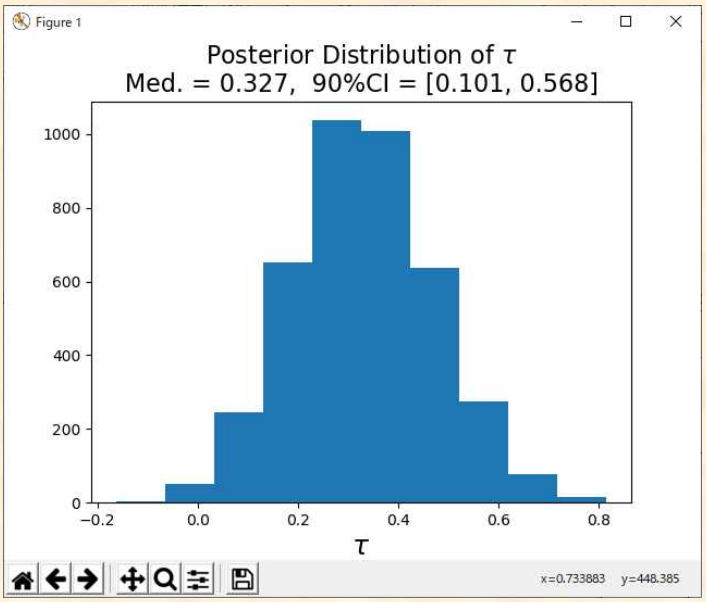

Figure A.4

Closing the form in Figure A.4, graphs of data and model predictions are shown (Figure A.5).

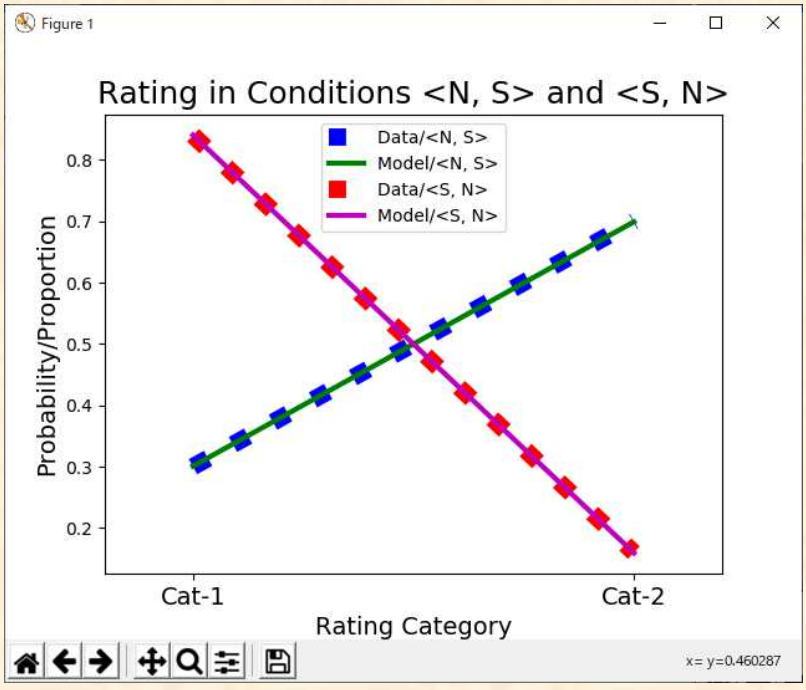

Figure A.5

Dotted lines represent proportions of rating categories calculated from the data, and solid

lines represent estimated probabilities of rating categories, which are medians of posterior distributions. “Cat-1” denotes judgment of category 1, and “Cat-2” denotes judgment of category 2.

Closing the form of Figure A.5, the program ends. After the program ends, output text file can be opened. The content is like this:

```
Input Data File = Data2CAt.csv
<N, S>: [30, 70]
<S, N>: [84, 16]
Inference for Stan model: anon_model_88e26578a5e2b024ad50d7db907e822c.
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
mu_s      1.08  2.5e-3   0.14   0.79   0.98   1.08   1.17   1.35   3130    1.0
tau       0.33  2.5e-3   0.14   0.06   0.24   0.33   0.42   0.62   3288    1.0
p_NS[0]    0.3  7.0e-4   0.04   0.22   0.27    0.3   0.33   0.39   4000    1.0
p_NS[1]    0.7  7.0e-4   0.04   0.61   0.67    0.7   0.73   0.78   4000    1.0
p_SN[0]   0.16  6.9e-4   0.04    0.1   0.14   0.16   0.19   0.24   2919    1.0
p_SN[1]   0.84  6.9e-4   0.04   0.76   0.81   0.84   0.86    0.9   2919    1.0
lp__    -106.0    0.02   0.98 -108.5 -106.4 -105.7 -105.3 -105.0   1677    1.0

Samples were drawn using NUTS at Thu Oct 24 19:29:23 2019.
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).

d':
Med. = 1.079,  90%CI = [0.842, 1.309
tau:
Med. = 0.327,  90%CI ~ [0.101, 0.568]
```

In case of K being odd:

Stan script for K being odd is shown in Listing B1.

Listing B.1　Stan script for K being odd (SDT2AFCOddCat.stan).

```
functions {
    real half_normal_lpdf(real y, real sgm){
        if (y > 0.0){
            return normal_lpdf(y | 0.0, sgm);
        } else {
            return log(0.0);
        }
    }
```

```
}
data {
    int K;
    int n_NS[K];
    int n_SN[K];
}

parameters {
    real mu_s;
    real tau;
    real<lower = 0.0> theta[(K - 1) / 2];
}
transformed parameters {
    real C[K - 1];
    simplex[K] p_NS;
    simplex[K] p_SN;
    C[(K - 1) / 2 + 1] = theta[1];
    C[(K - 1) / 2] = -C[(K - 1) / 2 + 1];
    for (k in 2: ((K - 1) / 2)){
        C[(K - 1) /2 + k] = C[(K - 1) / 2 + k - 1] + theta[k];
        C[(K - 1) / 2 - k + 1] = -C[(K - 1) / 2 + k];
    }
    p_NS[1] = Phi((C[1] - (mu_s - tau))/sqrt(2.0));
    p_NS[K] = 1 - Phi((C[K - 1] - (mu_s - tau))/sqrt(2.0));

    for (k in 2:(K-1)) {
        p_NS[k] = Phi((C[k] - (mu_s - tau))/sqrt(2.0)) -
                  Phi((C[k - 1] - (mu_s - tau))/sqrt(2.0));
    }

    p_SN[1] = Phi((C[1] - (mu_s + tau))/sqrt(2.0));
    p_SN[K] = 1 - Phi((C[K - 1] - (mu_s + tau))/sqrt(2.0));

    for (k in 2:(K-1)) {
        p_SN[k] = Phi((C[k] - (mu_s + tau))/sqrt(2.0)) -
                  Phi((C[k - 1] - (mu_s + tau))/sqrt(2.0));
    }
}
model {
    for (k in 1:((K - 1) / 2)) {
        theta[k] ~ half_normal(1000.0);
    }
    mu_s ~ normal(0.0, 100.0);
    tau ~ normal(0.0, 100.0);
    n_NS ~ multinomial(p_NS);
    n_SN ~ multinomial(p_SN);
}
```

Python script which uses the above Stan script SDT2AFCOddCat.stan is shown in Listing B.2.

