# Python勉強会@HACHINONE

# 第12章

# 確率、統計とプログラム

# お知らせ

Python勉強会@HACHINOHEでは、ジョン・V・グッターグ『Python言語による
プログラミングイントロダクション』近代科学社、2014年をみんなで勉強しています。

この本は自分で読んで考えて調べると力が付くように書かれています。

自分で読んで考えて調べる前に、このスライドを見るのは、いわばネタバレを
聞かされるようなものでもったいないです。

是非、本を読んでからご覧ください。

# 量子力学の話

Python勉強会@HACHINOHE

- コペンハーゲン解釈
  - 複数の状態の重ね合わせが、観測によりある状態に収束する
- アインシュタインの「隠れた変数理論」

# 確率を用いたプログラム

Python勉強会@HACHINOHE

- 決定的なプログラム
  - 同じデータに対し、決まった結果が出力される
- ある仕様は決定的な出力を求めているのか？　いないか？　どちらでもいいのか？
- 乱数
  - randomモジュールを使用
  - 擬似乱数
  - random.seed(値)で出るパターンを固定できる

```
# -*- coding: utf-8 -*-
import random

def rollDie():
    """1 から 6 までの整数を無作為に選んで出力する"""
    return random.choice([1,2,3,4,5,6])

def rollN(n):
    result = ''
    for i in range(n):
        result = result + str(rollDie())
    print result

rollN(10)
```

# 確率

## Python勉強会@HACHINOHE

- サイコロを連続で振ったとき、それぞれの出る目は「独立」
- サイコロを10回振ったとき
    - パターンは 6**10
    - 各パターンの出る確率は 1 / 6**10
- サイコロを1回振ったとき
    - 1が出る確率 1/6、出ない確率 5/6
- サイコロをn回振ったとき
    - 2回振り、1が出ない確率は (5/6) * (5/6)
    - 10回振り、1が出ない確率は (5/6) ** 10

# 推計統計学

Python勉強会@HACHINOHE

- 母集団から無作為抽出した標本は、母集団の性質を代表する傾向がある
- つまり、標本を調べると母集団が推計できる
- どのくらい標本があれば、どのくらい推計できるか

# コイン投げ

Python勉強会@HACHINOHE

- p.175

```
# -*- coding: utf-8 -*-
import random

def flip(numFlips):
    """numFlipsを整数とする
       コイン投げをnumFlips回実行したときの表の出た割合を返す"""
    heads = 0.0
    for i in range(numFlips):
        if random.random() < 0.5:
            heads += 1
    return heads/numFlips

def flipSim(numFlipsPerTrial, numTrials):
    """numFlipsPerTrial、numTrialsを整数とする
       コイン投げを1セットにつきnumFlipsPerTrial回、
       numTrialsセット実行したときの表の出た割合の平均値を返す"""
    fracHeads = []
    for i in range(numTrials):
        fracHeads.append(flip(numFlipsPerTrial))
    mean = sum(fracHeads)/len(fracHeads)
    return mean
```

```
flipSim(100, 10) # 0.39
flipSim(100, 10) # 0.44

flipSim(100, 100) # 0.5
flipSim(100, 100) # 0.5035

flipSim(100, 100000) # 0.4998146
flipSim(100, 100000) # 0.5001704
```

# コイン投げの考察

Python勉強会@HACHINOHE

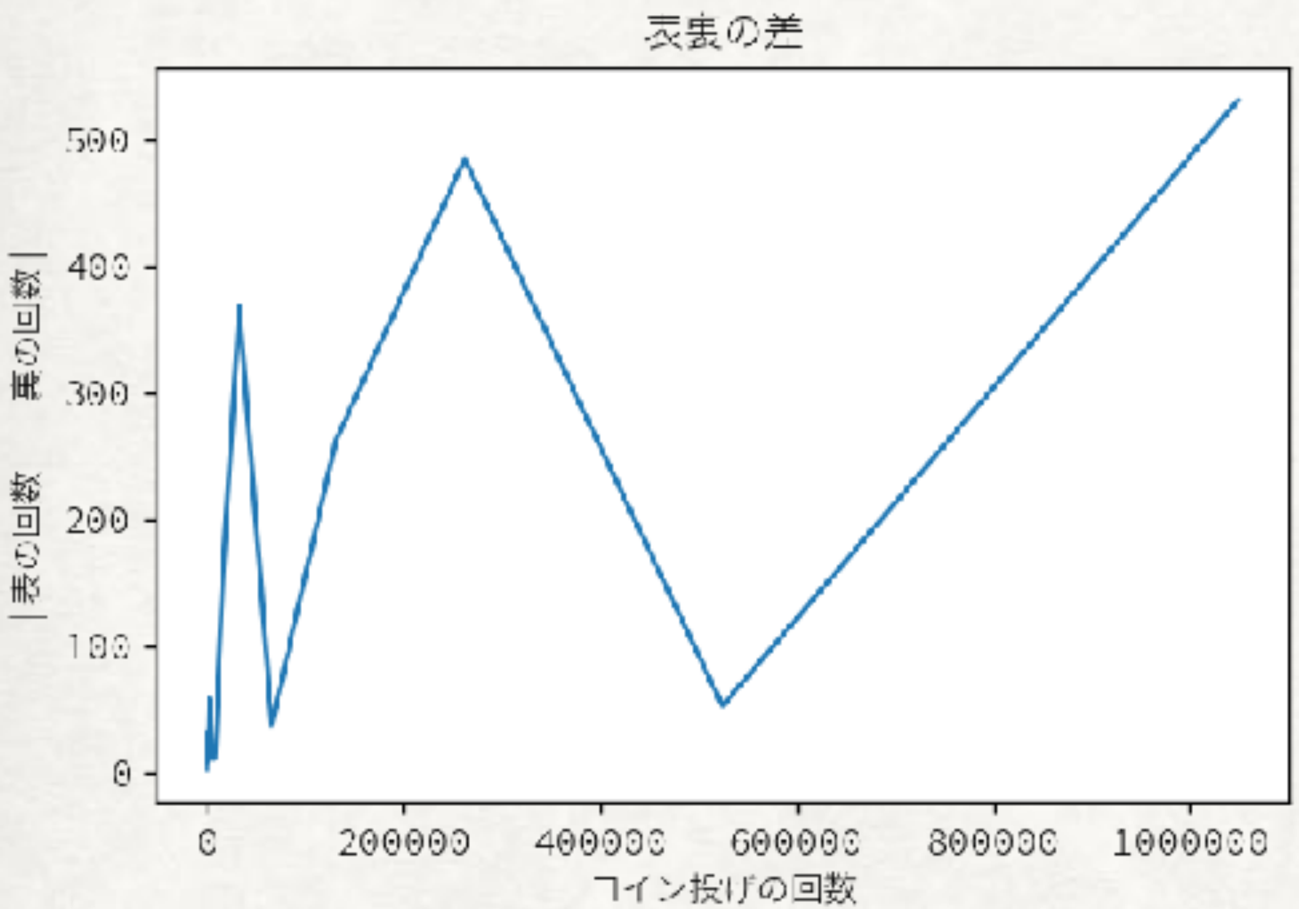
- ベルヌーイの試行
  - コイン投げのような2つの状態を取る各回独立な試行
- 大数の法則
  - 試行回数を増やすと、実際に観測される平均値は、理論的な期待値に近づく
- 賭博者の誤謬
  - 今まで裏がたくさんでているからといって、次は表がでると期待できるわけではない
- 「コインの表と裏の回数の差」/「総回数」は小さくなるが、「コインの表と裏の回数の差」が小さくなるわけではない

