(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３回：感覚尺度構成

確率分布

補遺（Q&A）

（日本女子大学）
岡本 安晴

感覚の識別・弁別

弁別閾、主観的等価点

感覚の強さ

対数法則

Stevens

べき法則：prothetic continuum/dimension

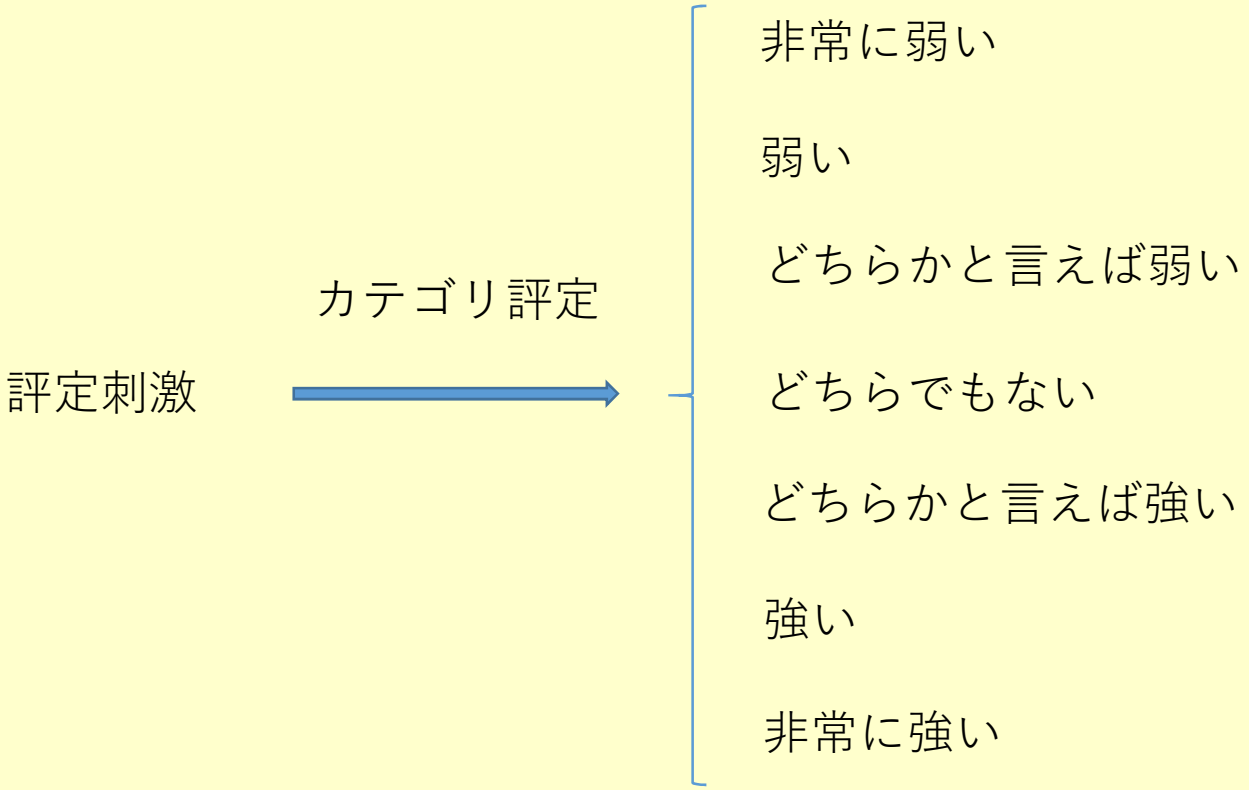
metathetic continuum/dimension

$$\psi = a\varphi^b$$

$$\psi = a(\varphi - c)^b$$

岡本安晴「心理学データ分析と測定」、第10章

順序カテゴリ尺度法


評定刺激
カテゴリ評定
非常に弱い
弱い
どちらかと言えば弱い
どちらでもない
どちらかと言えば強い
強い
非常に強い

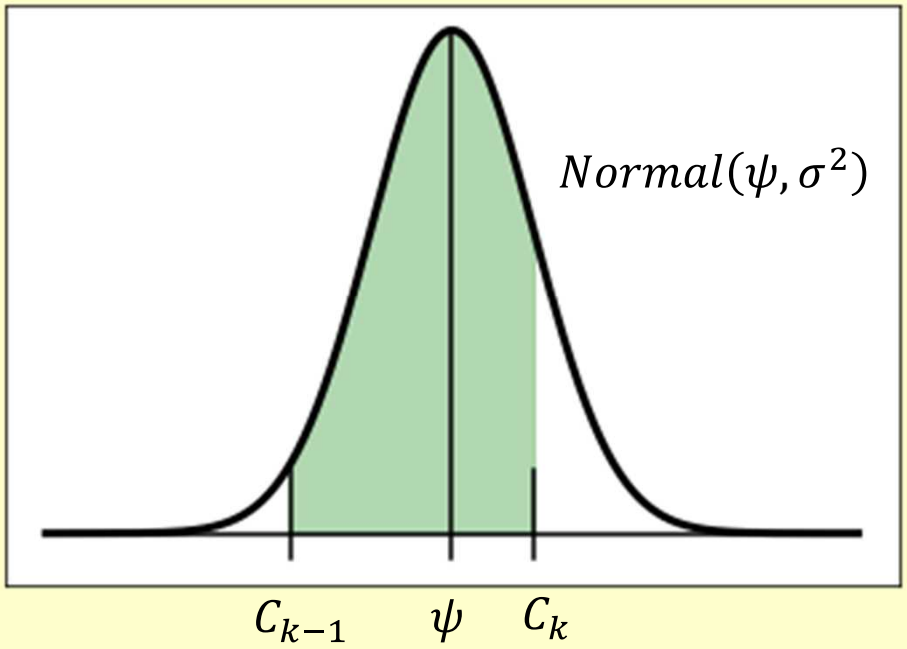
観察者は、刺激の感覚の強さを順序カテゴリで答え、
それに基づいて尺度を構成する

カテゴリ尺度法（Gescheider, 1997）、評定尺度法（難波・桑野、1998）
継次カテゴリ法／系列カテゴリ法（method of successive categories）、
継次間隔法（method of successive intervals）（中谷、1969）
などと呼ばれている。

カテゴリ尺度法では、刺激の感覚量とカテゴリ境界の関係にThurstoneの判断の基本モデルを適用したカテゴリ判断の法則（The law of categorical judgment; Torgerson, 1958）を用いて分析することができる。

Gescheider, G. A. (1997). *Psychophysics: The fundamentals, Third edition*. Mahwah: Lawrence Erlbaum Associations, Publishers.
中谷和夫（1969）．第 5 章　尺度構成法．　八木冕（監修）／田中良久（編）講座心理学 2　計量心理学．東京大学出版会.
難波精一郎・桑野園子（1998）．音の評価のための心理学的測定法．コロナ社.
Thurstone, L. L. (1927). A law of comparative judgment. *Psychological Review*, 34, 273-286.
Torgerson, W. S. (1958). *Theory and methods of scaling*. New York: John Wiley & Sons, Inc.

モデル



S（刺激） → ψ（感覚）

→ $C_{k-1} < \psi < C_k$

→ 評価カテゴリ k

$$確率 P(k|S) = P(C_{k-1} < \psi < C_k) = \Phi\left(\frac{C_k - \psi}{\sigma}\right) - \Phi\left(\frac{C_{k-1} - \psi}{\sigma}\right)$$

確率$P(k|S) = P(C_{k-1} < \psi < C_k) = \Phi\left(\frac{C_k - \psi}{\sigma}\right) - \Phi\left(\frac{C_{k-1} - \psi}{\sigma}\right)$

$C_0 = -\infty$ $C_K = +\infty$

Dynamic range＝一定 $C_1 = 0$ $C_{K-1} = 1$

Teghtsoonian, R. (1971). “On the exponents in Stevens’ law and the constant in Ekman’s law.” *Psychological Review*, 78, 71-80.

$\sigma_i = a + b(\psi_i - \psi_1)$ Cf. エックマンの法則 $\frac{\Delta\psi}{\psi} = const.$