Listing B.2 Python script which uses Stan script SDT2AFCOddCat.stan

(SDT2AFCOddCat.py)

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

flnm = input('Input data file (*.csv) = ')
with open(flnm, 'r') as f:
    data = [v for v in csv.reader(f)]
fout_nm = input('Output text file (*.txt) = ')
fout = open(fout_nm, 'w')
fout.write('¥nInput Data File = {}¥n'.format(flnm))

K = len(data[0]) - 1
print('K = ', K)
if (K % 2) == 0:
    print('The number of response categories is not odd.')
    import sys
    sys.exit()

Freq_NS = []
for k in range(K):
    Freq_NS.append(int(data[1][1 + k]))
Freq_SN = []
for k in range(K):
    Freq_SN.append(int(data[2][1 + k]))
print(Freq_NS)
print(Freq_SN)
fout.write('¥n<N, S>¥n{}¥n'.format(Freq_NS))
fout.write('¥n<S, N>¥n{}¥n'.format(Freq_SN))
TN_NS = sum(Freq_NS)
Rating_NS = Freq_NS
TN_SN = sum(Freq_SN)
Rating_SN = np.flip(np.array(Freq_SN))

Data = {'K': K, 'n_NS': Rating_NS, 'n_SN': Rating_SN}

sm = pystan.StanModel(file = 'SDT2AFCOddCat.stan')

fit = sm.sampling(data = Data, n_jobs = 1)

print(fit)
fout.write('¥n{}¥n'.format(fit))

mu = fit['mu_s']
tau = fit['tau']
C = fit['C']
p_NS = fit['p_NS']
p_SN = fit['p_SN']

mu_L05 = np.percentile(mu, 5)
mu_med = np.percentile(mu, 50)
mu_U95 = np.percentile(mu, 95)
fout.write("¥nd':¥nMedian = {0:<.3f},    90%CI = [{1:<.3f}, {2:<.3f}]¥n".
```

```
            format(mu_med, mu_LO5, mu_U95))


tau_LO5 = np.percentile(tau, 5)
tau_med = np.percentile(tau, 50)
tau_U95 = np.percentile(tau, 95)
fout.write('¥ntau:¥nMedian = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]¥n'.
           format(tau_med, tau_LO5, tau_U95))

C_LO5 = np.zeros(K-1)
C_med = np.zeros(K-1)
C_U95 = np.zeros(K-1)
for k in range(K-1):
    C_LO5[k] = np.percentile(C.T[k], 5)
    C_med[k] = np.percentile(C.T[k], 50)
    C_U95[k] = np.percentile(C.T[k], 95)
for k in range(K-1):
    fout.write(('¥nC[{0}]:¥n' + ¥
                'Median = {1:<.3f},   90%CI = [{2:<.3f}, {3:<.3f}]¥n').
                format(k+1, C_med[k], C_LO5[k], C_U95[k]))

plt.hist(mu)
plt.xlabel("d'($¥mu_S$)", fontsize = 14)
plt.title("Posterior Distribution of d'($¥mu_S$)" + ¥
          '¥nMed. = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]'.
          format(mu_med, mu_LO5, mu_U95), fontsize = 16)
plt.show()

plt.hist(tau)
plt.xlabel(r'$¥tau$', fontsize = 18)
plt.title(r'Posterior Distribution of $¥tau$' + ¥
          '¥nMed. = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]'.
          format(tau_med, tau_LO5, tau_U95), fontsize = 16)
plt.show()

for k in range(K-1):
    plt.hist(C.T[k], label = 'C{}'.format(k+1))
plt.title('Posterior Distributions of Category Boundaries', fontsize = 18)
plt.legend()
plt.show()

p1 = np.median(p_NS, axis = 0)
p2 = np.median(p_SN, axis = 0)
p2 = np.flip(p2)

xcat = np.arange(1, K+0.1, 1)
xlabels = ['{}'.format(int(v)) for v in xcat]
y1 = np.array(Freq_NS) / TN_NS
y2 = np.array(Freq_SN) / TN_SN

plt.plot(xcat, y1, 'b--', linewidth = 3, alpha = 0.7, label = 'Data/<N, S>')
plt.plot(xcat, p1, 'g-', linewidth = 3, alpha = 0.7, label = 'Model/<N, S>')
plt.plot(xcat, y2, 'r--', linewidth = 3, alpha = 0.7, label = 'Data/<S, N>')
plt.plot(xcat, p2, 'm-', linewidth = 3, alpha = 0.7, label = 'Model/<S, N>')
```

```
plt.xticks(xcat, xlabels, fontsize = 14)
plt.xlabel('Rating Category', fontsize = 14)
plt.ylabel('Probability/Proportion', fontsize = 14)
plt.title('Ratings in Conditions <N, S> and <S, N>', fontsize = 18)
plt.legend(loc = 'upper center', fontsize = 10)
plt.show()