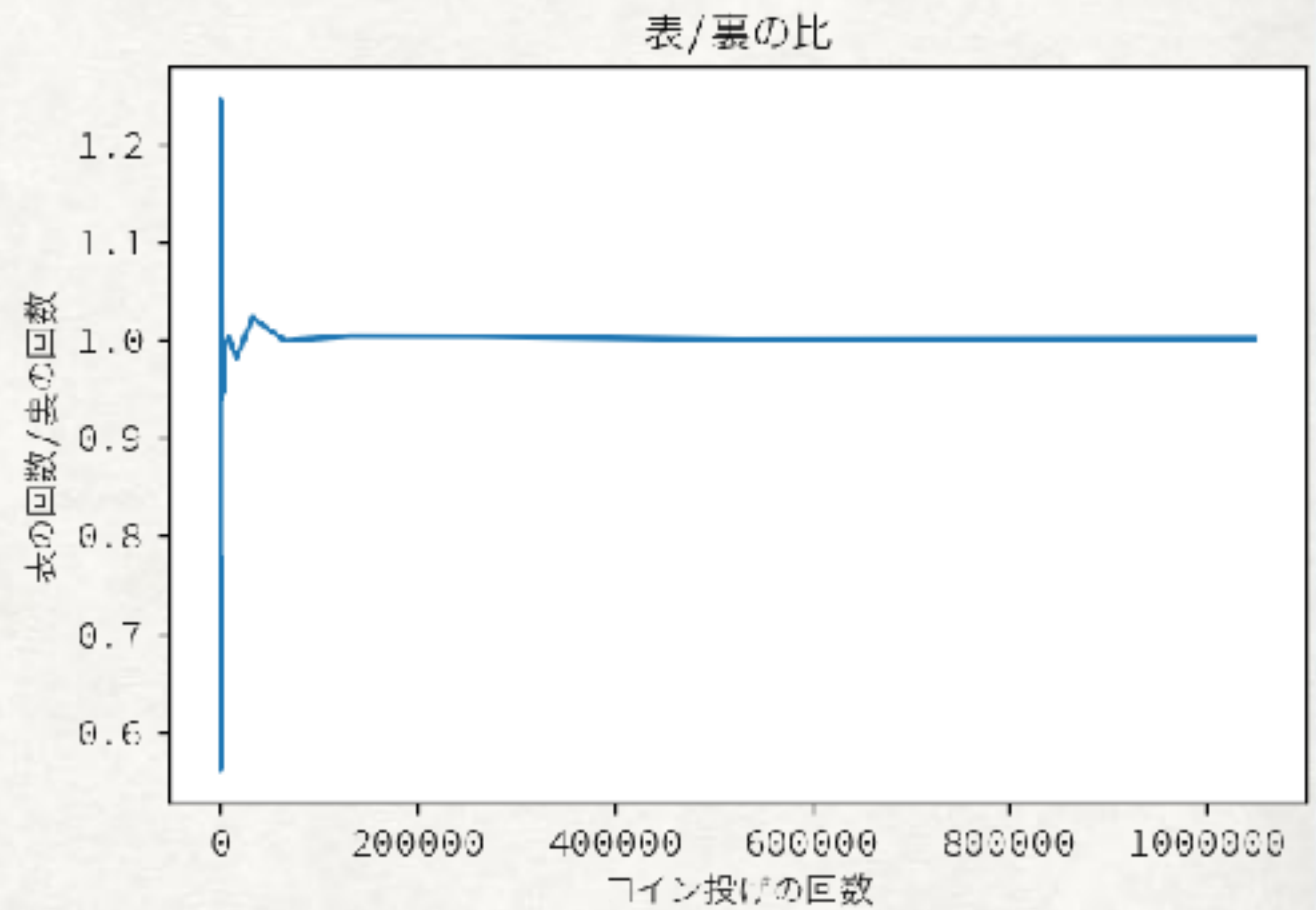
# コイン投げのプロット

Python勉強会@HACHINOHE

- コード12.3

```
# -*- coding: utf-8 -*-
import pylab
import random
pylab.rcParams['font.family'] = 'AppleGothic'

def flipPlot(minExp, maxExp):
    """minExp と maxExp を minExp < maxExp を満たす正の整数とする
       2**minExp から 2**maxExp 回のコイン投げの結果をプロットする"""
    ratios = []
    diffs = []
    xAxis = []
    for exp in range(minExp, maxExp + 1):
        xAxis.append(2**exp)
    for numFlips in xAxis:
        numHeads = 0
        for n in range(numFlips):
            if random.random() < 0.5:
                numHeads += 1
        numTails = numFlips - numHeads
        ratios.append(numHeads/float(numTails))
        diffs.append(abs(numHeads - numTails))
    pylab.title(u'表裏の差')
    pylab.xlabel(u'コイン投げの回数')
    pylab.ylabel(u'|表の回数 - 裏の回数|')
    pylab.plot(xAxis, diffs)
    pylab.figure()
    pylab.title(u'表/裏の比')
    pylab.xlabel(u'コイン投げの回数')
    pylab.ylabel(u'表の回数/裏の回数')
    pylab.plot(xAxis, ratios)

random.seed(0)
flipPlot(4, 20)
pylab.show()
```


表裏の差
|表の回数 裏の回数|
0
100
200
300
400
500
0
200000
400000
600000
800000
1000000
コイン投げの回数


表/裏の比
表の回数/裏の回数
0.6
0.7
0.8
0.9
1.0
1.1
1.2
0
200000
400000
600000
800000
1000000
コイン投げの回数

