Code Throwing Darts and the Poisson Distribution

Valentina Watanabe
21367661

October 2024

1 Part 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  5 09:53:47 2024

@author: vwitch
"""
import numpy as np
import math
import matplotlib.pyplot as plt
from tabulate import tabulate

n = range(0,51,1)

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

### For <n> = 1
Px = []
E = []
V = []

for i in n:
    p = P(1,i)
    e = i*p
    v = i**2 *p
    Px.append(p)
    E.append(e)
    V.append(v)
```

```
plt.bar(n, Px, color="pink", label="Simulated $P_{sim}(n)$", alpha=0.7,
edgecolor="black")
#plt.plot(n, Px, marker='o', color = 'pink')
plt.xlabel('n')
plt.ylabel('P(X=n)')
plt.title('Poisson Distribution with <n> = 1')
plt.grid(True)
plt.show()

sum1 = sum(Px)
#print('Sum of P(n) for <n> = 1, ' + str(sum1))
sume = sum(E)
#print('Sum of n*P(n) for <n> = 1, ' + str(sume))
sumv = sum(V)
#print('Sum of n^2 *P(n) for <n> = 1, ' + str(sumv))
Var = sumv - sume**2
stdv = np.sqrt(Var)

### For <n> = 5
Px2 = []
E2 = []
V2 = []

for i in n:
    p = P(5,i)
    e=i*p
    v = i**2 *p
    Px2.append(p)
    E2.append(e)
    V2.append(v)

plt.bar(n, Px2, color="palevioletred", label="Simulated $P_{sim}(n)$",
alpha=0.7, edgecolor="black")
#plt.plot(n, Px2, marker='o', color = 'palevioletred')
plt.xlabel('n')
plt.ylabel('P(X=n)')
plt.title('Poisson Distribution with <n> = 5, ')
plt.grid(True)
plt.show()

sum2 = sum(Px2)
#print('Sum of P(n) for <n> = 5, ' + str(sum2))
sume2 = sum(E2)
#print('Sum of n*P(n) for <n> = 1, ' + str(sume2))
sumv2 = sum(V2)
```

```
#print('Sum of n^2 *P(n) for <n> = 1, ' + str(sumv2))
Var2 = sumv2 - sume2**2
stdv2 = np.sqrt(Var2)

### For <n> = 10
Px3 = []
E3 = []
V3 = []

for i in n:
    p = P(10,i)
    e = i*p
    v = i**2 *p
    Px3.append(p)
    E3.append(e)
    V3.append(v)

plt.bar(n, Px3, color="mediumvioletred", label="Simulated $P_{sim}(n)$",
alpha=0.7, edgecolor="black")
#plt.plot(n, Px3, marker='o', color = 'mediumvioletred')
plt.xlabel('n')
plt.ylabel('P(X=n)')
plt.title('Poisson Distribution with <n> = 10')
plt.grid(True)
plt.show()

sum3 = sum(Px3)
#print('Sum of P(n) for <n> = 10, ' + str(sum3))
sume3 = sum(E3)
#print('Sum of n*P(n) for <n> = 1, ' + str(sume3))
sumv3 = sum(V3)
#print('Sum of n^2 *P(n) for <n> = 1, ' + str(sumv3))
Var3 = sumv3 - sume3**2
stdv3 = np.sqrt(Var3)

table = [
    ["Mean", "Sum P(n)", "Sum n*P(n)", "Sum n^2*P(n)", "Variance", "Std Dev"],
    [1, sum1, sume, sumv, Var, stdv3],
    [5, sum2, sume2, sumv2, Var2, stdv2],
    [10, sum3, sume3, sumv3, Var3, stdv3]
]

# Display table
print(tabulate(table, headers="firstrow", tablefmt="grid"))
```

2 Part 2

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  5 15:52:56 2024

@author: vwitch
"""

import numpy as np
import math
import matplotlib.pyplot as plt

N = 50 #Number of darts
L = 100 #Regions
T = 10 #Trials

H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L
```

```
def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)


# Plot the results
plt.bar(Nvalues, NormP, color="skyblue", label="Simulated $P_{sim}(n)$",
alpha=0.7, edgecolor="black")
#plt.plot(Nvalues, NormP, label="Simulated $P_{sim}(n)$", marker='o',  color =
'skyblue')
plt.plot(Nvalues, P_poisson, label="Poisson Distribution", linestyle='--',color
= 'mediumvioletred')
plt.xlabel("Number of Darts in a Region (n)")
plt.ylabel("Probability")
plt.title("Comparison of $P_{sim}(n)$ and Poisson Distribution")
plt.legend()
plt.grid(True)
plt.show()

# Plot the results with log
plt.bar(Nvalues, NormP, color="skyblue", label="Simulated $P_{sim}(n)$",
alpha=0.7, edgecolor="black")
#plt.plot(Nvalues, NormP, label="Simulated $P_{sim}(n)$", marker='o',  color =
'skyblue')
plt.plot(Nvalues, P_poisson, label="Poisson Distribution", linestyle='--',color
= 'mediumvioletred')
plt.xlabel("Number of Darts in a Region (n)")
plt.yscale("log")
plt.ylabel("Probability")
plt.title("Comparison of $P_{sim}(n)$ and Poisson Distribution")
plt.legend()
plt.grid(True)
plt.show()
```

3 Part 3

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Nov  6 14:25:11 2024
```

```
@author: vwitch
"""

import numpy as np
import math


N = 50 #Number of darts
L = 100 #Regions
T = 10 #Trials

H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)
```

```
#FOR 100

T = 100 #Trials
H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)

min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")

### FOR 1000


T = 1000 #Trials
```

```
H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)

min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")


### FOR 10000


T = 10000 #Trials
H = np.zeros(N+1)

def onetrial(N,L):
```

```
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)

min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")
```

4 Part 4

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Nov  6 21:27:09 2024

@author: vwitch
"""
```

```
import numpy as np
import math


N = 50 #Number of darts
L = 5 #Regions
T = 10 #Trials

H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)


min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")
```

```
### FOR 1000


T = 1000 #Trials
H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)

min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")


### FOR 10000
```

```
T = 10000 #Trials
H = np.zeros(N+1)

def onetrial(N,L):
    #Array with each region
    B = np.zeros(L, dtype = int)
    for i in range(N):
        region = np.random.randint(0, L)
        #goes through all the darts and sums the ones in eahc region
        B[region] = B[region] + 1
    #count number of regions with each dart count (0 to N)
    H1 = np.zeros(N + 1)
    for darts in B:
        if darts <= N:
            H1[darts] += 1
    return H1

#Now do the same for first trial to T trials

for darts in range(T):
    H1 = onetrial(N,L)
    H += H1

# normalizw
NormP = H / (L * T)

# mean darts
mean = N / L

def P(mean,n):
    p = ((mean**n)/math.factorial(n))*np.exp(-mean)
    return p

P_poisson = [P(mean, n) for n in range(N + 1)]
Nvalues = range(N + 1)

min_nonzero_P = np.min(NormP[NormP > 0])
print(f"For T = {T}, the smallest non-zero P(n) observed is: {min_nonzero_P:.5e}")
```