fout.close()
print('¥n{} was saved.¥n'.format(fout_nm))
```

Files in Listing B.1 and Listing B.2 and sample data files are archived into a zip file RatingOddCat.zip, which can be downloaded.

Run SDT2AFCOddCat.py in Listing B.2, names of the input data file and an output text file are required to be set (Figure B.1).

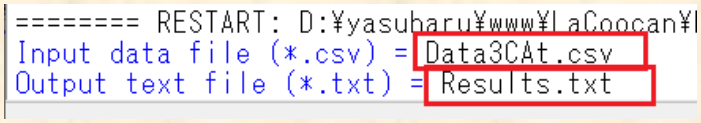

Figure B.1

Output text file name is arbitrary. The input datafile should be CSV format (Figure B.2).

	A	B	C	D	E
1	Cond	Cat_1	Cat_2	Cat_3	
2	N_S	20	26	54	
3	S_N	77	12	11	
4					

Figure B.2　Input data file Data3Cat.csv

In the first row, names for columns are set. The first column shows conditions, <N, S> and <S, N>, the second column shows frequencies of rating judgment of category 1 (e.g., "The signal is on the left"), the third column shows frequencies of rating judgment of category 2 (e.g., "Don't know"), and the fourth column shows frequencies of rating judgment of category 3 (e.g., "The signal is on the right").

In the second row, data from condition <N, S> are set, and in the third row, data from condition <S, N>.

When the file names are set, the program starts calculation. After sampling by Stan ends, posterior distribution of parameter d' is presented (Figure B.3).

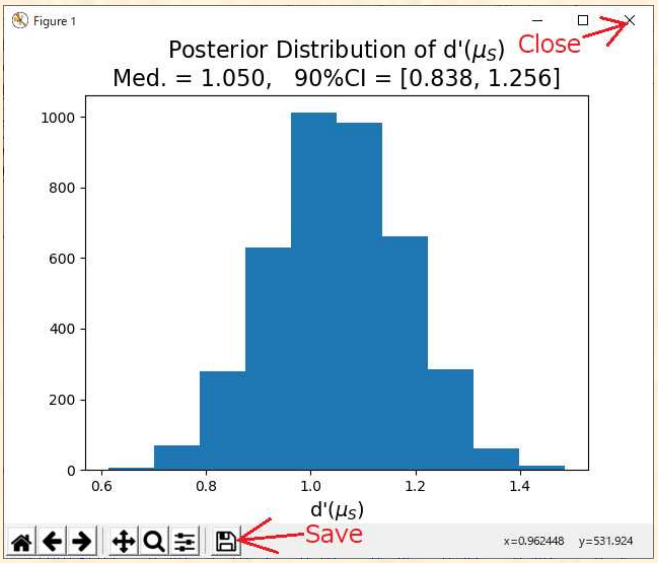

Figure B.3

Closing the form in Figure B.3, posterior distribution of τ is shown (Figure B.4).

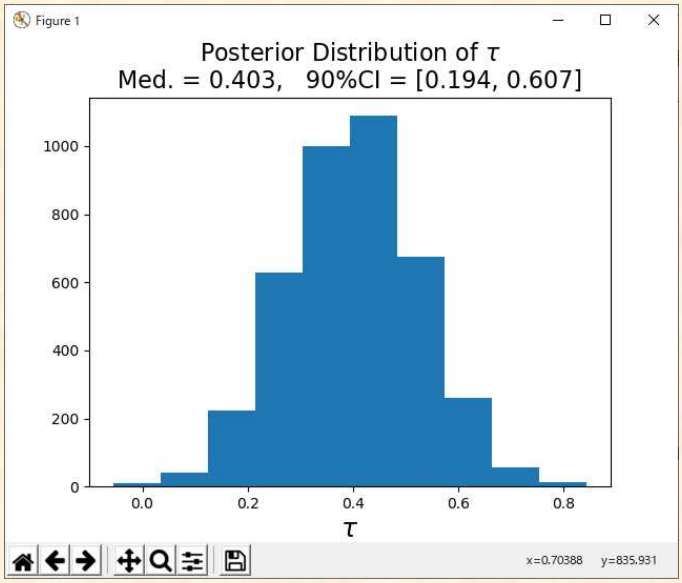

Figure B.4

Closing the form in Figure B.4, posterior distribution of category boundaries are shown (Figure B.5).

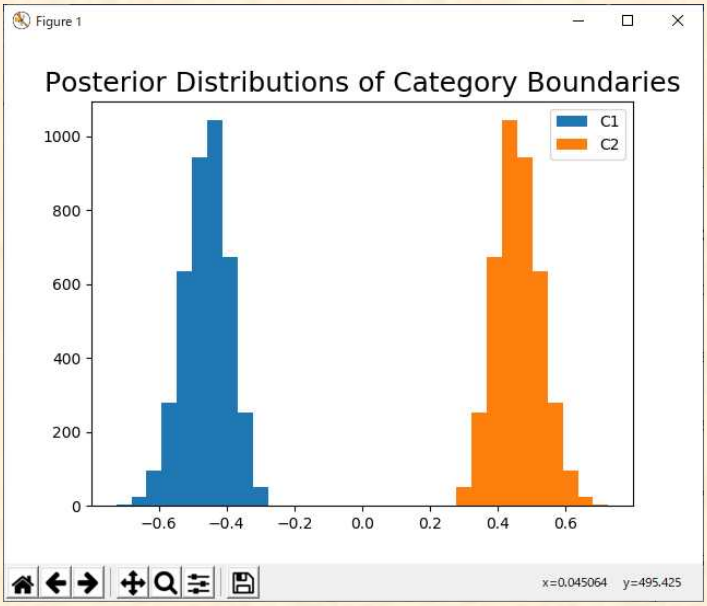

Figure B.5

Closing the form in Figure B,5, graphs of data and model predictions are shown (Figure B.6).

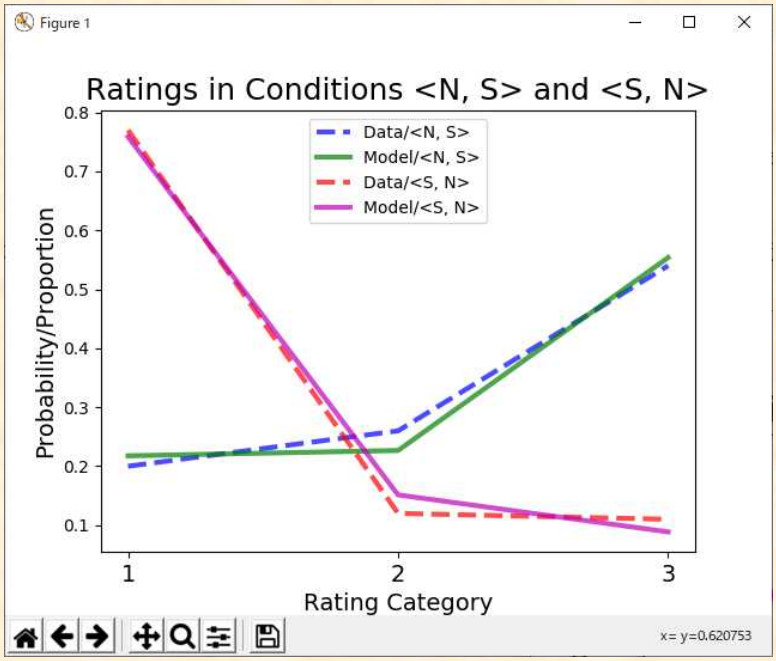

Figure B.6

Dotted lines represent proportions of rating categories calculated from the data, and solid lines represent estimated probabilities of rating categories, which are medians of posterior distributions. Rating categories 1 to 3 correspond to the columns 2 to 4 in Figure B.2, respectively.

Closing the form of Figure B.6, the program ends. After the program ends, output text file can be opened. The content is like this:

```
Input Data File = Data3Cat.csv

<N, S>
[20, 26, 54]

<S, N>
[77, 12, 11]

Inference for Stan model: anon_model_4f4c02dce251e874ae19399d5b4bfa08.
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
mu_s       1.05  2.2e-3   0.13    0.8   0.96   1.05   1.14    1.3   3528    1.0
tau         0.4  2.0e-3   0.13   0.16   0.31    0.4   0.49   0.64   4000    1.0
theta[0]   0.46  1.1e-3   0.07   0.34   0.41   0.46    0.5    0.6   3652    1.0
C[0]      -0.46  1.1e-3   0.07   -0.6   -0.5  -0.46  -0.41  -0.34   3652    1.0
C[1]       0.46  1.1e-3   0.07   0.34   0.41   0.46    0.5    0.6   3652    1.0
p_NS[0]    0.22  6.2e-4   0.04   0.15   0.19   0.22   0.24    0.3   4000    1.0
p_NS[1]    0.23  5.2e-4   0.03   0.17   0.21   0.23   0.25    0.3   4000    1.0
p_NS[2]    0.55  7.5e-4   0.05   0.46   0.52   0.55   0.58   0.65   4000    1.0
p_SN[0]    0.09  4.5e-4   0.02   0.05   0.07   0.09   0.11   0.15   3004    1.0
p_SN[1]    0.15  4.2e-4   0.03    0.1   0.13   0.15   0.17   0.21   4000    1.0
p_SN[2]    0.76  6.6e-4   0.04   0.67   0.73   0.76   0.79   0.83   3889    1.0
lp__     -181.3    0.03   1.21 -184.5 -181.9 -181.0 -180.4 -179.9   1907    1.0

Samples were drawn using NUTS at Fri Oct 25 09:04:47 2019.
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).

d':
Median = 1.050,    90%CI = [0.838, 1.256]

tau:
Median = 0.403,    90%CI = [0.194, 0.607]

C[1]:
Median = -0.455,    90%CI = [-0.574, -0.352]