フェヒナー $\Delta\psi = const.$ $\sigma_i = const.$

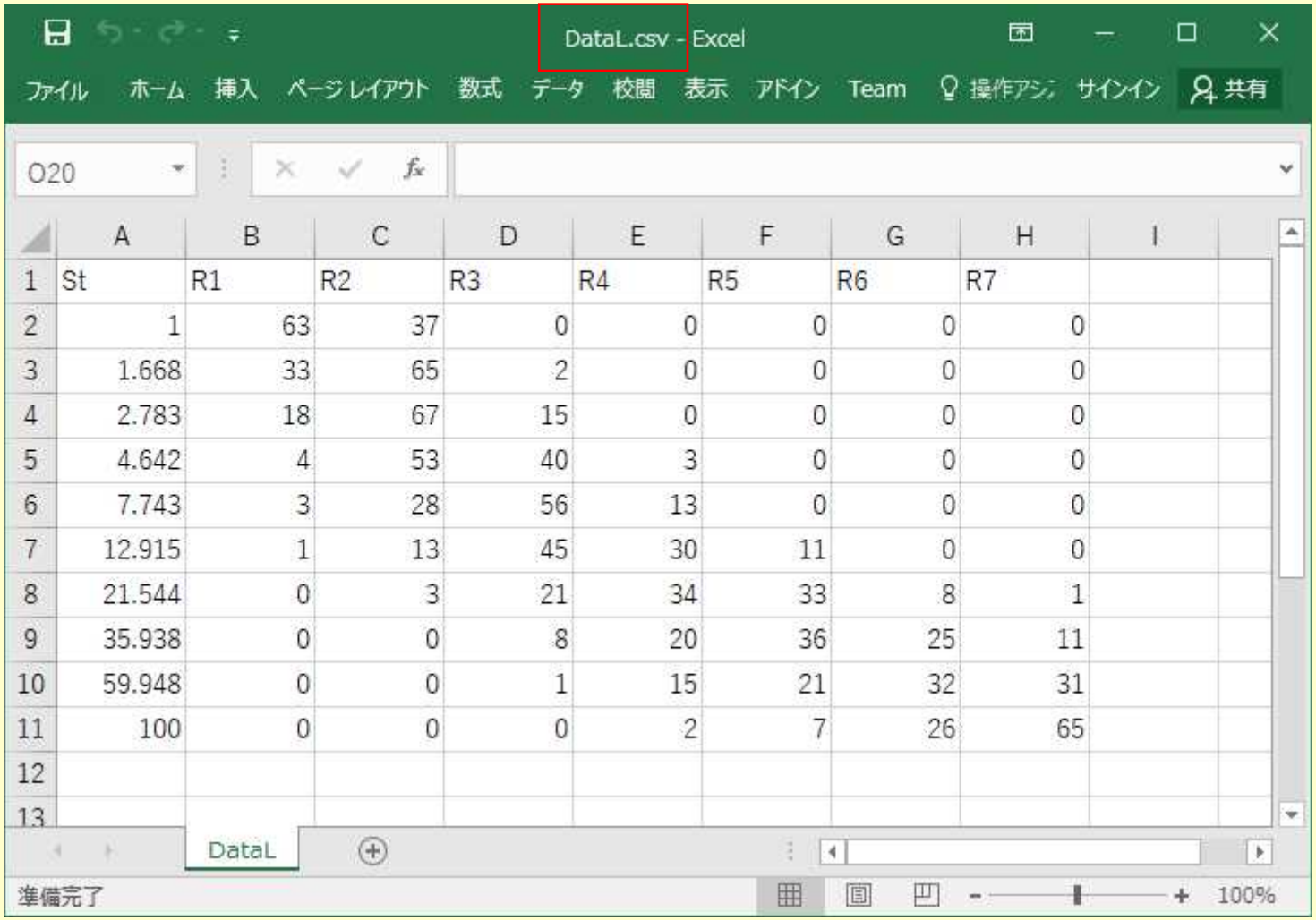
サーストンの一対比較法ケースV（べき法則に従わない）

$\sigma_i: free$

```
data {   int NSt;   int K;   real Sts[NSt];   int Rs[NSt, K];   }
transformed data {   vector[NSt+1] a_Psis;   vector[K-2] a_C;
  for (i in 1:(NSt+1)) {     a_Psis[i] = 1.0;   }
  for (i in 1:(K-2)) {     a_C[i] = 1.0;   }   }
parameters {   simplex[NSt+1] prePsis;   simplex[K-2] preC;
  real<lower = 0.0> a;   real<lower = 0.0> b;   }
transformed parameters {   real Psis[NSt];   real sgms[NSt];
  real C[K-1];   simplex[K] theta[NSt];   real v;   v = 0.0;
  for (s in 1:NSt) {       v = v + prePsis[s];       Psis[s] = logit(v);   }
  for (s in 1:NSt) {       sgms[s] = a + b * (Psis[s] - Psis[1]);   }
  C[1] = 0.0;   C[K-1] = 1.0;   v = 0.0;
  for (k in 2:(K-2)) {       v = v + preC[k-1];       C[k] = C[1] + v;   }
  for (s in 1:NSt) {       theta[s][1] = normal_cdf(C[1], Psis[s], sgms[s]);
    theta[s][K] = 1.0 - normal_cdf(C[K-1], Psis[s], sgms[s]);
    for (k in 2:(K-1)) {   theta[s][k] = normal_cdf(C[k], Psis[s], sgms[s]) -
                           normal_cdf(C[k-1], Psis[s], sgms[s]);  }   }   }
model {   prePsis ~ dirichlet(a_Psis);   preC ~ dirichlet(a_C);
  a ~ exponential(0.001);   b ~ exponential(0.001);
  for (s in 1:NSt) {     Rs[s] ~ multinomial(theta[s]);   }   }
```

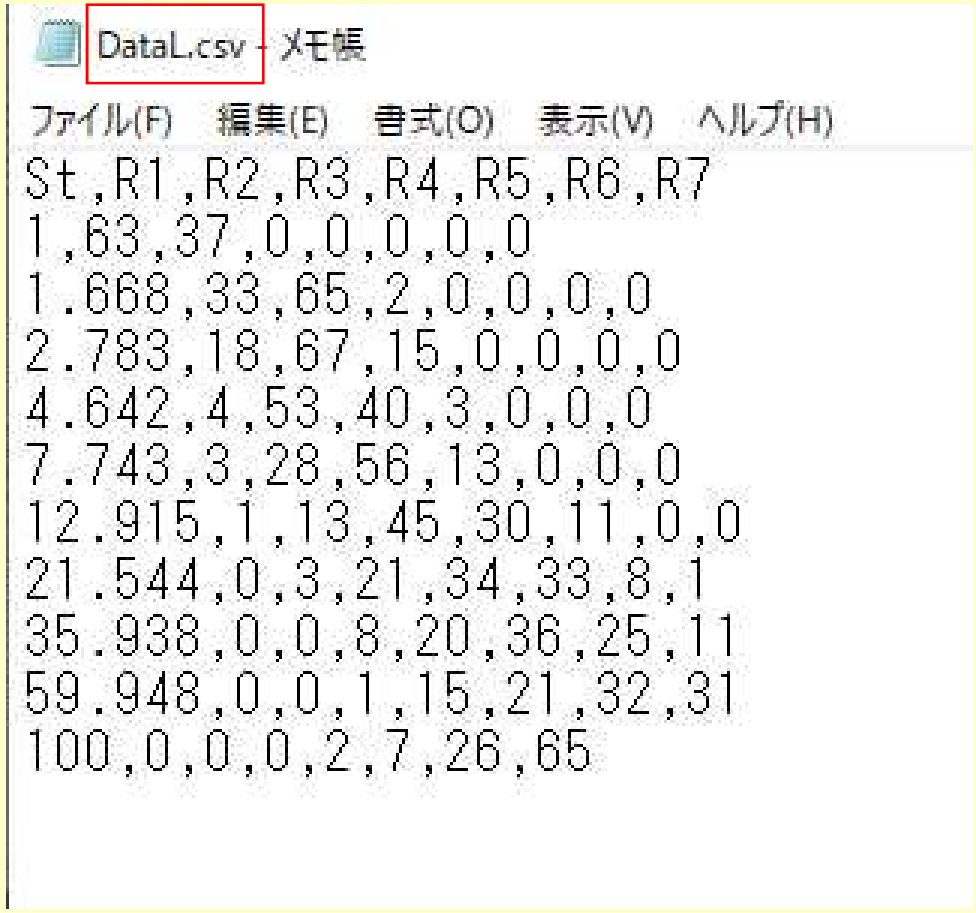
刺激値	非常に弱い	弱い	弱い？	？？？	強い？	強い	非常に強い
1	63	37	0	0	0	0	0
1.668	33	65	2	0	0	0	0
2.783	18	67	15	0	0	0	0
4.642	4	53	40	3	0	0	0
7.743	3	28	56	13	0	0	0
12.915	1	13	45	30	11	0	0
21.544	0	3	21	34	33	8	1
35.938	0	0	8	20	36	25	11
59.948	0	0	1	15	21	32	31
100	0	0	0	2	7	26	65

CSVファイルとして保存



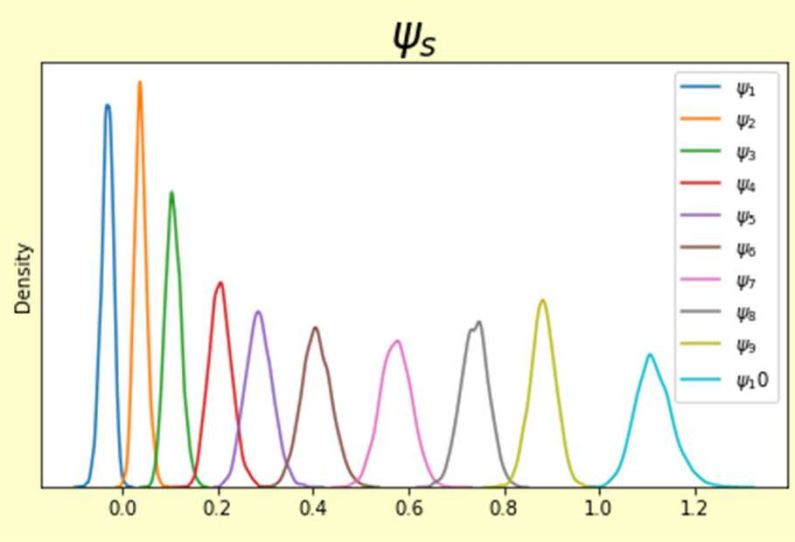
	A	B	C	D	E	F	G	H	I
1	St	R1	R2	R3	R4	R5	R6	R7	
2	1	63	37	0	0	0	0	0	
3	1.668	33	65	2	0	0	0	0	
4	2.783	18	67	15	0	0	0	0	
5	4.642	4	53	40	3	0	0	0	
6	7.743	3	28	56	13	0	0	0	
7	12.915	1	13	45	30	11	0	0	
8	21.544	0	3	21	34	33	8	1	
9	35.938	0	0	8	20	36	25	11	
10	59.948	0	0	1	15	21	32	31	
11	100	0	0	0	2	7	26	65	
12									
13									

