Bolas e Urnas

October 9, 2017

```
In [1]: import numpy as np
        import pandas as pd
        import random
        import math
        import scipy.stats  as stats
        import matplotlib.pyplot as plt
```

0.0.1 Introdução

Considere uma urna com t bolas, sendo g verdes e r vermelhas, com g + r = t. Quando uma bola é retirada com reposição, significa que esta bola volta para a urna antes da próxima ser retirada. Quando uma bola é retirada sem reposição, significa que esta bola não volta para a urna antes da próxima ser retirada.
Sejam os seguintes experimentos probabilísticos, em que n e k são parâmetros dos experimentos.

- Experimento 1a. Retira-se, uma por uma, um total de n bolas da urna, com reposição.
- Experimento 1b. Retira-se, uma por uma, um total de n bolas da urna, sem reposição.
- Experimento 2a. Retira-se, uma por uma, bolas da urna, com reposição. O experimento se encerra imediatamente após a k-ésima bola verde é retirada.
- Experimento 2b. Retira-se, uma por uma, bolas da urna, sem reposição. O experimento se encerra imediatamente após a k-ésima bola verde é retirada

Defina as seguintes variáveis aleatórias:
* Experimentos 1a e 1b. K = número de bolas verdes retiradas no experimento. * Experimentos 2a e 2b. N = número total de bolas retiradas no experimento.
Tais variáveis aleatórias estão relacionadas com as seguintes distribuições discretas: * Experimentos 1a. Distribuição binomial. * Experimentos 1b. Distribuição hipergeométrica. * Experimentos 2a. Distribuição binomial negativa (ou de Pascal). * Experimentos 2b. Distribuição hipergeométrica negativa.

1 - Escreva uma função que implementa uma única realização da variável aleatória K do Experimento 1a, para (t, g, n) genéricos. Em seguida, utilize as funções escritas com (t, g, n) = (8, 4, 6) e com (t, g, n) = (100, 20, 95).
* Plote figuras contendo a função massa de probabilidade teórica, bem como aquela obtida via simulação de Monte Carlo. * Calcule os valores teóricos da média e da variância, bem como aqueles obtidos via simulação.

Dados:

n = numero de bolas retiradas da urna
t = número de bolas

g = bolas verdes
r = bolas vermelhas
t = g+r
k = possivel resultado da variavel aleatoria
K = número de bolas verdes retiradas no experimento.

$$p = \frac{g}{t} \quad q = \frac{r}{t} \qquad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