C[2]:
Median = 0.455,    90%CI = [0.352, 0.574]
```

Next, consider another example, which uses K=5 number of rating categories (Figure B.7).

	A	B	C	D	E	F	G
1	Cond	Cat_1	Cat_2	Cat_3	Cat_4	Cat_5	
2	N_S	21	10	5	18	46	
3	S_N	66	10	7	12	5	
4							
5							

Figure B.7　Data5Cat.csv

Run SDT2AFCOddCat.py in Listing B.2, and set names of the input data file and an output text file as in Figure B.8.

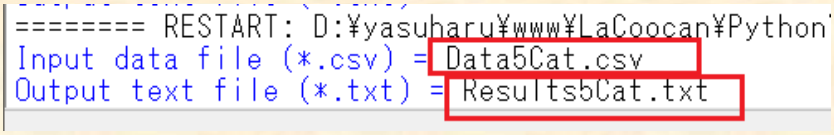

Figure B.8

When the file names are set, the program starts calculation. After sampling by Stan ends, posterior distribution of parameter d' is presented (Figure B.9).

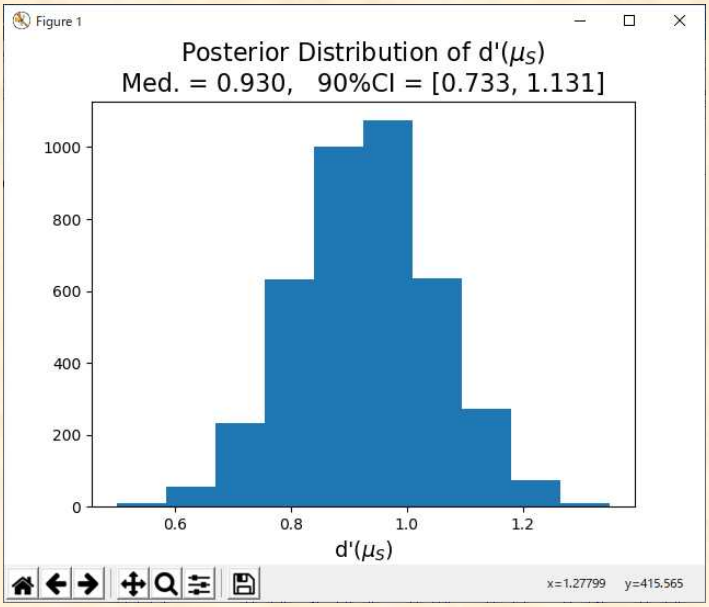

Figure B.9

Closing the form in Figure B.9, posterior distribution of τ is shown (Figure B.10).

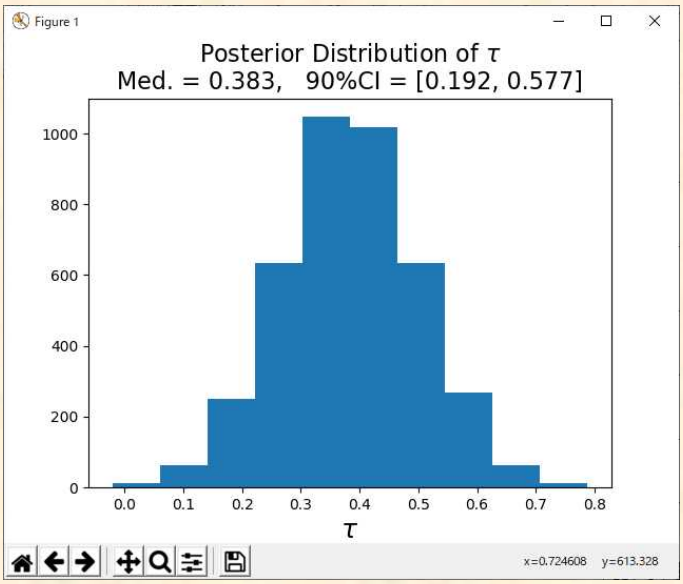

Figure B.10

Closing the form in Figure B.10, posterior distribution of category boundaries are shown (Figure B.11).

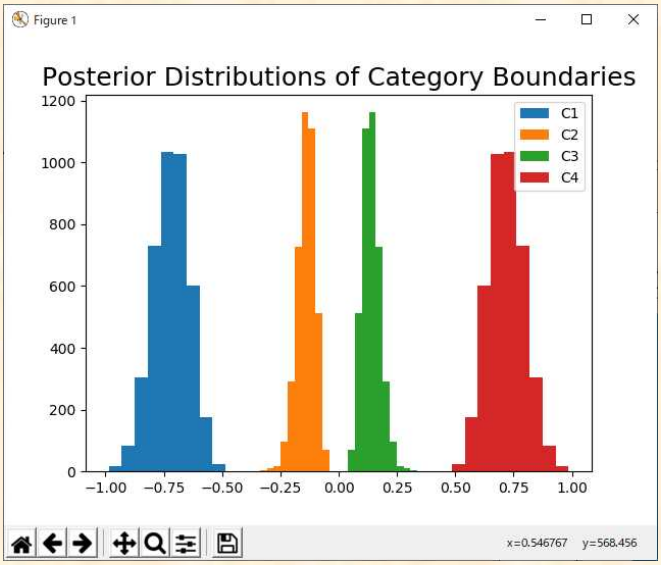

Figure B.11

Closing the form in Figure B,11, graphs of data and model predictions are shown (Figure B.12).

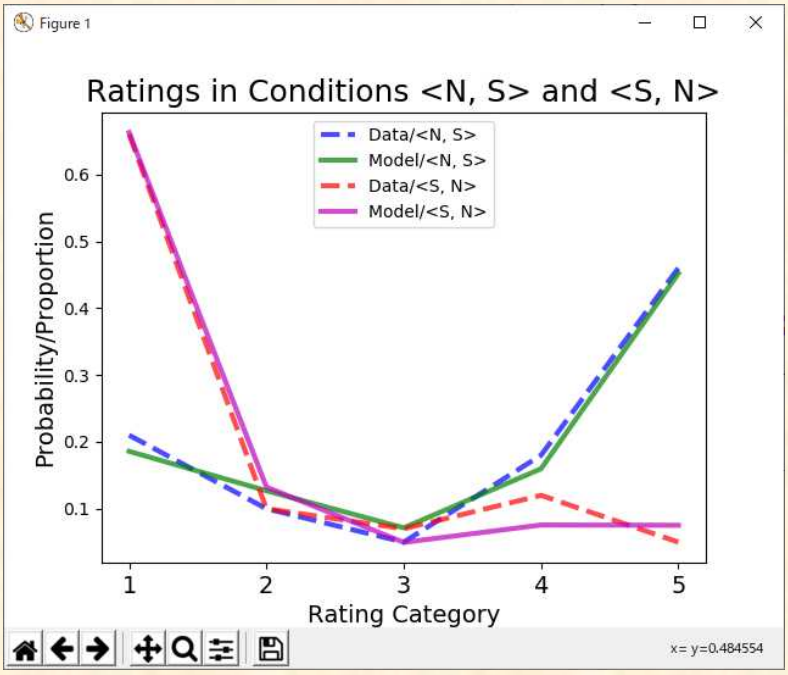

Figure B.12

Dotted lines represent proportions of rating categories calculated from the data, and solid lines represent estimated probabilities of rating categories, which are medians of posterior distributions. Rating categories 1 to 5 correspond to the columns 2 to 6 in Figure B.7, respectively.

Closing the form of Figure B.12, the program ends. After the program ends, output text file can be opened. The content is like this:

```
Input Data File = Data5Cat.csv

<N, S>
[21, 10, 5, 18, 46]

<S, N>
[66, 10, 7, 12, 5]

Inference for Stan model: anon_model_4f4c02dce251e874ae19399d5b4bfa08.
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
mu_s      0.93  1.9e-3   0.12   0.69   0.85   0.93   1.01   1.17   4000    1.0
tau       0.38  1.8e-3   0.12   0.16   0.31   0.38   0.46   0.61   4000    1.0
theta[0]  0.14  6.1e-4   0.04   0.07   0.11   0.14   0.16   0.22   4000    1.0
theta[1]  0.58  1.1e-3   0.07   0.44   0.53   0.58   0.63   0.73   4000    1.0
C[0]     -0.72  1.2e-3   0.08  -0.87  -0.77  -0.72  -0.66  -0.57   4000    1.0
C[1]     -0.14  6.1e-4   0.04  -0.22  -0.16  -0.14  -0.11  -0.07   4000    1.0
C[2]      0.14  6.1e-4   0.04   0.07   0.11   0.14   0.16   0.22   4000    1.0
```