# コイン投げのプロット

Python勉強会@HACHINOHE

- p.178指練習

```
pylab.loglog()
```

```
pylab.semilogx()
```

# 分散、標準偏差、変動係数

Python勉強会@HACHINOHE

- 分散: 個々の値xの平均値μからのズレの2乗の和を個数|X|で割ったもの

$$V(X) = \frac{1}{|X|}\sum_{x \in X}(x-\mu)^2$$

```
def variance(X):
    """Xを数のリストとする
       Xの分散を返す"""
    mean = sum(X) / len(X)
    tot = 0.0
    for x in X:
        tot += (x - mean) ** 2
    return tot / len(X)
```

- 標準偏差: 分散の平方根

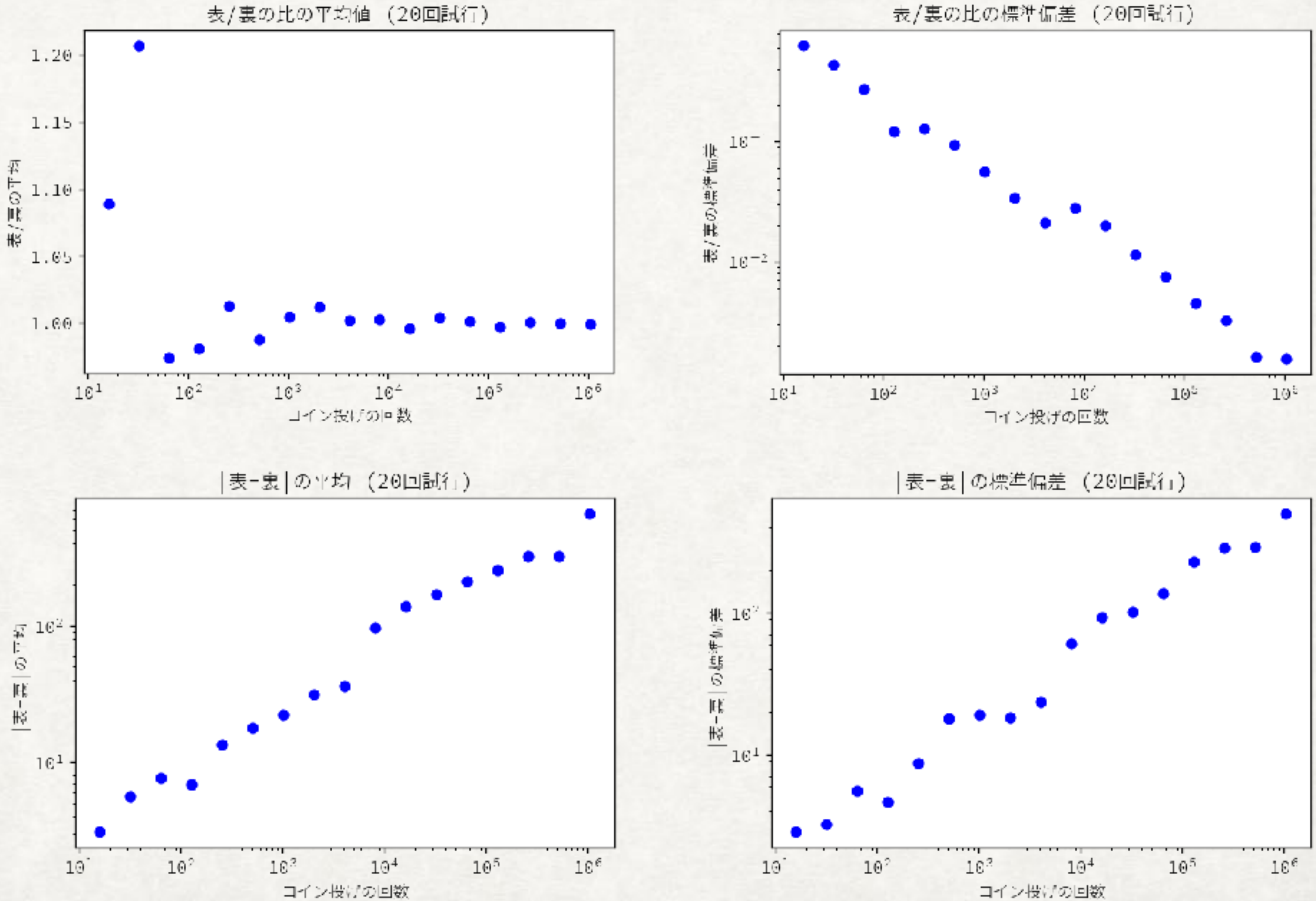
$$\sigma(X) = \sqrt{V(X)} = \sqrt{\frac{1}{|X|}\sum_{x \in X}(x-\mu)^2}$$

```
def stdDev(X):
    """Xを数のリストとする
       Xの標準偏差を返す"""
    return variance(X) ** 0.5
```

