Caminhadas Aleatórias - Guilherme de Abreu Lima Buitrago Miranda - 2018054788

June 1, 2021

1 Introdução à Física Estatística Computacional

1.1 Caminhadas Aleatórias

Aluno: Guilherme de Abreu Lima Buitrago Miranda

Matrícula: 2018054788

1.1.1 Imports

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('seaborn-colorblind')
plt.ion()
```

1.1.2 Funções

Abaixo, define-se as funções de gerar uma caminhada aleatória unidimensional e bidimensional

[2]:
```
def generate_walk_uni(n):
    generated_steps = []
    generated_steps.append(np.random.rand() - 0.5)
    for i in range(1, n):
        generated_steps.append(np.random.rand() - 0.5 + generated_steps[i-1])

    return generated_steps
```

[3]:
```
def generate_walk_bi(n):
    generated_steps = []
    generated_steps.append([np.random.rand() - 0.5, np.random.rand() - 0.5])
    for i in range(1, n):
        generated_steps.append([np.random.rand() - 0.5 +␣
 ↪generated_steps[i-1][0],
                                np.random.rand() - 0.5 +␣
 ↪generated_steps[i-1][1]])
```

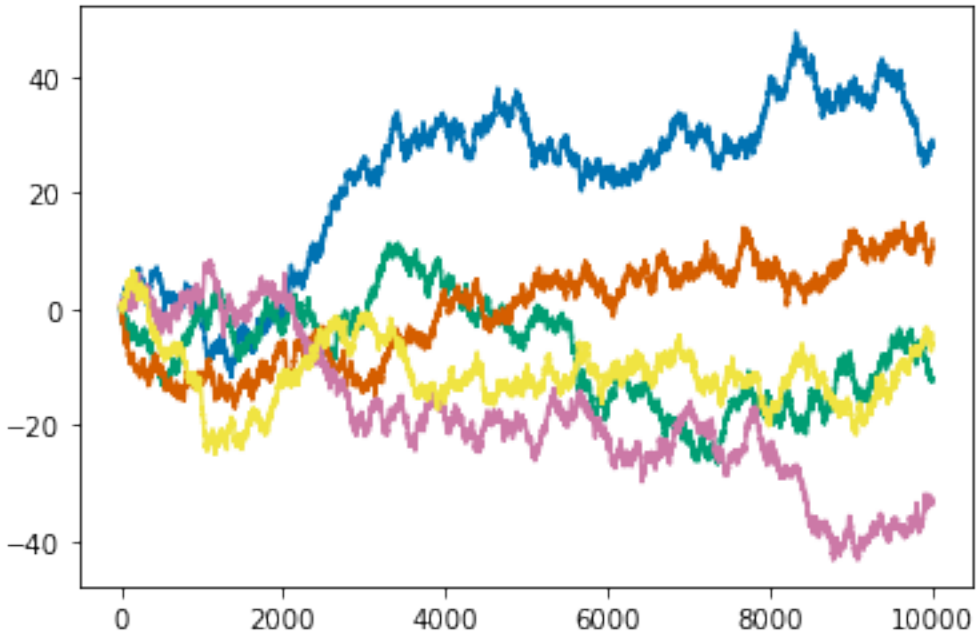
```
    return generated_steps
```

1.1.3 Item A

- Rotinas definidas acima;
- Gráfico de x_t por t para poucas caminhadas de 10.000 passos - próxima célula;
- Gráfico de x por y para poucas caminhadas aleatórias bidimensionais com N = 10, 1.000 e 100.000 - células subsequentes;

