Python勉強会@HACHINONE

第14章

モンテカルロ・シミュレーション

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。

モンテカルロ・シミュレーション

Python勉強会@HACHINOHE

- ある確率で起こる現象を、乱数を使ってシミュレーションして計算する
- スタニスワフ・ウラムが、物質中の中性子の動きを計算するために考案
- ソリティアの結果を確率を計算するのではなく、乱数を使って試してみてはどうか？という着想

ド・メレの2つのサイコロ

Python勉強会@HACHINOHE

- 騎士ド・メレがパスカルに相談
  - サイコロ1個を1回振って、6が出る確率は1/6
  - 4回振って、1回でも6が出る確率は4/6 ← 賭けると勝てそう
  - サイコロを2個を1回振って、6のゾロ目は1/36
  - 24回振れば、1回でも6のゾロ目が出る確率は24/36
  - なのに負けた
- 更にパスカルは「神の実在を信じることの期待値は？」という話へ

計算する方法

Python勉強会@HACHINOHE

- サイコロ1個を4回振って、1回でも6が出る確率は?
  - 全パターンは6*6*6*6=1296通り
  - そのうち6が含まれるのは、6XXX、X6XX、...、66XX、...、666X
  - これは大変
- そこで逆に考える
  - 6が出ない確率は？ 1回当たり5/6、4回なら(5/6)**4
  - その逆が答えなので、1-(5/6)**4=0.52
- 2個振るときも同様

シミュレーション

Python勉強会@HACHINOHE

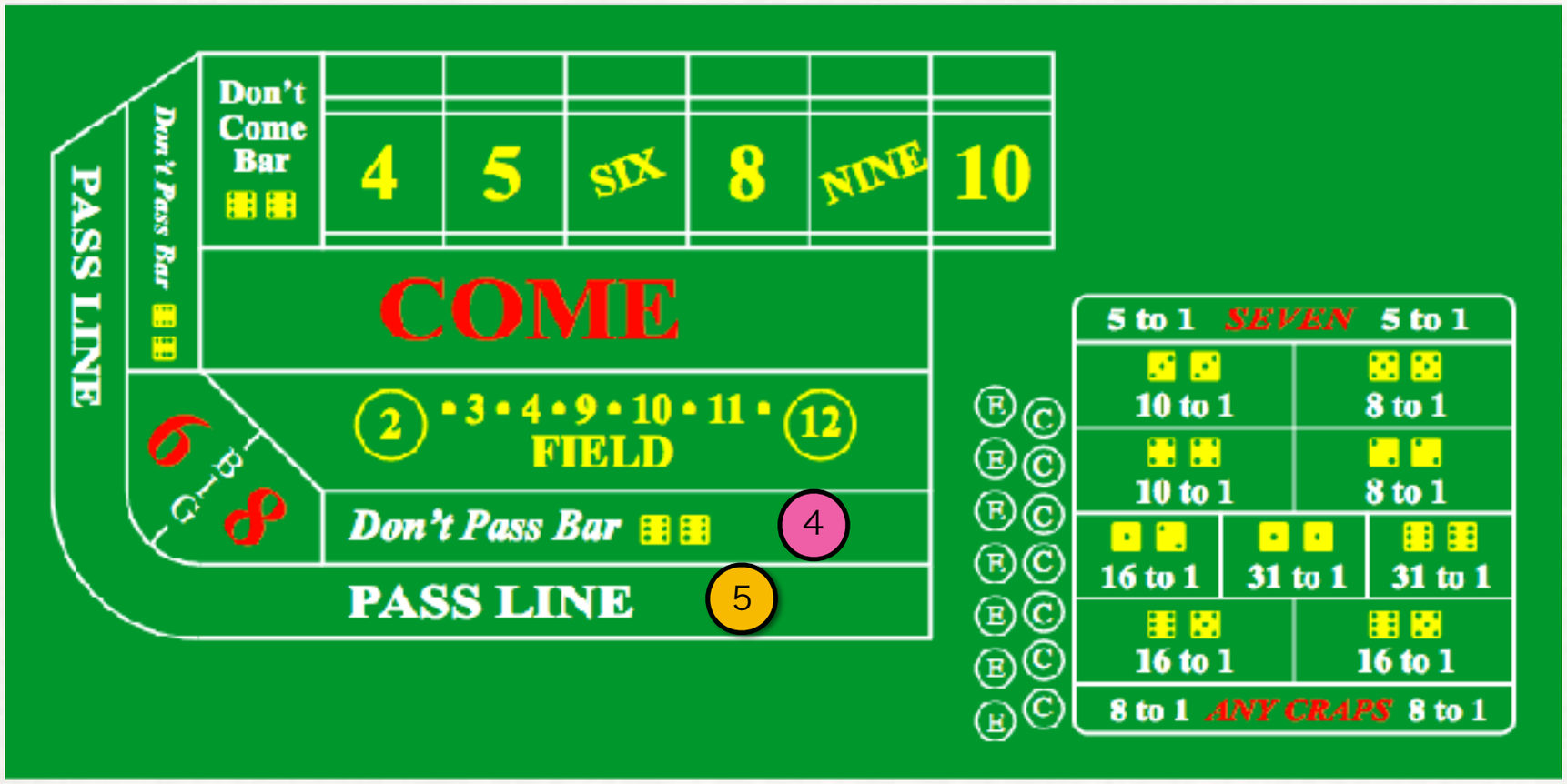
```
# -*- coding: utf-8 -*-
import random

def rollDie():
    return random.choice([1,2,3,4,5,6])

def checkPascal(numTrials):
    """numTrialsは1以上の整数（int）と仮定する
       勝利する確率の評価値を表示する"""
    numWins = 0.0
    for i in range(numTrials):
        for j in range(24):
            d1 = rollDie()
            d2 = rollDie()
            if d1 == 6 and d2 == 6:
                numWins += 1
                break
    print '勝利する確率 =', numWins/numTrials

checkPascal(1000000)
```