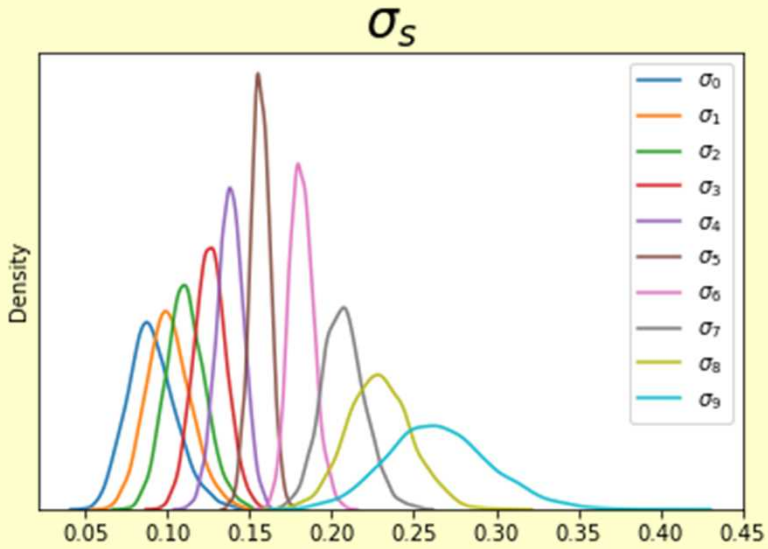
CSV (Comma Separated Values)

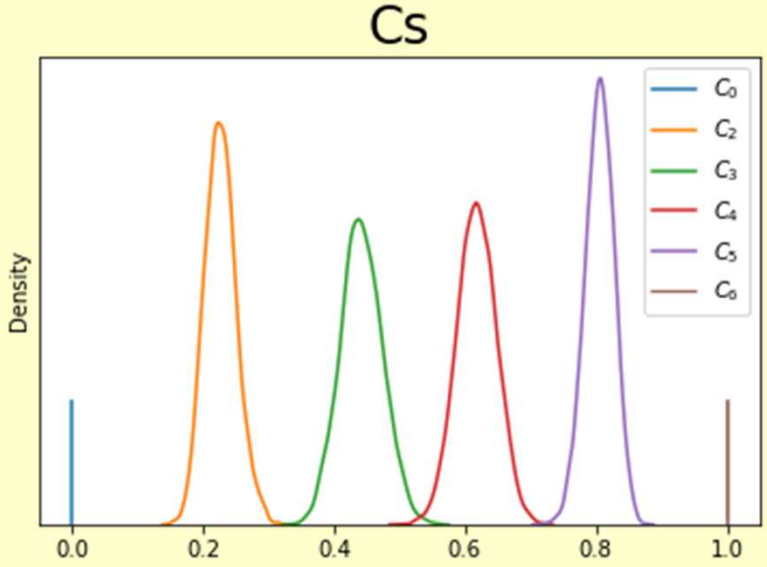


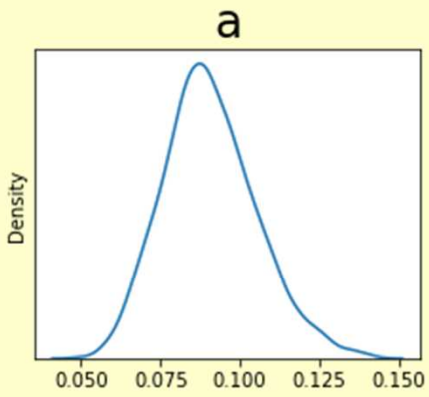

```
fin_nm = input('Input file (*.csv) = ')
with open(fin_nm, 'r') as f:
    Data_in = [v for v in csv.reader(f)]
```

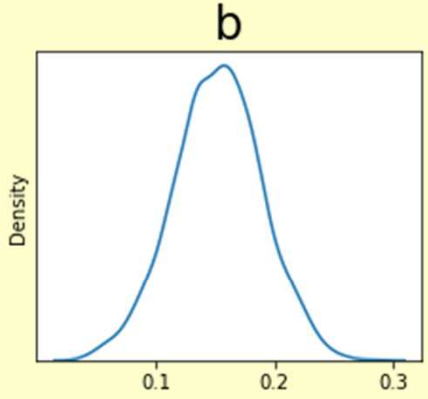
```
(py39) PS V:¥XXXXX¥SigmaLinear> python CategoryScaleL.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
Input file (*.csv) = DataL.csv
       1   63   37    0    0    0    0    0
   1.668   33   65    2    0    0    0    0
   2.783   18   67   15    0    0    0    0
   4.642    4   53   40    3    0    0    0
   7.743    3   28   56   13    0    0    0
  12.915    1   13   45   30   11    0    0
  21.544    0    3   21   34   33    8    1
  35.938    0    0    8   20   36   25   11
  59.948    0    0    1   15   21   32   31
     100    0    0    0    2    7   26   65
INFO:pystan:COMPILING THE C++ CODE FOR MODEL
anon_model_1d9b1507946c231a3d232412be5e37ee NOW.
```

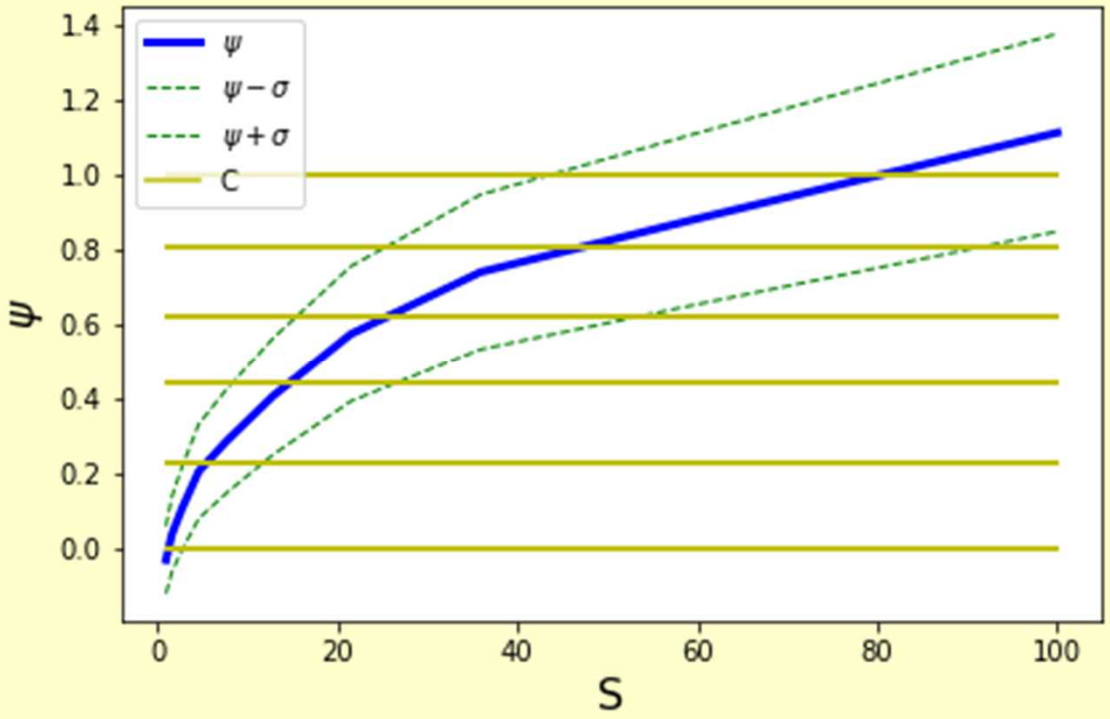

ψ_s
Density
ψ_1
ψ_2
ψ_3
ψ_4
ψ_5
ψ_6
ψ_7
ψ_8
ψ_9
ψ_{10}
0.0
0.2
0.4
0.6
0.8
1.0
1.2

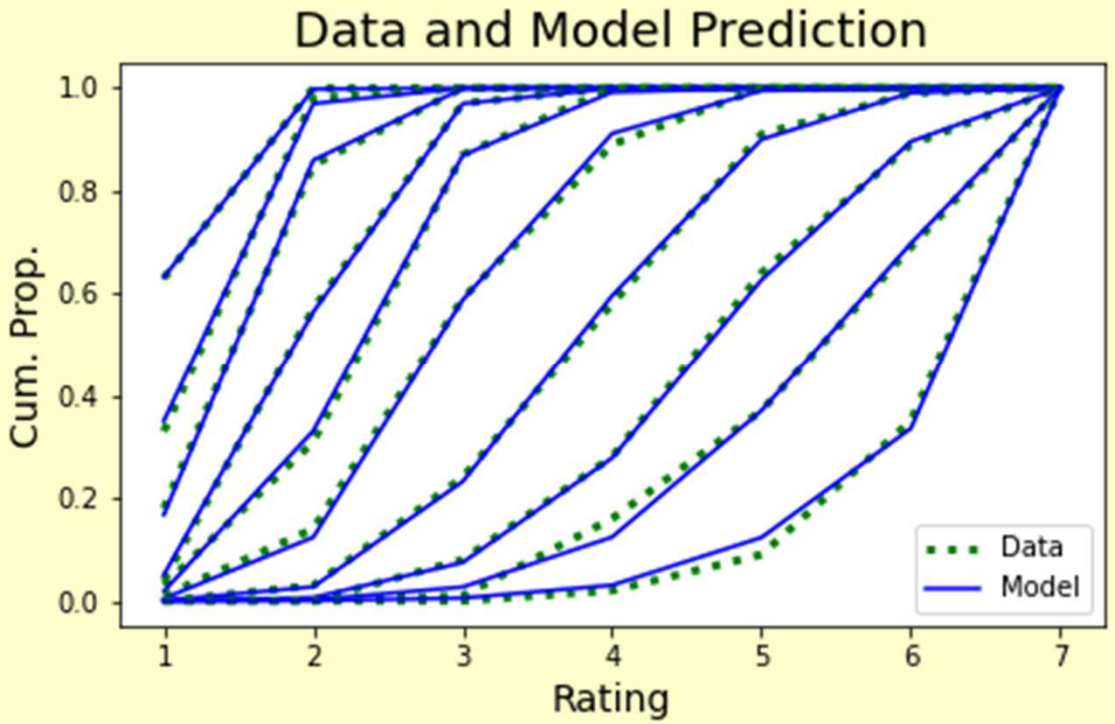

σ_s
Density
σ_0
σ_1
σ_2
σ_3
σ_4
σ_5
σ_6
σ_7
σ_8
σ_9
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45


Cs
Density
C_0
C_2
C_3
C_4
C_5
C_6
0.0
0.2
0.4
0.6
0.8
1.0


a
Density
0.050
0.075
0.100
0.125
0.150


b
Density
0.1
0.2
0.3


ψ
ψ − σ
ψ + σ
C
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
ψ
0
20
40
60
80
100
S


Data and Model Prediction
Cum. Prop.
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
Rating
Data
Model

差尺度法

Maximum likelihood difference scaling (MLDS)

Maloney, L. T. & Yang, J. N. (2003). “Maximum likelihood difference scaling.” *Journal of Vision*, 3, 573-585.