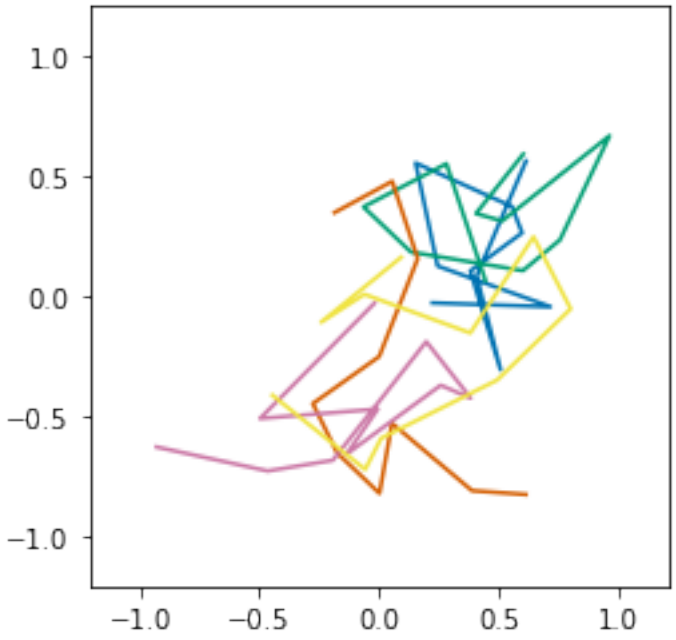
Se você multiplicar o número de passo por 100, a distância final da caminhada aumenta por cerca de 10 vezes? R: Sim. A distância final da caminhada aumenta por cerca de 10 vezes, ou seja, aproximadamente $\sqrt{100}$

[7]:
```
for i in range(5):
    plt.plot(generate_walk_uni(10000))
```

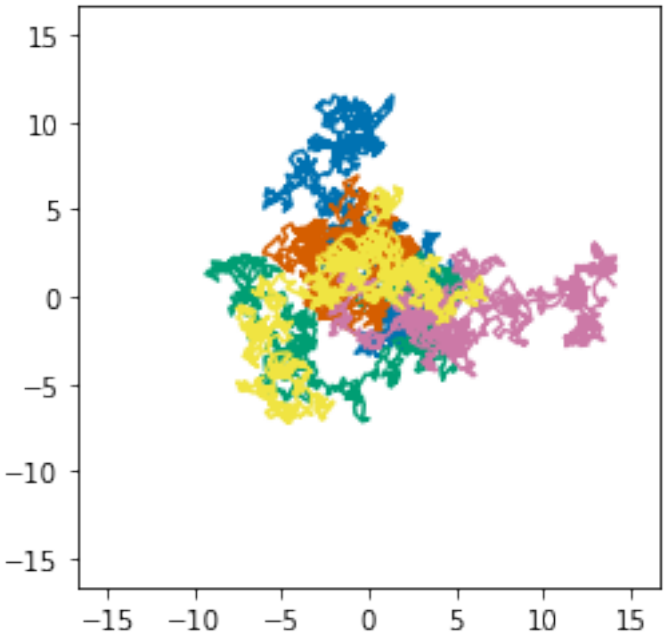


[8]:
```
ax_max = 0
for i in range(5):
    result = generate_walk_bi(10)
    x_val = [x[0] for x in result]
    y_val = [x[1] for x in result]
    x_max = max(x_val, key=abs)
    y_max = max(y_val, key=abs)
    max_result = max([x_max, y_max], key=abs)
    if abs(max_result) > ax_max:
        ax_max = abs(max_result)
```

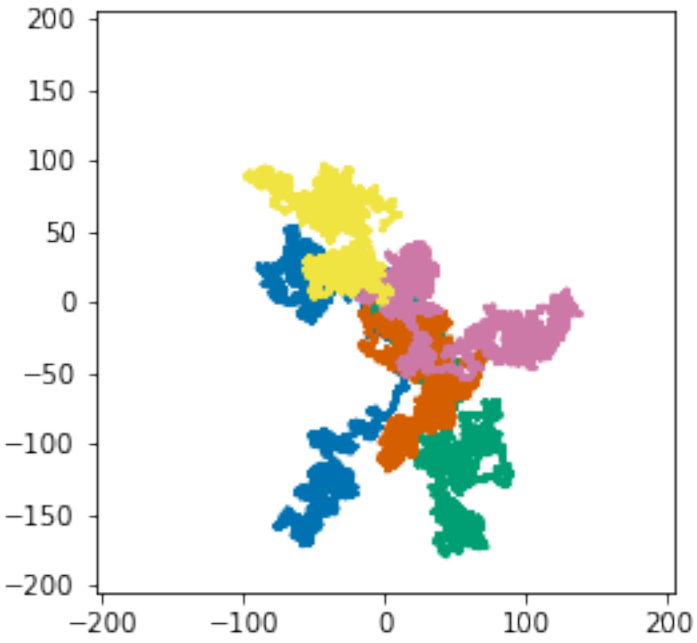
```
    plt.plot(x_val, y_val)

ax_max += 0.25
plt.xlim([-abs(ax_max), abs(ax_max)])
plt.ylim([-abs(ax_max), abs(ax_max)])
plt.gca().set_aspect('equal')
```



[10]:
```
ax_max = 0
for i in range(5):
    result = generate_walk_bi(1000)
    x_val = [x[0] for x in result]
    y_val = [x[1] for x in result]
    x_max = max(x_val, key=abs)
    y_max = max(y_val, key=abs)
    max_result = max([x_max, y_max], key=abs)
    if abs(max_result) > ax_max:
        ax_max = abs(max_result)
    plt.plot(x_val, y_val)

ax_max += 2.5
plt.xlim([-abs(ax_max), abs(ax_max)])
plt.ylim([-abs(ax_max), abs(ax_max)])
plt.gca().set_aspect('equal')
```



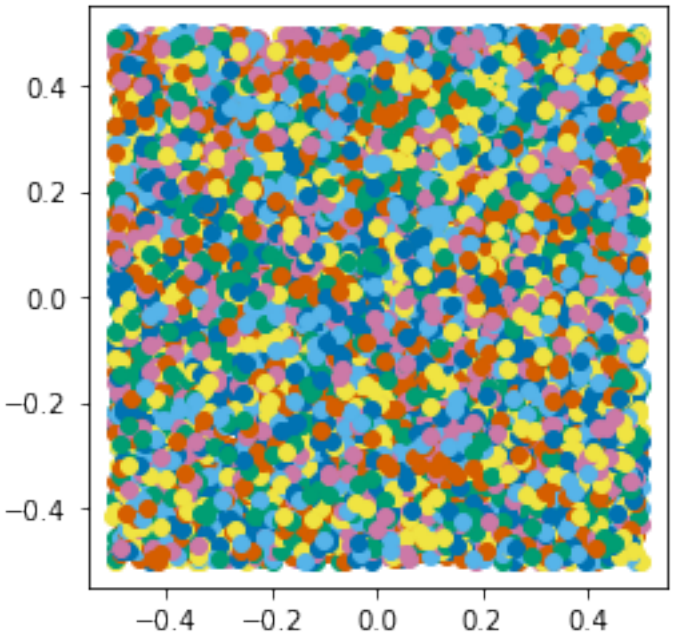
[18]:
```
ax_max = 0
for i in range(5):
    result = generate_walk_bi(100000)
    x_val = [x[0] for x in result]
    y_val = [x[1] for x in result]
    x_max = max(x_val, key=abs)
    y_max = max(y_val, key=abs)
    max_result = max([x_max, y_max], key=abs)
    if abs(max_result) > ax_max:
        ax_max = abs(max_result)
    plt.plot(x_val, y_val)