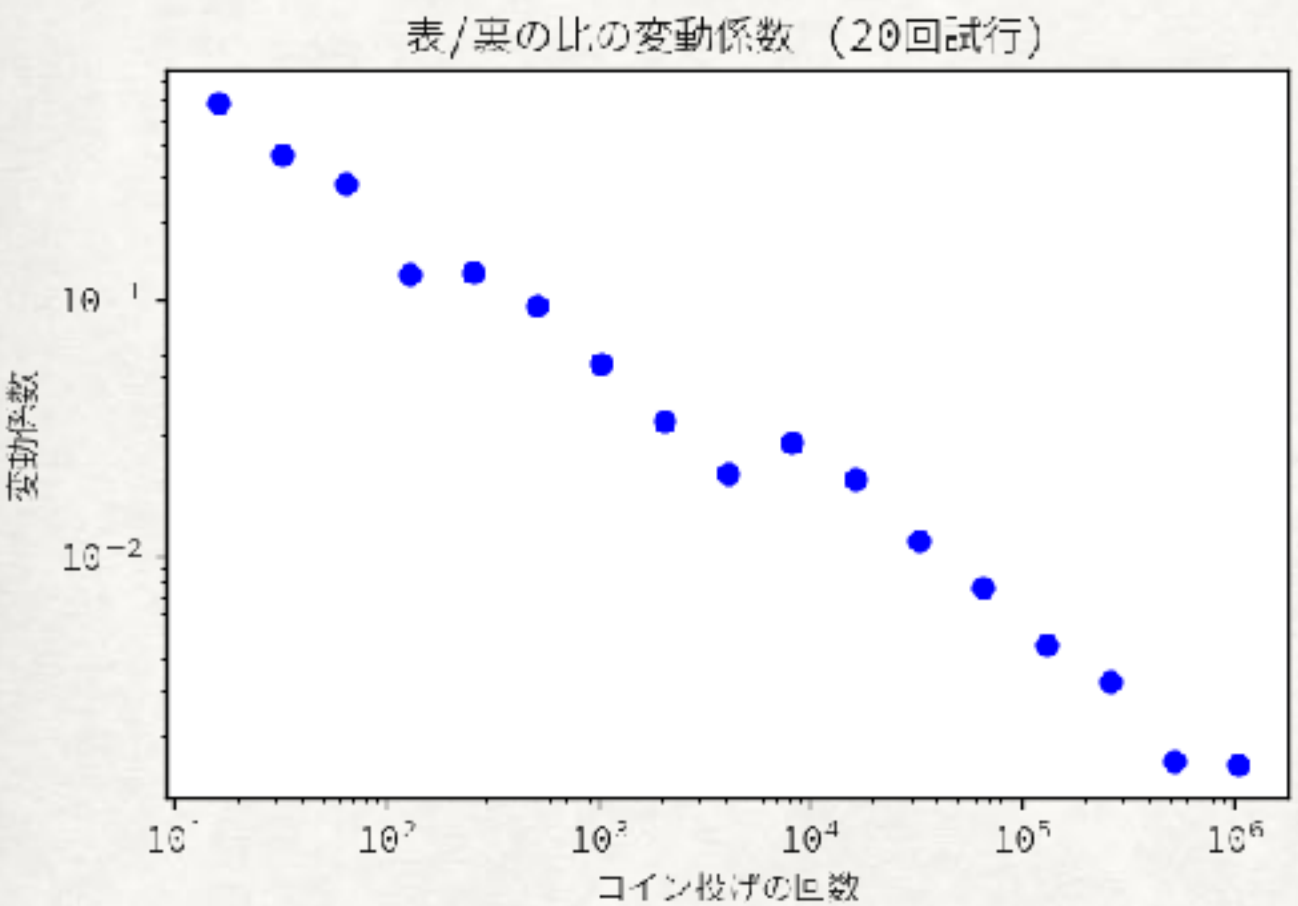
- 変動係数: 標準偏差を平均で割ったもの

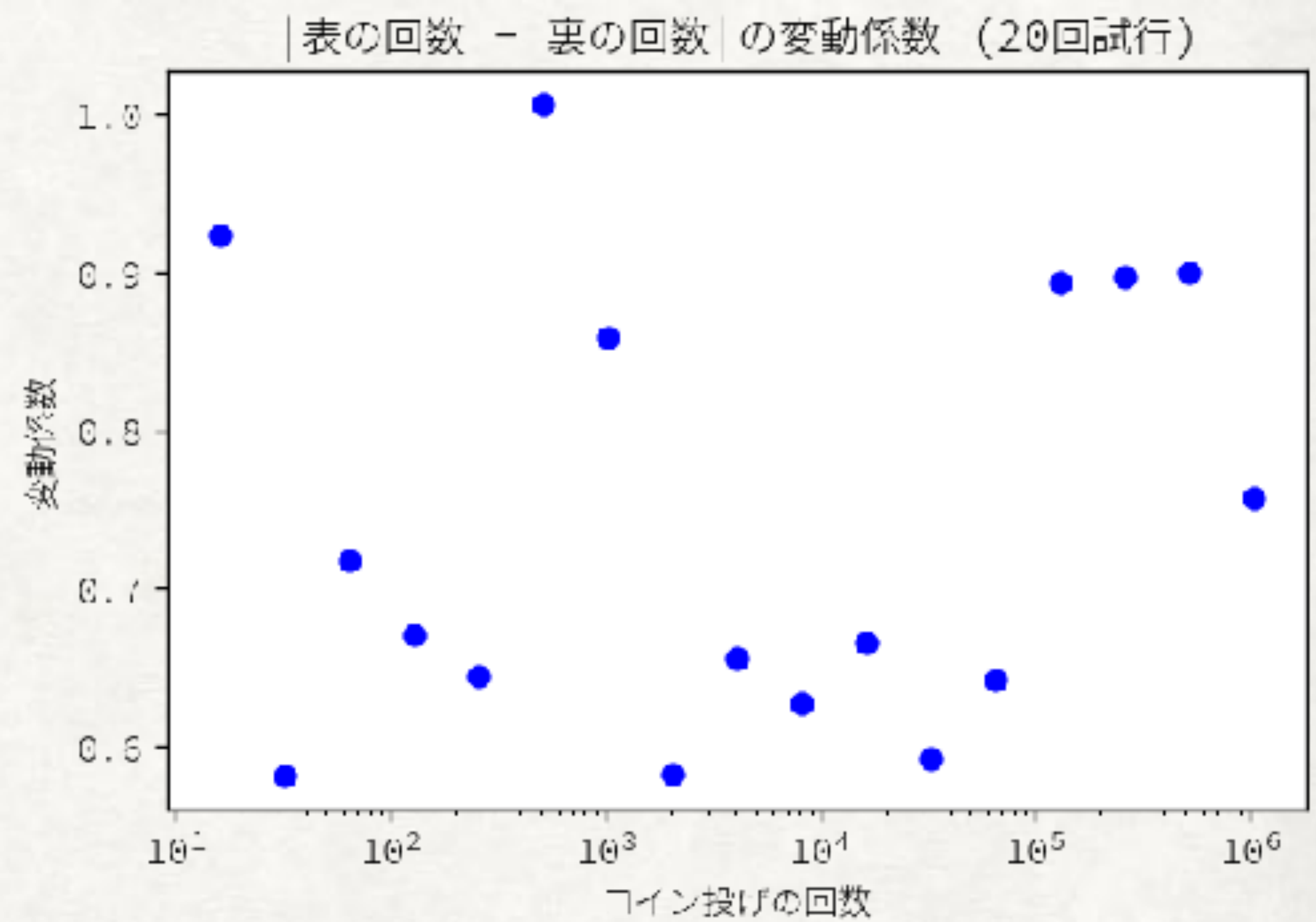
$$C.V.(X) = \frac{\sigma(X)}{\mu}$$

```
def CV(X):
    mean = sum(X)/float(len(X))
    try:
        return stdDev(X)/mean
    except ZeroDivisionError:
        return float('nan')
```



表/裏の比の平均値（20回試行）
表/裏の平均
コイン投げの回数
表/裏の比の標準偏差（20回試行）
表/裏の標準偏差
コイン投げの回数
|表-裏|の平均（20回試行）
|表-裏|の平均
コイン投げの回数
|表-裏|の標準偏差（20回試行）
|表-裏|の標準偏差
コイン投げの回数