PMF 1a -

$$p_K(k) = \binom{n}{k} p^k q^{n-k}$$

PMF 1b -

$$p_K(k) = \frac{\binom{g}{k}}{\binom{t}{n}} \binom{r}{l}$$

0.1 Formulas

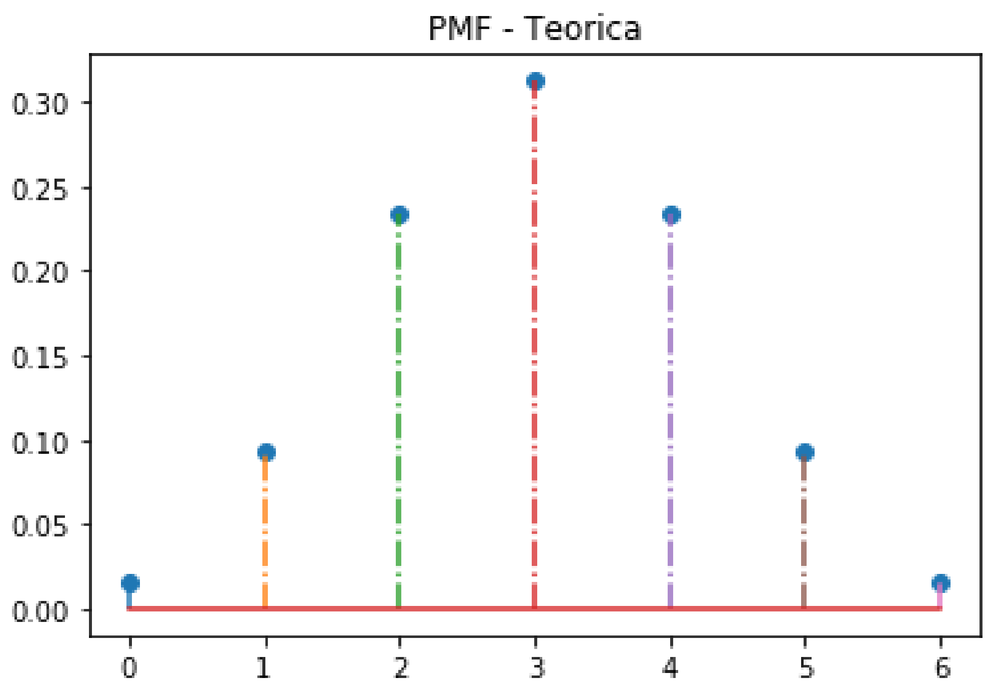
In [21]:
```
def coef_binon(n,k):

    if k > n:
        return 0
    return math.factorial(n)/(math.factorial(k)*math.factorial(n-k))

def prob(g,t):
    return float(g)/t

def pmf_1a(p,n,k):
    pmf = coef_binon(n,k)*(p**k)*(1-p)**(n-k)
    return pmf

def pmf_1b(g,r,t,n,k):
    l = n-k

    pmf = (coef_binon(g,k)/coef_binon(t,n))*coef_binon(r,l)
    return pmf

def media_1(n,p):
    return n*p

def var_1a(n,p,q):
    return n*p*q

def var_1b(n,p,q,t):
    return (n*p*q)*(t-n/t-1)

def exp_1a(t,g,n):
     p = prob(g,t)
```

```
    i = 0
    outcomes = []
    while i < n:
        a = random.uniform(0, 1)
        outcomes.append(a <= p)
        i += 1
    return np.sum(outcomes)

def exp_1b(t,g,n):
    i = 0
    outcomes = []
    while i < n:
        a = random.uniform(0, 1)
        p = prob(g,t)
        outcomes.append(a <= p)
        t -= 1
        i += 1
    return np.sum(outcomes)
```

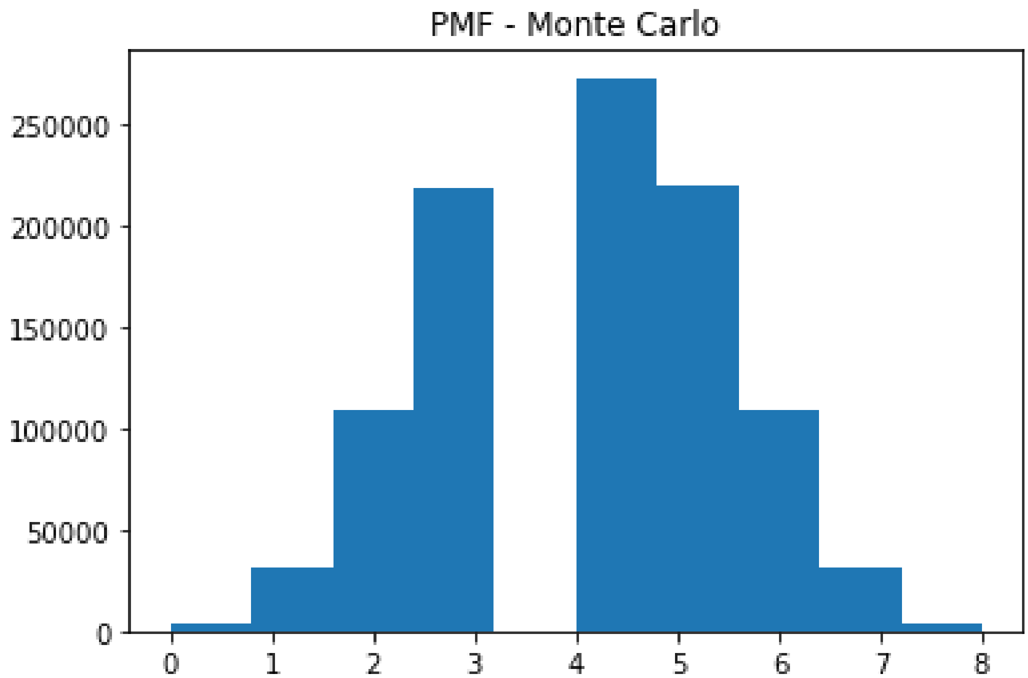
```
In [37]: #(t, g, n)
         t, g, n = 8, 4, 6
         X_a = exp_1a(t, g, n)
         print("Qutd. de bolas verdes retiradas: {}".format(X_a))
         pmf_vet = []
         p = prob(g,t)
         q = 1-p
         for i in range(n+1):
             pmf_vet.append(pmf_1a(p,n,i))

         plt.stem(range(n+1),pmf_vet,'-.')
         plt.title('PMF - Teorica')
         plt.show()
         media_t = media_1(n,p)
         var_t = var_1a(n,p,q)
         print("---Valores teoricos---")
         print("Media: {} Variancia: {}".format(media_t,var_t))

         #Monte Carlo
         n=1000
         pmf_vet = []
         for i in range(1000000):
             pmf_vet.append(exp_1a(t, g, t))

         plt.hist(pmf_vet)
```

```
plt.title('PMF - Monte Carlo')
plt.show()
media_t = media_1(n,p)
var_t = var_1a(n,p,q)
print("---Valores Monte Carlo---")
print("Media: {} Variancia: {}".format(media_t,var_t))
```

```
Qutd. de bolas verdes retiradas: 4
```




```
---Valores teoricos---
Media: 3.0 Variancia: 1.5
```