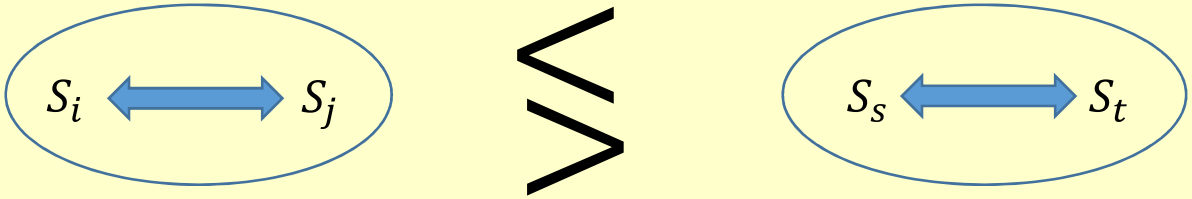
Kingdom, F. A. A. & Prins, N. (2016). “*Psychophysics*, 2nd ed.” AP.

ベイズ法

Bayesian Analysis for difference scaling (BADS)

モデルの調整に対応し易い

差の比較



ψ_i： S_iの感覚量　　$I_{ij} = |\psi_i - \psi_j|$

$X_{(i,j)(s,t)}$ = S_iとS_jの感覚の差とS_sとS_tの感覚の差の差

$$X_{(i,j)(s,t)} \sim N(I_{ij} - I_{st}, \sigma^2)$$

$$P_{(i,j)>(s,t)} = P(S_i\text{と}S_j\text{の差} > S_s\text{と}S_t\text{の差}) = P(X_{(i,j)(s,t)} > 0) = \Phi\left(\frac{I_{ij} - I_{st}}{\sigma}\right)$$

S_iとS_jの差とS_sとS_tの差の$TN_{i,j,s,t}$回の比較において

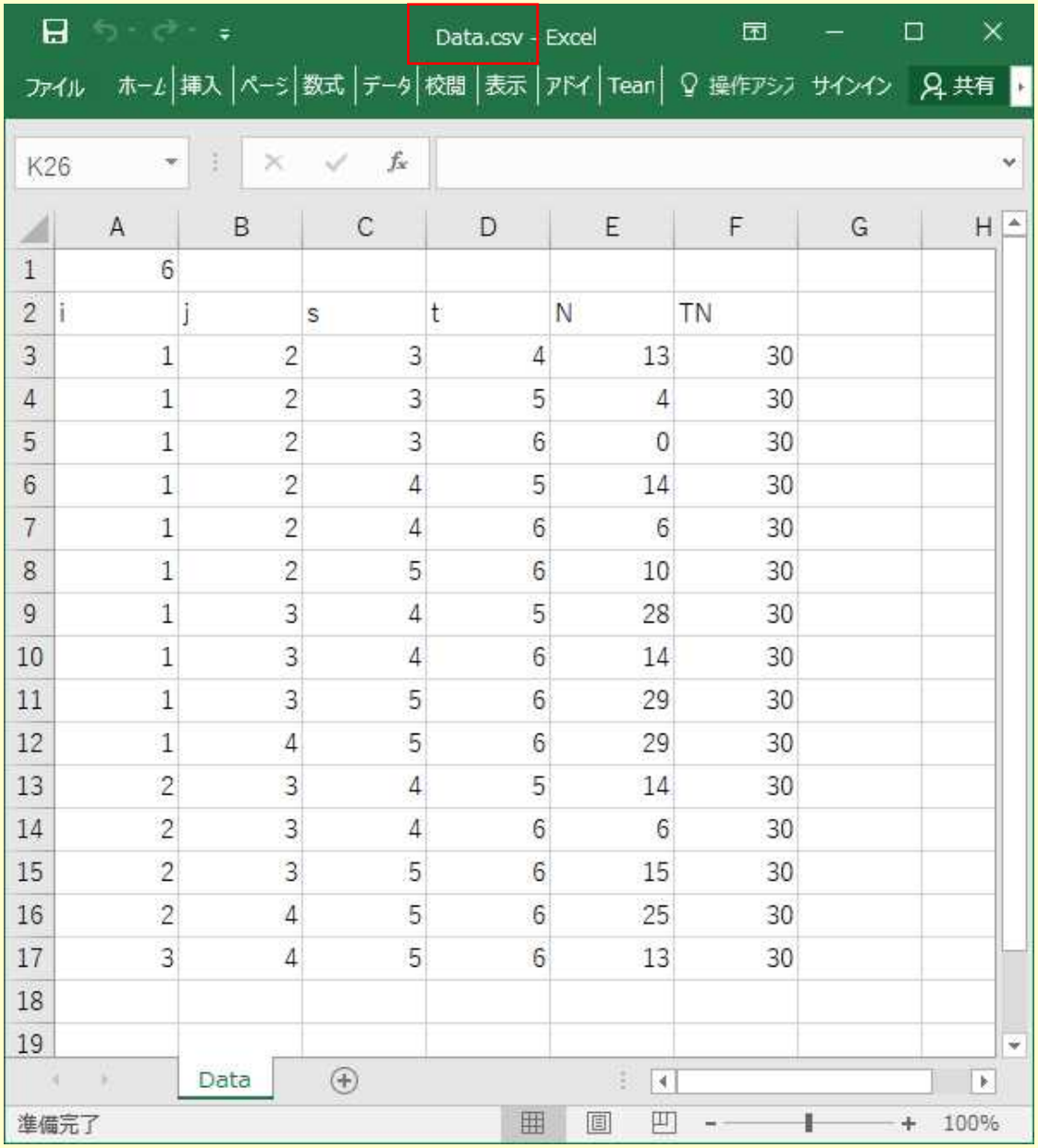
S_iとS_jの差の方がS_sとS_tの差より大きいと判断された回数が$N_{(i,j)>(s,t)}$回のとき、

$$\text{確率} = {}_{TN_{i,j,s,t}}C_{N_{(i,j)>(s,t)}} P_{(i,j)>(s,t)}{}^{N_{(i,j)>(s,t)}} \left(1 - P_{(i,j)>(s,t)}\right)^{TN_{i,j,s,t} - N_{(i,j)>(s,t)}}$$

ψ_i： S_iの感覚量

$$0 = \psi_1 < \cdots < \psi_N = 1$$

```
data {
   int N_data;    int N_st;
   int<lower = 1, upper = N_st> ID_i[N_data];
   int<lower = 1, upper = N_st> ID_j[N_data];
   int<lower = 1, upper = N_st> ID_s[N_data];
   int<lower = 1, upper = N_st> ID_t[N_data];
   int<lower = 0> N[N_data];    int<lower = 1> TN[N_data];
}
parameters {
   simplex[N_st - 1] theta;    real<lower = 0.0> sgm;
}
transformed parameters {
   real psi[N_st];    psi[1] = 0.0;
   for (i in 2:N_st)  psi[i] = psi[i - 1] + theta[i - 1];
}
model {
   for (i in 1:N_data)
      N[i] ~ binomial(TN[i],
                  Phi((fabs(psi[ID_i[i]] - psi[ID_j[i]])
                   - fabs(psi[ID_s[i]] - psi[ID_t[i]])) / sgm));
}
```