```
C[3]       0.72  1.2e-3   0.08   0.57   0.66   0.72   0.77   0.87    4000     1.0
p_NS[0]    0.19  5.6e-4   0.04   0.13   0.16   0.19   0.21   0.27    4000     1.0
p_NS[1]    0.13  2.6e-4   0.02    0.1   0.12   0.13   0.14   0.16    4000     1.0
p_NS[2]    0.07  3.2e-4   0.02   0.04   0.06   0.07   0.09   0.12    4000     1.0
p_NS[3]    0.16  3.2e-4   0.02   0.12   0.15   0.16   0.17    0.2    4000     1.0
p_NS[4]    0.45  7.5e-4   0.05   0.36   0.42   0.45   0.48   0.55    4000     1.0
p_SN[0]    0.08  3.6e-4   0.02   0.04   0.06   0.08   0.09   0.12    3523     1.0
p_SN[1]    0.08  2.0e-4   0.01   0.05   0.07   0.08   0.08    0.1    4000     1.0
p_SN[2]    0.05  2.4e-4   0.01   0.03   0.04   0.05   0.06   0.08    4000     1.0
p_SN[3]    0.13  3.0e-4   0.02    0.1   0.12   0.13   0.15   0.17    4000     1.0
p_SN[4]    0.66  7.1e-4   0.05   0.57   0.63   0.66   0.69   0.75    4000     1.0
lp__     -270.3    0.03   1.47 -274.0 -271.1 -270.0 -269.3 -268.5    2012     1.0

Samples were drawn using NUTS at Fri Oct 25 09:44:06 2019.
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).

d':
Median = 0.930,    90%CI = [0.733, 1.131]

tau:
Median = 0.383,    90%CI = [0.192, 0.577]

C[1]:
Median = -0.716,    90%CI = [-0.852, -0.596]

C[2]:
Median = -0.137,    90%CI = [-0.208, -0.084]

C[3]:
Median = 0.137,    90%CI = [0.084, 0.208]

C[4]:
Median = 0.716,    90%CI = [0.596, 0.852]
```

In case of K being even greater than 2:

Stan script for K being even greater that 2 is shown in Listing C.1.

Listing C.1　Stan script for K being even greater than 2 (SDT2AFCEvenCat.stan)

```
functions {
    real half_normal_lpdf(real y, real sgm){
        if (y > 0.0){
            return normal_lpdf(y | 0.0, sgm);
        } else {
            return log(0.0);
        }
```

```
    }
}
data {
    int K;
    int n_NS[K];
    int n_SN[K];
}

parameters {
    real mu_s;
    real tau;
    real<lower = 0.0> theta[K/2 - 1];
}
transformed parameters {
    real C[K - 1];
    simplex[K] p_NS;
    simplex[K] p_SN;
    C[K/2] = 0.0;
    for (k in 1: (K/2 -1)){
        C[K/2 + k] = C[K/2 + k - 1] + theta[k];
        C[K/2 - k] = -C[K/2 + k];
    }
    p_NS[1] = Phi((C[1] - (mu_s - tau))/sqrt(2.0));
    p_NS[K] = 1 - Phi((C[K - 1] - (mu_s - tau))/sqrt(2.0));

    for (k in 2:(K-1)) {
        p_NS[k] = Phi((C[k] - (mu_s - tau))/sqrt(2.0)) -
                  Phi((C[k - 1] - (mu_s - tau))/sqrt(2.0));
    }

    p_SN[1] = Phi((C[1] - (mu_s + tau))/sqrt(2.0));
    p_SN[K] = 1 - Phi((C[K - 1] - (mu_s + tau))/sqrt(2.0));

    for (k in 2:(K-1)) {
        p_SN[k] = Phi((C[k] - (mu_s + tau))/sqrt(2.0)) -
                  Phi((C[k - 1] - (mu_s + tau))/sqrt(2.0));
    }
}
model {
    for (k in 1:(K/2 - 1)) {
        theta[k] ~ half_normal(1000.0);
    }
    mu_s ~ normal(0.0, 100.0);
    tau ~ normal(0.0, 100.0);
    n_NS ~ multinomial(p_NS);
    n_SN ~ multinomial(p_SN);
}
```

Python script which uses the Stan script above is shown in Listing C.2.

Listing C.2　Python script which uses the Stan script in Listing C.2 (SDT2AFCEvenCat.py)

```
#
```

```
#       Number of rating categories should be even.
#
#           Yasuharu Okamoto, 2019.10
#
import numpy as np
import pystan
import matplotlib.pyplot as plt
import csv
import pickle

flnm = input('Input data file (*.csv) = ')
with open(flnm, 'r') as f:
    data = [v for v in csv.reader(f)]

fout_nm = input('Output text file (*.txt) = ')
fout = open(fout_nm, 'w')
fout.write('Input Data File = {}¥n'.format(flnm))

K = len(data[0]) - 1
print('K = ', K)
if (K % 2) == 1:
    print('The number of rating categories is not even!')
    import sys
    sys.exit()

Freq_NS = []
for k in range(K):
    Freq_NS.append(int(data[1][1 + k]))
Freq_SN = []
for k in range(K):
    Freq_SN.append(int(data[2][1 + k]))
print(Freq_NS)
print(Freq_SN)
fout.write('¥n<N, S>¥n{}¥n'.format(Freq_NS))
fout.write('¥n<S, N>¥n{}¥n'.format(Freq_SN))
TN_NS = sum(Freq_NS)
Rating_NS = Freq_NS
print(Rating_NS)
TN_SN = sum(Freq_SN)
Rating_SN = np.flip(np.array(Freq_SN))

Data = {'K': K, 'n_NS': Rating_NS, 'n_SN': Rating_SN}

sm = pystan.StanModel(file = 'SDT2AFCEvenCat.stan')

fit = sm.sampling(data = Data, n_jobs = 1)

print(fit)
fout.write('¥n{}¥n'.format(fit))

mu = fit['mu_s']
tau = fit['tau']
C = fit['C']
p_NS = fit['p_NS']
```

```
p_SN = fit['p_SN']

mu_L05 = np.percentile(mu, 5)
mu_med = np.percentile(mu, 50)
mu_U95 = np.percentile(mu, 95)
fout.write("¥nd':¥nMedian = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]¥n".
           format(mu_med, mu_L05, mu_U95))

tau_L05 = np.percentile(tau, 5)
tau_med = np.percentile(tau, 50)
tau_U95 = np.percentile(tau, 95)
fout.write('¥ntau:¥nMedian = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]¥n'.
           format(tau_med, tau_L05, tau_U95))