表/裏の比の変動係数 (20回試行)
変動係数
コイン投げの回数

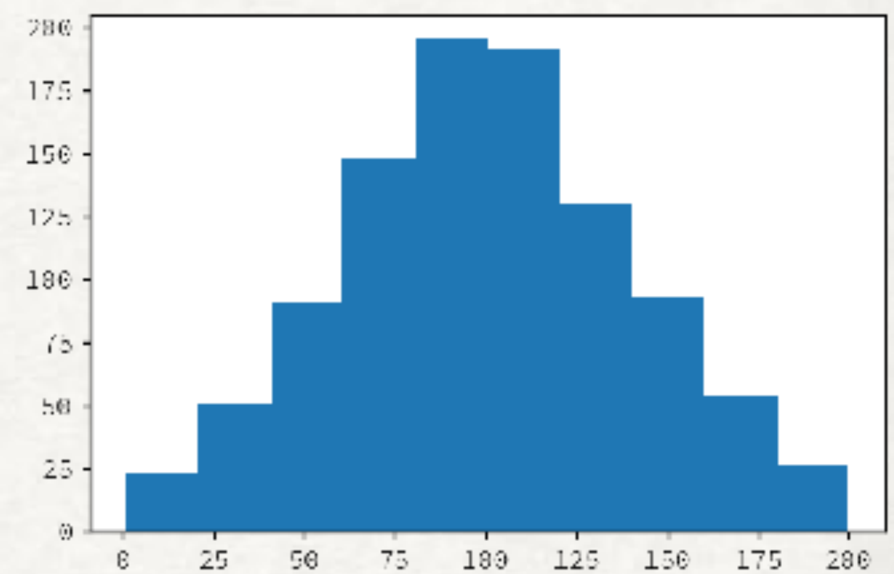

|表の回数 - 裏の回数|の変動係数 (20回試行)
1.0
0.9
0.8
0.7
0.6
変動係数
コイン投げの回数

# ヒストグラム

Python勉強会@HACHINOHE

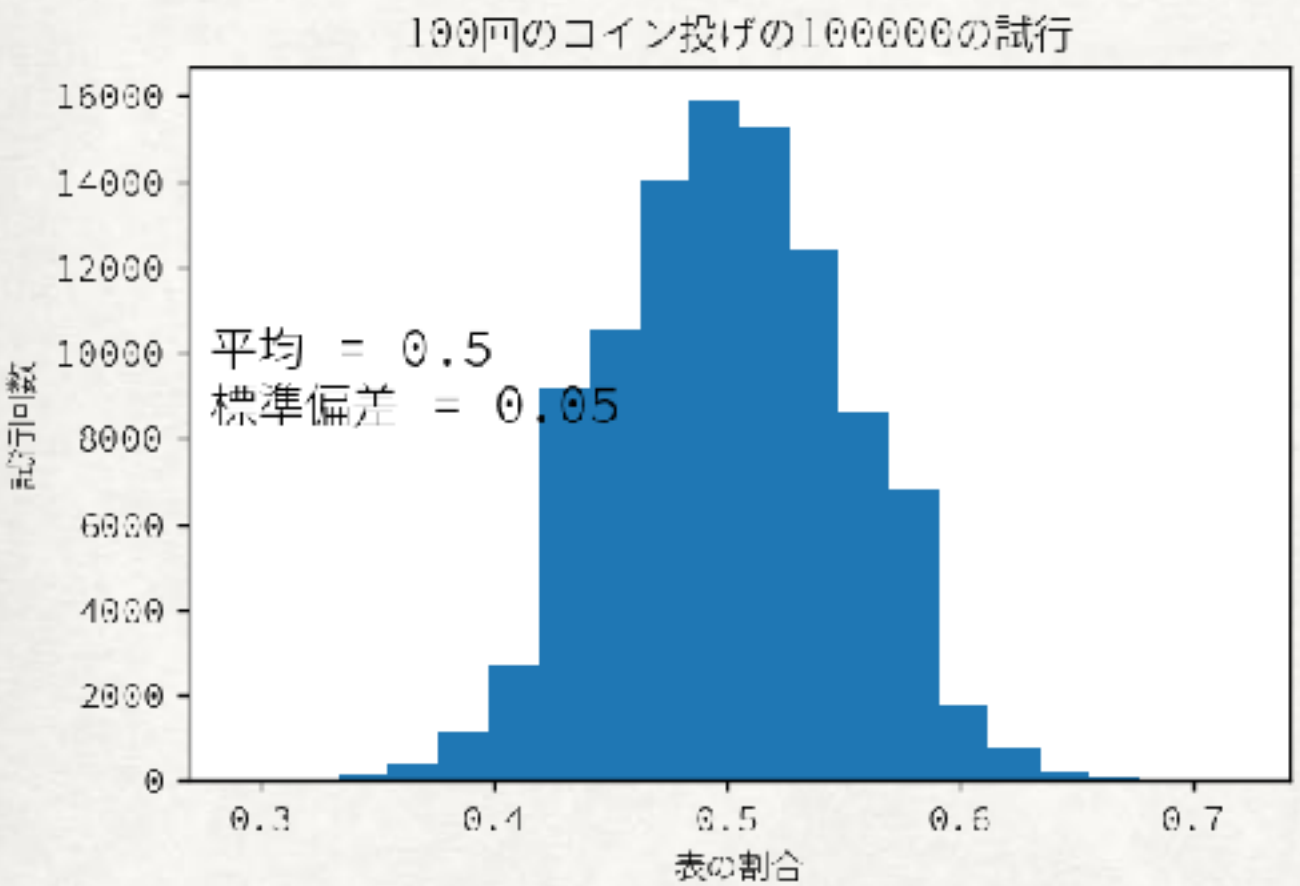
- hist()でヒストグラム生成

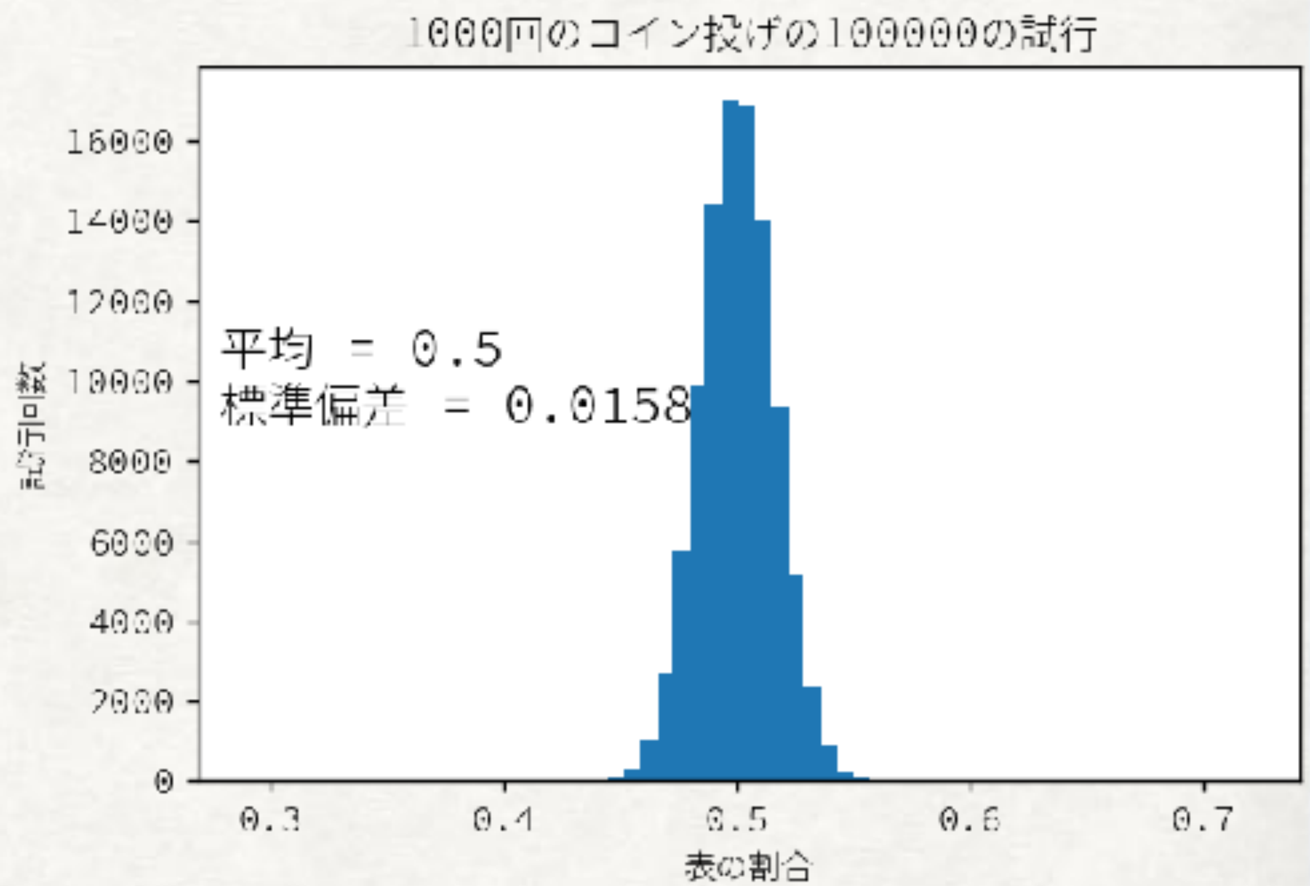
```
# -*- coding: utf-8 -*-
import random
import pylab

vals = [1, 200] # 値が 1 から 200 の範囲にあること仮定
for i in range(1000):