ax_max += 25
plt.xlim([-abs(ax_max), abs(ax_max)])
plt.ylim([-abs(ax_max), abs(ax_max)])
plt.gca().set_aspect('equal')
```



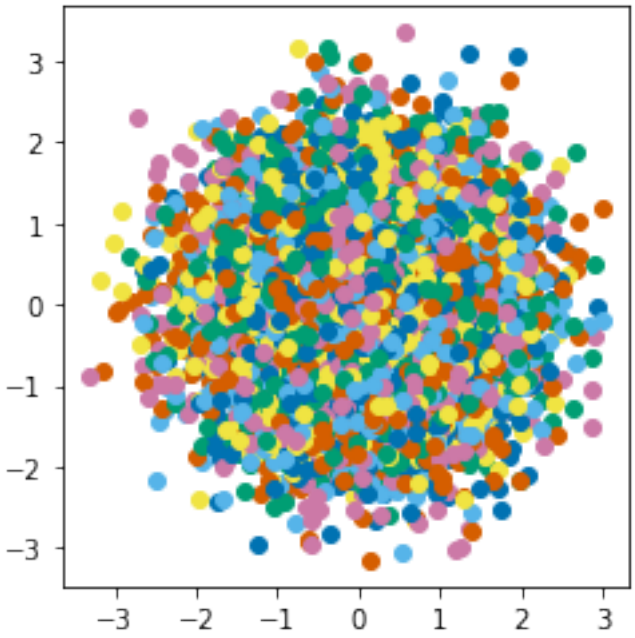
1.1.4 Item B

Abaixo, encontram-se os gráficos de dispersão dos pontos finais das 10000 caminhadas com 1 e 10 passos De fato, observa-se claramente o padrão quaderado nas caminhadas de 1 passo e o padrão simétrico e circular para a caminhada mais longa.

[19]:
```
for i in range(10000):
    result = generate_walk_bi(1)[0]
    x_val = result[0]
    y_val = result[1]
    plt.gca().set_aspect('equal')
    plt.scatter(x_val, y_val)
```



[20]:
```
for i in range(10000):
    result = generate_walk_bi(10)[9]
    x_val = result[0]
    y_val = result[1]
    plt.gca().set_aspect('equal')
    plt.scatter(x_val, y_val)
```

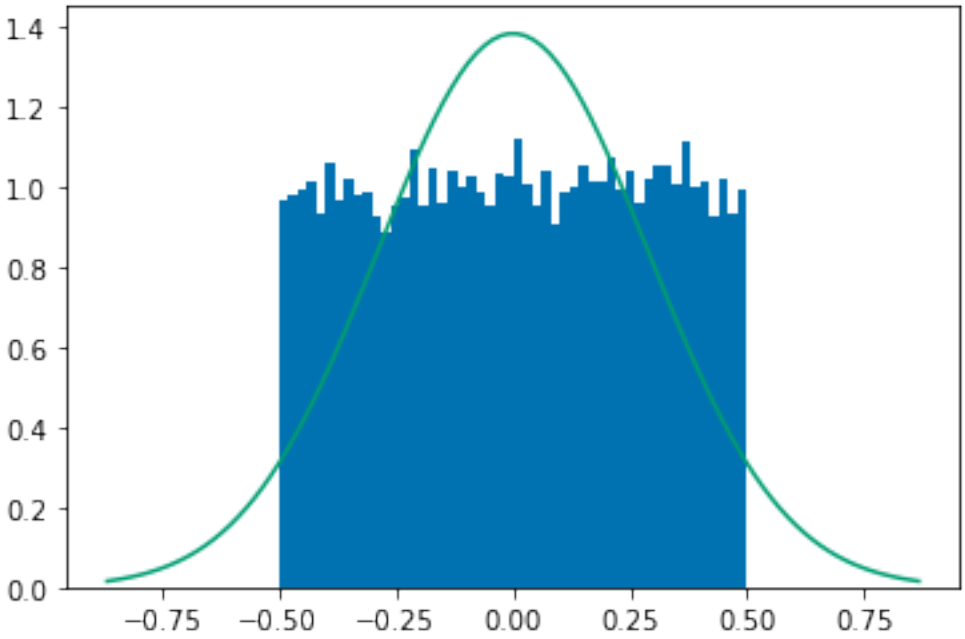


1.1.5 Item C

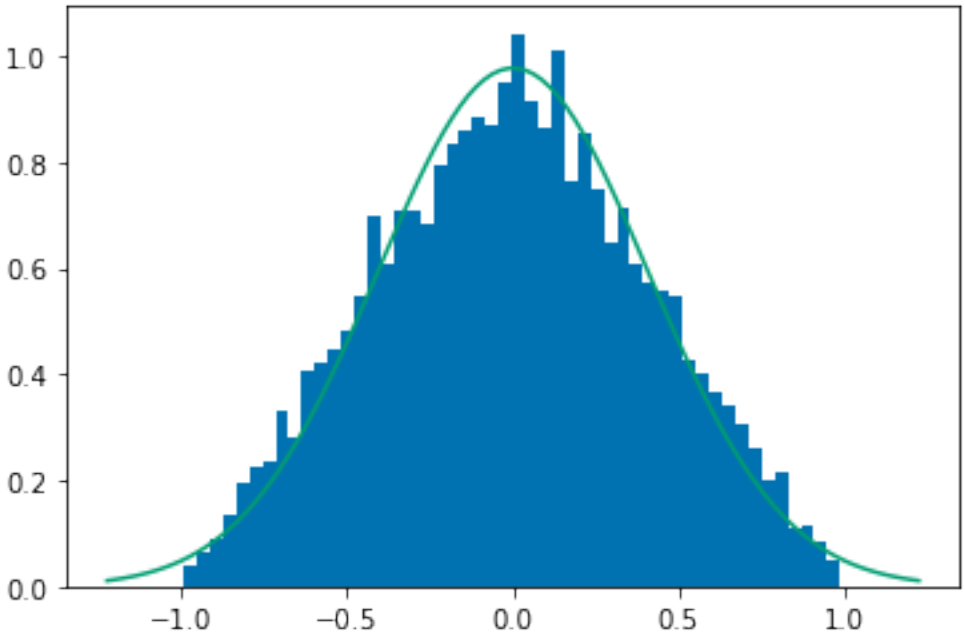
Abaixo, encontram-se o cálculo do desvio quadrático médio (RMS), assim como os histogramas dos pontos finais de W caminhadas aleatórias com N passos e 50 bins juntamente com a gaussiana. É notável que, conforme aumentamos o número de passos, rapidamente a gaussiana se torna uma boa aproximação para uma caminhada aleatória, sendo este padrão visível já a partir de 2 passos e ainda mais perceptível nos gráficos para 3 e para 5 passos.

[21]:
```
def plot_norm(steps):

    generated_steps = []
    for i in range(10000):
        generated_steps.append(generate_walk_uni(steps)[steps-1])

    a = -1/2
    b = 1/2
    varx = ((b - a) ** 2)/12

    sigma = np.sqrt(steps) * np.sqrt(varx)
    x = np.linspace(-3*sigma, 3*sigma, 100)
    y = (1/(np.sqrt(2 * 3.14159) * sigma)) * np.exp(-(x ** 2)/(2 * sigma ** 2))

    plt.hist(generated_steps, bins=50, density=True)
    plt.plot(x, y)
    plt.show()
```

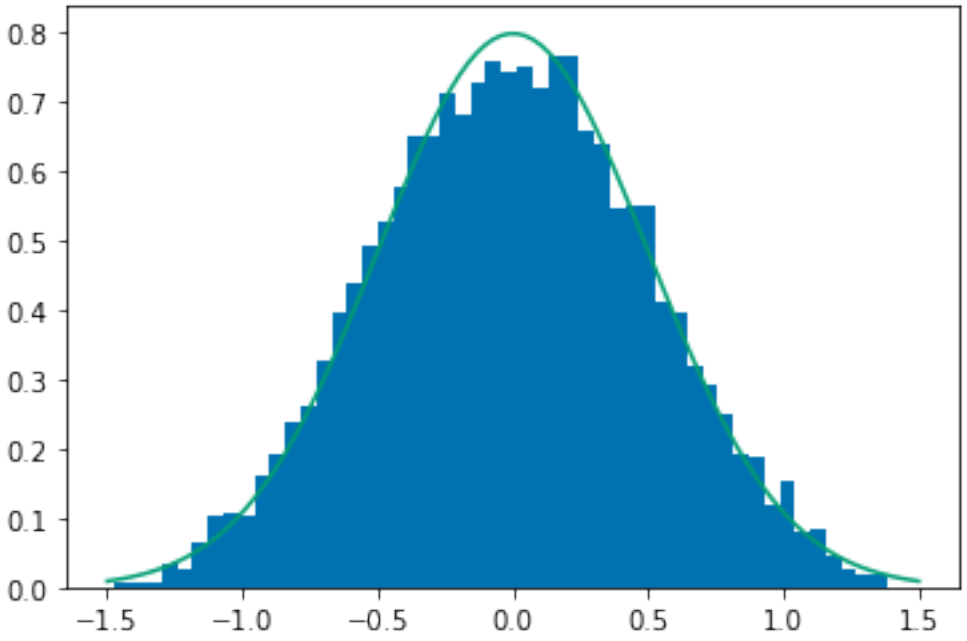
[22]:
```
plot_norm(1)
```



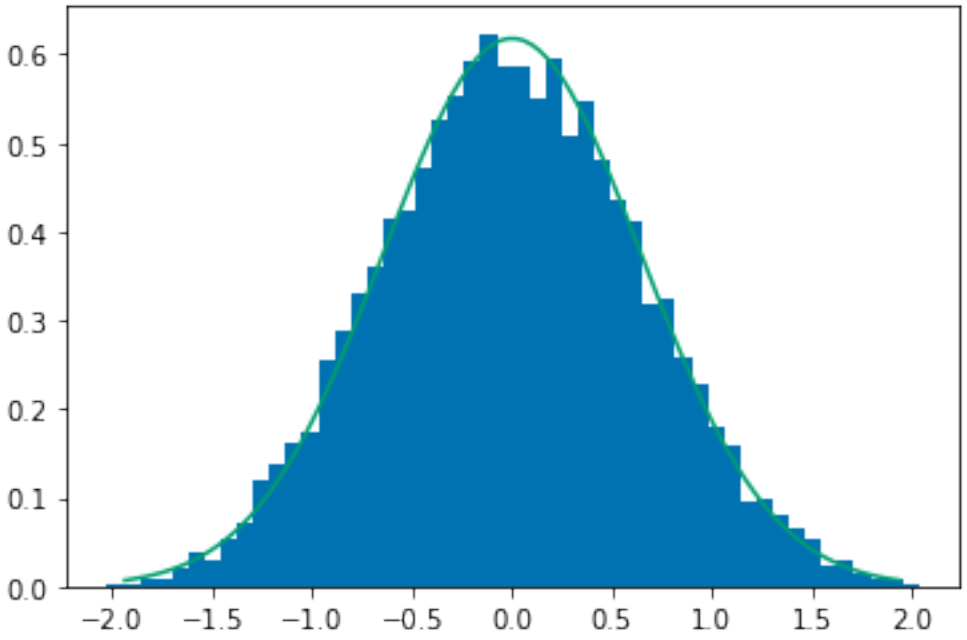
[23]:
```
plot_norm(2)
```



[24]:
```
plot_norm(3)
```



[25]:
```
plot_norm(5)
```



[ ]:
```

```