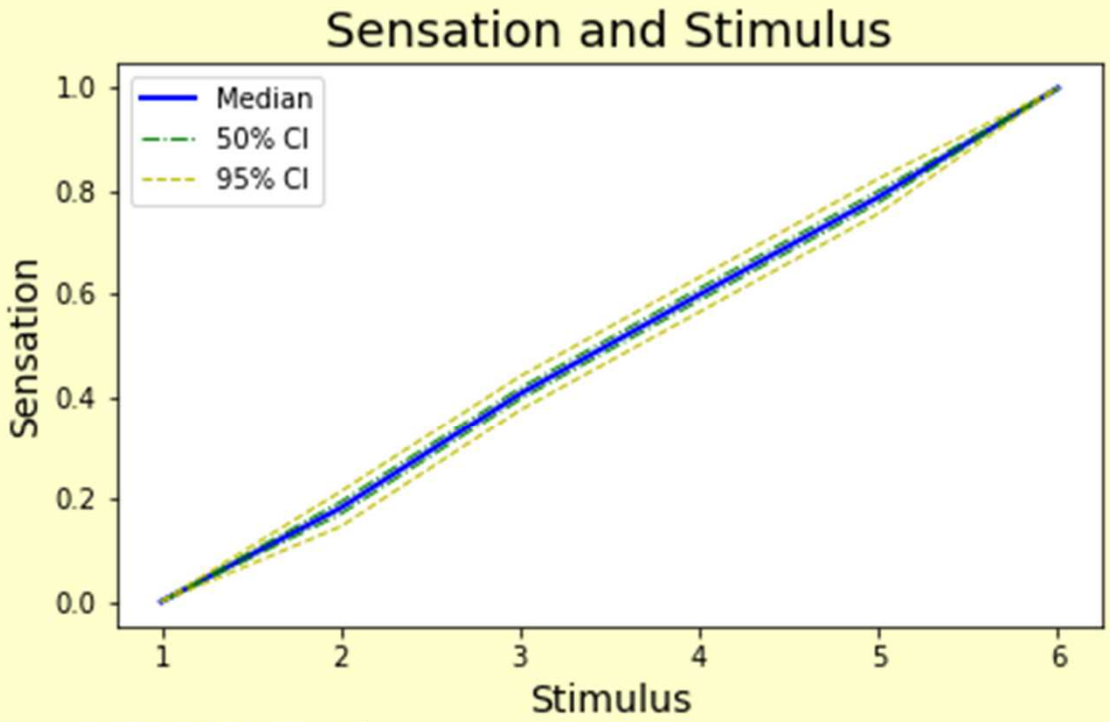


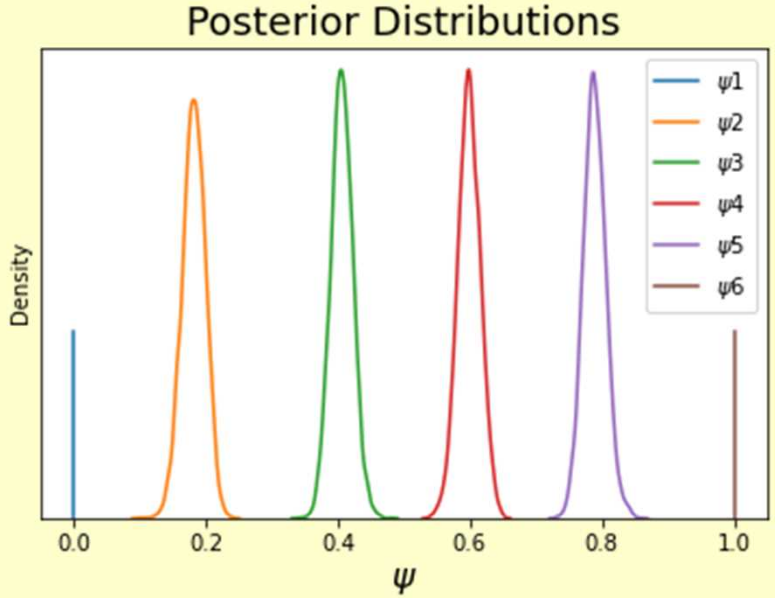
	A	B	C	D	E	F
1	6					
2	i	j	s	t	N	TN
3	1	2	3	4	13	30
4	1	2	3	5	4	30
5	1	2	3	6	0	30
6	1	2	4	5	14	30
7	1	2	4	6	6	30
8	1	2	5	6	10	30
9	1	3	4	5	28	30
10	1	3	4	6	14	30
11	1	3	5	6	29	30
12	1	4	5	6	29	30
13	2	3	4	5	14	30
14	2	3	4	6	6	30
15	2	3	5	6	15	30
16	2	4	5	6	25	30
17	3	4	5	6	13	30
18						
19						

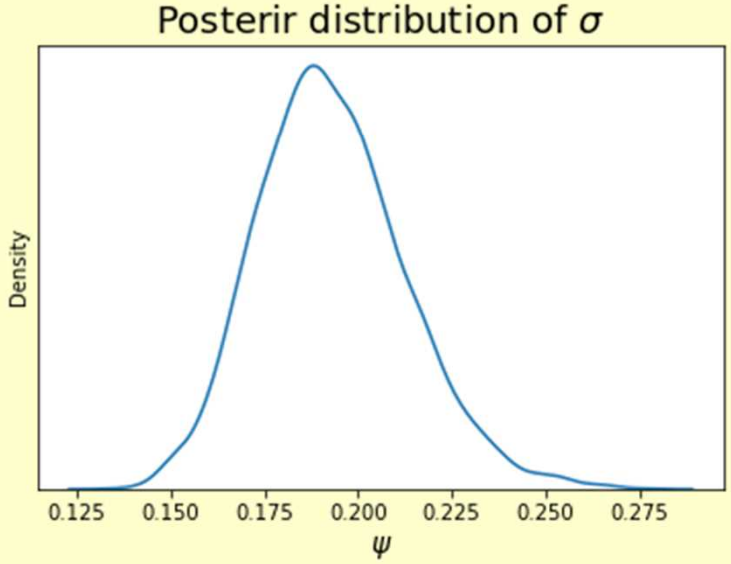
CSVファイル

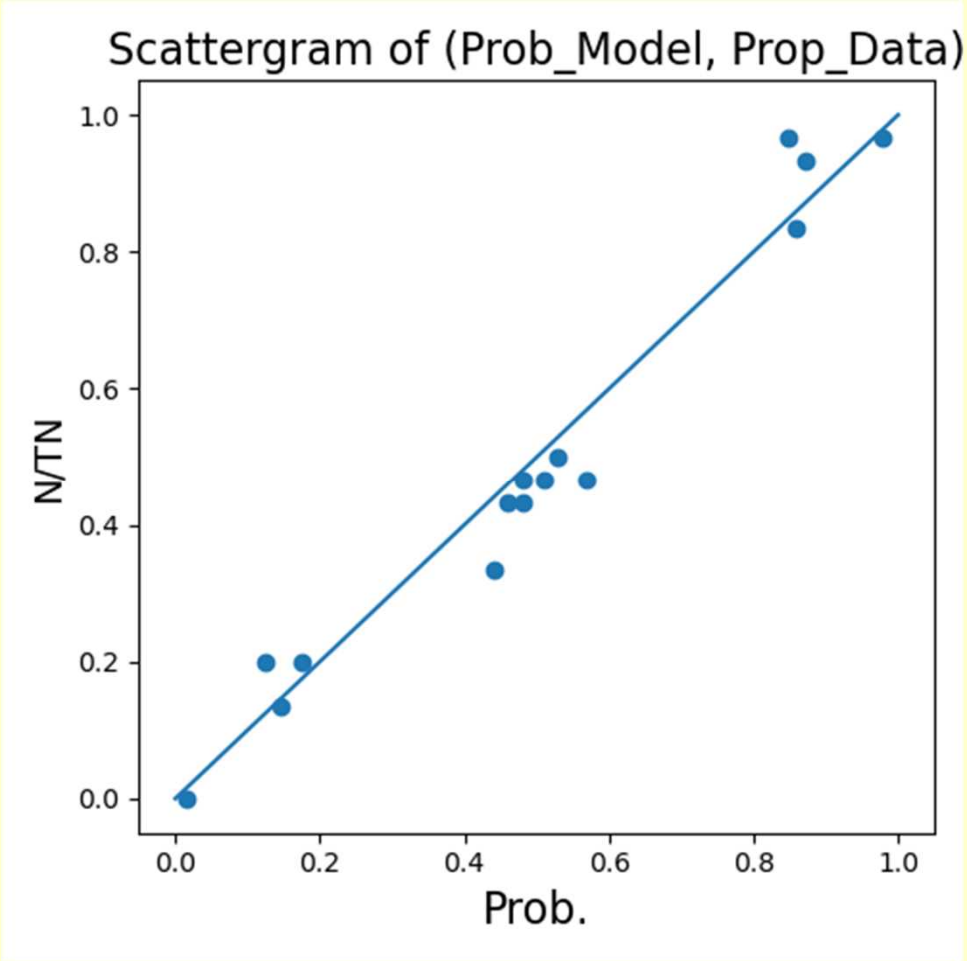
```
fin_nm = input('Data(*.csv) = ')
with open(fin_nm, 'r') as f:
    raw_data =
        [v for v in csv.reader(f)]
```

```
(py39) PS V:¥XXXXX¥BA_Diff_Scale> python BDS.py
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
Data(*.csv) = Data.csv
```


Sensation and Stimulus
Median
50% CI
95% CI
Sensation
Stimulus
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
6


