# MAC0110 — Segundo EP - Data da Entrega: 14/4/2016

Roberto Hirata Jr.

10 de abril de 2016

## 1 Introdução

O segundo EP deste semestre será implementar, em Python, as três soluções clássicas do interessantíssimo paradoxo das probabilidades de Joseph Bertrand [Tis84,Por10].

**O problema das cordas**: Qual a probabilidade de, ao escolhermos uma corda de uma circunferência, ela ser maior que o lado do triângulo equilátero inscrito a ela?

**Especificação matemática da primeira solução:** Dados dois pontos $A$ e $B$ aleatórios em uma circunferência, qual a probabilidade da corda que passa por $A$ e por $B$ seja maior que o lado do triângulo equilátero inscrito à circunferência?

**Solução**: Gire o triângulo até que um dos seus vértices coincida com o ponto $A$. A probabilidade do ponto $B$ ser tal que a corda $AB$ seja maior que o lado do triângulo é exatamente $\frac{1}{3}$ (examine a Fig. 1 e tente entender porquê).

**Especificação matemática da segunda solução:** Seja $O$ o centro da circunferência e $OC$ o raio resultante da escolha arbitrária de um ponto $C$ na circunferência. Tome um ponto qualquer, $P$, em $OC$. Trace uma perpendicular a $OC$ que passe por $P$ e que toque a circunferência nos pontos $A$ e $B$. A corda $AB$ é maior que o lado do triângulo equilátero se $P$ for escolhido entre $O$ e a metade do raio da circunferência (pois o lado do triângulo equilátero corta o raio na metade). Assim, a probabilidade de escolhermos uma corda maior que o lado do triângulo é exatamente $\frac{1}{2}$. (Fig. 2).

**Especificação matemática da terceira solução:** Seja $P$ um ponto qualquer no interior da circunferência de centro $O$. Seja $AB$ uma corda da circunferência cujo ponto médio é $P$.

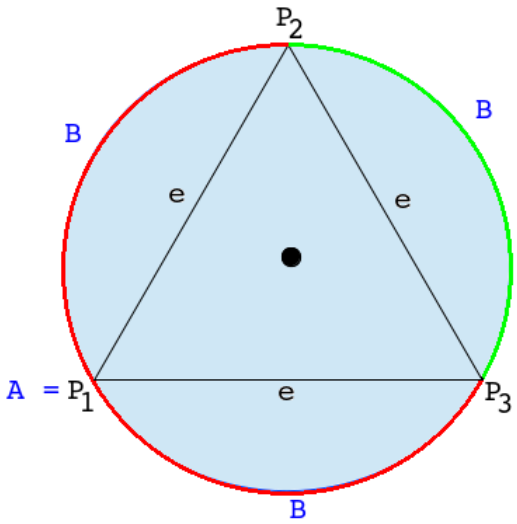


Figura 1: Ilustração da primeira solução.

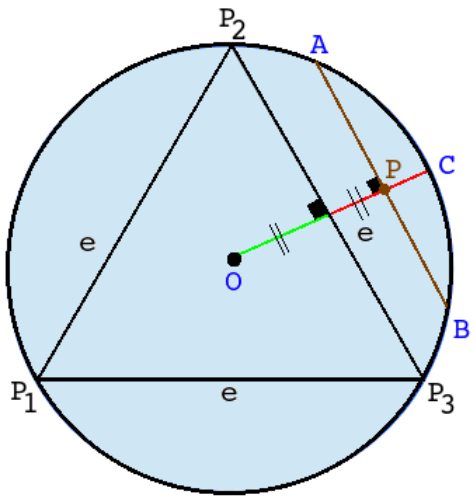


Figura 2: Ilustração da segunda solução.

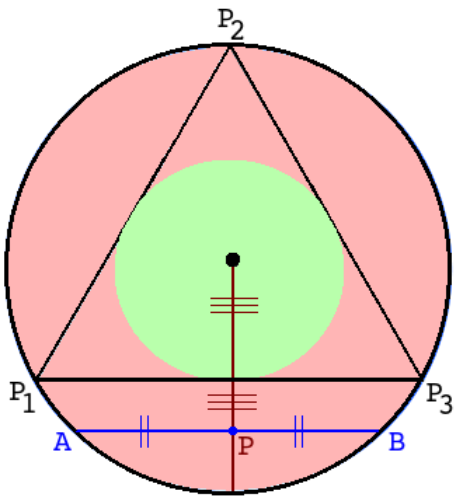


Figura 3: Ilustração da terceira solução.

Se $P$ cair dentro da circunferência de centro $O$ e raio igual a metade do raio da circunferência original, o arco $AB$ será maior, ou igual, ao lado do triângulo equilátero inscrito à circunferência. Assim, a probabilidade da corda $AB$ ser maior que o lado do triângulo equilátero inscrito à circunferência é igual à razão da área da circunferência menor, cujo raio é metade do raio original, e a circunferência maior, ou seja, $\frac{1}{4}$. (Fig. 3).

Há várias questões interessantes a respeito desse paradoxo. Uma delas é histórica. Ela refere-se ao princípio da indiferença, ou princípio da razão insuficiente ("Principle of non-sufficient reason"), como alguns preferem chamar [Key21], que pode ser colocado mais, ou menos, da seguinte maneira: "se não há conhecimento, ou razão, para prever o resultado de um experimento dentre um conjunto possível de alternativas, então qualquer alternativa do conjunto é igualmente provável de acontecer". Isso se aplica, como bem sabemos, a um dado não-viciado, a uma moeda não-viciada, e etc. Daí a importância do paradoxo.