    num1 = random.choice(range(1, 100))
    num2 = random.choice(range(1, 100))
    vals.append(num1+num2)
pylab.hist(vals, bins = 10)

pylab.show()
```



# コイン投げのヒストグラム

Python勉強会@HACHINOHE



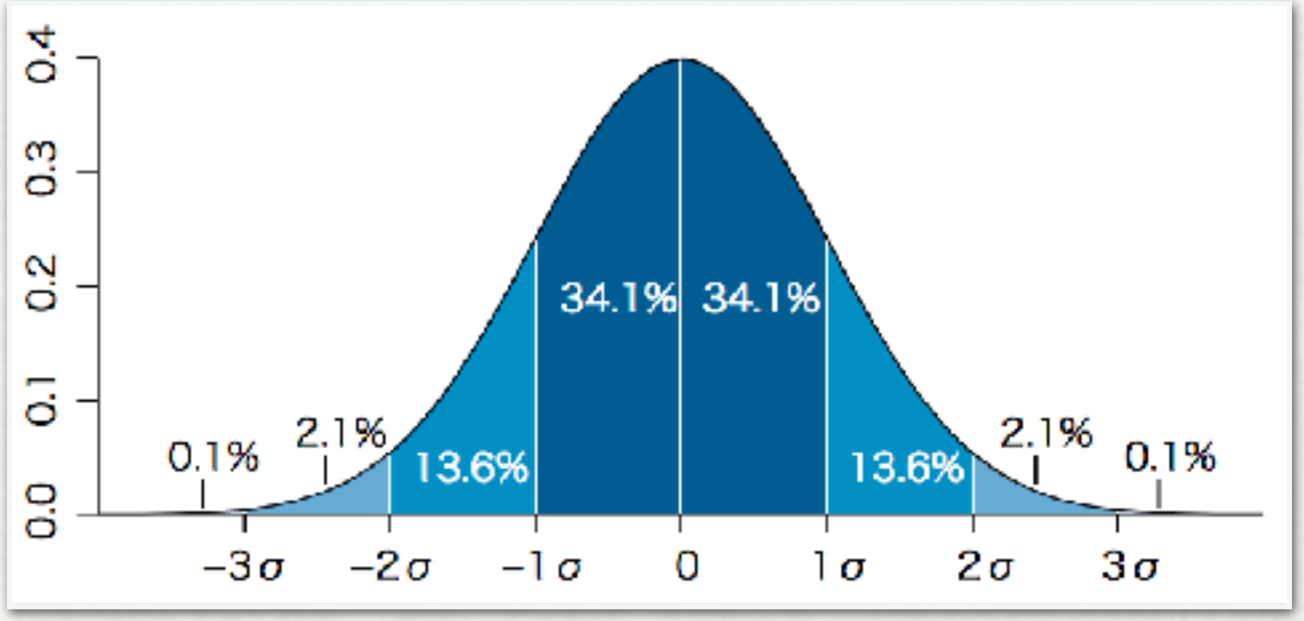
# 正規分布

Python勉強会@HACHINOHE

- コイン投げの回数を増やすと分布は正規分布に近づく

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- 信頼水準
  - 65%のデータが平均からσ
  - 95%のデータが平均から2**σ**
  - 99.7%のデータが平均から3**σ**