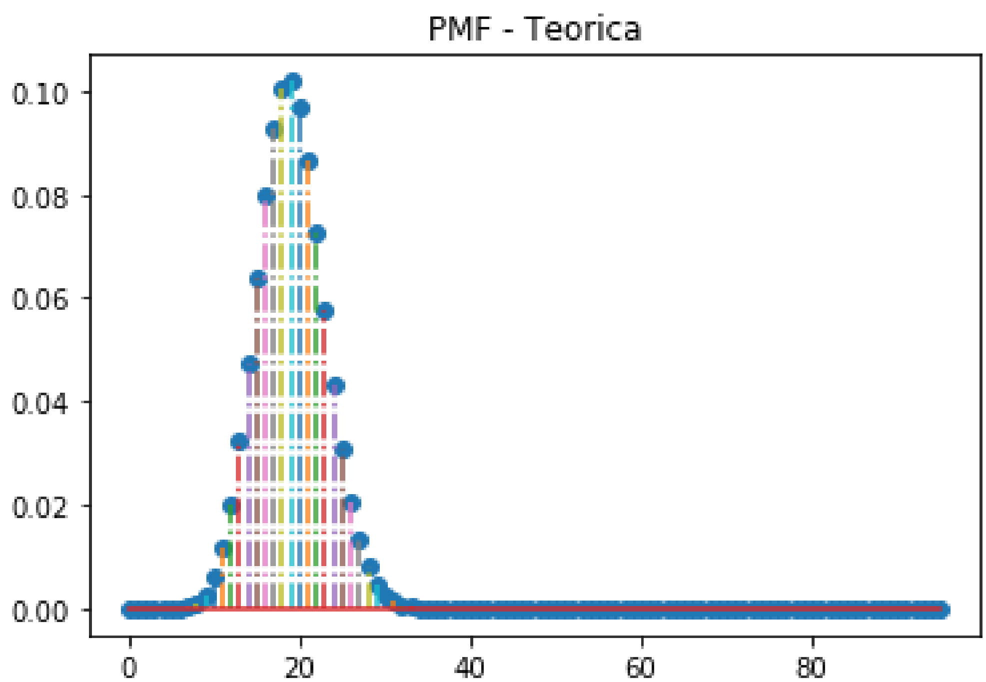

```
---Valores Monte Carlo---
Media: 500.0 Variancia: 250.0
```