勝利する確率 = 0.491265

real　0m20.077s
user　0m19.991s
sys 0m0.054s

クラップス:サイコロ2個投げる

Python勉強会@HACHINOHE



合計	場合の数
2	1
3	2
4	3
5	4
6	5
7	6
8	5
9	4
10	3
11	2
12	1

シューターの勝利条件

1回目に7か11。2回目以降は1回目と同じ目

シューターの敗北条件

1回目に2、3、12。2回目以降は7

パス・ライン

シューターの勝利に賭ける

ドント・パス

シューターの敗北に賭ける。

ただし、1回目に12が出ると引き分け

賭けと儲け

Python勉強会@HACHINOHE

- 賭け
  - パス・ライン
    - シューターの勝利に賭ける。
  - ドント・パス
    - シューターの敗北に賭ける。
    - ただし、1回目に12が出ると引き分け
- 儲け
  - 勝てば賭けただけ増える: 1ドル賭ける→2ドル戻る
  - 負けると賭け金が没収: 1ドル賭ける→0ドル戻る
  - 引き分けは賭け金が戻る: 1ドル賭ける→1ドル戻る

投資収益率

Python勉強会@HACHINOHE

- 投資収益率 Return on Investment(ROI)

$$\mathrm{ROI} = \frac{\text{儲け}}{\text{投資金額}} = \frac{\text{最後に残った金額 - 投資金額}}{\text{投資金額}}$$

$$= \frac{(2*\text{勝} + 1*\text{引分} + 0*\text{負}) - (\text{勝} + \text{引分} + \text{負})}{\text{勝} + \text{負} + \text{引分}}$$

$$= \frac{\text{勝 - 負}}{\text{勝} + \text{負} + \text{引分}}$$

プログラム

Python勉強会@HACHINOHE

- crapsSim(1ゲームの回数, ゲーム回数)
  - 指定した1ゲームの回数(手)xゲーム回数の勝負を行う
  - パスとドント・パスに賭けたときのROIを集計して表示