C_L05 = np.zeros(K-1)
C_med = np.zeros(K-1)
C_U95 = np.zeros(K-1)
for k in range(K-1):
    C_L05[k] = np.percentile(C.T[k], 5)
    C_med[k] = np.percentile(C.T[k], 50)
    C_U95[k] = np.percentile(C.T[k], 95)
for k in range(K-1):
    fout.write(('¥nC[{0}]:¥n' + ¥
                'Median = {1:<.3f},   90%CI = [{2:<.3f}, {3:<.3f}]¥n').
                format(k+1, C_med[k], C_L05[k], C_U95[k]))

plt.hist(mu)
plt.xlabel("d'($¥mu_S$)", fontsize = 14)
plt.title("Posterior Distribution of d'($¥mu_S$)" + ¥
          '¥nMed. = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]'.
          format(mu_med, mu_L05, mu_U95), fontsize = 16)
plt.show()

plt.hist(tau)
plt.xlabel(r'$¥tau$', fontsize = 18)
plt.title(r'Posterior Distribution of $¥tau$' + ¥
          '¥nMed. = {0:<.3f},   90%CI = [{1:<.3f}, {2:<.3f}]'.
          format(tau_med, tau_L05, tau_U95), fontsize = 16)
plt.show()

for k in range(K-1):
    if k + 1 != K/2:
        plt.hist(C.T[k], label = 'C{}'.format(k+1))
plt.title('Posterior Distributions of Category Boundaries' + ¥
          '¥nC{} = 0 fixed'.format(K//2), fontsize = 16)
plt.legend(loc = 'lower center')
plt.show()

xcat = np.arange(1, K+0.1, 1)
xlabels = ['{}'.format(int(v)) for v in xcat]
y1 = np.array(Freq_NS) / TN_NS
y2 = np.array(Freq_SN) / TN_SN

p1 = np.median(p_NS, axis = 0)
```

```
p2 = np.median(p_SN, axis = 0)
p2 = np.flip(p2)

plt.plot(xcat, y1, 'b--', linewidth = 3, alpha = 0.7, label = 'Data/<N, S>')
plt.plot(xcat, p1, 'g-', linewidth = 3, alpha = 0.7, label = 'Model/<N, S>')
plt.plot(xcat, y2, 'r--', linewidth = 3, alpha = 0.7, label = 'Data/<S, N>')
plt.plot(xcat, p2, 'm-', linewidth = 3, alpha = 0.7, label = 'Model/<S, N>')

plt.xticks(xcat, xlabels, fontsize = 14)
plt.xlabel('Rating Category', fontsize = 14)
plt.ylabel('Probability/Proportion', fontsize = 14)
plt.title('Ratings in Conditions <N, S> and <S, N>', fontsize = 18)
plt.legend(loc = 'upper center', fontsize = 10)
plt.show()

fout.close()
print('¥n{} was saved.¥n'.format(fout_nm))
```

Files in Listing C.1 and Listing C.2 and sample data files are archived into a zip file RatingEvenCat.zip, which can be downloaded.

Run SDT2AFCEvenCat.py in Listing C.2, names of the input data file and an output text file are required to be set (Figure C.1).

```
======= RESTART: D:¥yasuharu¥www¥LaCoocan¥F
Input data file (*.csv) = Data4Cat.csv
Output text file (*.txt) = Results.txt
```

Figure C.1

Output text file name is arbitrary. The input datafile should be CSV format (Figure C.2).

	A	B	C	D	E	F
1	Cond	Cat_1	Cat_2	Cat_3	Cat_4	
2	N_S	24	9	12	55	
3	S_N	74	13	7	6	
4						
5						

Figure C.2　Data4Cat.csv

In the first row, names for columns are set. The first column shows conditions, <N, S> and <S, N>, the second column shows frequencies of rating judgment of category 1 (e.g., “The signal is on the left”), the third column shows frequencies of rating judgment of category 2 (e.g., “Probably, the signal is on the left”), the fourth column shows frequencies of rating judgment of category 3 (e.g., “Probably, the signal is on the right”), and the fifth column shows frequencies of rating judgment of category 4 (e.g., “The signal is on the right”).

In the second row, data from condition <N, S> are set, and in the third row, data from condition <S, N>.

When the file names are set, the program starts calculation. After sampling by Stan ends, posterior distribution of parameter d' is presented (Figure C.3).

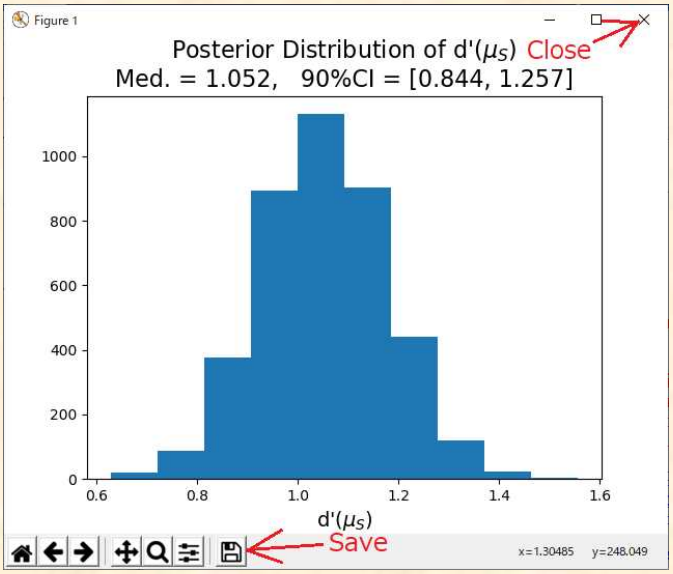

Figure C.3

Closing the form in Figure C.3, posterior distribution of τ is shown (Figure C.4).

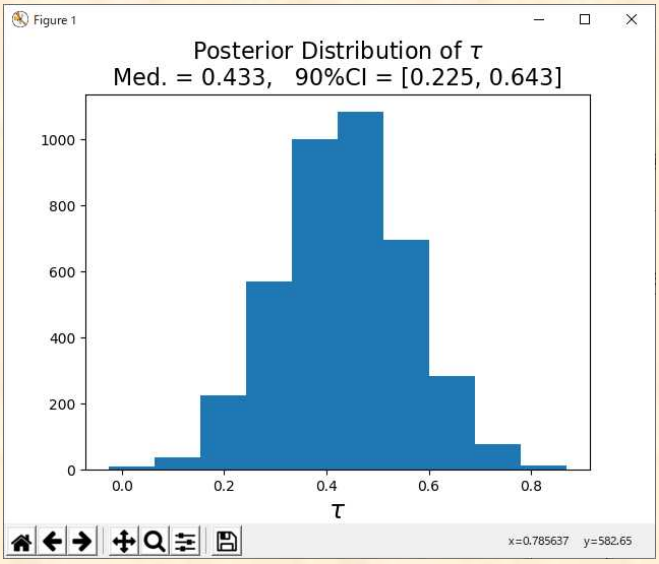

Figure C.4

Closing the form in Figure C.4, posterior distribution of category boundaries are shown (Figure C.5).

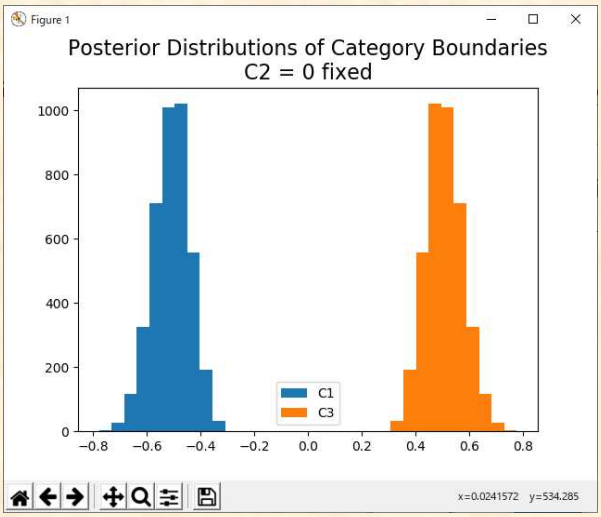

Figure C.5

Closing the form in Figure C,5, graphs of data and model predictions are shown (Figure C.6).

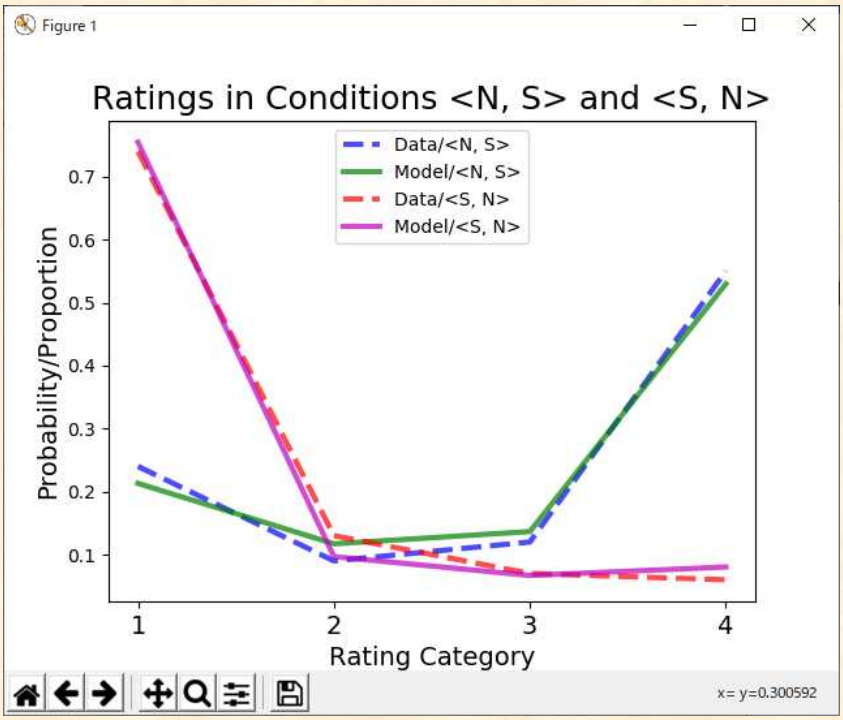

Figure C.6

Dotted lines represent proportions of rating categories calculated from the data, and solid lines represent estimated probabilities of rating categories, which are medians of posterior distributions. Rating categories 1 to 4 correspond to the columns 2 to 5 in Figure C.2, respectively.

Closing the form of Figure C.6, the program ends. After the program ends, output text file

can be opened. The content is like this:

```
Input Data File = Data4Cat.csv

<N, S>
[24, 9, 12, 55]

<S, N>
[74, 13, 7, 6]

Inference for Stan model: anon_model_f2c4de04275e32f42265169574310b77.
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
mu_s       1.05  2.2e-3   0.13   0.81   0.96   1.05   1.14    1.3   3432    1.0
tau        0.43  2.0e-3   0.13   0.18   0.35   0.43   0.52   0.68   4000    1.0
theta[0]   0.51  1.3e-3   0.07   0.38   0.46   0.51   0.55   0.65   2759    1.0
C[0]      -0.51  1.3e-3   0.07  -0.65  -0.55  -0.51  -0.46  -0.38   2759    1.0
C[1]        0.0     0.0    0.0    0.0    0.0    0.0    0.0    0.0   4000    nan
C[2]       0.51  1.3e-3   0.07   0.38   0.46   0.51   0.55   0.65   2759    1.0
p_NS[0]    0.22  6.1e-4   0.04   0.15   0.19   0.21   0.24    0.3   4000    1.0
p_NS[1]    0.12  2.5e-4   0.02   0.09   0.11   0.12   0.13   0.15   4000    1.0
p_NS[2]    0.14  3.6e-4   0.02    0.1   0.12   0.14   0.15   0.18   2950    1.0
p_NS[3]    0.53  7.8e-4   0.05   0.44    0.5   0.53   0.56   0.63   4000    1.0
p_SN[0]    0.08  4.4e-4   0.02   0.04   0.07   0.08    0.1   0.13   2688    1.0
p_SN[1]    0.07  1.9e-4   0.01   0.05   0.06   0.07   0.07   0.09   4000    1.0
p_SN[2]     0.1  2.7e-4   0.02   0.07   0.09    0.1   0.11   0.13   4000    1.0
p_SN[3]    0.75  6.8e-4   0.04   0.67   0.73   0.75   0.78   0.83   3710    1.0
lp__     -210.0    0.03   1.26 -213.3 -210.5 -209.6 -209.0 -208.6   2124    1.0

Samples were drawn using NUTS at Fri Oct 25 10:46:34 2019.
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).

d':
Median = 1.052,   90%CI = [0.844, 1.257]

tau:
Median = 0.433,   90%CI = [0.225, 0.643]

C[1]:
Median = -0.506,   90%CI = [-0.624, -0.400]

C[2]:
Median = 0.000,   90%CI = [0.000, 0.000]

C[3]:
Median = 0.506,   90%CI = [0.400, 0.624]
```

Next, consider another example, which uses K=6 number of rating categories (Figure C.7).

	A	B	C	D	E	F	G	H
1	Cond	Cat_1	Cat_2	Cat_3	Cat_4	Cat_5	Cat_6	
2	N_S	6	23	5	4	37	25	
3	S_N	42	38	2	5	11	2	
4								
5								

Figure C.7　Data6Cat.csv

Run SDT2AFCEvenCat.py in Listing C.2, and set names of the input data file and an output text file as in Figure C.8.