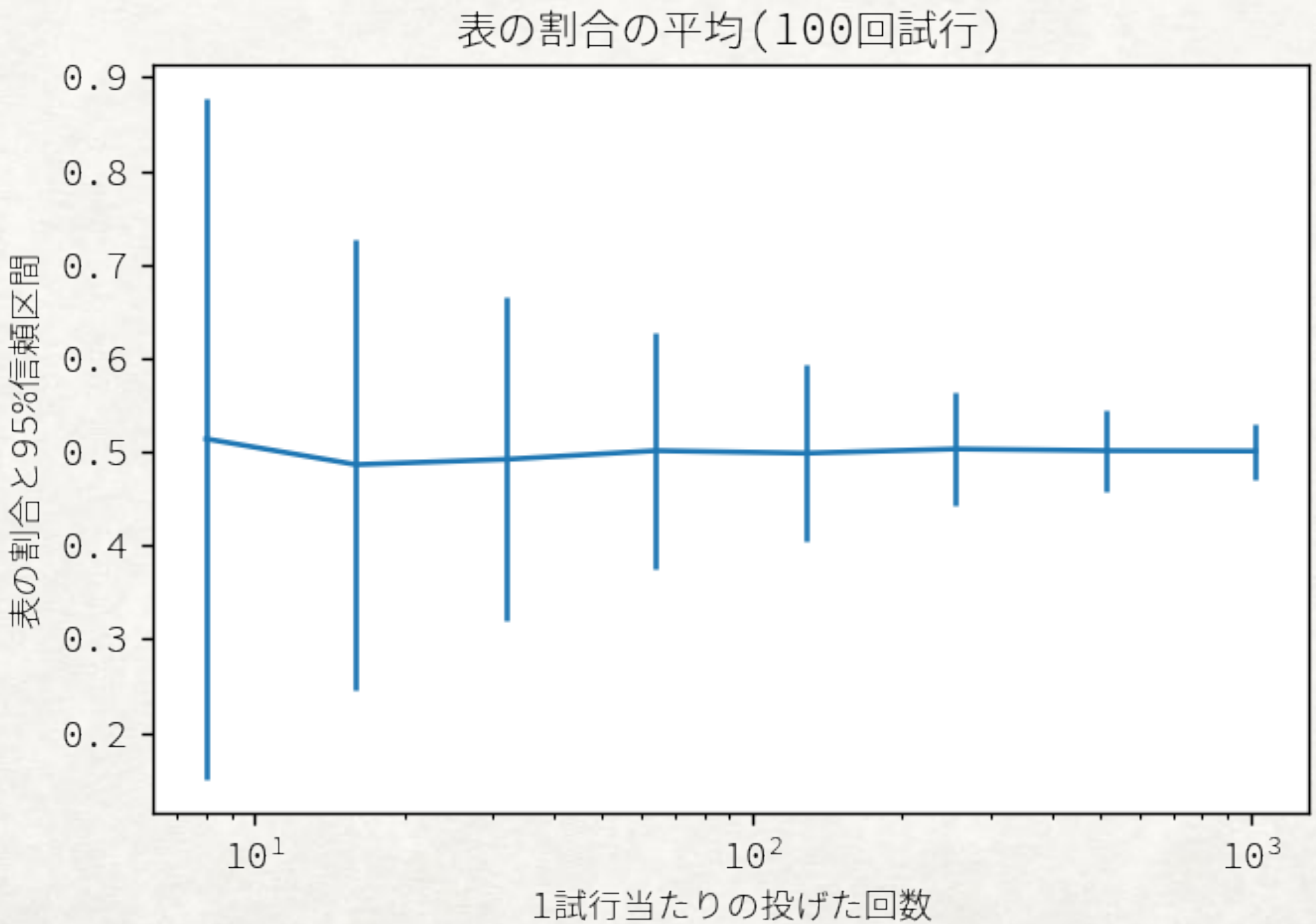



By M. W. Toews - Own work, based (in concept) on figure by Jeremy Kemp, on 2005-02-09, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1903871

# エラーバー

Python勉強会@HACHINOHE

- errorbar()でエラーバー生成



# さまざまな分布



- 正規分布 random.gauss(mu, sigma)
- 一様分布 random.uniform(min, max)
- 指数分布 random.expovariate
- 幾何分布: 指数分布を離散化
- ベンフォード分布: 対数で見た時に一様な分布幅の場合、その下桁の数値の出現確率

# 検定

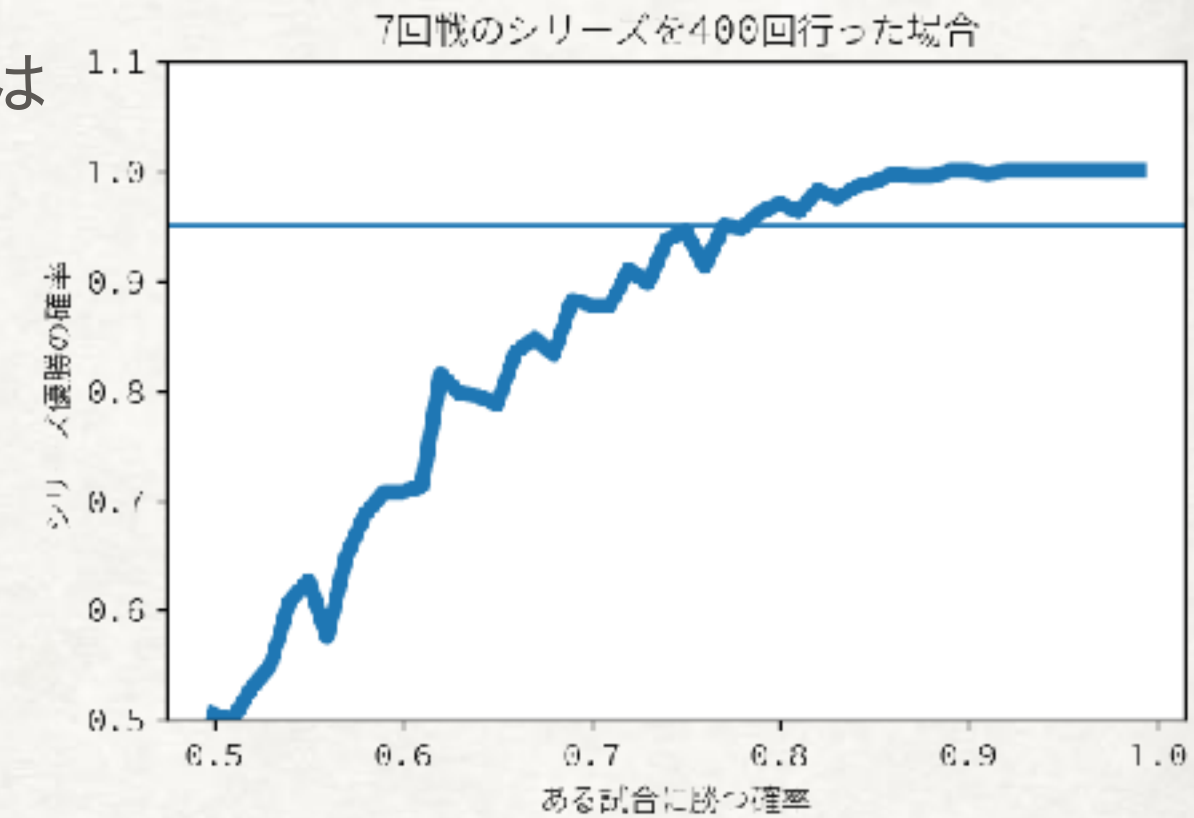
Python勉強会@HACHINOHE

- 例: 日本シリーズは7回戦で十分か？
- 統計的有意なことを検定する
- 帰無仮説
  - もしも両チームが同じ実力だとする
  - 7回戦を無限回実行すると、勝率は半々
- 観測
  - シミュレーション

