AI応用セミナー

第01回

賭け事と対数正規分布

改訂履歴

日付	担当者	内容
2022/11/13	武田 守	初版
2022/11/19	武田 守	章建て、項建てを見直し

目次

賭け事と対数正規分布

(1) はじめに

ここでは、一つの賭け事のモデルを設定し、

n 回賭け事を繰返して X 回成功した時の所持金 y(n) を、対数正規分布で計算します。

(1.1) 問題設定

・初期の所持金を y(0) とします。

・一回の賭け事で成功する確率を p (0≦ p ≦1) とします。

・賭け事を n 回繰返した後の所持金を y(n) とします。

・毎回の賭け金は、その時点の所持金の b 倍 (0< b <1) とします。
賭けに負けた場合、掛け金を失い、勝った場合、掛け金と同額を取得するものとします。

・このルールで、n 回賭け事を繰返して X 回成功した時の所持金 y(n) を計算します。

(1.2) 前提となる主な基礎知識

・「対数関数 (ﾀｲｽｳｶﾝｽｳ、logarithm fnction)」の性質

・「二項分布 (ﾆｺｳﾌﾞﾝﾌﾟ、binominal distribution)」　$P(k)={}_nC_k\, p^k\,(1-p)^{n-k}$

・「標準化確率変数 (ﾋｮｳｼﾞｭﾝｶｶｸﾘﾂﾍﾝｽｳ, standardized random variable)」$Z=(X-\mu)/\sigma$

・「標準正規分布 (ﾋｮｳｼﾞｭﾝｾｲｷﾌﾞﾝﾌﾟ, Standard Normal Distribution)」$N(0,1)$

【出典・参考】

賭け事⇒「その問題 数理モデルが解決します」ベレ出版　浜田宏著 2019年

二項分布⇒ 本セミナー「第６回 数学の基礎（３回目：統計学） (3.2.3) 二項分布」

対数正規分布⇒ 本セミナー「第６回 数学の基礎（３回目：統計学） (3.3.2) 対数正規分布」

期待値についての公式⇒ 本セミナー「第６回 数学の基礎（３回目：統計学） (3.4) 平均と確率変数の期待値」

分散についての公式⇒ 本セミナー「第６回 数学の基礎（３回目：統計学） (3.5) 確率変数の分散」

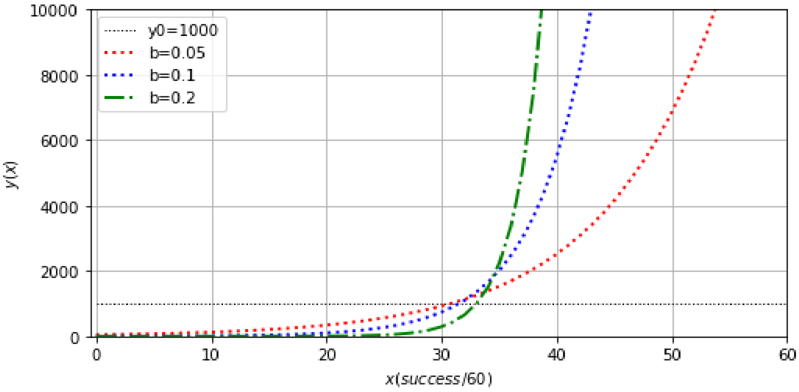
(2) 本論

(2.1) n 回賭け事を繰返したうち成功した回数が X の場合の所持金 y(n)