```
======= RESTART: D:¥yasuharu¥www¥LaCoocan¥Python¥e
Input data file (*.csv) = Data6Cat.csv
Output text file (*.txt) = Results6Cat.txt
```

Figure C.8

When the file names are set, the program starts calculation. After sampling by Stan ends, posterior distribution of parameter d' is presented (Figure C.9).

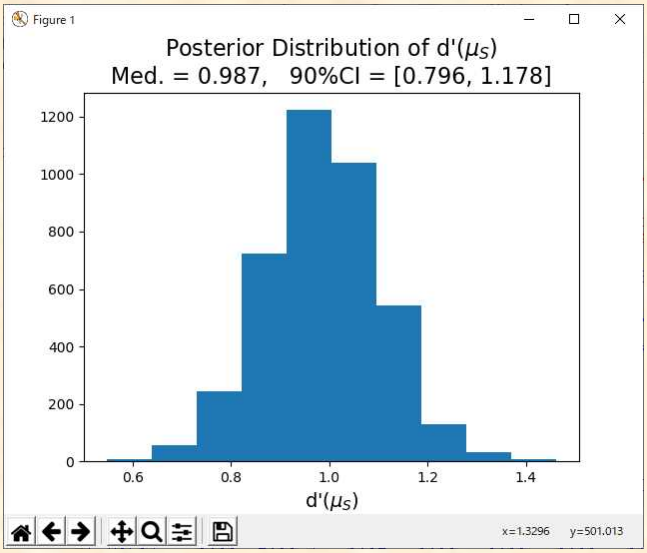

Figure C.9

Closing the form in Figure C.9, posterior distribution of τ is shown (Figure C.10).

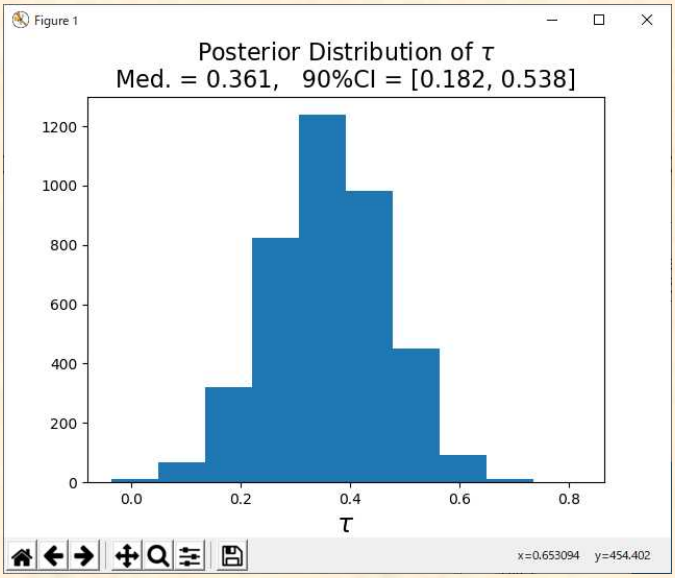

Figure C.10

Closing the form in Figure C.10, posterior distribution of category boundaries are shown (Figure C.11).

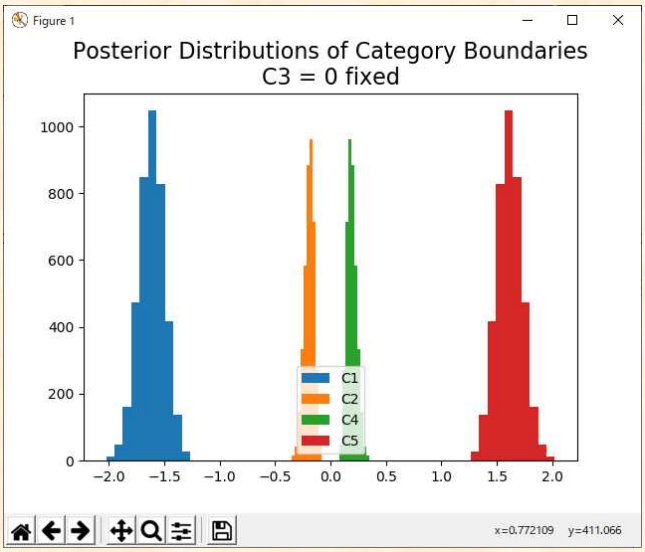

Figure C.11

Closing the form in Figure C,11, graphs of data and model predictions are shown (Figure C.12).

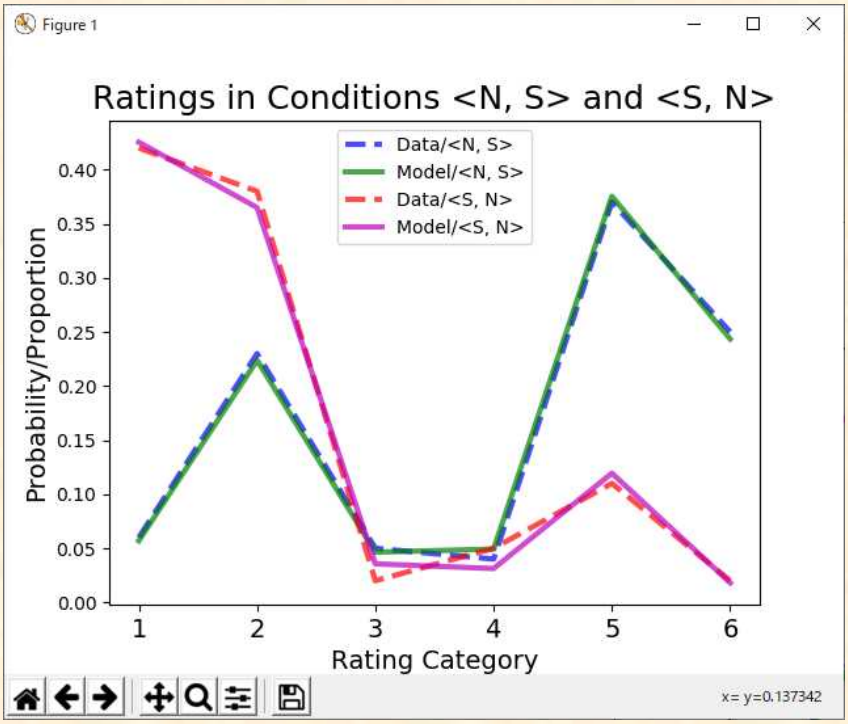

Figure C.12

Dotted lines represent proportions of rating categories calculated from the data, and solid lines represent estimated probabilities of rating categories, which are medians of posterior distributions. Rating categories 1 to 6 correspond to the columns 2 to 7 in Figure C.7, respectively.

Closing the form of Figure C.12, the program ends. After the program ends, output text file can be opened. The content is like this:

```
Input Data File = Data6Cat.csv

<N, S>
[6, 23, 5, 4, 37, 25]

<S, N>
[42, 38, 2, 5, 11, 2]

Inference for Stan model: anon_model_f2c4de04275e32f42265169574310b77.
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
mu_s      0.99  1.9e-3   0.12    0.76    0.91    0.99    1.06    1.22   3781    1.0
tau       0.36  1.7e-3   0.11    0.15    0.29    0.36    0.43    0.57   4000    1.0
theta[0]  0.19  7.1e-4   0.04    0.11    0.16    0.19    0.22    0.29   4000    1.0
theta[1]  1.42  1.8e-3   0.11    1.21    1.34    1.42    1.49    1.65   4000    1.0
C[0]     -1.61  1.8e-3   0.11   -1.84   -1.69   -1.61   -1.54   -1.39   4000    1.0
```

C[1]	-0.19	7.1e-4	0.04	-0.29	-0.22	-0.19	-0.16	-0.11	4000	1.0
C[2]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4000	nan
C[3]	0.19	7.1e-4	0.04	0.11	0.16	0.19	0.22	0.29	4000	1.0
C[4]	1.61	1.8e-3	0.11	1.39	1.54	1.61	1.69	1.84	4000	1.0
p_NS[0]	0.06	2.9e-4	0.02	0.03	0.05	0.06	0.07	0.1	3455	1.0
p_NS[1]	0.22	4.1e-4	0.03	0.17	0.21	0.22	0.24	0.28	4000	1.0
p_NS[2]	0.05	1.7e-4	0.01	0.03	0.04	0.05	0.05	0.07	4000	1.0
p_NS[3]	0.05	1.9e-4	0.01	0.03	0.04	0.05	0.06	0.08	4000	1.0
p_NS[4]	0.38	4.3e-4	0.03	0.32	0.36	0.38	0.39	0.43	3947	1.0
p_NS[5]	0.24	6.0e-4	0.04	0.17	0.22	0.24	0.27	0.32	4000	1.0
p_SN[0]	0.02	1.4e-4	8.0e-3	7.9e-3	0.01	0.02	0.02	0.04	3464	1.0
p_SN[1]	0.12	3.4e-4	0.02	0.08	0.11	0.12	0.13	0.17	3899	1.0
p_SN[2]	0.03	1.2e-4	7.6e-3	0.02	0.03	0.03	0.04	0.05	4000	1.0
p_SN[3]	0.04	1.5e-4	9.5e-3	0.02	0.03	0.04	0.04	0.06	4000	1.0
p_SN[4]	0.37	4.5e-4	0.03	0.31	0.35	0.36	0.38	0.42	4000	1.0
p_SN[5]	0.43	7.1e-4	0.04	0.34	0.4	0.43	0.46	0.51	4000	1.0
lp__	-298.1	0.03	1.42	-301.7	-298.8	-297.8	-297.1	-296.3	1956	1.0

Samples were drawn using NUTS at Fri Oct 25 11:12:13 2019.
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).

d':
Median = 0.987, 90%CI = [0.796, 1.178]

tau:
Median = 0.361, 90%CI = [0.182, 0.538]

C[1]:
Median = -1.611, 90%CI = [-1.802, -1.427]

C[2]:
Median = -0.189, 90%CI = [-0.272, -0.124]

C[3]:
Median = 0.000, 90%CI = [0.000, 0.000]

C[4]:
Median = 0.189, 90%CI = [0.124, 0.272]

C[5]:
Median = 1.611, 90%CI = [1.427, 1.802]

References

Klein, S. A. (2001). Measuring, estimating, and understanding the psychometric function: A commentary. *Perception & Psychophysics*, 63, 1421-1455.

Up