0.1.1 Para (t, g, n) = (100, 20, 95)

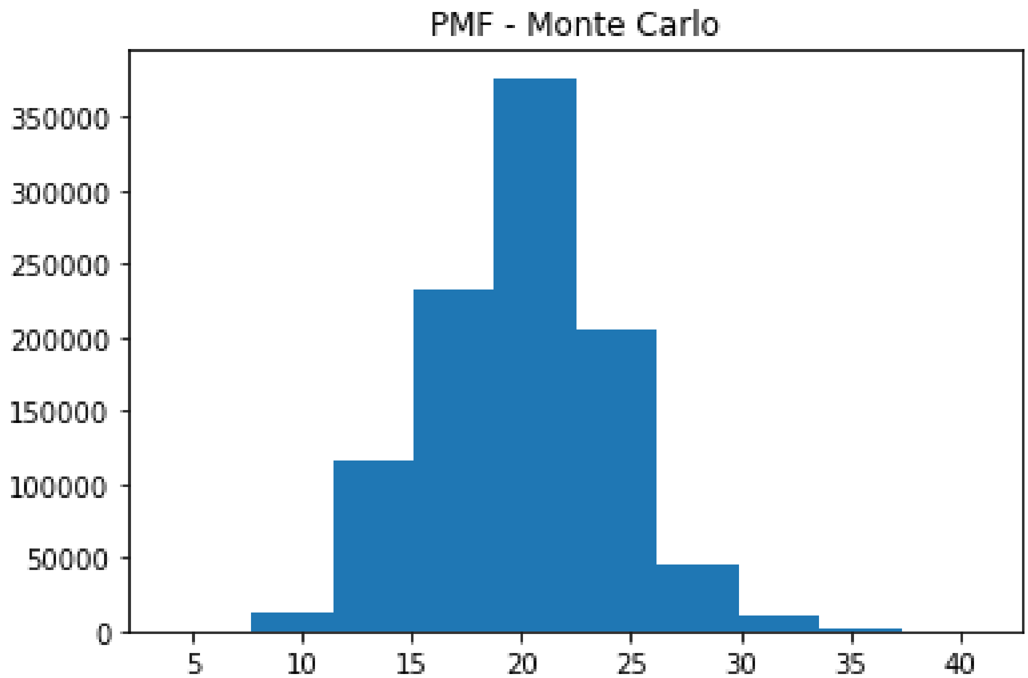
```
In [38]: #(t, g, n)
         t, g, n = 100, 20, 95
         X_b = exp_1a(t, g, n)
         print("Qutd. de bolas verdes retiradas: {}".format(X_b))
         pmf_vet = []
         p = prob(g,t)
         q = 1-p
         for i in range(n+1):
             pmf_vet.append(pmf_1a(p,n,i))

         plt.stem(range(n+1),pmf_vet,'-.')
         plt.title('PMF - Teorica')
         plt.show()
         media_t = media_1(n,p)
         var_t = var_1a(n,p,q)
         print("---Valores teoricos---")
         print("Media: {} Variancia: {}".format(media_t,var_t))

         #Monte Carlo
```

```python
n=1000
pmf_vet = []
for i in range(1000000):
    pmf_vet.append(exp_1a(t, g, t))

plt.hist(pmf_vet)
plt.title('PMF - Monte Carlo')
plt.show()
media_t = media_1(n,p)
var_t = var_1a(n,p,q)
print("---Valores Monte Carlo---")
print("Media: {} Variancia: {}".format(media_t,var_t))
```

```
Qutd. de bolas verdes retiradas: 17
```




```
---Valores teoricos---
Media: 19.0 Variancia: 15.200000000000001
```




```
---Valores Monte Carlo---
Media: 200.0 Variancia: 160.0
```

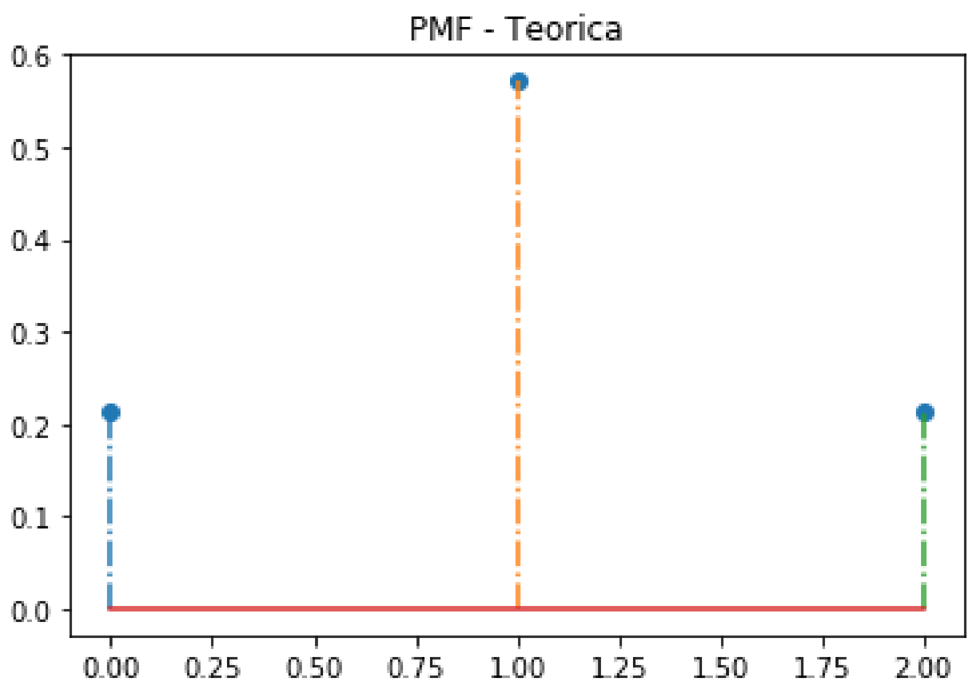
0.2 2 Repita o exercício anterior, agora considerando o Experimento 1b.