Do ponto de vista de simulações físicas, há também grande interesse no problema. Por exemplo, Jaynes [Jay73] refuta a ideia que o problema é um paradoxo e o reformula como um problema bem-posto. **Um problema é dito bem-posto quando existe uma solução para ele e ela é única.** Além disso, a solução tem que ser estável, isto é, se perturbarmos ligeiramente as condições iniciai, a solução também muda pouco.

O EP a ser entregue consiste na codificação, em Python, de três funções que implementam cada uma das soluções, e de um programa principal que chama cada uma das funções e imprime o valor da probabilidade calculado para cada uma das funções.

# 2 Especificação das funções

O EP consiste em simular as três soluções usando três funções e um programa principal que irá chamar as funções e imprimir os resultados. As funções devem ter **obrigatoriamente** os nomes e os parâmetros abaixo especificados. Assuma que o centro da circunferência é a origem do plano, que o raio é dado por um parâmetro $r$ e que $n$ é o número de vezes que será executado o método.

**Nota bene:** É importante deixar claro que você está livre para organizar o código de outras maneiras, isto é, pode criar outras funções que achar convenientes. Porém, o requisito mínimo é que as funções especificadas abaixo sejam implementadas.

- Método 1 - `def metodo1(r,n)` - Calcule o lado do triângulo equilátero inscrito na circunferência de raio $r$, digamos $L$. A sua função deve executar $n$ vezes os passos: (i) escolha aleatoriamente dois pontos da circunferência de raio $r$, digamos $A$ e $B$; (ii) verifique se o arco $AB$ é maior que $L$ e conte caso verdadeiro. Executados os $n$ passos, retorne a razão entre o número de vezes que o arco $AB$ foi maior que $L$ e $n$.

- Método 2 - `def metodo2(r,n)` - A sua função deve executar $n$ vezes os passos: (i) escolha aleatoriamente um ponto da circunferência de raio $r$, digamos $C$; (ii) escolha aleatoriamente um ponto do raio $OC$, digamos $P$; (iii) verifique se $OP < \frac{r}{2}$ e conte caso verdadeiro. Executados os $n$ passos, retorne a razão entre o número de vezes que $OP$ foi menor que $\frac{r}{2}$ e $n$.

- Método 3 - `def metodo3(r,n)` - A sua função deve executar $n$ vezes os passos: (i) escolha aleatoriamente um ponto dentro da circunferência de raio $r$, digamos $P$; (ii) verifique se $OP < \frac{r}{2}$ e conte caso verdadeiro. Executados os $n$ passos, retorne a razão entre o número de vezes que $OP$ foi menor que $\frac{r}{2}$ e $n$.

Escreva uma função principal (`main`) que leia os parâmetros $r$ e $n$ do teclado, chame cada uma das funções e imprima cada um dos valores retornados. Experimente os seu programa com vários valores de $r$ e $n$.

Para escolher aleatóriamente um valor, ou mais que um, use a função `random` da biblioteca `random`. Veja o help da função `random` para saber como melhor usá-la. Por exemplo,

```
from random import random

...

def main():

    n = 50
    i = 0
    # imprime 50 valores aleatórios entre 0 e 1
```

```
    while i < 50:
       aleatorio = random()
       print(aleatorio)
       i = i + 1
....
```

**Nota bene:** Na verdade, o processo de escolher pontos aleatórios no espaço precisa de um pouco mais cuidado pois a função `random` não produz números verdadeiramente aleatórios, mas pseudo-aleatórios. Em verdade, a função gera números através de uma fórmula iterativa [Knu98]. Embora eu vá falar mais tarde sobre essa função, não é o escopo da disciplina investigá-la profundamente. O importante é você saber que existe uma correlação entre dois pontos consecutivos e, por isso, a distribuição dos pontos no espaço não será uniforme. Para os mais curiosos, o livro do Donald Knuth, Seminumerical Algorithms, é uma ótima referência. Para efeitos práticos da nossa simulação e da correção do EP, não será cobrado o sorteio uniforme dos pontos no espaço.

# 3 Entrega com cabeçalho e comentários

A entrega do EP deve ser feita pelo **paca**, como anteriormente. Desta vez, porém, todos os arquivos (pois você pode decidir fazer mais que um) devem ter um cabeçalho simples e comentários.

O cabeçalho é formado por duas linhas:

```
#.n <seu nome>
#.u <seu número usp>

Por exemplo:

#.n Roberto Hirata Jr.
#.u 1234567
```

# 4 Plágio

Plágio é a cópia/modificação não autorizada e/ou sem o conhecimento do autor original. O plágio é um problema grave que pode levar até a expulsão do aluno da universidade. Para quaisquer dúvidas, consulte o texto que disponibilizamos para uma disciplina irmã desta (`http://www.ime.usp.br/~mac2166/plagio/`).

# 5 Referências

[Jay73] E. T. Jaynes. "The Well-Posed Problem", Foundations of Physics 3: 477:493.

[Key21] John M. Keynes, “A Treatise on Probability”. Macmillan and Co., 1921.
[Knu98] D.E. Knuth, “Seminumerical algorithms”. Addison-Wesley, 1998.
[Por10] P. Di. Porto et al., “Bertrand’s paradox: a physical solution”, http://arxiv.org/abs/1008.1878
[Tis84] P.E. Tissler, ”Bertrand’s Paradox”. The Mathematical Gazette 68 (443): 15:19.