・上記の問題設定の下で、n 回賭け事を繰返して X 回成功した（n-X 回失敗した）時に、
最終的な所持金 y(n) は次式で与えられます：

$$y(n) = y(0) \times (1 + b)^{X} \times (1 - b)^{n-X} \qquad \text{(式2.1-1)}$$

この式をグラフ化すると以下のようになります（y(0)=1000、n=60 でbを変えたもの）：



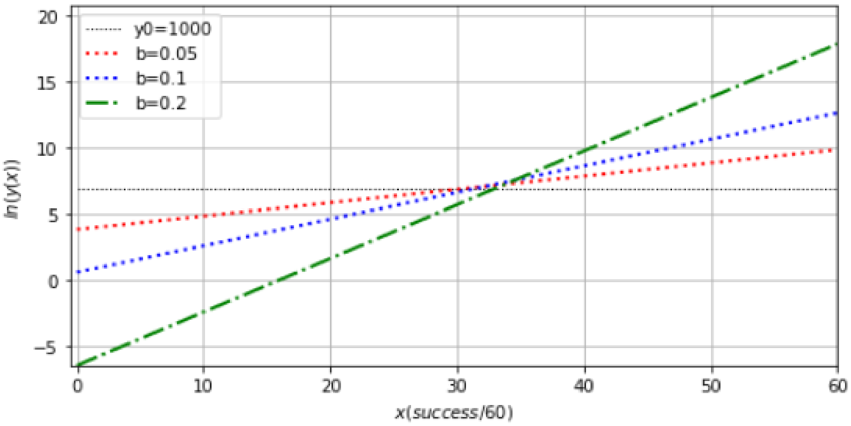
（リスト：賭け事_リスト_1_所持金の分布関数）

・(式2.1-1)の両辺の対数を取ると、以下の関係式が得られます。

$$\begin{aligned} \ln(y(n)) &= \ln(y(0)) + X*\ln(1+b) + (n-X)*\ln(1-b) \\ &= X*\ln\{(1+b)/(1-b)\} + \ln(y(0)) + n*\ln(1-b) \qquad \text{(式2.1-2)} \\ &= \alpha X + \beta \qquad \left\{ \begin{array}{l} \alpha \equiv \ln\{(1+b)/(1-b)\} \\ \beta \equiv \ln(y(0)) + n*\ln(1-b) \\ \ln(a) : a \text{ の自然対数 } (\log_e(a)) \end{array} \right. \end{aligned}$$

これの意味するところは、以下のとおりです：
「n 回賭け事を繰返した結果の所持金 y(n) の対数 ln(y(n)) は、成功回数 X の一次変換になる。」

この式をグラフ化すると以下のようになります（y(0)=1000、n=60 でbを変えたもの）：



（リスト：賭け事_リスト_2_所持金の対数の分布関数）

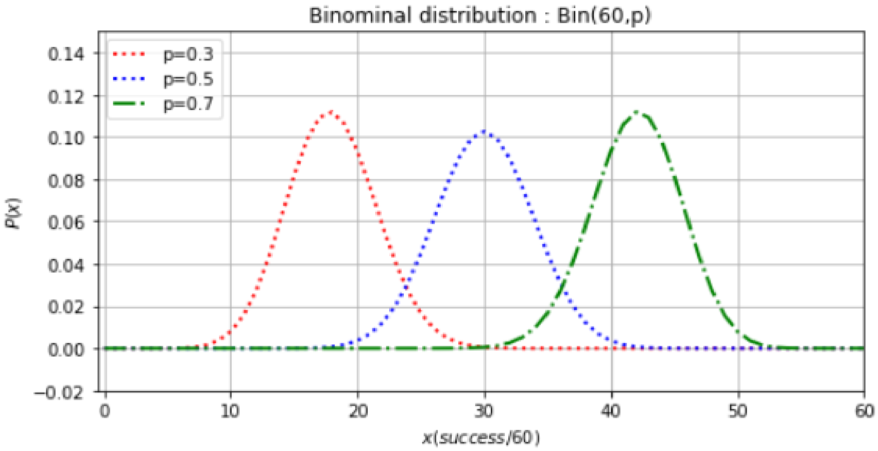
(2.2) n 回賭け事を繰返したうち成功した回数 X が従う確率分布

・一方、賭け事の n 回の試行で成功する回数 X は確率変数であり、
X が従う確率分布は「二項分布 (ﾆｺｳﾌﾞﾝﾌﾟ、binominal distribution)」になります。
その確率質量関数Ｐ(X=x)、平均μ、分散Ｖ、標準偏差σは次式で表されます：

$$P(X=x) = {}_{n}C_{x}\ p^{x}\ (1-p)^{n-x}$$
$$\mu = np$$
$$V = np(1-p)$$
$$\sigma = \mathrm{sqrt}(\ np(1-p)\ )$$
(式2.2-1)