0.2.1 Para (t, g, n) = (8, 4, 6)

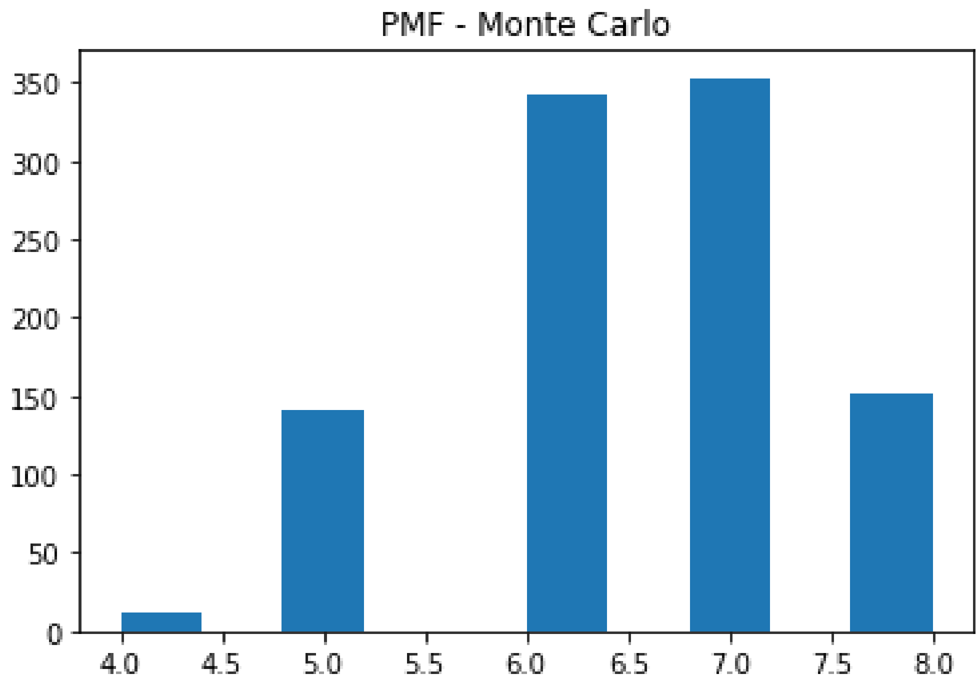
```
In [3]: #pmf_1b(g,r,t,n,k)
        g, r, t, n = 4, 4, 8, 6
        X_a = exp_1b(t, g, n)
        print("Qutd. de bolas verdes retiradas: {}".format(X_a))
        pmf_vet = []
        p = prob(g,t)
        q = 1-p
        for i in range(n+1):
            va = pmf_1b(g,r,t,n,i)
            if va > 0:
                pmf_vet.append(va)
            ##print("pmf: {}".format(pmf_vet[i]))

        plt.stem(range(len(pmf_vet)),pmf_vet,'-.')
        plt.title('PMF - Teorica')
        plt.show()
        media_t = media_1(n,p)
```

```
var_t = var_1b(n,p,q,t)
print("---Valores teoricos---")
print("Media: {} Variancia: {}".format(media_t,var_t))

#Monte Carlo
n=1000
pmf_vet = []
for i in range(1000):
    pmf_vet.append(exp_1b(t, g, t))


plt.hist(pmf_vet,histtype='bar')
plt.title('PMF - Monte Carlo')
plt.show()
media_t = media_1(n,p)
var_t = var_1b(n,p,q,t)
print("---Valores Monte Carlo---")
print("Media: {} Variancia: {}".format(media_t,var_t))
```

```
Qutd. de bolas verdes retiradas: 5
```




```
---Valores teoricos---
Media: 3.0 Variancia: 9.375
```