Posterior Distributions
ψ1
ψ2
ψ3
ψ4
ψ5
ψ6
Density
0.0
0.2
0.4
0.6
0.8
1.0
ψ


Posterir distribution of σ
Density
0.125
0.150
0.175
0.200
0.225
0.250
0.275
ψ


Scattergram of (Prob_Model, Prop_Data)
N/TN
Prob.
0.0
0.2
0.4
0.6
0.8
1.0

PythonとPyStan（Pthon用Stan）のインストールの説明ウェブサイト例

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

PythonとPyStan（Pthon用Stan）の説明

岡本安晴「いまさら聞けないPythonでデータ分析」丸善出版

確率分布

Stan

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

scipy.stats

https://docs.scipy.org/doc/scipy/reference/stats.html#

一様分布

区間[0, 1]で一様分布

$$P(U) = \begin{cases} 1 & 0 \leq U \leq 1 \\ 0 & \text{その他} \end{cases}$$

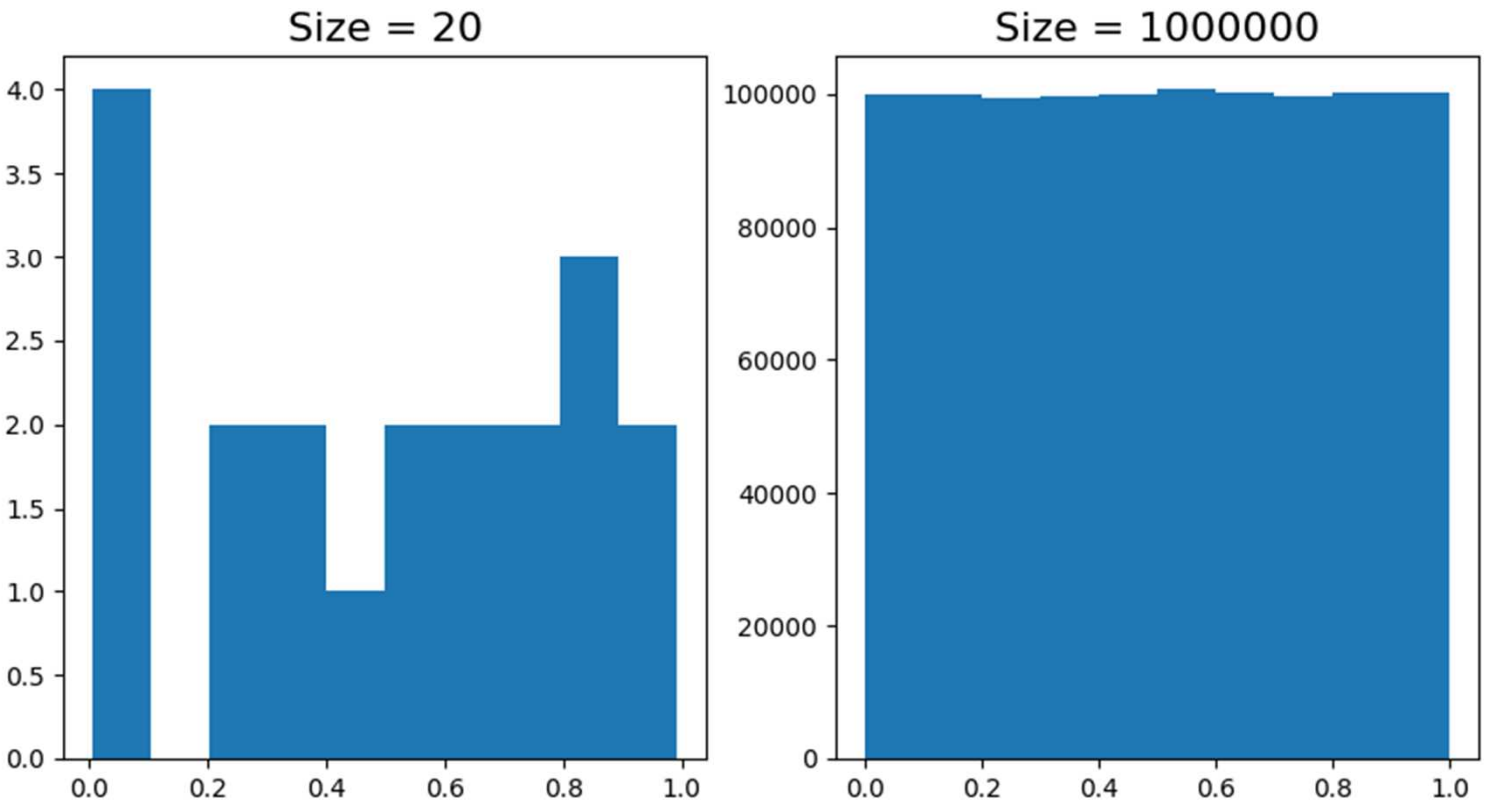
区間[a, b]で一様分布

$$P(U) = \begin{cases} 1/(b-a) & a \leq U \leq b \\ 0 & \text{その他} \end{cases}$$

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

s_size = 20
sample = ss.uniform.rvs(size = s_size)
plt.figure(figsize = (10, 5))
plt.subplot(1,2,1)
plt.hist(sample)
plt.title(f'Size = {s_size}', fontsize = 16)

s_size = 1000_000
sample = ss.uniform.rvs(size = s_size)
plt.subplot(1,2,2)
plt.hist(sample)
plt.title(f'Size = {s_size}', fontsize = 16)
plt.savefig('Fig_uni1.png')
plt.show()
```

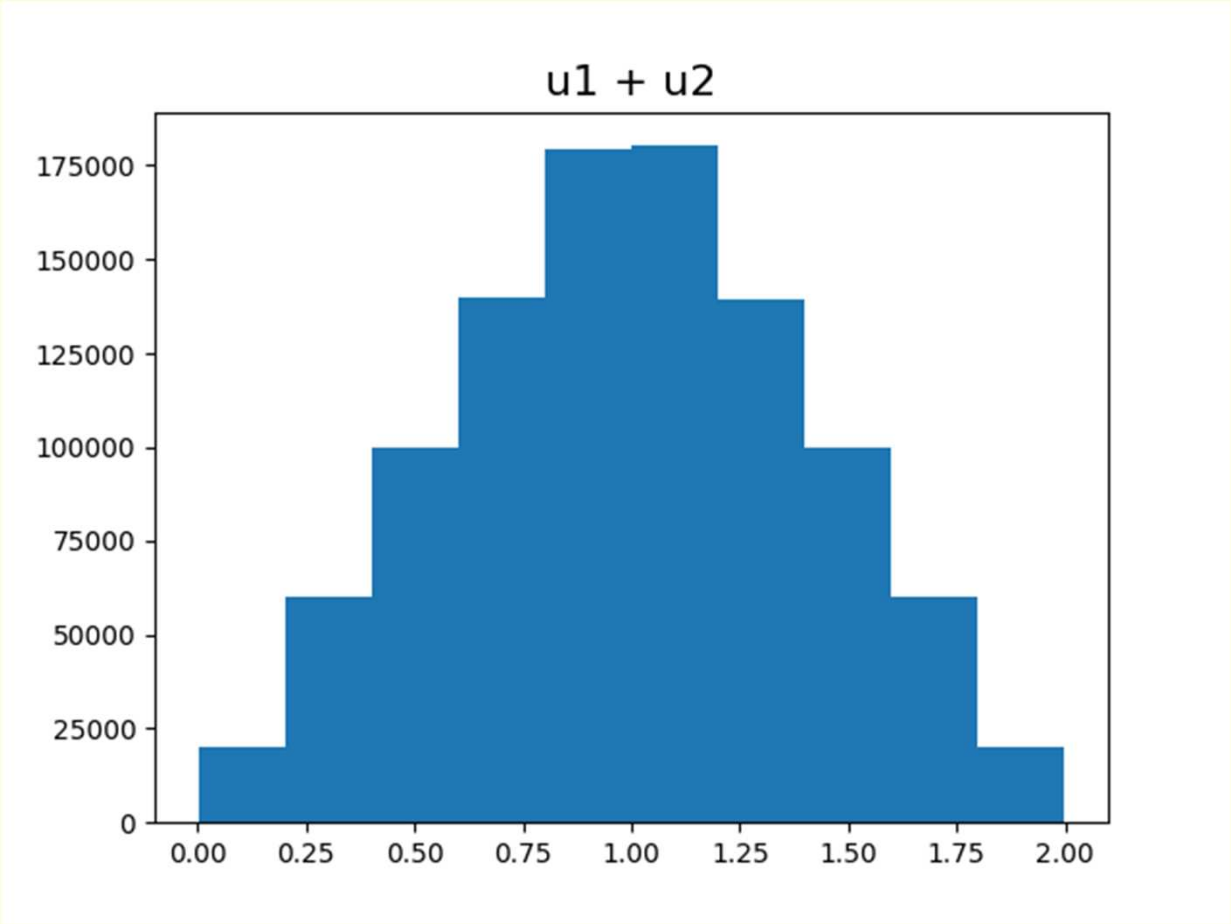

Size = 20
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Size = 1000000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

一様分布の和

$S = U_1 + U_2$、　U_1：一様分布

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

s_size = 1000_000
u1 = ss.uniform.rvs(size = s_size)
u2 = ss.uniform.rvs(size = s_size)
sample = u1 + u2
plt.hist(sample)
plt.title('u1 + u2', fontsize = 16)
plt.savefig('Fig_uni2.png')
plt.show()
```

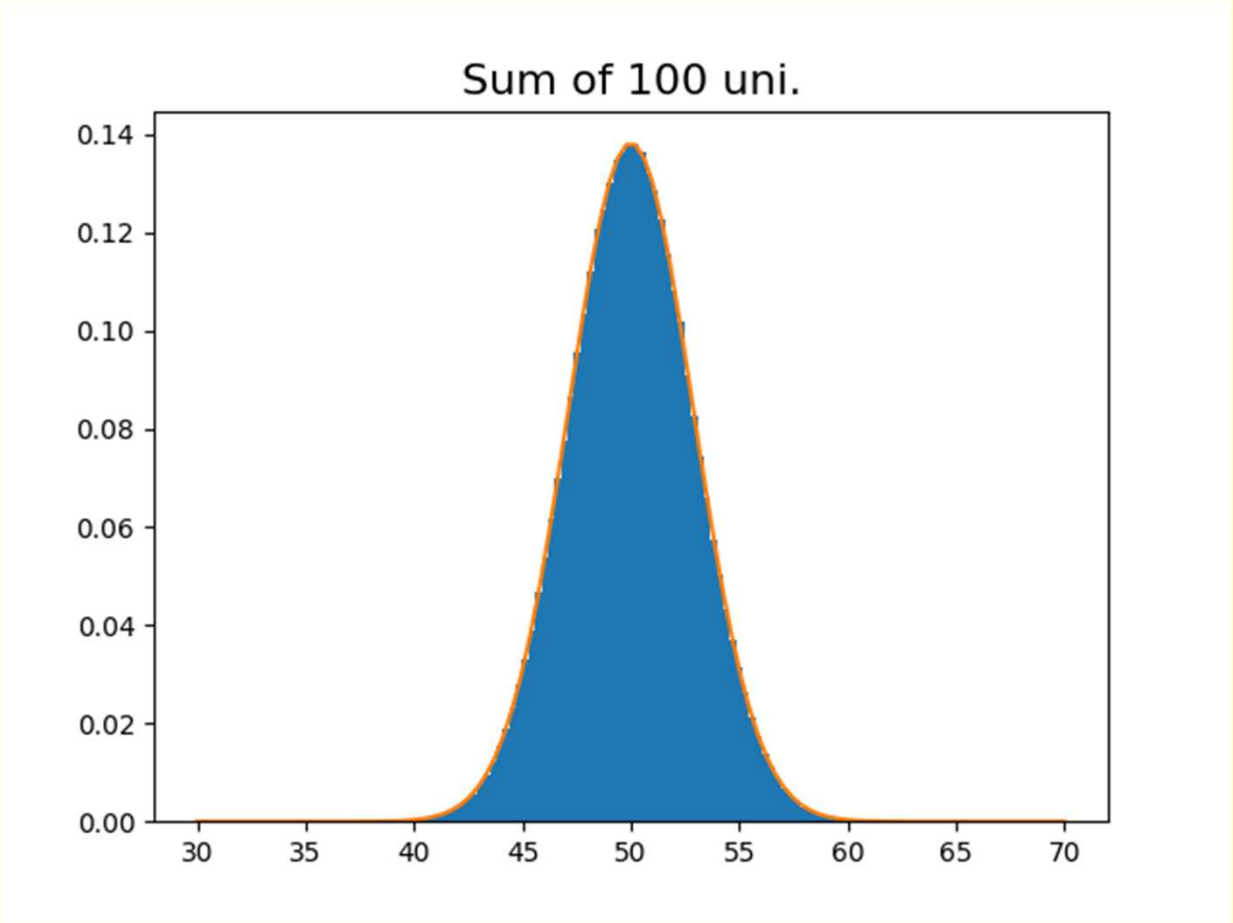

u1 + u2
175000
150000
125000
100000
75000
50000
25000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

一様分布の100個の和

$S = U_1 + \cdots + U_{100}$、　U_1：一様分布

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

s_size = 1000_000
us = np.empty((100, s_size))
for i in range(100):
    us[i] = ss.uniform.rvs(size = s_size)
sample = np.sum(us,axis = 0)
plt.hist(sample, bins = 100, density = True)
plt.title('Sum of 100 uni.', fontsize = 16)
x_coord = np.linspace(30, 70, 100)
y_coord = ss.norm.pdf(x_coord, loc = 50, scale = 5 / (3**0.5))
plt.plot(x_coord, y_coord)
plt.savefig('Fig_uni100.png')
plt.show()
```

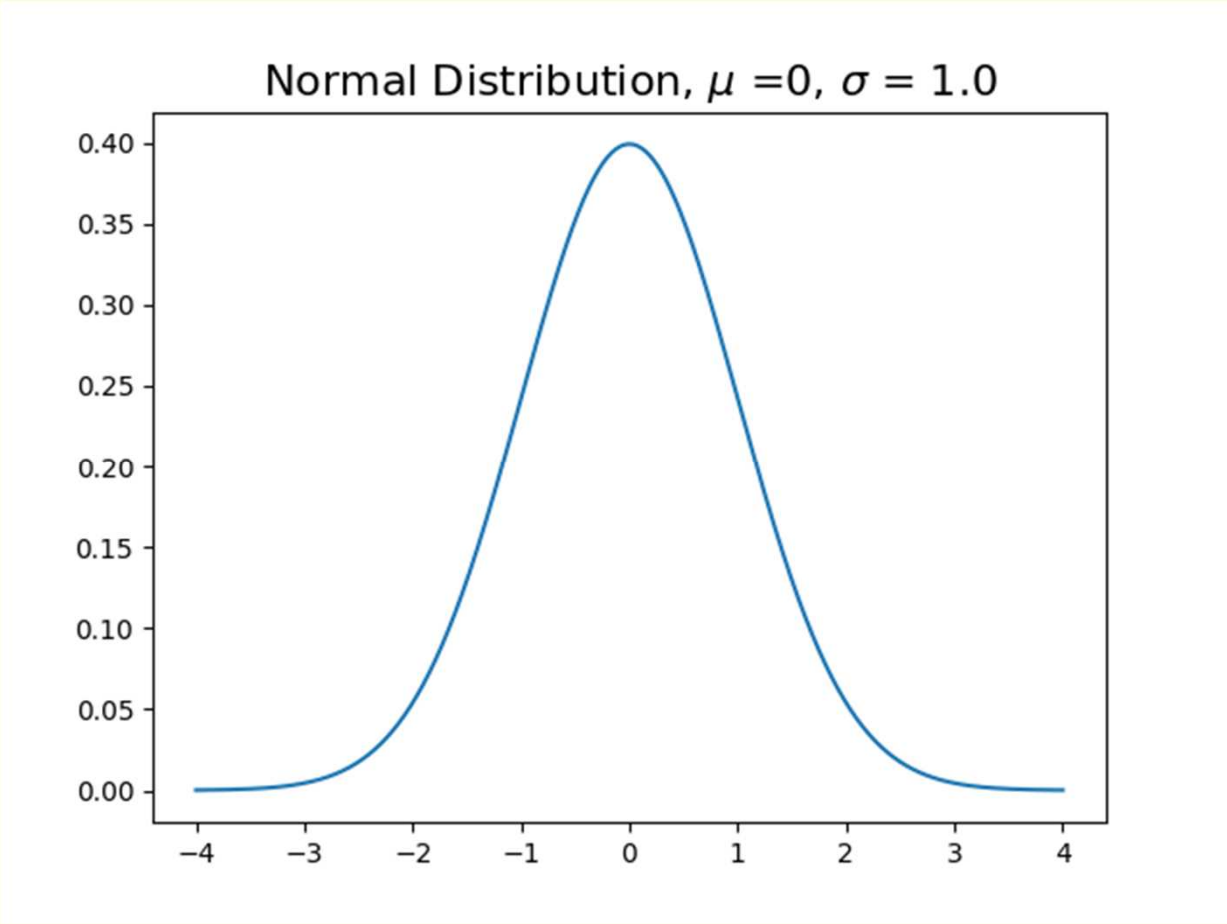

Sum of 100 uni.
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
30
35
40
45
50
55
60
65
70

正規分布

$$Normal(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\cdot\sigma} exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

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

x_coord = np.linspace(-4, 4, 1001)
y_coord = ss.norm.pdf(x_coord, loc = 0.0, scale = 1.0)
plt.plot(x_coord, y_coord)
plt.title('Normal Distribution, $¥mu$ =0, $¥sigma$ = 1.0', fontsize = 16)
plt.savefig('Fig_normal.png')
plt.show()
```


Normal Distribution, μ =0, σ = 1.0
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4

2 項分布

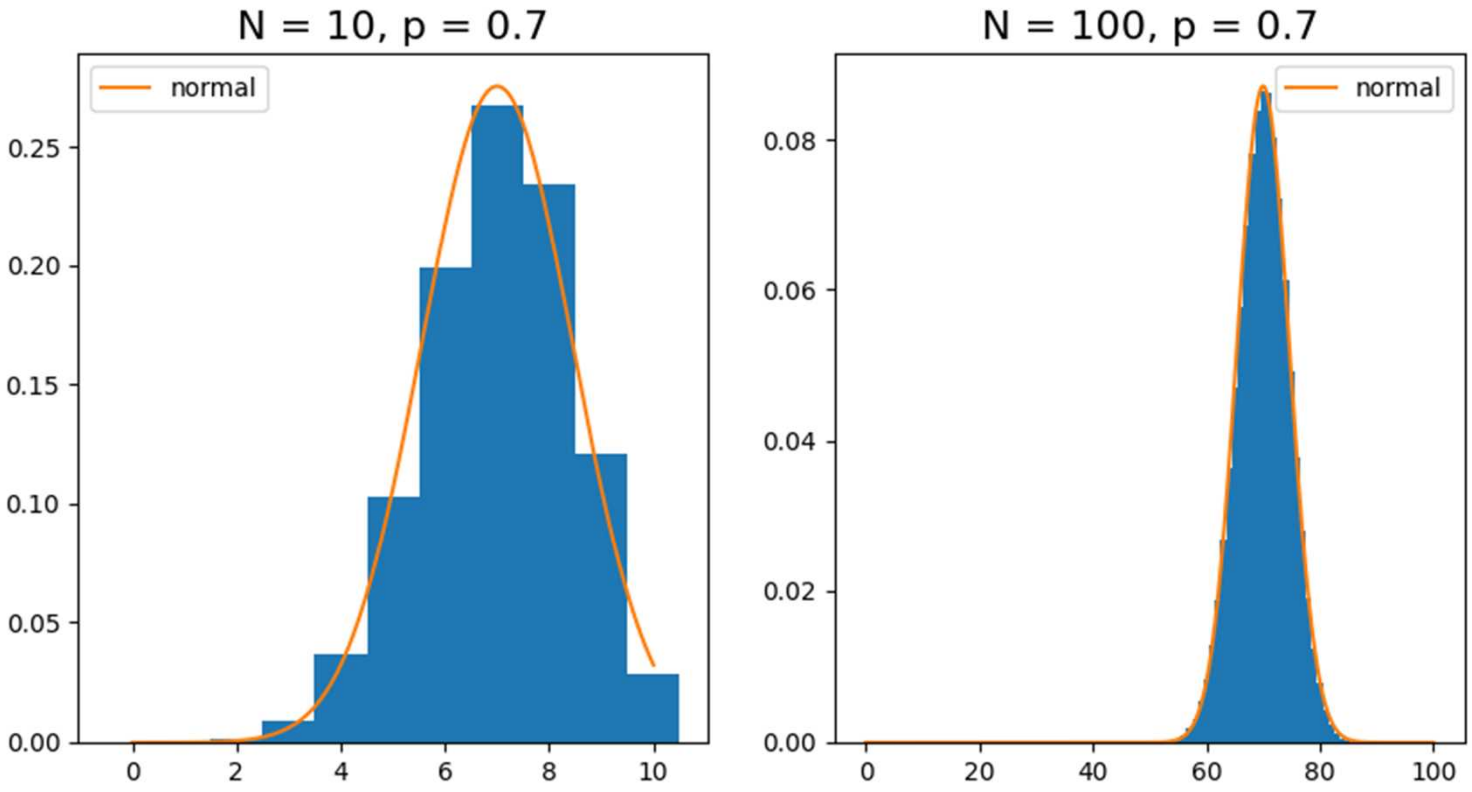
N回の試行において k 回成功：　各試行における成功確率 p

$$P(k) = \binom{N}{k} p^k (1-p)^{N-k}$$

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

p = 0.7
s_size = 1000_000

plt.figure(figsize = (10, 5))
for i, N in enumerate([10, 100]):
    plt.subplot(1,2,i+1)
    s = ss.binom.rvs(N, p, size = s_size)
    plt.hist(s, bins = np.linspace(-0.5, N+0.5, N+2), density = True)
    x_coord = np.linspace(0, N, 1001)
    y_coord = ss.norm.pdf(x_coord, loc = p*N, scale = (p*(1-p)*N)**0.5)
    plt.plot(x_coord, y_coord, label = 'normal')
    plt.title(f'N = {N}, p = {p}', fontsize = 16)
    plt.legend()
    plt.savefig('FigBin10.png')
plt.savefig('Figbinomials.png')
plt.show()
```

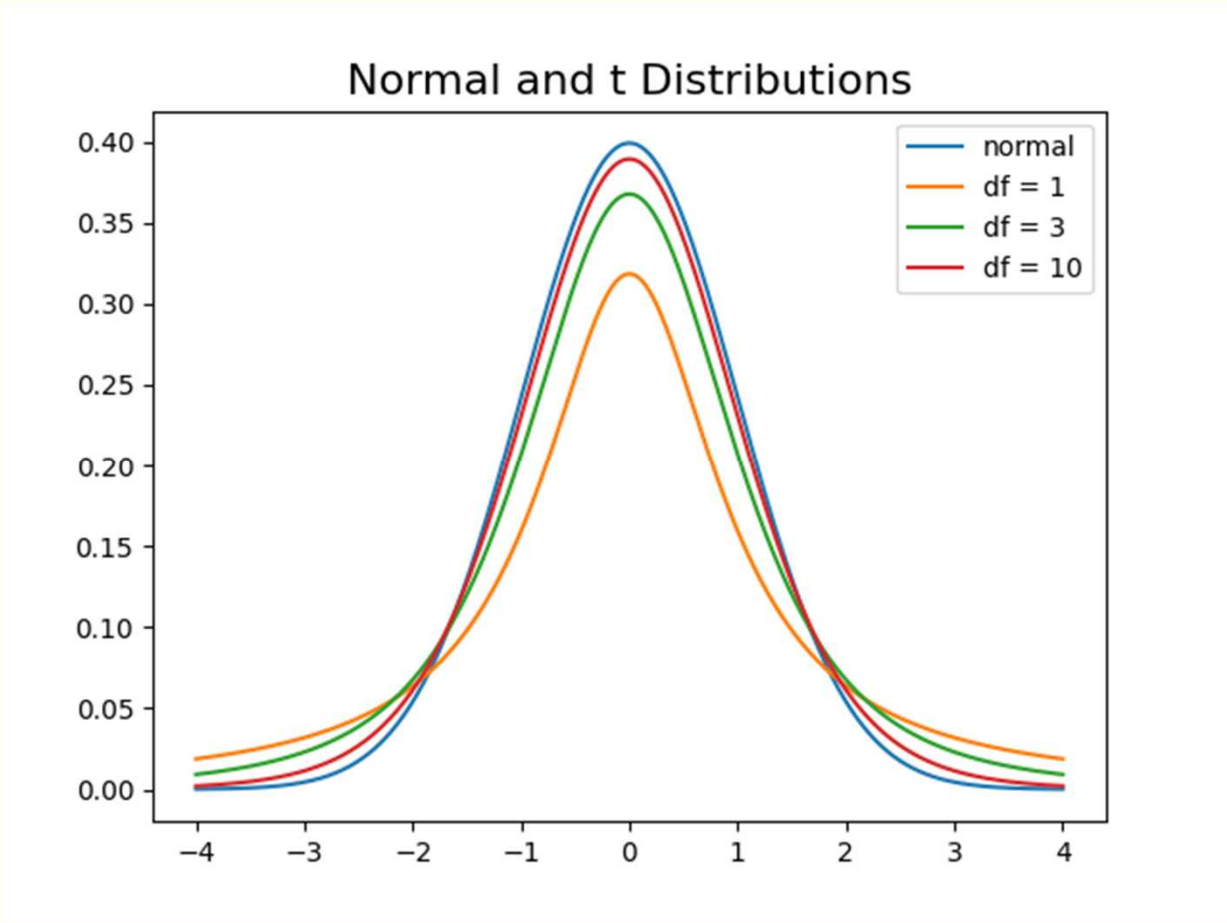

N = 10, p = 0.7
normal
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
N = 100, p = 0.7
normal
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100

t 分布

$$t_\nu(x;\mu,\sigma^2) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\cdot\sigma}\left(1+\frac{1}{\nu}\left(\frac{x-\mu}{\nu}\right)^2\right)^{-(\nu+1)/2}$$

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

x_coord = np.linspace(-4, 4, 1001)
y_coord = ss.norm.pdf(x_coord, loc = 0.0, scale = 1.0)
plt.plot(x_coord, y_coord, label = 'normal')
for df in [1,3,10]:
    y_coord = ss.t.pdf(x_coord, df)
    plt.plot(x_coord, y_coord, label = f'df = {df}')
plt.title('Normal and t Distributions', fontsize = 16)
plt.legend()
plt.savefig('Fig_normal_t.png')
plt.show()
```


Normal and t Distributions
normal
df = 1
df = 3
df = 10
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4

指数分布

$G(t) = P(w \leq t)$：　待ち時間wがt以下の確率

$$\frac{G(t+\Delta t) - G(t)}{1 - G(t)} = \beta \cdot \Delta t + o(\Delta t)$$

：　待ち時間が
t以上であるとき、
時刻tから時刻$t + \Delta t$
の間に事象が生起する確率

$$\frac{d}{dt}(G(t) - 1) = -\beta(G(t) - 1)$$

$$Exponential(t, \beta) = G'(t) = P(w = t) = \beta \cdot exp(-\beta t)$$

指数分布

$$Exponential(t, \beta) = G'(t) = P(w = t) = \beta \cdot exp(-\beta t)$$

$$E(t) = \frac{1}{\beta}, \qquad t > 0$$

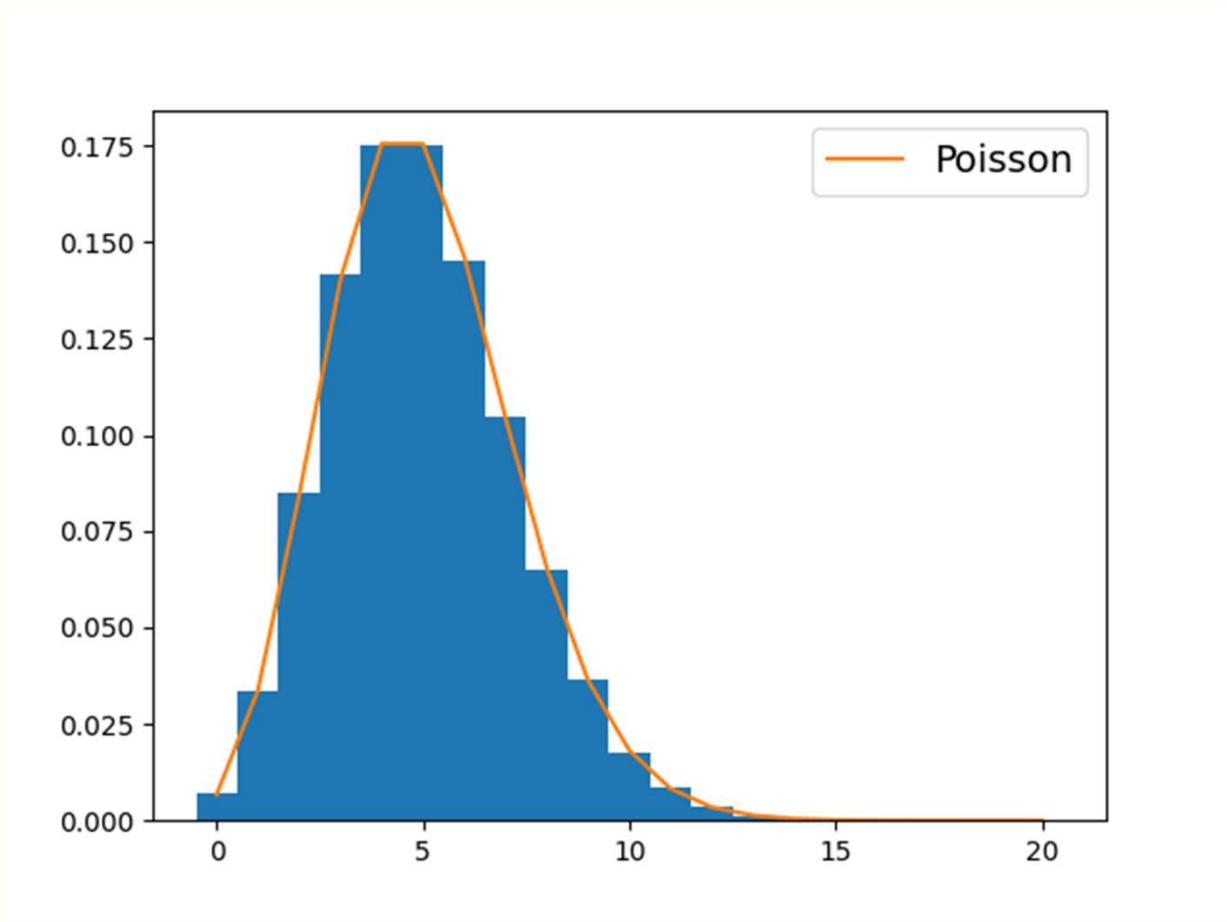
ポアッソン分布

$$Poisson(x, \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}, \qquad x = 0,1,\cdots$$

$$E[x] = \lambda$$

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

beta = 1.0 / 5.0;  lmbd = 1.0 / beta;  n_simu = 100_000
counts = np.empty(n_simu, dtype = 'int')
for i in range(n_simu):
    if i % 1000 == 0:
        print(i, end = ' ')
    c = 0;      sum_t = 0
    while True:
        sum_t += ss.expon.rvs(scale = beta)
        if sum_t > 1.0:
            break
        c += 1
    counts[i] = c
plt.hist(counts, bins = np.linspace(-0.5, 20.5, 22), density = True)
x_coord = np.linspace(0, 20, 21)
y_coord = ss.poisson.pmf(x_coord, lmbd)
plt.plot(x_coord, y_coord, label = 'Poisson')
plt.legend(fontsize = 14)
plt.show()
```


Poisson
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
5
10
15
20

まとめ

順序カテゴリ尺度

カテゴリ判断の法則に基づく尺度構成

差尺度

差の 2 肢強制選択較データに基づく尺度構成

確率分布

幾つかの確率分布のシミュレーション例

Q and A