x∈{ 0,1,..,n }
p：一回の試行で成功する確率
sqrt(a)：a の平方根

この式をグラフ化すると以下のようになります (n=60 で pを変えたもの)：



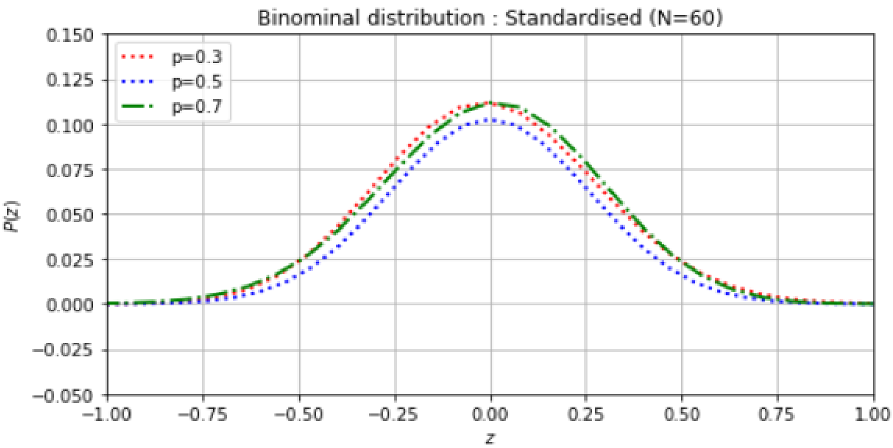
(ﾘｽﾄ：賭け事_ﾘｽﾄ_3_成功回数の二項分布)

この統計諸量により、n 回の試行で成功する回数 X の標準化確率変数 Z を作成します：

$$Z = (X-\mu)\ /\ \sigma$$
$$= (X-np)\ /\ \mathrm{sqrt}(np(1-p))$$
(式2.2-2)

この標準化確率変数 Z は「二項分布に関する中心極限定理 (ド・モアブル＝ラプラスの定理)」
により、n⇒∞の時、標準化確率変数 Z は標準正規分布 N(0,1) に近づきます。

標準化をグラフ化すると以下のようになります (n=60 で pを変えたもの)：



(ﾘｽﾄ：賭け事_ﾘｽﾄ_4_成功回数の標準化)

確率変数 X (n 回の試行で成功する回数)は、(式2.2-2)より、

$$X = \sigma Z+\mu \quad = \mathrm{sqrt}(np(1-p))Z + np$$
(式2.2-3)

であり、Z の一次変換である X も正規分布となるので、
結局 n⇒∞の時、確率変数 X は正規分布 N(μ,σ)＝N(np, np(1-p)) に近づきます。
これは、上記の(ﾘｽﾄ：賭け事_ﾘｽﾄ_3_成功回数の二項分布) のｸﾞﾗﾌでも見て取れます。

(2.3) n 回賭け事を繰返した結果の所持金 y(n) が従う確率分布

(式2.1-2) より、n 回賭け事を繰返した結果の所持金 y(n) について、

$$\ln(y(n)) \;=\; \alpha X + \beta \qquad \text{(式2.1-2)}$$

となりますが、
正規分布N(μ, σ) に従う X の一次変換である ln(y(n)) も正規分布となります。
ln(y(n)) が従う正規分布N(μ', σ') の平均μ'、分散V'(=σ'^2) は、
期待値と分散についての公式、および(式2.1-2)、(式2.2-1)により

$$\mu' \;=\; \beta+\alpha\mu \;=\; [\ln(y(0))+n*\ln(1-b)] + [\ln\{(1+b)/(1-b)\}]\times np \qquad \text{(式2.3-1)}$$
$$V' \;=\; \alpha^2 V \;=\; [\ln\{(1+b)/(1-b)\}]^2 \times [np(1-p)]$$

なる統計量μ'、V'(=σ'^2) を用いて、以下の結論を得ます：

(1) n⇒∞の時、確率変数 ln(y) (y=y(n)) は正規分布 N(μ', V') に近づく

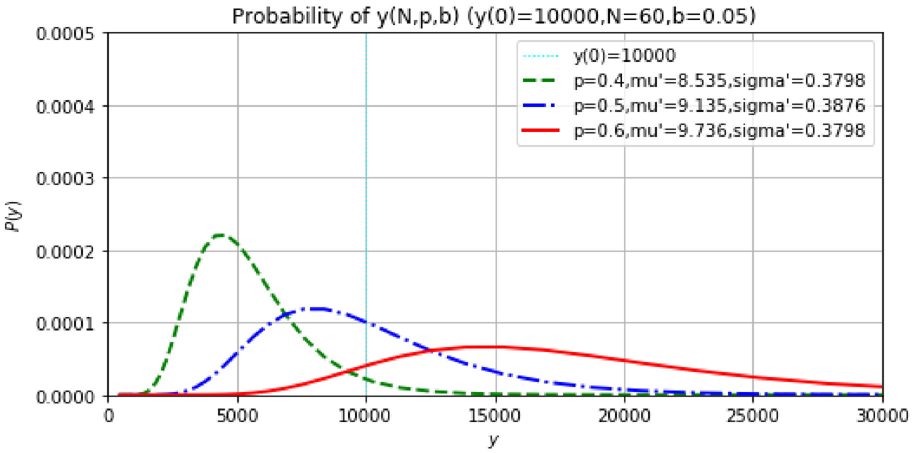
$$f(\ln(y)) \;=\; 1/(\sqrt{2\pi}\,\sigma')\ \exp(-(\ln(y)-\mu')^2/2\sigma'^2) \qquad (-\infty<\ln(y)<\infty) \qquad \text{(式2.3-2)}$$