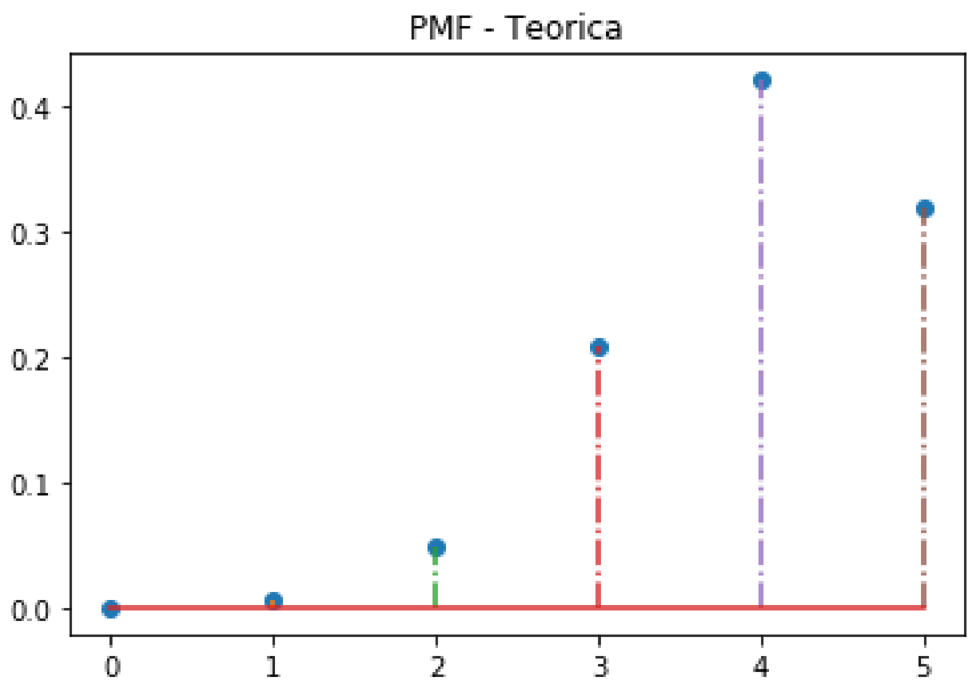

```
---Valores Monte Carlo---
Media: 500.0 Variancia: -29500.0
```