# ワールドシリーズの例

Python勉強会@HACHINOHE

- 95%の確率でワールドシリーズに勝つには1試合に勝つ確率が75%程度必要

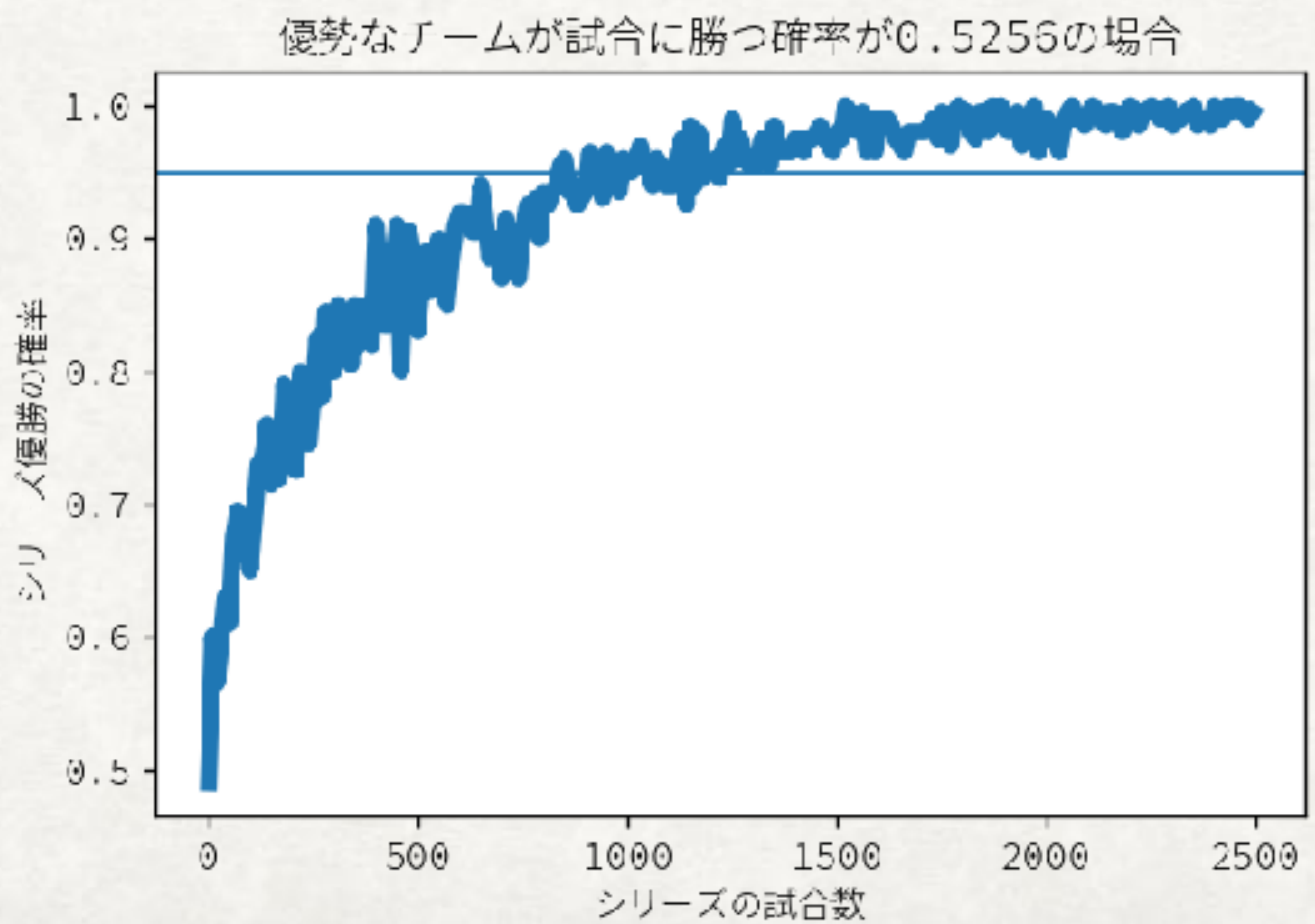



- 勝:負=75:25=3:1
- 例: 2009年のワールドシリーズ
  - レギュラーシーズン
    - ヤンキース63.6%、フィリーズ57.4%
  - すると、1試合当たり、52.5%(63.6/(63.6+57.4))の勝率でヤンキースが勝ちそう
  - ヤンキースのシリーズの優勝確率は60%以下

# ワールドシリーズでヤンキースが優勝するには？

Python勉強会@HACHINOHE

- 例: 2009年のワールドシリーズ
  - 予測勝率から計算すると…
  - 1000シリーズやらないと確実には勝てない



# ハッシュの衝突

Python勉強会@HACHINOHE

- n個の箱に、K個のものをランダムに入れる
- たまたま同じ箱に入る(衝突する)確率は？

$$1-\left(1\cdot\frac{n-1}{n}\cdot\frac{n-2}{n}\cdot\cdots\frac{n-(K-1)}{n}\right)$$

- n個に1個入れるときは100%衝突しない。つまり衝突する確率0
- n個に2個入れるときは(n-1)/nの確率で衝突しない、1/nの確率で衝突
- …

# ハッシュの衝突の期待値とシミュレーション

## Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
import random

def collisionProb(n, k):
    prob = 1.0
    for i in range(1, k):
        prob = prob * ((n - i)/float(n))
    return 1 - prob

def simInsertions(numIndices, numInsertions):
    """numIndices と numInsertions は正の整数
       衝突が起これば 1，そうでなければ 0 を出力する"""
    choices = range(numIndices) # キーの候補となるリスト
    used = []
    for i in range(numInsertions):
        hashVal = random.choice(choices)
        if hashVal in used: # 衝突あり
            return 1
        else:
            used.append(hashVal)
    return 0

def findProb(numIndices, numInsertions, numTrials):
    collisions = 0.0
    for t in range(numTrials):
        collisions += simInsertions(numIndices, numInsertions)
    return collisions/numTrials
```

```
候補数: 1000 、挿入数: 50
衝突の発生する本当の確率 = 0.71226865688
候補数: 1000 、挿入数: 200
衝突の発生する本当の確率 = 0.999999999478
10000 回のシミュレーション結果 = 1.0
```