クラップス
パスに賭けて勝った回数 パスに賭けて負けた回数 ドント・パスに賭けて勝った回数 ドント・パスに賭けて負けた回数 ドント・パスに賭けて引き分けた回数
勝負() パスに賭けた結果() ドント・パスに賭けた結果()

シミュレーション結果:1

Python勉強会@HACHINOHE

- 20回勝負を10セット
  - 標準偏差が大きい→95%信頼区間が広くてどちらとも言えない

```
>>> crapsSim(20,10)
パス: ROIの平均値 = -7.0%、標準偏差 = 23.6854%
→-54.3708～40.3708
ドント・パス: ROIの平均値 = 4.0%、標準偏差 = 23.5372%
→-43.0744～51.0744
```

- 1千万回勝負を10セット
  - 95%信頼区間がかぶっている

```
>>> crapsSim(10000000,10)
※7分くらい
パス: ROIの平均値 = -1.4216%、標準偏差 = 0.0322%
→ -1.4860～-1.3572
ドント・パス: ROIの平均値 = -1.3579%、標準偏差 = 0.0334%
→ -1.4247～-1.2911
```

シミュレーション結果:2

Python勉強会@HACHINOHE

- 20回勝負を1千万回セット
  - 平均は確からしい値になるが、不確実性が高い

```
>>> crapsSim(20,10000000)
パス: ROIの平均値 = -1.4133%、標準偏差 = 22.3571%
→-54.3708～40.3708
ドント・パス: ROIの平均値 = -1.3649%、標準偏差 = 22.0446%
→-43.0744～51.0744
```

- 勝負回数を莫大な回数にしないと(しても)、結果は不確実
  - ギャンブルとしてあり

歪んだサイコロ

Python勉強会@HACHINOHE

- 目の出方に偏りがある

```
def rollDie():
    return random.choice([1,1,2,3,3,4,4,5,5,5,6,6])
```

- パスの方がいい

```
>>> crapsSim(20,10000000)
パス: ROIの平均値 = 6.7066%、標準偏差 = 0.0208%
ドント・パス: ROIの平均値 = -9.4824%、標準偏差 = 0.02%
```

参照表による高速化

Python勉強会@HACHINOHE

- 2回目以降の勝負の計算が1回で済む: 7分半が30秒くらい

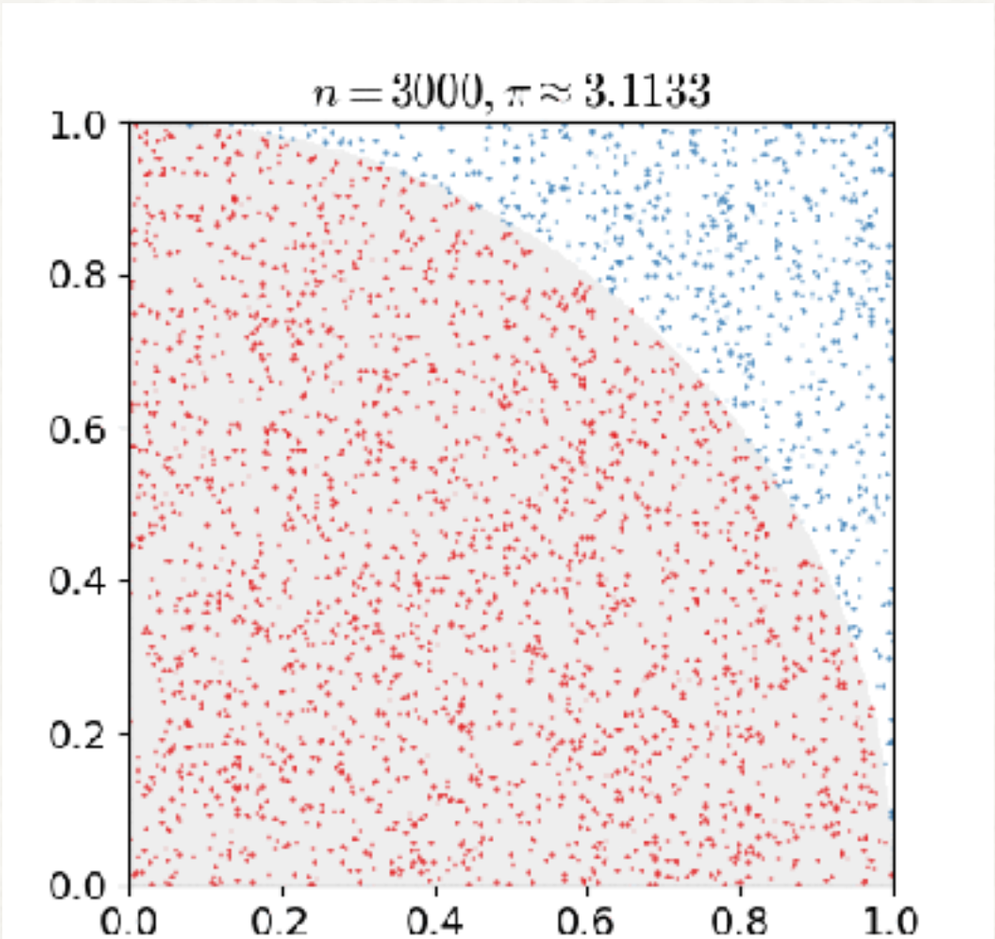
```
def playHand(self):
    #playHandの，より高速な，もう1つの実装
    pointsDict = {4:1/3.0, 5:2/5.0, 6:5/11.0, 8:5/11.0,
                  9:2/5.0, 10:1/3.0}
    throw = rollDie() + rollDie()
    if throw == 7 or throw == 11:
        self.passWins += 1
        self.dpLosses += 1
    elif throw == 2 or throw == 3 or throw == 12:
        self.passLosses += 1
        if throw == 12:
            self.dpPushes += 1
        else:
            self.dpWins += 1
    else:
        if random.random() <= pointsDict[throw]:
            self.passWins += 1
            self.dpLosses += 1
        else:
            self.passLosses += 1
            self.dpWins += 1
```

```
point = throw
while True:
    throw = rollDie() + rollDie()
    if throw == point:
        self.passWins += 1
        self.dpLosses += 1
        break
    elif throw == 7:
        self.passLosses += 1
        self.dpWins += 1
        break
```

円周率の計算

Python勉強会@HACHINOHE

- 乱数で座標を作り、円の内側か外側かで面積を計算し、円周率を求める



By nicoguaro - 投稿者自身による作品, CC BY 3.0
https://commons.wikimedia.org/w/index.php?curid=14609430

プログラム

Python勉強会@HACHINOHE

- throwNeedle(本数)
  - 針を投げて、円の内側に落ちた本数を返す
- getEst(本数, 試行回数)
  - 針を指定した本数投げるのを、試行回数セット実施し、円周率と標準偏差を集計し、表示
- estPi(精度, 試行回数)
  - 精度に達するまで針の本数を倍に増やしていく

プログラムと実行結果

Python勉強会@HACHINOHE

```
def estPi(precision, numTrials):
    numNeedles = 1000
    sDev = precision
    while sDev >= precision/2.0:
        curEst, sDev = getEst(numNeedles, numTrials)
        numNeedles *= 2
    return curEst
```

```
>>> estPi(0.01,100)
評価値 = 1.56984、標準偏差 = 0.02564、針の数 = 1000
評価値 = 1.56853、標準偏差 = 0.0196、針の数 = 2000
評価値 = 1.57157、標準偏差 = 0.01163、針の数 = 4000
評価値 = 1.57063、標準偏差 = 0.00913、針の数 = 8000
評価値 = 1.5705、標準偏差 = 0.00669、針の数 = 16000
評価値 = 1.57104、標準偏差 = 0.00421、針の数 = 32000
```