0.3 Para (t, g, n) = (100, 20, 95)

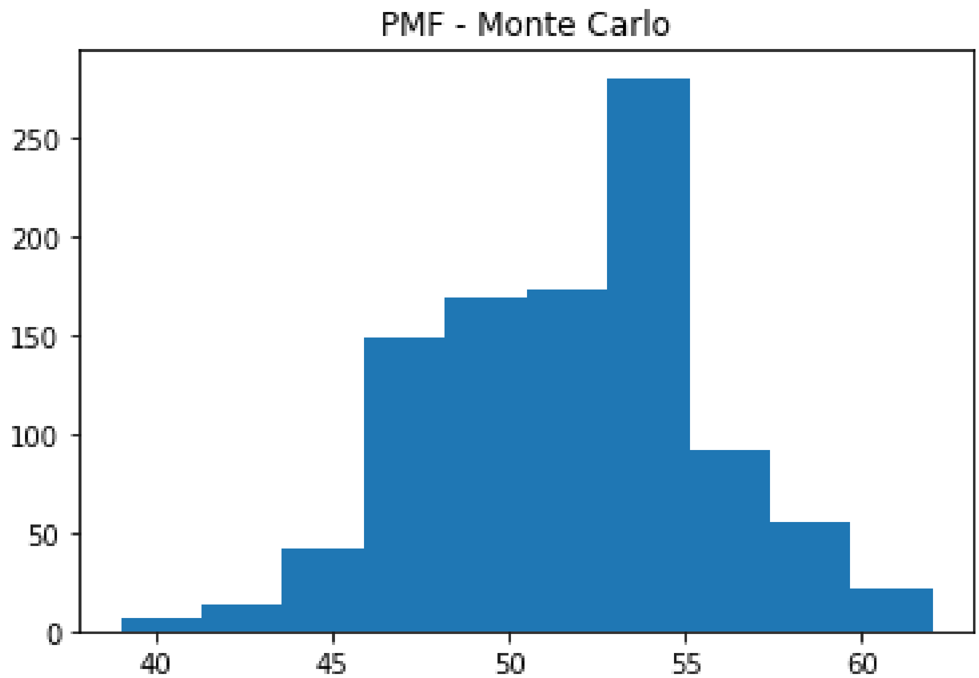
```
In [4]: #pmf_1b(g,r,t,n,k)
        g, r, t, n = 20, 80, 100, 95
        X_a = exp_1b(t, g, n)
        print("Qutd. de bolas verdes retiradas: {}".format(X_a))
        pmf_vet = []
        p = prob(g,t)
        q = 1-p
        i=0
        for i in range(n+1):
            va = pmf_1b(g,r,t,n,i)
            if va > 0:
                pmf_vet.append(va)
            ##print("pmf: {}".format(pmf_vet[i]))

        plt.stem(range(len(pmf_vet)),pmf_vet,'-.')
        plt.title('PMF - Teorica')
        plt.show()
        media_t = media_1(n,p)
        var_t = var_1b(n,p,q,t)
```

```
print("---Valores teoricos---")
print("Media: {} Variancia: {}".format(media_t,var_t))

#Monte Carlo
n=1000
pmf_vet = []
for i in range(n+1):
    pmf_vet.append(exp_1b(t, g, t))


plt.hist(pmf_vet,histtype='bar')
plt.title('PMF - Monte Carlo')
plt.show()
media_t = media_1(n,p)
var_t = var_1b(n,p,q,t)
print("---Valores Monte Carlo---")
print("Media: {} Variancia: {}".format(media_t,var_t))
```

```
Qutd. de bolas verdes retiradas: 48
```




```
---Valores teoricos---
Media: 19.0 Variancia: 1490.3600000000001
```




```
---Valores Monte Carlo---
Media: 200.0 Variancia: 14240.0
```

0.4 Experimento 2 a/b

- Experimento 2a. Retira-se, uma por uma, bolas da urna, com reposição. O experimento se encerra imediatamente após a k-ésima bola verde é retirada.
- Experimento 2b. Retira-se, uma por uma, bolas da urna, sem reposição. O experimento se encerra imediatamente após a k-ésima bola verde é retirada

VA's: * Experimentos 2a e 2b. N = número total de bolas retiradas no experimento.

PMF's: * Experimentos 2a. Distribuição binomial negativa (ou de Pascal). * Experimentos 2b. Distribuição hipergeométrica negativa.

Formulas:

$$p = \frac{g}{t} \quad q = \frac{r}{t} \qquad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

PMF 2a -

$$p_K(k) = \binom{n-1}{k-1} p^k q^{n-k}$$

PMF 2b -

$$p_K(k) = \frac{\binom{g}{k-1}}{\binom{t}{n-1}} \frac{g-k+1}{t-n+1} \binom{r}{l}$$

Media 2a -

$$E[N] = \frac{k}{p}$$

Media 2b -

$$E[N] = k\frac{t+1}{g+1}$$

Variância 2a -

$$var[N] = k\frac{q}{p^2}$$

Variância 2b -

$$var[N] = k\frac{(t-g)(t+1)(g+1-k)}{(g+1)^2(g+2)}$$

In [12]:
```
def pmf_2a(n,k):
    return coef_binon(n-1,k-1)*(p**k)*(1-p)**(n-k)

def pmf_2b(t,g,n,k):
    return (coef_binon(g,k-1)/coef_binon(t,n-1))*((g-k+1)/(t-n+1))*coef_binon(t-n,n-k

def media_2a(p,k):
    return k/p

def media_2b(t,g,k):
    return k*(t+1/g+1)

def var_2a(p,k):
    q = 1-p
    return k*(q/p**2)

def var_2b(t,g,k):
    return k*(((t-g)*(t+1)*(g+1-k))/(((g+1)**2)*(g+2)))

def exp_2a(t,g,n):
     p = prob(g,t)
     i = 1
     while i < n+1:
            a = random.uniform(0, 1)
            if a <= p:
                return i
            i += 1
     return 0

def exp_2b(t,g,n):
     i = 0
     outcomes = []
     while i < n:
            a = random.uniform(0, 1)
            p = prob(g,t)
```