(2) n⇒∞の時、確率変数 y (y=y(n)) は対数正規分布 Λ(μ', V') に近づく

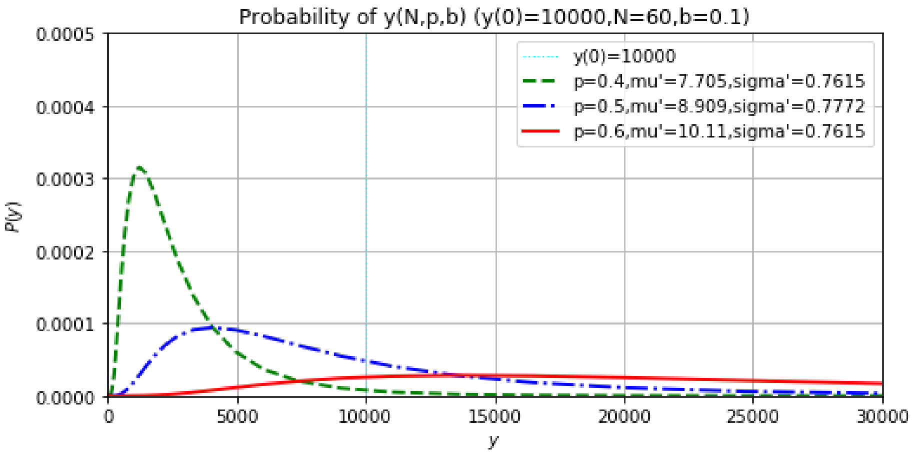
$$f(y) \;=\; 1/(\sqrt{2\pi}\,\sigma' y)\ \exp(-(\ln(y)-\mu')^2/2\sigma'^2) \qquad (0<\ln(y)<\infty) \qquad \text{(式2.3-3)}$$
$$\;=\; 0 \qquad (\ln(y)\leqq 0)$$

確率変数 y(n) が近づく対数正規分布 Λ(μ', V')をグラフ化すると
以下のようになります (y(0)=10000, n=60 で b,p を変えたもの)：

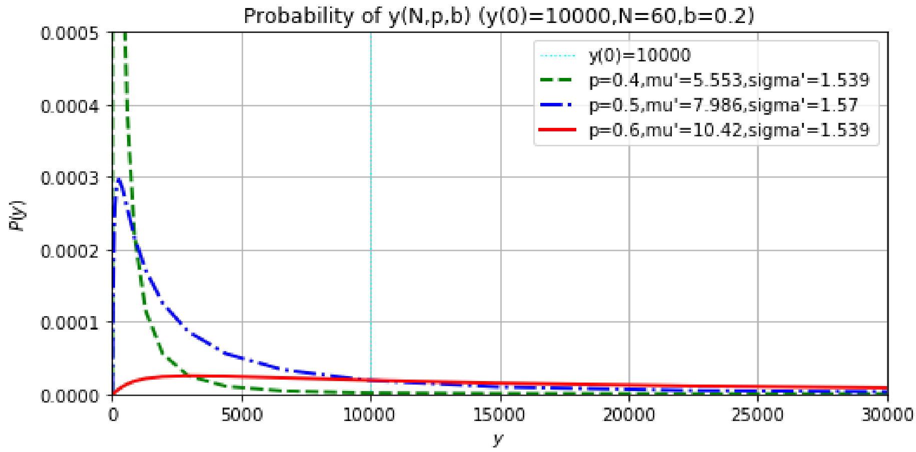
```
Probability of y(N,p,b) (y(0)=10000,N=60,b=0.05)
max(y(n))=186791.85894122993
```



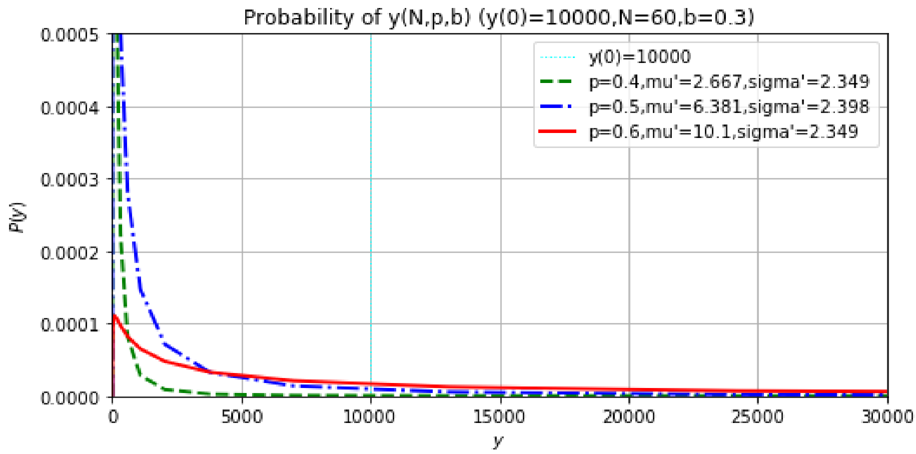

```
Probability of y(N,p,b) (y(0)=10000,N=60,b=0.1)
max(y(n))=3044816.3954141955
```



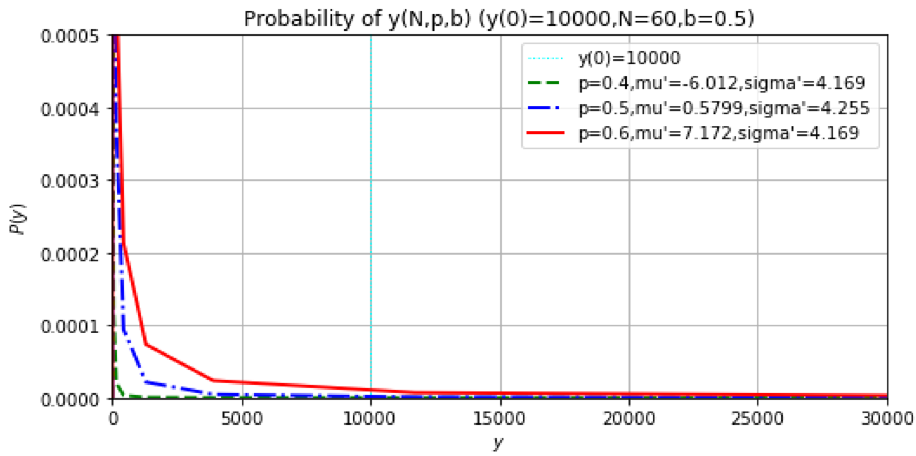
```
Probability of y(N,p,b) (y(0)=10000,N=60,b=0.2)
max(y(n))=563475143.5316666
```




```
Probability of y(N,p,b) (y(0)=10000,N=60,b=0.3)
max(y(n))=68643771727.44704
```




```
Probability of y(N,p,b) (y(0)=10000,N=60,b=0.5)
max(y(n))=367684687169330.2
```



(ﾘｽﾄ：賭け事_ﾘｽﾄ_5_試行回数N、成功割合p、掛け率bの時の最終所持金y(N, p, b))

・「(2.3) n 回賭け事を繰返した結果の所持金 y(n) が従う確率分布」のグラフから
読みとれることは幾つかあります。

(1) 成功確率 p が同じでも、掛け率 b を大きくとるほど、元金が無くなる傾向にある。

(2) 掛け率 b が同じでも、成功確率 p が小さいほど、元金が無くなる傾向にある。

等々・・・・

(3) 乗算過程

・上記の問題設定の(式2.1-1)のように、試行回数 n における確率変数の値 y(n) が、
一つ前の試行回数 n-1 における確率変数の値 y(n-1) により、次の漸化式で与えられるとき、
これを乗算過程と言います：

$$y(n) = \eta(n-1) \times y(n-1),\quad y(0) = y0 \qquad \eta(n)$$ ：(ηはｷﾞﾘｼｱ文字ｲｰﾀ)

この漸化式は、以下のようになります。

$$\begin{aligned} y(n) &= \eta(n-1) \times \eta(n-2) \times \cdots \times \eta(0) \times y0 \\ &= y0 \times \Pi_{k=0}^{n-1} \eta(k) \end{aligned}$$

∴

$$\ln(y(n)) = \ln(y0) + \Sigma_{k=0}^{n-1} \ln(\eta(k))$$

上記の問題設定の(式2.1-1)の場合、

$$\begin{aligned} \eta(k) &= (1 + b) && \text{（賭けに勝った場合）} \\ &= (1 - b) && \text{（賭けに負けた場合）} \end{aligned}$$

であり、試行回数 n のうち勝った回数を X とすると(式2.1-1)が得られます：

$$y(n) = y(0) \times (1 + b)^X \times (1 - b)^{n-X}$$

(4) まとめ

・この問題と解のように、
確率分布を持った変数（上記例では「n 回賭け事を繰返した時の成功回数 X」）と紐づく
変数（上記例では「最終的な所持金 y(n)」）の取り得る値の分布を求める問題は
少なからず有ります。

本例が参考になれば幸いです。

(5) 実装例

(ﾘｽﾄ：賭け事_ﾘｽﾄ_1_所持金の分布関数)

```
#************************************************
# 賭け事_ﾘｽﾄ_1_所持金の分布関数
#************************************************
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 所持金の分布関数
# 0 ≦ y0
# 1 ≦ X  ≦ n
# 0 ≦ b  ≦ 1
def f_GetShojikin(y0, b, n, xlist):
    ylist = []
    for x in xlist:
        y = y0 * (1.0 + b)**x * (1.0 - b)**(n - x)
        ylist.append( y )
    return ylist

# データとグリッド
y0 = 1000   # 初期の所持金
xn = 60     # 賭け回数
xlist = []
ylist0 = []
for ii in range(xn+1):
    xlist.append(ii)
    ylist0.append(y0)

# 所持金の分布
label0 = 'y0={0}'.format(y0)

b1 = 0.05      # 賭け金のその時点の所持金に対する割合
label1 = 'b={0}'.format(b1)
ylist1 = f_GetShojikin(y0, b1, xn, xlist)

b2 = 0.1       # 賭け金のその時点の所持金に対する割合
label2 = 'b={0}'.format(b2)
ylist2 = f_GetShojikin(y0, b2, xn, xlist)

b3 = 0.2       # 賭け金のその時点の所持金に対する割合
label3 = 'b={0}'.format(b3)
ylist3 = f_GetShojikin(y0, b3, xn, xlist)

# グラフ描画
xlabel = '$x (success/{0})$'.format(xn)
ylabel = '$y(x)$'

plt.figure(figsize=(8,4))
plt.plot( xlist, ylist0, 'k:', linewidth=1, label=label0 )
plt.plot( xlist, ylist1, 'r:', linewidth=2, label=label1 )
plt.plot( xlist, ylist2, 'b:', linewidth=2, label=label2 )
plt.plot( xlist, ylist3, 'g-.', linewidth=2, label=label3 )
plt.legend(loc='upper left')
plt.ylim(-0.1, y0*10)
plt.xlim(-0.5, xn)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid(True)
plt.show()
```

(ﾘｽﾄ：賭け事_ﾘｽﾄ_2_所持金の対数の分布関数)

```
#**************************************************
# 賭け事_ﾘｽﾄ_2_所持金の対数の分布関数
#**************************************************
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 所持金の対数の分布関数
# 0 ≦ y0
# 1 ≦ X  ≦ n
# 0 ≦ b  ≦ 1
def f_GetLnShojikin(y0, b, n, xlist):
    alpha = np.log((1.0 + b) / (1.0 - b))
    beta = np.log(y0) + n * np.log(1.0 - b)
    lnylist = []
    for x in xlist:
        lny = alpha * x + beta
        lnylist.append( lny )
    return alpha, beta, lnylist

# データとグリッド
y0 = 1000   # 初期の所持金
xn = 60     # 賭け回数
xlist = []
lnylist0 = []
lny0 = np.log(y0)
for ii in range(xn+1):
    xlist.append(ii)
    lnylist0.append(lny0)

# 所持金の分布
label0 = 'y0={0}'.format(y0)

b1 = 0.05      # 賭け金のその時点の所持金に対する割合
label1 = 'b={0}'.format(b1)
alpha, beta, lnylist1 = f_GetLnShojikin(y0, b1, xn, xlist)
betaMin = beta

b2 = 0.1       # 賭け金のその時点の所持金に対する割合
label2 = 'b={0}'.format(b2)
alpha, beta, lnylist2 = f_GetLnShojikin(y0, b2, xn, xlist)
if betaMin > beta:
    betaMin = beta

b3 = 0.2       # 賭け金のその時点の所持金に対する割合
label3 = 'b={0}'.format(b3)
alpha, beta, lnylist3 = f_GetLnShojikin(y0, b3, xn, xlist)
if betaMin > beta:
    betaMin = beta

# グラフ描画
xlabel = '$x (success/{0})$'.format(xn)
ylabel = '$ln(y(x))$'
plt.figure(figsize=(8,4))
plt.plot( xlist, lnylist0, 'k:', linewidth=1, label=label0 )
plt.plot( xlist, lnylist1, 'r:', linewidth=2, label=label1 )
plt.plot( xlist, lnylist2, 'b:', linewidth=2, label=label2 )
plt.plot( xlist, lnylist3, 'g-.', linewidth=2, label=label3 )
plt.legend(loc='upper left')
plt.ylim(betaMin, 3*lny0)
plt.xlim(-0.5, xn)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid(True)
plt.show()
```

（ﾘｽﾄ：賭け事_ﾘｽﾄ_3_成功回数の二項分布）

```
#************************************************
# 賭け事_ﾘｽﾄ_3_成功回数の二項分布(P=0.3/0.5/0.7)
#************************************************
import numpy as np
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 二項分布の確率質量関数
def fBinominal(n, p, k):
    return fCombinaton(n,k) * ((p)**k) * ((1-p)**(n-k))

# 組合せ nCr
def fCombinaton(n, r):
    return  math.factorial(n) // ¥
           (math.factorial(n - r) * math.factorial(r))

# パラメータに応じた分布計算
xn = 60     # 賭け回数
xlist = range(xn+1)

# p1
p1 = 0.3
label1 = 'p={0}'.format(p1)
plist1 = []
for x in xlist:
    plist1.append( fBinominal(xn, p1, x) )

# p2
p2 = 0.5
label2 = 'p={0}'.format(p2)
plist2 = []
for x in xlist:
    plist2.append( fBinominal(xn, p2, x) )

# p3
p3 = 0.7
label3 = 'p={0}'.format(p3)
plist3 = []
for x in xlist:
    plist3.append( fBinominal(xn, p3, x) )

# グラフ描画
xlabel = '$x (success/{0})$'.format(xn)
ylabel = '$P(x)$'
title = 'Binominal distribution : Bin({0},p)'.format(xn)

plt.figure(figsize=(8,4))
plt.plot( xlist, plist1, 'r:', linewidth=2, label=label1 )
plt.plot( xlist, plist2, 'b:', linewidth=2, label=label2 )
plt.plot( xlist, plist3, 'g-.', linewidth=2, label=label3 )
plt.title(title)
plt.legend(loc='upper left')
plt.ylim(-0.02, 0.15)
plt.xlim(-0.5, xn)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid(True)
plt.show()
```

（ﾘｽﾄ：賭け事_ﾘｽﾄ_4_成功回数の標準化）

```
#************************************************
# 賭け事_ﾘｽﾄ_4_成功回数の標準化
#************************************************
import numpy as np
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 二項分布の確率質量関数
def fBinominal(n, p, k):
    return fCombinaton(n,k) * ((p)**k) * ((1-p)**(n-k))

# 組合せ nCr
def fCombinaton(n, r):
    return  math.factorial(n) // ¥
           (math.factorial(n - r) * math.factorial(r))

# 標準化確率変数 Z の分布
def fStdNormalDist(xlist, xn, p):
    mu = xn * p
    sigma = xn * p * (1.0 - p)
    zlist = []
    plist = []
    for x in xlist:
        z = (x - mu) / sigma
        zlist.append( z )
        plist.append( fBinominal(xn, p, x) )
    return zlist, plist

# パラメータに応じた分布計算
xn = 60     # 賭け回数
xlist = range(xn+1)

# p1
p1 = 0.3
label1 = 'p={0}'.format(p1)
zlist1, plist1 = fStdNormalDist(xlist, xn, p1)

# p2
p2 = 0.5
label2 = 'p={0}'.format(p2)
zlist2, plist2 = fStdNormalDist(xlist, xn, p2)

# p3
p3 = 0.7
label3 = 'p={0}'.format(p3)
zlist3, plist3 = fStdNormalDist(xlist, xn, p3)

# グラフ描画
xlabel = '$z$'.format(xn)
ylabel = '$P(z)$'
title = 'Binominal distribution : Standardised (N={0})'.format(xn)

plt.figure(figsize=(8,4))
plt.plot( zlist1, plist1, 'r:', linewidth=2, label=label1 )
plt.plot( zlist2, plist2, 'b:', linewidth=2, label=label2 )
plt.plot( zlist3, plist3, 'g-.', linewidth=2, label=label3 )
plt.title(title)
plt.legend(loc='upper left')
plt.ylim(-0.05, 0.15)
plt.xlim(-1.0, 1.0)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid(True)
plt.show()
```

(ﾘｽﾄ：賭け事_ﾘｽﾄ_5_試行回数N、成功割合p、掛け率bの時の最終所持金y(N,p,b))

```
#**********************************************************************
# 賭け事_ﾘｽﾄ_5_試行回数N、成功割合p、掛け率bの時の最終所持金y(N,p,b)
#**********************************************************************
import numpy as np
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

#======================================================================
# 対数正規分布関数 (y(0),N,p,b を指定)
#======================================================================
def getLambdaList(xlist, y0, xN, p, b):

    # ln(y)が従う正規分布Ｎ(μ',σ')を算出
    alpha = np.log((1.0 + b) / (1.0 - b))
    beta = np.log(y0) + xN * np.log(1.0 - b)
    mu = xN * p
    sigma = np.sqrt(xN * p * (1.0 - p))
    mudash = beta + alpha * mu
    sigmadash = alpha * sigma

    # yが従う対数正規分布Λ(μ',σ')を算出
    pynList = []
    ynList = []
    ynMax = 0.0
    pi2root= np.sqrt(2 * np.pi)

    for x in xlist:
        yn = y0 * ((1.0 + b)**x) * ((1.0 - b)**(xN- x))
        ynList.append(yn)
        if( ynMax < yn ):
            ynMax = yn

        lnyn = np.log(yn)
        if lnyn <= 0:
            pynList.append(0.0)
        else:
            pynList.append(
                1 / (pi2root * sigmadash * yn)¥
                * np.exp(-(lnyn - mudash)**2 / (2 * sigmadash**2)) )
    return pynList, ynList, ynMax, mudash, sigmadash

#======================================================================
# N回試行時のy(n)の分布関数 (y(0),N,b を指定)
#======================================================================
def showDistribution(xList, y0, xN, b):
    # 初期の所持金のグラフ
    ynList0 = [y0, y0]
    pynList0 = [0.0, 1.0]
    label0 = "y(0)={0}".format(y0)

    # p1
    p1 = 0.4
    pynList1,ynList1,ynMax1,mudash1,sigmadash1 = getLambdaList(xList, y0, xN, p1, b)
    label1 = "p={0},mu'={1:4.4},sigma'={2:4.4}".format(p1, mudash1,sigmadash1)

    # p2
    p2 = 0.5
    pynList2,ynList2,ynMax2,mudash2,sigmadash2 = getLambdaList(xList, y0, xN, p2, b)
    label2 = "p={0},mu'={1:4.4},sigma'={2:4.4}".format(p2, mudash2,sigmadash2)

    # p3
    p3 = 0.6
    pynList3,ynList3,ynMax3,mudash3,sigmadash3 = getLambdaList(xList, y0, xN, p3, b)
    label3 = "p={0},mu'={1:4.4},sigma'={2:4.4}".format(p3, mudash3,sigmadash3)

    # y(n)の最大値とﾊﾟﾗﾒｰﾀを表示
```

```
    ynMax = ynMax1
    if( ynMax < ynMax2 ):
        ynMax = ynMax2
    if( ynMax < ynMax3 ):
        ynMax = ynMax3
    rangeY = y0*3

    # 文言表示
    print( 'Probability of y(N,p,b) (y(0)={0},N={1},b={2:3.3})'.format(y0, xN, b))
    print( "max(y(n))={0}".format(ynMax))

    # グラフ描画
    xlabel = '$y$'
    ylabel = '$P(y)$'
    title = 'Probability of y(N,p,b) (y(0)={0},N={1},b={2:3.3})'.format(y0, xN, b)

    plt.figure(figsize=(8,4))
    plt.plot( ynList0, pynList0, color='cyan', linestyle='dotted', linewidth=1, label=label0 )
    plt.plot( ynList1, pynList1, color='green', linestyle='dashed', linewidth=2, label=label1 )
    plt.plot( ynList2, pynList2, color='blue', linestyle='dashdot', linewidth=2, label=label2 )
    plt.plot( ynList3, pynList3, color='red', linestyle='solid', linewidth=2, label=label3 )

    plt.title(title)
    plt.legend(loc='upper right')
    plt.xlim(-5, rangeY)
    plt.ylim(-0.00, 0.0005)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.show()

#=========================================================================
# パラメータに応じた分布計算
#=========================================================================
y0 = 10000   # 初期の所持金
xN = 60      # 賭け回数
xList = []
for x in range(xN+1):
    xList.append(x)

b = 0.05 # 賭け金の所持金に対する割合
showDistribution(xList, y0, xN, b)

b = 0.1 # 賭け金の所持金に対する割合
showDistribution(xList, y0, xN, b)

b = 0.2 # 賭け金の所持金に対する割合
showDistribution(xList, y0, xN, b)

b = 0.3 # 賭け金の所持金に対する割合
showDistribution(xList, y0, xN, b)

b = 0.5 # 賭け金の所持金に対する割合
showDistribution(xList, y0, xN, b)
```