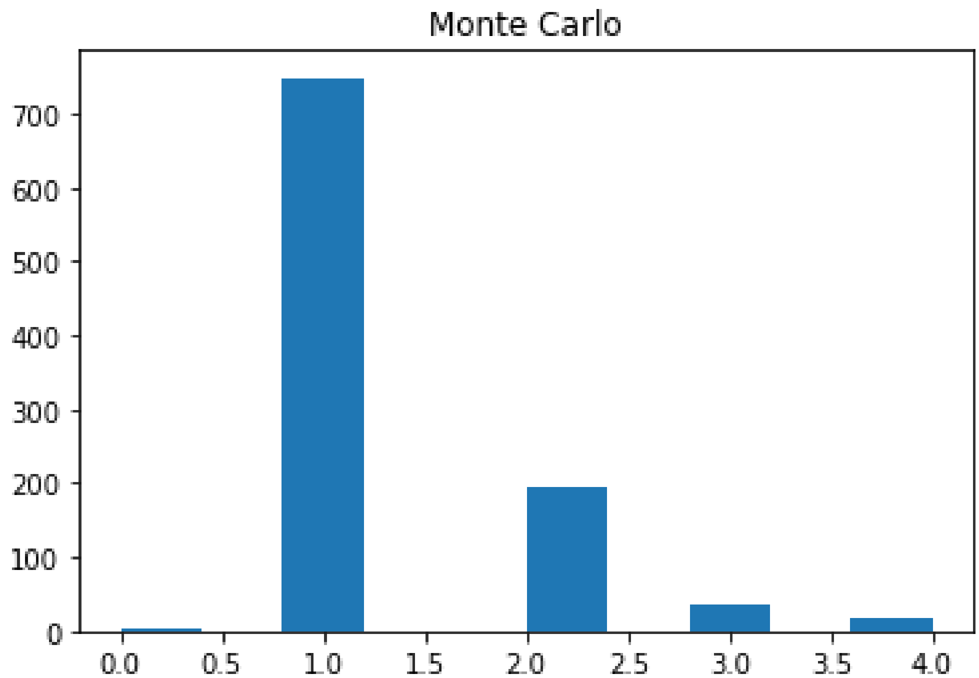
```
            outcomes.append(a < p)
            t -= 1
            i += 1
        return np.sum(outcomes)
```

3 - Escreva uma função que implementa uma única realização da variável aleatória N do Experimento 2a, para (t, g, k) genéricos. Em seguida, utilize as funções escritas com (t, g, k) = (8, 6, 4) e com (t, g, k) = (100, 20, 20).

- Plote figuras contendo a função massa de probabilidade teórica, bem como aquela obtida via simulação de Monte Carlo.
- Calcule os valores teóricos da média e da variância, bem como aqueles obtidos via simulação.

0.4.1 Para (t, g, k) = (8, 6, 4)

```
In [23]: t, g, k = 8, 6, 4
         outcomes_m = []
         for i in range(1000):
             outcomes_m.append(exp_2a(t,g,k))

         plt.hist(outcomes_m)
         plt.title('Monte Carlo')
         plt.show()

         outcomes_t = []
         for i in range(t):
             outcomes_t.append(pmf_2a(t,i))

         outcomes_t
```




```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-23-b91d072d7e68> in <module>()
     10 outcomes_t = []
     11 for i in range(t):
---> 12     outcomes_t.append(pmf_2a(t,i))
     13 
     14 outcomes_t


<ipython-input-12-4ea45a139b3e> in pmf_2a(n, k)
      1 def pmf_2a(n,k):
----> 2     return coef_binon(n-1,k-1)*(p**k)*(1-p)**(n-k)
      3 
      4 def pmf_2b(t,g,n,k):
      5     return (coef_binon(g,k-1)/coef_binon(t,n-1))*((g-k+1)/(t-n+1))*coef_binon(t-n,r


<ipython-input-21-e83685ef53d7> in coef_binon(n, k)
      3     if k > n:
      4         return 0
----> 5     return math.factorial(n)/(math.factorial(k)*math.factorial(n-k))
```

```

def prob(g,t):


ValueError: factorial() not defined for negative values
```

In [ ]: